

数字集成电路基础

作者：Pannenets.F

时间：November 3, 2020

Je reviendrai et je serai des millions. ——«Spartacus»

第一章 数制



数字集成电路基础

张悦

微电子学院

费尔北京研究院 / 自旋电子交叉学科中心

2020-10-2





课程信息

- 教师：张悦、康旺
- 助教：王硕、魏少芊
- 学时安排（56学时）
 - 课堂授课（48学时）
 - 习题/讨论/课堂测试（4学时）
 - 总复习课（2学时）
 - 考试（2学时）

考核方法

- 平时成绩（考勤+交流+作业）+期末考试



课程内容

- 一、数制与编码（2课时）
- 二、布尔代数与逻辑函数（6课时）
- 三、逻辑门电路（5课时）
- 四、组合逻辑电路（5课时）
- 五、触发器（6课时）
- 六、时序逻辑电路（10课时）
- 七、存储器（4课时）
- 八、可编程逻辑器件（2学时）
- 九、脉冲波形的产生与变换（6学时）
- 十、数模和模数转换（2学时）



学习方法

- 知识结构

- **逻辑代数**是理论基础，熟练掌握
- **单元电路**是物理基础，掌握逻辑功能、外部特性、功能扩展和使用方法
- 掌握数字电路系统的**分析方法**和**设计方法**

- 学习方式

- 课堂讲授
- 主题研讨与小测试
- 过程强调交互

- 学习要求

- 预习和复习
- 主动提出问题，通过讨论加深理解
- 将每道习题都作为设计项目



课内参考教材（中文）：

1. 阎 石 主编：数字电子技术基础（第五版），
高等教育出版社。（面向二十一世纪教材）

课内参考教材（外文）：

1. **Digital Logic Circuit Analysis and Design**

Victor P.Nelson 等著

清华大学出版社（英文影印版）

2. **Digital Fundamentals (Seventh Edition)**

Thomas L.Floyd 著

科学出版社（英文影印版）

集成电路发展

第一台计算机

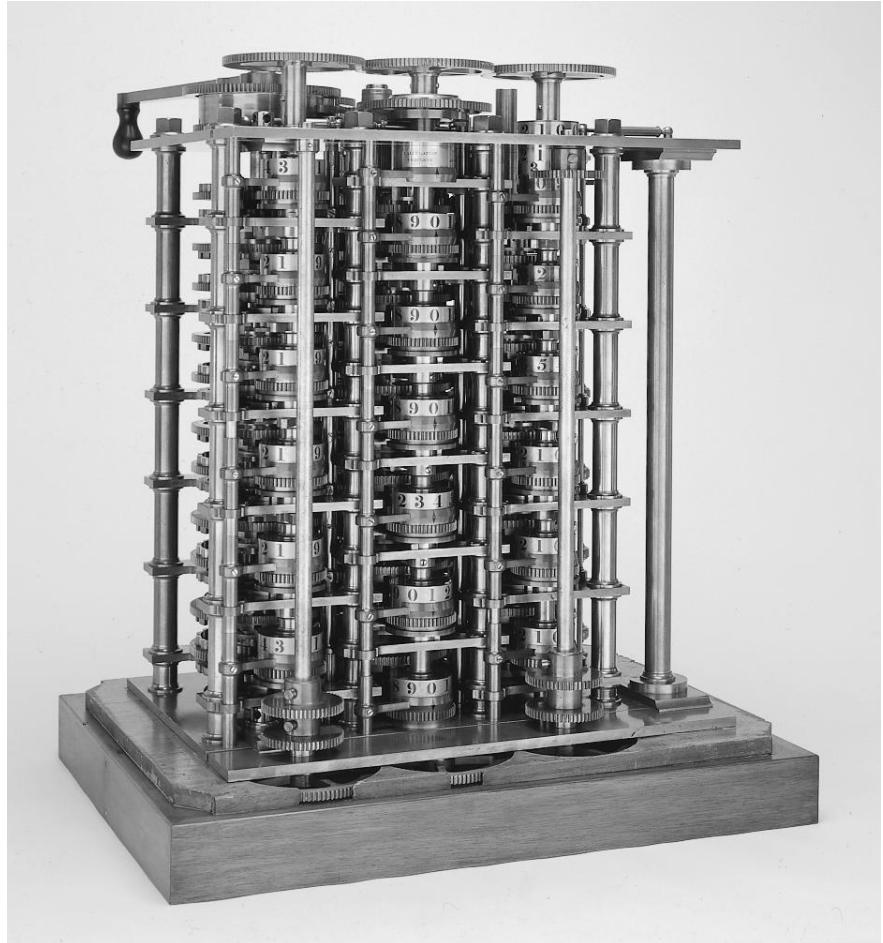
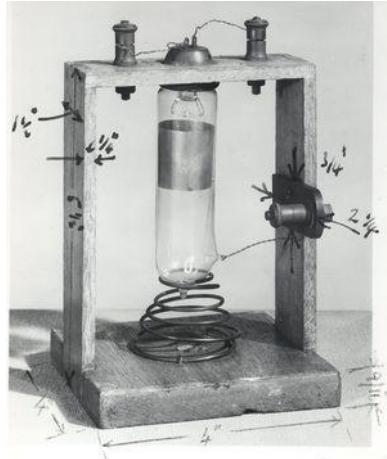


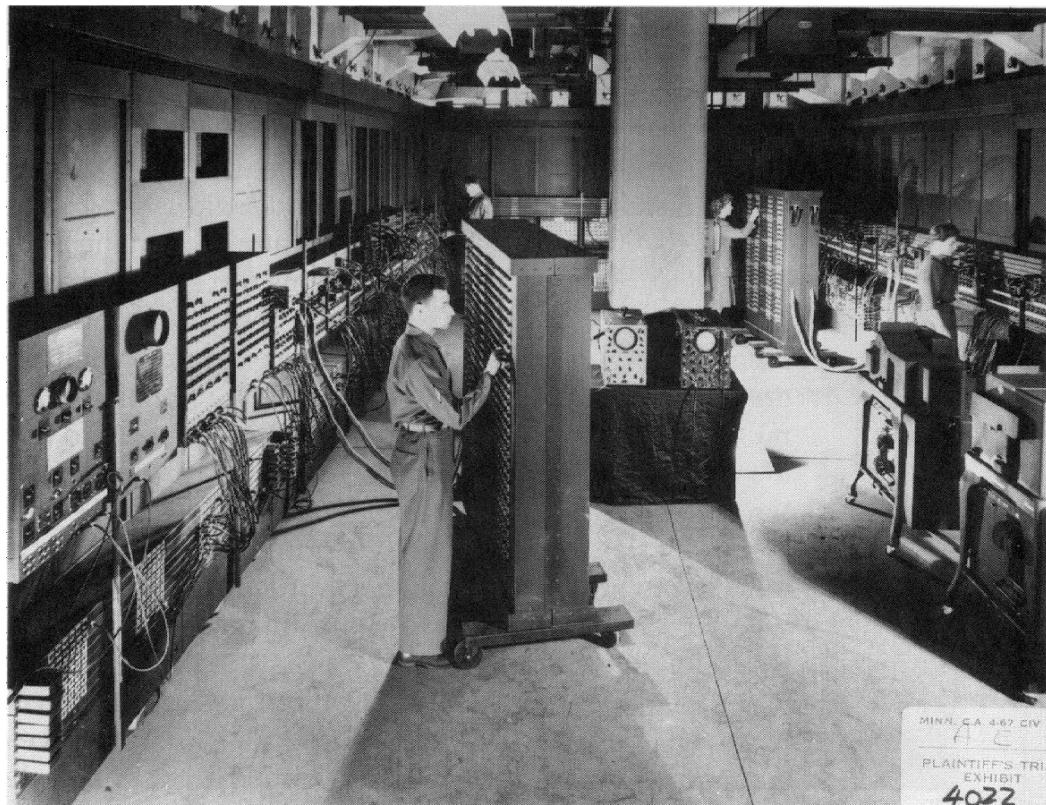
图1.1 世界上已知的第一个自动计算器
Babbage的Difference Engine I
(1832年)的工作部件
(摘自 [Swade93] , 由伦敦科学博物馆提供)

集成电路发展

ENIAC – 第一台电子计算机 (1946)



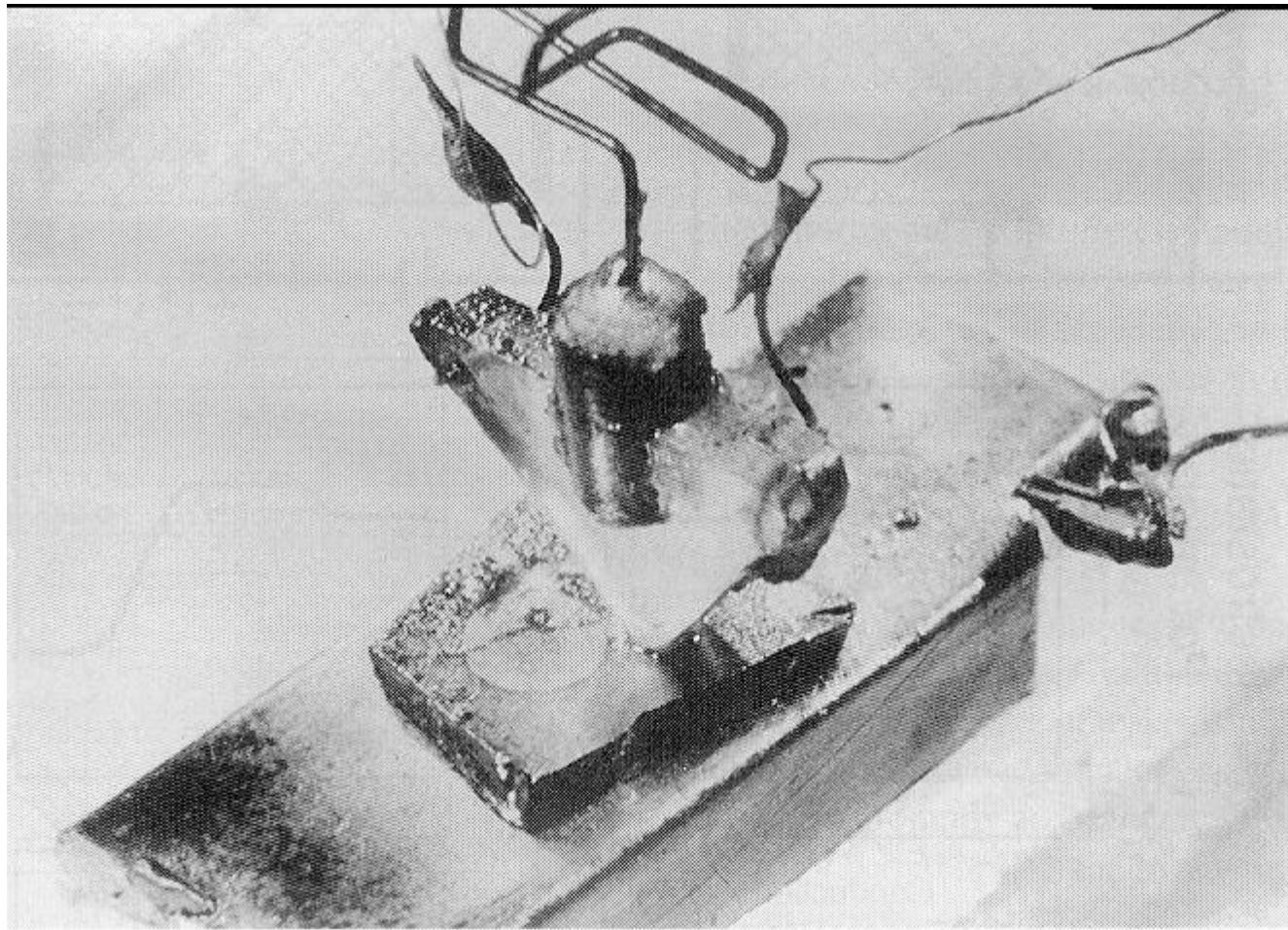
电子管



17468个电子管， 150KW

集成电路发展

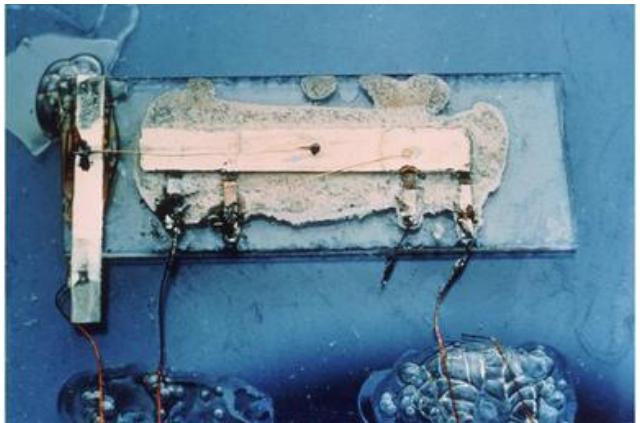
晶体管革命



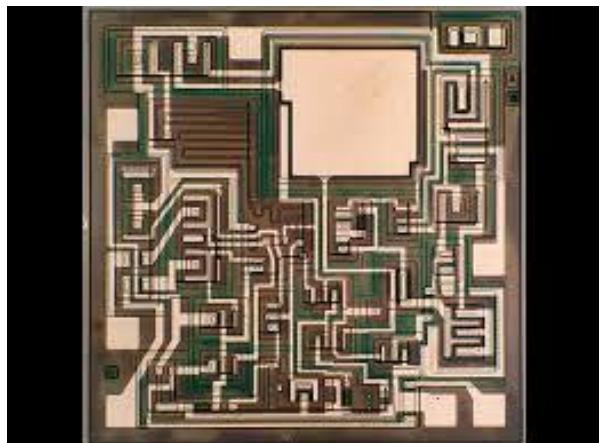
第一支晶体管
贝尔实验室, 1947

集成电路发展

第一个集成电路



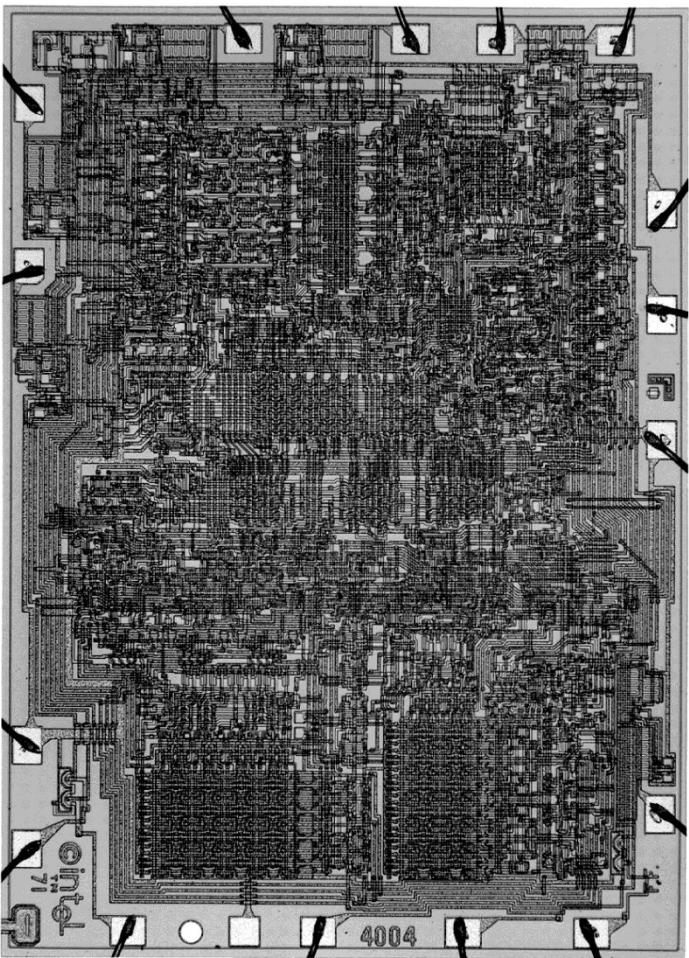
Jack Kilby (2000 Nobel Prize)
Texas Instruments (TI) 1958



741运算放大器电路
仙童公司 1963

集成电路发展

Intel 4004 微处理器

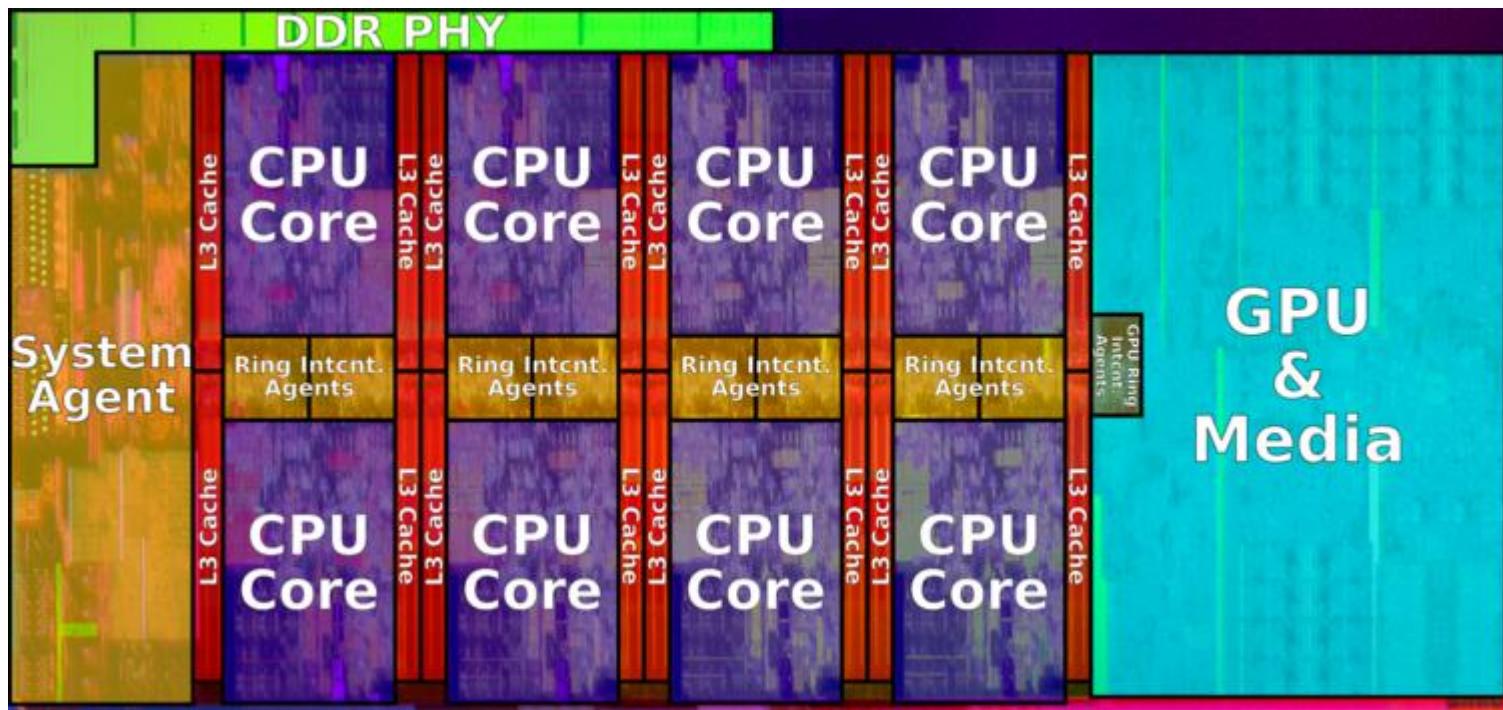


1971
1000个晶体管
1 MHz操作频率

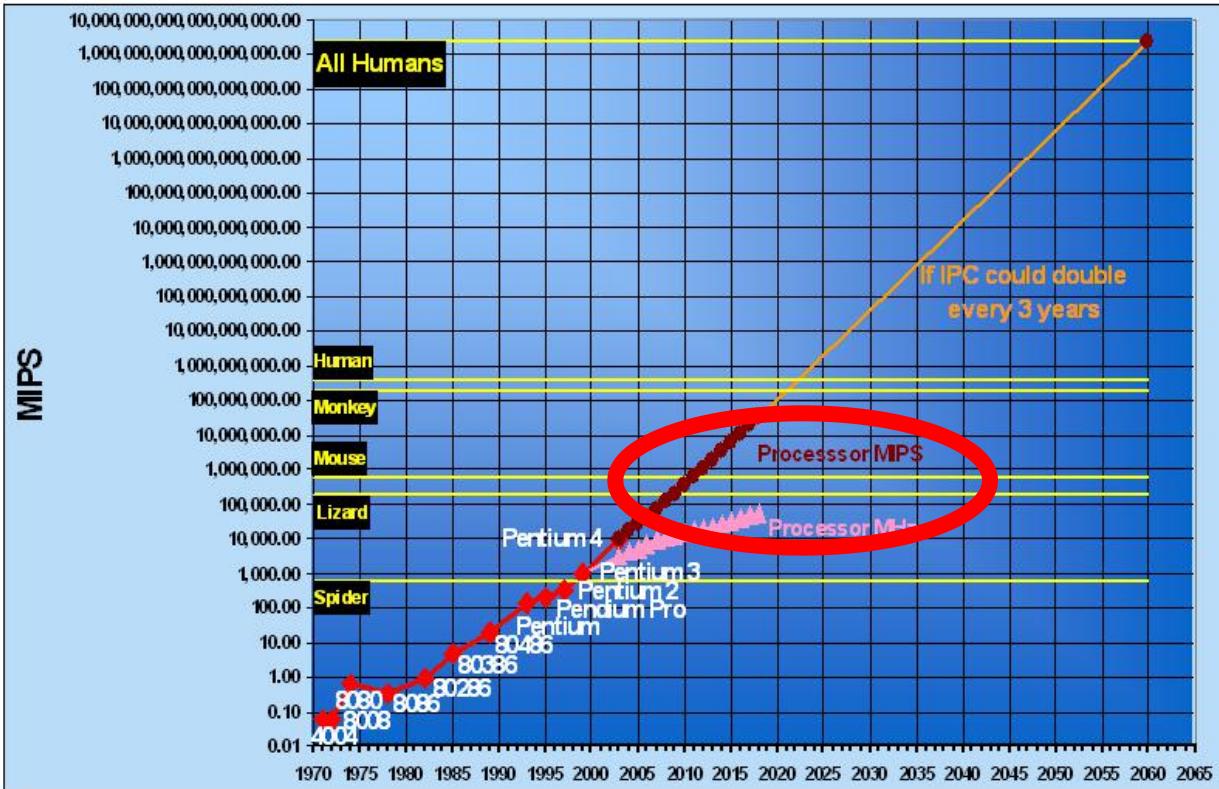
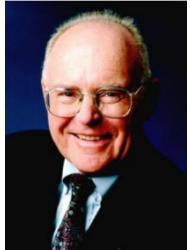
集成电路发展

2019

> 70亿个晶体管
5 GHz 单核操作频率



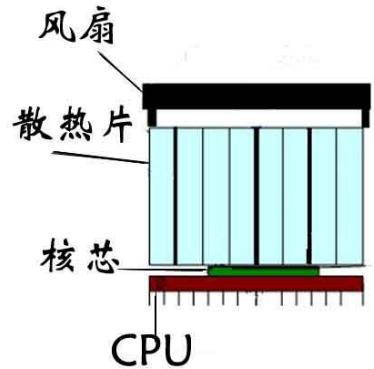
摩尔定律



摩尔定律是由英特尔（Intel）创始人之一戈登·摩尔（Gordon Moore）提出来的。其内容为：当价格不变时，集成电路上可容纳的元器件的数目，约每隔18-24个月便会增加一倍，性能也将提升一倍。换言之，每一美元所能买到的电脑性能，将每隔18-24个月翻一倍以上。这一定律揭示了信息技术进步的速度。

散热是个大问题！

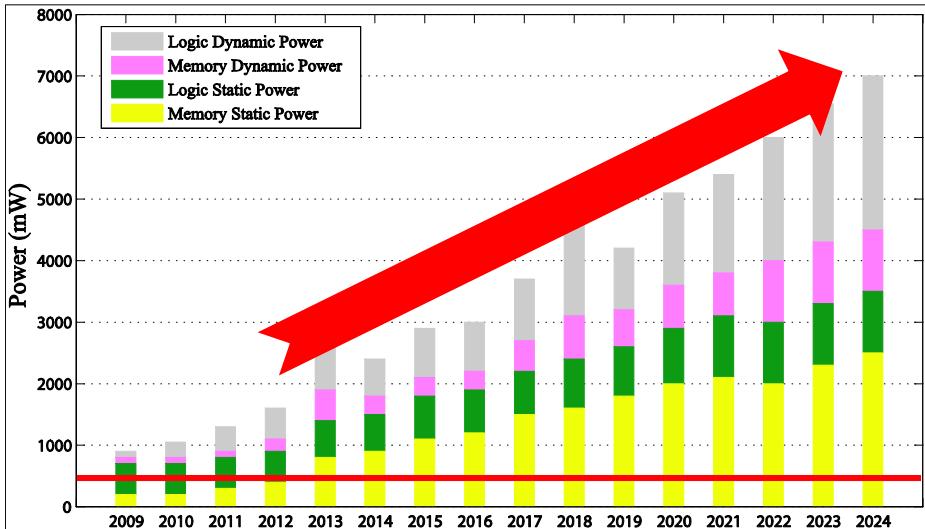
CPU



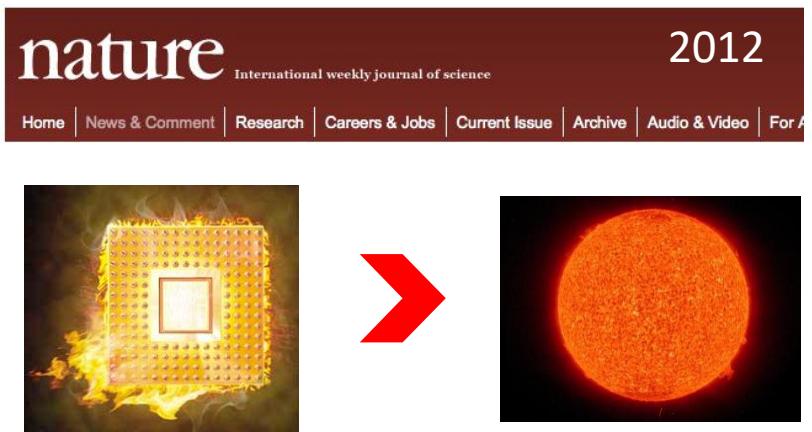
GPU



功耗问题



功耗持续增加！！



NATURE | NEWS FEATURE

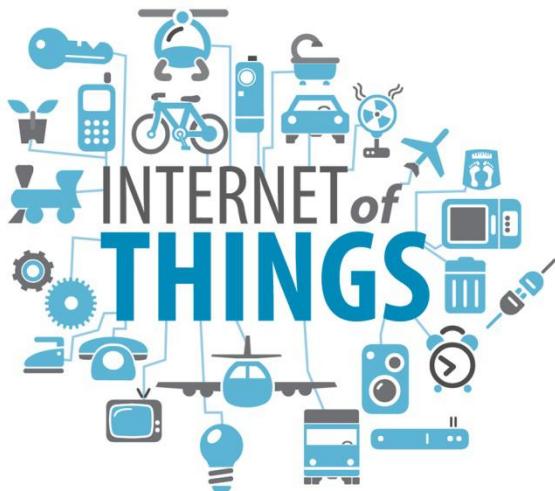
عربي

The chips are down for Moore's law

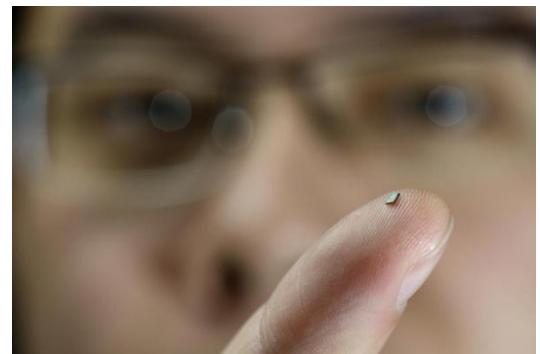
The semiconductor industry will soon abandon its pursuit of Moore's law. Now things could get a lot more interesting.

