# 通知 实验安排

数字逻辑电路课程有16个学时的实验，初步安排如下：

计算机1班 第8,10周 周三上午 1-4 (8:00-11:40)
计算机2班 第9,11周 周三上午 1-4 (8:00-11:40)
计算机3班 第9-12周 周二的下午5-6节
计算机4班 第10-13周 周一晚上 9-10 (19:00-20:50)
计算机5班 第9-12周 周三晚上9-10 (19:00-20:50)
物联网1班 第8,10周 周三下午5-8 (14:00-17:40)
物联网2班 第9,11周 周三下午5-8 (14:00-17:40)

计算机1班 第13，14周 周二上午4节 (8:00-11:40)
计算机2班 第13，14周 周二下午4节 (14:00-17:40)
计算机3班 第13，14周 周二晚上4节 (18:30-22:10)
计算机4班 第13，14周 周四上午4节 (8:00-11:40)
计算机5班 第13，14周 周四下午4节 (14:00-17:40)
物联网1班 第13，14周 周三下午4节 (14:00-17:40)
物联网2班 第13，14周 周三晚上4节(18:30-22:10)

实验设备台套数有限，所以一次只能安排一个班实验。

实验地点：综合实验楼305房间

# 数 字 逻 辑

丁 贤 庆

ahhfdxq@163.com

# 第 四 章

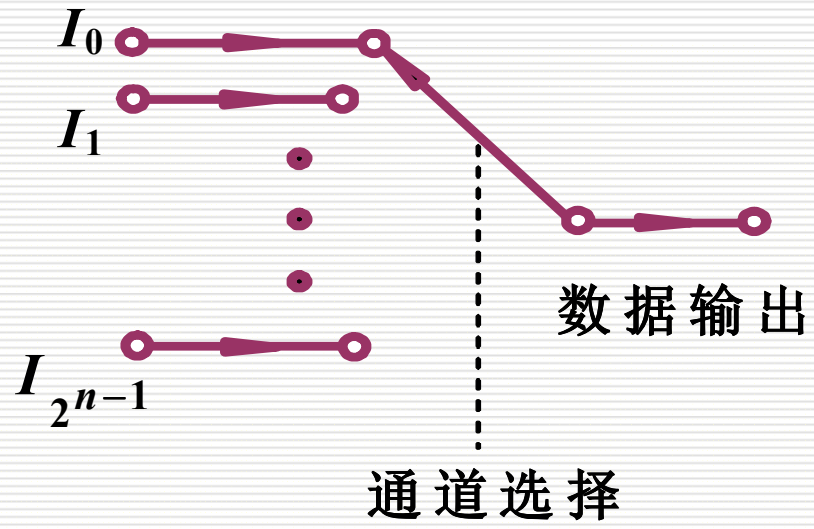
## 组合逻辑电路

# 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”。

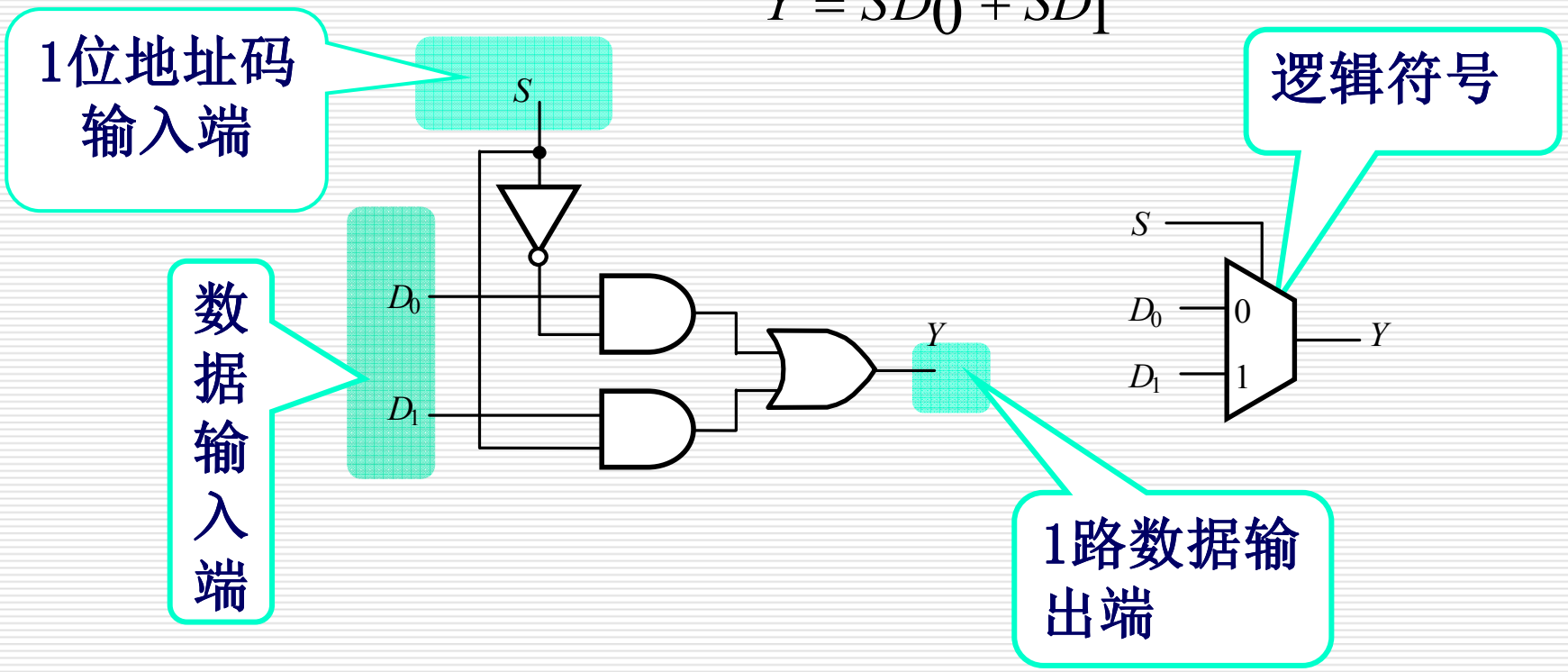
数据选择的功能：在通道选择信号的作用下，将多个通道的数据分时传送到公共的数据通道上去的。



