

4 组合逻辑电路

4.1 组合逻辑电路的分析

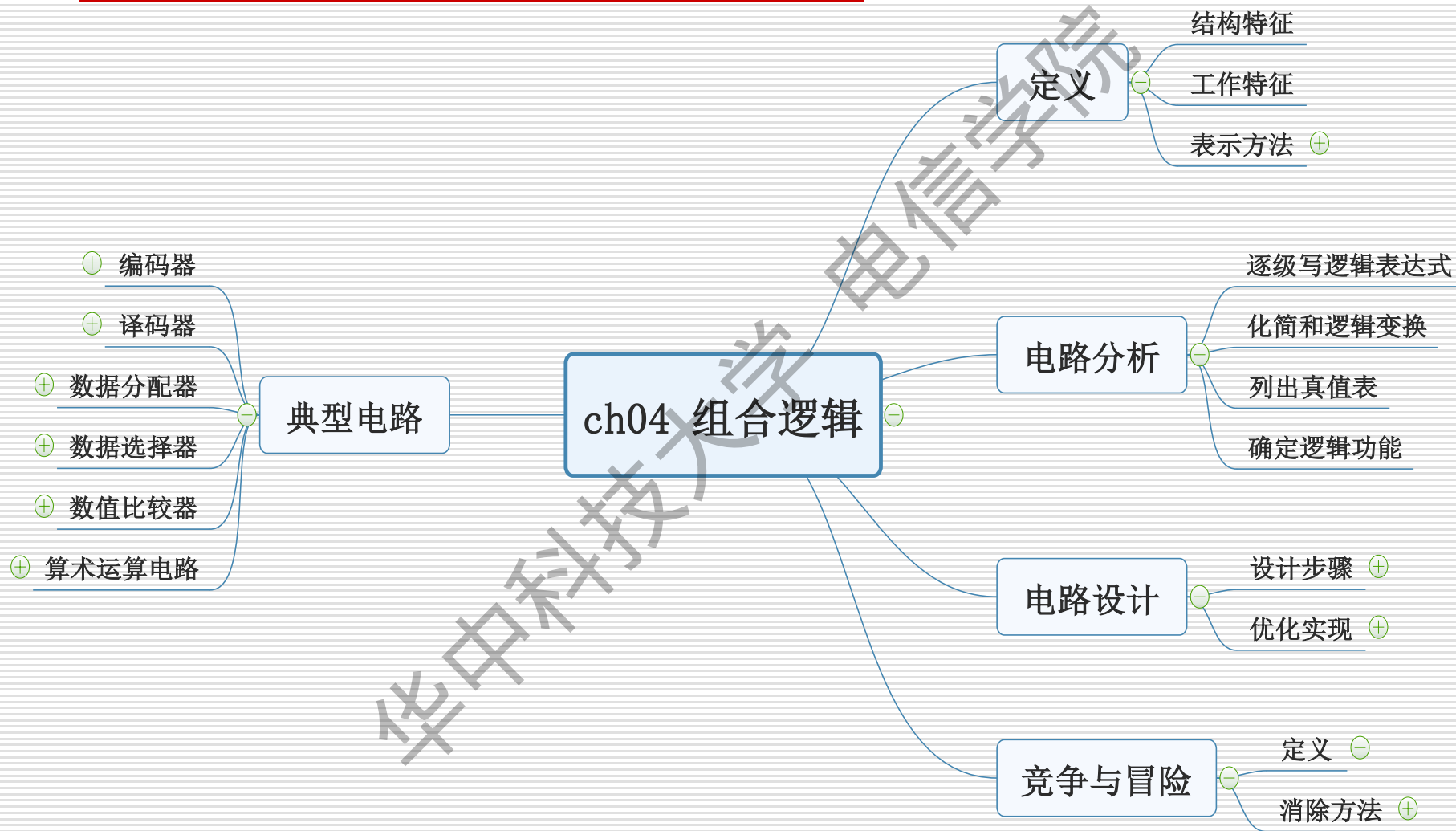
4.2 组合逻辑电路的设计

4.3 组合逻辑电路中的竞争和冒险

4.4 若干典型的组合逻辑电路

4.5 组合可编程逻辑器件

4.6 用Verilog HDL描述组合逻辑电路

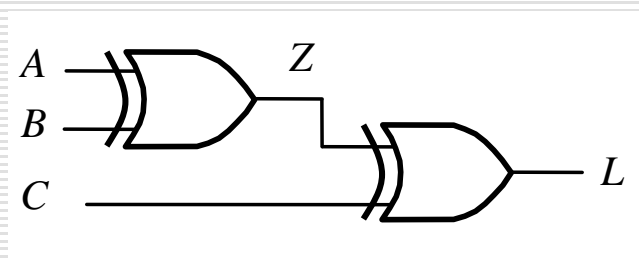


教学基本要求

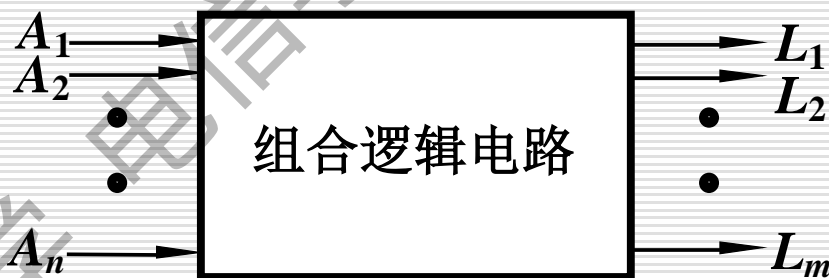
- 1.熟练掌握组合逻辑电路的分析方法和设计方法
- 2.掌握编码器、译码器、数据选择器、数值比较器和加法器的逻辑功能及其应用；
- 3.学会阅读器件的功能表，并能根据设计要求完成电路的正确连接。
- 4.掌握可编程逻辑器件的表示方法,会用PLD实现组合逻辑电路

4.1 组合逻辑电路分析

4.1.1 组合逻辑电路的定义



组合逻辑电路的一般框图



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

结构特征:

- 1、输出、输入之间没有反馈延迟通路,
- 2、不含记忆单元

工作特征:

在任何时刻, 电路的输出状态只取决于同一时刻的输入状态而与电路原来的状态无关。

组合电路逻辑功能的表示方法

◆ 组合电路逻辑功能表示方法

真值表，卡诺图，逻辑表达式，逻辑图，波形图

◆ 组合电路分类

加法器

比较器

编码器

译码器

数据选择器和分配器

码变换器

只读存储器

4.1.2 组合逻辑电路的分析方法

一. 组合逻辑电路分析

根据已知逻辑电路，经分析确定电路的逻辑功能。

二. 组合逻辑电路的分析步骤：

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

分析目的：确定电路的逻辑功能

逻辑图



逻辑表达式



说明功能



化简

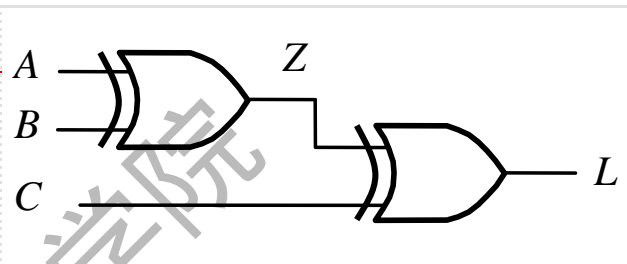


真值表



三、组合逻辑电路的分析举例

例1 分析如图所示逻辑电路的功能。



解：1.根据逻辑图写出输出函数的逻辑表达式

$$L = Z \oplus C$$

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

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

2. 列写真值表。

3. 确定逻辑功能：

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

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

如要实现偶校验，电路应做何改变？

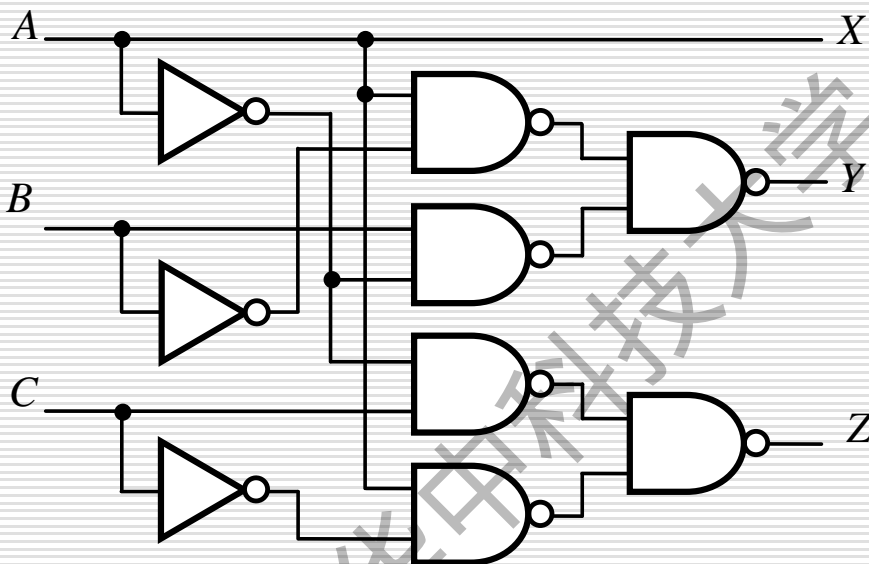
例2 试分析下图所示组合逻辑电路的逻辑功能。

解：1、根据逻辑电路写出各输出端的逻辑表达式，并进行化简和变换。

$$X = A$$

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

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



2、列写真值表

$$X = A$$

$$\begin{aligned} Y &= \overline{\overline{A}\overline{B}} \cdot \overline{\overline{A}B} = A\overline{B} + \overline{A}B \\ &= A\overline{B}(C + \overline{C}) + \overline{A}B(C + \overline{C}) \\ &= \sum m(2, 3, 4, 5) \end{aligned}$$

$$\begin{aligned} Z &= \overline{\overline{A}\overline{C}} \cdot \overline{\overline{A}C} = A\overline{C} + \overline{A}C \\ &= A\overline{C}(B + \overline{B}) + \overline{A}C(B + \overline{B}) \\ &= \sum m(1, 3, 4, 6) \end{aligned}$$

真值表

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

3、确定电路逻辑功能

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

真值表

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

4.2 组合逻辑电路的设计

4.2.1 组合逻辑电路的设计过程

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

二、组合逻辑电路的设计步骤

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

例1 某综艺比赛有三名评委，比赛规定至少两位评委给YES才能晋级，请设计一个根据评委的按键状态控制舞台灯光效果(晋级 or 淘汰)的逻辑电路。

解：（1）逻辑抽象。

输入信号： A 、 B 、 C 分别表示三位评委，且YES时为1，NO时为0。

输出信号： L 表示选手的比赛结果，晋级为1，淘汰为0。

(2) 根据题意列出真值表

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

(3) 写出输出逻辑表达式,并化简。

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

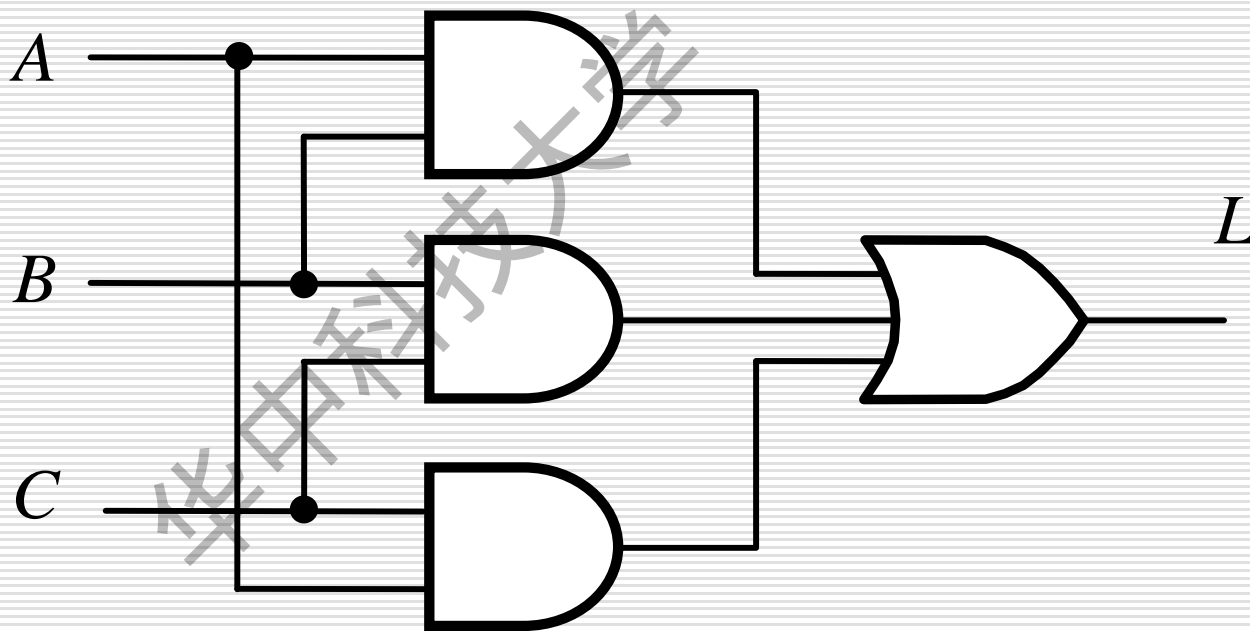
		<i>B C</i>			
		00	01	11	10
<i>A</i>	0	0	0	1	0
	1	0	1	1	1

$$L = AB + AC + BC$$

(4) 根据输出逻辑表达式画出逻辑图。

表达式为最简与或式，用与门和或门实现两级“与-或”结构的最简电路如图。

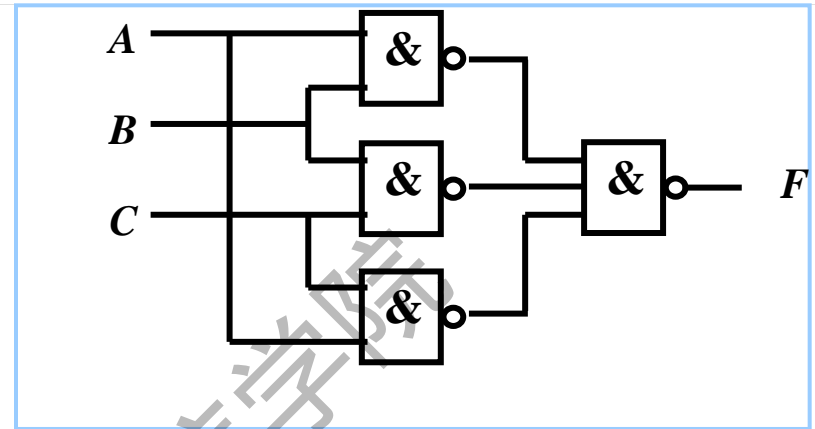
$$L = AB + AC + BC$$



(1) 若采用与非门:

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

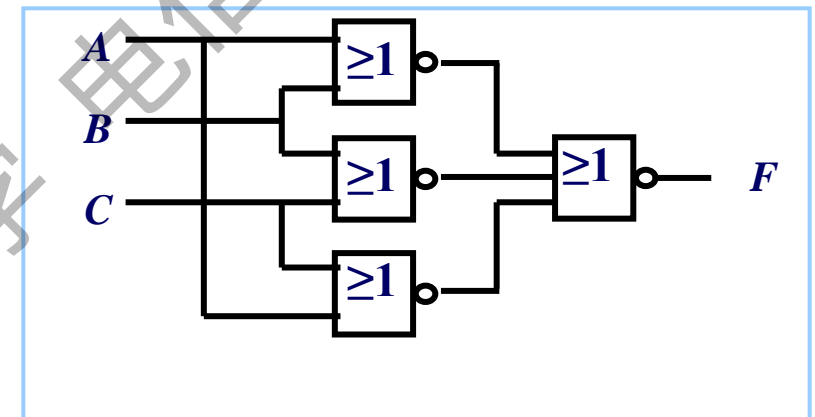
如果只有两输入的与非门?



(2) 若采用或非门:

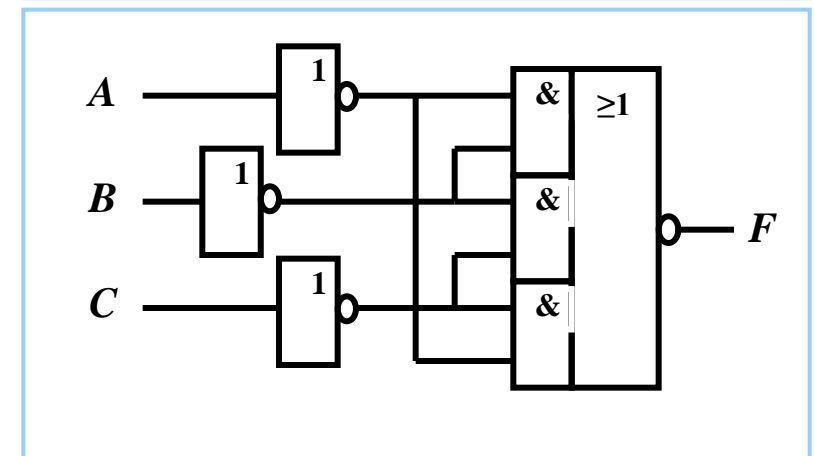
$$F = \overline{\overline{(A + B)(A + C)(B + C)}}$$
$$= \overline{A + B + A + C + B + C}$$

如果只有两输入的或非门?



(3) 若采用与或非器件?

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



例2 试设计一个码转换电路，将4位格雷码转换为自然二进制码。可以采用任何逻辑门电路来实现。

解：(1) 明确逻辑功能，列出真值表。

设输入变量为 G_3 、 G_2 、 G_1 、 G_0 为格雷码，

输出变量 B_3 、 B_2 、 B_1 和 B_0 为自然二进制码。

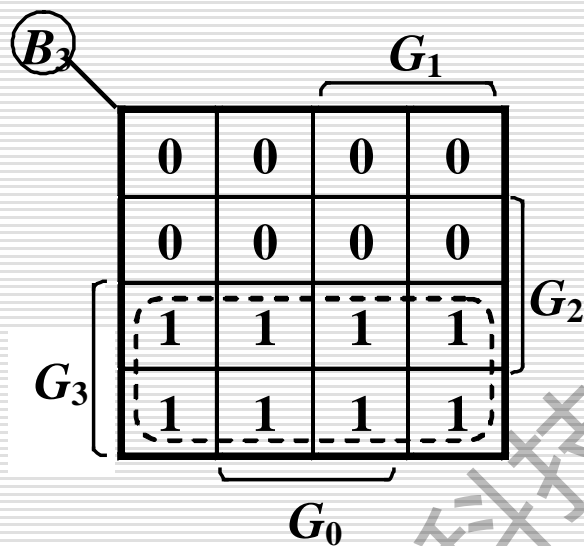
当输入格雷码按照从0到15递增排序时，

可列出逻辑电路真值表

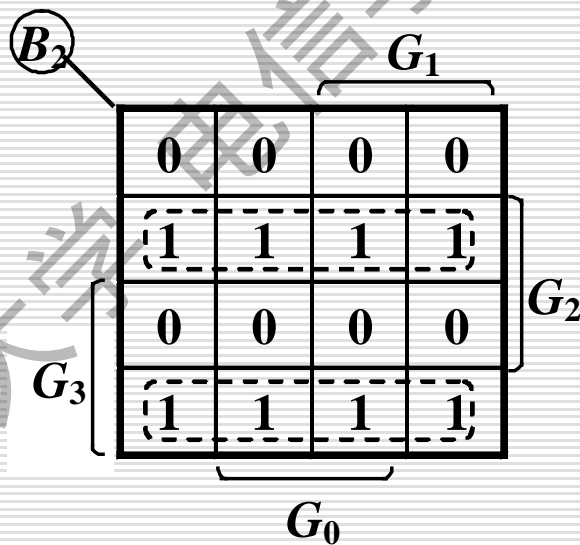
逻辑电路真值表

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

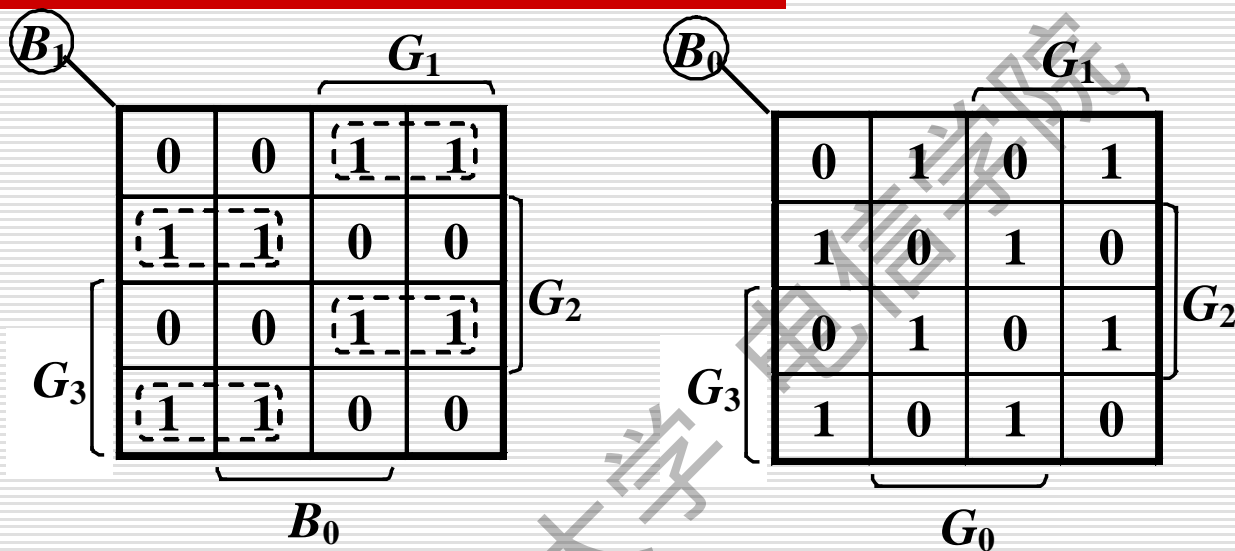
(2) 画出各输出函数的卡诺图，并化简和变换。



$$B_3 = G_3$$



$$\begin{aligned}
 B_2 &= G_3 \overline{G_2} + \overline{G_3} G_2 \\
 &= G_3 \otimes G_2
 \end{aligned}$$

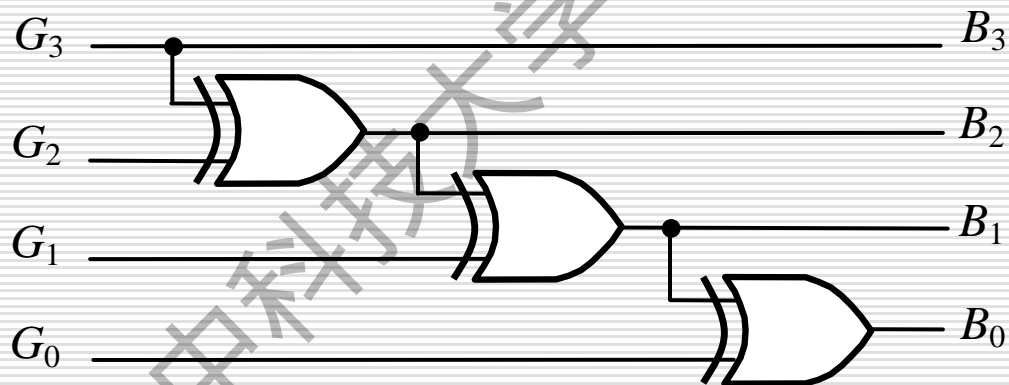


$$\begin{aligned}
 B_1 &= G_3 \overline{G_2} \overline{G_1} + \overline{G_3} G_2 \overline{G_1} + G_3 G_2 G_1 + \overline{G_3} \overline{G_2} G_1 \\
 &= (G_3 \overline{G_2} + \overline{G_3} G_2) \overline{G_1} + (G_3 G_2 + \overline{G_3} \overline{G_2}) G_1 \\
 &= (G_3 \overline{G_2} + \overline{G_3} G_2) \overline{G_1} + (\overline{G_3} \overline{G_2} + G_3 G_2) G_1 \\
 &= G_3 \oplus G_2 \oplus G_1
 \end{aligned}$$

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

(3) 根据逻辑表达式，画出逻辑图

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

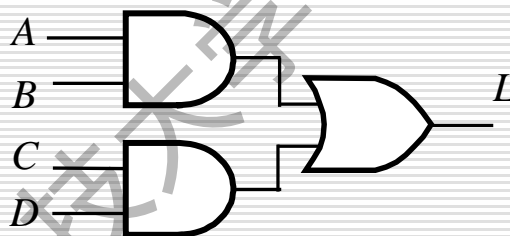


4.2.2 组合逻辑电路的优化实现

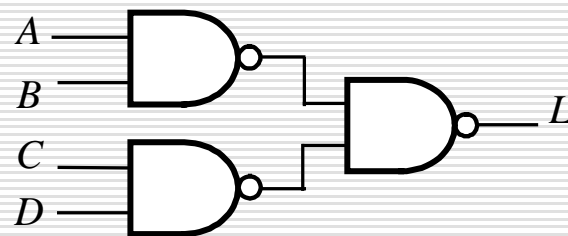
用指定芯片中特定资源实现逻辑函数，使电路的成本低并且工作速度快。因此需要对逻辑表达式进行变换，以减少芯片资源的数目和连线。

1、单输出电路

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



(a)与-或结构



(b)与非门结构

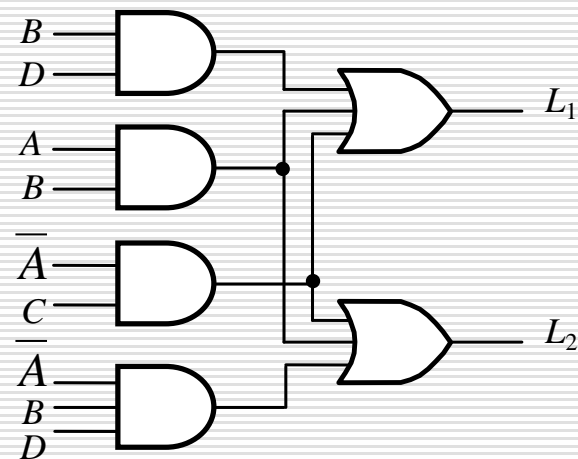
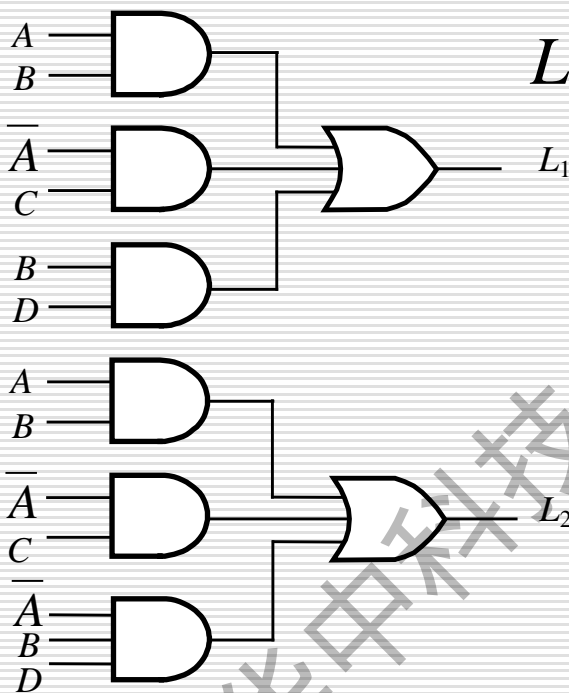
相同输入端的与非门比与门或者或门所用晶体管少，速度快。图(b)电路最优

2、多输出电路

输出多个逻辑函数时需要考虑**共享相同乘积项**，减少逻辑门数目。

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

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



(a) 如果分别实现两个逻辑函数，需要6个与门和两个或门。

(b) 如果考虑相同乘积项，需要4个与门两个或门，如图。

3、多级逻辑电路

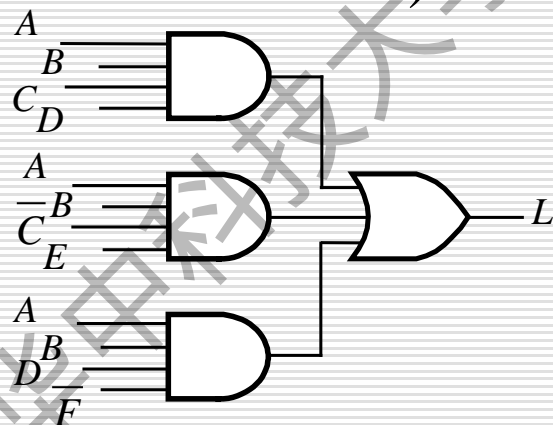
当限定逻辑门输入端数目，则需要进行逻辑变换。

(1) 提取公因子

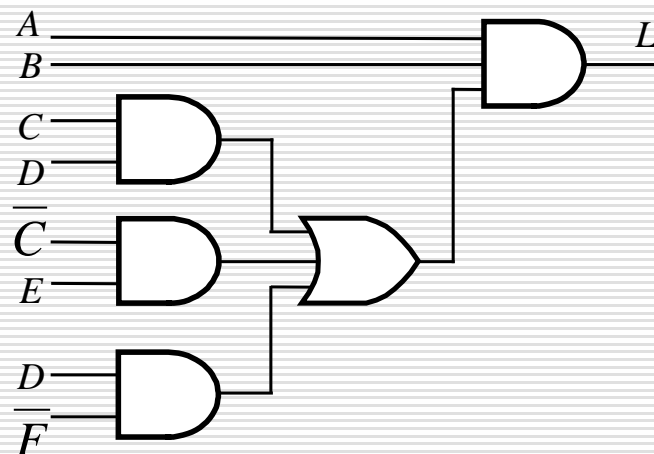
$$L = ABCD + AB\bar{C}E + ABD\bar{F}$$

用与门、或门实现时，限定逻辑门的扇入数为3，需要变换成：

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



(a)



(b)

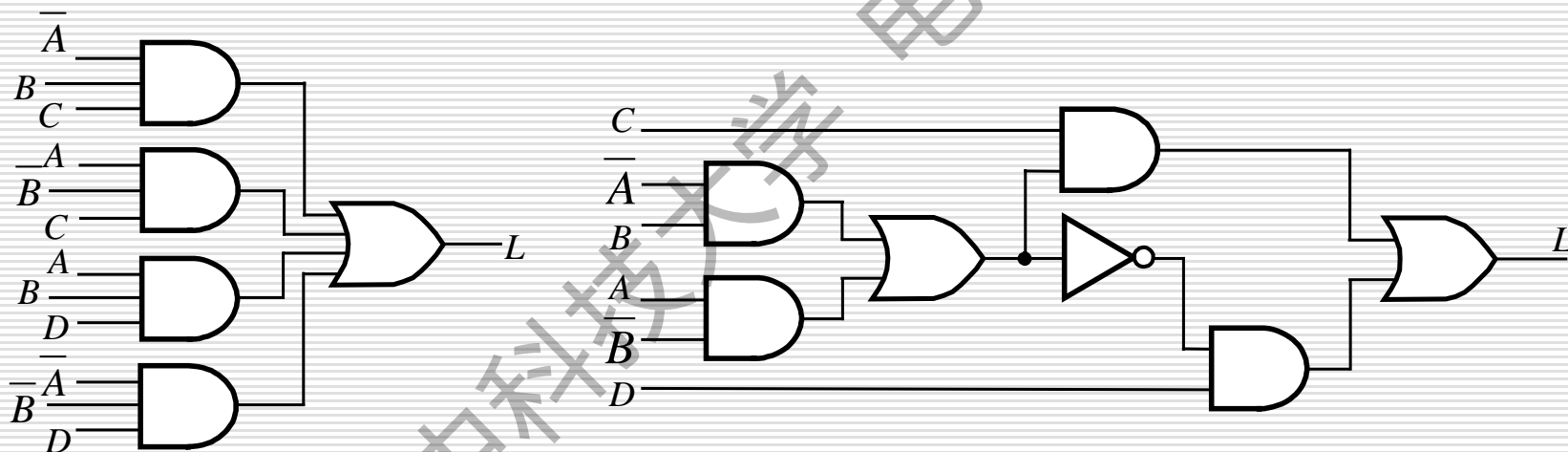
图(a)电路为2级，图(b)为3级，但电路连线减少了。图(a)16根连线，图(b)13根。

(2) 函数分解

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

用与门、或门实现时，限定逻辑门的扇入数为3，需要变换成：

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

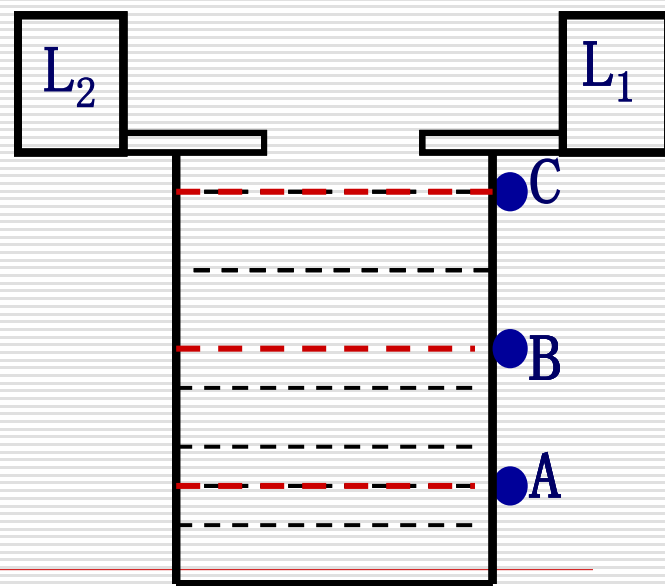


图(a)电路为2级，图(b)为5级。

上述变换方法只适合手工化简，当变量数很多时，优化策略写入程序由计算机完成。

例3 水槽由两台水泵 L_1 、 L_2 供水。 A 、 B 、 C 为三个水位检测仪，当水位低于水位检测仪时，它们输出高电平，当水位高于水位检测仪时，它们输出低电平。试用逻辑门设计一个控制两台水泵供水的电路，要求：

- 1、当水位超过C点时：
水泵 L_1 、 L_2 均停止工作；
- 2、当水位超过B点，低于C点时：
仅 L_1 工作；
- 3、当水位超过A点，低于B点时：
仅 L_2 工作；
- 4、当水位低于A点时：水泵 L_1 、 L_2 同时工作。



1、逻辑抽象:

输入变量 (A 、 B 、 C) : 为三个检测仪的输出

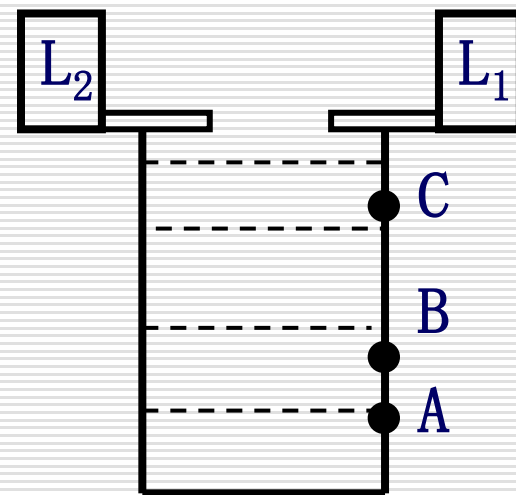
逻辑1: 水位低于水位检测仪;

逻辑0: 水位高于水位检测仪。

输出变量 (L_1 , L_2): 两个水泵

逻辑1: 水泵工作;