物联网-IoT时代

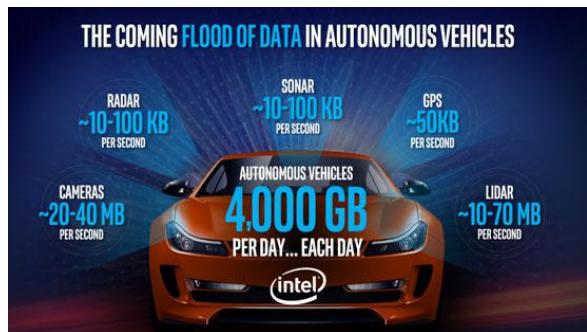
第三次信息革命



物联网终端——
形形色色的微型电路



5G



数字电路

模拟电路与数字电路

- 模拟信号

- 时间上连续 或 数值上连续的信号
- 来自于自然界客观存在的物理量

- 模拟电路

- 处理模拟信号的电路

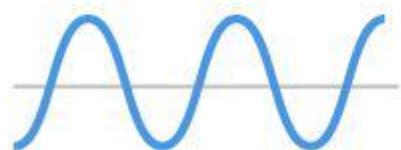
- 数字信号

- 时间和数值均离散的信号
- 例如：电子表的计时信号、流水线上的零件数

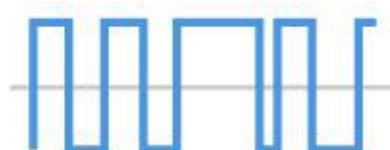
- 数字电路：

- 处理数字信号的电路

模拟



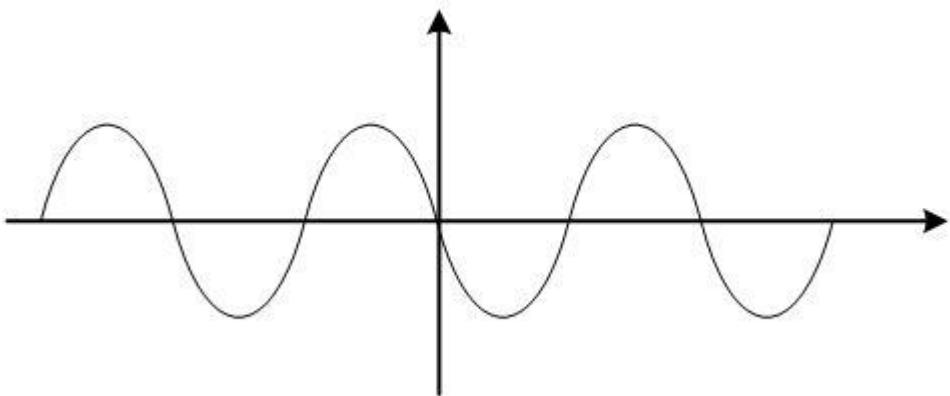
数字



数字电路

模拟电路与数字电路

1) 信号不同



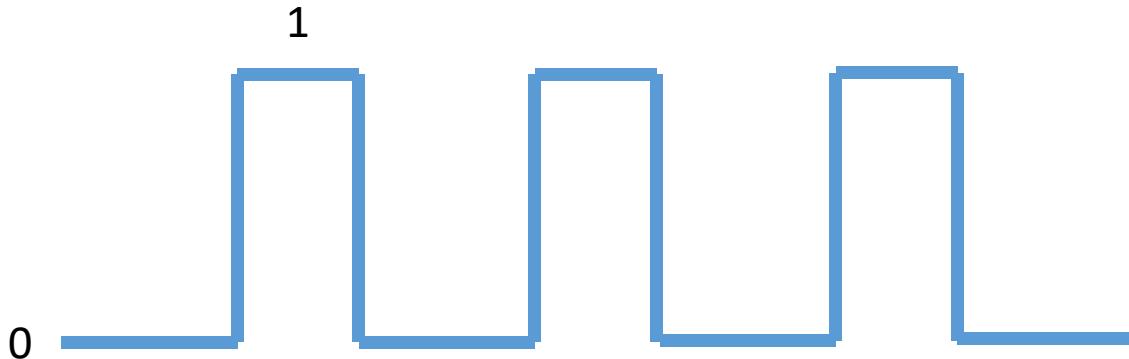
$$y = A \sin(\omega t)$$

正负峰值之间连续取值

数字电路

模拟电路与数字电路

1) 信号不同



在电路中用低、高电平表示0、1两种逻辑状态

逻辑电平与电压值的关系（正逻辑）

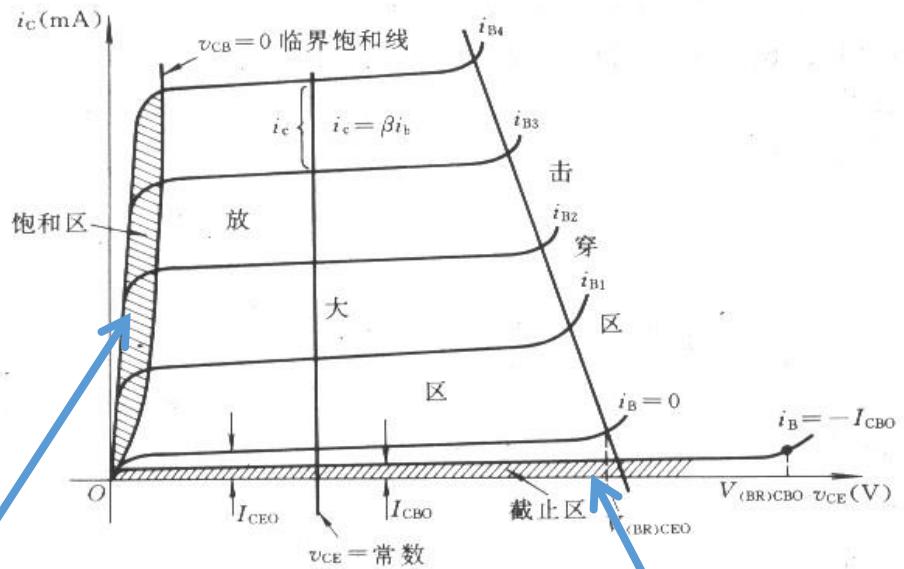
电压(V)	二值逻辑	电 平
+5	1	H(高电平)
0	0	L(低电平)

数字电路

模拟电路与数字电路

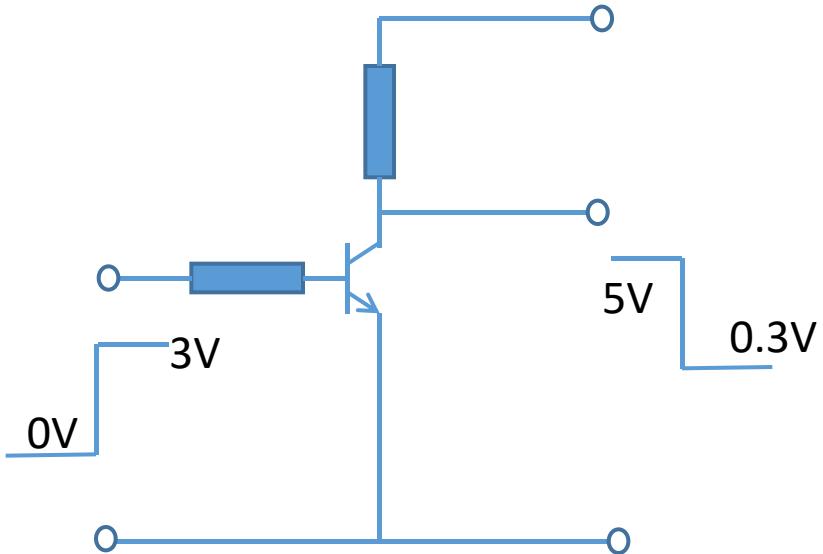
2) 晶体管的工作状态不同

在数字电路中，晶体管工作在**开关状态**



开态——饱和

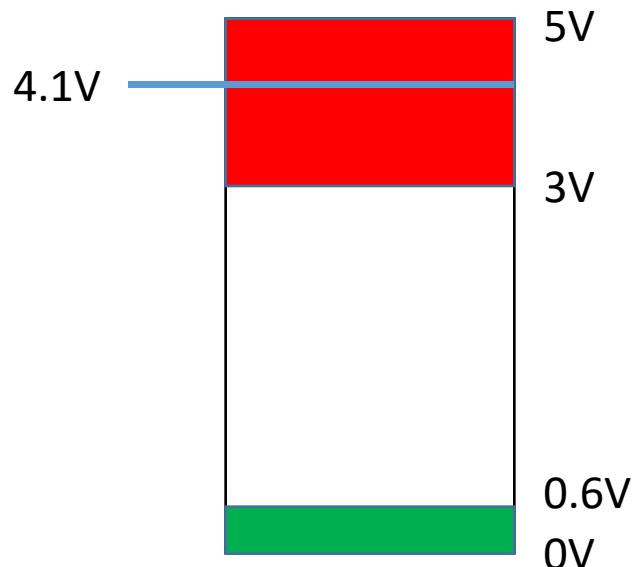
关态——截止



模拟电路与数字电路

3) 抗干扰能力不同

数字电路中的高低电平都指的是一定的电压范围，在所受到的干扰不足以改变信号的状态时，不影响电路的正常工作。





数字电路

模拟电路与数字电路

4) 分析方法不同

模拟电路

微变等效电路

——电路分析

性能指标

数字电路

逻辑分析方法

数学工具：
布尔代数

描述方法：
真值表
表达式
功能表等

功能实现





数字集成电路基础

——数制与编码

张悦

微电子学院

费尔北京研究院 / 自旋电子交叉学科中心

2020-10-2





数制

数制:多位数码中的每一位数的构成及低位向高位进位的规则

1. 常用数制

1. 1 十进制

- (1) 计数符号: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.
- (2) 进位规则: 逢十进一.

例: $1983.62 = 1 \times 10^3 + 9 \times 10^2 + 8 \times 10^1 + 3 \times 10^0 + 6 \times 10^{-1} + 2 \times 10^{-2}$

- (3) 十进制数按权展开式

$$(N)_{10} = \sum_{i=-m}^{n-1} a_i \times 10^i$$

系数  权 



数制

1. 2. 二进制

- (1) 计数符号: 0, 1 .
- (2) 进位规则: 逢二进一.
- (3) 二进制数按权展开式

$$(N)_2 = \sum_{i=-m}^{n-1} a_i \times 2^i$$

数字电路中采用二进制的原因:

- 1) 数字装置简单可靠;
- 2) 二进制数运算规则简单;
- 3) 数字电路既可以进行算术运算, 也可以进行逻辑运算.



数制

1.3. 十六进制和八进制

(1) 十六进制数计数符号:

(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15)

(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F)

(2) 十六进制数进位规则: 逢十六进一.

(3) 按权展开式:

$$(N)_{16} = \sum_{i=-m}^{n-1} a_i \times 16^i$$

例: $(6D.4B)_{16} = 6 \times 16^1 + D \times 16^0 + 4 \times 16^{-1} + B \times 16^{-2}$

$$= 6 \times 16^1 + 13 \times 16^0 + 4 \times 16^{-1} + 11 \times 16^{-2}$$

数制

(1) 八进制数计数符号: 0, 1, . . . 6, 7.

(2) 八进制数进位规则: 逢八进一.

(3) 按权展开式:

$$(N)_8 = \sum_{i=-m}^{n-1} a_i \times 8^i$$

例:

$$(63.45)_8 = 6 \times 8^1 + 3 \times 8^0$$

$$+ 4 \times 8^{-1} + 5 \times 8^{-2}$$

不同进制数的对照表

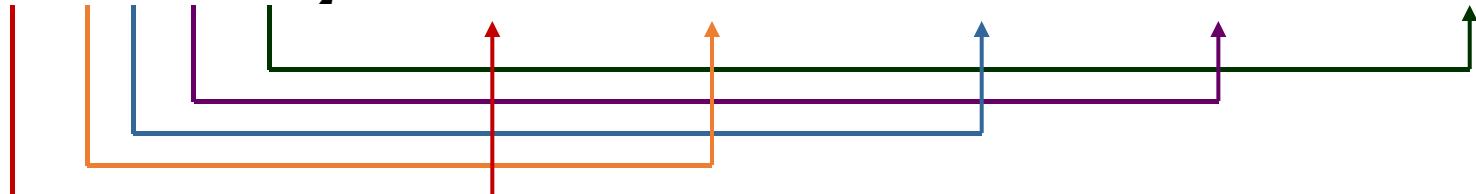
十进制数	二进制数	八进制数	十六进制数
00	0000	00	0
01	0001	01	1
02	0010	02	2
03	0011	03	3
04	0100	04	4
05	0101	05	5
06	0110	06	6
07	0111	07	7
08	1000	10	8
09	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F
16	10000	20	20
17	10001	21	21
18	10010	22	22
19	10011	23	23
20	10100	24	24

数制转换

- 数值相等，计数方法（数制）不同，
- 本质：权值的转换
- 数制转换之任意进制到十进制的转换

利用任意进制数的按权展开式，可以将一个任意进制数转换成等值的十进制数。

例： $(1011.101)_2 = 1 \times 2^3 + 1 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1} + 1 \times 2^{-3}$



$$\begin{aligned}
 &= 8 + 2 + 1 + 0.5 + 0.125 \\
 &= (11.625)_{10}
 \end{aligned}$$

数制转换

例: $(8FA.C)_{16} = 8 \times 16^2 + F \times 16^1 + 10 \times 16^0 + 12 \times 16^{-1}$
 $= 2048 + 240 + 10 + 0.75 = 2298.75$

2.1 十进制数转换为二进制数

2.1.1 十进制整数的转换

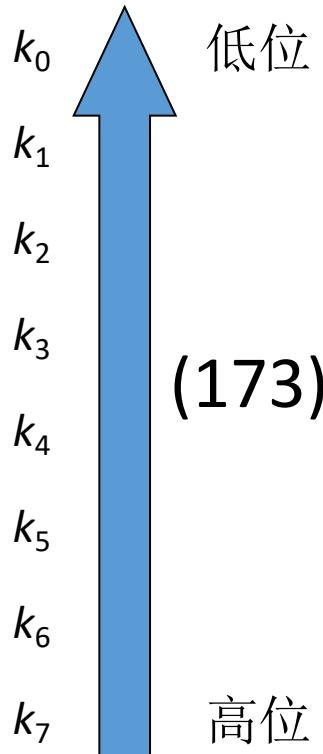
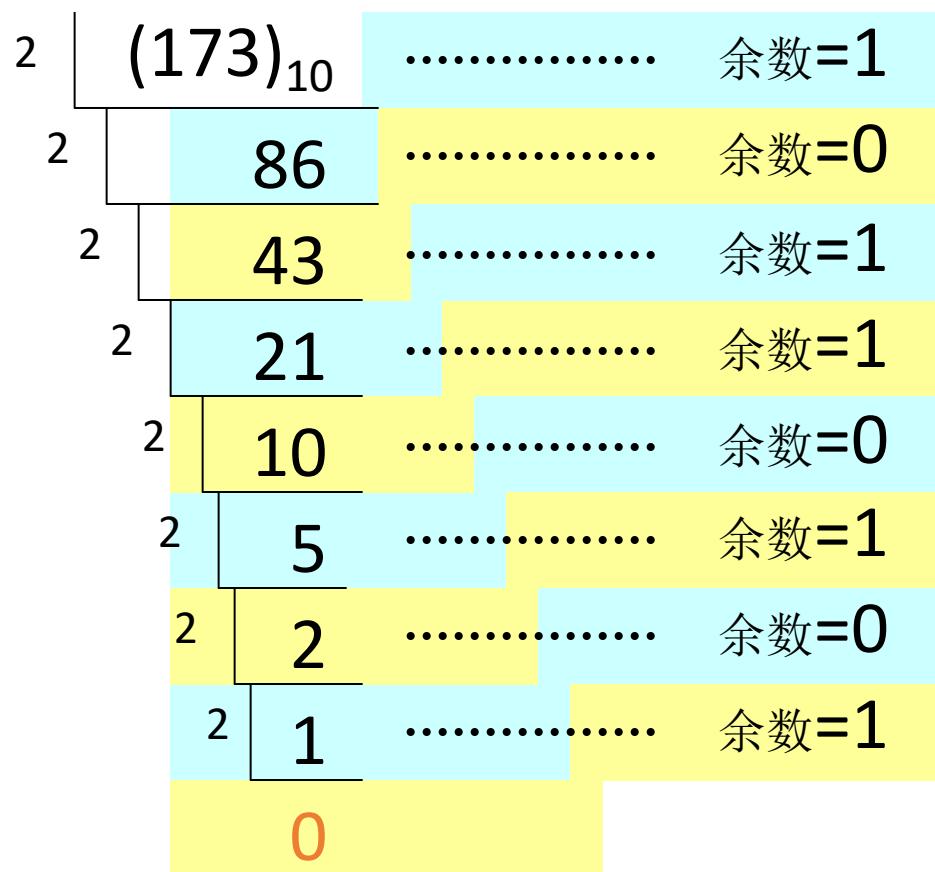
$$k_{n-1} \times 2^{n-3} + k_{n-2} \times 2^{n-4} + \dots + k_2 \times 2^0 + \underbrace{k_1 / 2}_{\text{余数为 } k_1}$$

余数为 k_0

依次类推, “除2取余” 法

数制转换

例：(173)



$$(173)_{10} = (1010\ 1101)_2$$

数制转换

2.1.2 十进制小数的转换

考虑：

$$(D)_{10} = k_{-1} \times 2^{-1} + k_{-2} \times 2^{-2} + \dots + k_{-(m-1)} \times 2^{-(m-1)} + k_{-m} \times 2^{-m}$$

$$2 \times (D)_{10} = \underbrace{k_{-1} + k_{-2} \times 2^{-1} + \dots + k_{-(m-1)} \times 2^{-(m-2)}}_{\text{整数部分为 } k_{-1}} + k_{-m} \times 2^{-(m-1)}$$

依次类推，“乘2取整”法

例：(0.6875)

$$\begin{array}{r} 0.6875 \\ \times 2 \\ \hline k_{-1} \quad 1 \end{array} \cdots \cdots \cdots \quad 1.3750$$

$$\begin{array}{r} 0.3750 \\ \times 2 \\ \hline k_{-2} \quad 0 \end{array} \cdots \cdots \cdots \quad 0.7500$$

$$\begin{array}{r} 0.7500 \\ \times 2 \\ \hline k_{-3} \quad 1 \end{array} \cdots \cdots \cdots \quad 1.5000$$

$$\begin{array}{r} 0.5000 \\ \times 2 \\ \hline k_{-4} \quad 1 \end{array} \cdots \cdots \cdots \quad 1.0000$$

$$(0.6875)_{10} = (0.1011)_2$$



数制转换

2.2 二进制与十六进制数间转换

- 数制转换之“二 - 十六”进制的转换

- “分组对应”法

- 由于4位二进制数恰好有16个状态，而将这4位二进制数看作一个整体时，它的进位输出又正好是逢十六进一。

$$(1011101.101001)_2 = (\underline{0101} \ \underline{1101}. \underline{1010} \ \underline{0100})_2 \\ = (5D.A4)_{16}$$

- 数制转换之“十六 - 二”进制的转换

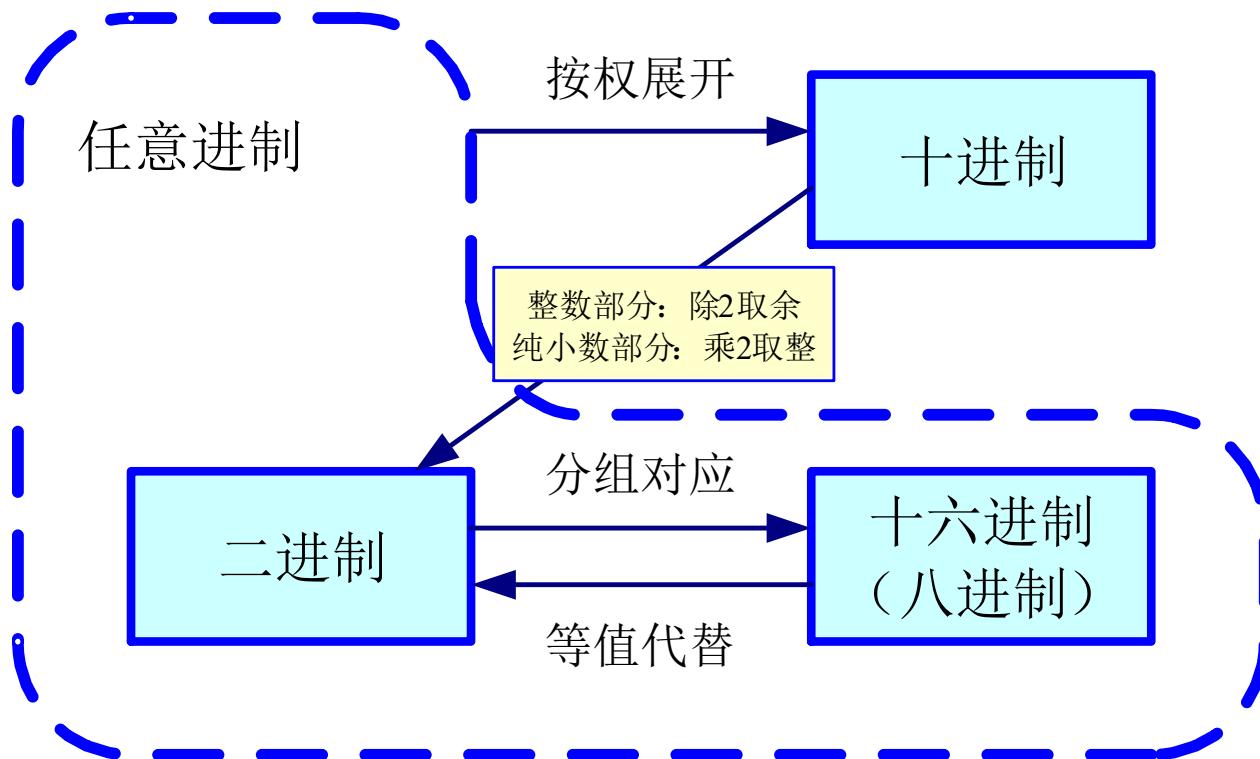
- 将十六进制数的每一位用等值的4位二进制数代替

- 例：(8 FA.C6)₁₆

$$(\ 8 \quad F \quad A \ . \ C \quad 6)_{16} \\ (\ \underline{1000} \ \underline{1111} \ \underline{1010} \ . \ \underline{1100} \ \underline{0110})_{16}$$

数制转换

2.3 小结：进制之间的转换方法



算数运算

3.1 二进制正负数的表示及运算

■ 二进制四则运算

加

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

乘

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

除

$$\begin{array}{r}
 1.1 \ 1 \ \dots \\
 \hline
 0 \ 1 \ 0 \ 1 \left| \begin{array}{r} 1 \ 0 \ 0 \ 1 \\ 0 \ 1 \ 0 \ 1 \\ \hline 1 \ 0 \ 0 \ 0 \\ 0 \ 1 \ 0 \ 1 \\ \hline 0 \ 1 \ 1 \ 0 \\ 0 \ 1 \ 0 \ 1 \\ \hline 0 \ 0 \ 1 \ 0 \end{array} \right.
 \end{array}$$

减

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

移位与加法

移位与减法

算数运算

■ 二进制正负数的表示

➤ 二进制的原码、反码及补码（有符号二进制数）

✓ 正数的三种表示法一样：

符号位为0，随后是二进制的绝对值，即（正数）“原码”。

■ 原码

- 最高位表示正、负号
- 0表示正，1表示负
- 其余各位表示数的绝对值

例：

（设：为8-bit有符号数）

$$[(+43)_{10}]_{\text{原}} = 00101011$$

$$[(-43)_{10}]_{\text{原}} = 10101011$$



■ 负数的原码、反码和补码表示方法

➤ 原码 例: $[-25]_{\text{原}} = 10011001$

➤ 反码 例: $[-25]_{\text{反}} = 11100110$

- 负数的反码是对正数的编码（正数原码）

取反；

- 注：绝对值位域取反，符号位为“1”；
也可以认为是对整个码字逐位取反。

➤ 补码 例: $[-25]_{\text{补}} = 11100111$

- 计算方法

- ❖ “反码加1”

确切地说：将相反符号数（这里指“正数原码”）
的码字含符号位逐位取反，然后从最低位加1。

算数运算

➤ 补码的算例：

设以8-bit存储有符号整数，最高位为符号位

1. 求 $[-39]_{10}$ 补，即以补码表示 $(-39)_{10}$
2. 给定补码为 $[11101010]_{\text{补}}$ ，求该数，以十进制表示

解 1：

绝对值的补（原）码表示为： $(+39)_{10} = 00100111$

原码取反： $[-39]_{10}^{\text{反}} = 11011000$

“反码加1” $[-39]_{10}^{\text{补}} = 11011001$

解 2：“符号位”为“1”，说明为负数，则：

求相反符号数（负负为正）的补码表示——“反码加1”

答案： $[11101010]_{\text{补}} = (-0010110)_2 = (-22)_{10}$



算数运算

• 补码——加减法运算

- 负数采用补码表示后，就可以把减法转换为加法

例： $39-22=39+(-22)=17$

• 注： $(+39)_{10}$ $(+22)_{10}$ $[(-22)_{10}]_{\text{补}}$

• 注意：补码加减法运算应在相应位数表示的数值范围内进行。

$(0010\ 0111)_2$ $(0001\ 0110)_2$ $(1110\ 1010)_2$

原码：

$$\begin{array}{r} 010\ 0111 \\ - 001\ 0110 \\ \hline \end{array}$$

要保证被减数
不小于减数；
否则调换次序。

判断得到运算
结果的符号。 自动丢弃

补码：

$$\begin{array}{r} 0010\ 0111 \\ + 1110\ 1010 \\ \hline \end{array}$$

$001\ 0001 = (+17)_{10}$

1 0001 0001





码制

码制: 编制代码所要遵循的一定的规则

4.1 二—十进制代码 (BCD码) (**Binary Coded Decimal codes**)

用四位二进制代码来表示一位十进制数码, 这样的代码称为二-十进制码, 或BCD码.