# 2选1数据选择器

| 输入 | 输 出 |
|---|---|
| $S$ | $Y$ |
| **0** | $D_0$ |
| **1** | $D_1$ |

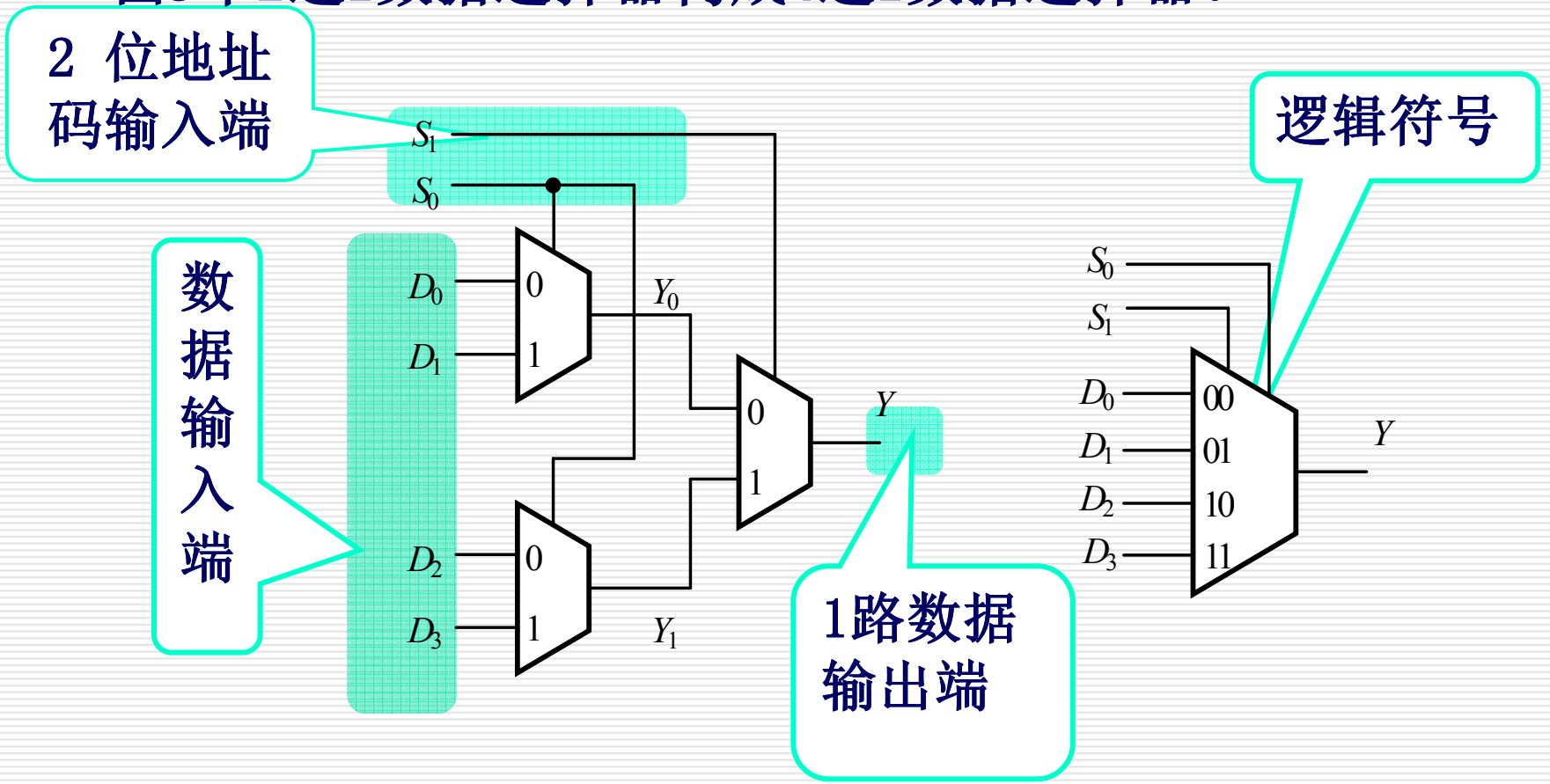
$$Y = \bar{S}D_0 + SD_1$$



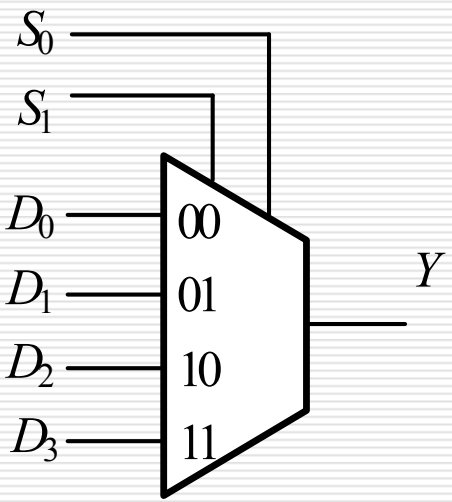
# 4选1数据选择器

## （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** | $D_3$ |

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

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

## （3）数据选择器实现逻辑函数

**例4.4.8　试用数据选择器实现下列逻辑函数**

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

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

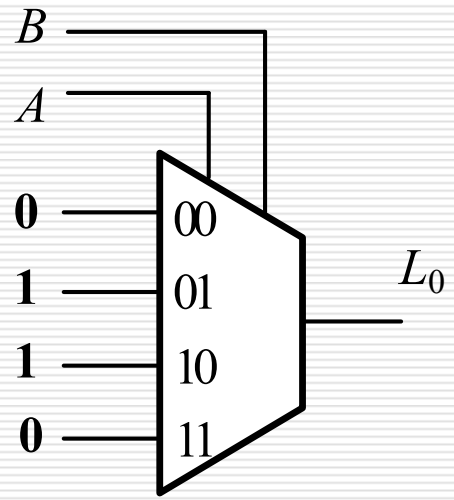
$$Y = \overline{S_1}\,\overline{S_0}D_0 + \overline{S_1}S_0D_1 + S_1\overline{S_0}D_2 + S_1S_0D_3$$

$$L_1 = \overline{A}\,B + A\,\overline{B}$$

$$= 0 + \overline{A}\,B * 1 + A\,\overline{B} * 1 + 0$$

① 当 $S_1 = A,\ S_0 = B,$

$D_0 = D_3 = 0,\ D_1 = D_2 = 1$



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

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

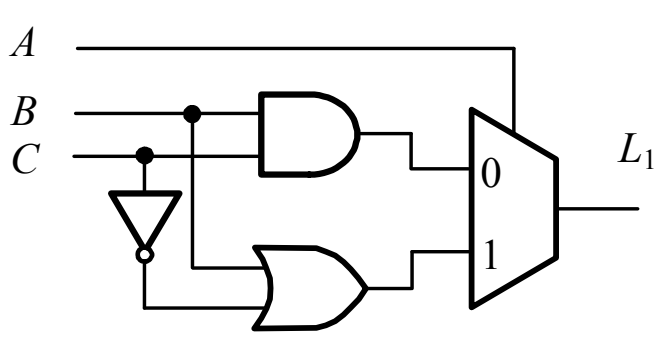
$$L_1 = AB + A\overline{C} + BC$$

$$= AB + A\overline{C} + (A + \overline{A})BC$$

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

$$Y = \overline{S}D_0 + SD_1$$

| 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 | |




(c)

## 总结：

利用数据选择器实现函数的一般步骤：（变量数=选通端数）

a、将函数变换成最小项表达式

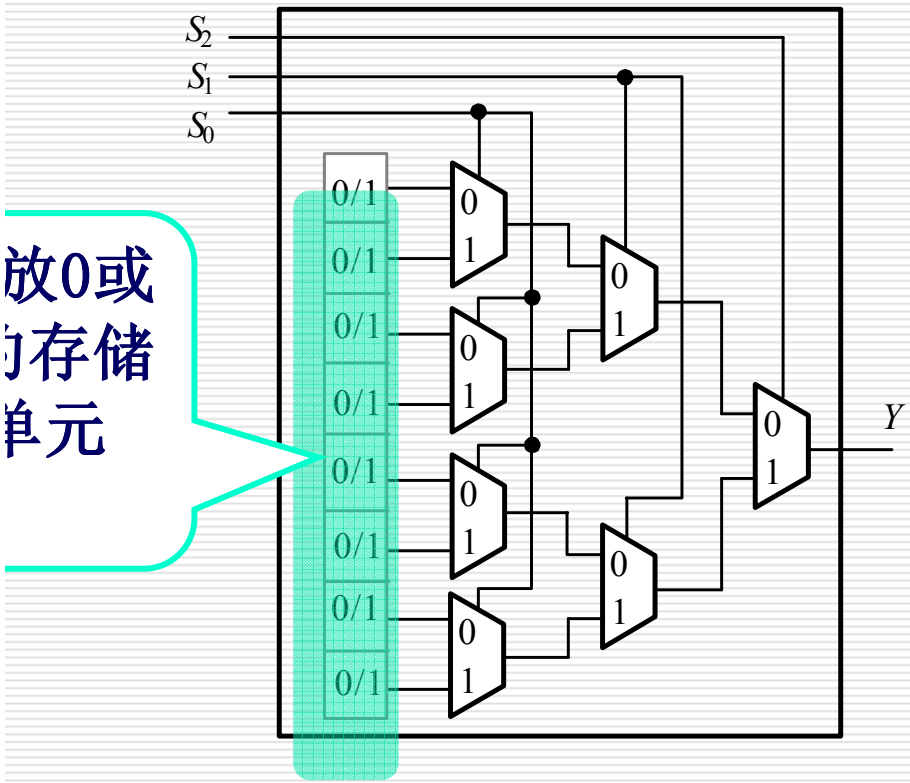
b、地址信号$S_2$、$S_1$、$S_0$作为函数的输入变量

c、处理数据输入$D_0$~$D_7$信号电平。逻辑表达式中有$m_i$,则相应$D_i$=1，其他的数据输入端均为0。

当变量数>选通端数，考虑如何将某些变量接入数据端。

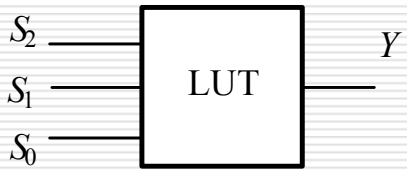
## （4）数据选择器构成查找表LUT

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。




放0或
的存储
单元

(a) 结构




(b)符号

| 输 入 | | | 输出 |
|---|---|---|---|
| $S_2$ | $S_1$ | $S_0$ | Y |
| 0 | 0 | 0 | $D_0$ |
| 0 | 0 | 1 | $D_1$ |
| 0 | 1 | 0 | $D_2$ |
| 0 | 1 | 1 | $D_3$ |
| 1 | 0 | 0 | $D_4$ |
| 1 | 0 | 1 | $D_5$ |
| 1 | 1 | 0 | $D_6$ |
| 1 | 1 | 1 | $D_7$ |