逻辑0: 水泵不工作。



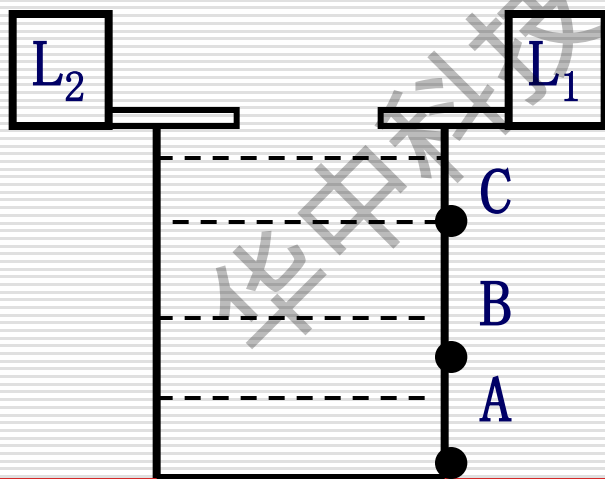
2、列真值表

当水位超过C点时， L_1 、 L_2 均停止工作；

当水位超过B点，低于C点时仅 L_1 工作；

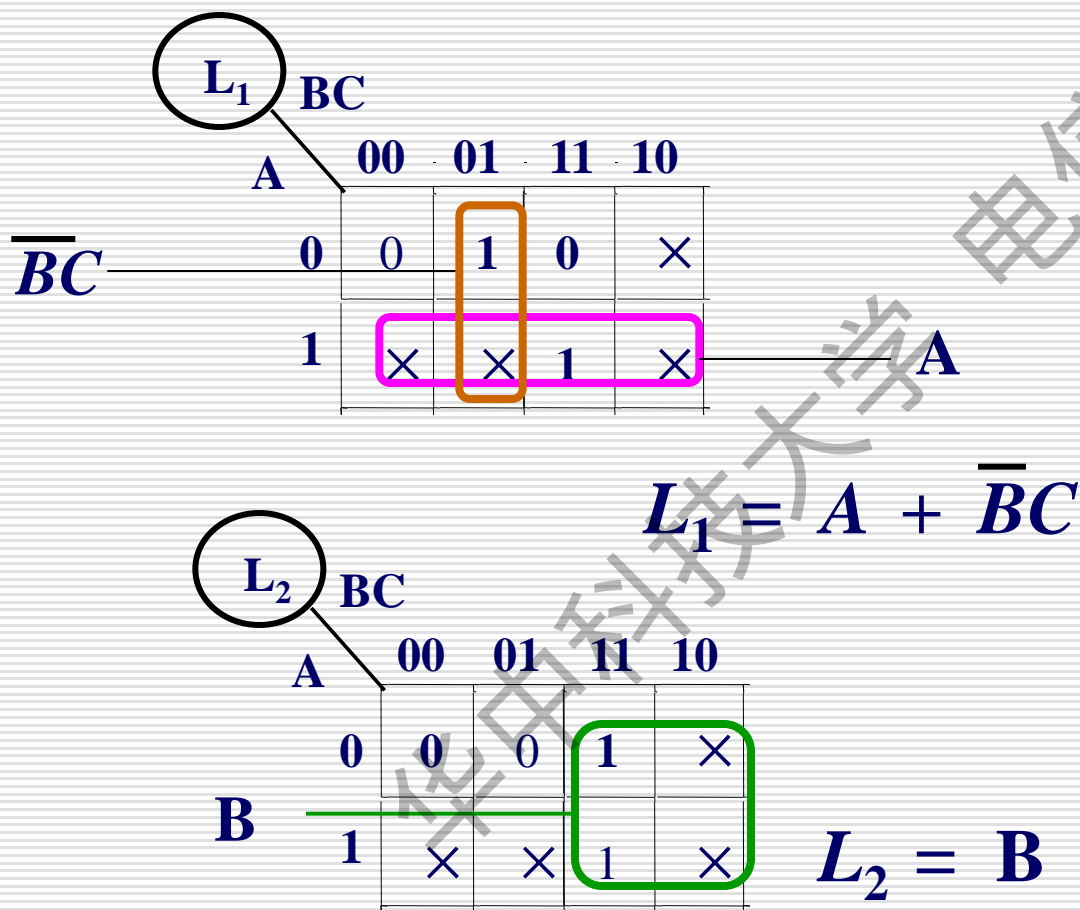
当水位超过A点，低于B点时仅 L_2 工作；

当水位低于A点时， L_1 、 L_2 同时工作。



A	B	C	L_1	L_2
0	0	0	0	0
0	0	1	1	0
0	1	0	×	×
0	1	1	0	1
1	0	0	×	×
1	0	1	×	×
1	1	0	×	×
1	1	1	1	1

3、由真值表写出逻辑表达式简化和变换逻辑表达式



A	B	C	L ₁	L ₂
0	0	0	0	0
0	0	1	1	0
0	1	0	×	×
0	1	1	0	1
1	0	0	×	×
1	0	1	×	×
1	1	0	×	×
1	1	1	1	1

4、画出逻辑电路（略）

4.3 组合逻辑电路中的竞争冒险

4.3.1 产生的竞争冒险的原因

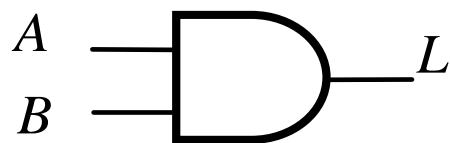
4.3.2 消去竞争冒险的方法

4.3 组合逻辑电路中的竞争冒险

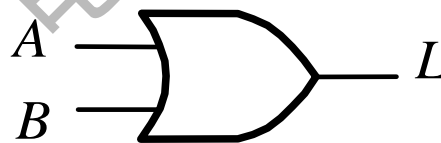
4.3.1 产生的竞争冒险的原因

不考虑门的延时时间，且 $B=\bar{A}$

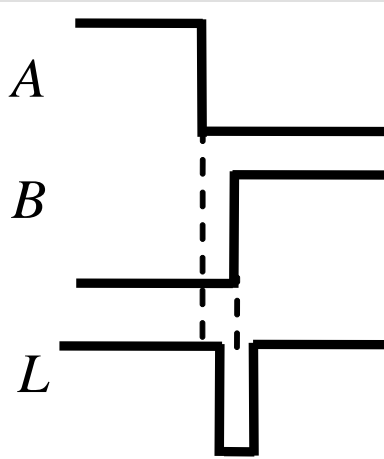
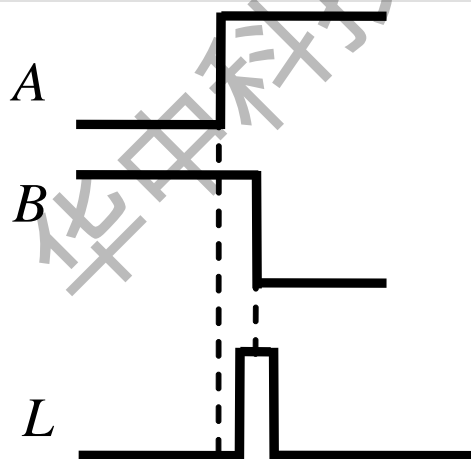
$$L = AB = 0$$

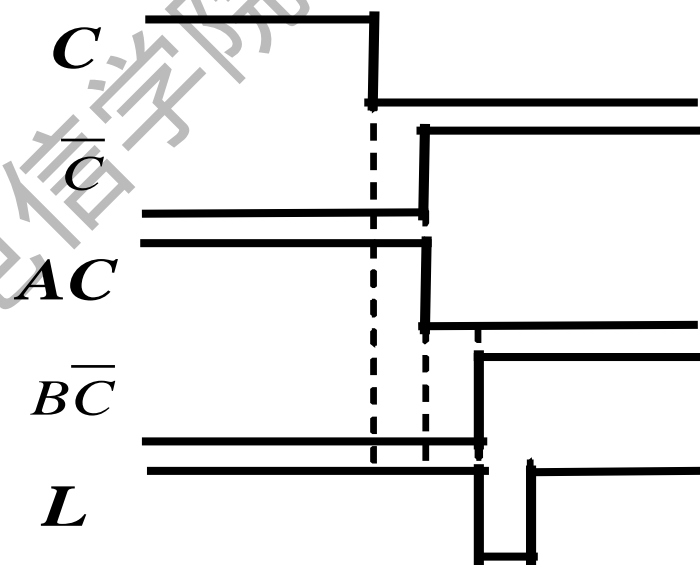
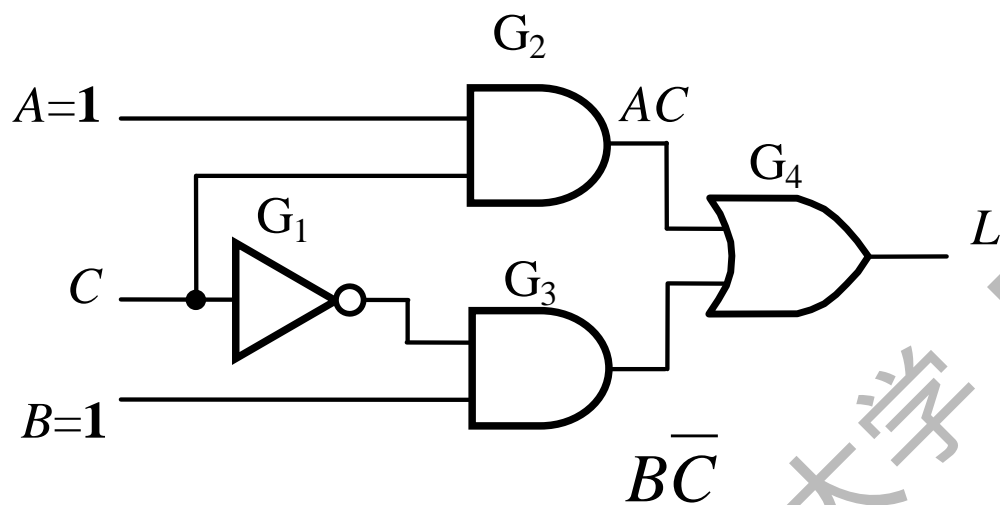


$$L = A + B = 1$$



考虑门的延时时间，且用非门实现 $B=\bar{A}$ 时



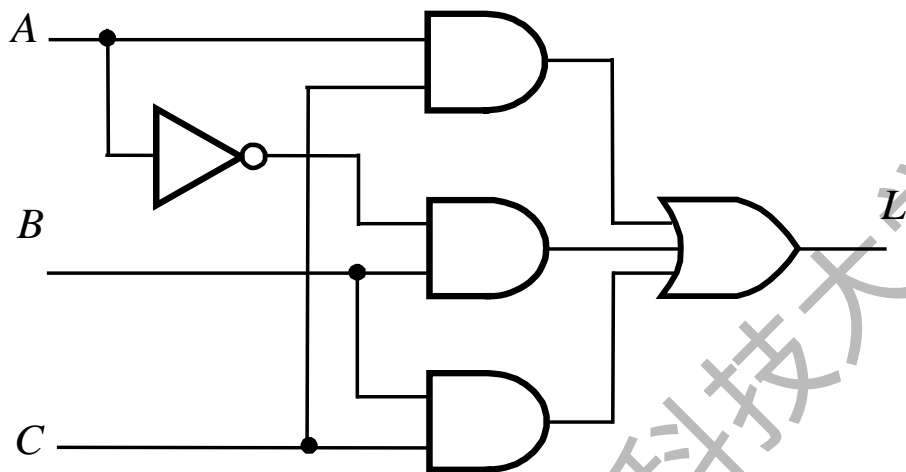


竞争:当一个逻辑门的两个输入端的信号同时向相反方向变化, 而变化的时间有差异的现象。

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

4.3.2 消去竞争冒险的方法

1. 发现并消除互补变量



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

$B = C = 0$ 时

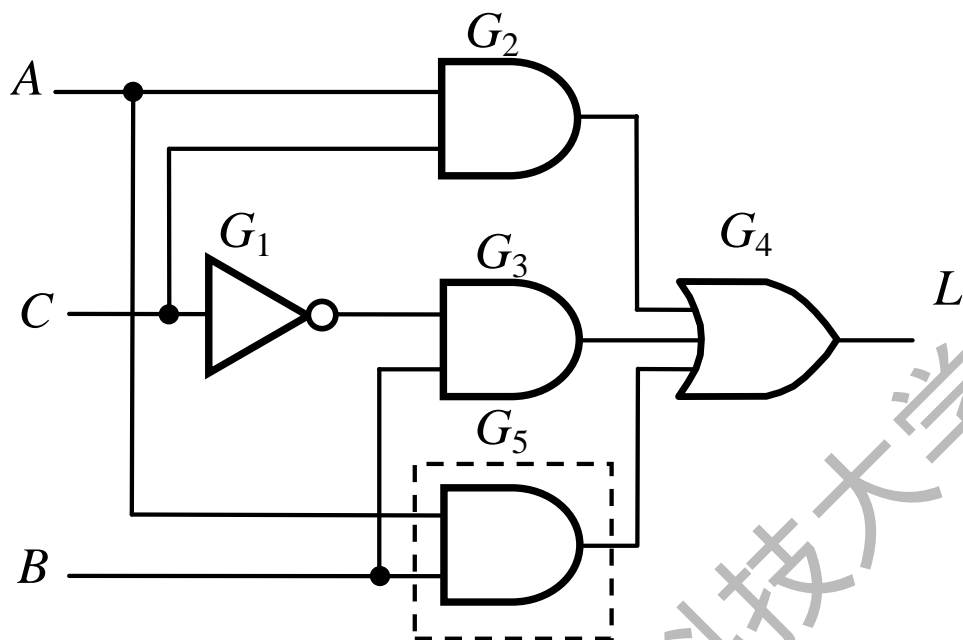
$$L = A\bar{A}$$

可能出现竞争冒险。

为消掉 $A\bar{A}$ ，变换逻辑函数式为

$$L = AC + \bar{A}B + BC$$

2. 增加乘积项, 避免互补项相加



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

当 $A=B=1$ 时

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

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

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

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

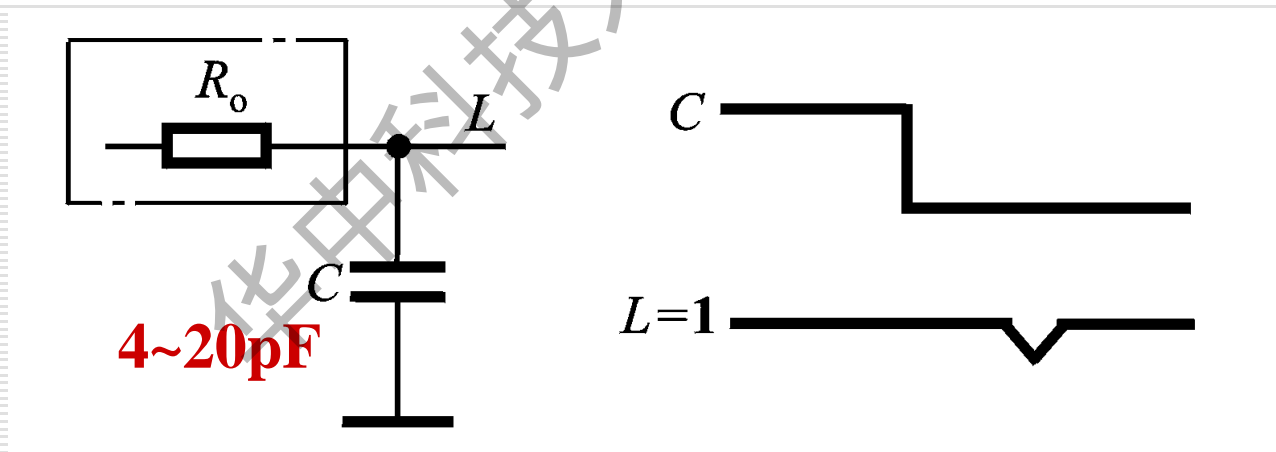
AB

当 $A=B=1$ 时, 根据逻辑表达式有

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

3. 输出端并联电容器

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



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

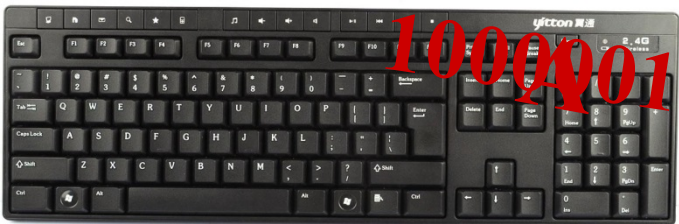
4.4.4 数值比较器

4.4.5 算术运算电路

生活中的编码器



问题：按键信息如何传输？



信道中传递的是**电平信号**

所以在发送端要将按键的信息**转换**成对应的一串高低电平信号，并在信道中传递

4.4.1 编码器

1、编码器 (Encoder)的定义

编码：赋予二进制代码特定含义的过程称为编码。

如：8421BCD码中，用1000表示数字8

如：ASCII码中，用1000001表示字母A等

编码器：具有编码功能的逻辑电路。

4.4.1 编码器

编码器的逻辑功能：

能将每一个编码**输入信号**变换为不同的**二进制的代码**输出。

如BCD编码器：将10个编码输入信号分别编成10个4位码输出。

如8线-3线编码器：将8个输入的信号分别编成 8个3位二进制数码输出。

4.4.1 编码器

1、编码器 (Encoder)的分类

编码器的分类：普通编码器和优先编码器。

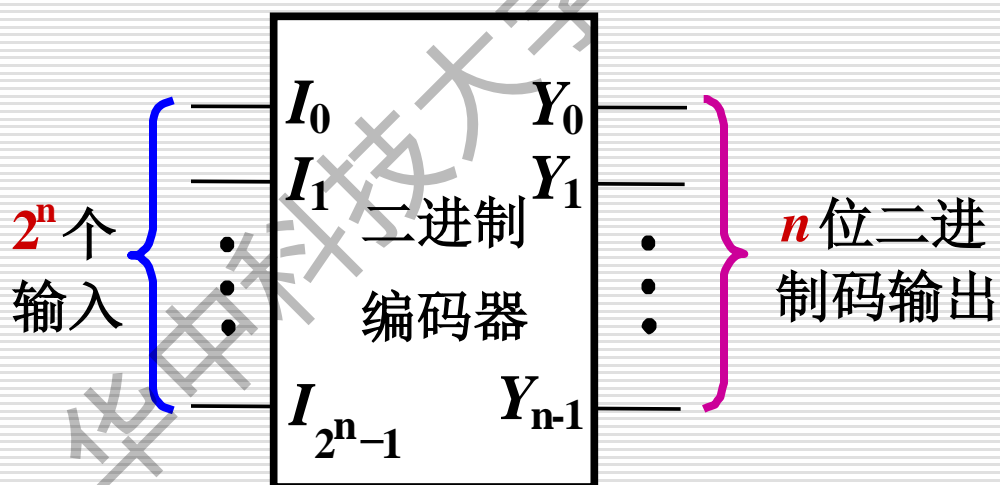
普通编码器：任何时候只允许输入一个有效编码信号，否则输出就会发生混乱。

优先编码器：允许同时输入两个以上的有效编码信号。当同时输入几个有效编码信号时，优先编码器能按预先设定的优先级别，只对其中优先权最高的一个进行编码。

2、编码器的工作原理

普通二进制编码器

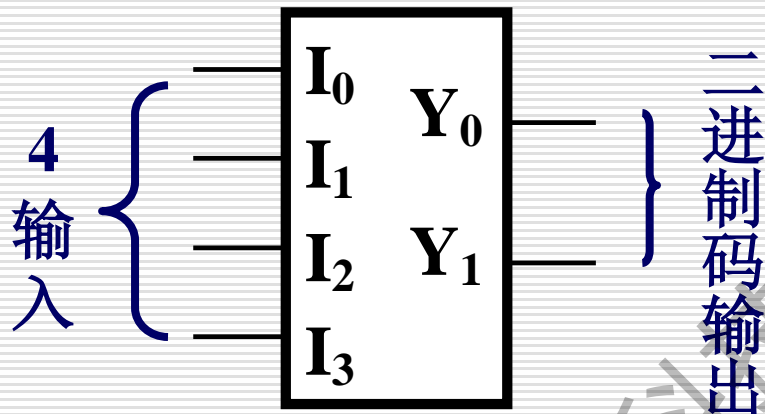
二进制编码器的结构框图



2、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



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

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

(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。

该表达式是否可以再简化？

(3) 4线—2线编码器真值表

N	I ₃	I ₂	I ₁	I ₀	Y ₁	Y ₀
0	0	0	0	0	0	0
1	0	0	0	1	0	0
2	0	0	1	0	0	1
3	0	0	1	1	0	0
4	0	1	0	0	1	0
5	0	1	0	1	0	0
6	0	1	1	0	0	0
7	0	1	1	1	0	0
8	1	0	0	0	1	1
9	1	0	0	1	0	0
10	1	0	1	0	0	0
11	1	0	1	1	0	0
12	1	1	0	0	0	0
13	1	1	0	1	0	0
14	1	1	1	0	0	0
15	1	1	1	1	0	0

$$Y_1 = I_2 + I_3$$

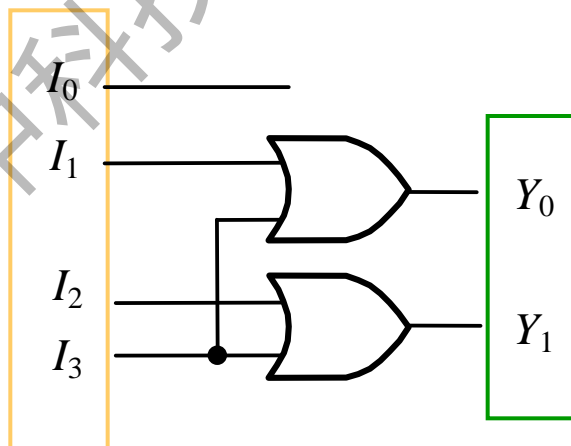
$$Y_0 = I_1 + I_3$$

$I_1 I_0$		$I_3 I_2$			
		00	01	11	10
		00	01	11	10
00	00	×	0	×	0
01	01	1	×	×	×
11	11	×	×	×	×
10	10	1	×	×	×

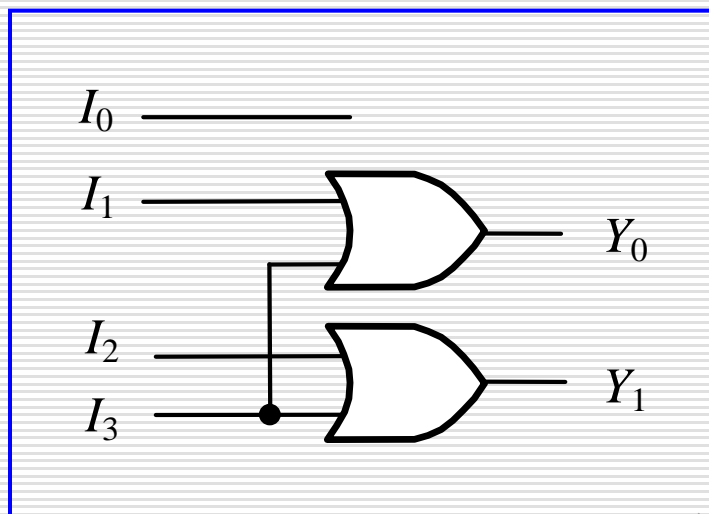
$I_1 I_0$		$I_3 I_2$			
		00	01	11	10
		00	01	11	10
00	00	×	0	×	1
01	01	0	×	×	×
11	11	×	×	×	×
10	10	1	×	×	×

Y_1

Y_0



若有2个以上的输入为有效信号？



当只有 I_3 为1时，

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

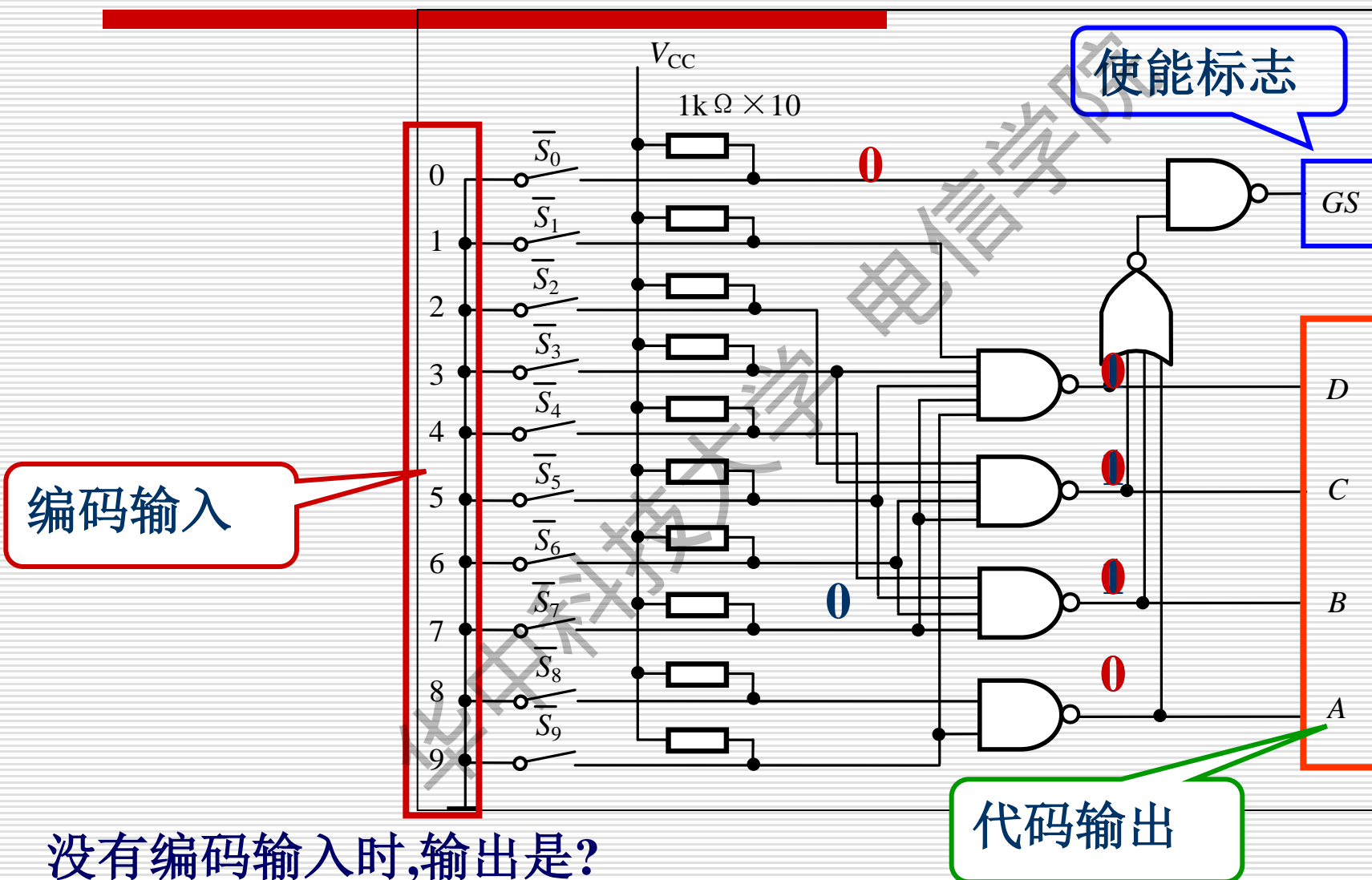
$I_1 = I_2 = 1$, $I_0 = I_3 = 0$ 时，

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

无法输出有效编码。

结论：普通编码器不能同时输入两个以上的有效编码信号

(2.) 键盘输入8421BCD码编码器（分析）



没有编码输入时,输出是?

使能标志有什么作用?

2. 键盘输入8421BCD码编码器功能表

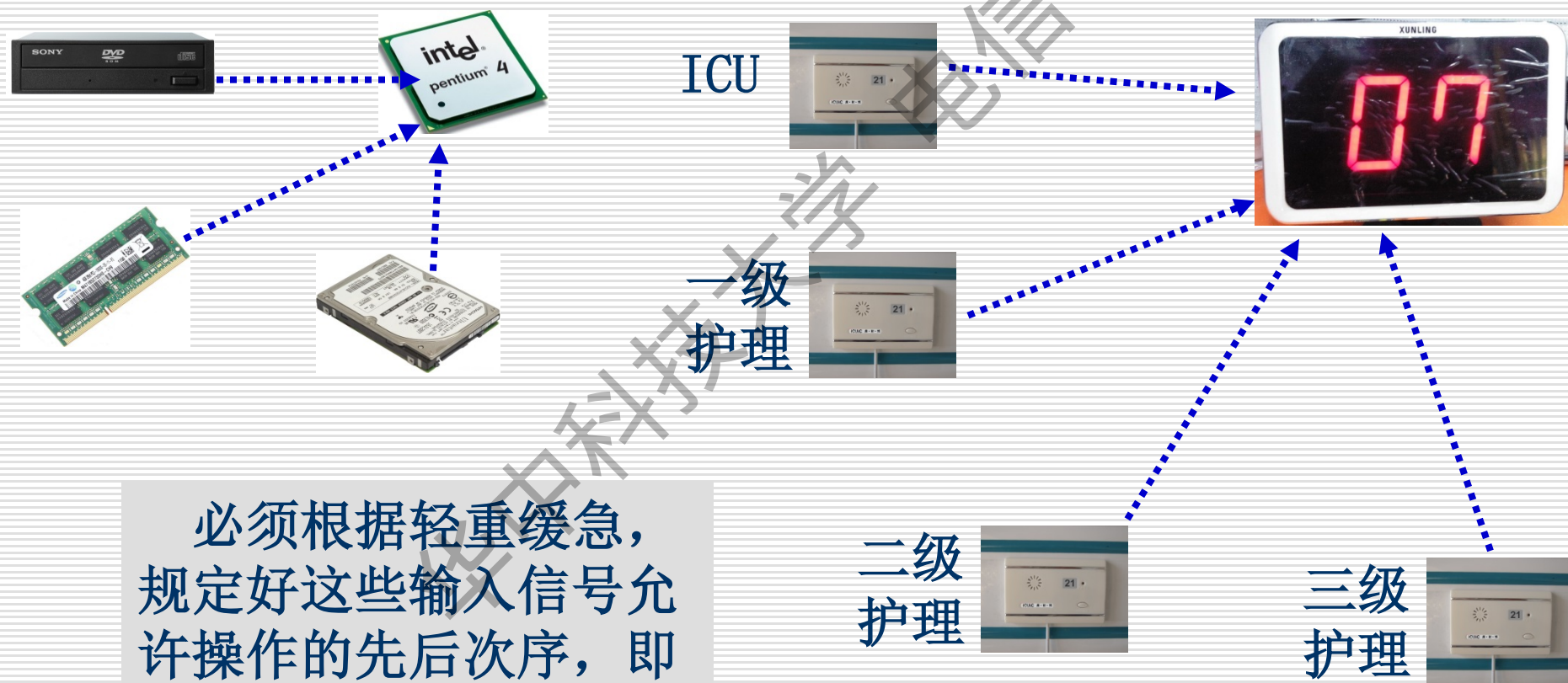
输 入										输 出				
$\overline{S_0}$	$\overline{S_1}$	$\overline{S_2}$	$\overline{S_3}$	$\overline{S_4}$	$\overline{S_5}$	$\overline{S_6}$	$\overline{S_7}$	$\overline{S_8}$	$\overline{S_9}$	A	B	C	D	GS
1	1	1	1	1	1	1	1	1	1	0	0	0	0	0
1	1	1	1	1	1	1	1	1	0	1	0	0	1	1
1	1	1	1	1	1	1	1	0	1	1	0	0	0	1
1	1	1	1	1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

该编码器为输入低电平有效，输出高电平有效，**GS**为标志位。

3. 优先编码器：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

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，规定好这些输入信号允许操作的先后次序，即优先级别。

(2) 优先编码器 (4线—2 线优先编码器) (设计)

输入为编码信号 $I_3 \sim I_0$ 输出为 $Y_1 Y_0$

输入编码信号高电平有效, 输出为二进制代码

(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	1	×	×	0	1
0	0	1	×	1	0
0	0	0	1	1	1

高

低

I_0 : ICU病人呼叫

I_1 : 一级护理病人呼叫

I_2 : 二级护理病人呼叫

I_3 : 三级护理病人呼叫

输入编码信号优先级
从高到低为 $I_0 \sim I_3$

(2) 优先编码器 (4线—2 线优先编码器) (设计)

(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	1	×	×	0	1
0	0	1	×	1	0
0	0	0	1	1	1

(2) 写出逻辑表达式

$$Y_0 = I_3 \cdot \overline{I_2} \cdot \overline{I_1} \cdot \overline{I_0} + I_1 \cdot \overline{I_0}$$

$$Y_1 = I_3 \cdot \overline{I_2} \cdot \overline{I_1} \cdot \overline{I_0} + I_2 \cdot \overline{I_1} \cdot \overline{I_0}$$

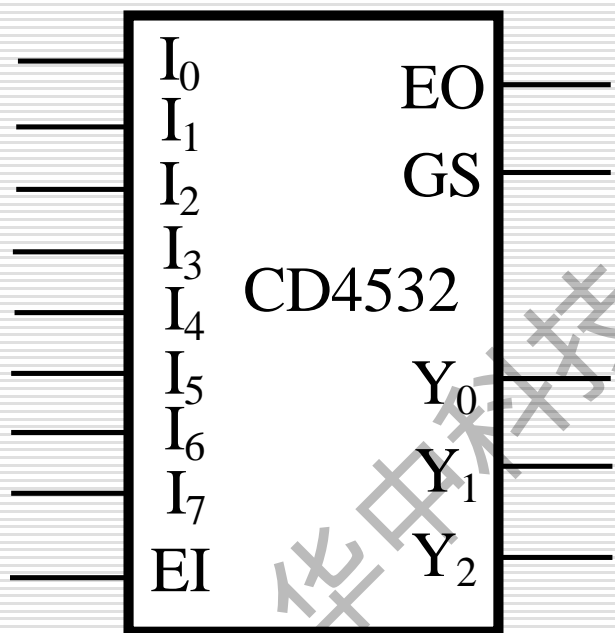
(3) 画出逻辑电路 (略)

思考题:

- 1、列出这个例题的完整真值表，并利用卡诺图完成最简与或式的设计
- 2、思考用无关项参与设计的好处？

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

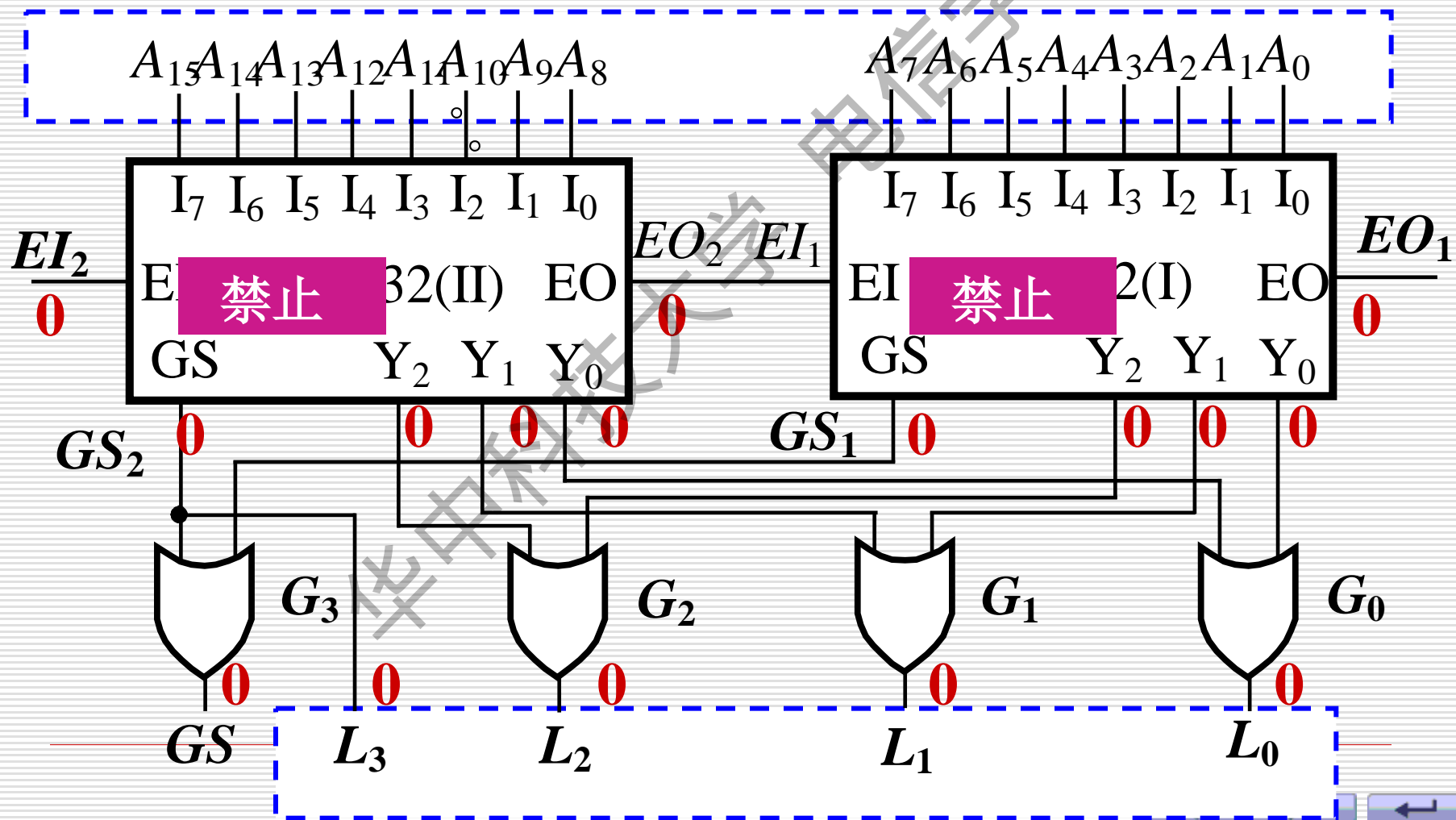
$EI=1$ ，电路工作，输入 $I_0 \sim I_7$ 分别有高电平输入时， $Y_2 \sim Y_0$ 为 $I_0 \sim I_7$ 的编码输出， $GS = 1$ ， $EO = 0$ 。优先级 $I_7 \sim I_0$