四位二进制有16种不同的组合, 可以在这16种代码中任选10种表示十进制数的10个不同符号, 选择方法很多. 选择方法不同, 就能得到不同的编码形式.



码制

8421码又称BCD（Binary Coded Decimal）码，是十进制代码中最常用的二种。在这种编码方式中，每一位二值代码的1都代表一个固定数值，将每一位的1代表的十进制数加起来，得到的结果就是它所代表的十进制数码。由于代码中从左到右每一位的1分别表示8、4、2、1，所以将这种代码称为**8421码**。每位的1代表的十进制数称为这一位的权。**8421码**中每一位的权是固定不变的，它属于恒权代码。

5211码每位的权正好与**8421码**十进制计数器4个触发器输出脉冲的分频比相对应。这种对应关系在构成某些数字系统时很有用。

2421码是一种恒权代码，它的0和9、1和8、2和7、3和6、4和5也互为反码，这个特点和余3码相仿。

余3码的编码规则与**8421码**不同，如果把每一个余3码看作4位二进制数，则它的数值要比它所表示的十进制数码多3，故而将这种代码称为余3码。如果将两个余3码相加，所得的和将比十进制数和所对应的二进制数多6。因此，在用余3码做十进制加法运算时，若两数之和为10，正好等于二进制数的16，于是便从高位自动产生进位信号。此外，从表1.5.1中还可以看出，0和9、1和8、2和7、3和6、4和5的余3码互为反码，这对于求取对10的补码是很方便的。余3码不是恒权代码。如果试图将每个代码视为二进制数，并使它等效的十进制数与所表示的代码相等，那么代码中每一位的1所代表的十进制数在各个代码中不能是固定的。

余3循环码是一种变权码，每一位的1在不同代码中并不代表固定的数值。它的主要特点是相邻的两个代码之间仅有位的状态不同。



码制

十进制	8421码	5211码	2421码	余3码	余3循环码
0	0000	0000	0000	0011	0010
1	0001	0001	0001	0100	0110
2	0010	0100	0010	0101	0111
3	0011	0101	0011	0110	0101
4	0100	0111	0100	0111	0100
5	0101	1000	1011	1000	1100
6	0110	1001	1100	1001	1101
7	0111	1100	1101	1010	1111
8	1000	1101	1110	1011	1110
9	1001	1111	1111	1100	1010
位权	8421	5211	2421	无权	无权

(1) **有权BCD码**: 每位数码都有确定的位权的码,

例如: 8421码、5211码、2421码.

如: 5211码1011代表 $5+0+1+1=7$;

2421码1100代表 $2+4+0+0=6$.

* 5211BCD码和2421BCD码不唯一.

例: 2421BCD码**0110**也可表示6

* 在表中:

① 8421BCD码和代表0~9的二进制数一一对应;



② 2421BCD码的前5个码和8421BCD码相同，后5个码以中心对称取反，这样的码称为自反代码。

例： $4 \rightarrow 0100$ $5 \rightarrow 1011$
 $0 \rightarrow 0000$ $9 \rightarrow 1111$

(2) 无权BCD码：每位数码无确定的位权，例如：余3码。

余3码的编码规律为：在8421BCD码上加0011，

例 6的余3码为： $0110 + 0011 = 1001$

余3码也是自反代码



4.2 格雷码(Gray码)

格雷码为无权码, 特点为: 相邻两个代码之间仅有位不同, 其余各位均相同. 具有这种特点的代码称为**循环码**, 格雷码是**循环码**.

数制

格雷码 (Gray Code) 又称循环码。从表中5位格雷码编码表中可以看出格雷码的构成方法，这就是每一位的状态变化都按一定的顺序循环。如果从00000开始，最右边一位的状态按0110顺序循环变化，右边第二位的状态按00111100顺序循环变化，右边第三位按00001111110000序循环变化。可见，自右向左，每一位状态循环中连续的0、1数目增加一倍。

- **循环码（格雷码）的特性**
- 1. 单位距离特性
- 2. 循环相邻特性
- 3. 镜像反射特性

十进制数	b_4	b_3	b_2	b_1	b_0	十进制数	b_4	b_3	b_2	b_1	b_0
0	0	0	0	0	0	16	1	1	0	0	0
1	0	0	0	0	1	17	1	1	0	0	1
2	0	0	0	1	1	18	1	1	0	1	1
3	0	0	0	1	0	19	1	1	0	1	0
4	0	0	1	1	0	20	1	1	1	1	0
5	0	0	1	1	1	21	1	1	1	1	1
6	0	0	1	0	1	22	1	1	1	0	1
7	0	0	1	0	0	23	1	1	1	0	0
8	0	1	1	0	0	24	1	0	1	0	0
9	0	1	1	0	1	25	1	0	1	0	1
10	0	1	1	1	1	26	1	0	1	1	1
11	0	1	1	1	0	27	1	0	1	1	0
12	0	1	0	1	0	28	1	0	0	1	0
13	0	1	0	1	1	29	1	0	0	1	1
14	0	1	0	0	1	30	1	0	0	0	1
15	0	1	0	0	0	31	1	0	0	0	0

作业

第五版

- 1.2-(4); 1.4-(2);
1.5-(4); 1.6-(2,3)
1.7-(1,4); 1.10-(2,4);
1.12-(3.5)

第四版

- 1.1-(2,4); 1.2-(2,3);
1.3-(1,4); 1.4-(2,4)

第二章 TTL 门电路



数字电路基础

第三章、逻辑门电路

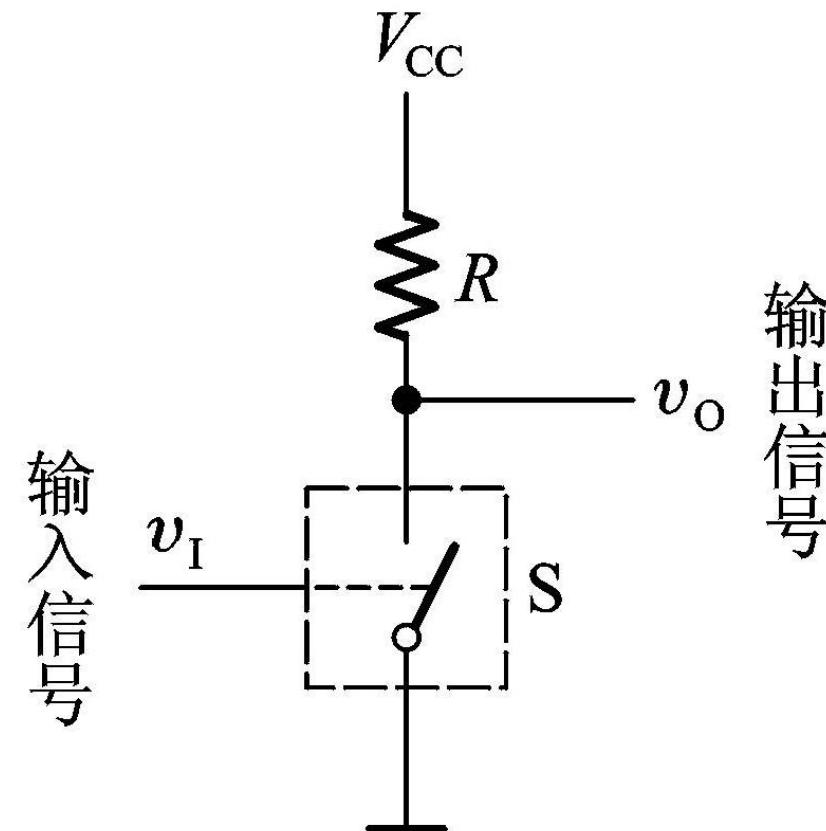
Part 1 二极管、BJT三极管 和
TTL门电路

第三章 逻辑门电路——引言

- 在电子线路中，用**高、低**电平分别表示二值逻辑的**1**和**0**两种逻辑状态。

高、低 电平值是相对的！！！

- 例如：
 - 当开关 S 断开后，输出电压 v_o 为高电平，当开关 S 闭合后，输出为低电平；
 - 通常，开关 S 由晶体管电路构成，可以通过输入信号来控制输出信号的电平。





第三章 逻辑门电路——引言

■ 正逻辑与负逻辑

- 如果以输出高电平表示逻辑1，以低电平表示逻辑0，则这种表示方法称为正逻辑，反之，称为负逻辑。
- 同一个逻辑电路，在不同的逻辑假定下，其逻辑功能是不同的。

例：

A	B	F
V_L	V_L	V_L
V_L	V_H	V_L
V_H	V_L	V_L
V_H	V_H	V_H

(a) 电平关系

A	B	F	逻辑
0	0	0	
0	1	0	
1	0	0	
1	1	1	与

(b) 正逻辑

A	B	F	逻辑
1	1	1	
1	0	1	
0	1	1	
0	0	0	或

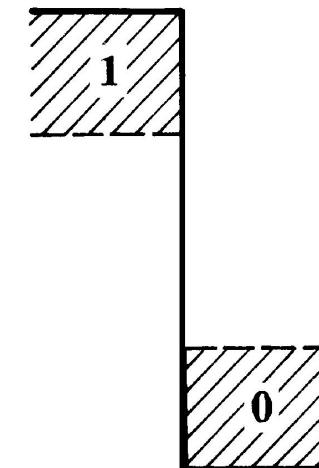
(c) 负逻辑

本门课通常采用正逻辑

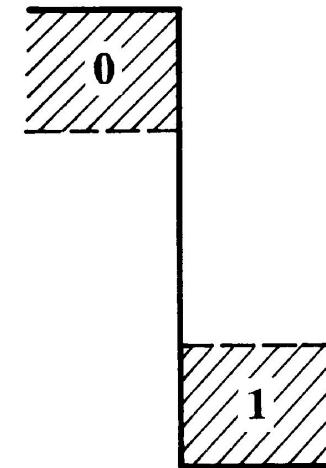
第三章 逻辑门电路——引言

■ 正逻辑与负逻辑（如图）

- 既然只要能够区分出高低电平就可以确定所表示的逻辑状态，那么，**高低电平都有一个允许的范围；**
- 数字电路对元器件的精度以及电源的稳定性要求都要比模拟电路低。



正逻辑



负逻辑



第三章 逻辑门电路 —— 引言

- 定义：实现基本逻辑运算和复合逻辑运算的电路称为**门电路**；
- 常用的门电路有与非门、或非门、与或非门、异或门、同或门等；
- 门电路中的晶体管经常处于“开”、“关”的工作状态（对应逻辑函数的高、低电平）：
 - PN结与二极管：导通、截止；
 - 双结型三极管（BJT）：饱和、截止。
 - 场效应管（CMOS）：导通、截止；



第三章 逻辑门电路

§ 3.1 二极管和BJT三极管的开关特性

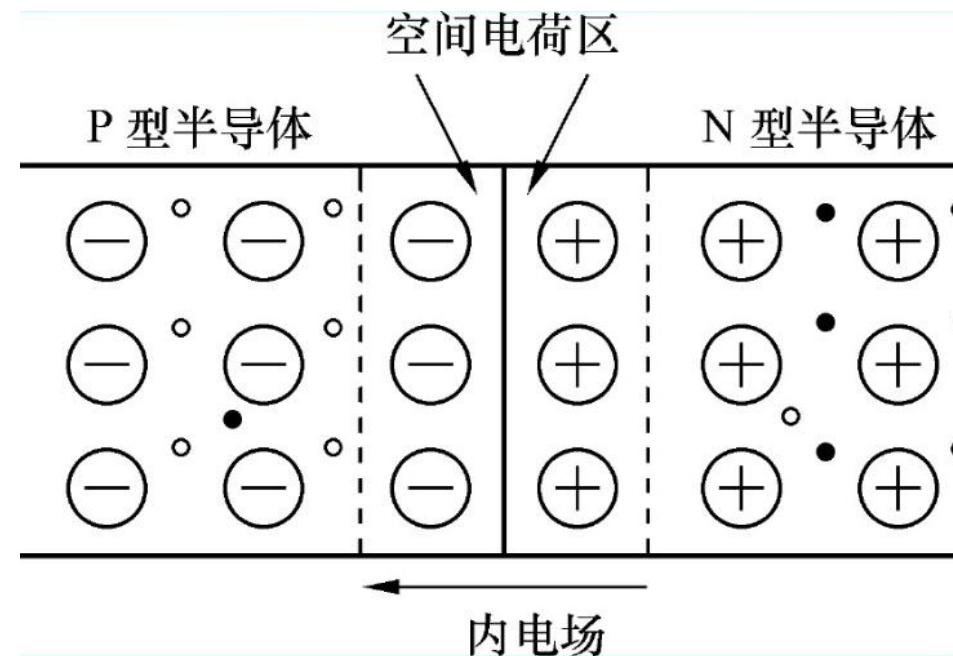
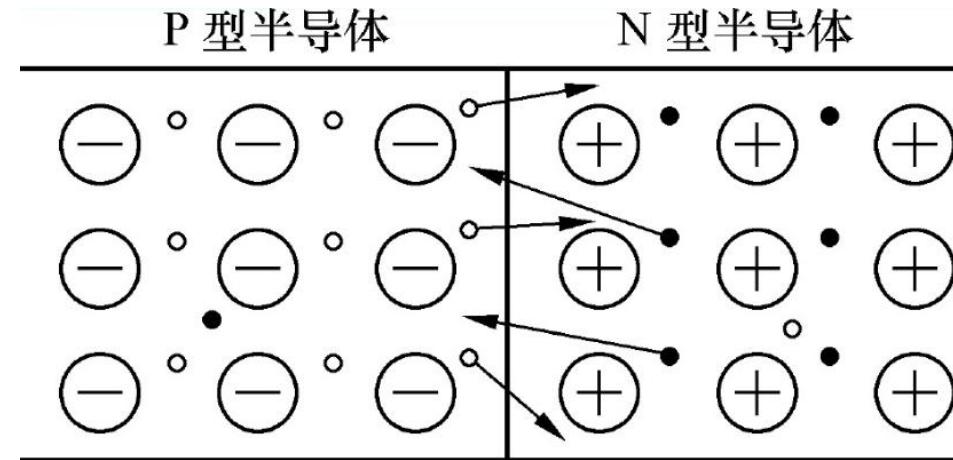
§ 3.2 TTL门电路

§ 3.3 MOS-FET元件的开关特性

§ 3.4 CMOS门电路

§ 3.5 TTL电路与CMOS电路的接口

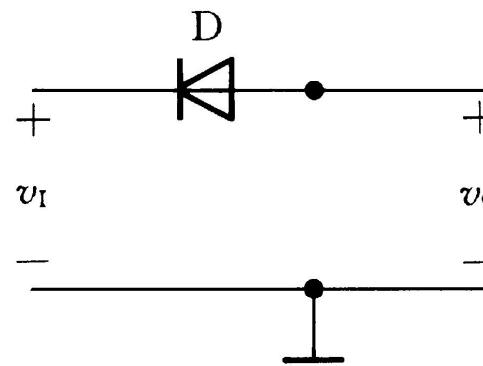
§ 3.1 分立元件门电路——PN结



§ 3.1 分立元件门电路——二极管、三极管

■ 半导体二极管的开关特性

二极管

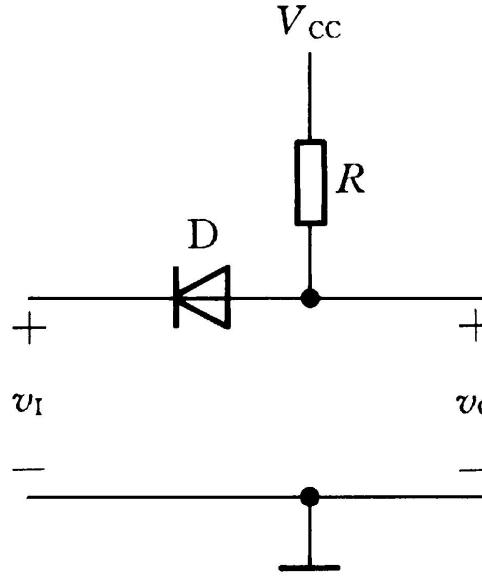


- (1) 当 $V_o - V_i$ 大于某阈值时，导通，
等效于开关闭合，导通电阻很小
- (2) 当 $V_o - V_i$ 小于某阈值时，截止，
等效于开关断开，
截止电阻通常等效为无穷大

硅二极管的阈值电压约等于0.7V

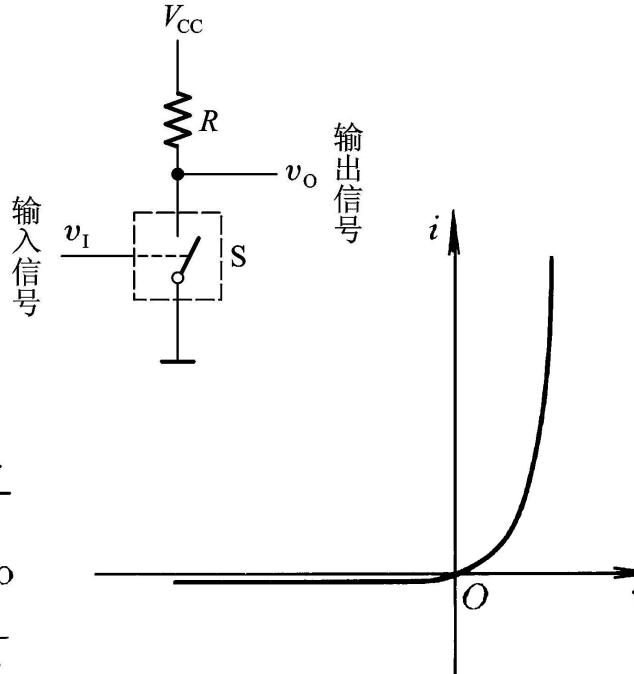
§ 3.1 分立元件门电路——二极管、三极管

■ 半导体二极管的开关特性



二极管开关电路

$v_I = \text{低时}$ 导通，输出为低
 $v_I = \text{高时}$ 截止，输出为高



$$i_D = I_S (e^{v/V_T} - 1)$$

理想二极管伏安特性

I_S : 反向饱和电流，与材料、工艺、几何尺寸有关，定值；

热力学电压 $V_T = kT/q$

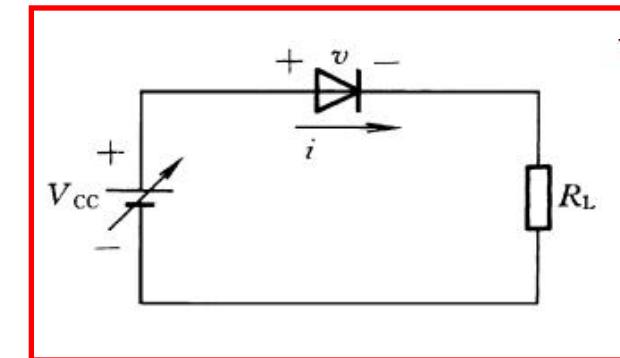
V_T : 常温下， 26mV

PN结电流方程

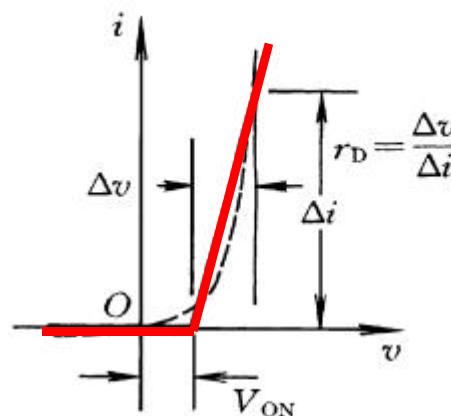
§ 3.1 分立元件门电路——二极管、三极管

■ 二极管伏安特性的近似方法

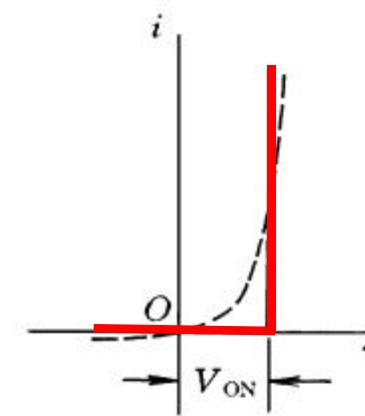
V_{CC} 外部等效电压； R_L 外部等效电阻



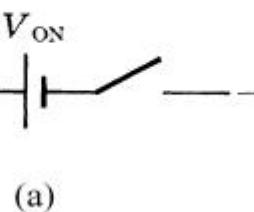
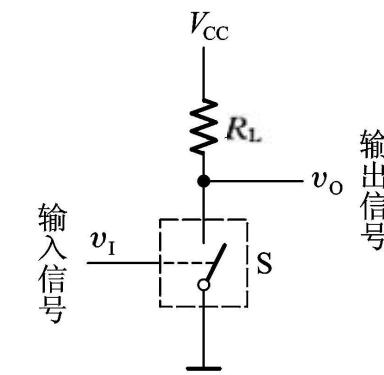
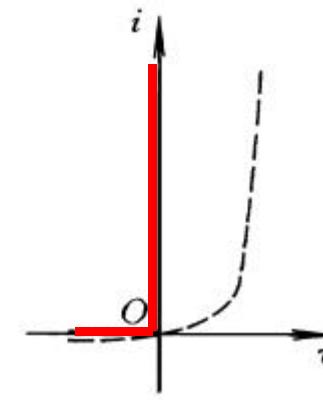
V_{CC} 、 R_L 较小



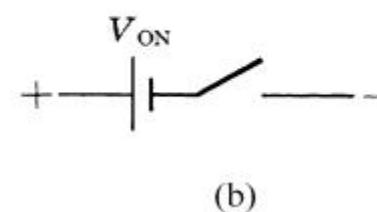
V_{CC} 较小， R_L 大



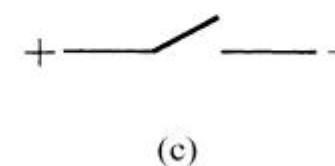
V_{CC} 、 R_L 大



(a)



(b)



(c)