# 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。

根据前面例题已知

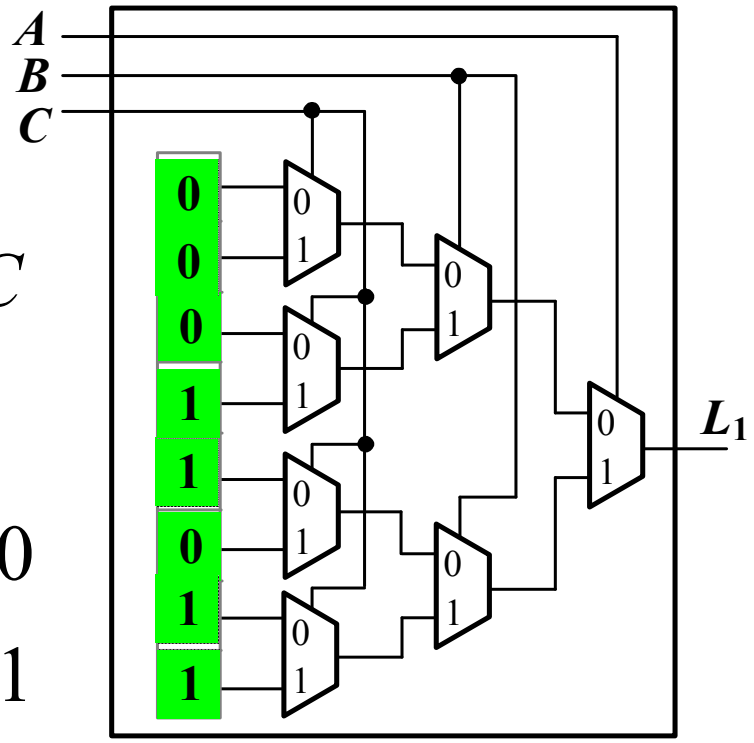
| 输　入 | | | 输出 |
|---|---|---|---|
| $S_2$ | $S_1$ | $S_0$ | L1 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

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

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

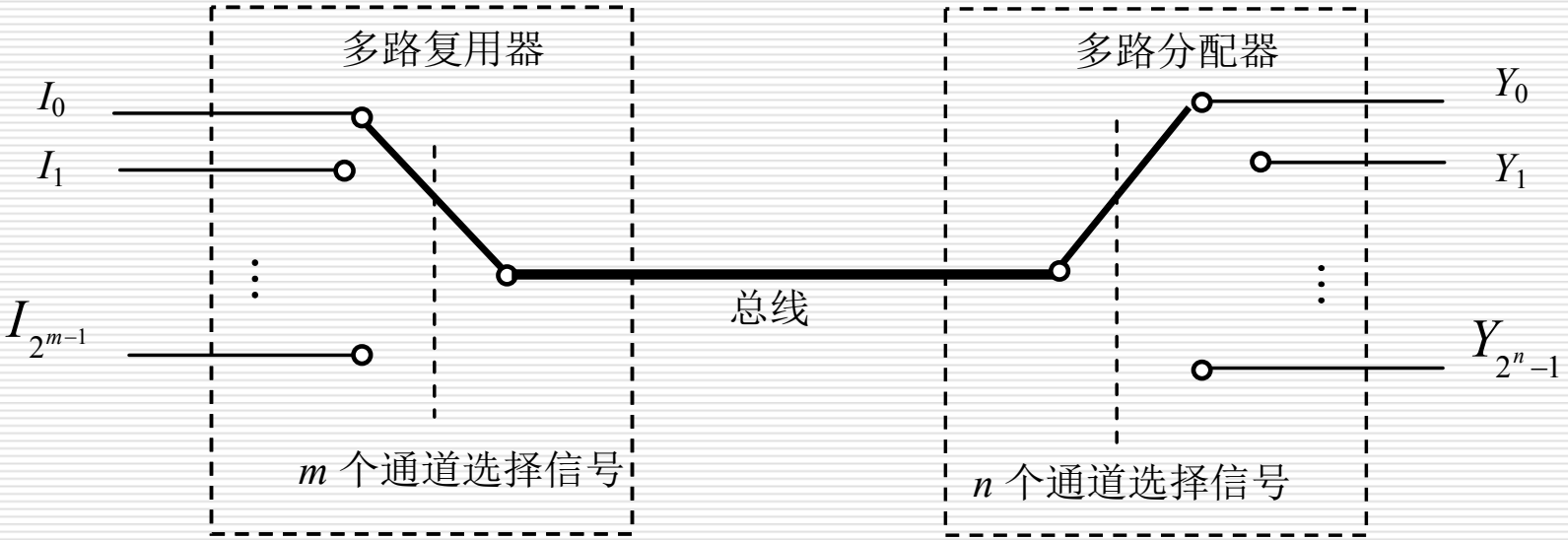
$$D_0 = D_1 = D_2 = D_5 = 0$$

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



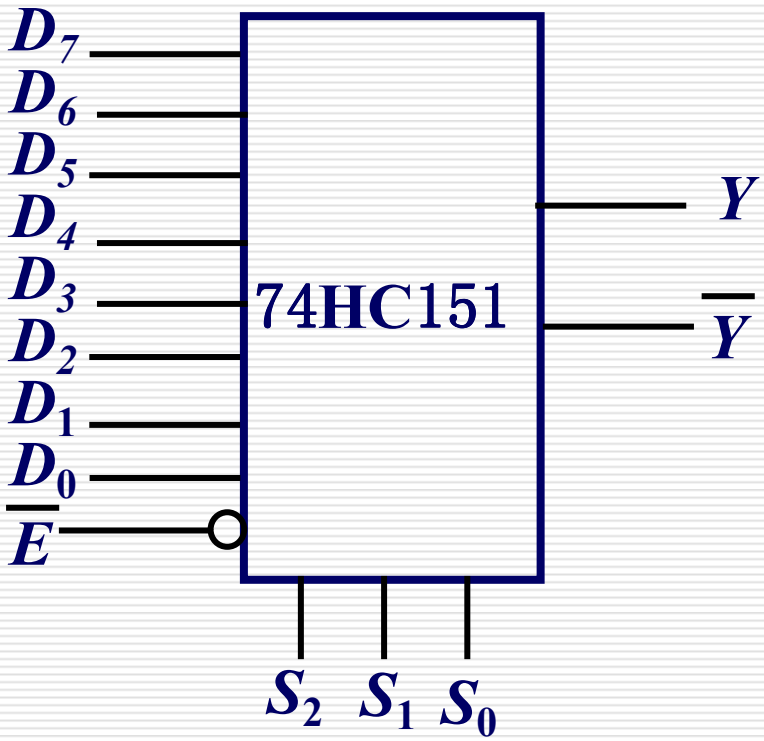
## （5）数据选择器、数据分配器与总线的连接

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。



## （6）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=1 。
- 当$\overline{E}$=0时

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

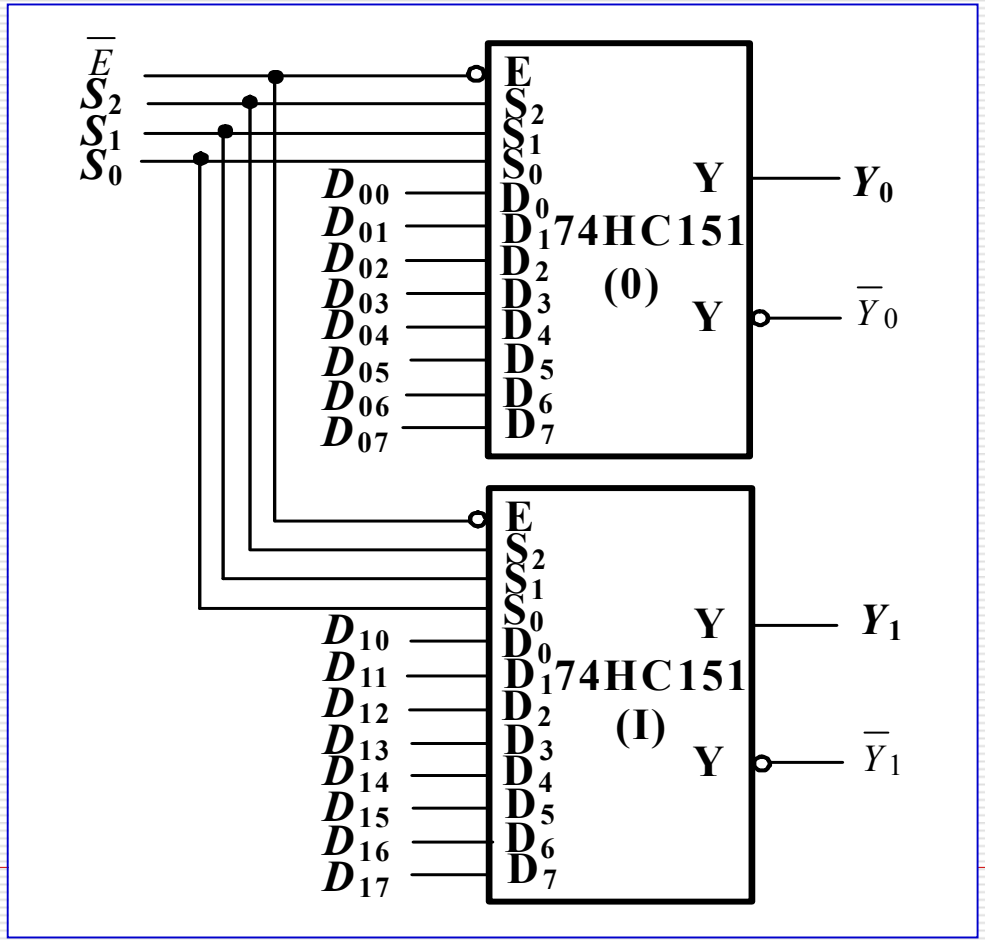
$$Y=\sum_{i=0}^{7}D_i m_i$$

| 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 | $D_7$ | $\overline{D_7}$ |