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

为什么要添加GS、EO输出信号？

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。

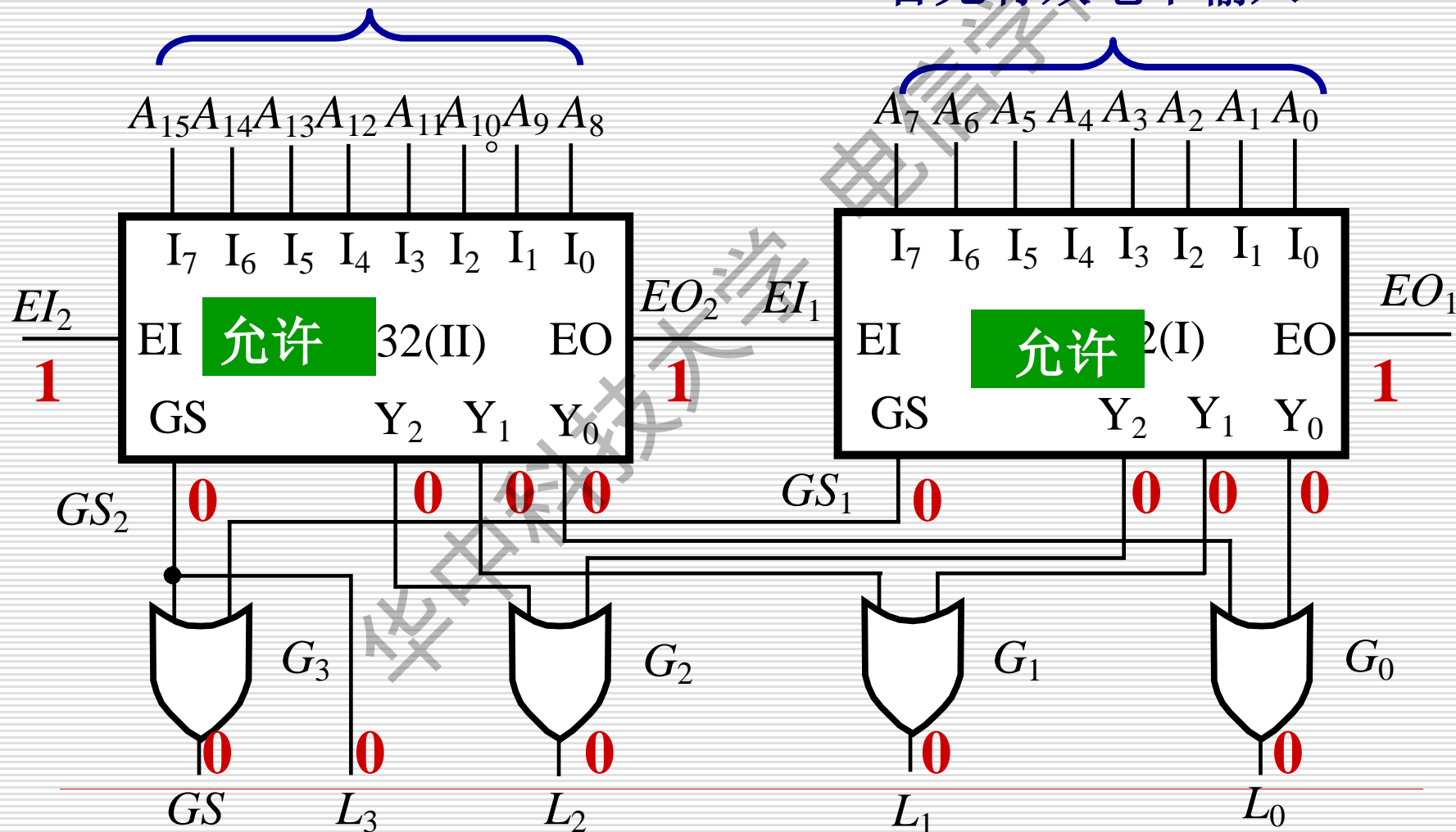
$EI_2=0$, 电路不工作, $GS = EO_1=0$,
 $L_3 L_2 L_1 L_0=0000$ 无编码输出



$EI_2=1$, 电路工作, 无有效高电平输入,
 $L_3 L_2 L_1 L_0 = 0000$, $GS = 0$, $EO_1=1$;

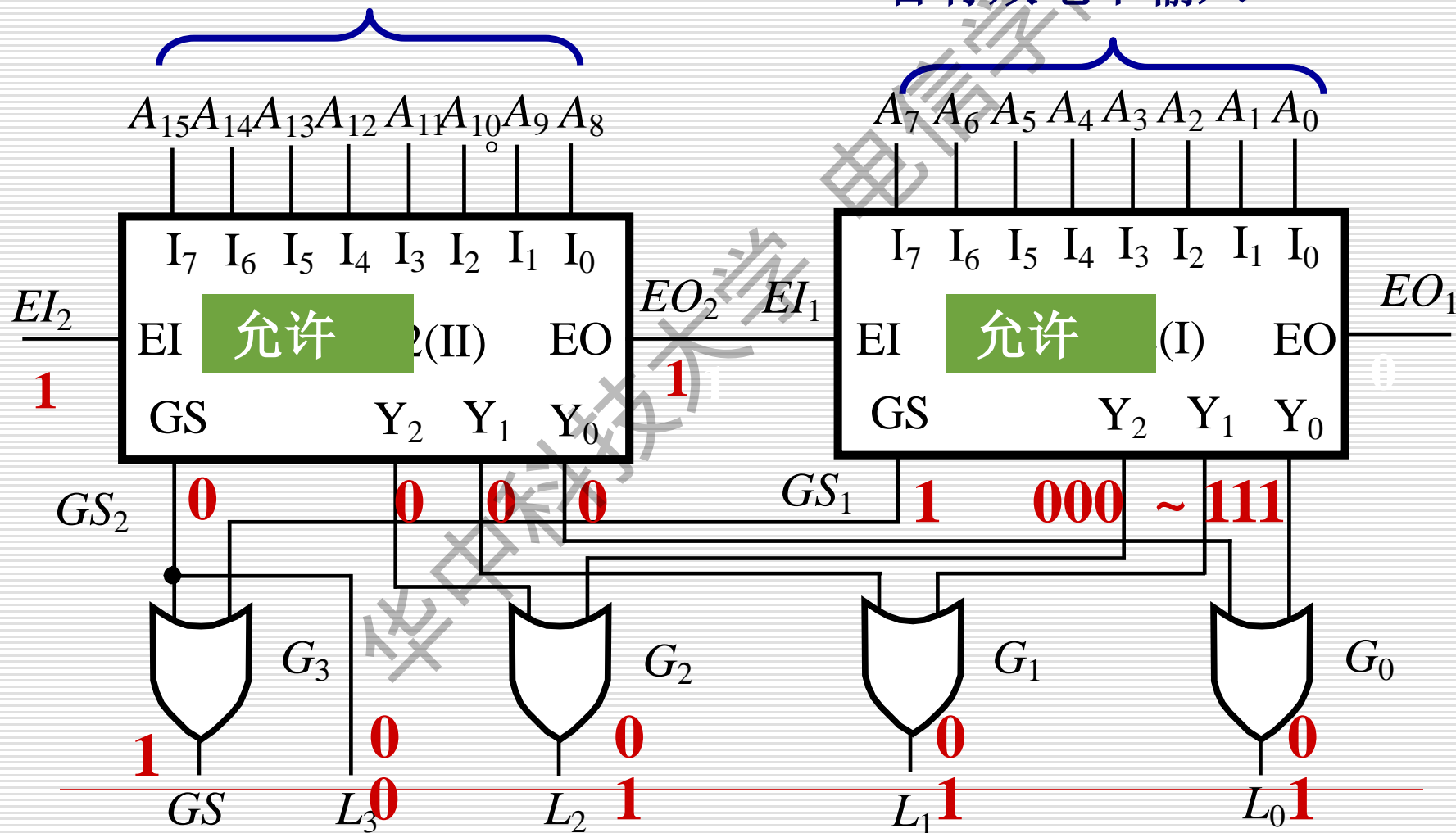
若无有效电平输入

若无有效电平输入



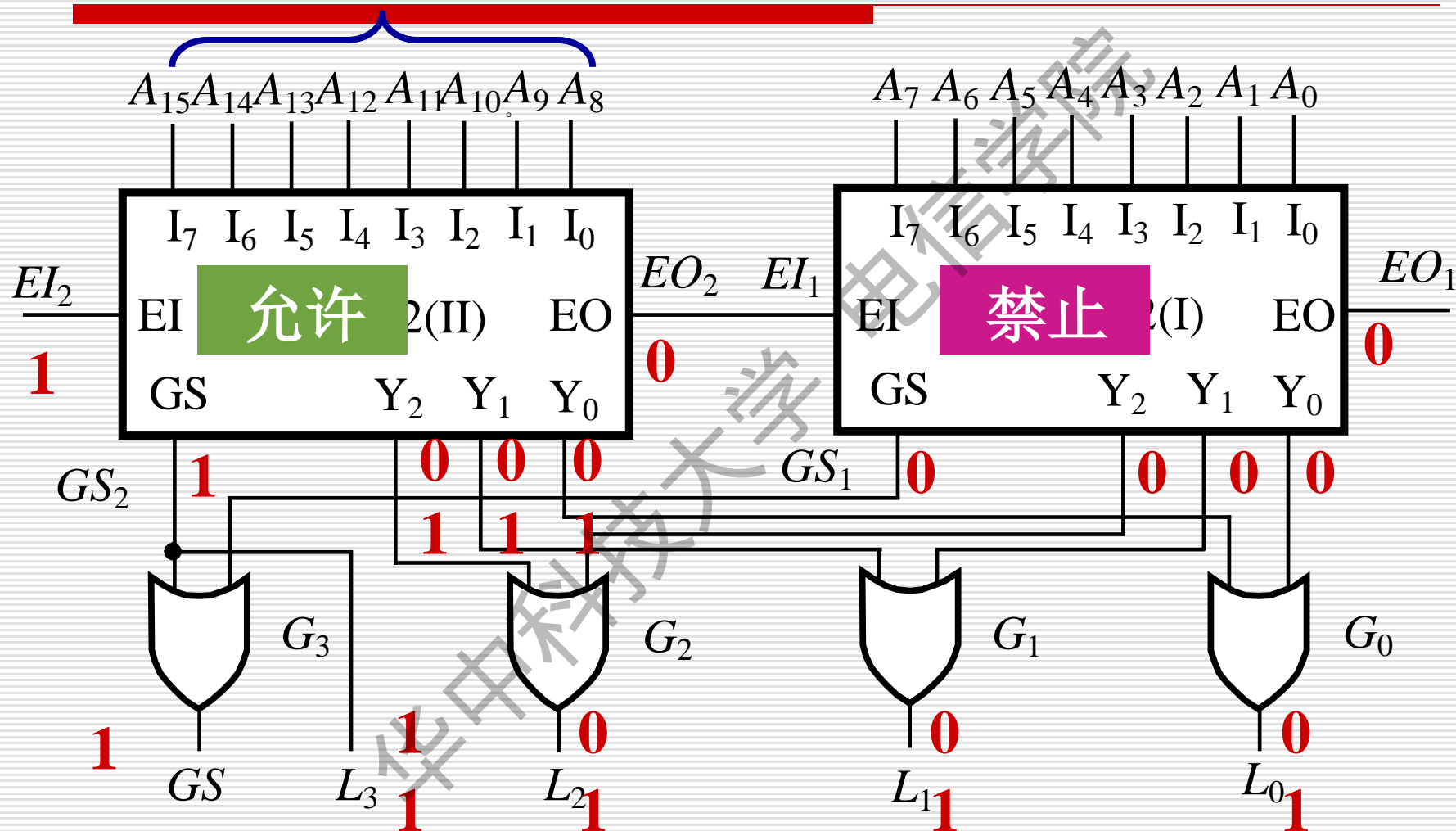
若无有效电平输入

若有效电平输入



若有效电平输入

哪块芯片的优先级高？



4.4.2 译码器/数据分配器

1 译码器的定义与分类

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

译码器：具有译码功能的逻辑电路称为译码器。

译码器的分类：

唯一地址译码器

将一系列代码转换成与之一一对应的有效信号。

常见的唯一地址译码器：

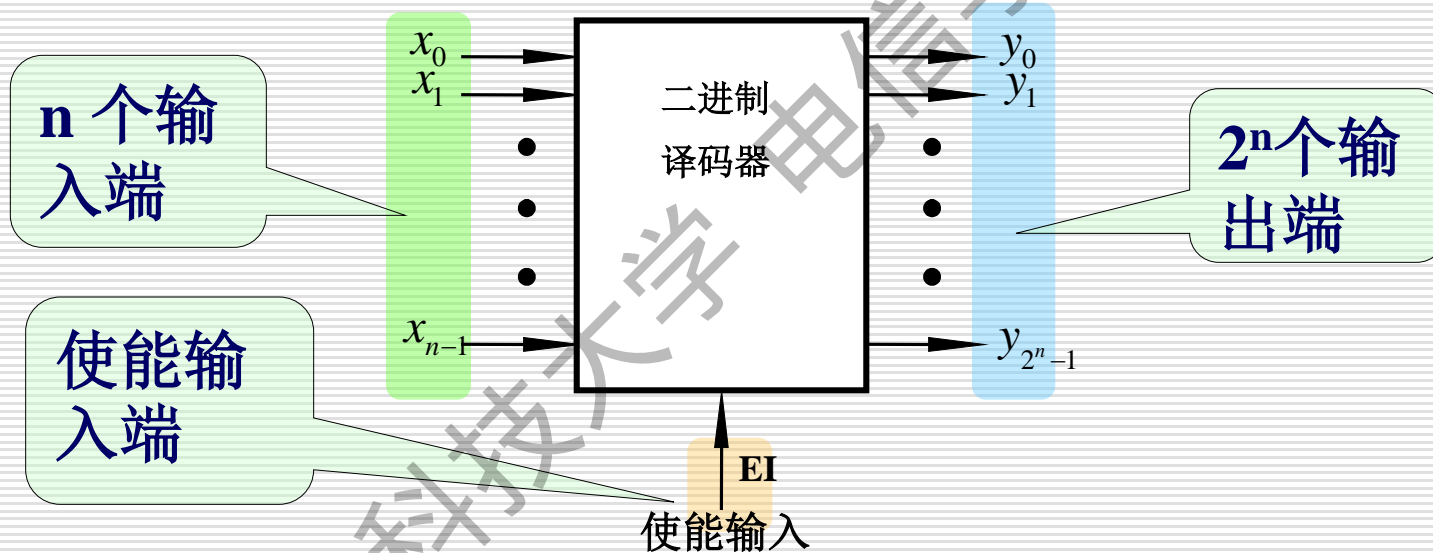
- 二进制译码器
- 二—十进制译码器
- 显示译码器

代码变换器

将一种代码转换成另一种代码。

2. 典型译码器电路及应用

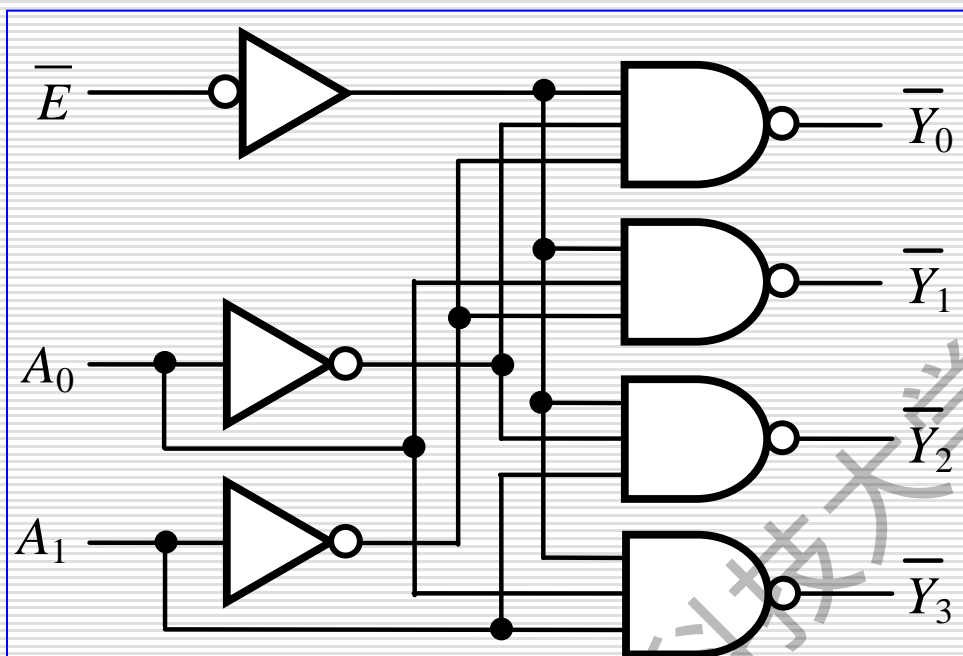
(1) 二进制译码器



设输入端的个数为 n ，输出端的个数为 M
则有

$$M = 2^n$$

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

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

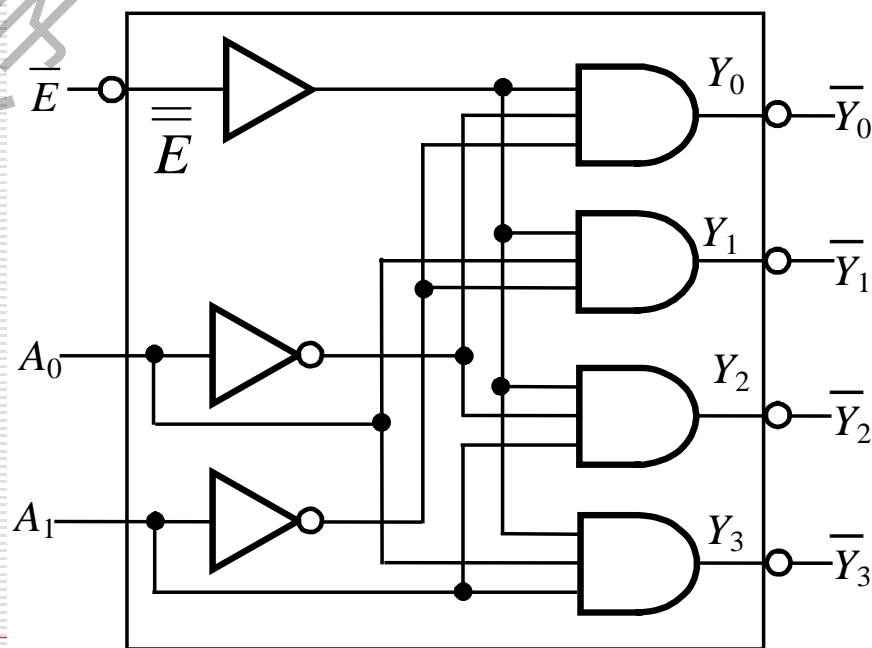
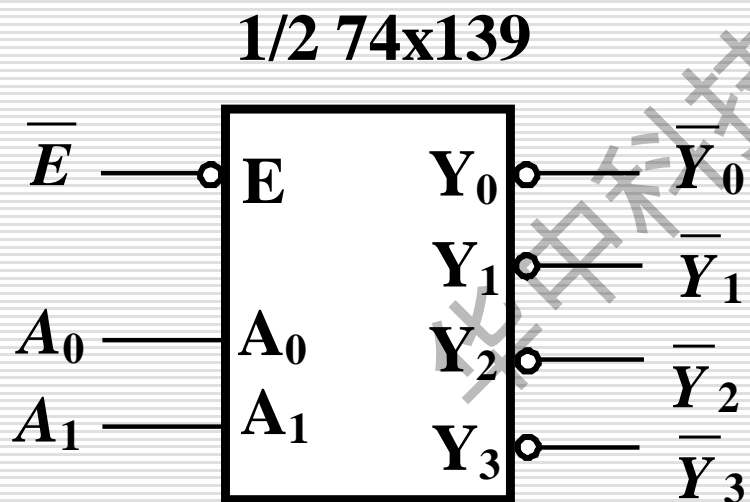
$$\overline{Y_2} = \overline{\overline{\overline{E} A_1 A_0}}$$

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

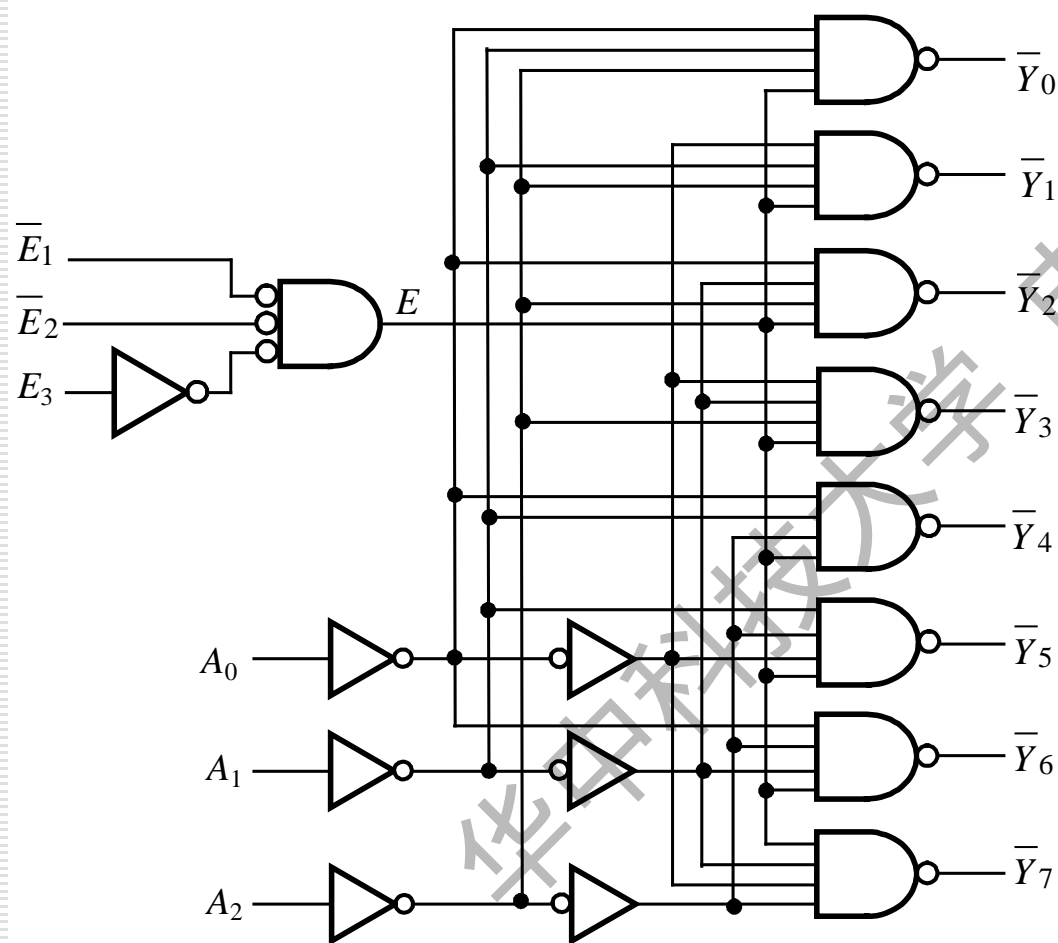
$$\overline{Y_3} = \overline{\overline{\overline{E} A_1 A_0}}$$

(a) 2线-4线译码器 (74HC139) ——逻辑符号说明

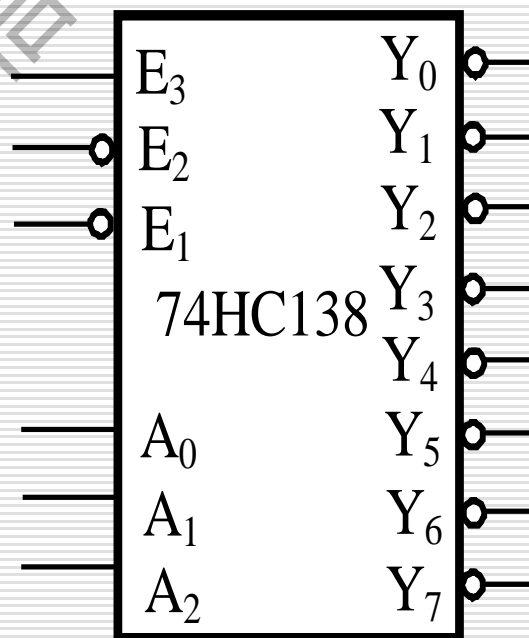
逻辑符号框外部的符号，表示外部输入或输出信号名称，字母上面的“—”号说明该输入或输出是低电平有效。符号框内部的输入、输出变量表示其内部的逻辑关系。



(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

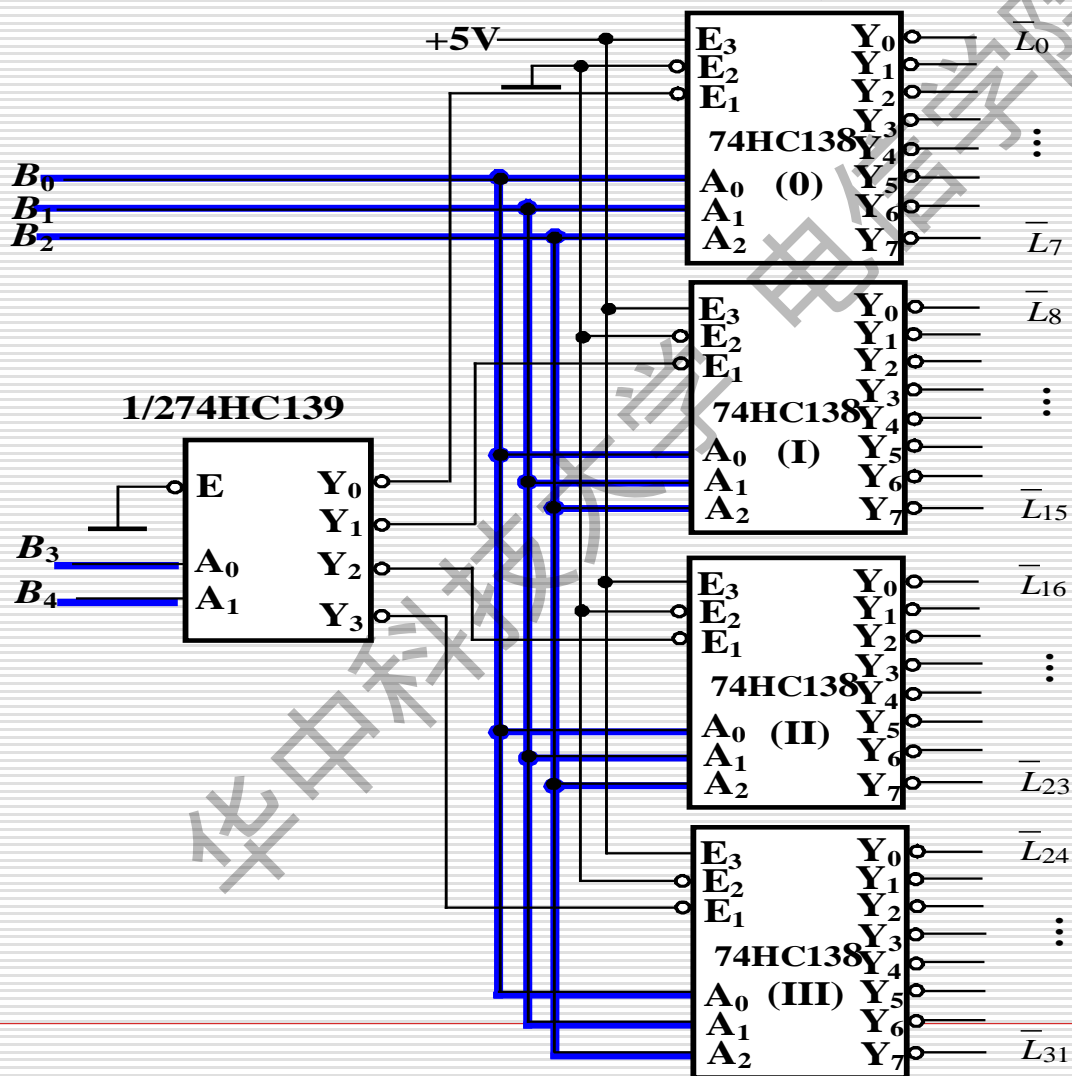
$$\bar{Y}_0 = \bar{A}_2 \cdot \bar{A}_1 \cdot \bar{A}_0 \quad \bar{Y}_1 = \bar{A}_2 \cdot \bar{A}_1 \cdot A_0 \quad \bar{Y}_2 = \bar{A}_2 \cdot A_1 \cdot \bar{A}_0 \quad \bar{Y}_3 = \bar{A}_2 \cdot A_1 \cdot A_0$$

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

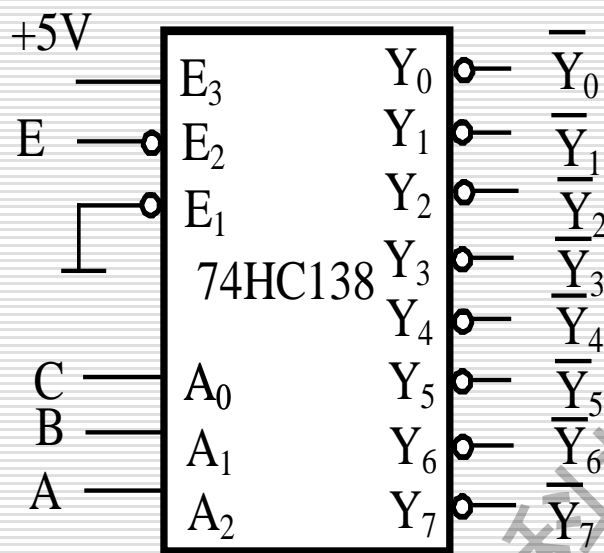
输 入						输 出							
E_3	\bar{E}_2	\bar{E}_1	A_2	A_1	A_0	\bar{Y}_0	\bar{Y}_1	\bar{Y}_2	\bar{Y}_3	\bar{Y}_4	\bar{Y}_5	\bar{Y}_6	\bar{Y}_7
×	1	×	×	×	×	1	1	1	1	1	1	1	1
×	X	1	×	×	×	1	1	1	1	1	1	1	1
0	×	×	×	×	×	1	1	1	1	1	1	1	1
1	0	0	0	0	0	0	1	1	1	1	1	1	1
1	0	0	0	0	1	1	0	1	1	1	1	1	1
1	0	0	0	1	0	1	1	0	1	1	1	1	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

1、译码器的扩展

用74X139和74X138构成5线-32线译码器



2、用译码器实现逻辑函数。当 $E_3=1$ ， $E_2=E_1=0$ 时



$$\overline{Y}_0 = \overline{A_2 \cdot A_1 \cdot A_0} = \overline{m_0}$$

$$\overline{Y}_1 = \overline{A_2 \cdot A_1 \cdot A_0} = \overline{m_1}$$

$$\overline{Y}_2 = \overline{A_2 \cdot A_1 \cdot A_0} = \overline{m_2}$$

⋮

⋮

⋮

$$\overline{Y}_7 = \overline{A \cdot B \cdot C} = \overline{m_7}$$

3线-8线译码器的 $Y_0 \sim Y_7$ 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 $L = \overline{A}\overline{C} + AB$

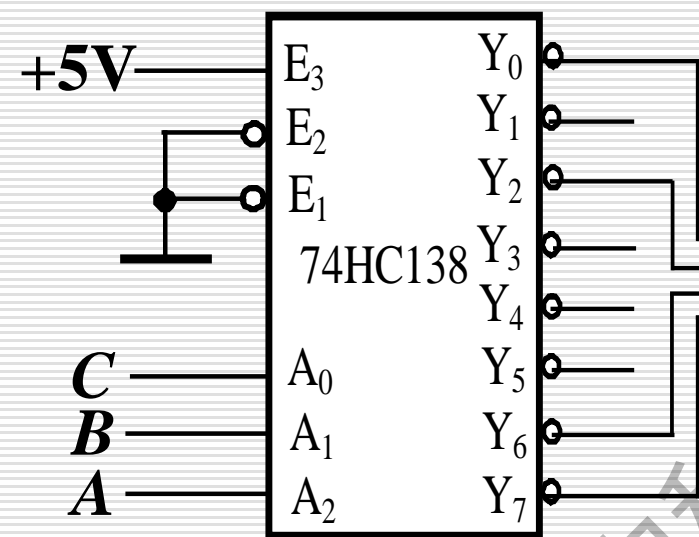
首先将函数式变换为最小项之和的形式

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

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

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

$$= \overline{Y_0 \cdot Y_2 \cdot Y_6 \cdot Y_7}$$



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.