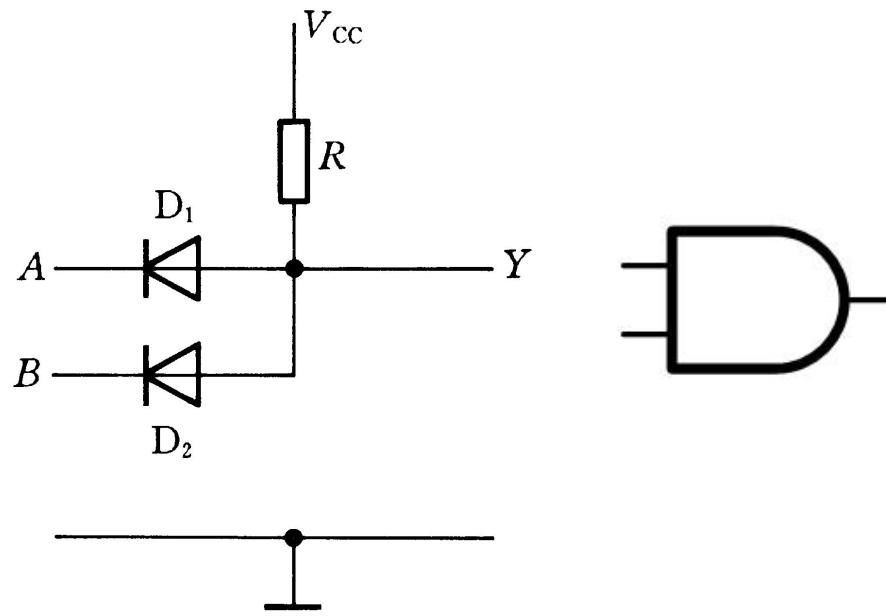
Von约为0.7V

§ 3.1 分立元件门电路——二极管、三极管

■ 二极管与门电路

$V_{CC}=5V$

输入0V代表逻辑0
输入3V代表逻辑1



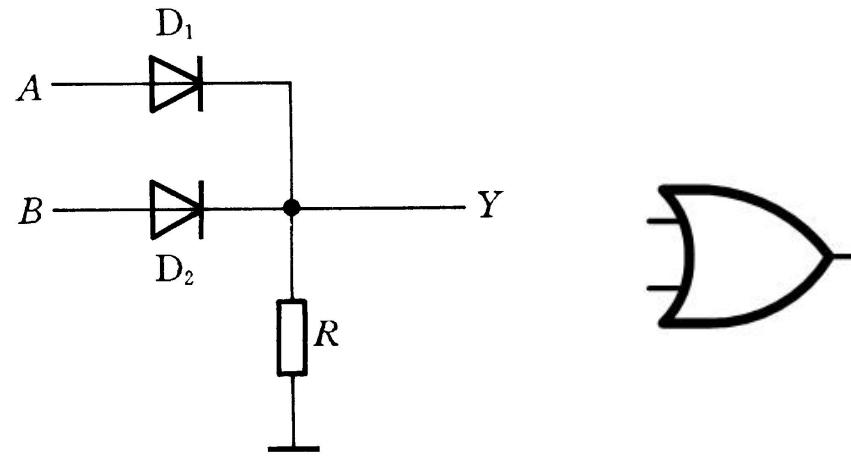
输入		输出
V_A (V)	V_B (V)	V_Y (V)
0	0	0.7
0	3	0.7
3	0	0.7
3	3	3.7

§ 3.1 分立元件门电路——二极管、三极管

■ 二极管或门电路

$$V_{CC}=5V$$

输入0V代表逻辑0
输入3V代表逻辑1



输入		输出 V_Y (V)
V_A (V)	V_B (V)	
0	0	0
0	3	2.3
3	0	2.3
3	3	2.3

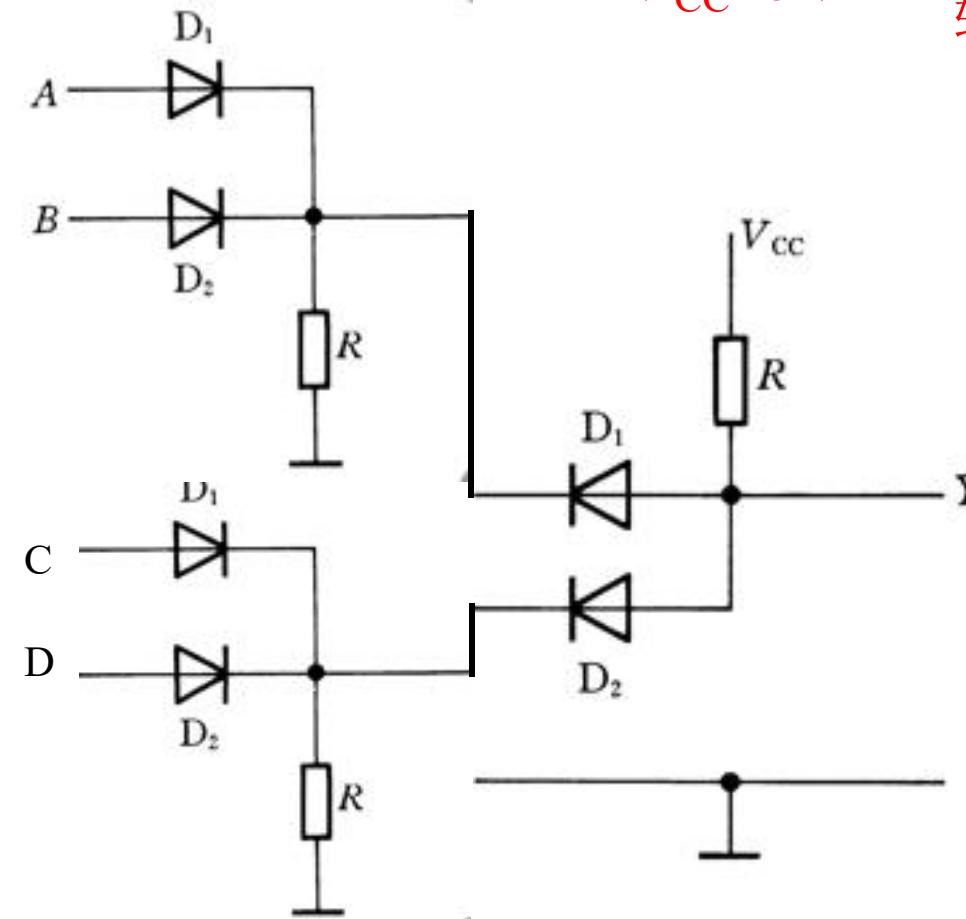
二极管门电路虽然简单，但是存在严重缺点。输出高低电平与输入高低电平不相同，相差一个二极管的导通压降，无法级联！

§ 3.1 分立元件门电路——二极管、三极管

■ 二极管或门电路

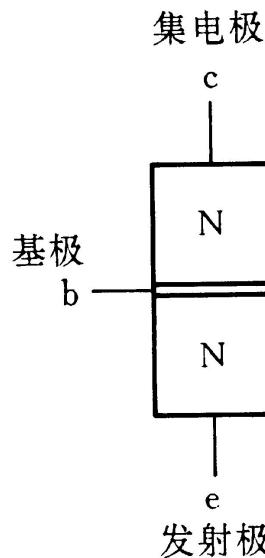
$V_{CC}=5V$

输入0V代表逻辑0
输入3V代表逻辑1

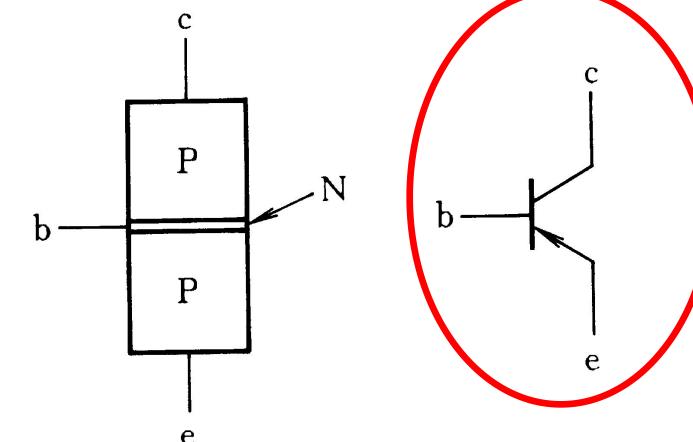


§ 3.1 分立元件门电路——二极管、三极管

- 半导体BJT三极管的开关特性
 - BJT三极管的结构



NPN型



PNP型



§ 3.1 分立元件门电路——二极管、三极管

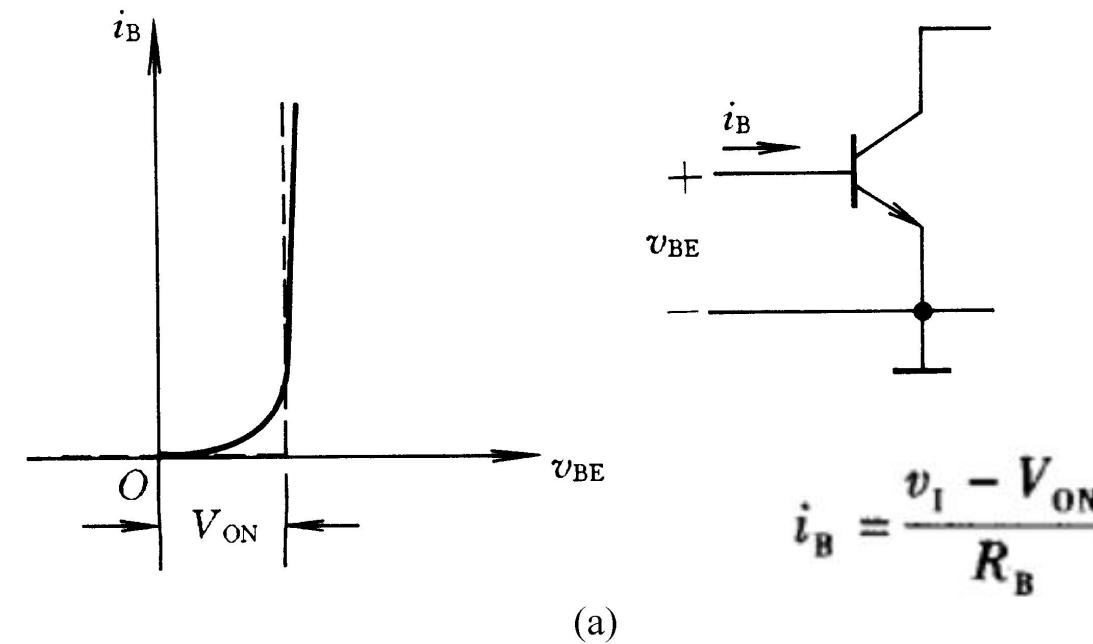
- 发射区的掺杂浓度最高；
发射载流子
- 基区很薄，且掺杂浓度最低，
一般在几个微米至几十个微米
传送控制载流子
- 集电区掺杂浓度低于发射区，
且面积大；
收集载流子

§ 3.1 分立元件门电路——二极管、三极管

■ 双极型三极管输入特性（基极-发射极回路）

- 三极管的输入电压电流特性近似为指数曲线
- 通常用折线近似
- 硅管 V_{ON} ：导通电压约为0.7V

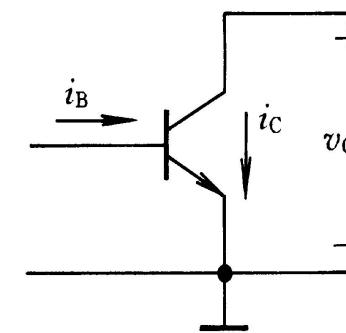
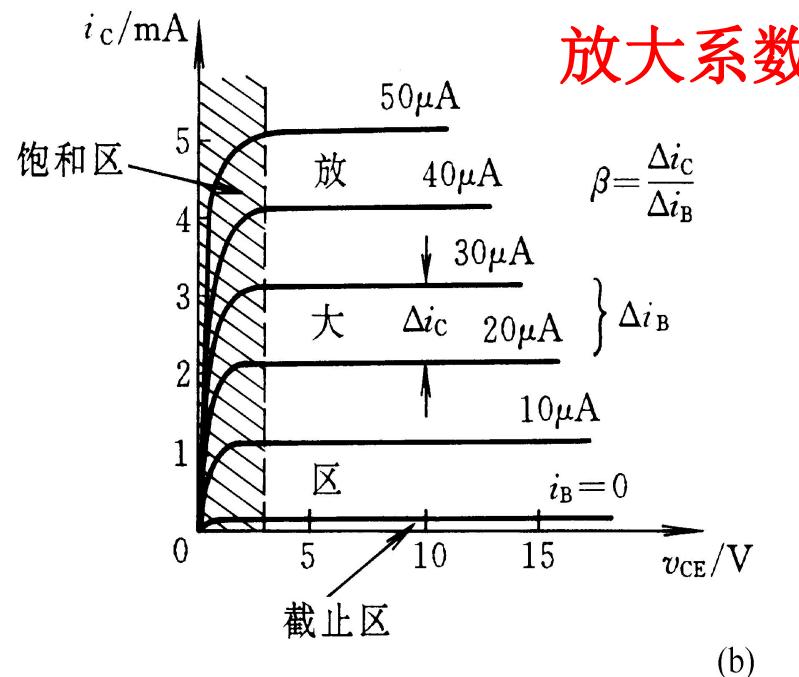
等效于PN结



■ 双极型三极管输出特性（集电极-发射极回路）

➤ 输出特性曲线可分为三个区域：放大区，饱和区，截止区

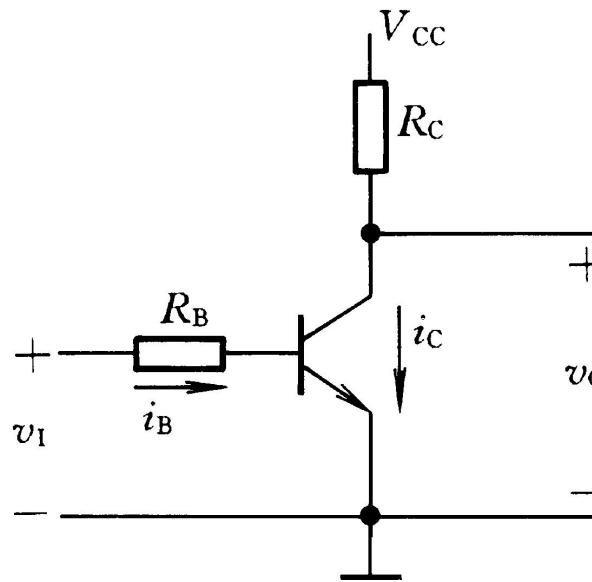
- 在放大区 i_C 随 i_B 的变化成正比变化(放大系数)，几乎不受 v_{CE} 的影响；
- 在饱和区， i_C 不随 i_B 成正比变化，而趋向饱和，硅三极管的饱和 $v_{CE(sat)}=0.6\sim0.7V$ ，在深度饱和下， $v_{CE(sat)}$ 在0.3V以下；



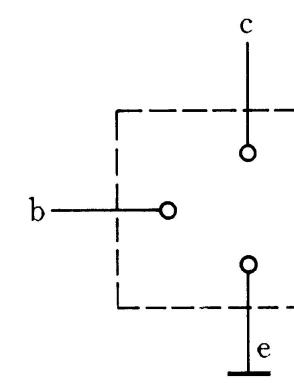
三极管：Ic受Ib控制
电流控制的电流源。

§ 3.1 分立元件门电路——二极管、三极管

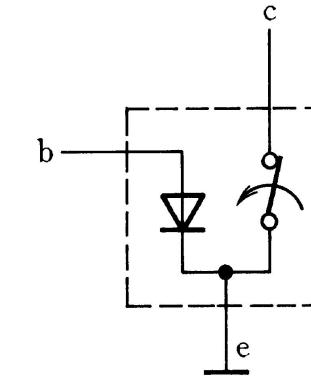
■ 双极型三极管开关电路



输出高



输出低



等效：截止 导通

($V_i = 0$)

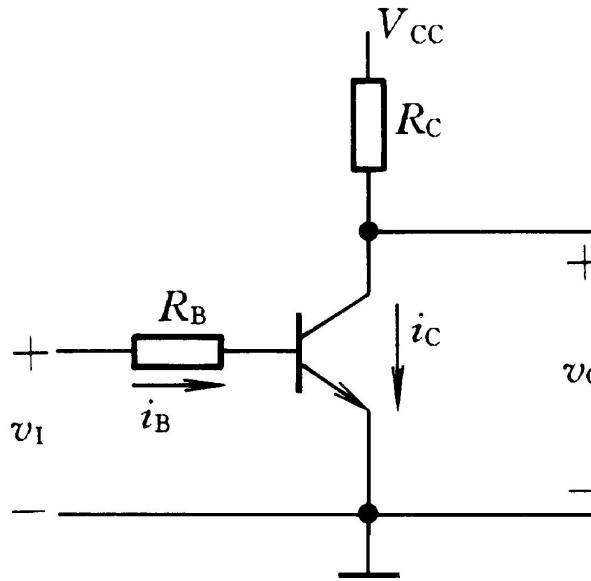
($V_i > V_{on}$)

$$i_B = \frac{v_I - V_{on}}{R_B}$$

$$\begin{aligned} v_O &= v_{CE} = V_{CC} - i_C R_C \\ &= V_{CC} - \beta i_B R_C \end{aligned}$$

§ 3.1 分立元件门电路——二极管、三极管

■ 双极型三极管开关电路（深度饱和）



当 v_I 持续升高， i_B 增加， R_C 上压降增大并接近 V_{CC} 时，**三极管压降接近0（理想开关）**，从而具有很小的饱和导通压降 $V_{CE(sat)}$ 和导通内阻 $R_{CE(sat)}$ ，处于深度饱和状态。此时，深度饱和所需的基极电流为

$$I_{BS} = \frac{V_{CC} - V_{CE(sat)}}{\beta(R_C + R_{CE(sat)})}$$

因此，为了使得三极管处于深度饱和状态，所需基极电流必须大于深度饱和电流。

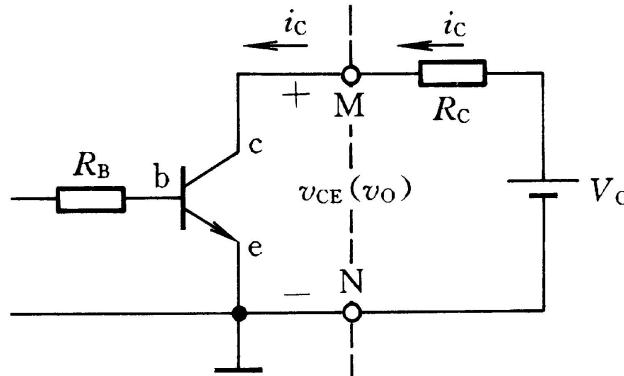
$$i_B = \frac{v_I - V_{ON}}{R_B}$$

§ 3.1 分立元件门电路——二极管、三极管

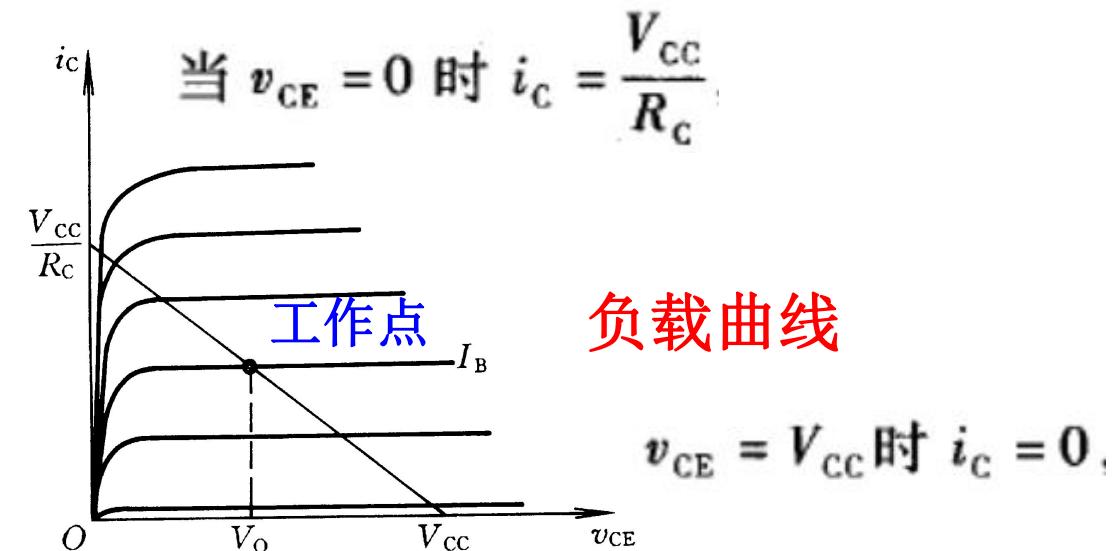
如何判断三极管工作点状态？

➤ 作图法分析：

- 取固定的负载；
- 输出特性曲线 v.s. 负载曲线。



$$\begin{aligned}v_0 &= v_{CE} = V_{CC} - i_C R_C \\&= V_{CC} - \beta i_B R_C\end{aligned}$$

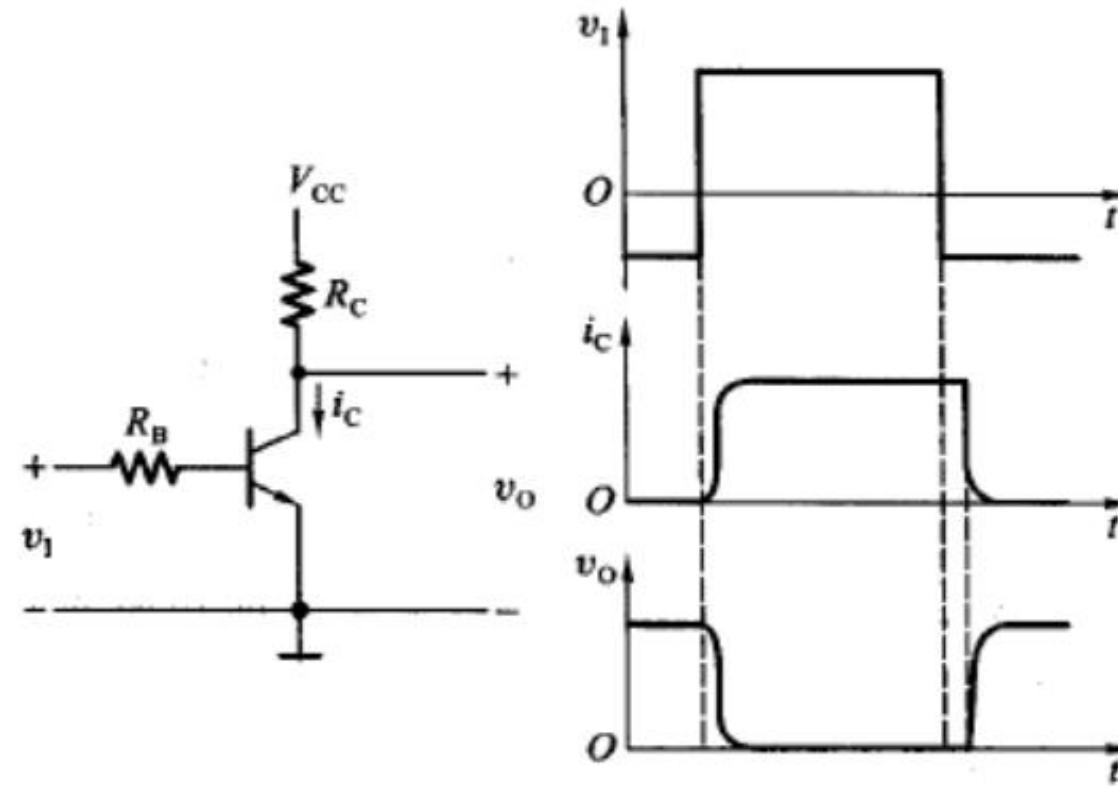


(b)