# 数据选择器的扩展

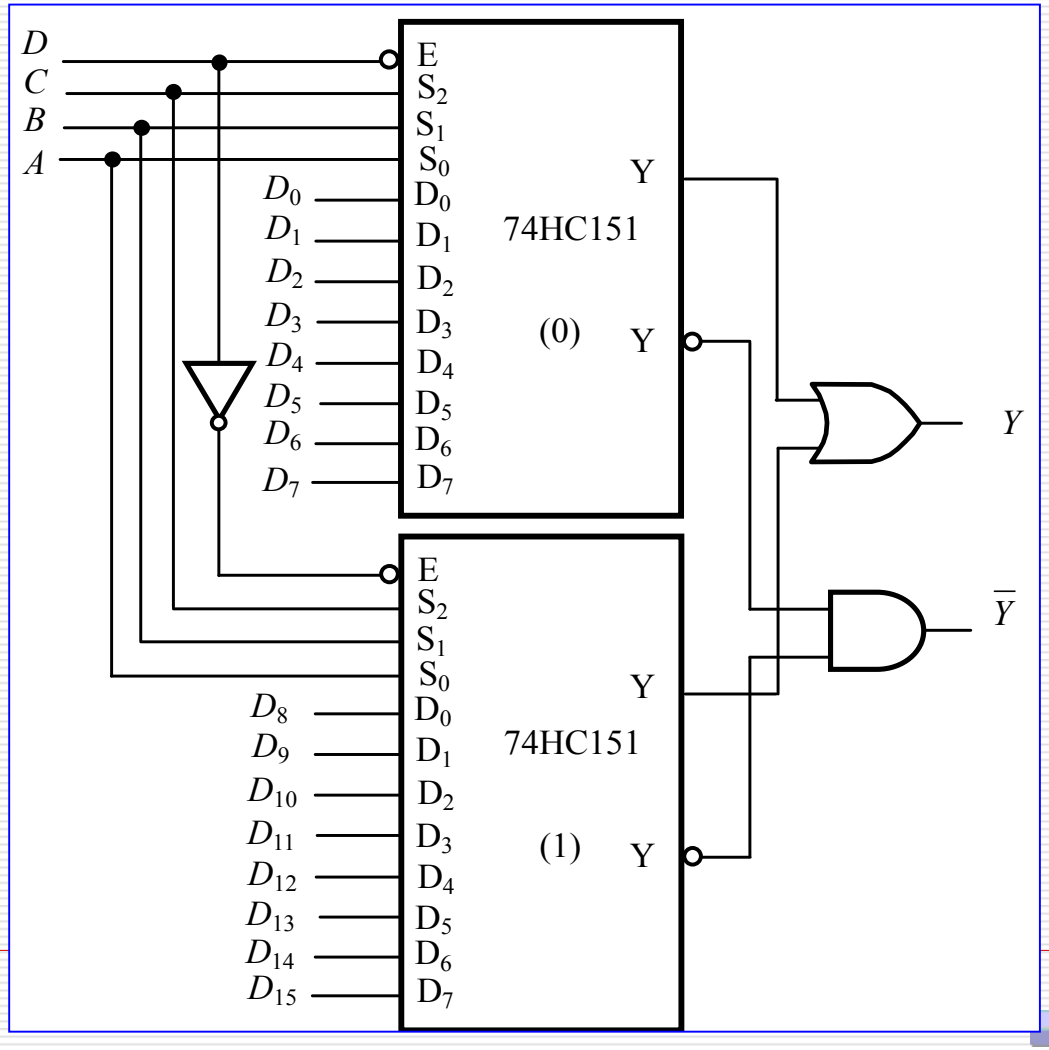
## 位的扩展

用两片74151组成二位八选一的数据选择器



# 字的扩展

## 将两片74LS151连接成一个16选1的数据选择器，



# 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

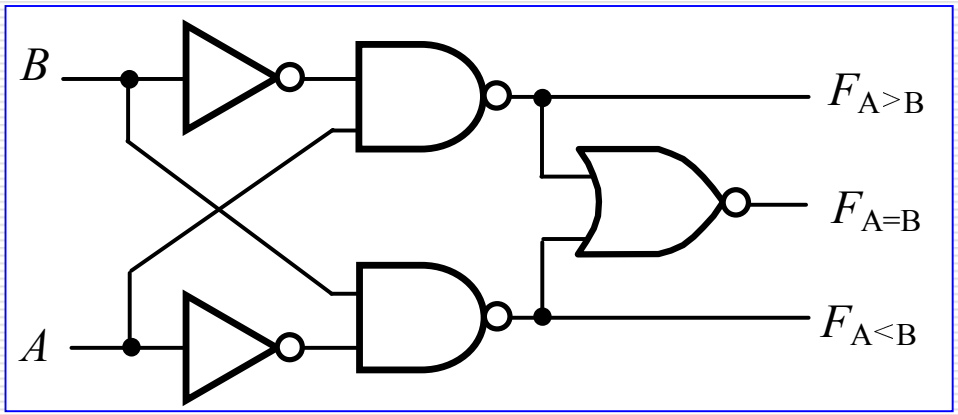
$$F_{A>B} = A\overline{B}$$

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

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

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 |



## 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 $A=A_1A_0$ 、$B=B_1B_0$

能否用1位数值比较器设计两位数值比较器?

用一位数值比较器设计多位数值比较器的原则

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

当高位相等时，两数的比较结果由低位比较的结果决定。

# 真值表

| 输入 | | | | 输出 | | |
|---|---|---|---|---|---|---|
| $A_1$ | $B_1$ | $A_0$ | $B_0$ | $F_{A>B}$ | $F_{A<B}$ | $F_{A=B}$ |
| $A_1 > B_1$ | | × | | 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 |