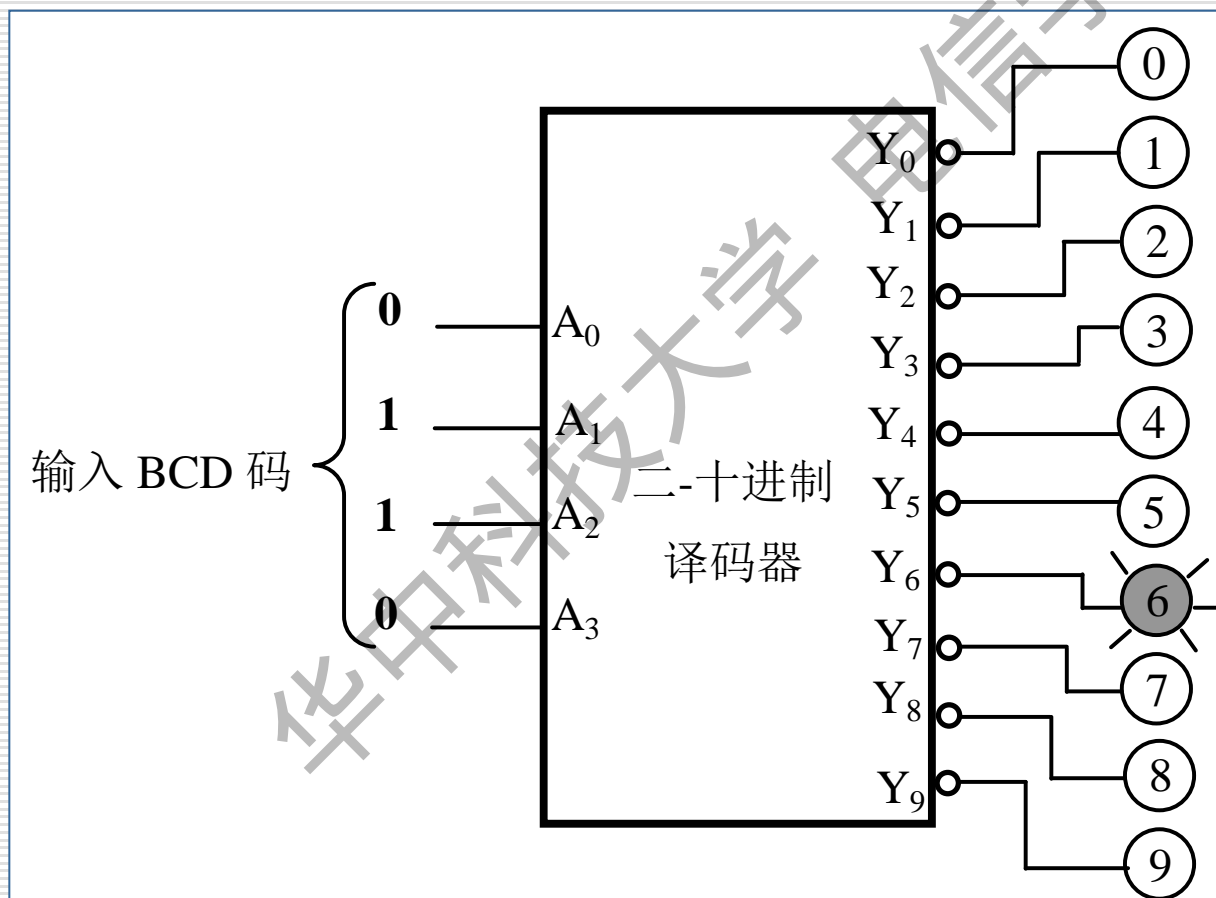
(2) 二-十进制译码器的真值表

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

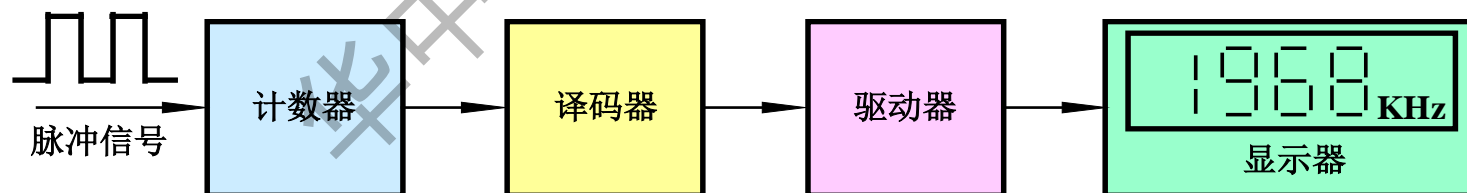
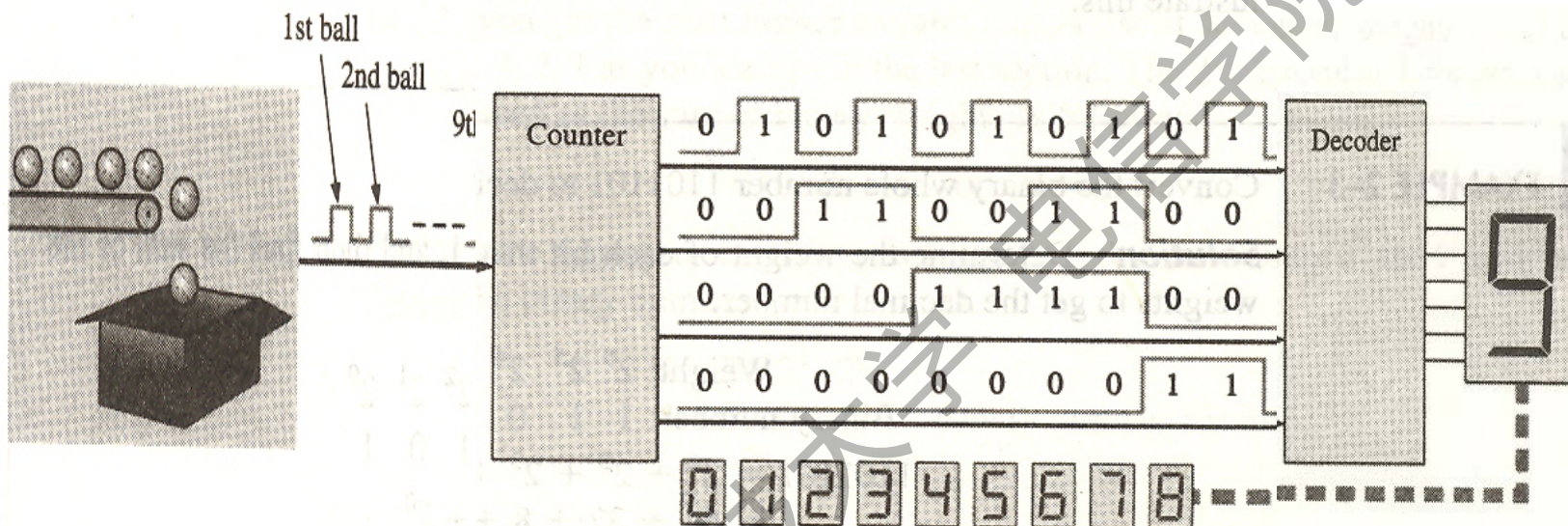
0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
1	0	0	0	1	1	0	1	1	1	1	1	1	1	1
2	0	0	1	0	1	1	0	1	1	1	1	1	1	1
3	0	0	1	1	1	1	1	0	1	1	1	1	1	1
4	0	1	0	0	1	1	1	1	0	1	1	1	1	1
5	0	1	0	1	1	1	1	1	1	0	1	1	1	1
6	0	1	1	0	1	1	1	1	1	1	0	1	1	1
7	0	1	1	1	1	1	1	1	1	1	1	0	1	1
8	1	0	0	0	1	1	1	1	1	1	1	1	0	1
9	1	0	0	1	1	1	1	1	1	1	1	1	1	0

二-十进制译码器

功能：将8421BCD码译成为10个状态输出。

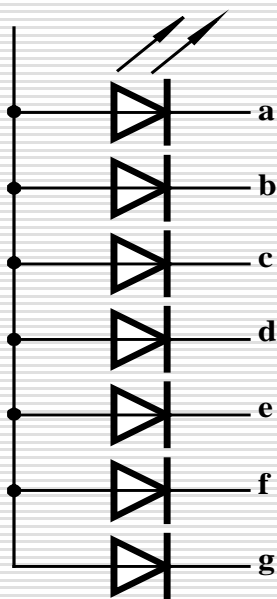


(3) 显示译码器

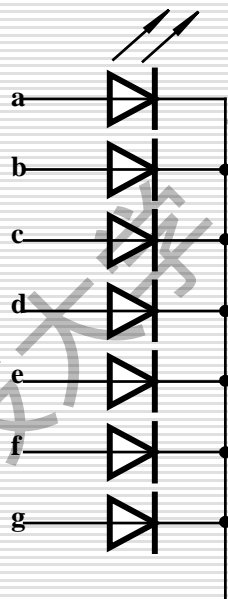


1. 七段显示译码器

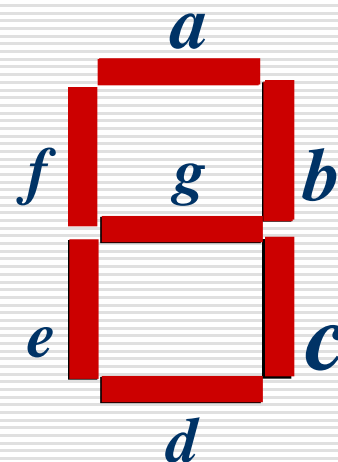
(1) 最常用的显示器有：半导体发光二极管和液晶显示器。



共阳极显示器



共阴极显示器

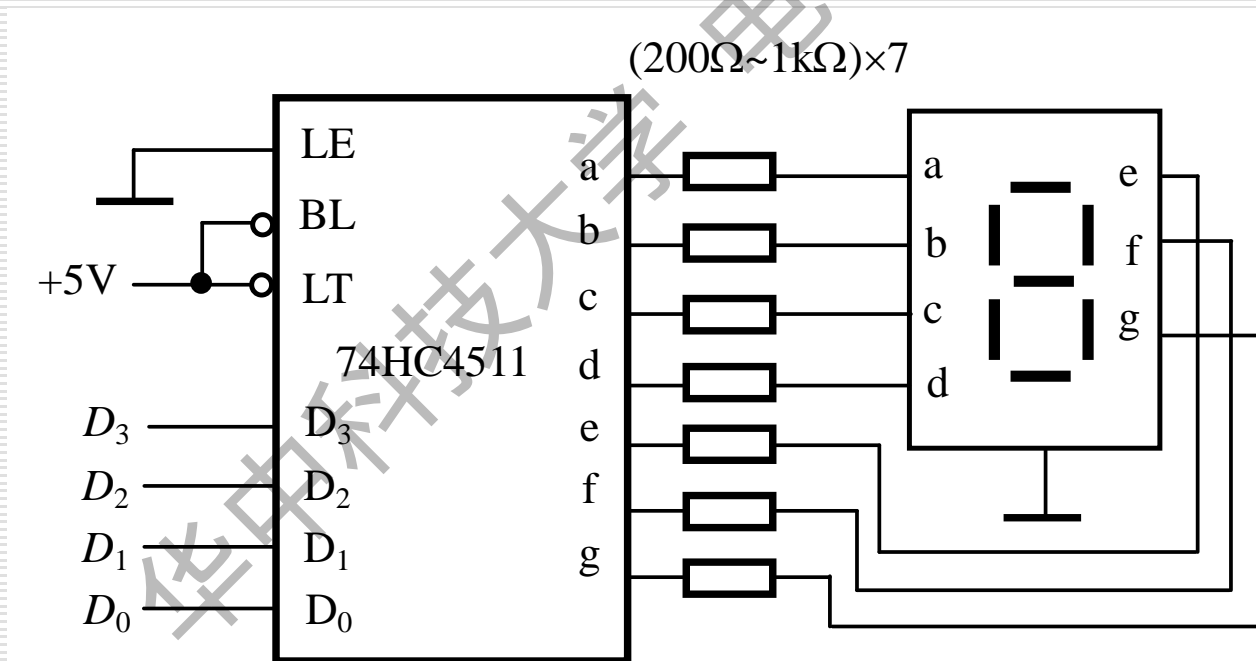


显示器分段布局图

常用的集成七段显示译码器

-----CMOS七段显示译码器74HC4511

显示译码器与显示器的连接方式



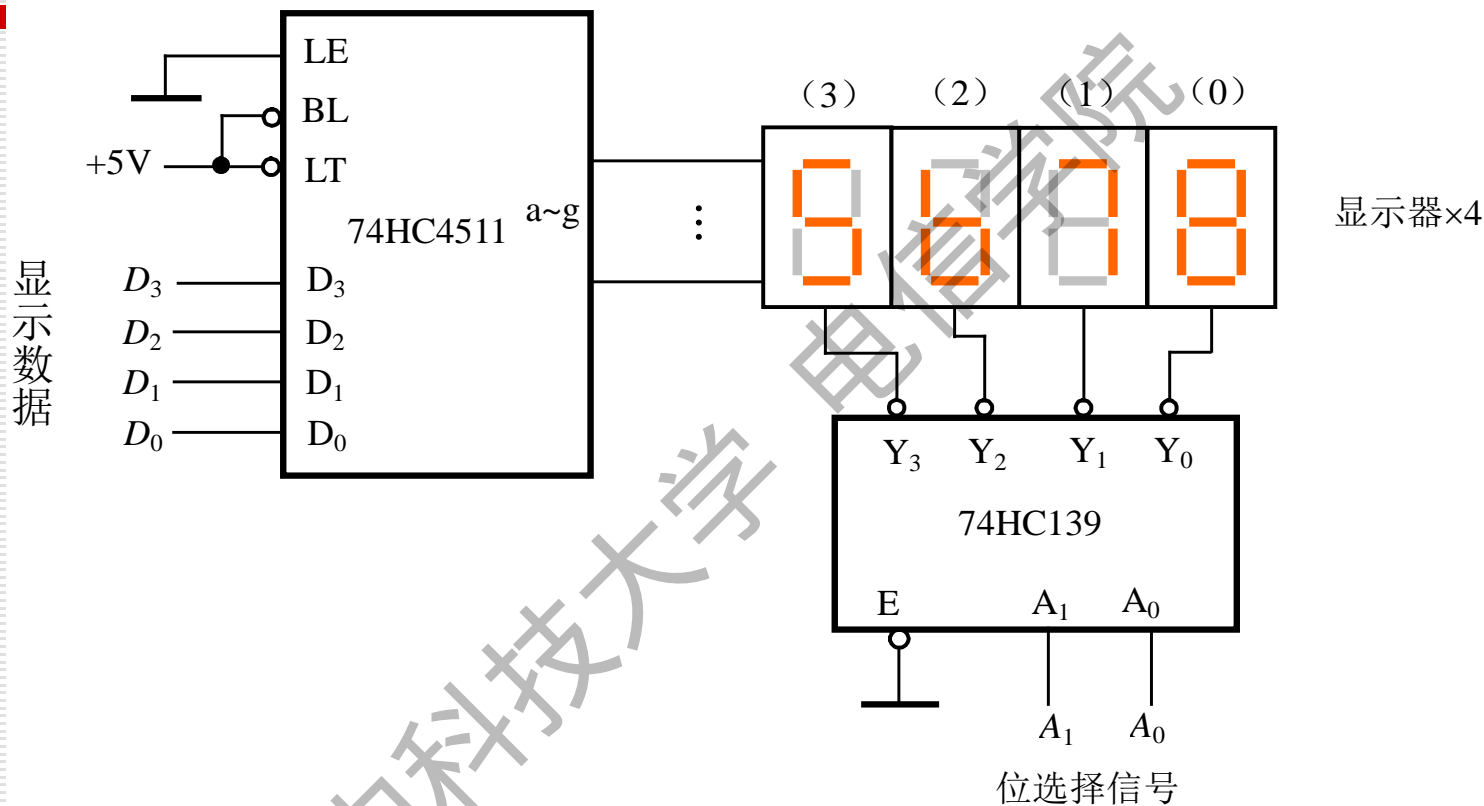
CMOS七段显示译码器74HC4511功能表

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

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
	<i>LE</i>	<i>BL</i>	<i>LT</i>	<i>D</i> ₃	<i>D</i> ₂	<i>D</i> ₁	<i>D</i> ₀	<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>	<i>e</i>	<i>f</i>	<i>g</i>	
10	0	1	1	1	0	1	0	0	0	0	0	0	0	0	熄灭
11	0	1	1	1	0	1	1	0	0	0	0	0	0	0	熄灭
12	0	1	1	1	1	0	0	0	0	0	0	0	0	0	熄灭
13	0	1	1	1	1	0	1	0	0	0	0	0	0	0	熄灭
14	0	1	1	1	1	1	0	0	0	0	0	0	0	0	熄灭
15	0	1	1	1	1	1	1	0	0	0	0	0	0	0	熄灭
灯 测 试	×	×	0	×	×	×	×	1	1	1	1	1	1	1	8
灭 灯	×	0	1	×	×	×	×	0	0	0	0	0	0	0	熄灭
锁 存	1	1	1	×	×	×	×	*							*

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

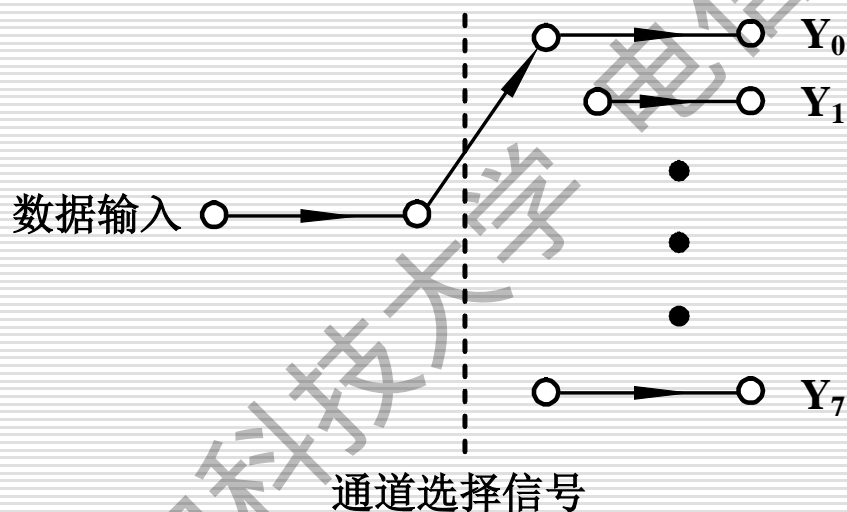


位选择信号 $\overline{A_1}$ 、 $\overline{A_0}$ 控制 $\overline{Y_3} \sim \overline{Y_0}$ 依次产生低电平，使4个显示器轮流显示。要显示的数据组依次送到 $D_3 D_2 D_1 D_0$ 分别在4个显示器上显示。利用人的视觉暂留时间，可以看到稳定的数字。

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

用74HC138组成数据分配器

数据分配器示意图

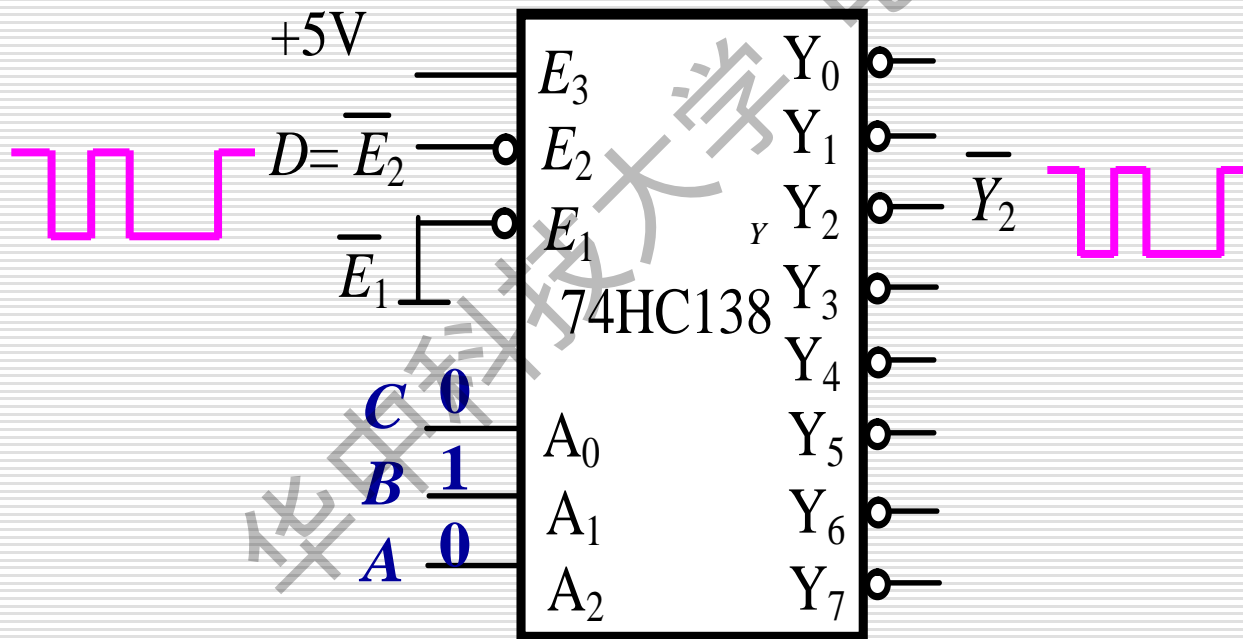


数据分配器：相当于多输出的单刀多掷开关，是将公共数据线上的数据按需要送到不同的通道上去的逻辑电路。

用译码器实现数据分配器

$$\overline{Y_2} = \overline{E_3 D \overline{E_1} \cdot ABC}$$

当 $ABC = 010$ 时, $\overline{Y_2} = D$



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	$\overline{Y_0}$	$\overline{Y_1}$	$\overline{Y_2}$	$\overline{Y_3}$	$\overline{Y_4}$	$\overline{Y_5}$	$\overline{Y_6}$	$\overline{Y_7}$
L	X	L	X	X	X	H	H	H	H	H	H	H	H
H	D	L	L	L	L	D	H	H	H	H	H	H	H
H	D	L	L	L	H	H	D	H	H	H	H	H	H
H	D	L	L	H	L	H	H	D	H	H	H	H	H
H	D	L	L	H	H	H	H	H	D	H	H	H	H
H	D	L	H	L	L	H	H	H	H	D	H	H	H
H	D	L	H	L	H	H	H	H	H	H	D	H	H
H	D	L	H	H	L	H	H	H	H	H	H	D	H
H	D	L	H	H	H	H	H	H	H	H	H	H	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。

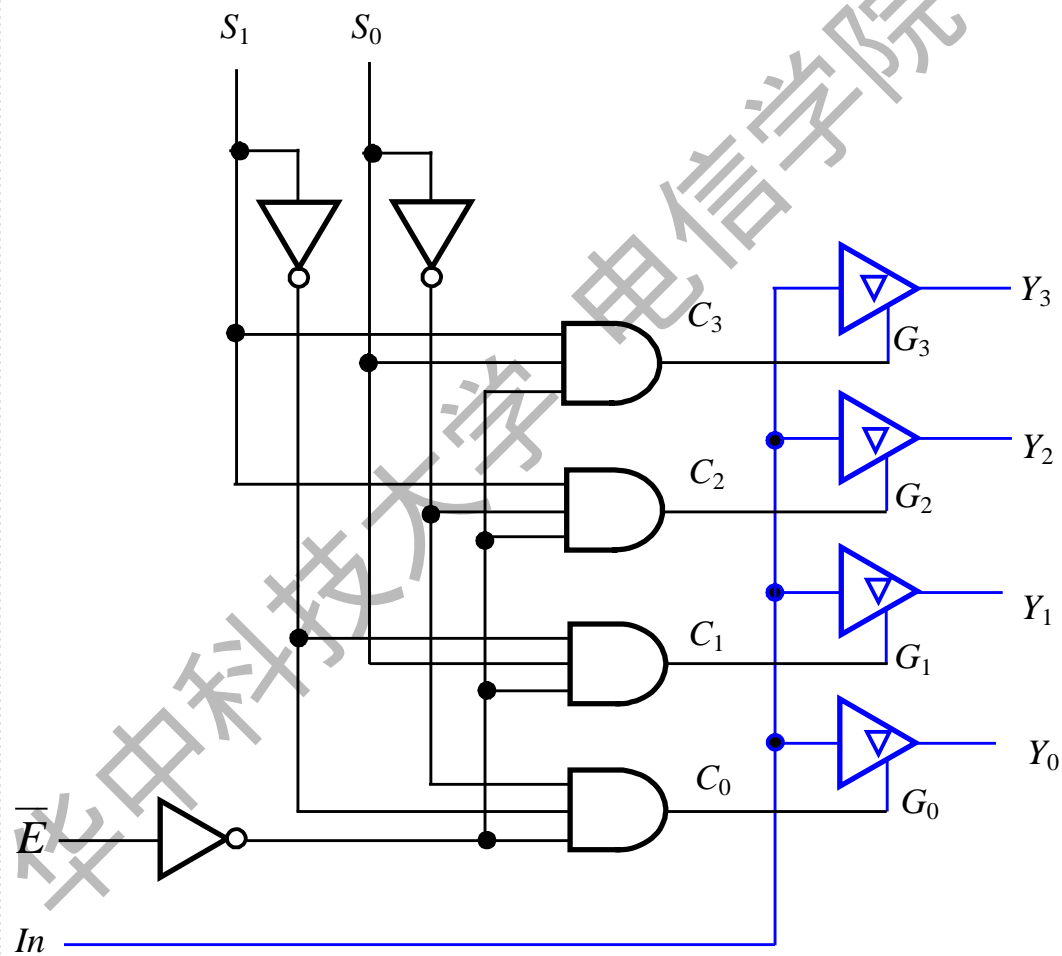
1. 列真值表
输出端有3种状态(0、1、z), 输出级是4个三态门组成。其控制信号由 \overline{E} 、 S_1 、 S_0 共同作用产生。

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

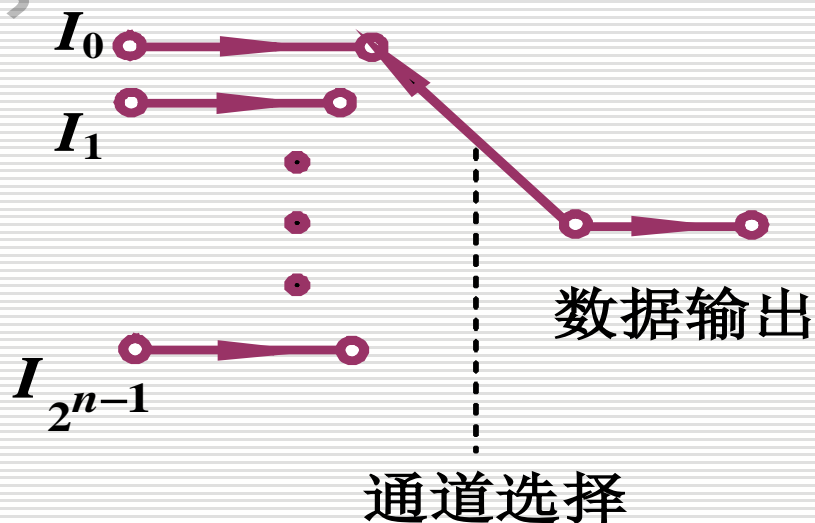


4.4.3 数据选择器

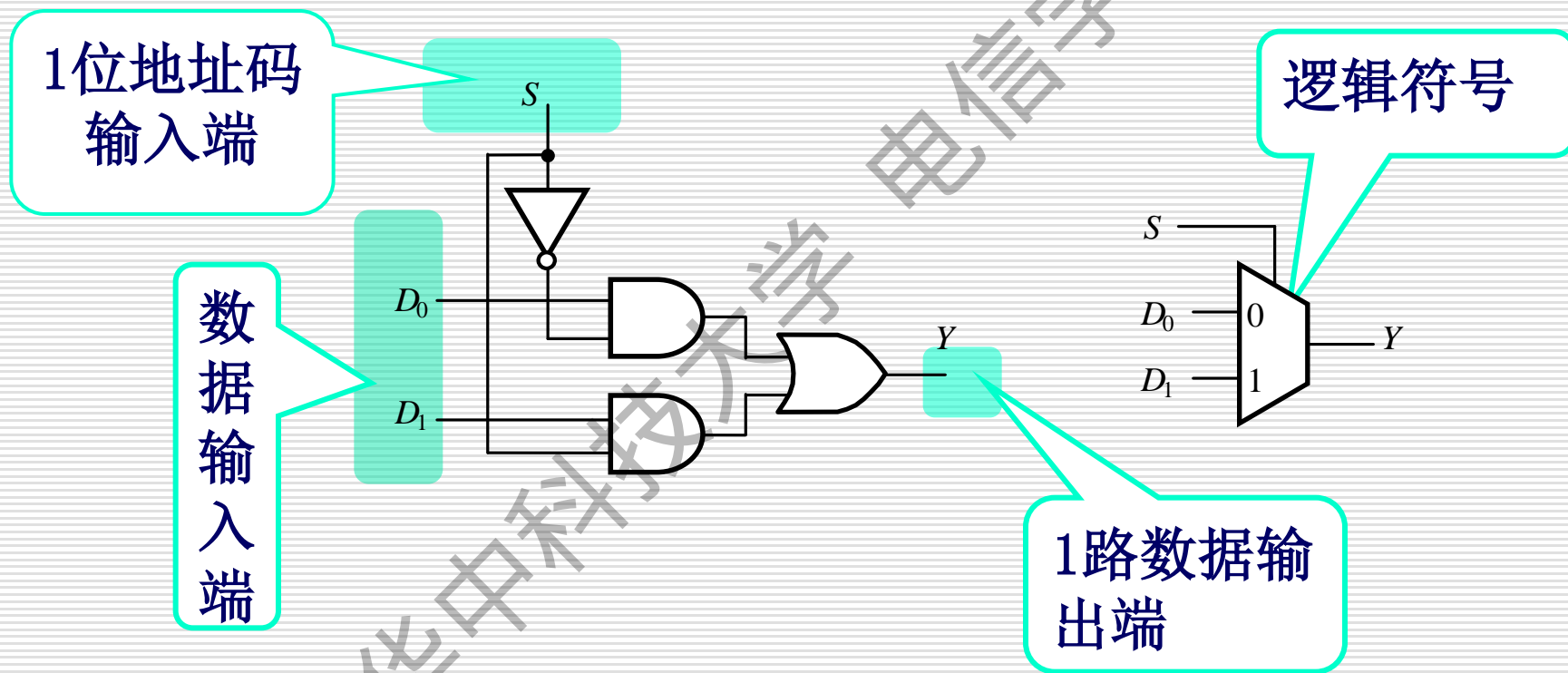
1、数据选择器的定义与功能

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



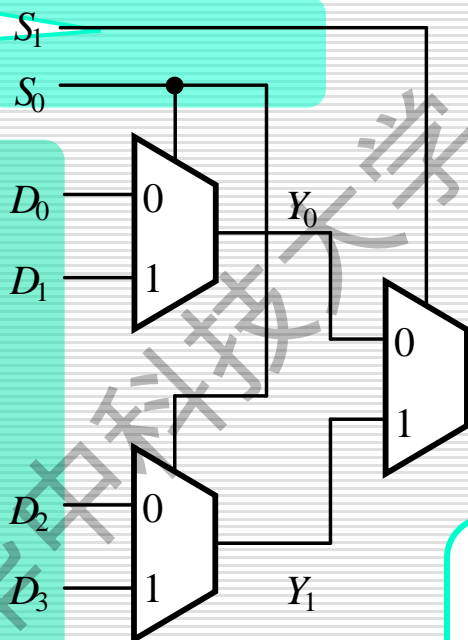
4选1数据选择器

(1) 逻辑电路

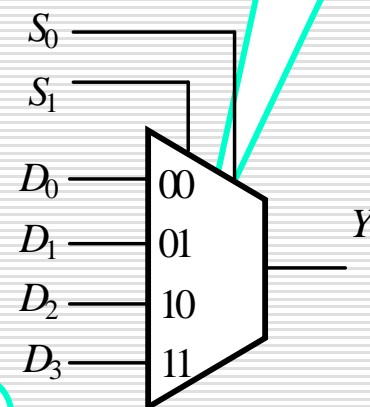
由3个2选1数据选择器构成4选1数据选择器。

2 位地址
码输入端

数据
输入端



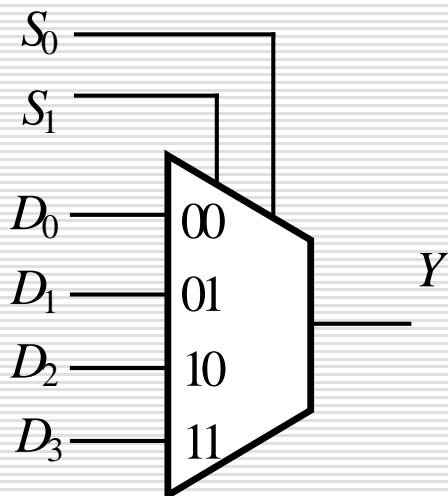
逻辑符号



1路数据
输出端

(2) 工作原理及逻辑功能

真值表



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

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

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

(3) 数据选择器实现逻辑函数

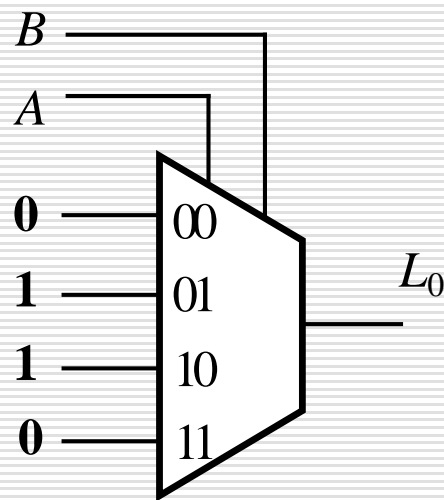
例4.4.8 试用数据选择器实现下列逻辑函数

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

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

① 当 $S_1 = A, S_0 = B,$
 $D_0 = D_3 = 0, D_1 = D_2 = 1$

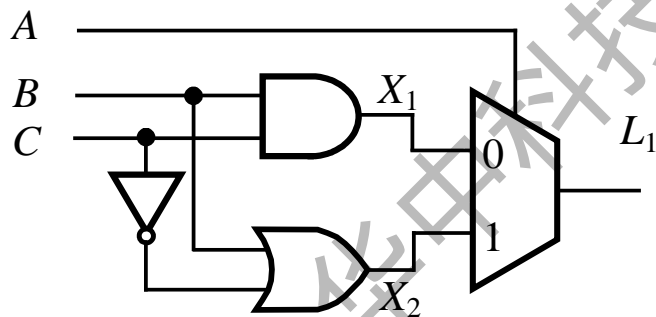


② 用2选1数据选择器和必要的逻辑门实现 $L_1 = AB + \overline{A}C + BC$

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。

因此需要将逻辑函数式转化为如下形式：

$$L_1 = \overline{S}X_1 + SX_2$$



(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:(变量数=选通端数)

a、将函数变换成最小项表达式

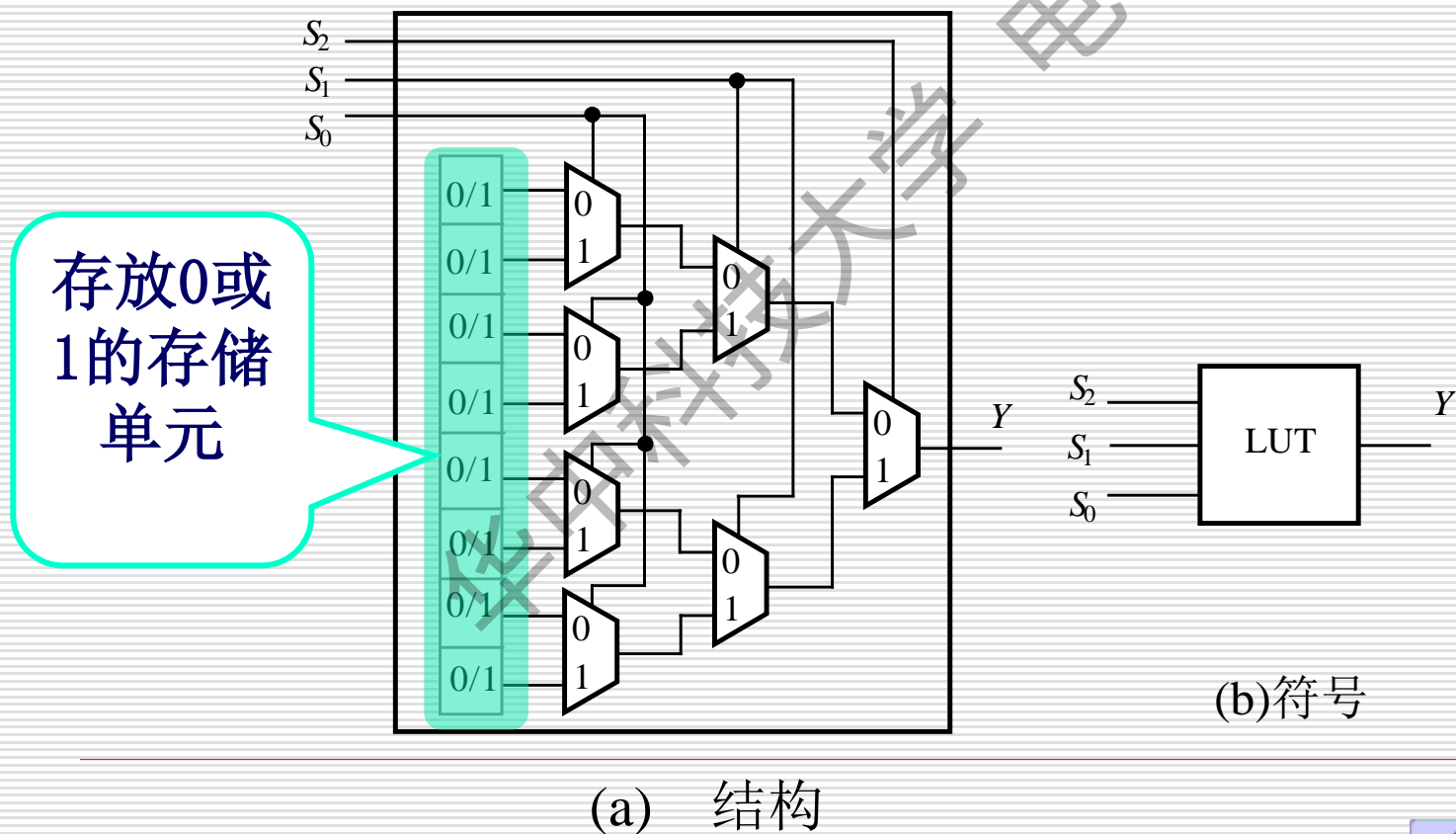
b、地址信号 S_2 、 S_1 、 S_0 作为函数的输入变量

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

当变量数>选通端数, 考虑如何将某些变量接入数据端。

(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

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

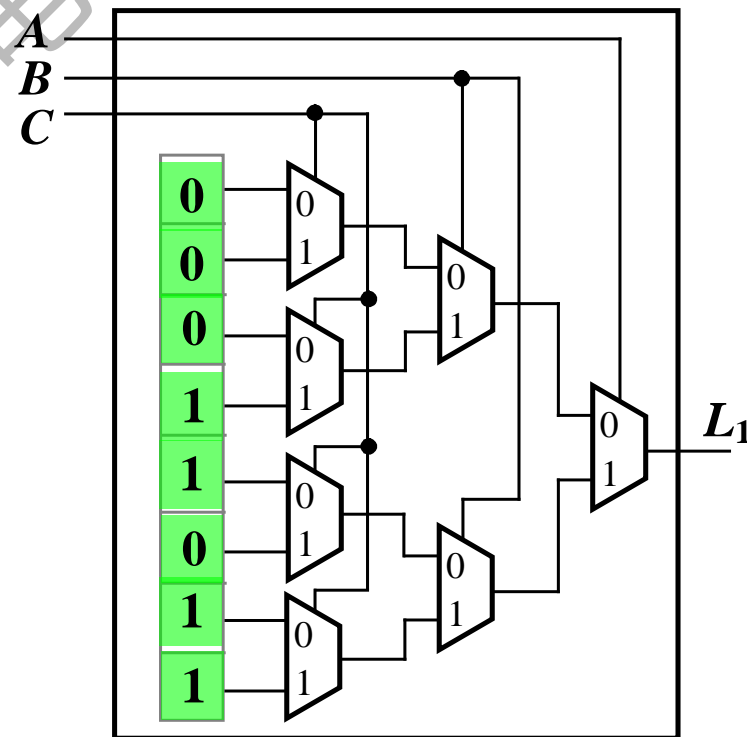
用LUT实现逻辑函数，变量A、B、C接选择输入端，对存储单元进行编程。

根据前面例题已知

$$\begin{aligned} L_1 &= AB + \overline{A}\overline{C} + BC \\ &= \sum m(3,4,6,7) \end{aligned}$$