§ 3.1 分立元件门电路——二极管、三极管

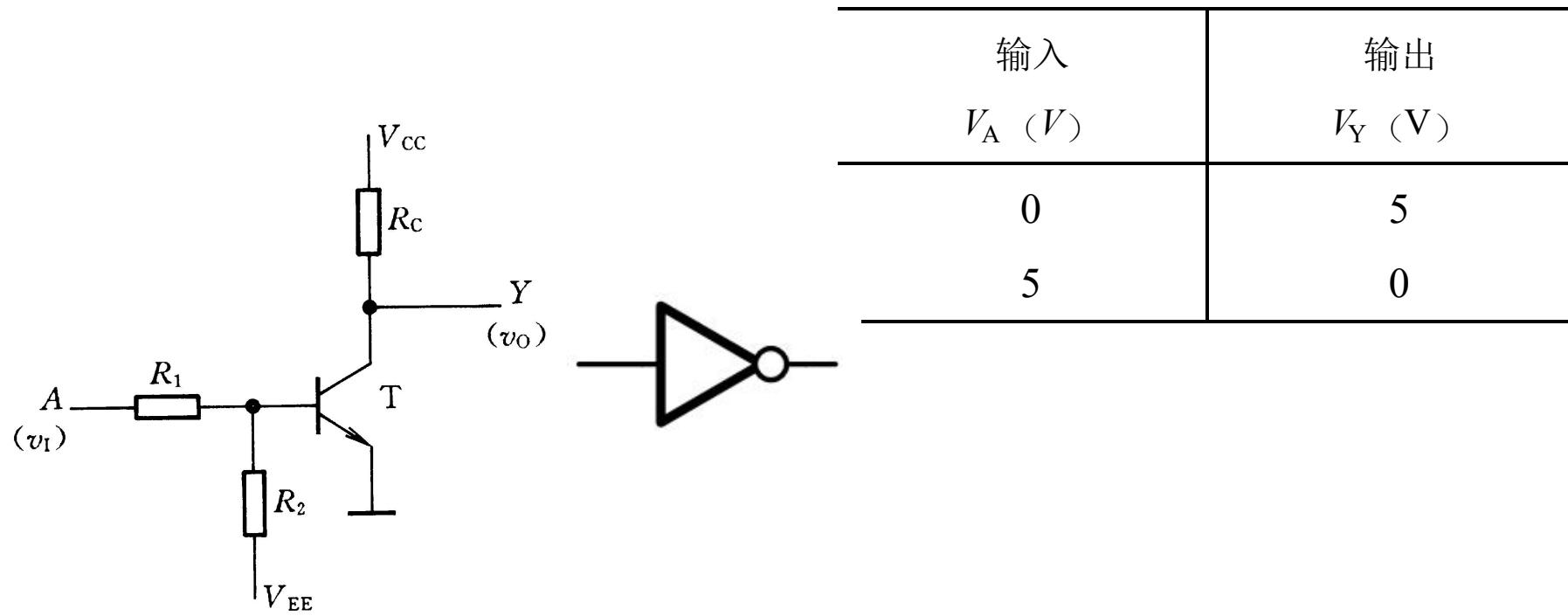
三极管动态开关特性

- 内部电荷建立与消散需要的时间



§ 3.1 分立元件门电路——二极管、三极管

■ 三极管非门电路



实际反相器考虑到功耗、级联、稳定性等，不会这么简单



§ 3.2 TTL门电路

74系列 TTL : Transistor-Transistor Logic

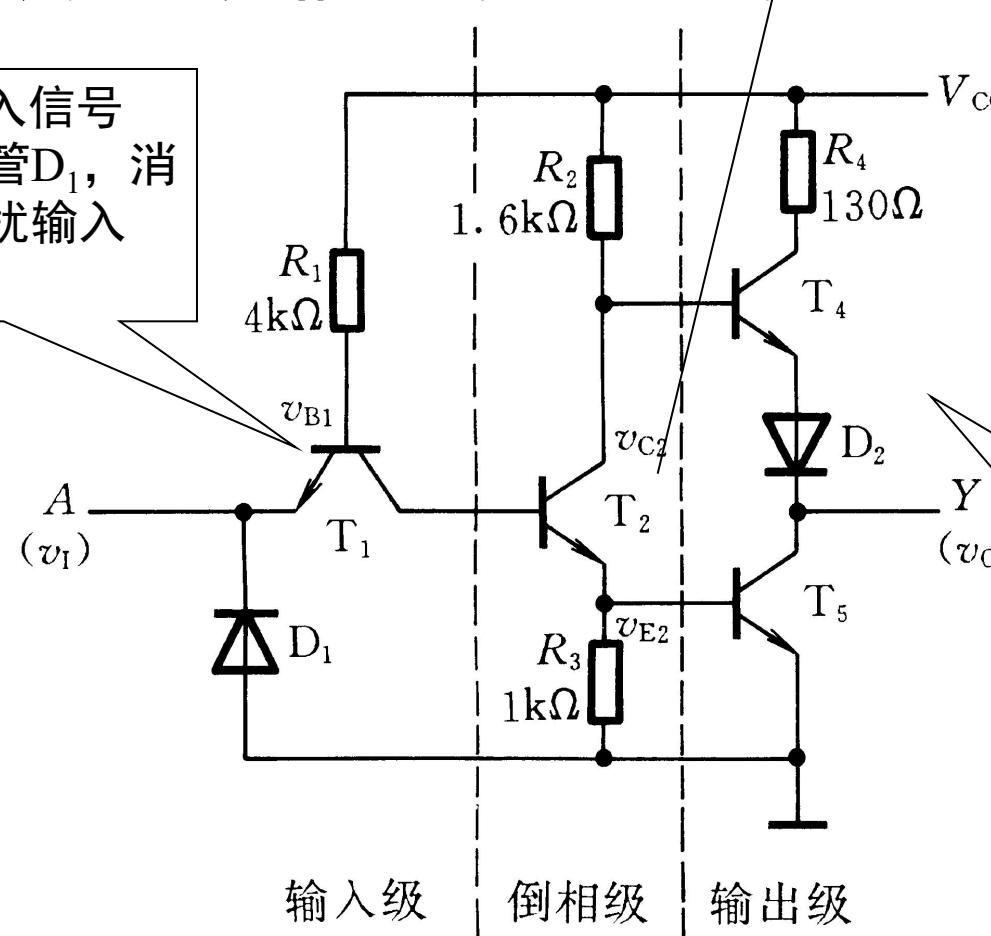
提纲:

- TTL反相器的结构与原理
- TTL反相器的输入输出特性
- TTL反相器的动态特性
- TTL门电路及其扩展

§ 3.2.1 TTL反相器的结构与原理

■ 74系列TTL反相器的结构与原理

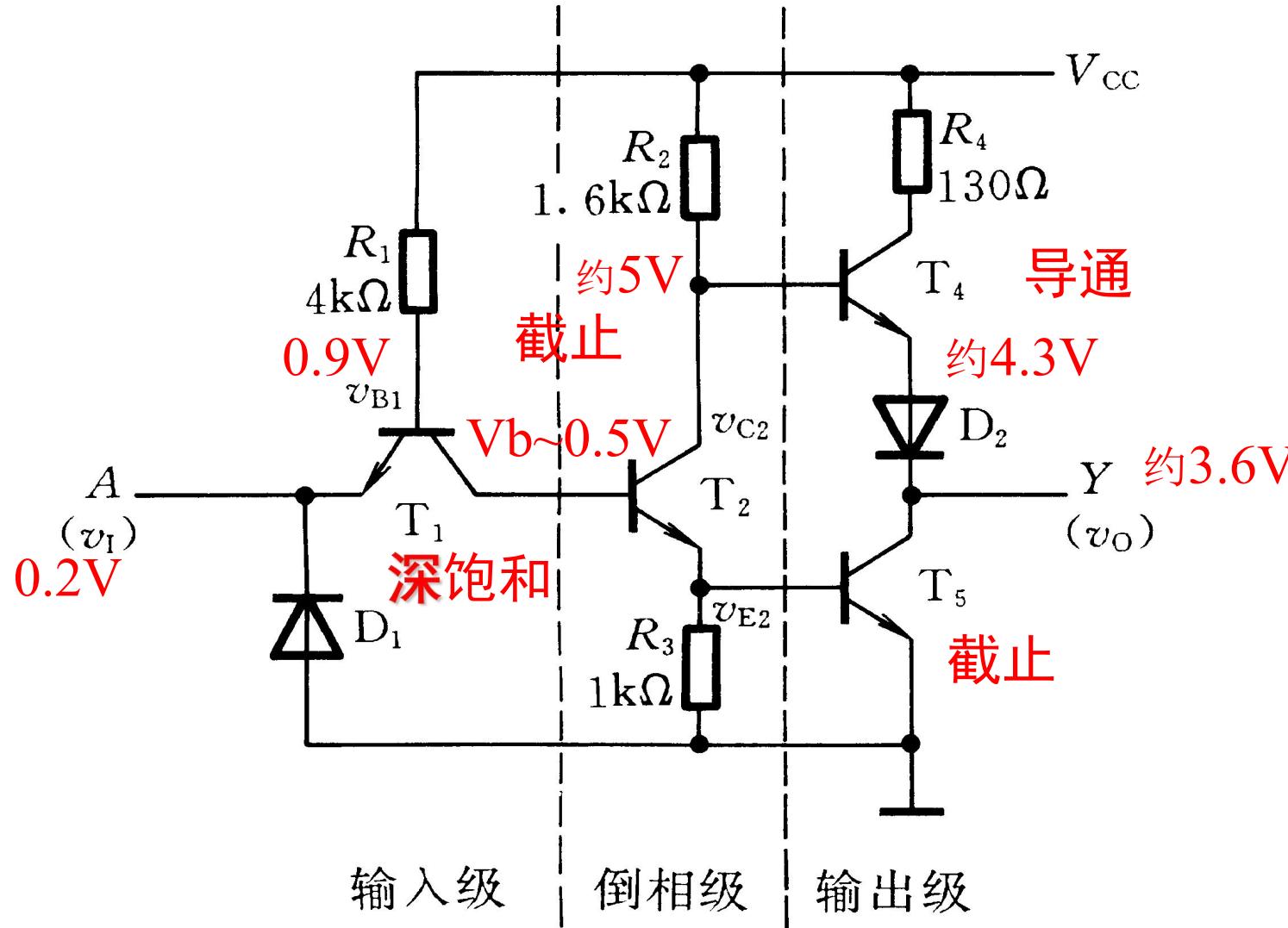
- ✓ T_1 传递输入信号
- ✓ 钳位二极管 D_1 ，消除负相干扰输入



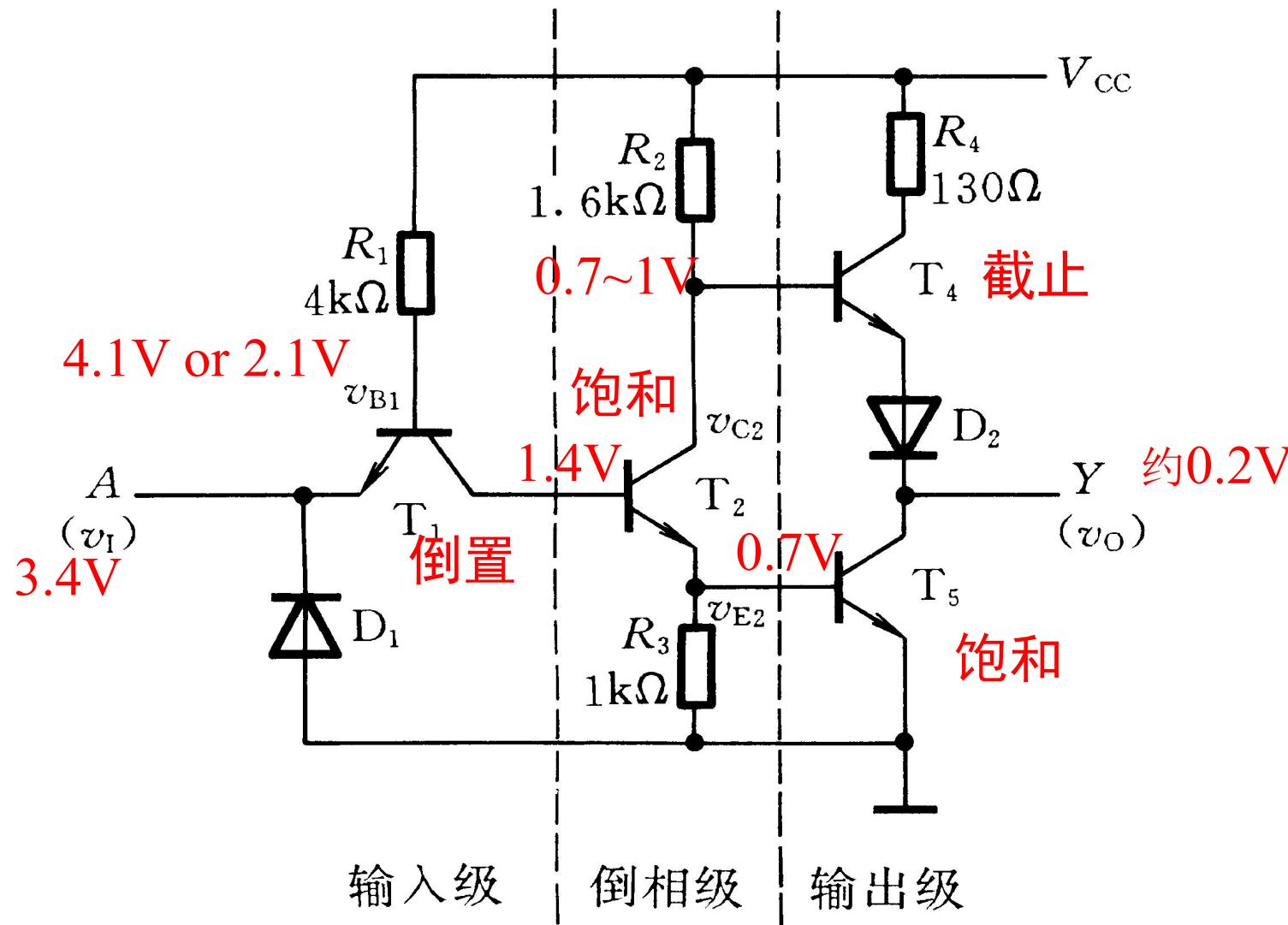
- ✓ 利用 T_2 放大作用，为 T_5 提供较大的基极电流，加速 T_5 导通/截止
- ✓ T_2 和电阻 R_2 、 R_3 组成的放大器有两个反相的输出端 V_{C2} 和 V_{E2} ，驱动 T_5 、 T_4 组成的推挽式输出级

- ✓ T_5 和 T_4 受两个互补信号 V_{e2} 和 V_{c2} 的驱动，因此总是一个导通，另一个截止
- ✓ 推挽式输出级
- ✓ D_2 确保 T_5 饱和导通时 T_4 可靠截止

$$V_{CC} = 5V \quad V_{IH} = 3.4V, V_{IL} = 0.2V \quad V_{ON} \text{ 为 } 0.7V$$



输入为低电平时，输出为高电平



输入为高电平时，输出为低电平

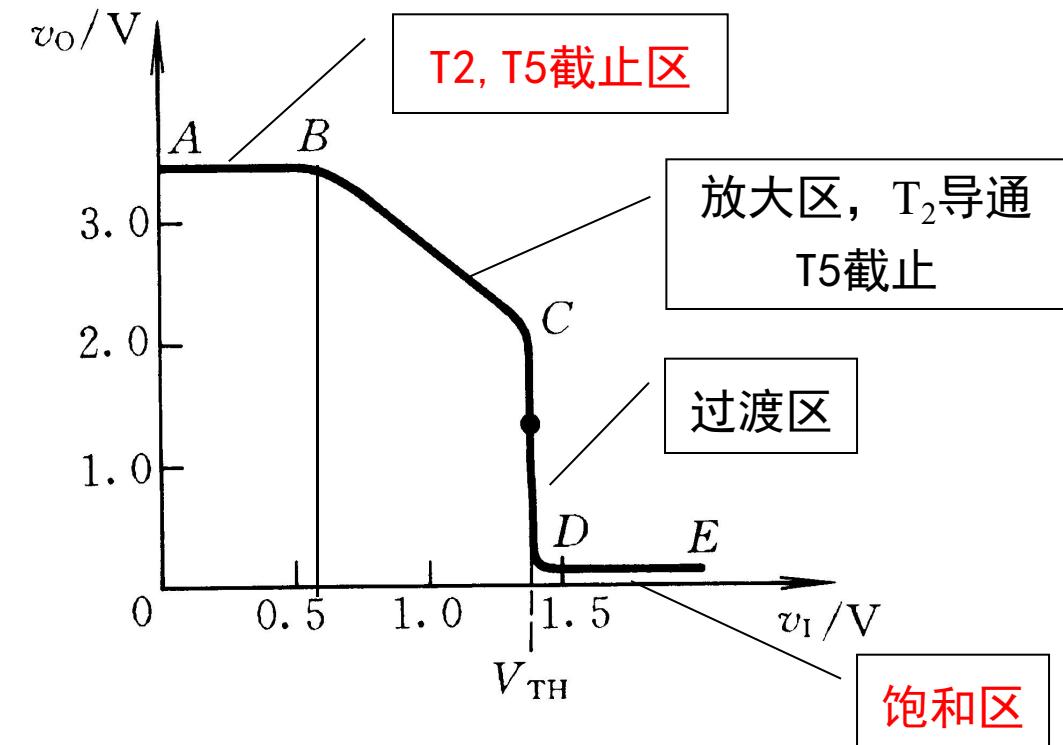
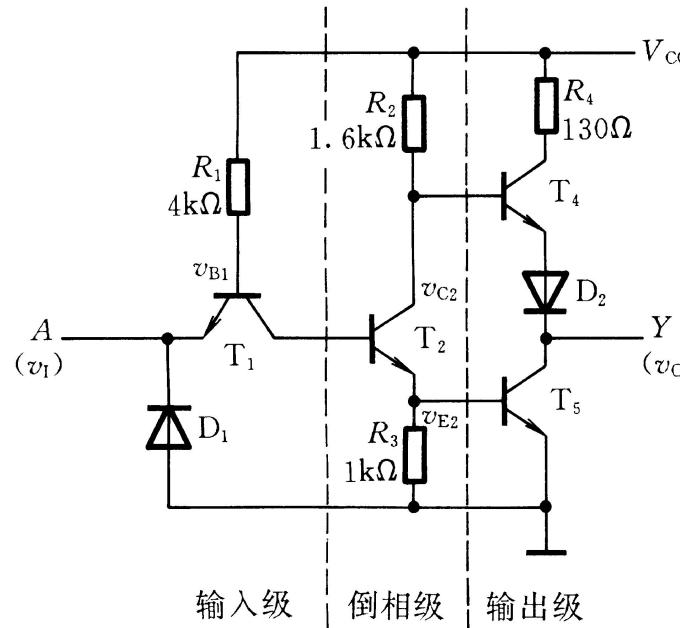
T2集电极与发射极输出电压变化方向相反，称为倒相级



§ 3.2.2 TTL反相器的输入输出特性

输入	T1	T2	T4	T5	输出
有低电平	深饱和	截止	导通	截止	高
全高电平	倒置	导通	截止	饱和	低

§ 3.2.2 TTL反相器的输入输出特性



阈值电压 V_{TH}

§ 3.2.2 TTL反相器的输入输出特性

■ 噪声容限

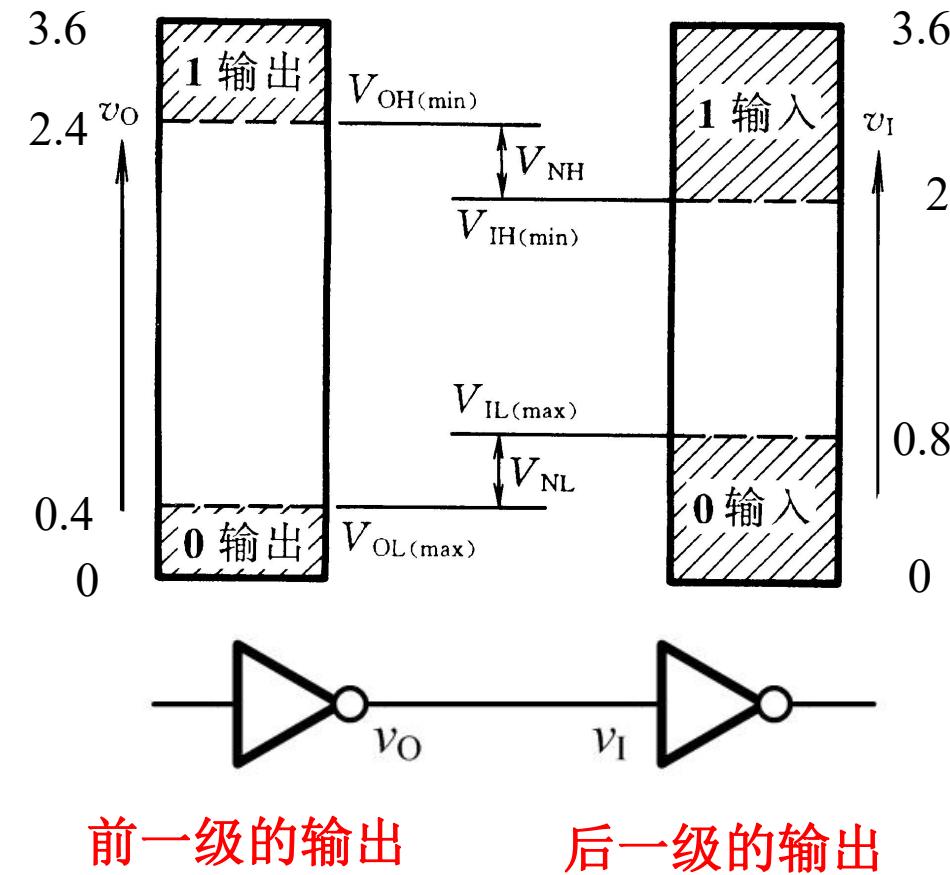
➤ 噪声容限表示门电路的抗干扰能力

➤ 低电平噪声容限

$$V_{NL} = V_{IL(\max)} - V_{OL(\max)}$$

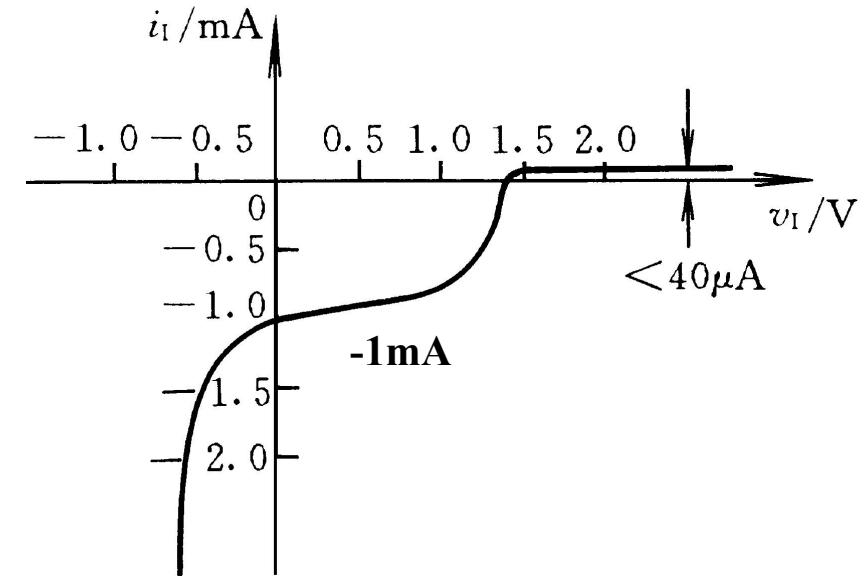
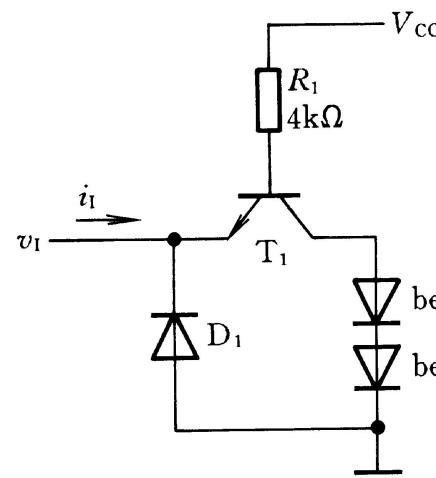
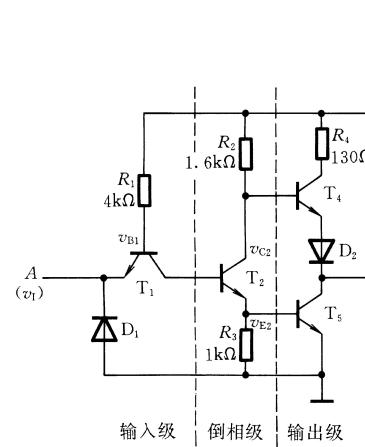
➤ 高电平噪声容限

$$V_{NH} = V_{OH(\min)} - V_{IH(\min)}$$



§ 3.2.2.1 TTL反相器的输入特性

■ 输入特性



当 $V_{cc} = 5 \text{ V}$, $v_I = V_{IL} = 0.2 \text{ V}$ 时, 输入低电平电流为

$$I_{IL} = -\frac{V_{cc} - v_{BE1} - V_{IL}}{R_1} \approx -1 \text{ mA}$$

当 $v_I > 1.4 \text{ V}$ 时, 因为 $v_{b1} = 2.1 \text{ V}$, $v_{bc} < 0$, 所以 T₁ 倒置, 电流很小 30

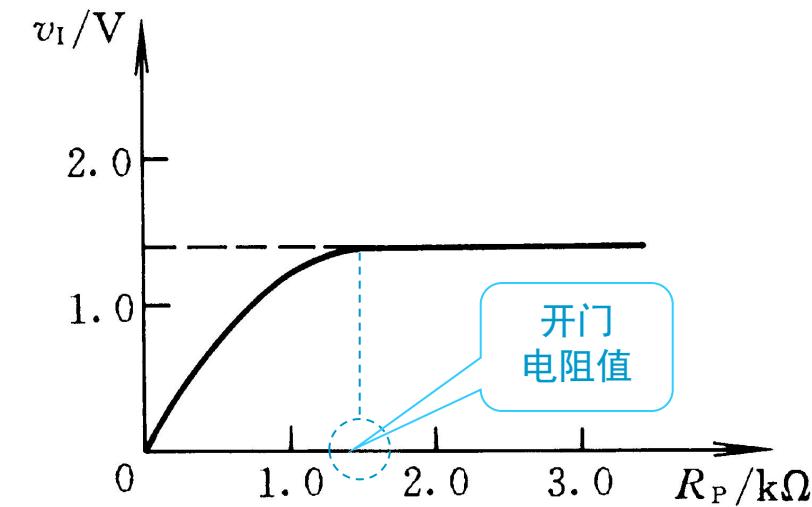
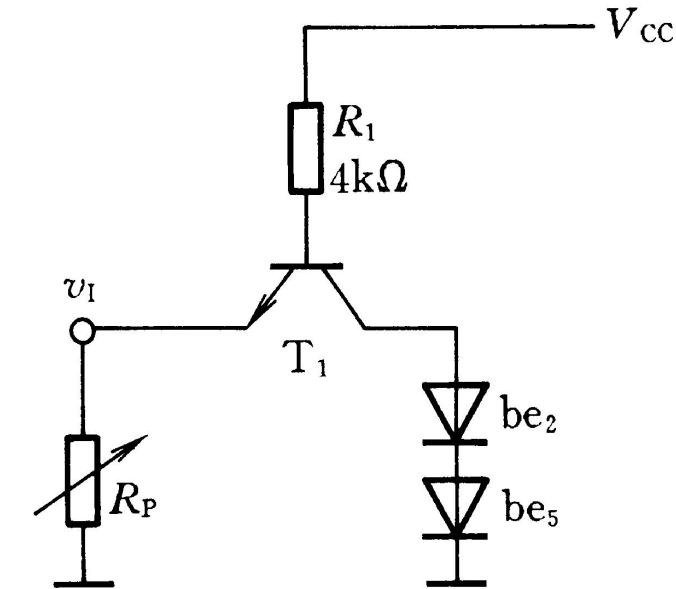
§ 3.2.2.1 输入特性