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

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

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

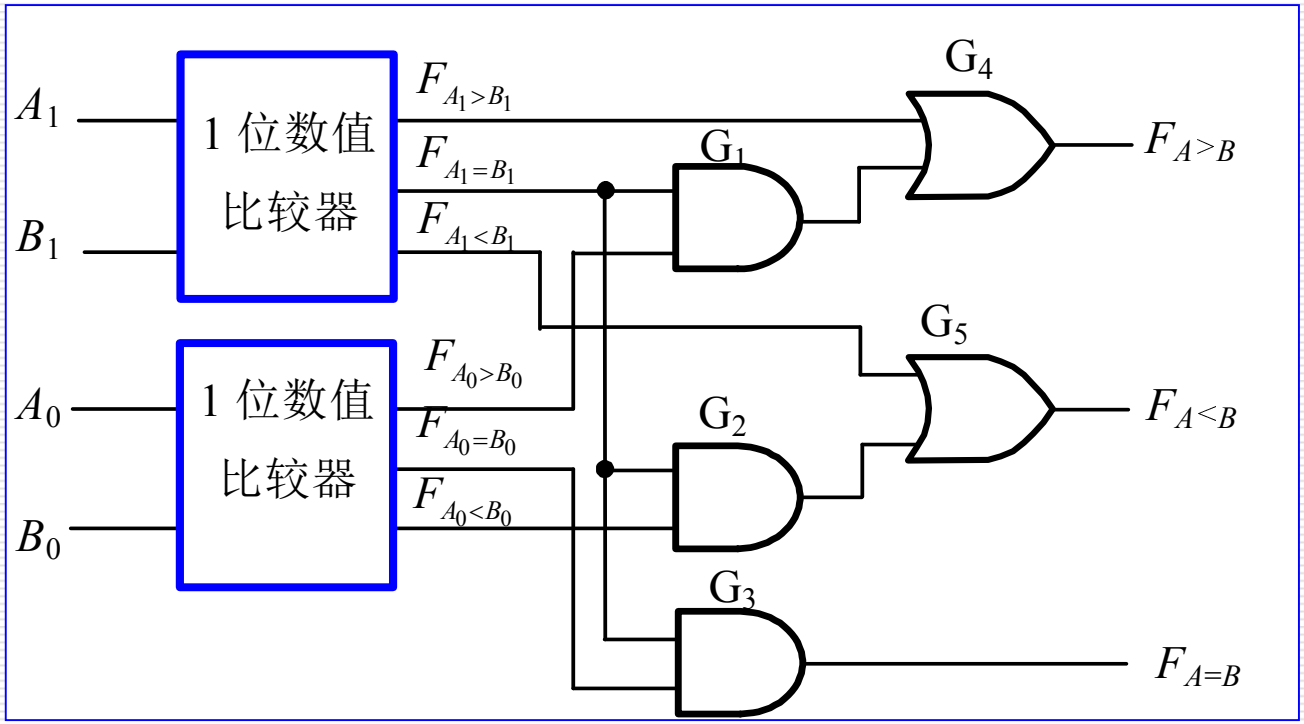
注意：上述不是真正的逻辑函数表达式，只示意逻辑关系。

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

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

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

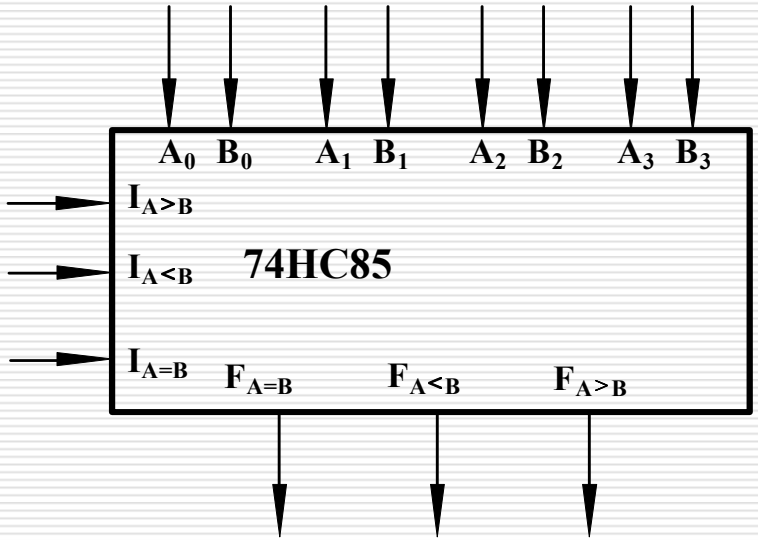
## 两位数值比较器逻辑图



## 3、 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

# 4位数值比较器74HC85的功能表

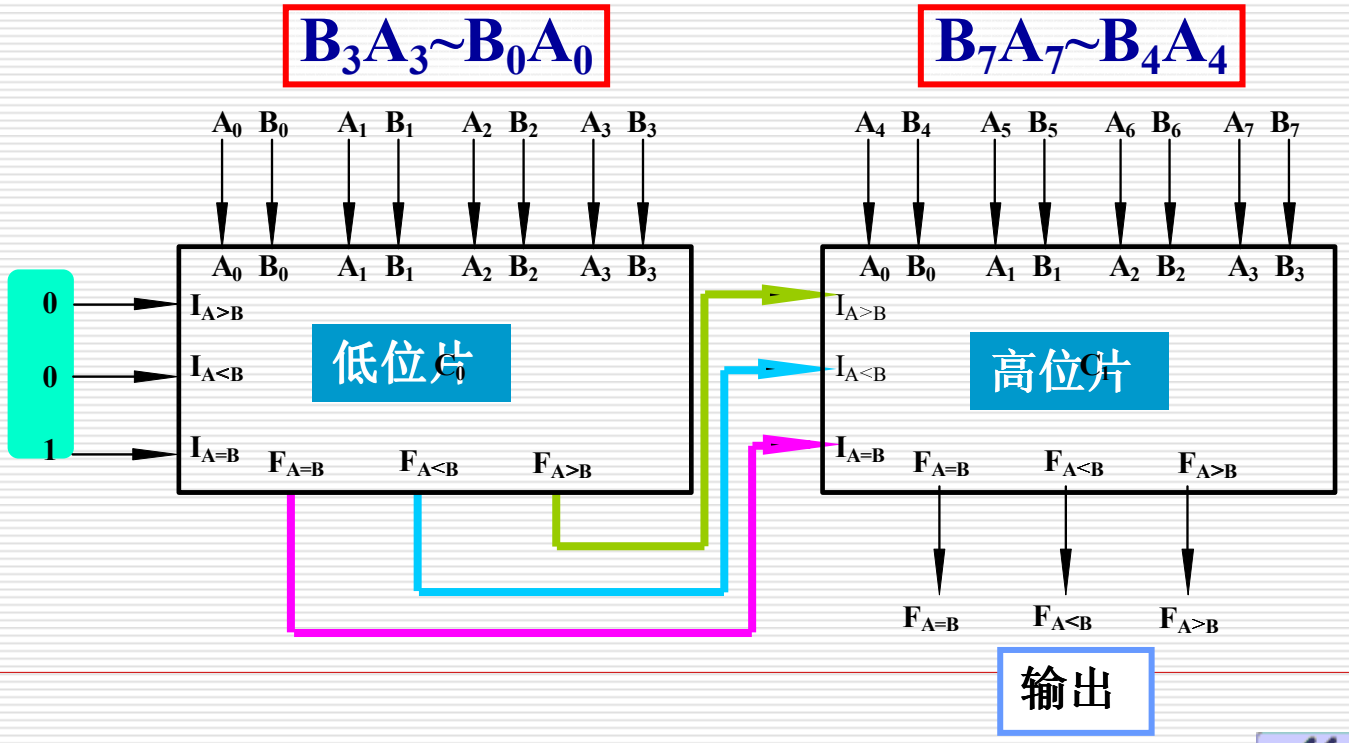
| 输入 | | | | | | | | 输出 | | |
|---|---|---|---|---|---|---|---|---|---|---|
| $A_3$ $B_3$ | $A_2$ $B_2$ | $A_1$ $B_1$ | $A_0$ $B_0$ | $I_{A>B}$ | $I_{A<B}$ | $I_{A=B}$ | $F_{A>B}$ | $F_{A<B}$ | $F_{A=B}$ |
| $A_3 > B_3$ | × | × | × | × | × | × | 1 | 0 | 0 |
| $A_3 < B_3$ | × | × | × | × | × | × | 0 |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

# 4、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）。

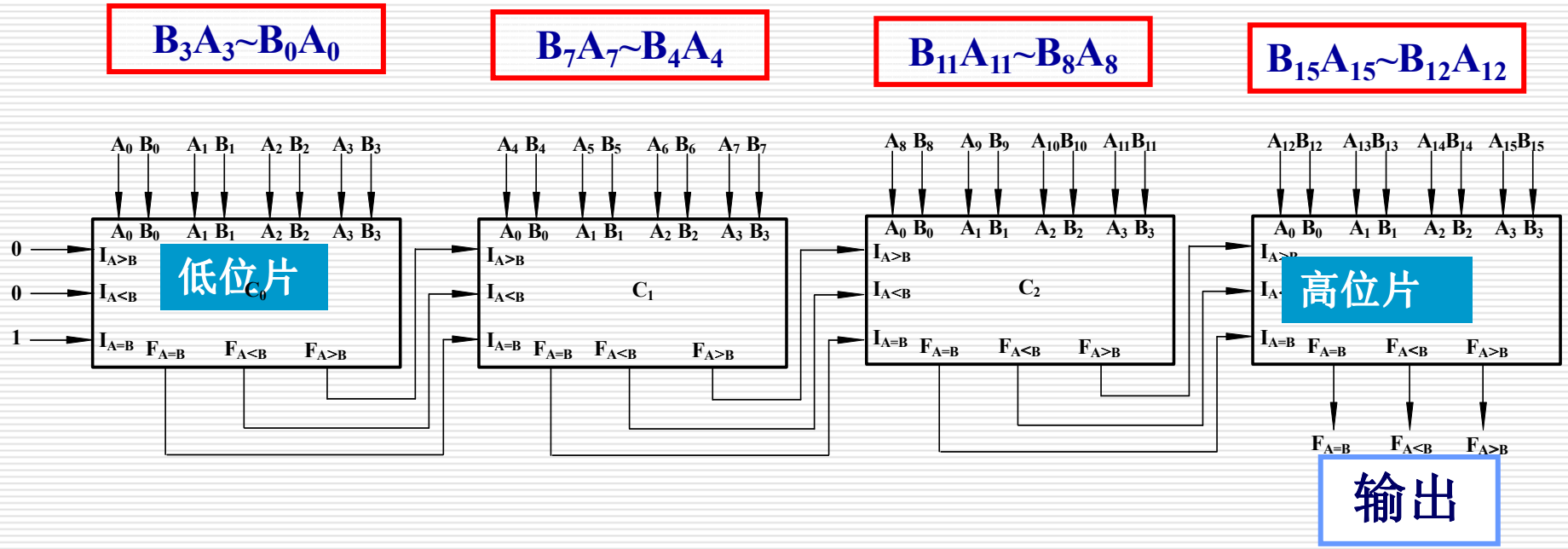
输入：$A=A_7A_6A_5A_4A_3A_2A_1A_0$　　$B=B_7B_6B_5B_4B_3B_2B_1B_0$