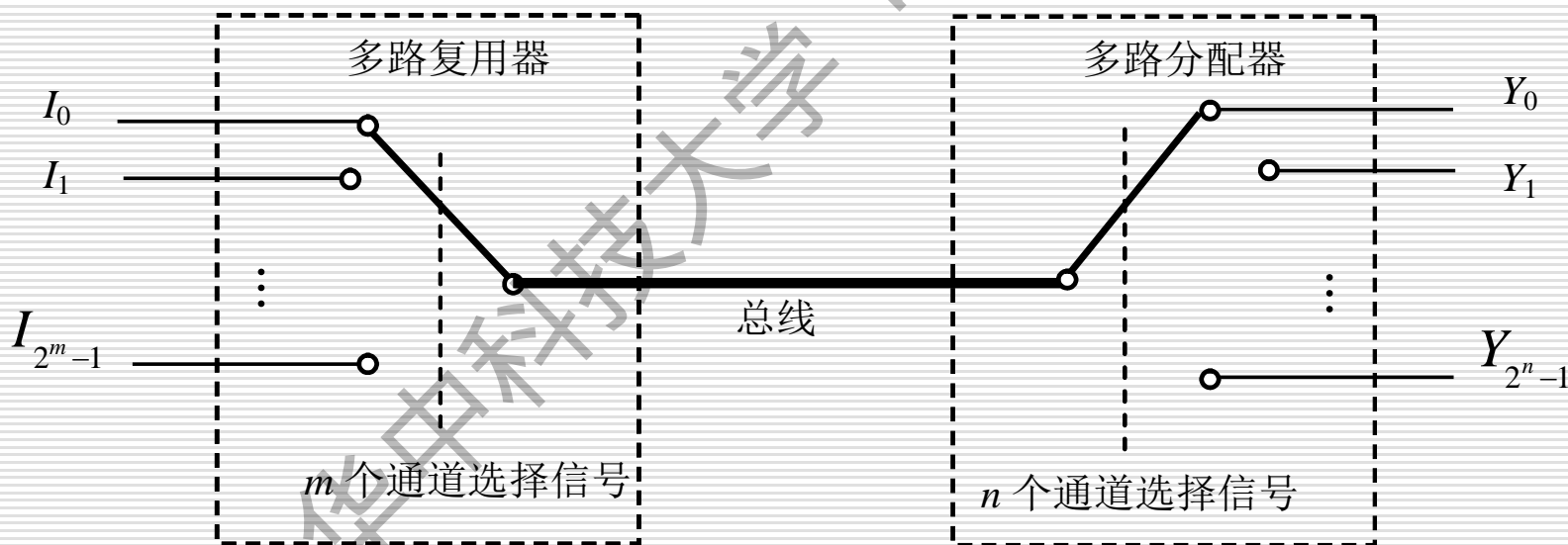
$$D_0 = D_1 = D_2 = D_5 = 0$$

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



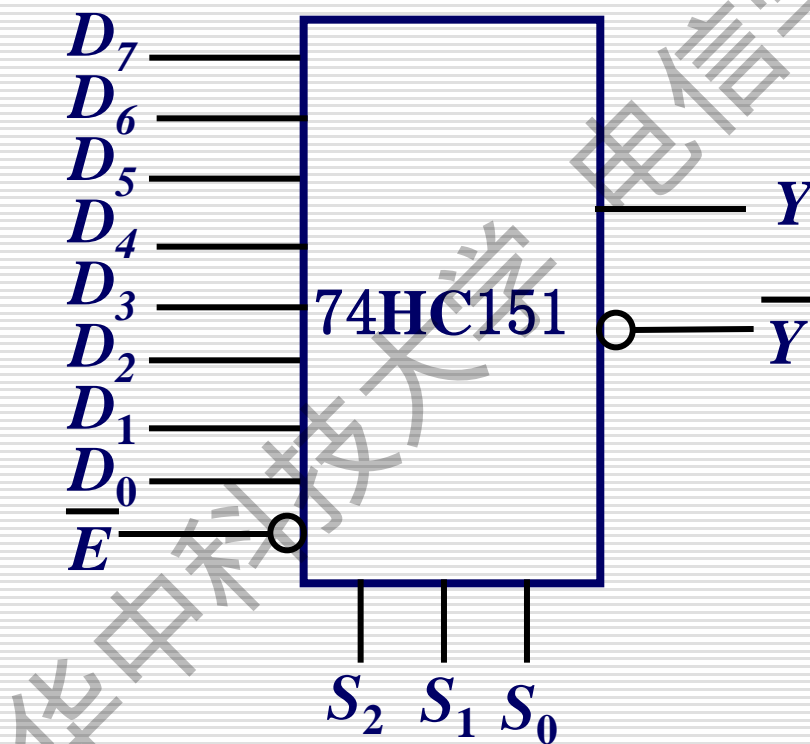
(6) 数据选择器、数据分配器与总线的连接

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。



(7) 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

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

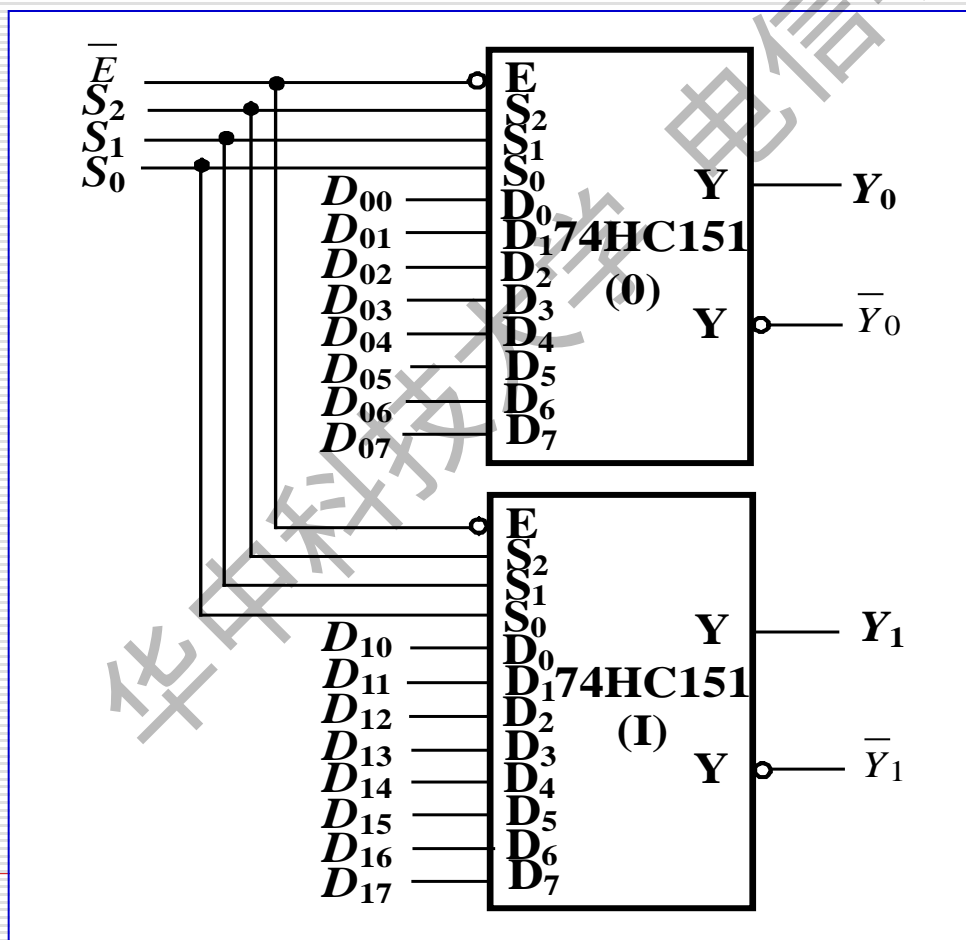
$$Y = \sum_{i=0}^7 D_i m_i$$

输 入				输 出	
使能 \overline{E}	选 择			Y	\overline{Y}
	S_2	S_1	S_0		
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

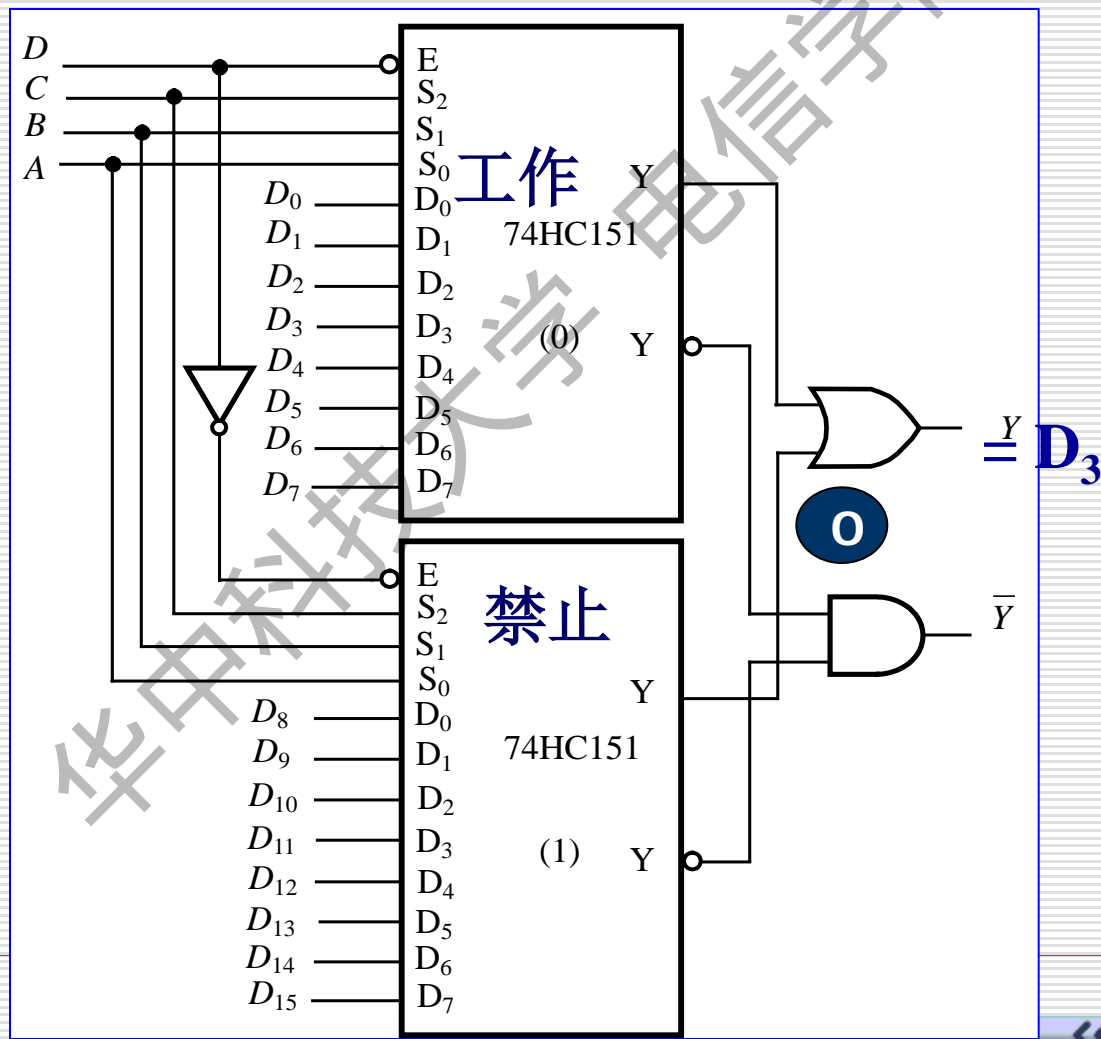
用两片74151组成二位八选一的数据选择器



字的扩展

将两片74LS151连接成一个16选1的数据选择器，

DCBA=0011



4.4.4 数值比较器

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

1. 1位数值比较器(设计)

输入：两个一位二进制数 A 、 B 。

输出： $F_{A>B}=1$ ，表示 A 大于 B

$F_{A<B}=1$ ，表示 A 小于 B

$F_{A=B}=1$ ，表示 A 等于 B

1位数值比较器

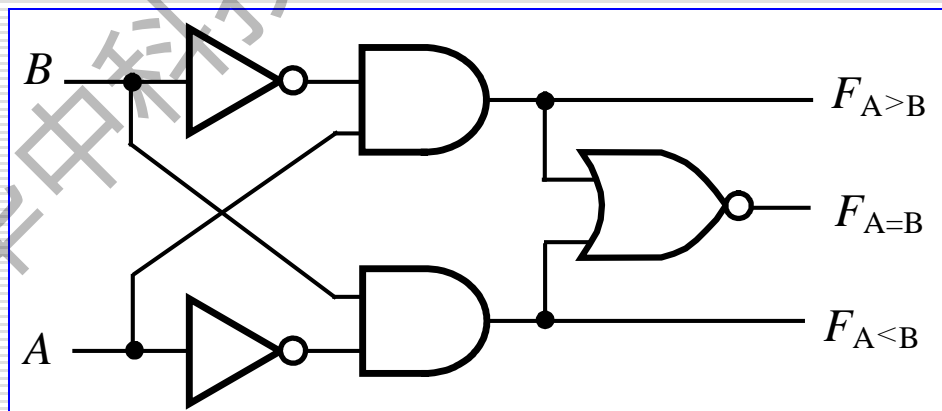
一位数值比较器真值表

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

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

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

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

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

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

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

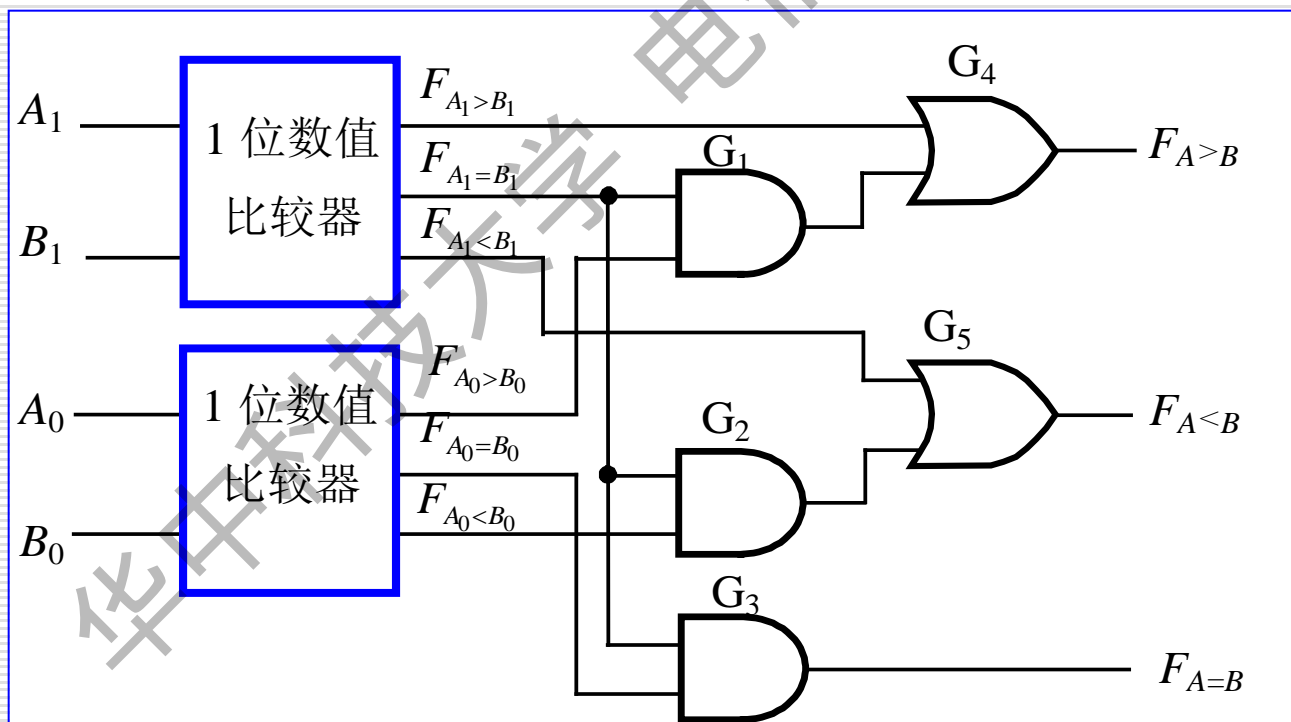
注意：上述不是真正的逻辑函数表达式，只示意逻辑关系。

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

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

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

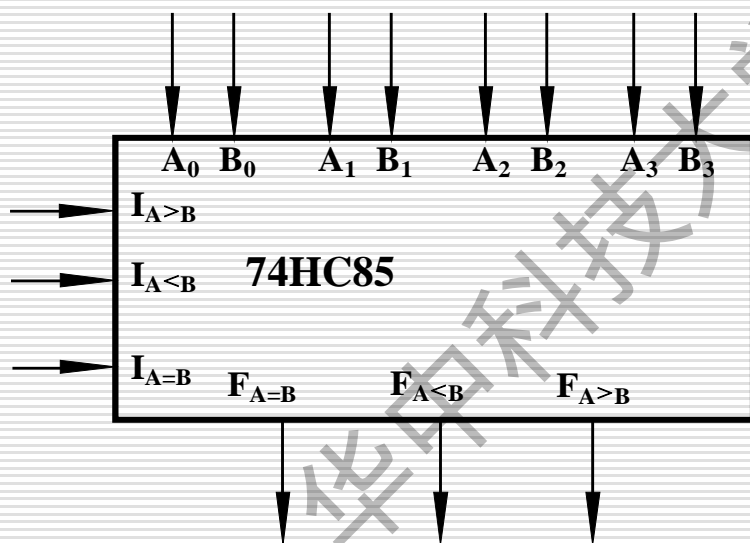
两位数值比较器逻辑图



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

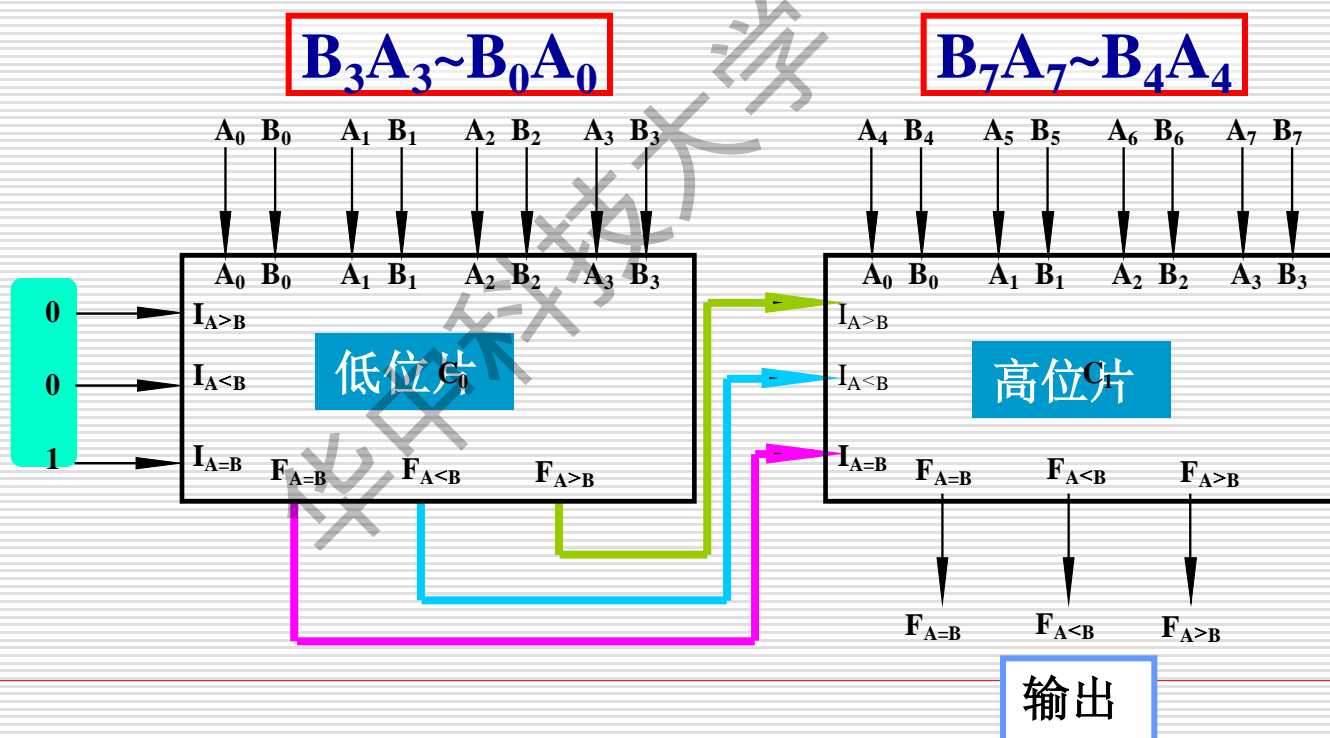
输 入							输 出		
$A_3 B_3$	$A_2 B_2$	$A_1 B_1$	$A_0 B_0$	$I_{A>B}$	$I_{A<B}$	$I_{A=B}$	$F_{A>B}$	$F_{A<B}$	$F_{A=B}$
$A_3 > B_3$	×	×	×	×	×	×	1	0	0
$A_3 < B_3$	×	×	×	×	×	×	0	1	0
$A_3 = B_3$	$A_2 > B_2$	×	×	×	×	×	1	0	0
$A_3 = B_3$	$A_2 < B_2$	×	×	×	×	×	0	1	0
$A_3 = B_3$	$A_2 = B_2$	$A_1 > B_1$	×	×	×	×	1	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
$A_3 = B_3$	$A_2 = B_2$	$A_1 = B_1$	$A_0 = B_0$	×	×	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： $A=A_7A_6A_5A_4A_3A_2A_1A_0$ $B=B_7B_6B_5B_4B_3B_2B_1B_0$

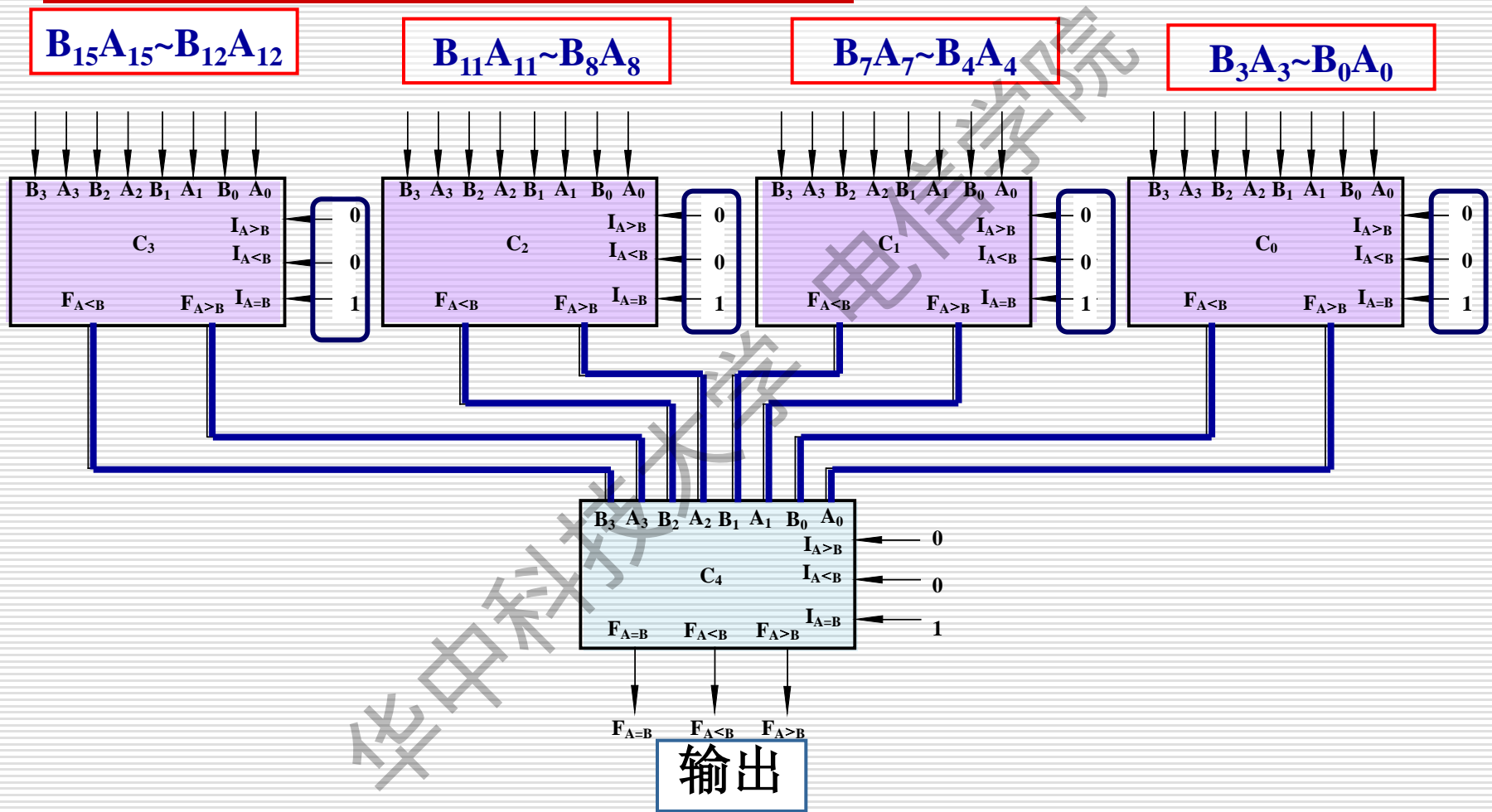
输出： $F_{A>B}$ $F_{A<B}$ $F_{A=B}$



用两片74HC85组成16位数值比较器（串联扩展方式）。



用74HC85组成16位数值比较器的并联扩展方式。

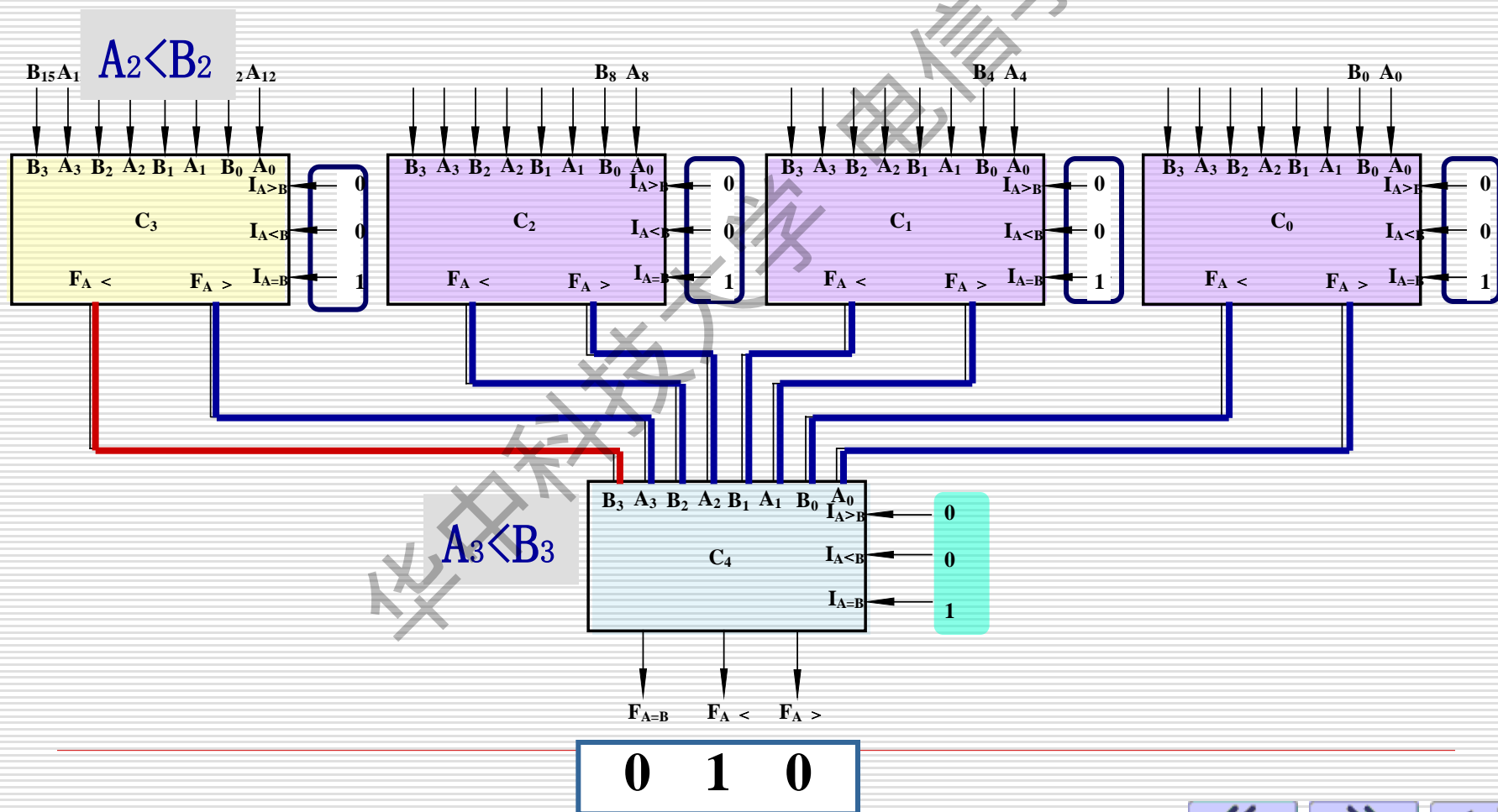


问题：如果每一片延迟时间为10ns，16位并行比较器延迟时间？

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

1、半加器和全加器

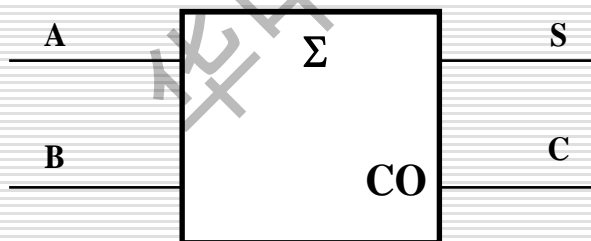
$$\begin{array}{r} 1\ 1\ 0\ 1 \\ +\ 1\ 0\ 0\ 1 \\ \hline 1\ 0\ 1\ 1\ 0 \end{array}$$

两个1位二进制数相加时，不考虑低位来的进位的加法
---半加

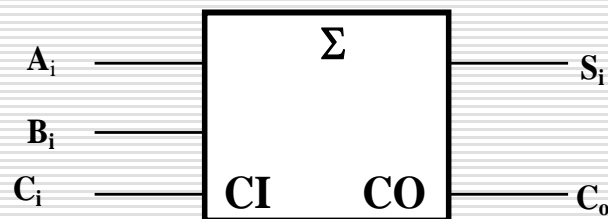
在两个1位二进制数相加时，考虑低位进位的加法
---全加

加法器分为半加器和全加器两种。

半加器



全加器



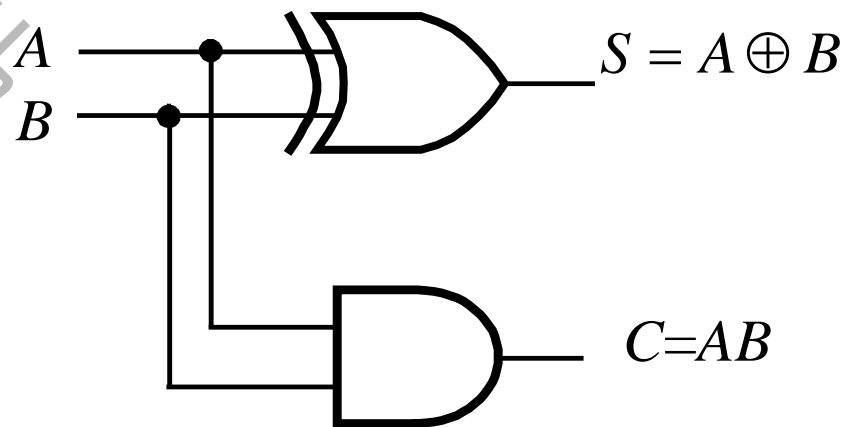
(1) 1位半加器 (Half Adder)

不考虑低位进位，将两个1位二进制数A、B相加的器件。

- 半加器的真值表
- 逻辑表达式

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

$$C = AB$$



如用与非门实现最少要几个门？

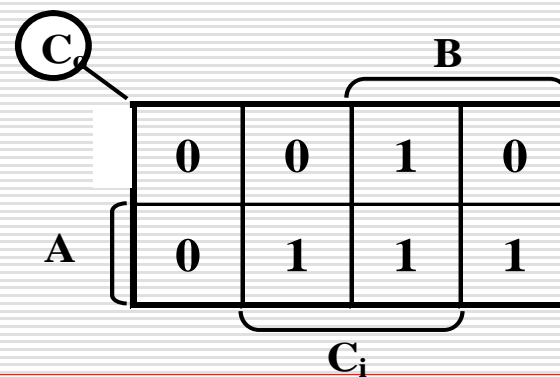
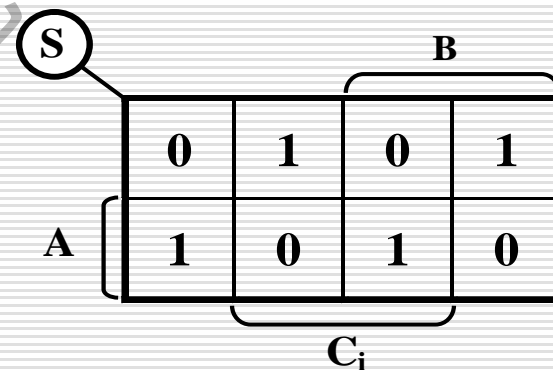
- 逻辑图