■ 输入负载特性

$$v_I = \frac{R_P}{R_1 + R_P} (V_{CC} - V_{BE1})$$

随着 R_P 增大， v_I 上升，到1.4V以后将不再上升，因为T2与T5同时导通， V_{BE1} 钳制在2.1V。此时相当于输入接高电平，输出为低电平。

开门电阻

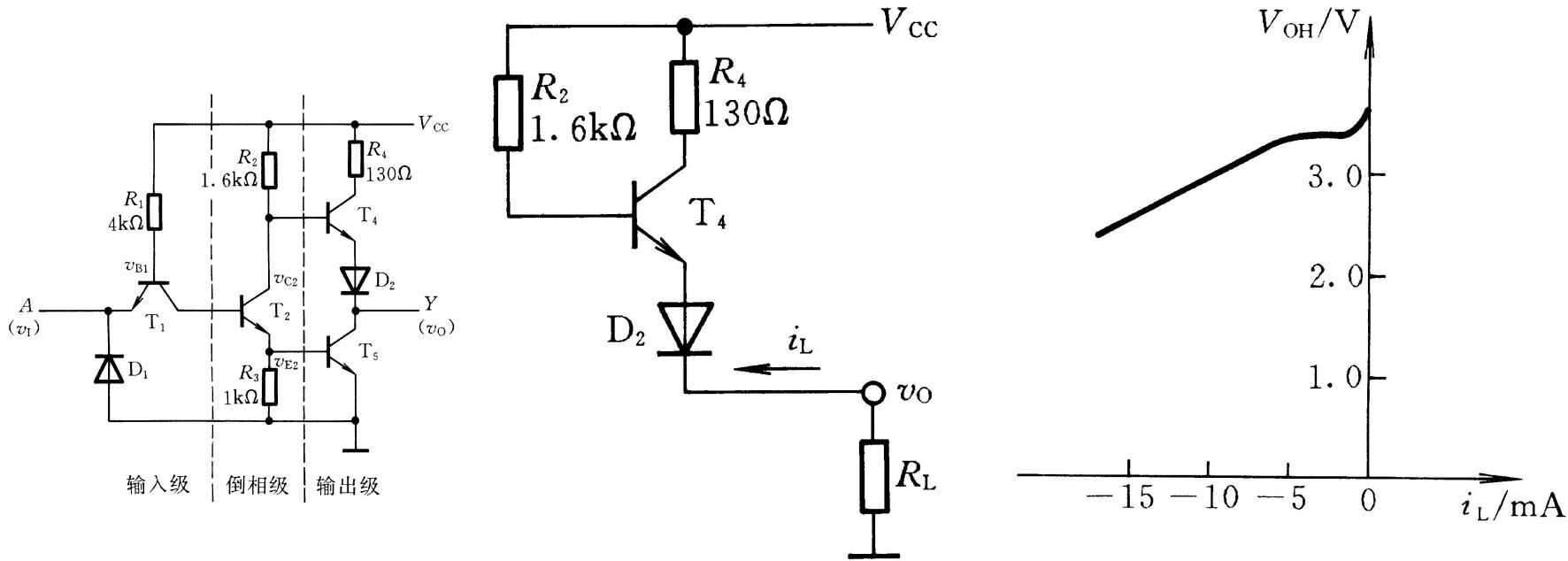


§ 3.2.2 TTL反相器的输出特性

■ 输出特性

➤ 高电平输出特性

- 受功耗限制，输出高电平时，负载电流一般不可超过0.4mA

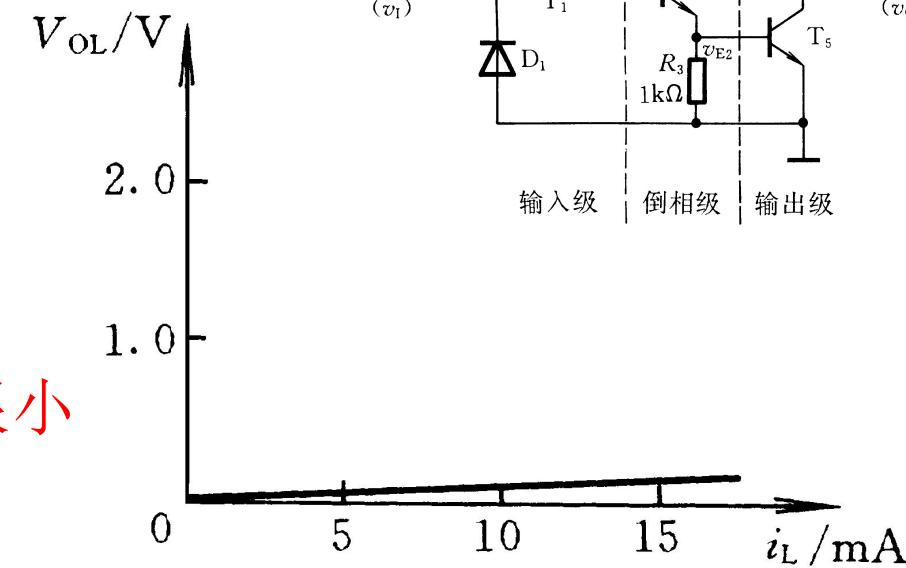
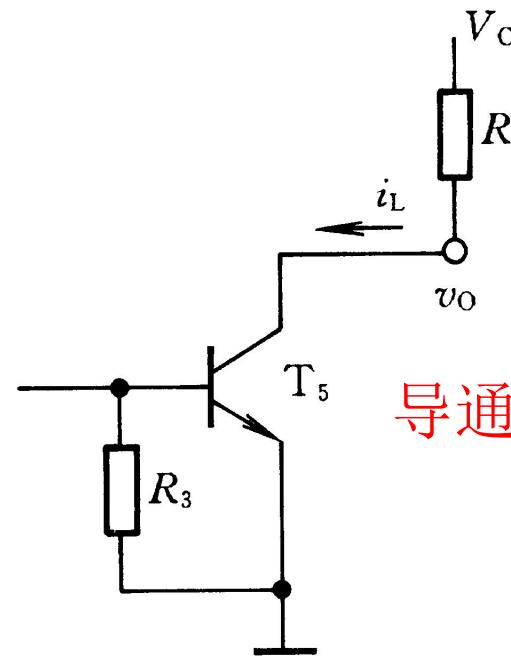


当 $v_O = V_{OH}$ 时，图 3.5.9 电路中的 T₄ 和 D₂ 导通，T₅ 截止。

§ 3.2.2.2 TTL反相器的输出特性

■ 输出特性

- 低电平输出特性

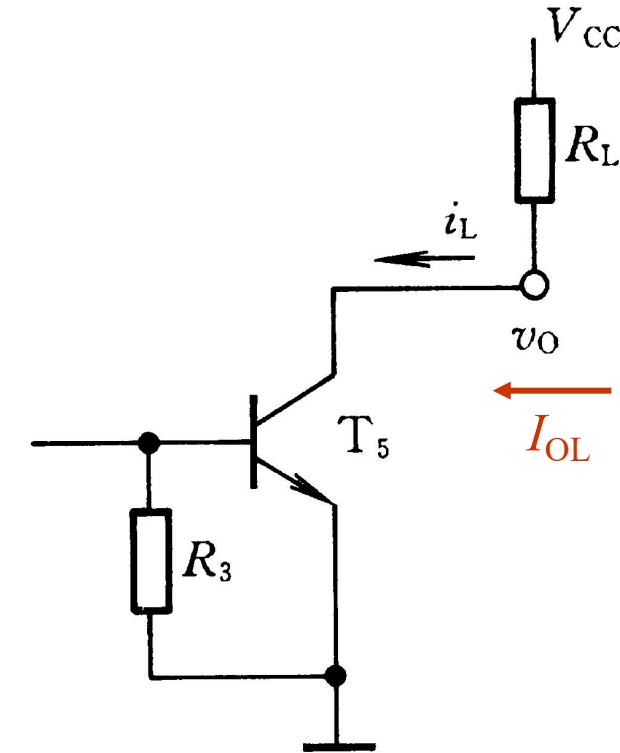


当输出为低电平时,门电路输出级的 T_5 管饱和导通而 T_4 管截止

§ 3.2.2 TTL反相器的输出特性

■ 输出带载能力 ——灌电流负载

- 驱动门输出低电平时，电流从负载门的输入端灌入驱动门的T₅管
- 输出低电平时允许灌入输出端的电流定义为输出低电平电流 I_{OL} 一般产品 $I_{OL}=16mA$
- 输出低电平时所能驱动同类门的个数 N_{OL} 称为输出低电平时的**扇出系数**。



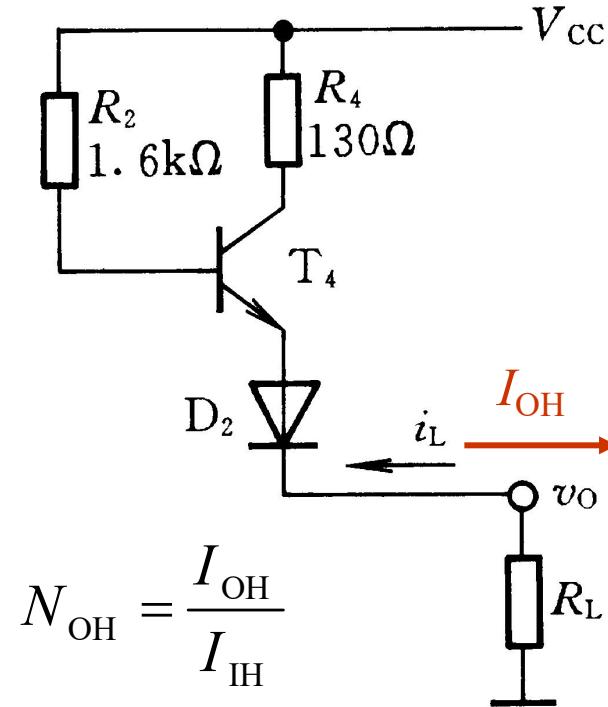
$$N_{OL} = \frac{I_{OL}}{I_{IL}}$$

§ 3.2.2 TTL反相器的输出特性

■ 输出带载能力

——拉电流负载

- 驱动门输出高电平时，电流从驱动门的T₄、D拉出而流至负载门的输入端
- 输出高电平时允许拉出输出端的电流定义为输出高电平电流 I_{OH} ，一般产品规定 $I_{OH} = 0.4\text{mA}$
- 输出高电平时所能驱动同类门的个数 N_{OH} 称为输出高电平时的**扇出系数**。



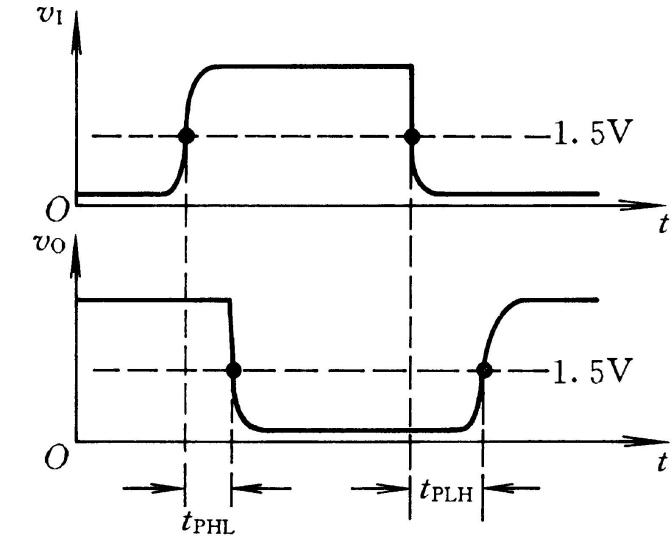
$$N_{OH} = \frac{I_{OH}}{I_{IH}}$$

一般 $N_{OL} \neq N_{OH}$ ，常取两者中的较小值作为门电路的扇出系数，用 N_O 表示。

§ 3.2.3 TTL反相器的动态特性

■ 传输延迟

- 三极管存储电荷的注入和消散；PN结寄生电容和负载电容的充放电导致门延迟。
- ✓ 导通延迟时间 t_{PHL} ——从输入波形上升沿的中点到输出波形下降沿的中点所经历的时间；
- ✓ 截止延迟时间 t_{PLH} ——从输入波形下降沿的中点到输出波形上升沿的中点所经历的时间。



✓ 传输延迟时间 t_{pd} 是 t_{PLH} 和 t_{PHL} 的平均值

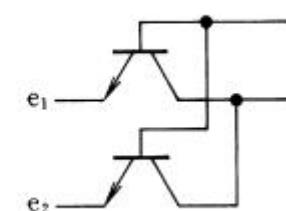
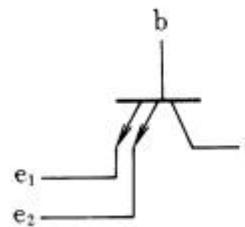
$$t_{pd} = \frac{t_{PLH} + t_{PHL}}{2}$$

标准工艺TTL门，
一般几纳秒~十几个纳秒

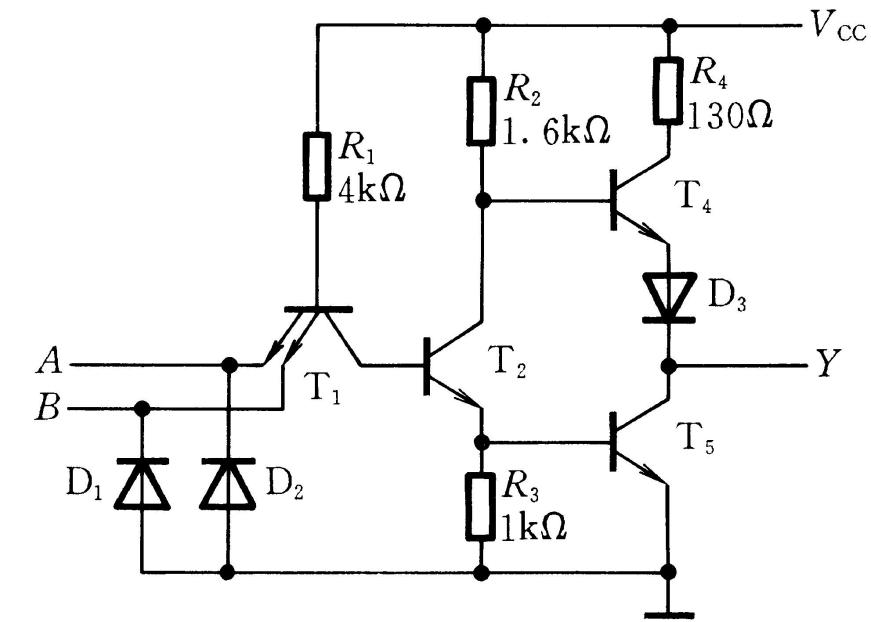
§ 3.2.4 TTL门电路及其扩展

■ 与非门

多发射极三极管



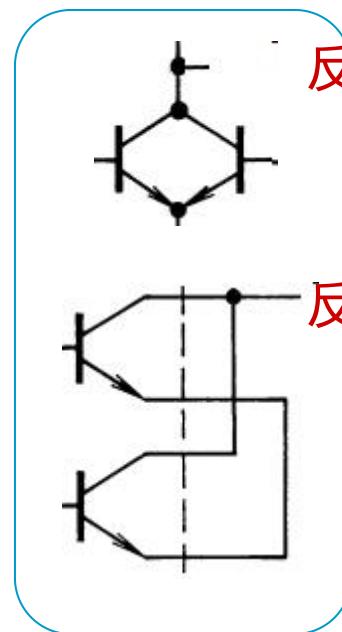
“与” 扩展器



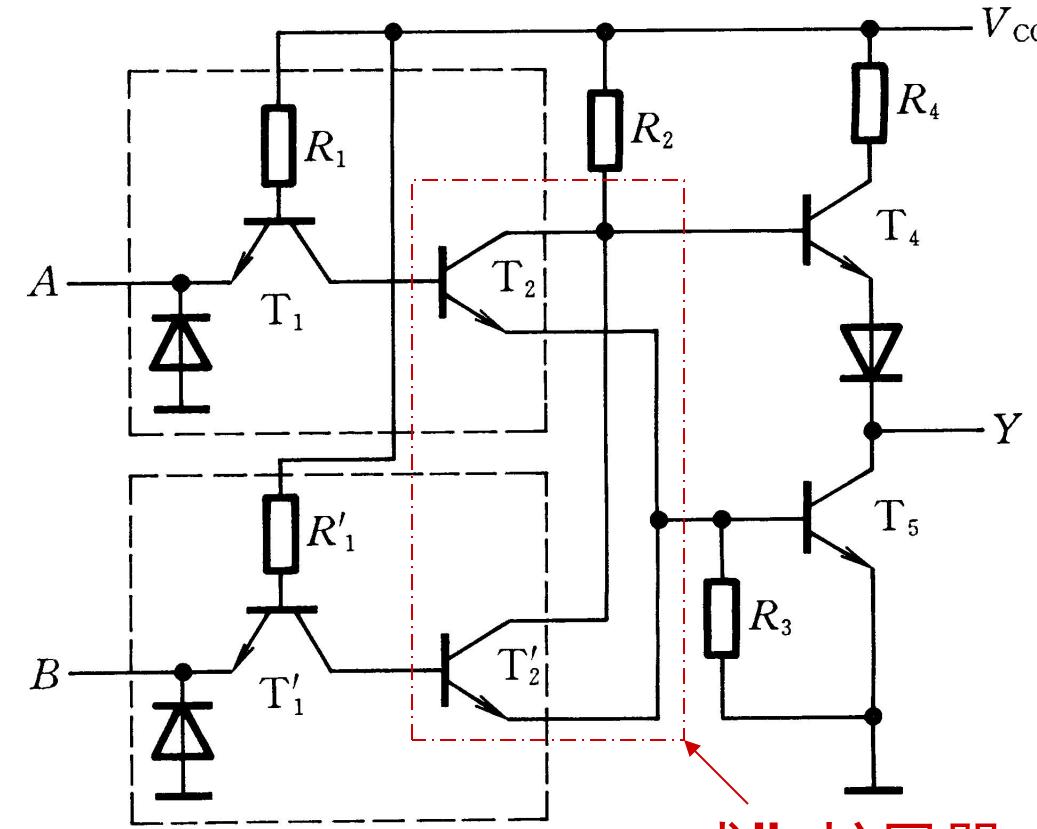
$$Y = \overline{AB}$$

§ 3.2.4 TTL门电路及其扩展

■ 或非门



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

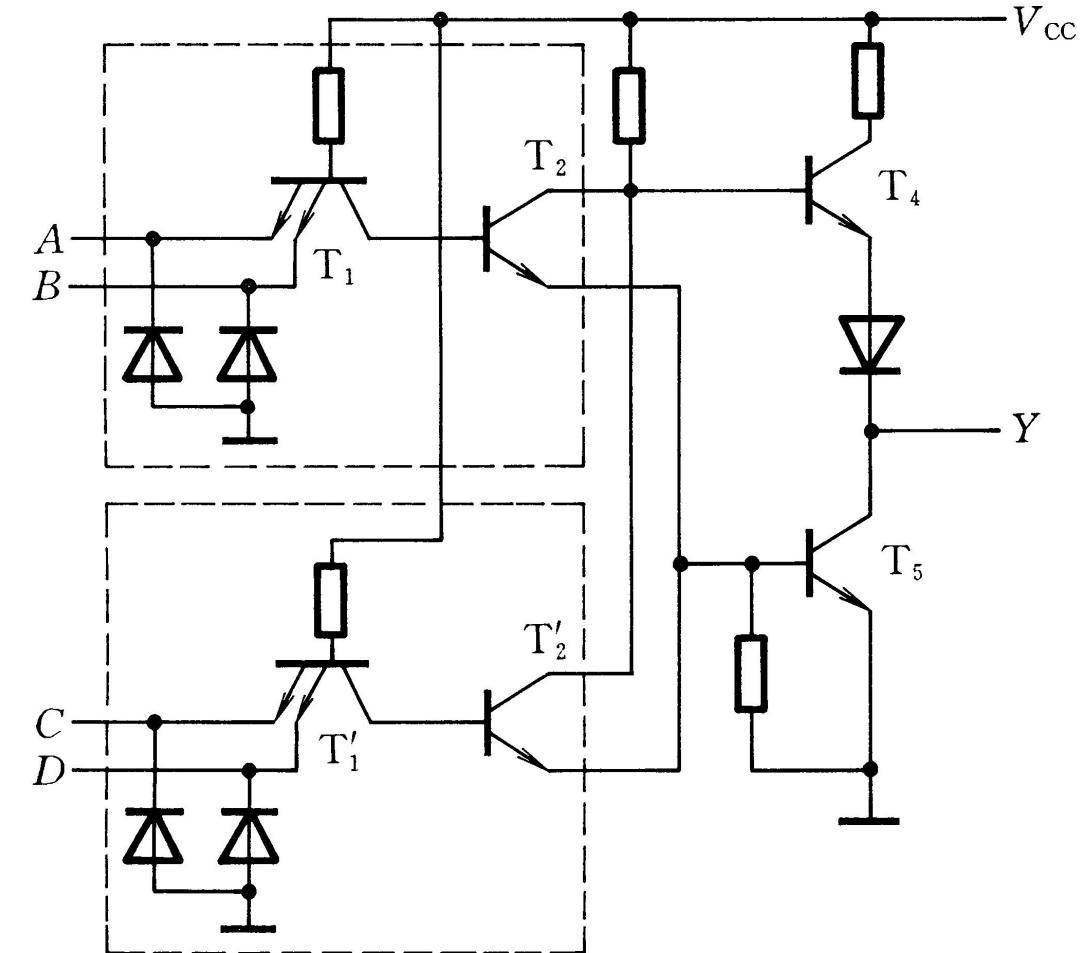


“或”扩展器是俗称，实际上是或非逻辑。

§ 3.2.4 TTL门电路及其扩展

■ 与或非门

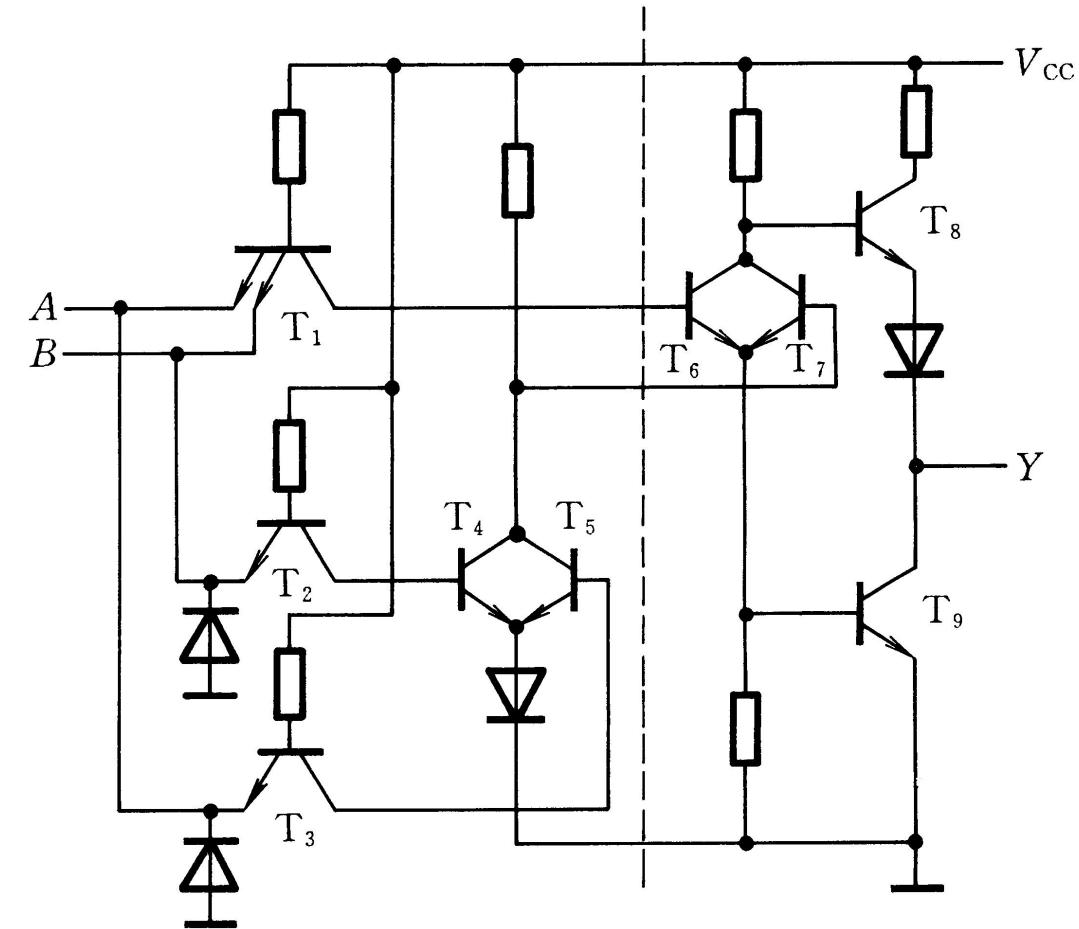
$$Y = \overline{AB + CD}$$



§ 3.2.4 TTL门电路及其扩展

■ 异或门

$$Y = A \oplus B$$



§ 3.2.4 TTL门电路及其扩展

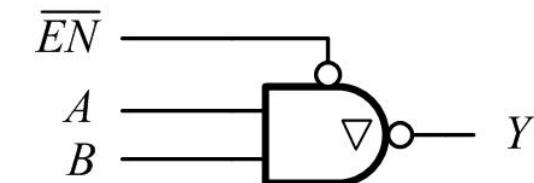
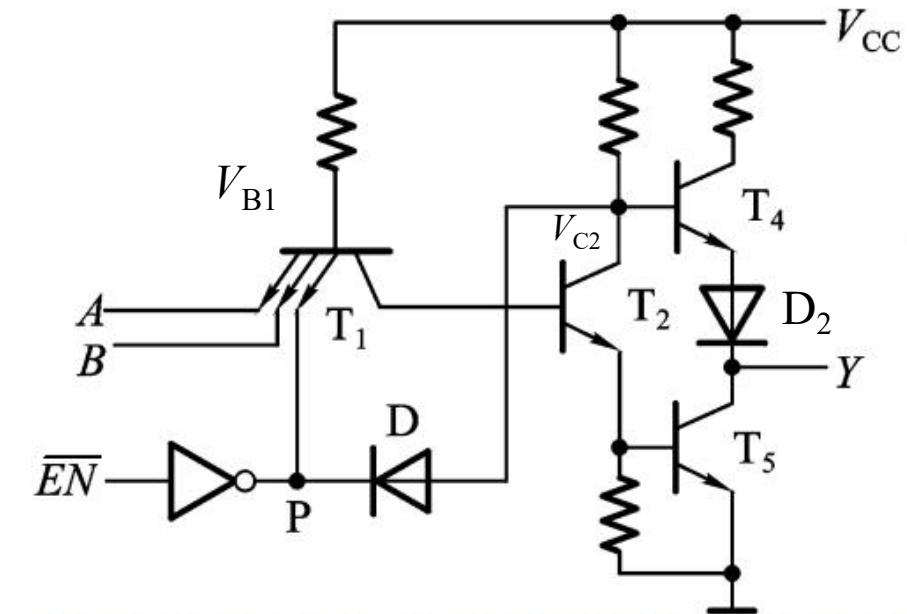
TTL三态输出门