输出：　$F_{A>B}$　$F_{A<B}$　$F_{A=B}$



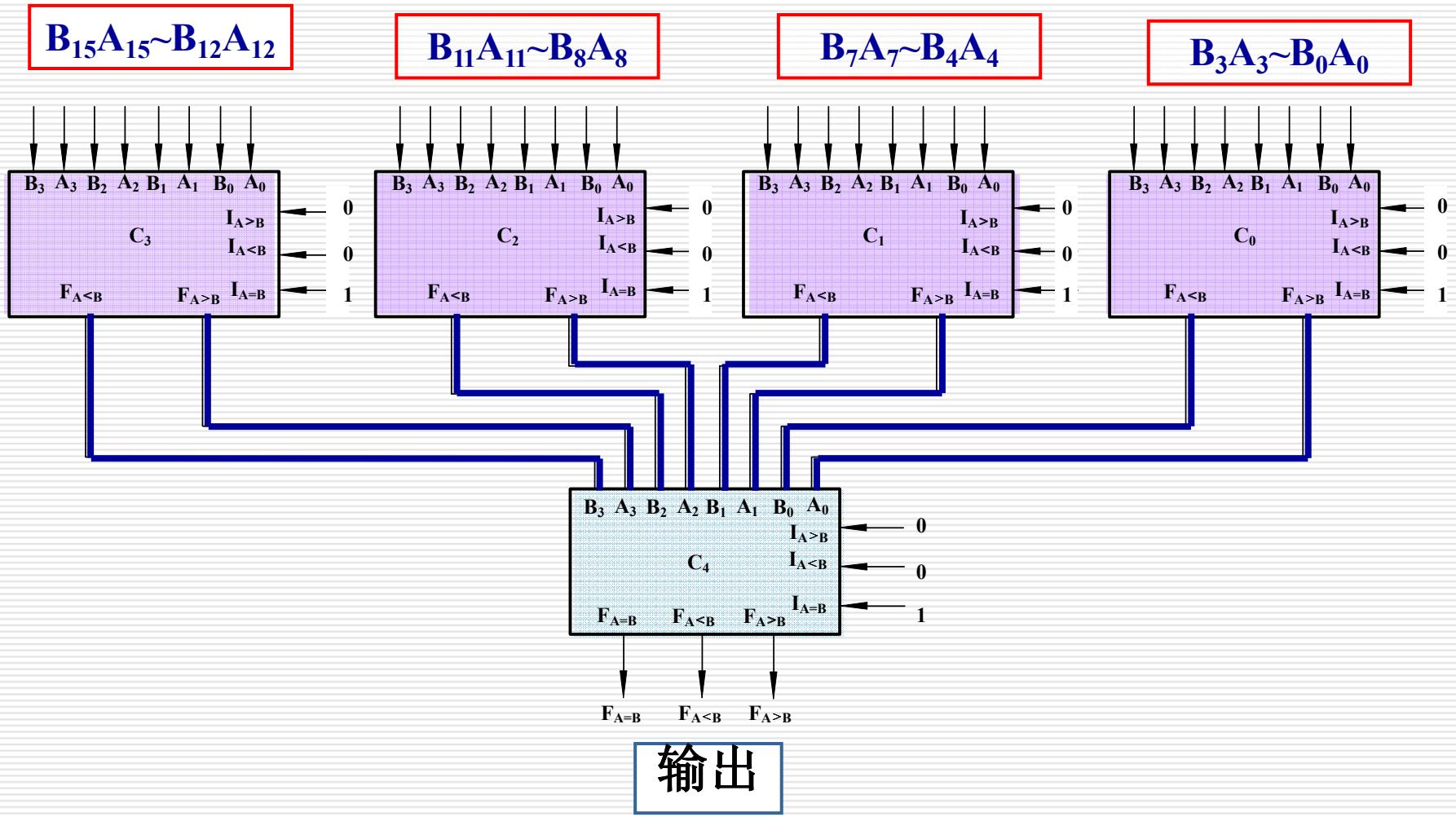
# 采用串联扩展方式数值比较器

用两片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？

# 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？

# 4.4.5 算术运算电路

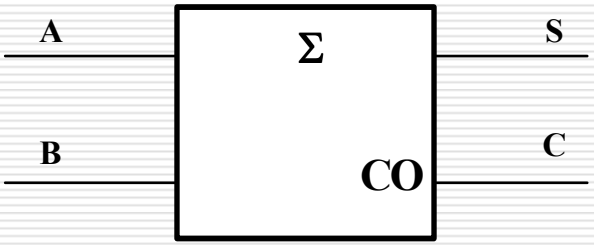
## 1、半加器和全加器

两个1位二进制数相加时，不考虑低位来的进位的加法---半加

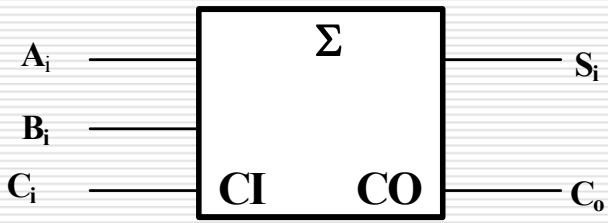
在两个1位二进制数相加时，考虑低位进位的加法---全加

加法器分为半加器和全加器两种。

半加器




全加器



# （1） 1位半加器（Half Adder）

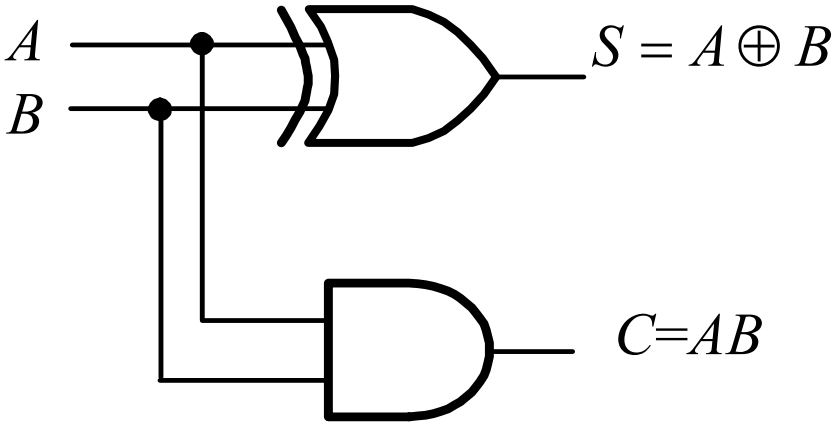
不考虑低位进位，将两个1位二进制数A、B相加的器件。

- 半加器的真值表
- 逻辑表达式

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

$$C=AB$$

如用与非门实现最少要几个门?



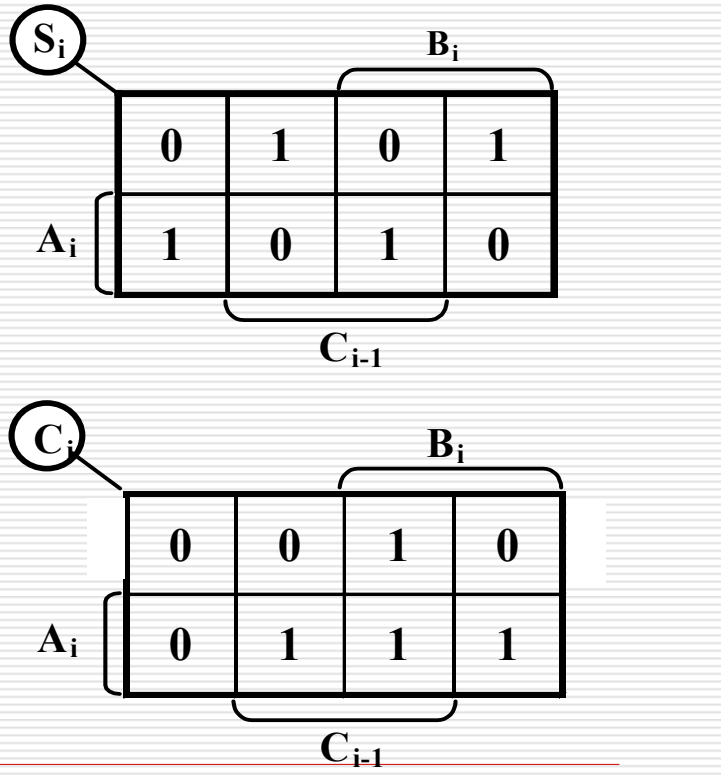

逻辑图

# （2） 全加器（Full Adder）

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

全加器真值表

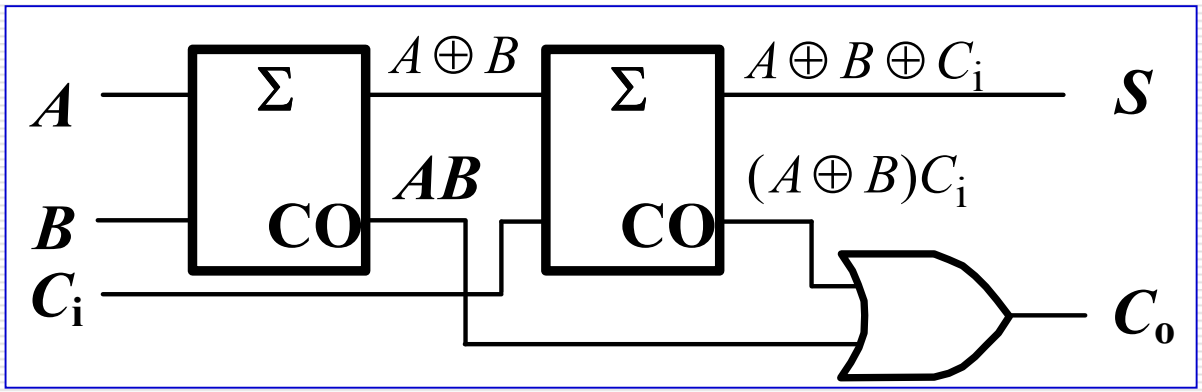
| A | B | $C_i$ | S | $C_o$ |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |



## 于是可得全加器的逻辑表达式为

$$S = \overline{A}\,\overline{B}C_{\mathrm{i}} + \overline{A}B\overline{C}_{\mathrm{i}} + A\overline{B}\,\overline{C}_{\mathrm{i}} + ABC_{\mathrm{i}}$$
$$= A \oplus B \oplus C_{\mathrm{i}}$$

$$C_{\mathrm{o}} = AB + A\overline{B}C_{\mathrm{i}} + \overline{A}BC_{\mathrm{i}}$$
$$= AB + (A \oplus B)C_{\mathrm{i}}$$




- 你能用74151\74138设计全加器吗?
- 用这两种器件组成逻辑函数产生电路, 有什么不同?

# 加法器的应用

全加器真值表

| $A$ | $B$ | $C_{i-1}$ | $S$ | $C_i$ |
|---|---|---|---|---|
| 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*有奇数个1时S为1；

*ABC*有偶数个1和全为0时

S为0。

-----用全加器组成三位二进制代码奇偶校验器

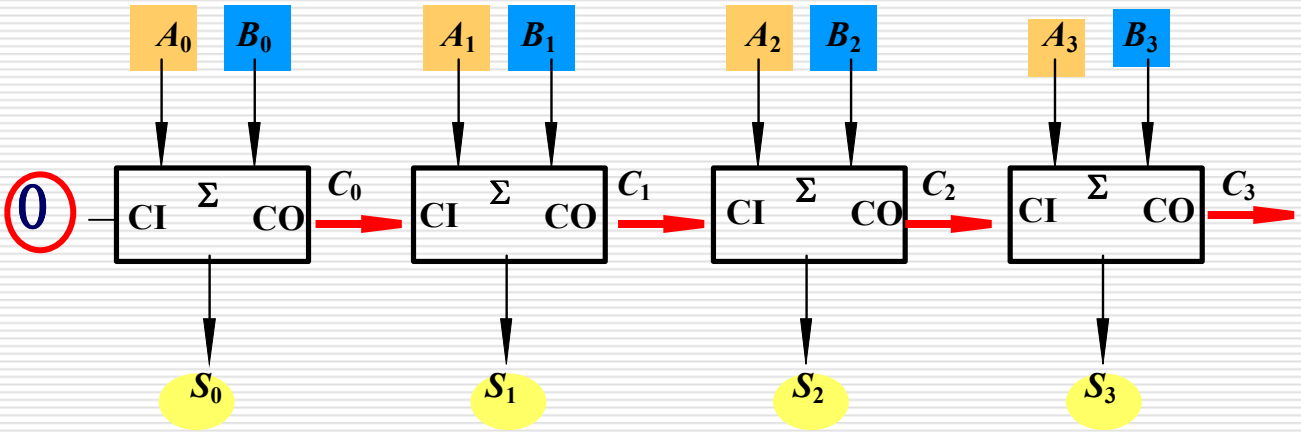
用全加器组成八位二进制代码奇偶校验器，电路应如何连接？

# 2、多位数加法器

•如何用1位全加器实现两个四位二进制数相加？

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

（1）串行进位加法器




•低位的进位信号送给邻近高位作为输入信号，采用串行进位加法器运算速度不高。

## （2）超前进位加法器

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

定义第$i$位的进位信号（$C_i$）：

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

定义两个中间变量$G_i$和$P_i$：

$$G_i = A_iB_i$$

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

$$C_i = G_i + P_iC_{i-1}$$

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

# 4位全加器进位信号的产生：

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

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

$C_1 = G_1 + P_1 C_0$

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

$C_2 = G_2 + P_2 C_1$

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

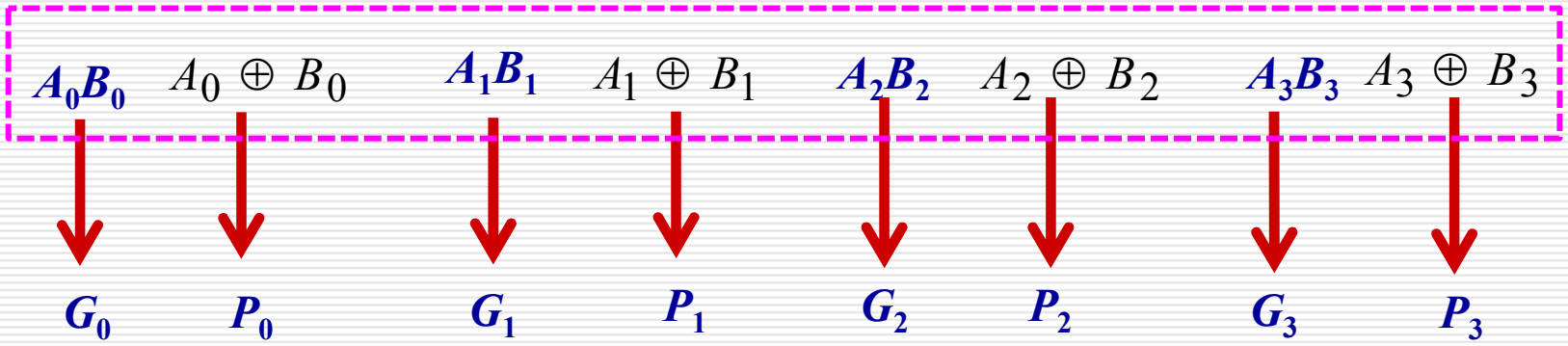
$C_3 = G_3 + P_3 C_2 = G_3 + P_3 (G_2 + P_2 C_1) = G_3 + P_3 G_2 + P_3 P_2 C_1$

$= G_3 + P_3 G_2 + P_3 P_2 (G_1 + P_1 C_0)$

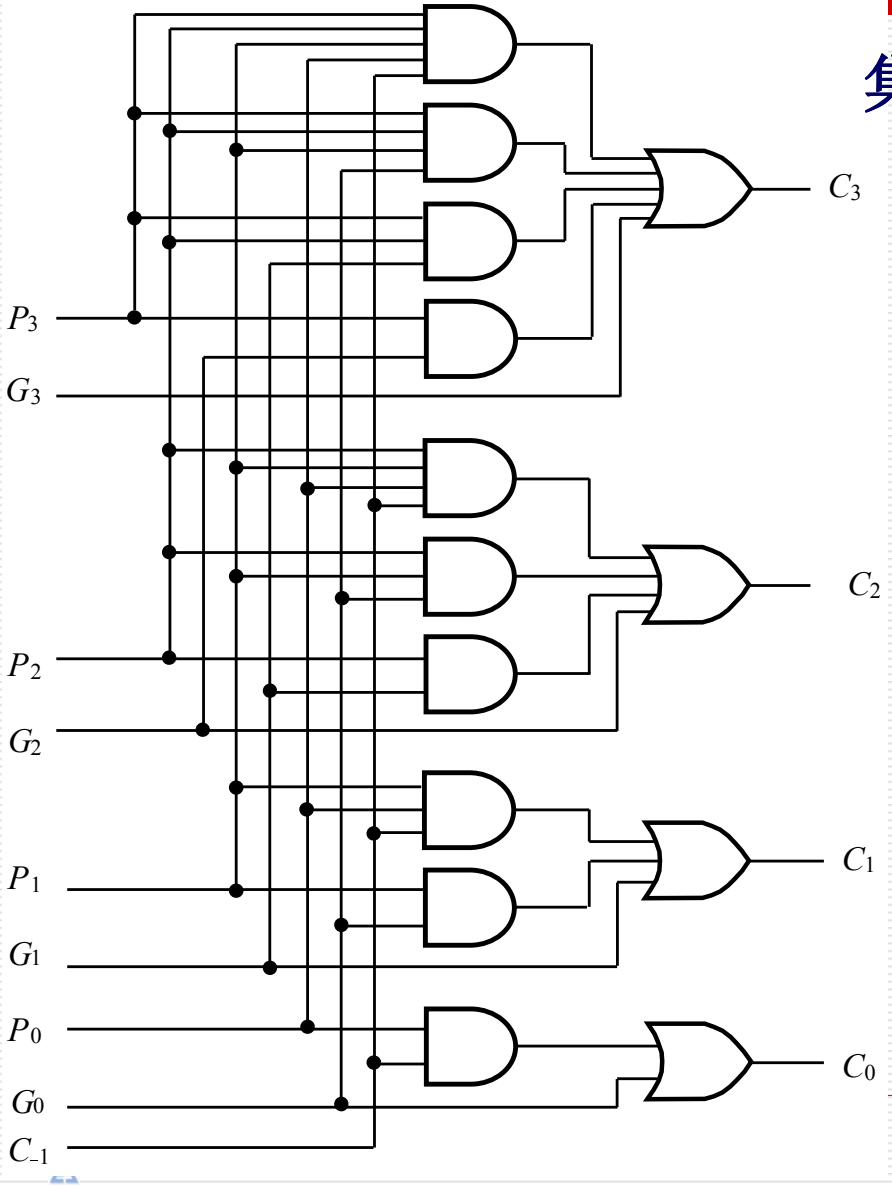
$C_3 = G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 (G_0 + P_0 C_{-1})$