(2) 全加器 (Full Adder)

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

全加器真值表

A	B	C_i	S	C_o
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



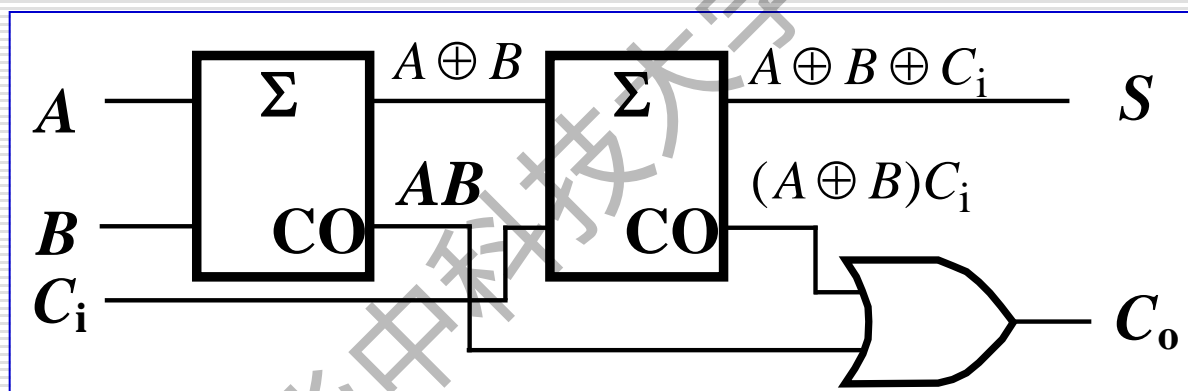
于是可得全加器的逻辑表达式为

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

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

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

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



- 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	S	C_o
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

ABC 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

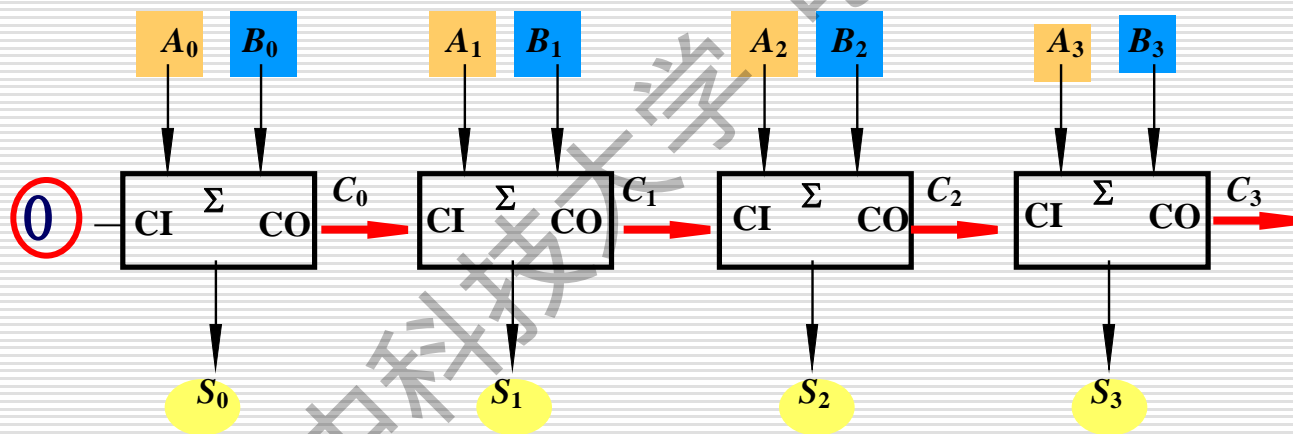
用全加器组成八位二进制代码
奇偶校验器，电路应如何连接？

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)：

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

定义两个中间变量 G_i 和 P_i ：

$$G_i = A_i B_i$$

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

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

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

4位全加器进位信号的产生:

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

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

$$C_1 = G_1 + P_1 C_0$$

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

$$C_2 = G_2 + P_2 C_1$$

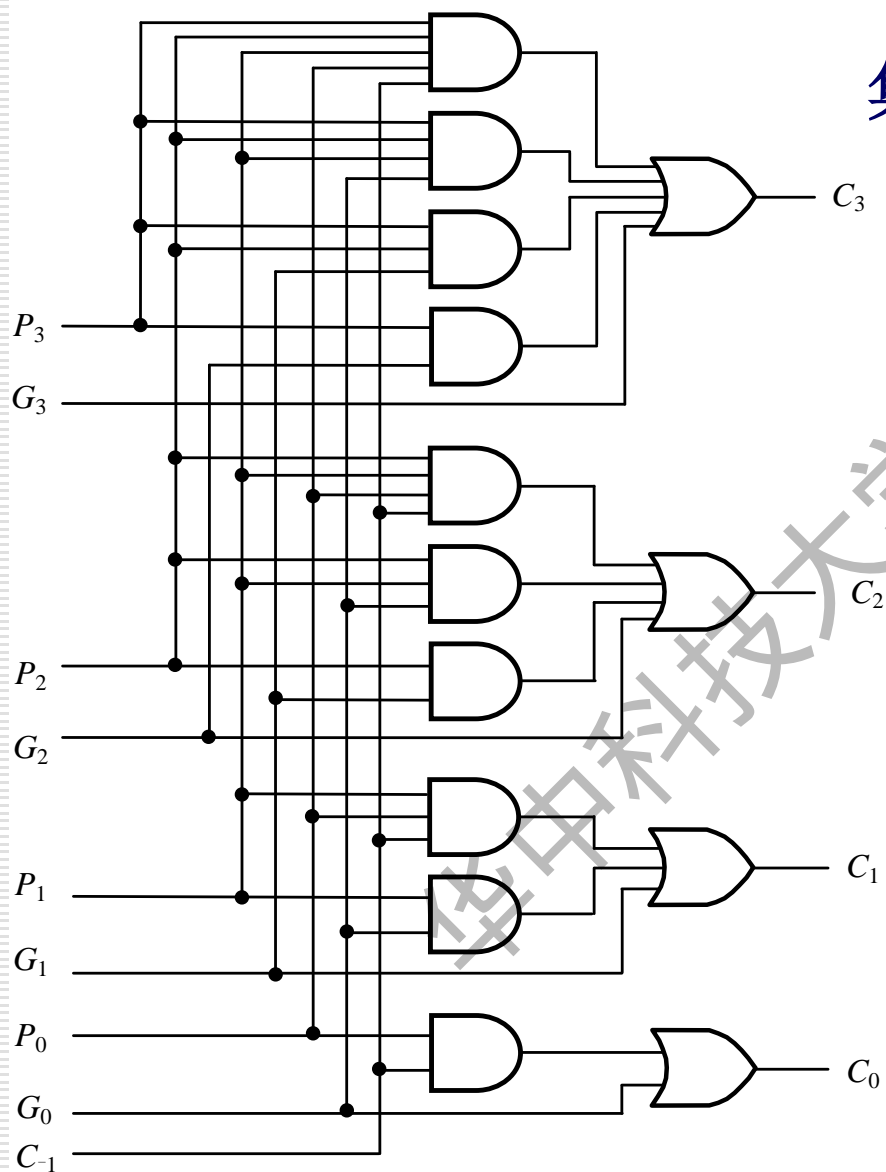
$$C_2 = G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_{-1}$$

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

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

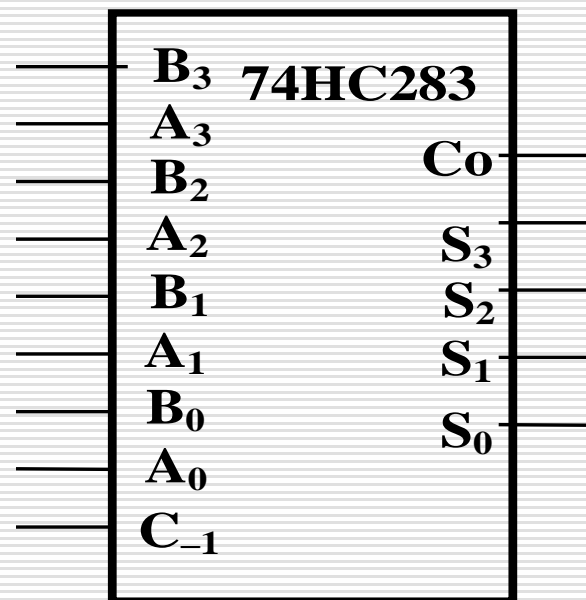
进位信号只由被加数A、加数B和 C_{-1} 决定，而与其它低位的进位无关。提高了速度，但位数增加时，进位电路复杂度增加。

超前进位产生电路(74HC182)



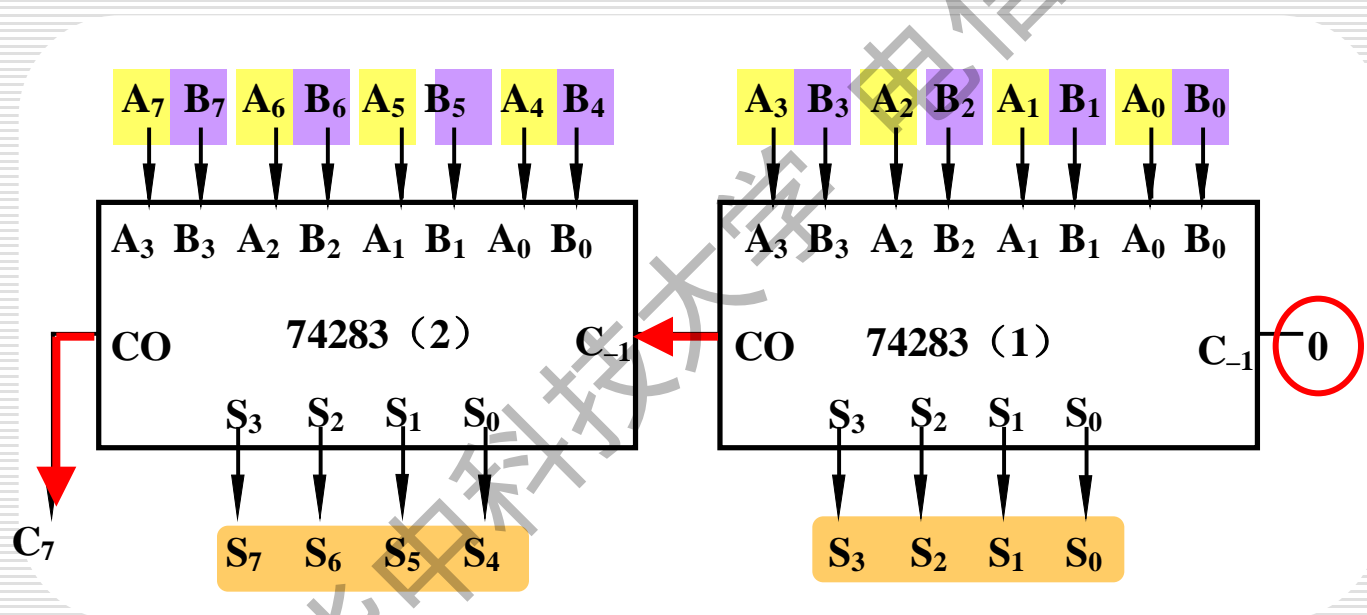
集成4位超前进位加法器74HC283

74HC283逻辑框图



4. 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

0000

0001

0010

⋮

+0011

+0011

+0011

余3码

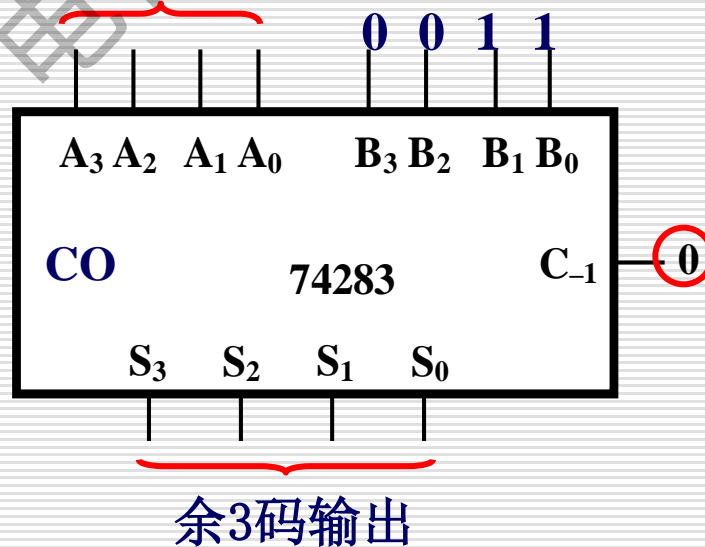
0011

0100

0101

⋮

8421码输入



3 减法运算

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

$$N_{\text{补}} = 2^N - N_{\text{原}}$$

补码与反码的关系式

$$N_{\text{补}} = N_{\text{反}} + 1$$

设两个数 A 、 B 相减，利用以上两式
可得

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

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

1) $A-B \geq 0$ 的情况。

2) $A-B < 0$ 的情况。

$A=0101$, $B=0010$

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

舍弃

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

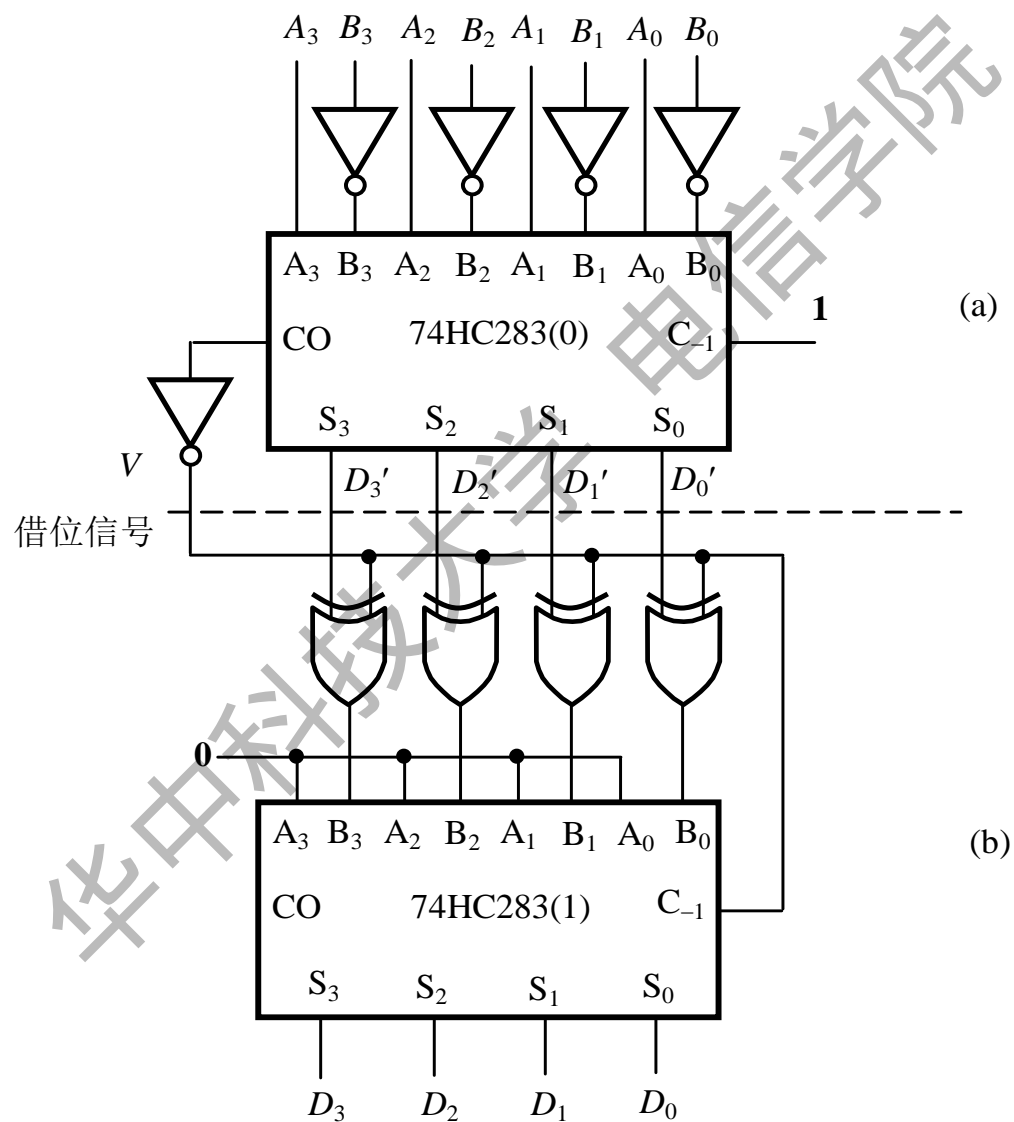
$A=0010$, $B=0101$

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

舍弃

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

输出为原码的4位减法运算逻辑图



4.5 组合可编程逻辑器件

4.5.1 PLD的结构、表示方法及分类

4.5.2 组合逻辑电路的PLD实现

4.5 组合可编程逻辑器件

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

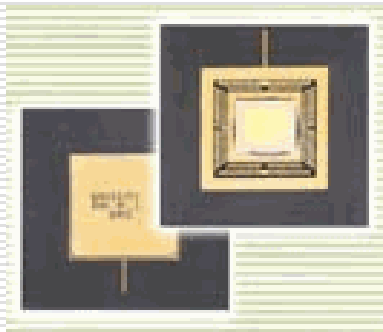
4.5 组合可编程逻辑器件

概 述

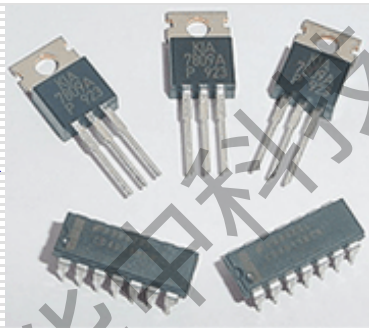
一、数字电路的发展与可编程器件的出现

集成度：

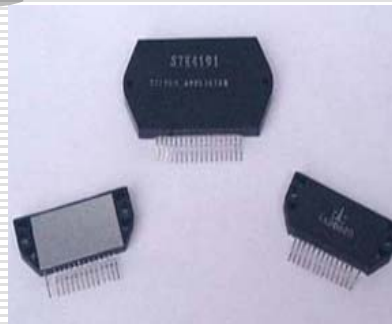
SSIC



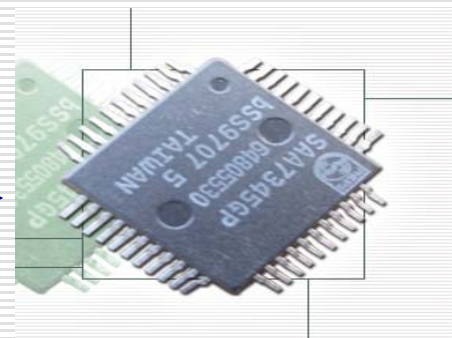
MSIC



LSIC



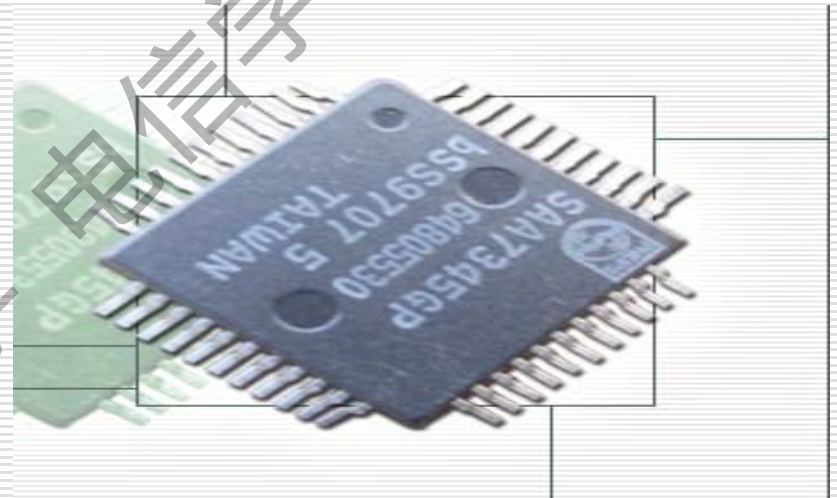
VLSIC



高效、低耗、高精度、高稳定、智能化。

逻辑功能：通用型：54/74系列、74HC系列、74HCT系列等

VLSIC



随系统规模扩大：
焊点多，可靠性下降
功耗增加、成本升高
占用空间扩大

专用型：ASIC

(Application Specific Integratel Circuit)

要承担设计风险、
周期长、成本高

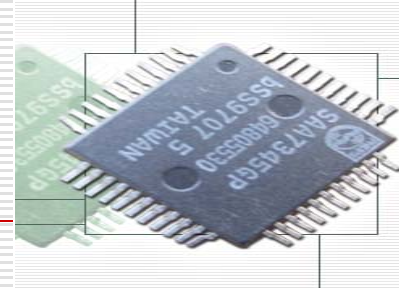
系统设计师们希望自己设计 ASIC芯片，缩短设计周期，
能在实验室设计好后，立即投入实际应用。

可编程器件 (PLD : Programmable Logic Device)

二、PLD的发展态势

- 向高集成度、高速度方向发展
集成度已达到400万门以上
- 向低电压和低功耗方向发展,
 $5V \rightarrow 3.3V \rightarrow 2.5V \rightarrow 1.8V \rightarrow$ 更低
- 向内嵌多种功能模块方向发展
RAM, ROM, DSP, CPU等
- 向数、模混合可编程方向发展





三、PLD的性能特点

- 1、逻辑功能强：**PLD如一堆积木，它能完成任何数字器件的功能，用户可以自己设计上至高性能CPU,下至简单的MSIC电路。
- 2、集成度高：**可以替代多至几千块通用IC芯片 极大减小电路的面积和电路连接，从而大大降低功耗，提高抗干扰能力，和可靠性。
- 3、设计方法灵活：**可通过传统的原理图输入法或是硬件描述语言，自由的设计一个数字系统。使用PLD器件设计的系统，可以不受标准系列器件在逻辑功能上的限制。

4、具有完善先进的开发工具：

- 提供语言、图形等设计方法，十分灵活
- 通过仿真工具来验证设计的正确性

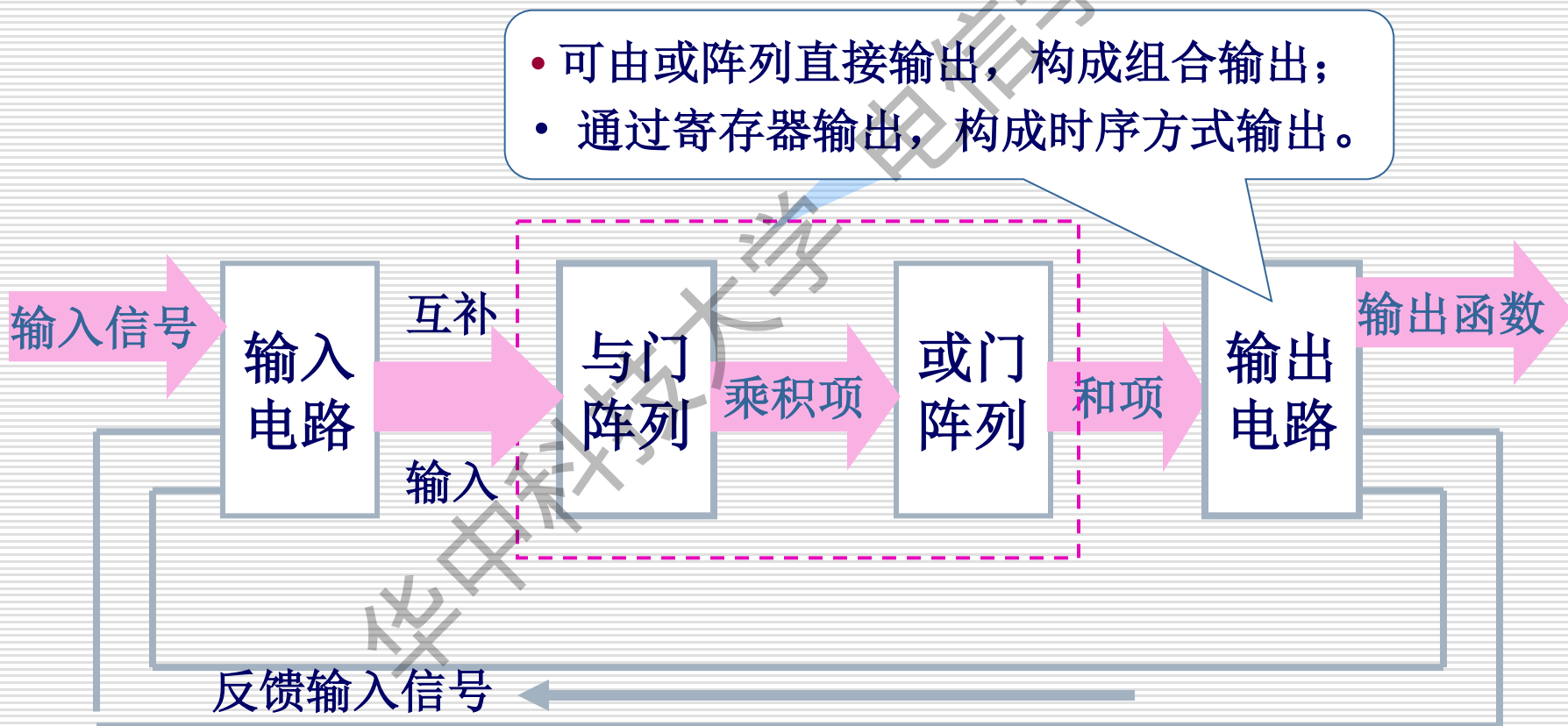
5、系统处理速度快：用PLD与或两级结构实现任何逻辑功能，所需的逻辑级数少。这不仅简化了系统设计，而且减少了级间延迟，提高了系统的处理速度。

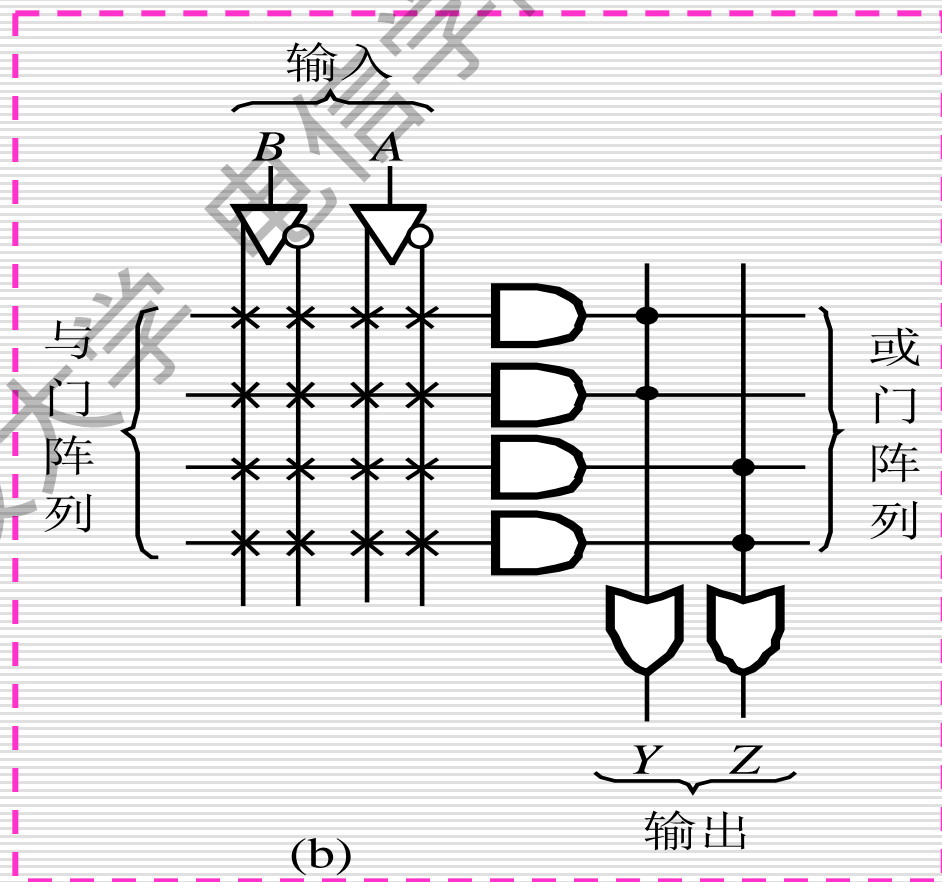
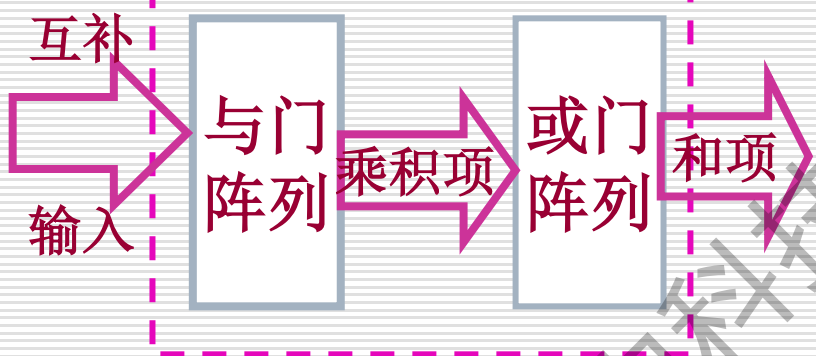
6、系统具有加密功能：设计者在设计时选中加密项，可编程逻辑器件就被加密，器件的逻辑功能无法被读出，有效地防止逻辑系统被抄袭。