➤ 正常工作状态

- 当 \overline{EN} 输入为低时, V_P 为高, D 截止, 与 P 端相连的 T_1 的发射结也截止。
- 三态门相当于一个正常的二输入端与非门。

➤ 高阻态

- ✓ 当 \overline{EN} 输入为高时, $V_P = 0.2V$, 这一方面使 D 导通, $V_{C2} \approx 1V$, T_4 、 D_2 截止; 另一方面使 $V_{B1} \approx 1V$, T_2 、 T_5 也截止。
- ✓ 这时, 从输出端看进去, 对地和对电源都呈现高阻, 相当于开路。

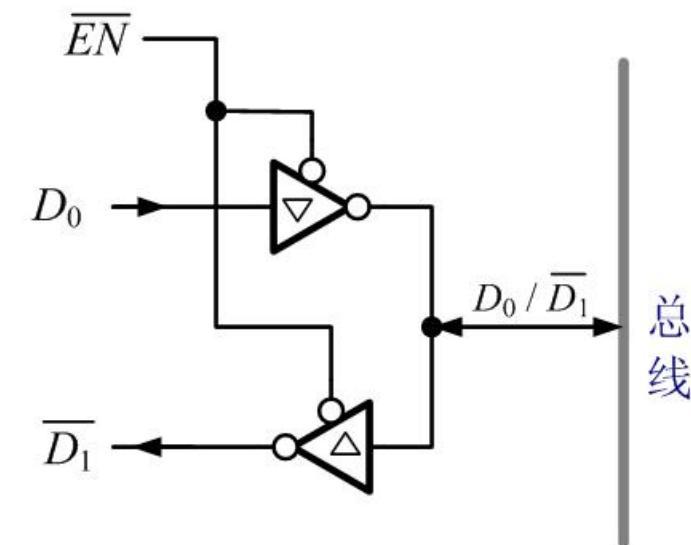
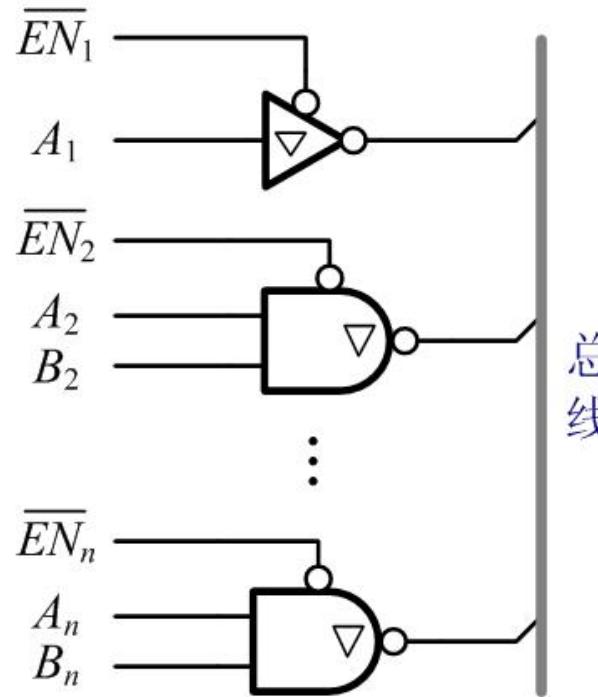


§ 3.4.4 TTL门电路及其扩展

■ 三态门的应用（续）

➤ 单向总线

➤ 双向总线





第三章 习题

第五版（阎石主编）

- 3.11、3.12、3.13

第三章 CMOS 门电路



数字电路基础

第三章、逻辑门电路

Part 2 CMOS门电路





第三章 逻辑门电路

§ 3.3 二极管和BJT三极管的开关特性

§ 3.4 TTL门电路

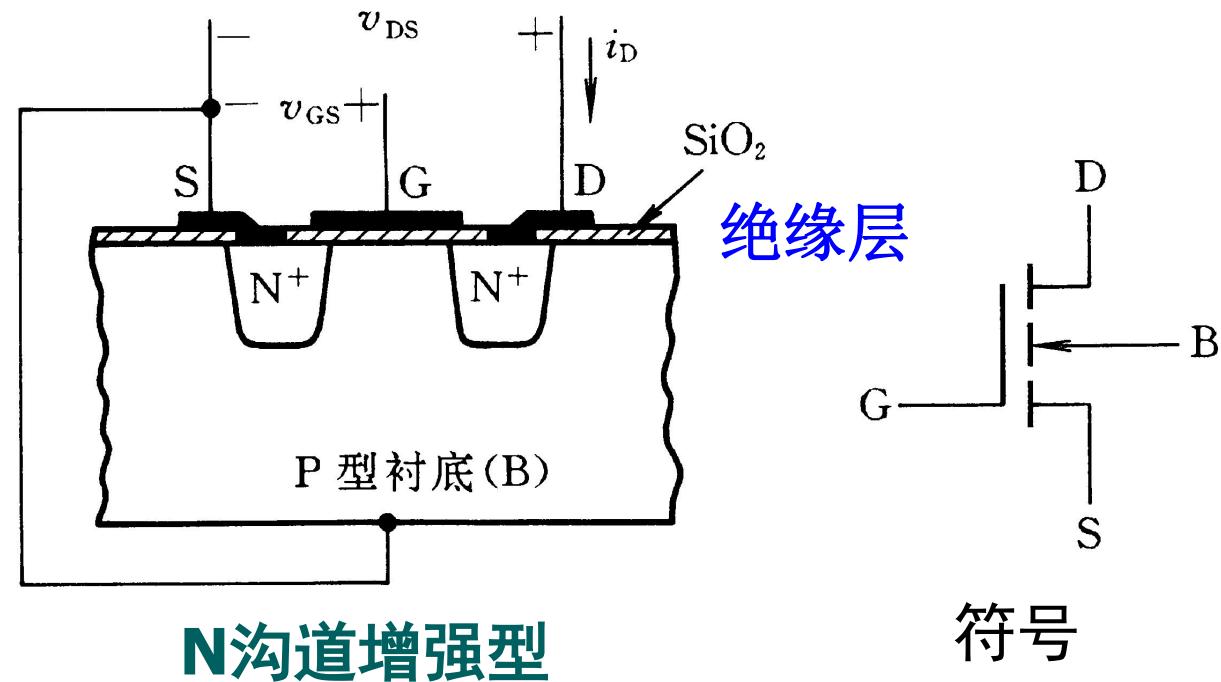
§ 3.3 MOS-FET元件的开关特性

§ 3.4 CMOS门电路

§ 3.3 MOS-FET元件的开关特性

■ Metal-Oxide-Semiconductor Field Effect Transistor

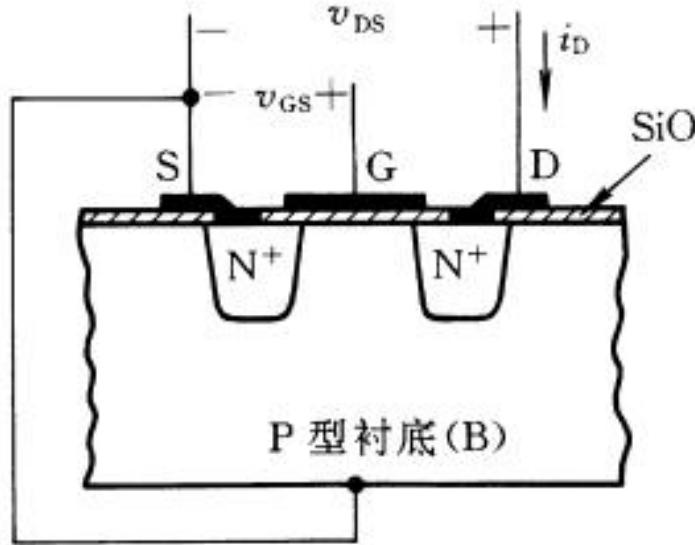
✓ MOS管结构



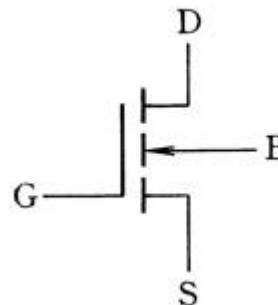
在P型半导体衬底上制作2个高掺杂浓度的N型区，形成MOS管的源极和漏极，第三个电极为栅极

§ 3.3 MOS-FET元件的开关特性

■ MOS管开关特性

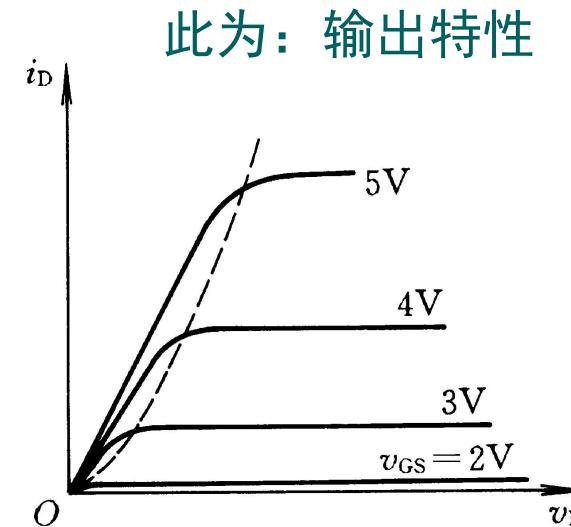
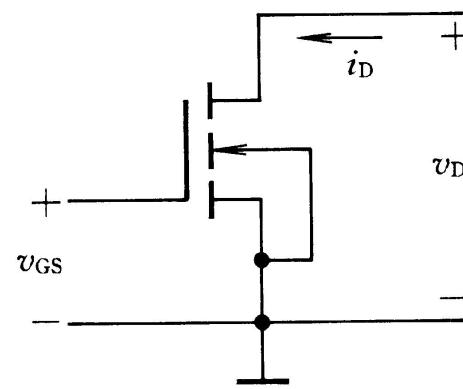


- 在漏极（D）和源极（S）加电压 v_{DS} ，如果栅极（G）和源极之间的电压 $v_{GS}=0$ ，由于源极漏极之间相当于两个PN结背向相连，电阻很大，所以DS不导通， $i_D=0$ ；
- 当 v_{GS} 大于某个电压阈值 $V_{GS(th)}$ 时，电子被吸引到栅极下面的衬底表面，形成N型反型层，构成DS之间的导电沟道，当外加 v_{DS} 电压时，DS之间将形成电流。



- $V_{GS(th)}$ 称为开启电压或阈值电压。
- 为防止漏极电流直接流入衬底，常将衬底与源极相连。

MOS管输入输出特性

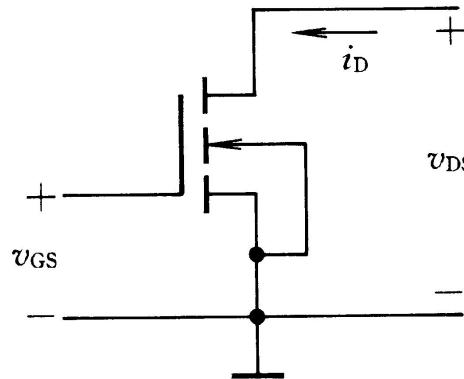


□ 输入特性：栅极绝缘层，栅极输入电流为0（对比三极管）

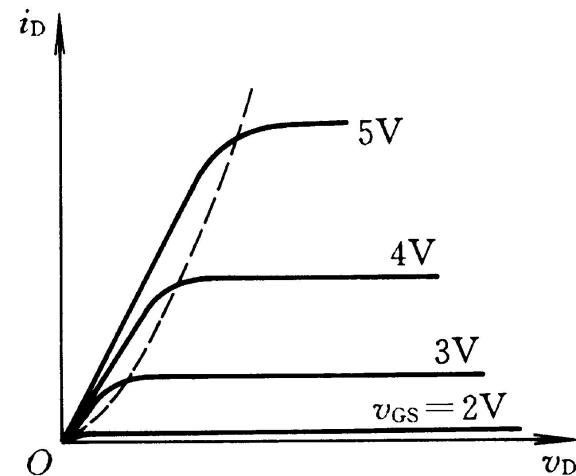
□ 漏极的输出特性可分为三个区域：

- $v_{GS} < V_{GS(\text{th})}$ 时，为截止区，此时尚未形成导电沟道，电阻巨大
- $v_{GS} > V_{GS(\text{th})}$ 时，漏极特性可分为两个区域，在虚线左称为可变电阻区（线性区），当 v_{GS} 一定时 i_D 与 v_{DS} 之比为常数；
- 虚线右侧的部分为恒流区， i_D 的大小基本上由 v_{GS} 决定， v_{DS} 对电流的影响很小。

MOS管输入输出特性

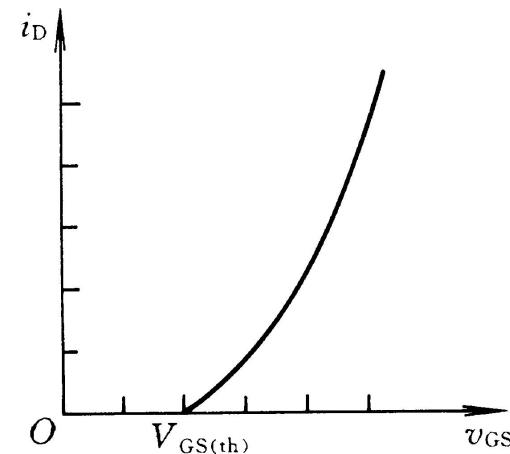


(a)



(b)

➤ 共源连接，输入端栅极无电流



转移特性

可变电阻区： R_{ON} 与 V_{GS} 成反比

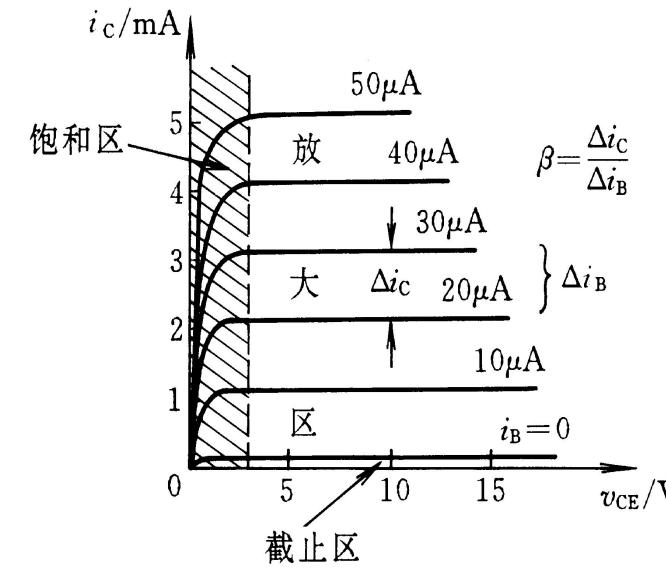
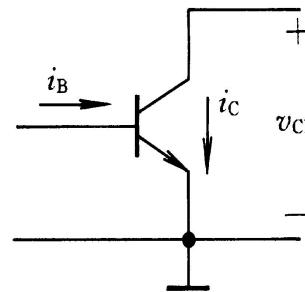
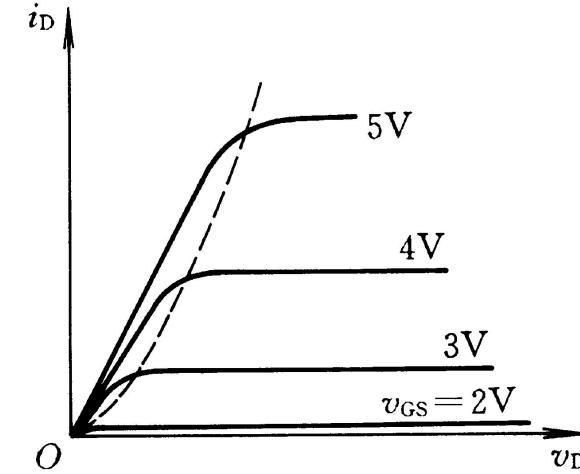
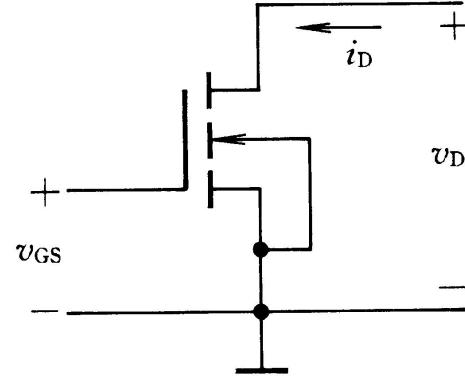
$$R_{ON} \Big|_{v_{DS}=0} = \frac{1}{2K(v_{GS} - V_{GS(th)})}$$

$$i_D = I_{DS} \left(\frac{v_{GS}}{V_{GS(th)}} - 1 \right)^2 \quad \text{其中 } I_{DS} \text{ 是 } v_{GS} = 2V_{GS(th)} \text{ 时的 } i_D \text{ 值}$$

恒流区电流主要由 V_{GS} 决定，与 v_{DS} 关系不大

恒流区

■ MOS管 VS 双极型三极管



电压控制
电流源

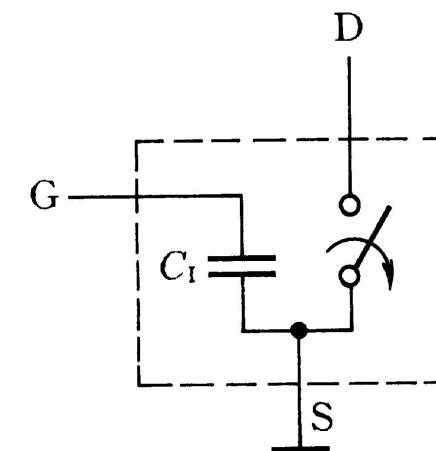
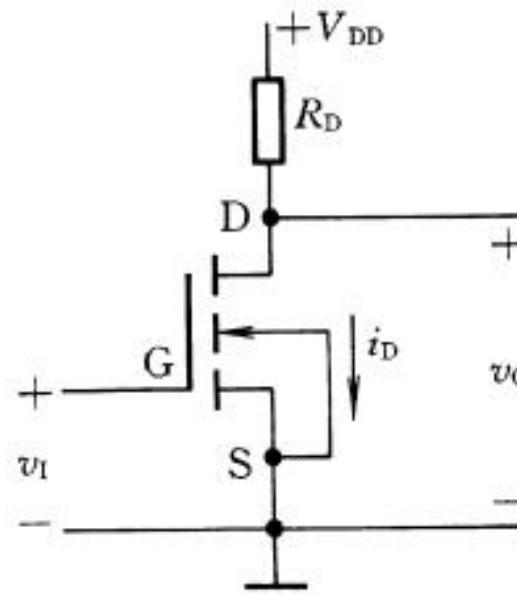
V_{GS}

电流控制
电流源

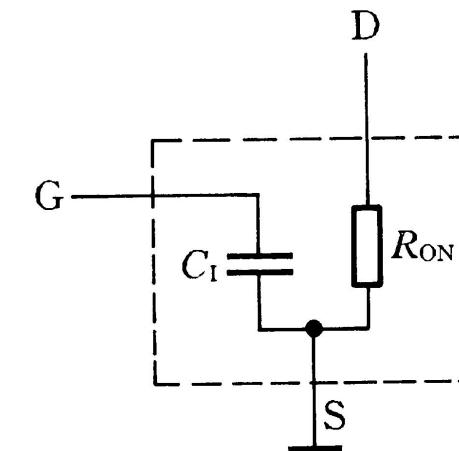
i_B

§ 3.3 MOS-FET元件的开关特性

■ MOS管开关特性与等效电路



等效：截止



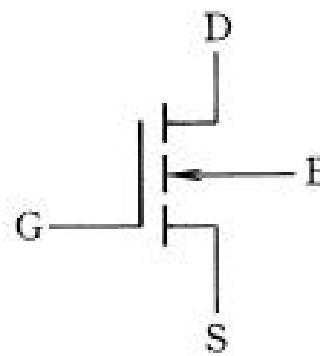
导通

C_I 为栅极输入电容，通常为几个pF；

R_{ON} 为导通电阻，约为1K欧姆以内，与 V_{GS} 有关，不可忽略

§ 3.3 MOS-FET元件的开关特性

MOS管的四种类型（衬底与沟道类型）

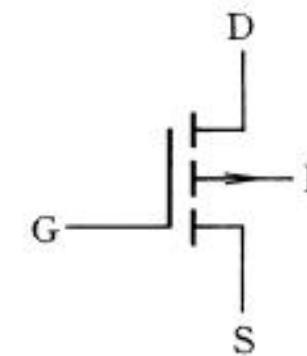


N沟道增强型

N沟道, P衬底

载流子为电子

正电压导通

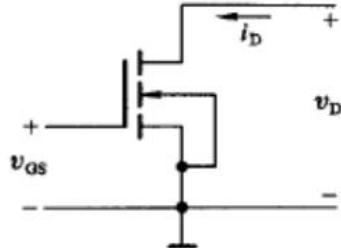


P沟道增强型

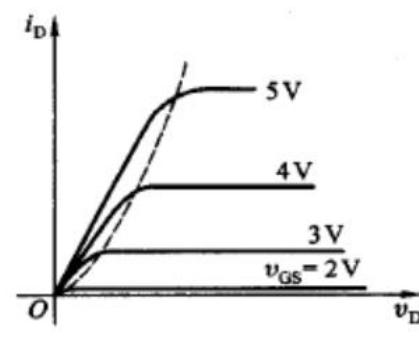
P沟道, N衬底

载流子为空穴

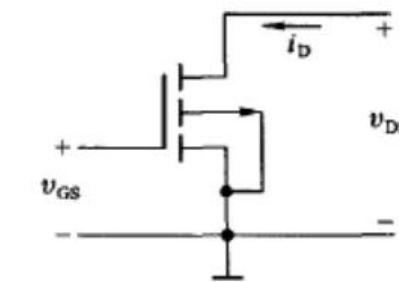
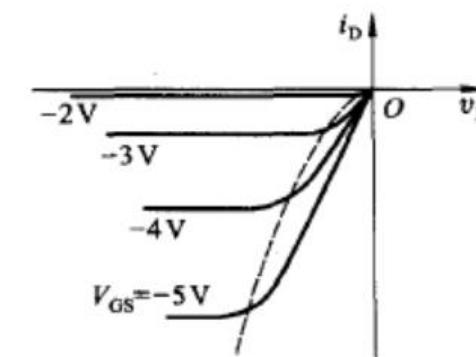
负电压导通



(a)

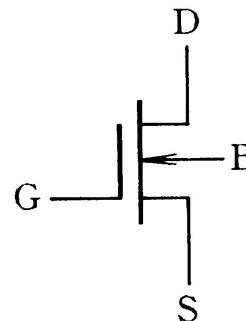


(b)



§ 3.3 MOS-FET元件的开关特性

■ MOS管符号的四种类型

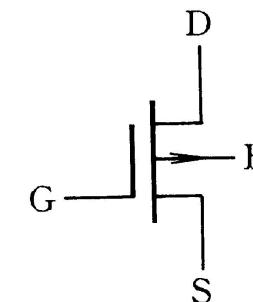


N沟道耗尽型

掺正离子

N沟道, P衬底

载流子为电子



P沟道耗尽型

掺负离子

P沟道, N衬底

载流子为空穴

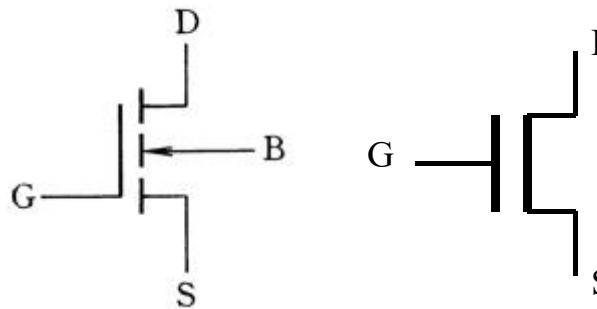
V_{GS} 为负时, 导通沟道变窄,
直到小于某个阈值 (夹断电
压 $V_{GS(off)}$) , 截止