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

# 多个部件并行工作

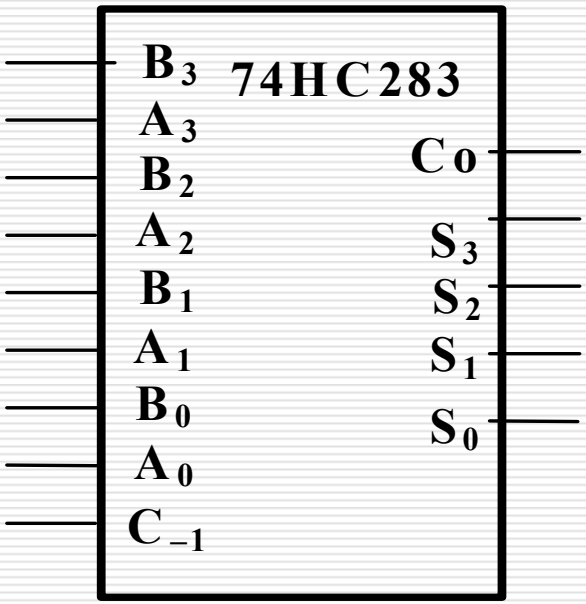

$A_0B_0$
$A_0 \oplus B_0$
$A_1B_1$
$A_1 \oplus B_1$
$A_2B_2$
$A_2 \oplus B_2$
$A_3B_3$
$A_3 \oplus B_3$
$G_0$
$P_0$
$G_1$
$P_1$
$G_2$
$P_2$
$G_3$
$P_3$

# 超前进位产生电路



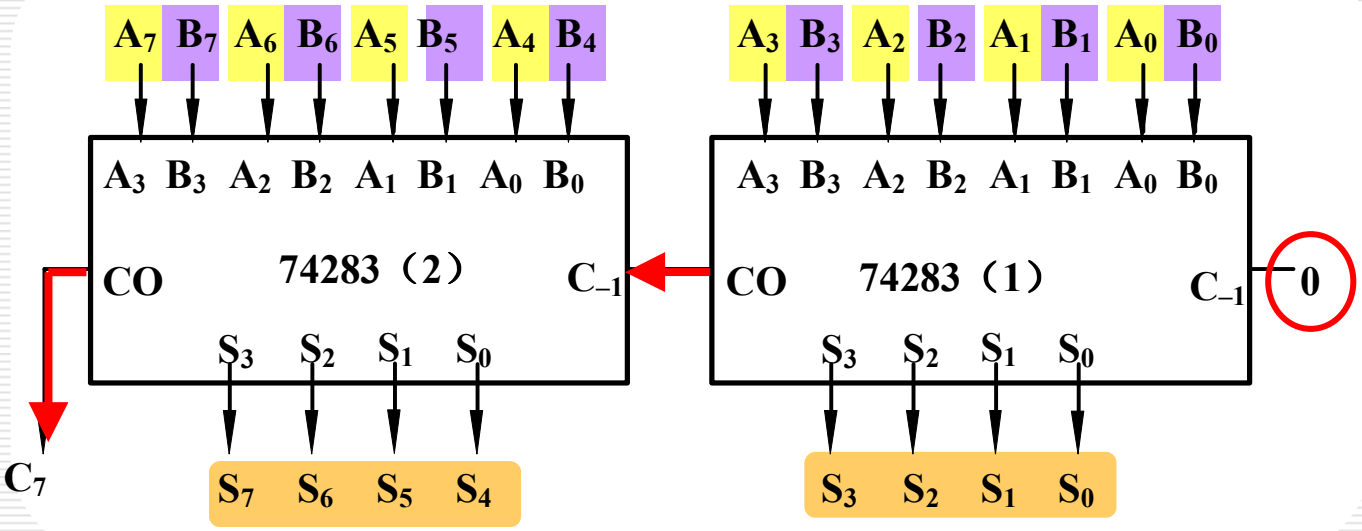

## 集成4位超前进位加法器74HC283

### 74HC283逻辑框图



# 3、超前进位加法器74LS283的应用

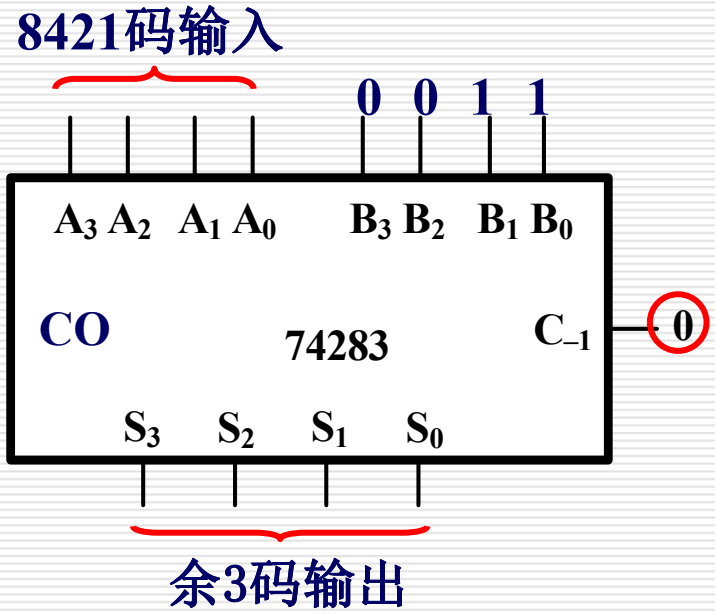
例1. 用两片74LS283构成一个8位二进制数加法器。




在片内是超前进位，而片与片之间是串行进位。

## 例. 用74283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、减法运算

若$n$位二进制的原码为$N_{原}$，则与它相对应的2 的补码为

$$N_{补}=2^N-N_{原}$$

补码与反码的关系式

$$N_{补}=N_{反}+1$$

设两个数$A$、$B$相减，利用以上两式

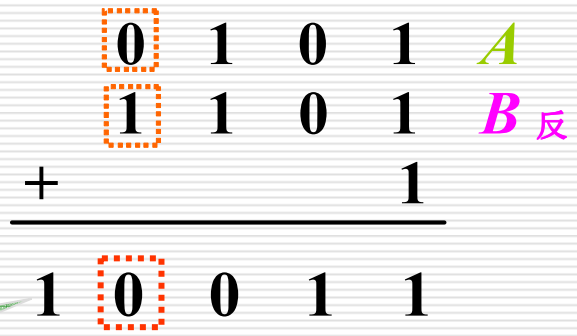
可得

$$A-B=A+B_{补}-2^n=A+B_{反}+1-2^n$$

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

1）$A-B \geq 0$的情况。

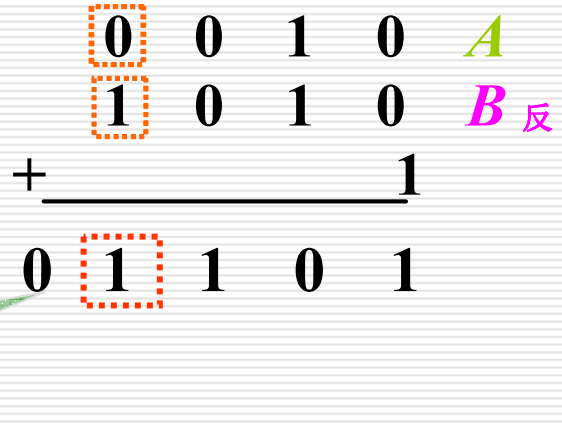
$A$=0101 ， $B$=0010




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

2）$A-B$ <0的情况。

$A$= 0001 ， $B$=0101




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

# 输出为原码的4位减法运算逻辑图