7、使用方便：可以反复地擦除、编程，方便设计的修改和升级

4.5.1 PLD的结构、表示方法及分类

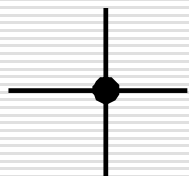
1、PLD的基本结构



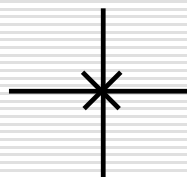


2. PLD的逻辑符号表示方法

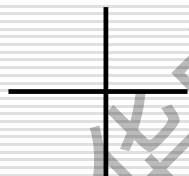
(1) 连接的方式



硬线连接单元



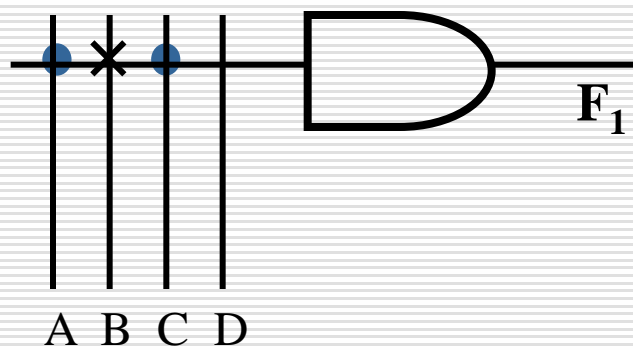
被编程接通单



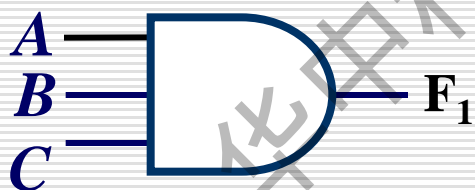
被编程擦除单元

(2) 基本门电路的表示方式

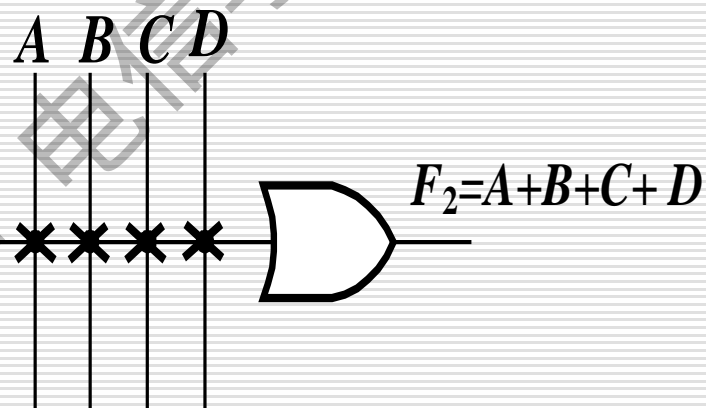
与门



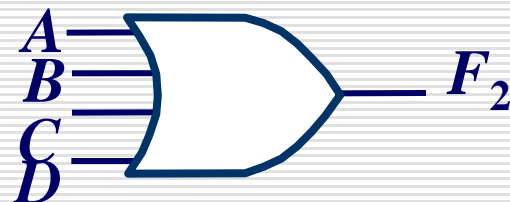
$$F_1 = A \cdot B \cdot C$$

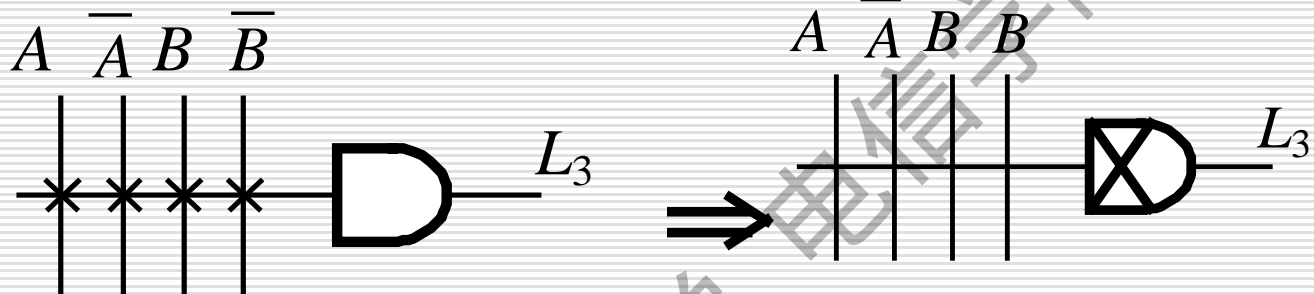


或门

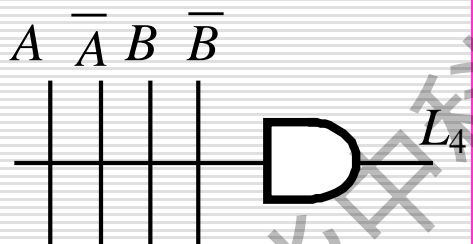


$$F_2 = A + B + C + D$$

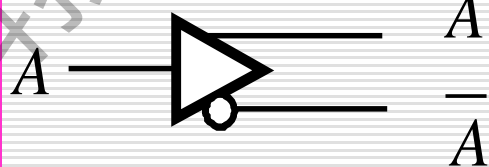




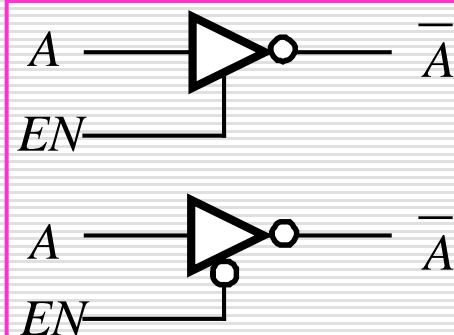
输出恒等于0的与门



输出为1的与门



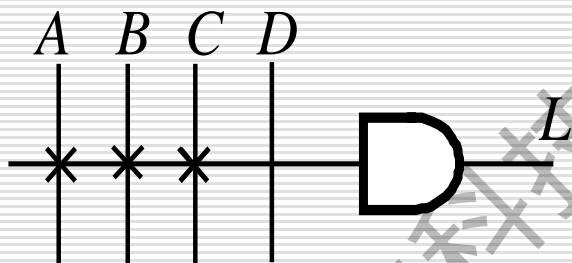
输入缓冲器



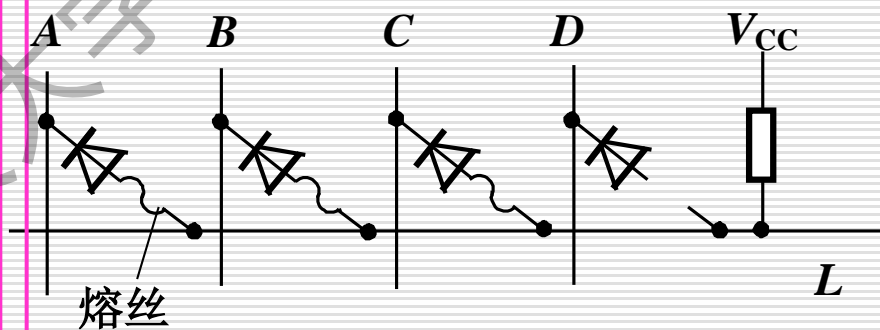
三态输出缓冲器

(3) 编程连接技术

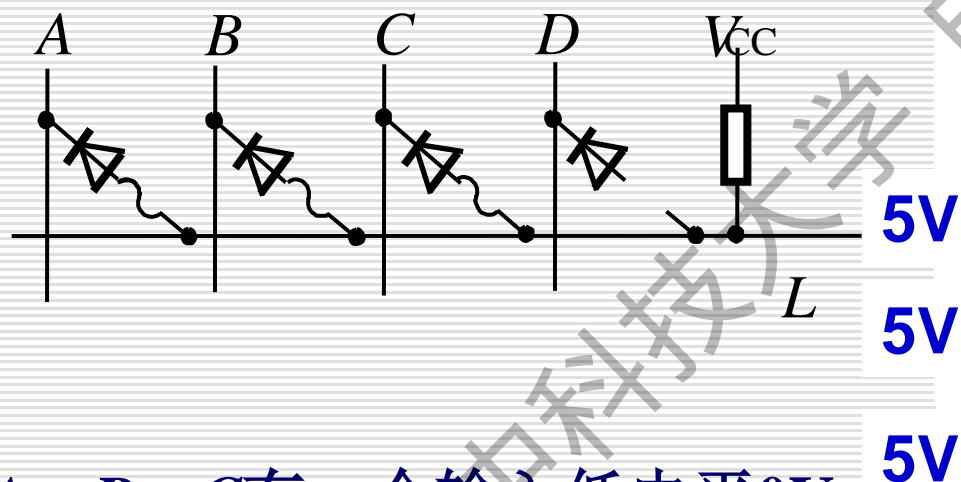
PLD表示的与门



熔丝工艺的与门原理图

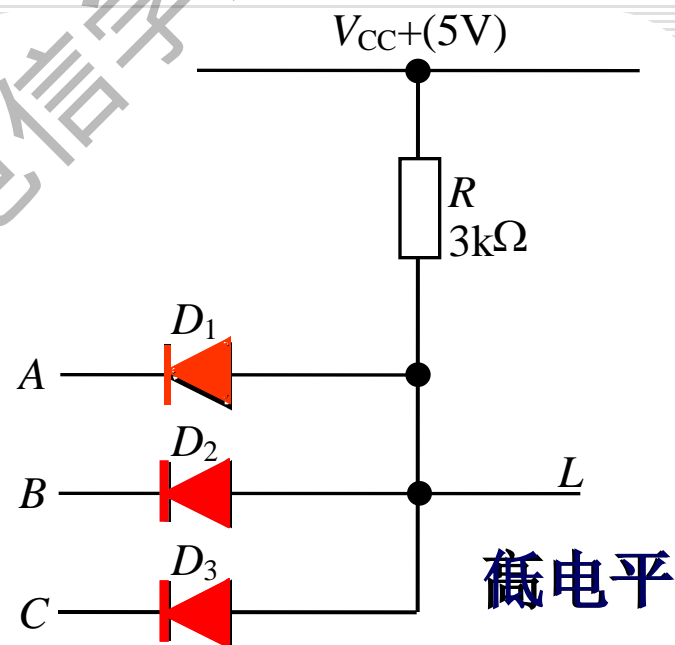


$$L=A \cdot B \cdot C$$

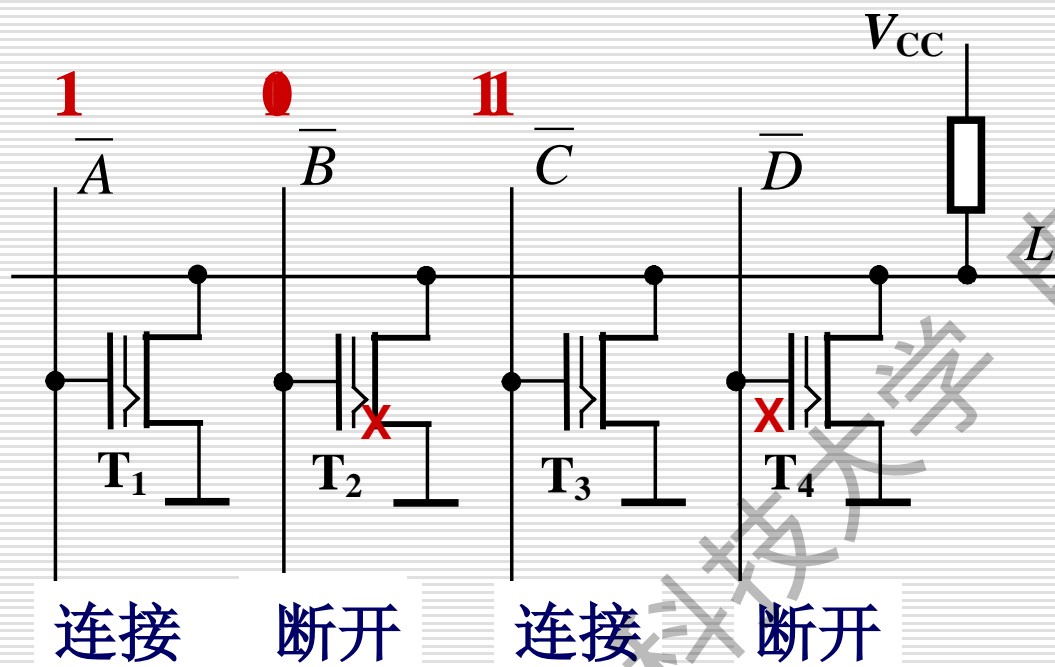


A、B、C有一个输入低电平0V

A、B、C三个都输入高电平+5V



$$L=A \cdot B \cdot C$$



A 、 B 、 C 中有一个为0
输出为0;

A 、 B 、 C 都为1
输出为1。

$$L = ABC$$

$$L = AC$$

器件的开关状态不同, 电路实现逻辑函数也就不同

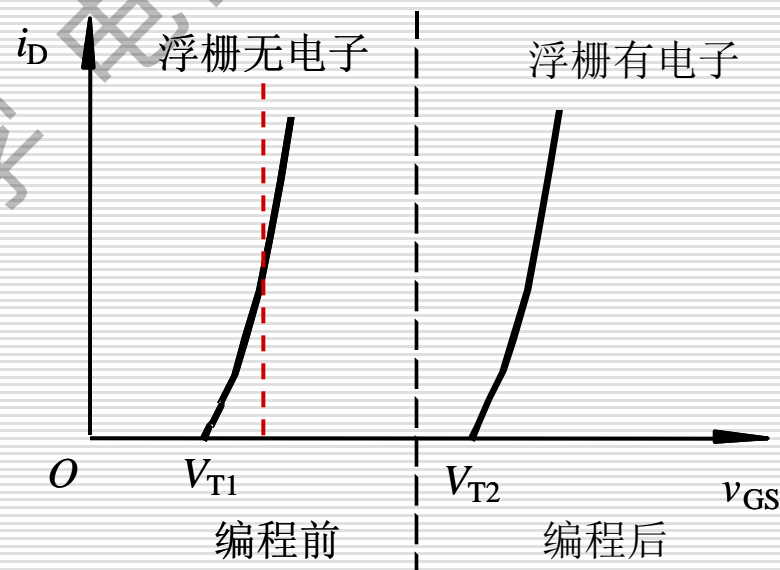
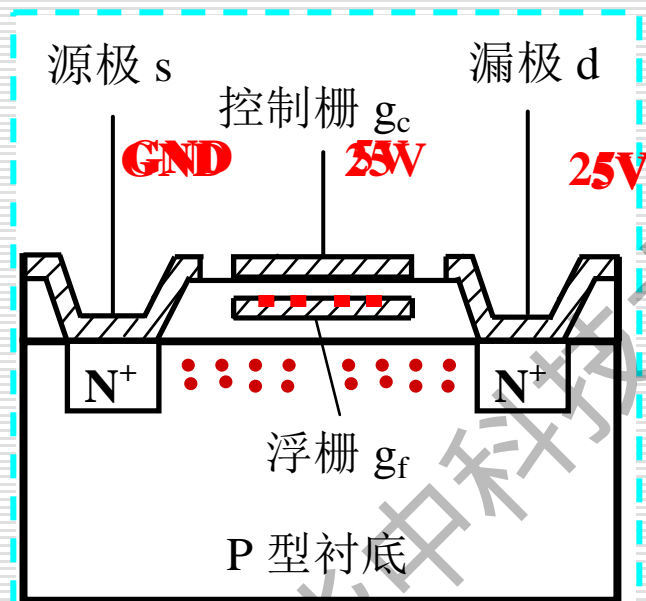
(4) 浮栅MOS管开关



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

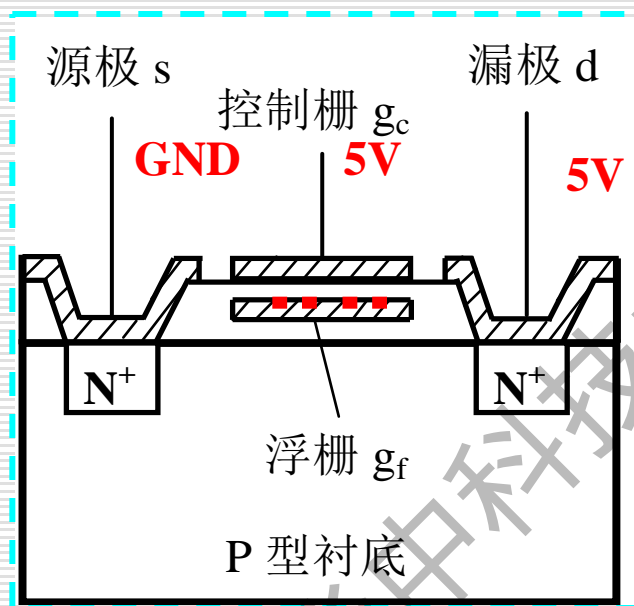
a. 叠栅注入MOS(SIMOS)管

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

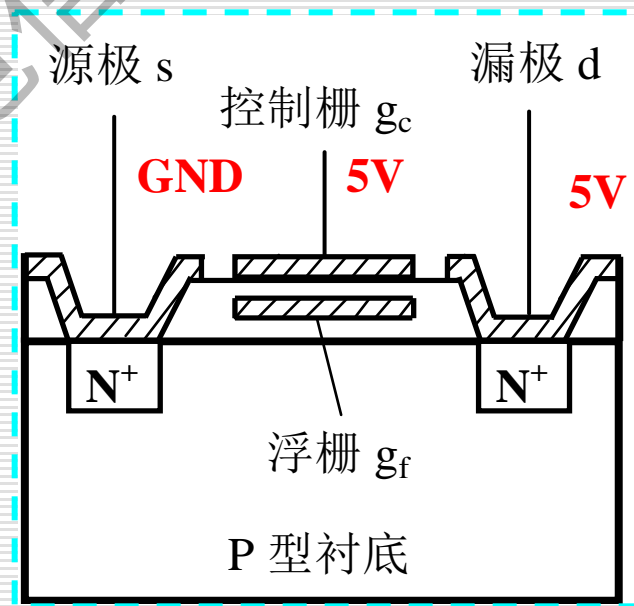


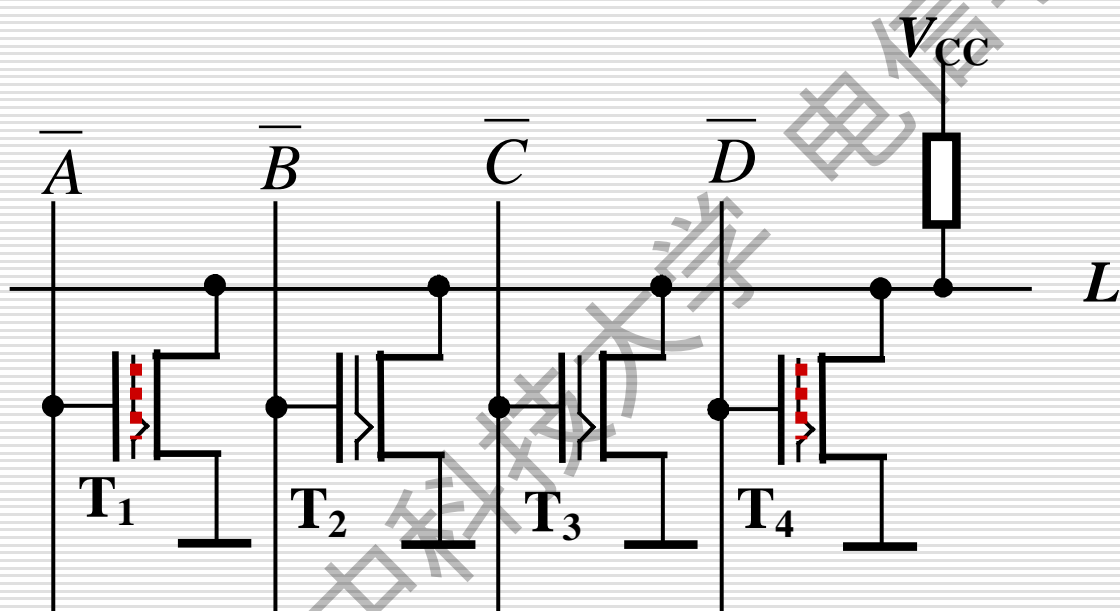
若要擦除，可用紫外线或X射线，距管子2厘米处照射15-20分钟。

截止



导通





断开 连接 连接 断开

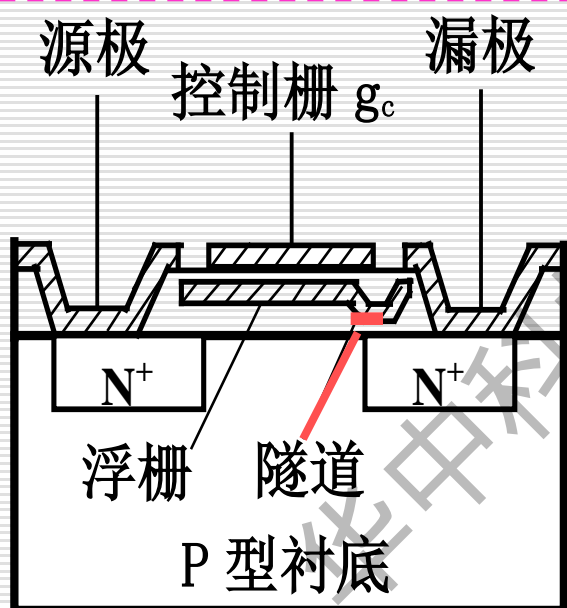
$$L = B \cdot C$$

b. 浮栅隧道氧化层MOS(Flotox MOS)管

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

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

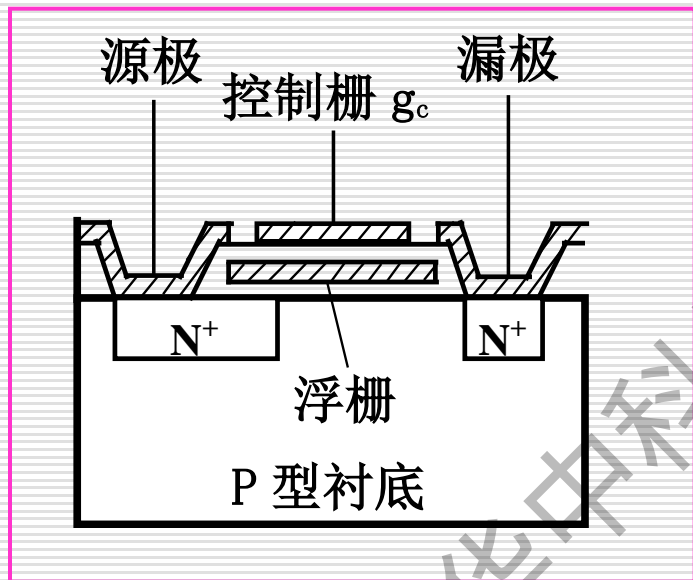
隧道MOS管是用电擦除的，擦除速度快。



c.快闪叠栅MOS管开关（Flash Memory）（自学）

结构特点：

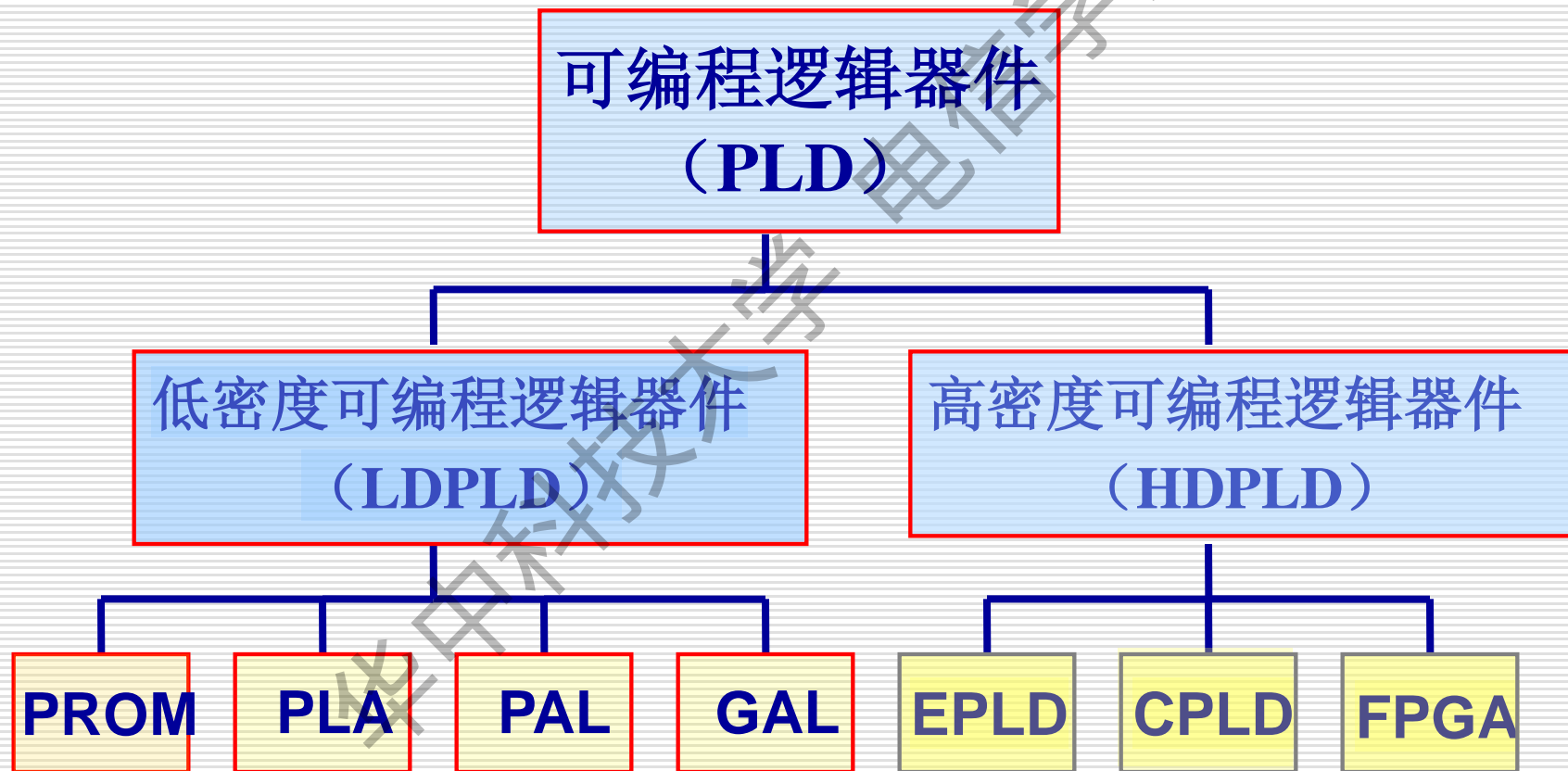
1. 闪速存储器存储单元MOS管的源极N+区大于漏极N+区，而SIMOS管的源极N+区和漏极N+区是对称的；
2. 浮栅到P型衬底间的氧化绝缘层比SIMOS管的更薄。



特点：结构简单、集成度高、编程可靠、擦除快捷。

3.PLD的分类

按集成密度划分为



2、按结构特点划分

- 简单PLD (PAL, GAL)

- 复杂的可编程器件(CPLD):

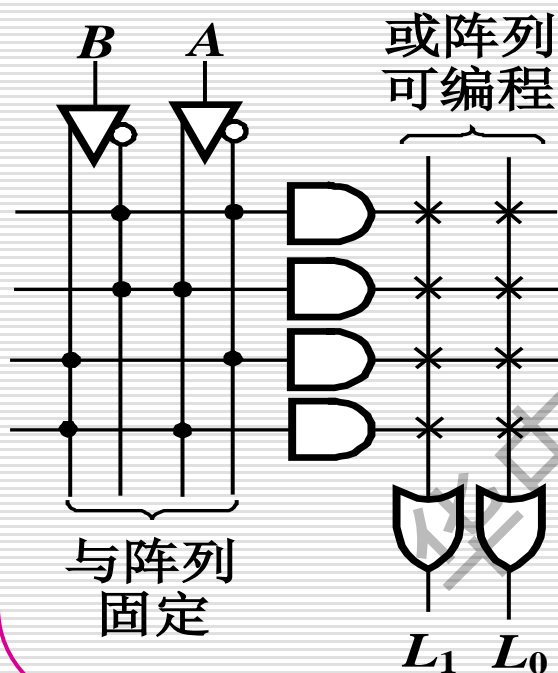
CPLD的代表芯片如: Altera的MAX系列

- 现场可编程门阵列(FPGA)

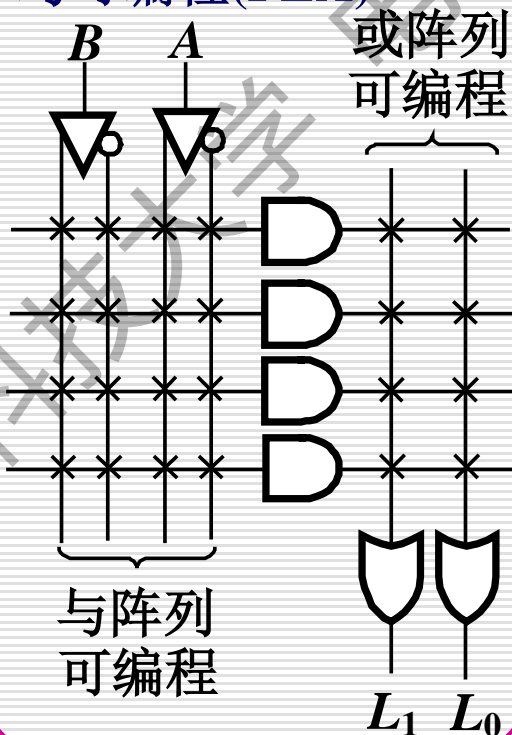
按PLD中的与、或阵列是否编程分

PLD中的三种与、或阵列

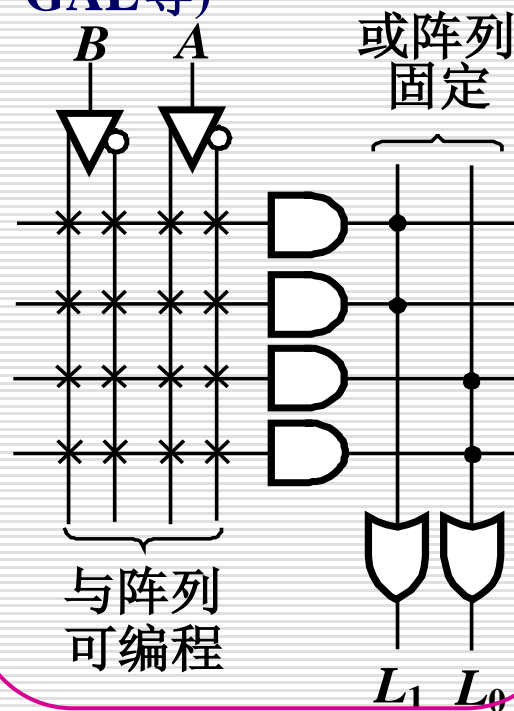
与阵列固定，或阵列可编程(PROM)



与阵列、或阵列均可编程(PLA)



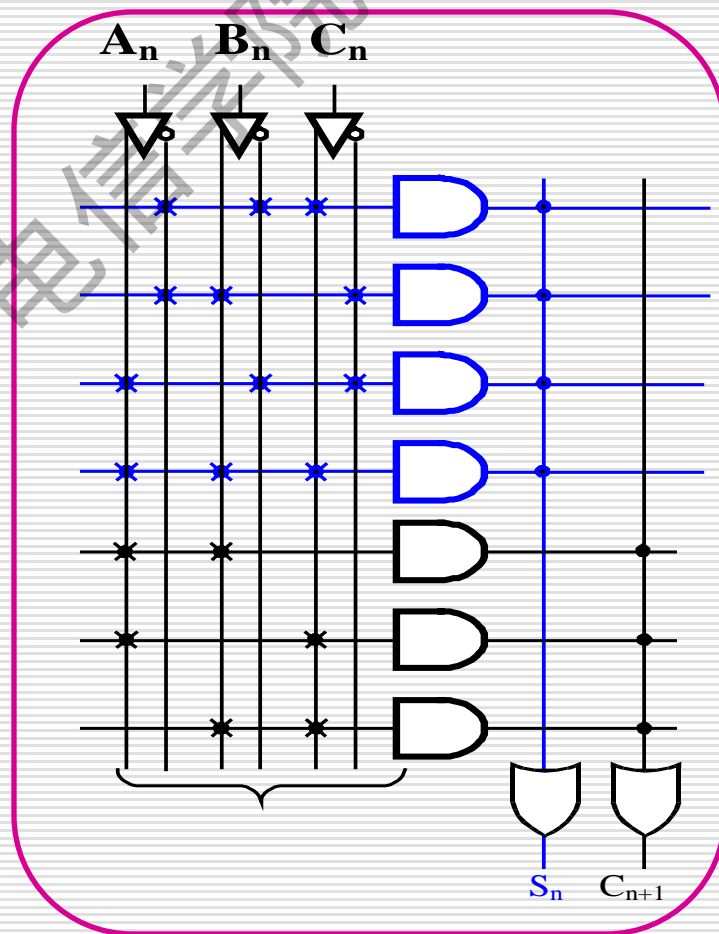
与阵列可编程，或阵列固定(PAL和GAL等)



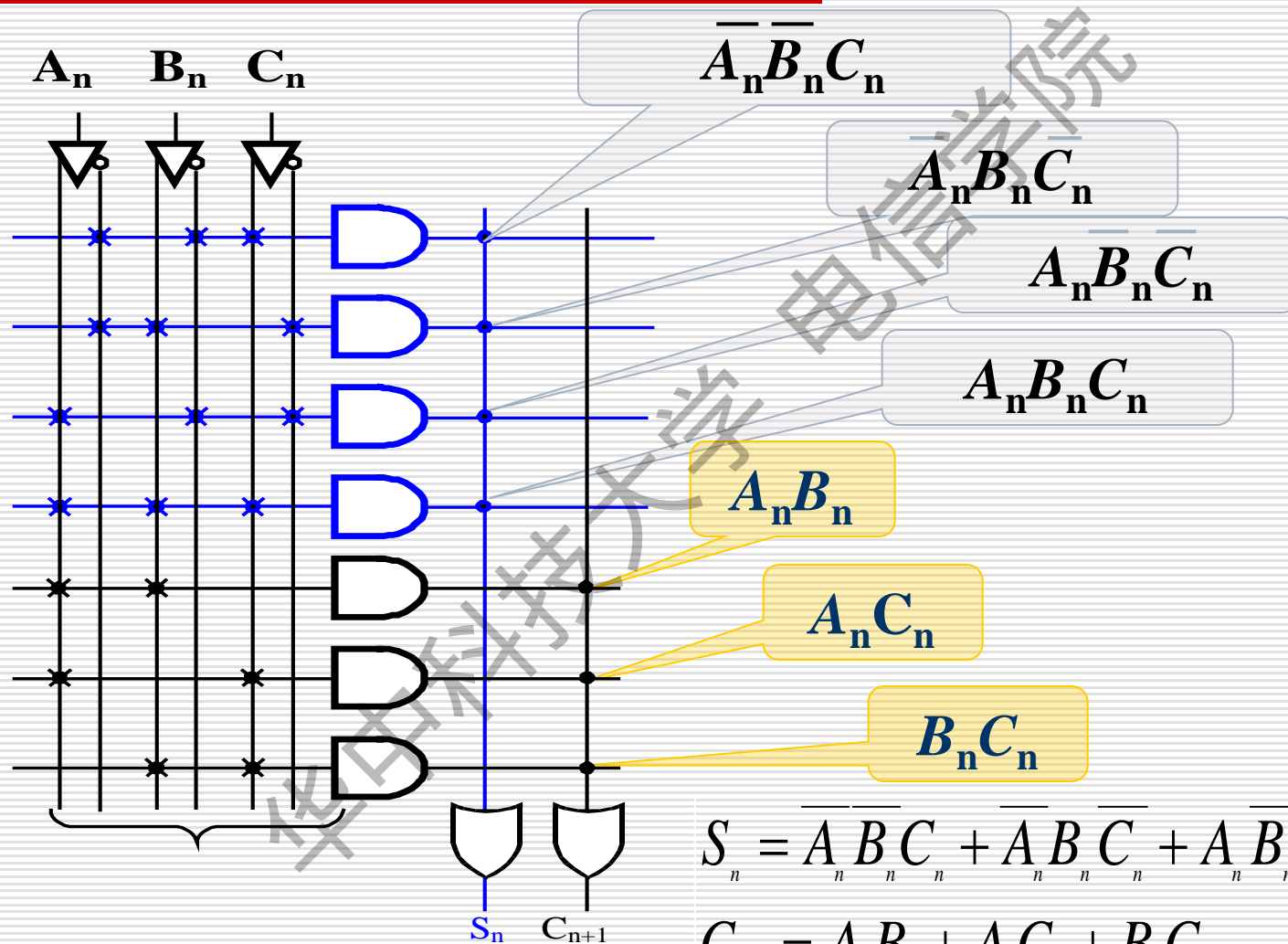
4.5.2 组合逻辑电路的 PLD 实现

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

写出该电路的逻辑表达式：



全加器

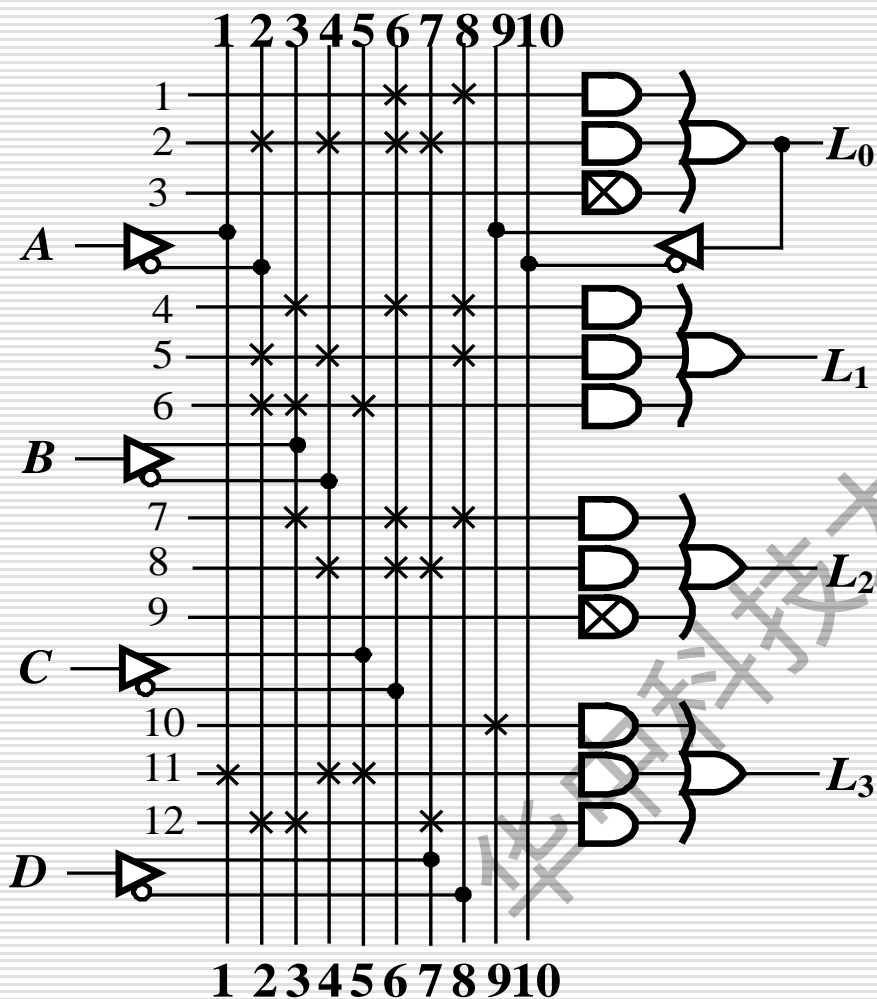


$$S_n = \overline{A_n} \overline{B_n} C_n + \overline{A_n} B_n \overline{C_n} + A_n \overline{B_n} \overline{C_n} + A_n B_n C_n$$

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

全加器

试写出该电路的逻辑表达式。



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

$$L_1 = \overline{B}CD + \overline{A}BD + \overline{A}BC$$

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

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

4.6 用Verilog HDL描述组合逻辑电路

4.6.1 组合逻辑电路的行为级建模

4.6.2 分模块、分层次的电路设计

4.6.1 组合逻辑电路的行为级建模

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

1、条件语句（if语句）

条件语句就是根据判断条件是否成立，确定下一步的运算。

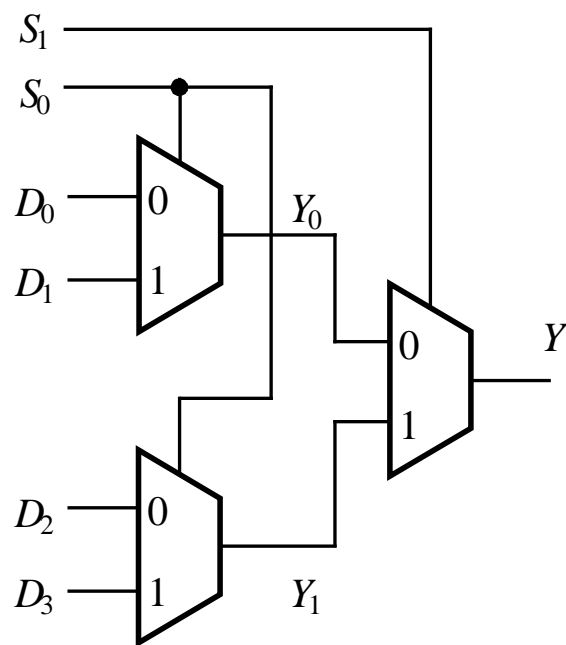
Verilog语言中有3种形式的if语句：

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

 else default_statement;

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

例：使用if-else语句对4选1数据选择器的行为进行描述



```
module mux4to1_bh(D, S, Y);
```

```
    input [3:0] D; //输入端口
```

```
    input [1:0] S; //输入端口
```

```
    output reg Y; //输出端口及变量数据类型
```

```
    always @(D, S) //电路功能描述
```

```
        if (S == 2'b00)    Y = D[0];
```

```
        else if (S== 2'b01) Y = D[1];
```

```
        else if (S== 2'b10) Y = D[2];
```

```
        else              Y = D[3];
```

```
endmodule
```

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

2、多路分支语句（case语句）

是一种多分支条件选择语句，一般形式如下

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

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

另外，用关键词**casex**和**casez**表示含有无关项**x**和高阻**z**的情况。

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

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


例：对基本的4线-2线优先编码器的行为进行Verilog描述。

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

思考题：优先级如何体现？哪一位的优先级最高？
能否用if-else语句来实现同样的功能？

3、for循环语句

一般形式如下

```
for (initial_assignment; condition; step_assignment)  
    statement;
```

initial_assignment为循环变量的初始值。

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

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

试用Verilog语言描述具有高电平使能的3线-8线译码器.

```
module decoder3to8_bh(A,En,Y);  
    input [2:0] A;  
    input En;  
    output reg [7:0]Y;  
    integer k;    //声明一个整型变量k  
    always @(A, En) //  
        begin  
            Y = 8'b1111_1111; //设译码器输出的默认值  
            for(k = 0; k <= 7; k = k+1) //下面的if-else语句循环8次  
                {  
                    if ((En==1) && (A== k) )  
                        Y[k] = 0; //当En=1时，根据A进行译码  
                    else  
                        Y[k] = 1; //处理使能无效或输入无效的情况  
                }  
        end  
endmodule
```

循环8次

例：用条件运算符描述了一个2选1的数据选择器。

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

在连续赋值语句中，如果SEL=1，则输出L=A；否则L=B。

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

例：用数据流建模方法对2线-4线译码器的行为进行描述。

```
module decoder_df (A1,A0,E,Y);
```

```
  input A1,A0,E;
```

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

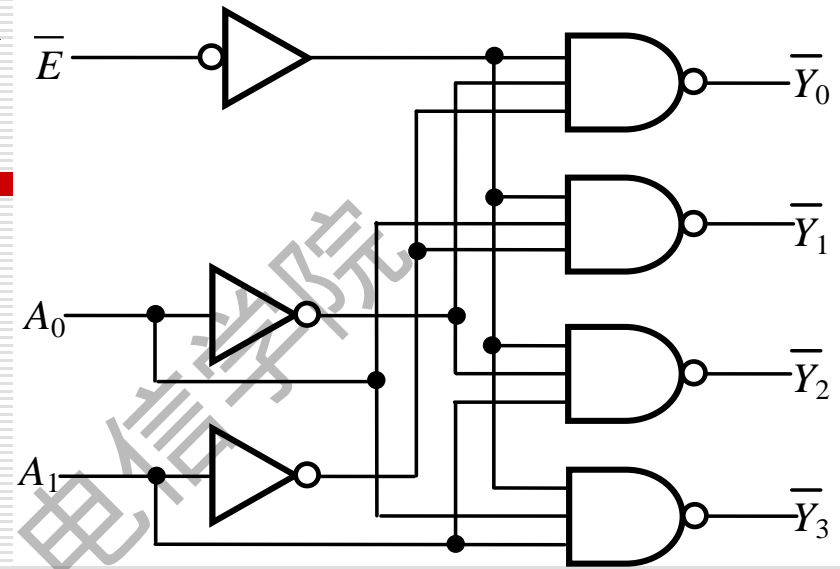
```
    assign Y[0] = ~(~A1 & ~A0 & ~E);
```

```
    assign Y[1] = ~(~A1 & A0 & ~E);
```

```
    assign Y[2] = ~(A1 & ~A0 & ~E);
```

```
    assign Y[3] = ~(A1 & A0 & ~E);
```

```
endmodule
```



4.6.2 分模块、分层次的电路设计

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

设计方法: 自顶向下和自底向上两种常用的设计方法

4位全加器的层次结构框图



```
module halfadder (S,C,A,B);
```

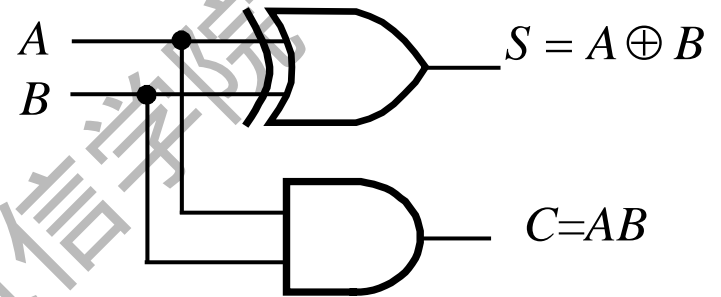
```
    input A,B;
```

```
    output S,C;
```

```
    xor (S,A,B);
```

```
    and (C,A,B);
```

```
endmodule
```



半加器的门级描述

```
module fulladder (S,CO,A,B,Ci);
```

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

```
    output S,CO;
```

```
    wire S1,D1,D2; //内部节点信号
```

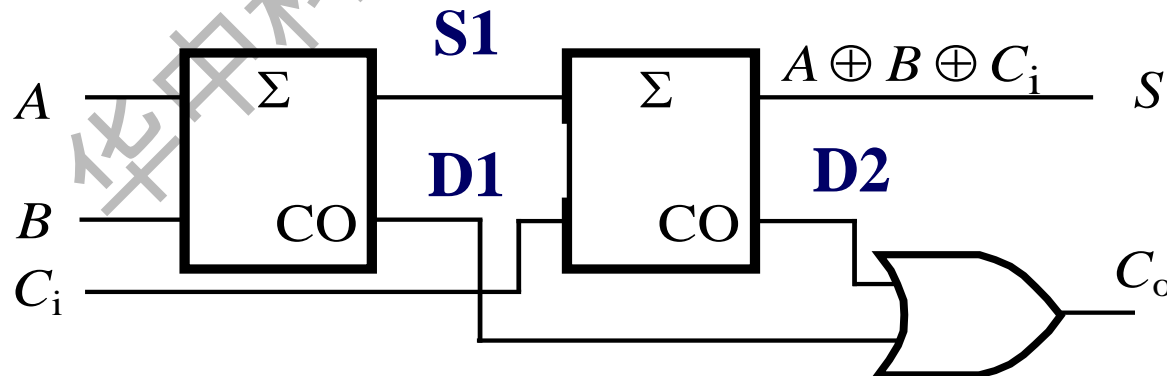
```
    halfadder HA1 (S1,D1,A,B);
```

```
    halfadder HA2 (S,D2,S1,Ci);
```

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

```
endmodule
```

全加器的描述-调用半加器




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

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

```
    input C_1;
```

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

```
    output C3;
```

```
    wire C0,C1,C2; //内部进位信号
```

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

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

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

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

4位全加器的描述
--调用1位全加器

```
endmodule
```

模块的调用方法

基本方式： 模块名 调用名（端口名表项）

调用方式一： 位置对应调用方式

调用方式二： 端口名对应调用方式

调用方式三： 存在不连接端口的调用方式

（未连**PORT**允许用（，）号空出其位置）

2. 顶层模块调用底层模块实例一通过位置关联

Order must match exactly

```
module add4 (result, carry, r1, r2, ci):  
    output [3:0] result;  
    output      carry;  
    input  [3:0] r1, r2;  
    input      ci;  
    wire [3:0] r1, r2, result;  
    wire      ci, carry, c1, c2, c3;  
    addbit u1 (r1[0], r2[0], ci, result[0], c1);  
    addbit u2 (r1[1], r2[1], c1, result[1], c2);  
    addbit u3 (r1[2], r2[2], c2, result[2], c3);  
    addbit u4 (r1[3], r2[3], c3, result[3], carry);  
endmodule
```

```
module addbit (a, b, ci, sum,co);  
    input  a, b, ci;  
    output sum, co;
```

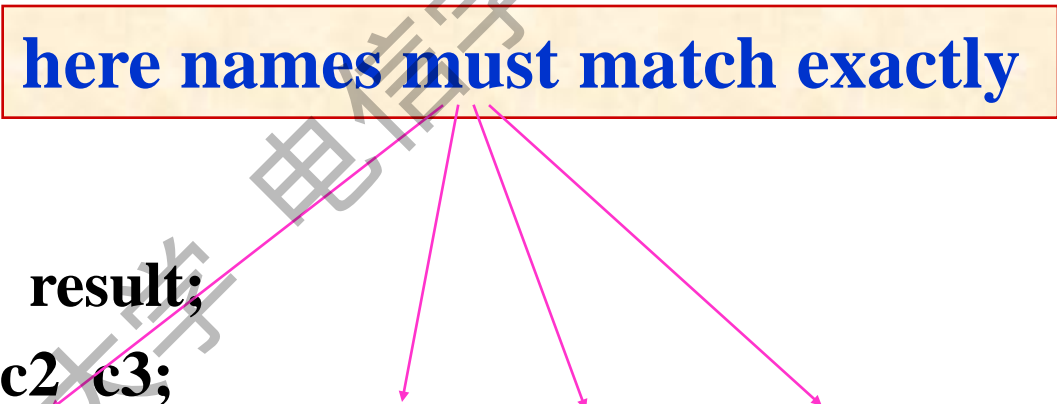
Structural or behavioral
model

```
endmodule
```

3. 顶层模块调用底层模块实例—通过名字关联

```
module add4 (result, carry, r1, r2, ci);  
    output [3:0] result;  
    output carry;  
    input [3:0] r1, r2;  
    input ci;  
    wire [3:0] r1, r2 , result;  
    wire ci, carry, c1, c2, c3;  
    addbit u0 (.co(c1) , .sum(result[0]), .ci(ci),.b(r2[0]),.a(r1[0]));  
    addbit u1 (.co(c2) , .sum(result[1]), .ci(c1),.b(r2[1]),.a(r1[1]));  
    addbit u2 (.co(c3) , .sum(result[2]), .ci(c2),.b(r2[2]),.a(r1[2]));  
    addbit u3 (.co(carry), .sum(result[3]), .ci(c3),.b(r2[3]),.a(r1[3]));  
endmodule
```

here names must match exactly

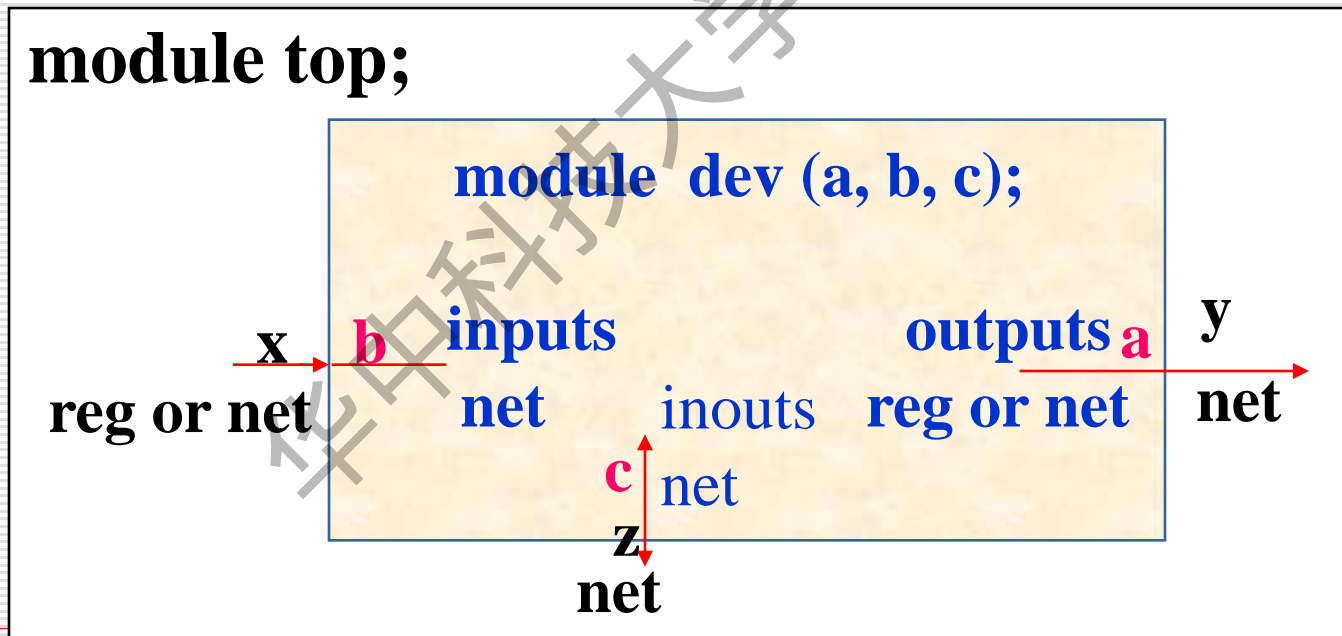


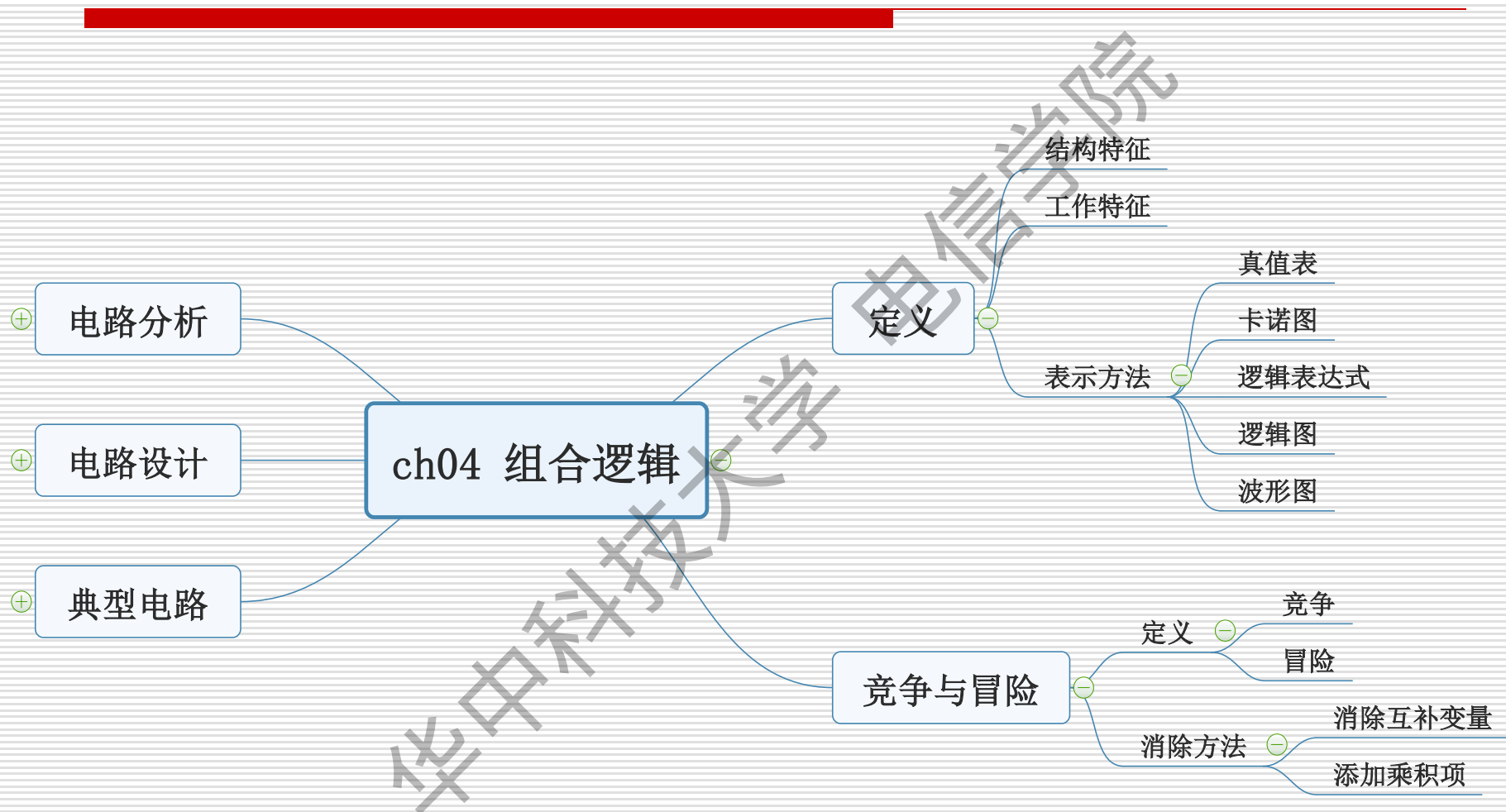
The diagram shows a yellow box with the text "here names must match exactly". Three pink arrows point from this box to the following pairs of names in the code: (ci, ci), (c1, c1), and (c2, c2). This illustrates that the names of the signals connected to the instance ports must match exactly with the names of the signals declared in the module.

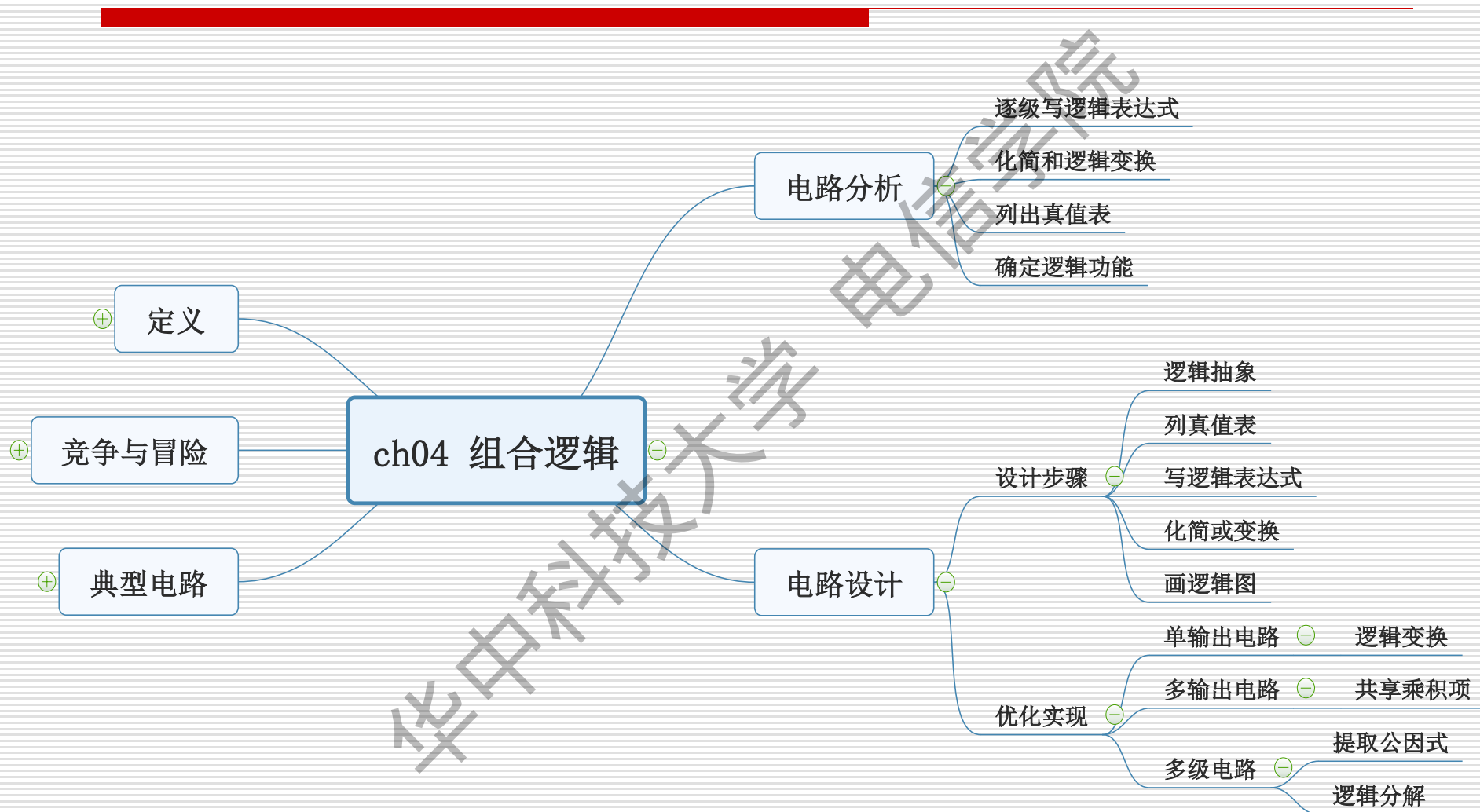
注意：该描述应严格保持名字的一致！

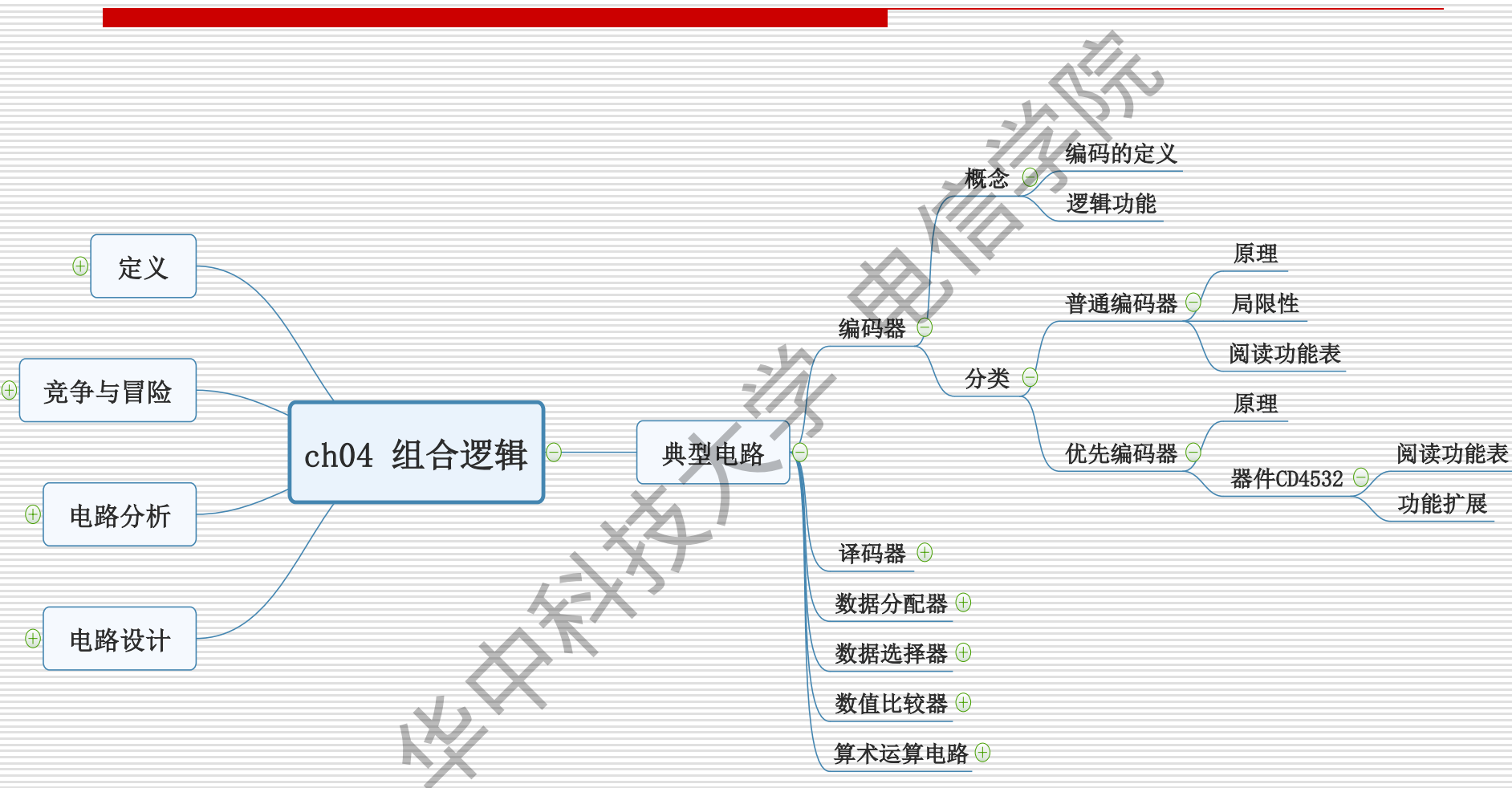
PORT连接的规则

- ***input**: 符号内部总是net, 外部可连net和reg数据类型
- ***output**: 其内部可为net或reg, 而外部必须连各种net数据类型
- ***inouts**: 它的内外都用net且只能连各种net数据类型

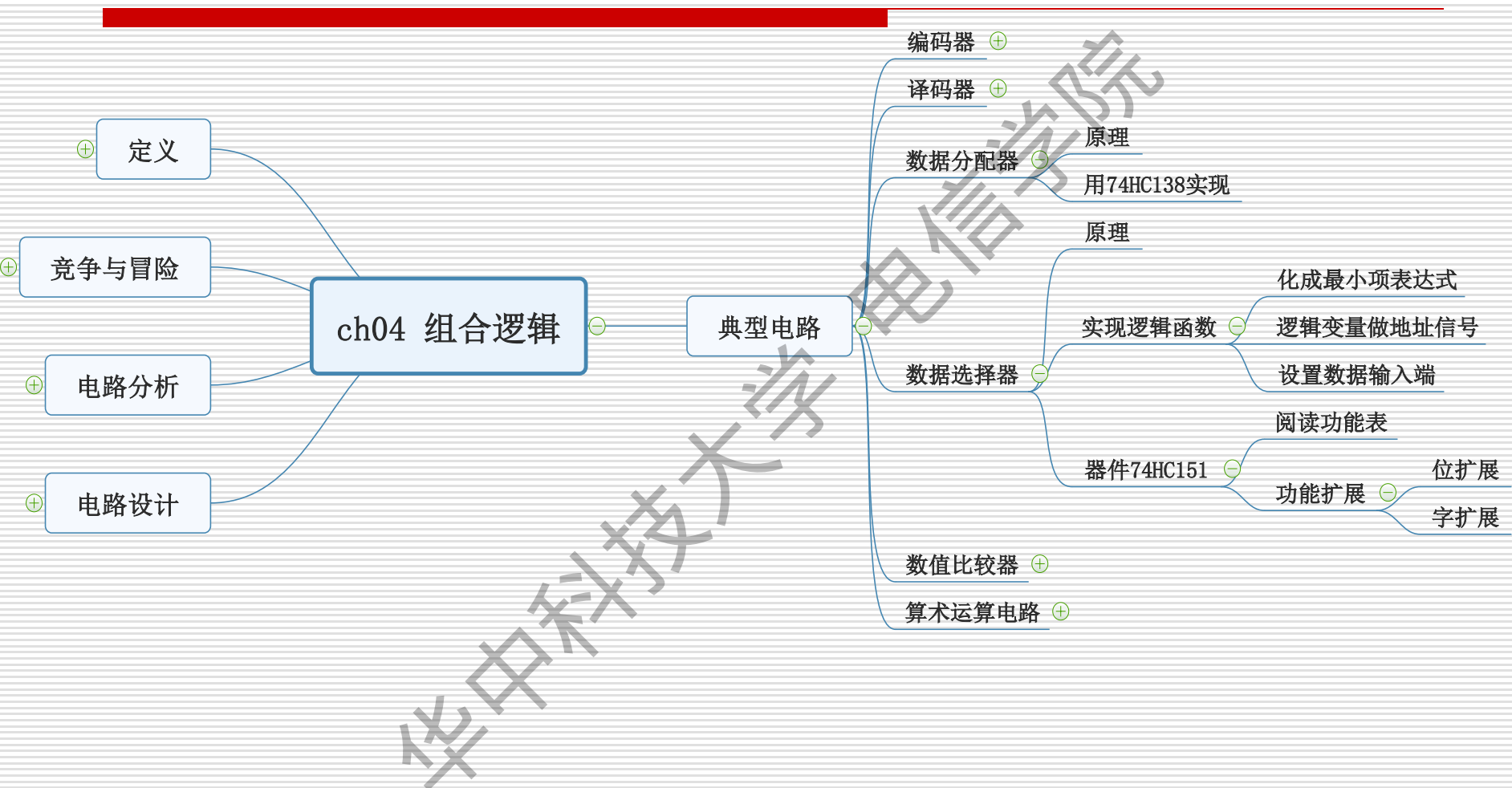




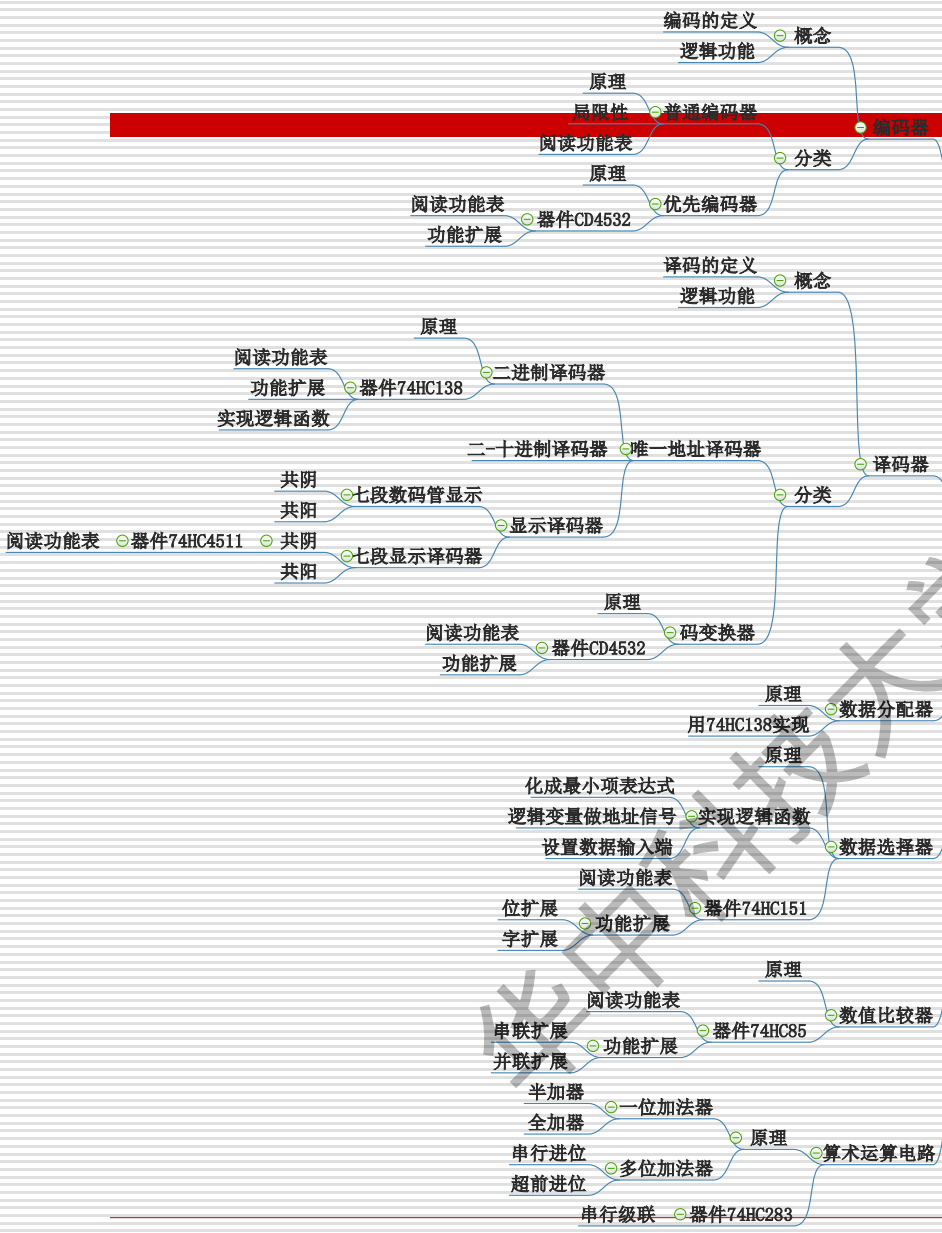












作业

➤4.1.2 ➤4.4.36

➤4.1.3 ➤4.5.1

➤4.2.7 ➤4.5.2

➤4.2.9 ➤4.6.6

➤4.4.26 ➤4.6.10

➤4.4.27

补充作业1:

- 四名学生 (A, B, C, D) 申请出国奖学金, 现有如下条件需要满足:
 - ① A和B至少有一人被选中;
 - ② A和D不能同时被选中;
 - ③ C和D中有且只有一人获得奖学金;
 - ④ B和C或者同时获得奖学金, 或者同时失去奖学金;
- 请写出申请结果的逻辑表达式并化简为最简与或式。

补充作业2:

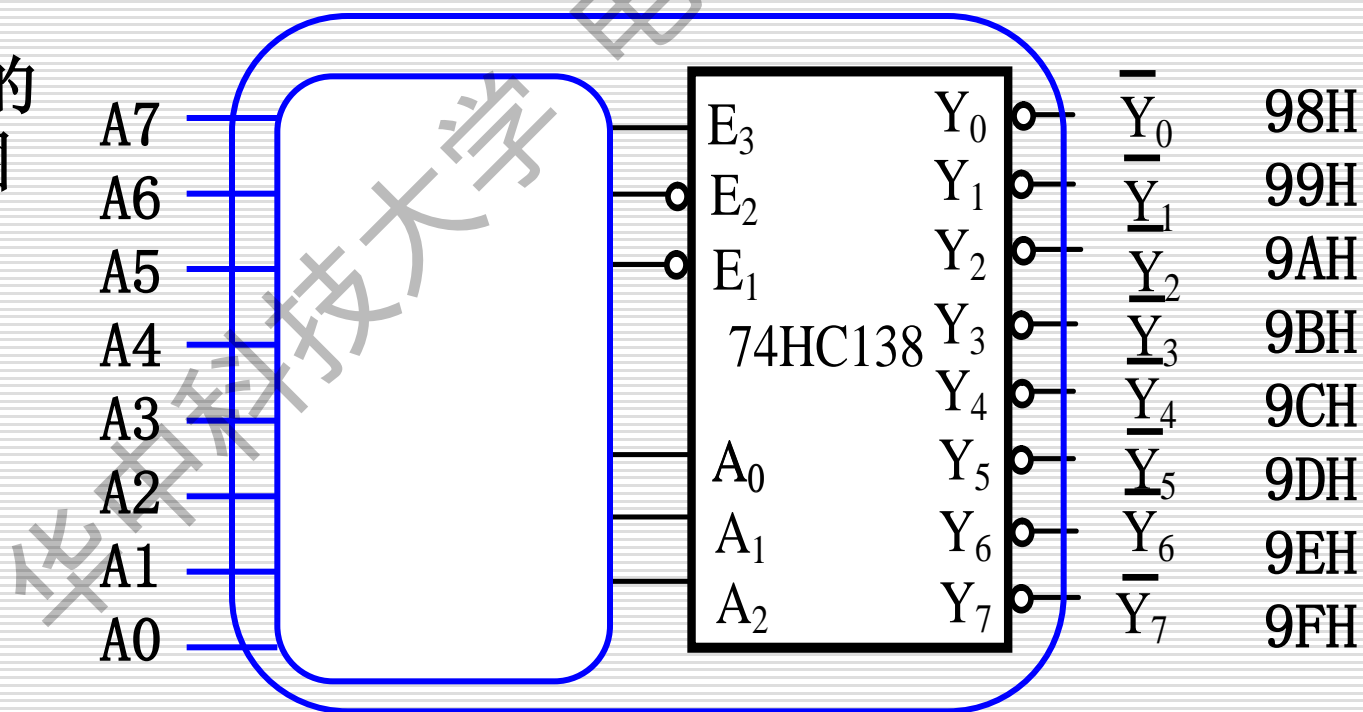
用两片74138设计4—16线译码器，说明设计原理并画出逻辑电路图

补充作业3:

仅用一片74LS138设计地址译码器，译出输入地址 $A_7, \dots, A_0 = 98H, \dots, 9FH$ 。

思路:

8根输入地址线的可能取值范围是00H...FFH



补充作业4:

仅用全加器组成八位二进制代码奇校验器，
电路应如何连接？

补充作业5:

试仅用一片74LS151，不加任何门电路
实现逻辑函数：

$$F(ABCD) = \sum m(2,5,6,7,8,10,11,12,14,15)$$

华中科技大学

补充作业6:

- 分别用74HC151/74HC138设计全加器，说明设计原理并画出逻辑电路图

华中科技大学