V_{GS} 为正时, 导通沟道变窄,
直到大于某个阈值 (夹断电
压 $V_{GS(off)}$) , 截止

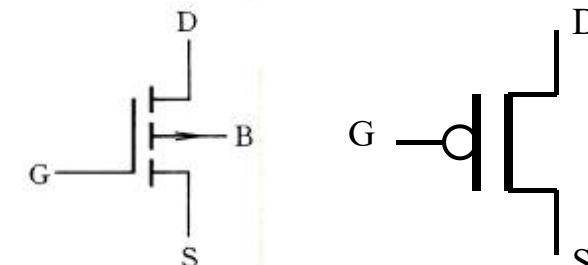
$V_{GS} = 0$ 时导电沟道即已经存在

§ 3.3 MOS-FET元件的开关特性

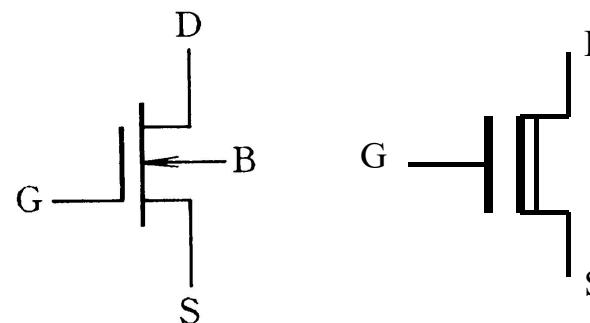
■ MOS管符号的符号与简化画法



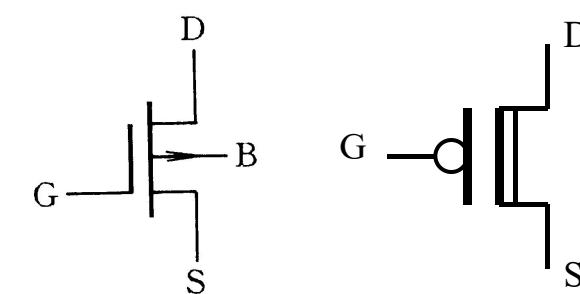
N沟道增强型



P沟道增强型



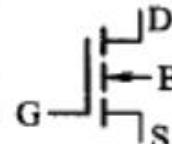
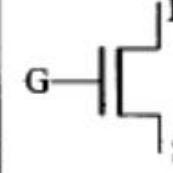
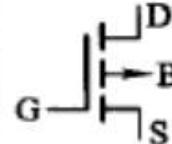
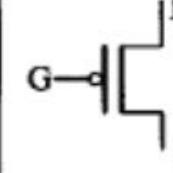
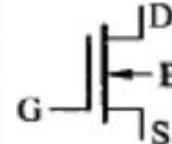
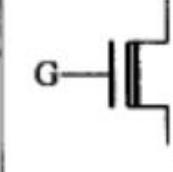
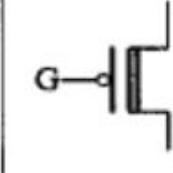
N沟道耗尽型



P沟道耗尽型

§ 3.3 4种MOS-FET比较

表 3.3.1 四种类型 MOS 管的比较

MOS 管 类型	衬底 材料	导电 沟道	开启 电压	夹断 电压	电压极性		标准符号	简化符号
					v_{DS}	v_{GS}		
N 沟道 增强型	P 型	N 型	+		+	+		
P 沟道 增强型	N 型	P 型	-		-	-		
N 沟道 耗尽型	P 型	N 型		-	+	\pm		
P 沟道 耗尽型	N 型	P 型		+	-	\mp		



§ 3.4 CMOS门电路

■ CMOS

- Complement Metal-Oxide-Semiconductor
- 互补金属氧化物半导体

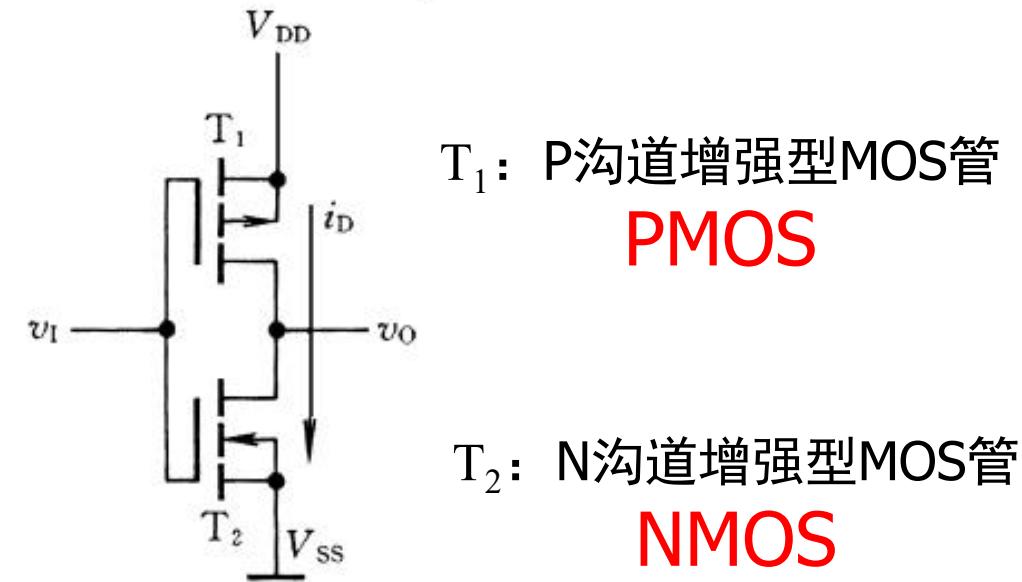
■ 提纲：

- CMOS非门的结构与原理
- CMOS逻辑门（及其扩展）
- CMOS逻辑门电路系列

§ 3.4.1 CMOS非门的结构与原理

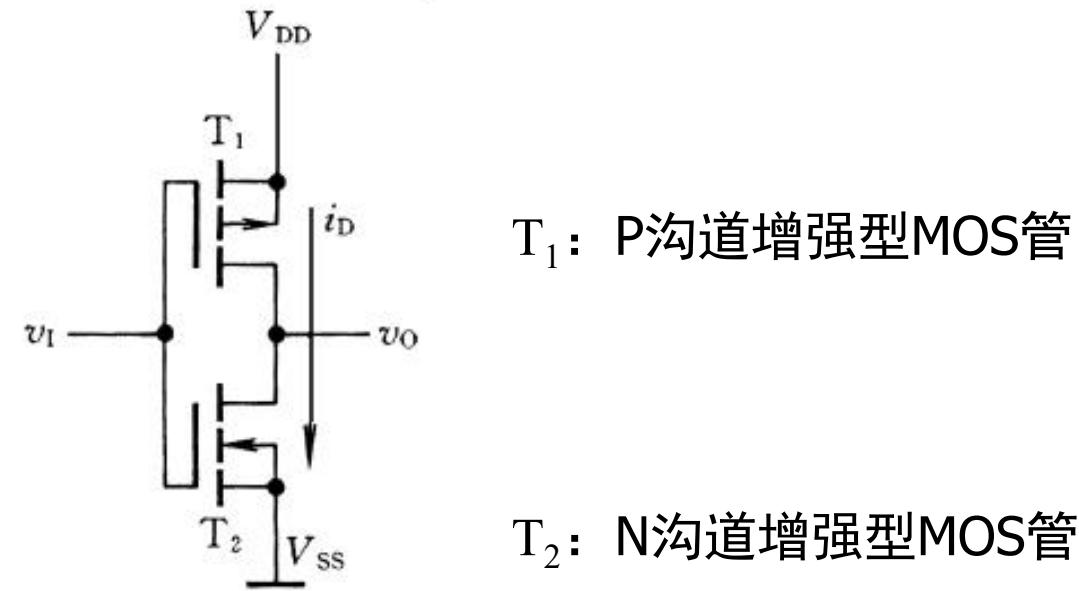
■ CMOS非门

- 由参数对称的增强型N沟道和P沟道MOS FET构成；
- 通常称为**互补型**MOS逻辑电路。



在CMOS门电路中，通常PMOS与NMOS晶体管总是成对出现

CMOS非门



- 假设电源 V_{DD} 大于两管开启电压绝对值之和，即 $V_{DD} > (V_{GS(th)N} + |V_{GS(th)P}|)$ ，且 $V_{GS(th)N} = |V_{GS(th)P}|$ ；
- ✓ $v_I=0V$ 时， T_2 截止， T_1 导通（有沟道）， T_2 的截止电阻约为 $1000M\Omega$ ， T_1 的导通电阻约为 750Ω ，所以输出 $v_O \approx V_{DD}$ ，即 v_O 为高电平；
- ✓ $v_I=V_{DD}$ 时， T_2 导通（有沟道）， T_1 截止， T_2 的导通电阻约为 750Ω ， T_1 的截止电阻约为 $1000M\Omega$ ，所以输出 $v_O \approx 0V$ ，即 v_O 为低电平；
- ✓ 电路实现了非门逻辑。



CMOS非门特点

- 1、无论输入是高还是低，T1和T2总是工作在一个导通而另一个截止，处于**互补状态**，因此称为互补对称式金属-氧化物半导体电路，简称CMOS电路
- 2、由于静态情况下，T1和T2总有一个处于截止状态，截止电阻很大（近似无穷大），因此**静态电流很小**，这是CMOS电路最突出的特点

§ 3.4.1 CMOS非门的结构与原理

■ 电压传输（输入输出）特性

➤ 设：CMOS非门的电源电压

$$V_{DD}=10V,$$

➤ 设：两管的开启电压为

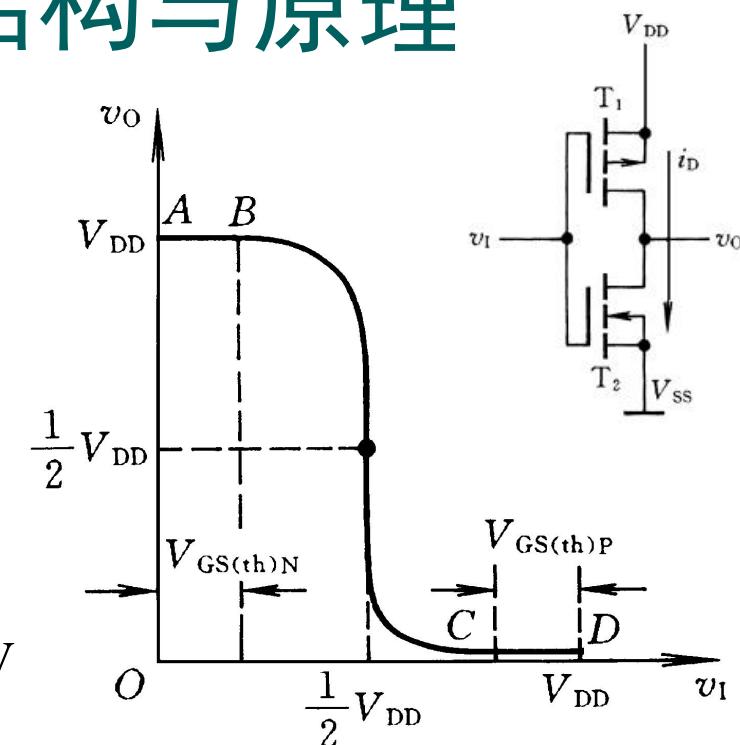
$$V_{GS(th)N}=|V_{GS(th)P}|=2V。$$

✓ 当 $v_I < 2V$, T_2 截止, T_1 导通, 输出 $v_O \approx V_{DD} = 10V$

✓ 当 $8V < v_I < 10V$, T_1 截止, T_2 导通, 输出 $v_O \approx 0V$

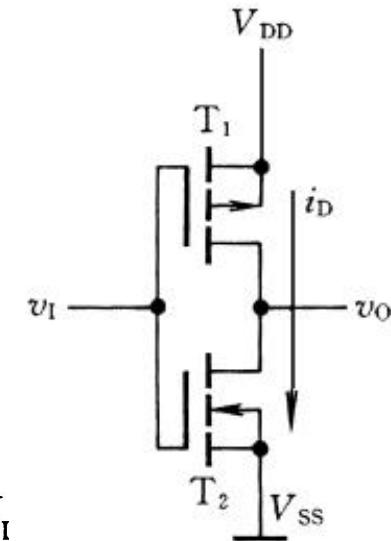
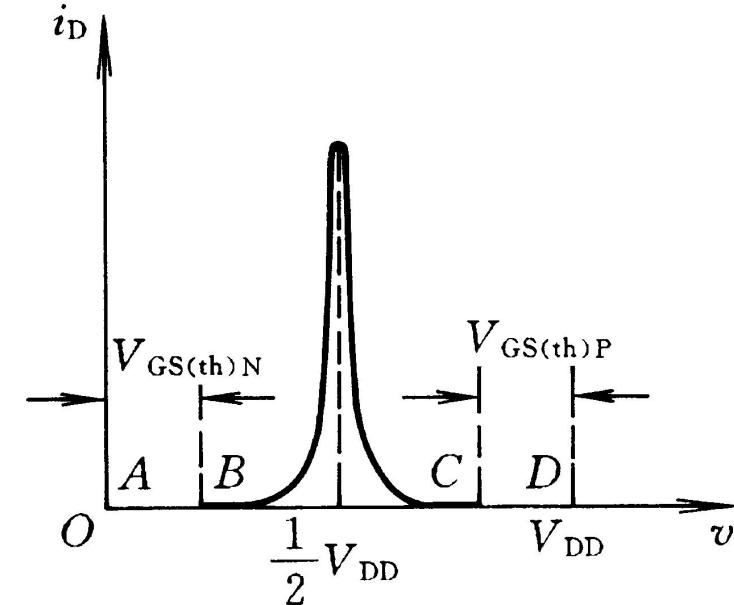
✓ 当 $2V < v_I < 8V$, T_2 和 T_1 都导通, 由于两管参数的对称性, 当 $v_I = 5V$, T_2 的栅源电压 = T_1 栅源电压绝对值, $v_O = (V_{DD}/2) = 5V$

✓ 两管在 $v_I = V_{DD}/2$ 处于转换状态, CMOS门电路的阈值电压 $V_{th} = V_{DD}/2$



§ 3.4.1 CMOS非门的结构与原理

■ 电流传输特性



- ✓ 在AB段， T_2 截止，内阻高，漏极电流几乎为0；
- ✓ 在CD段， T_1 截止，内阻高，漏极电流几乎为0；
- ✓ 在BC段， T_2 和 T_1 都导通，两管在处转换状态，在 $v_I=V_{DD}/2$ 处，电流最大。
- ✓ CMOS门电路**不可长期工作**在BC段。

§ 3.4.2 CMOS反相器的输入输出特性

■ 噪声容限

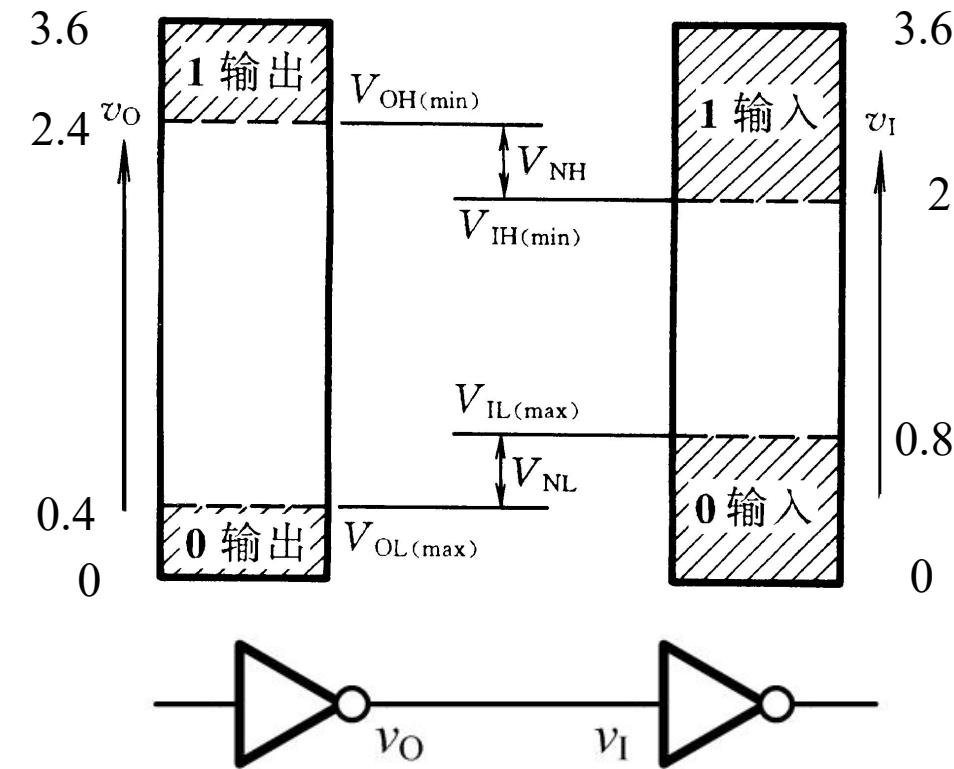
➤ 噪声容限表示门电路的抗干扰能力

➤ 低电平噪声容限

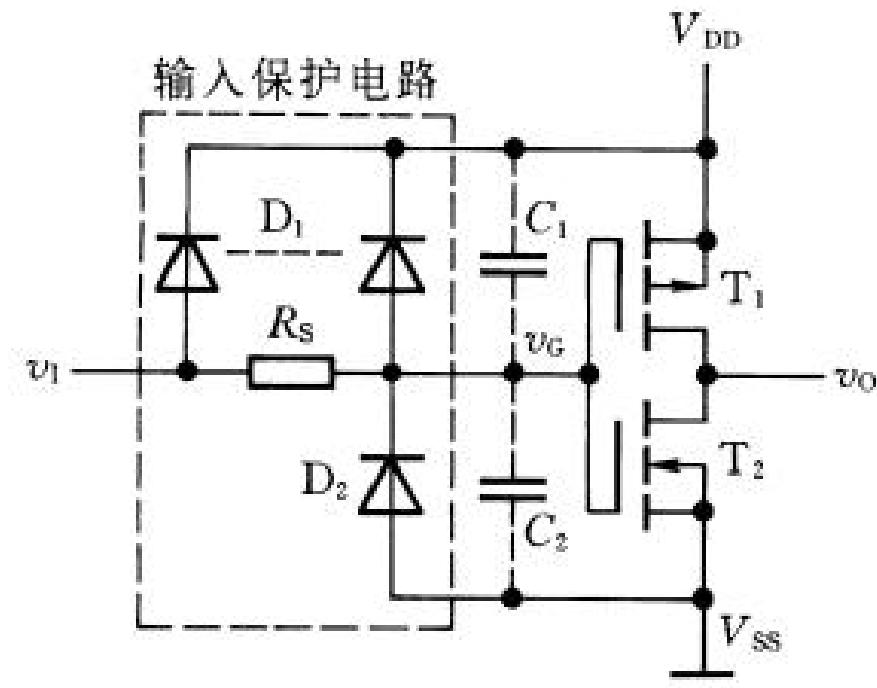
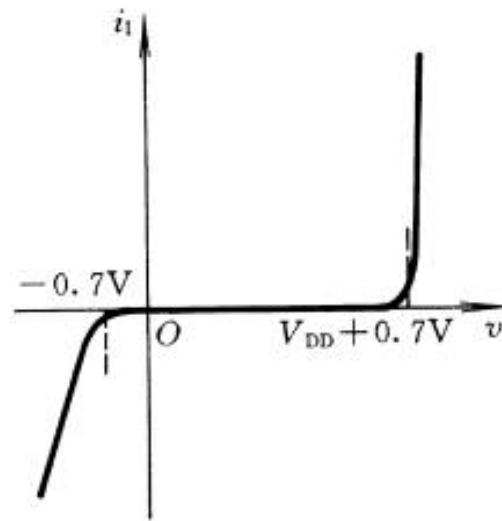
$$V_{NL} = V_{IL(\max)} - V_{OL(\max)}$$

➤ 高电平噪声容限

$$V_{NH} = V_{OH(\min)} - V_{IH(\min)}$$



■ 输入特性

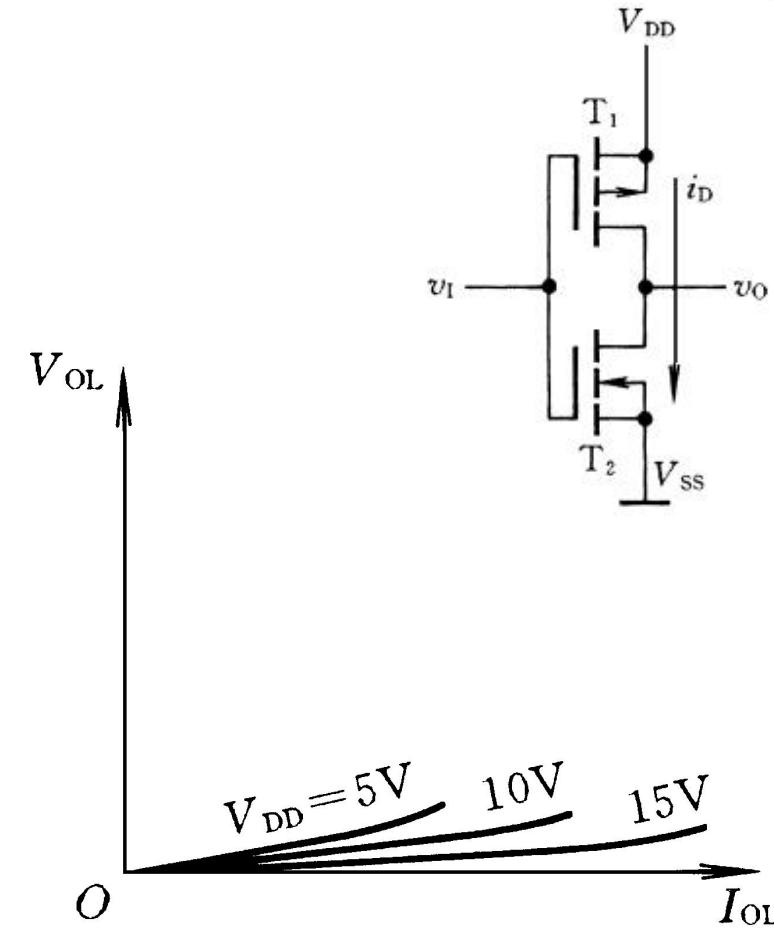
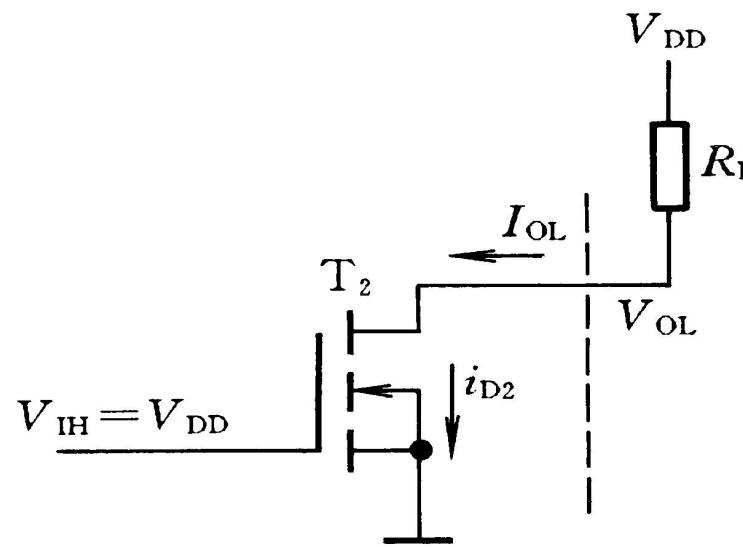


(4000系列)

- ✓ 棚极绝缘层易击穿，接入保护电路
- ✓ 当电压低于 $-0.7V$ 时， D_2 管导通， V_G 钳制在 $-0.7V$
- ✓ 当电压高于 $V_{DD}+0.7V$ 时， D_1 管导通， V_G 钳制在 $V_{DD}+0.7V$
- ✓ 当电压在 $-0.7V$ 到 $V_{DD}+0.7V$ 之间，保护电路透明，不影响正常工作

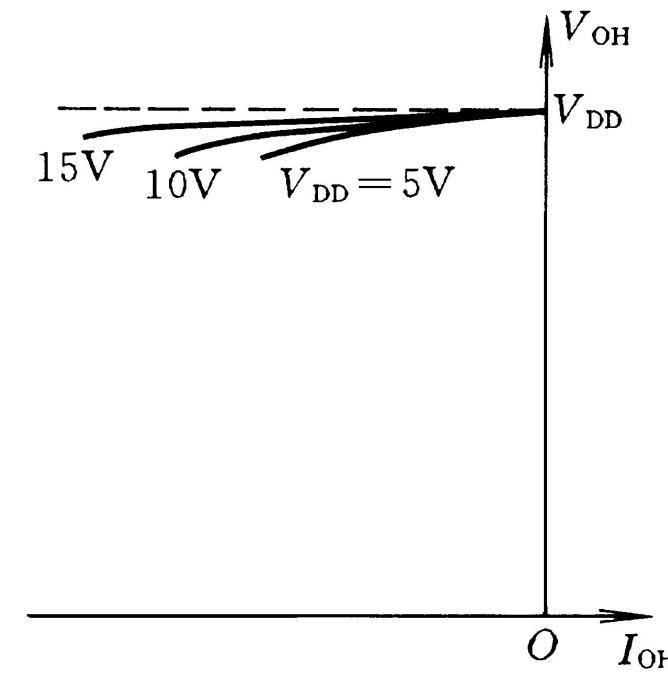
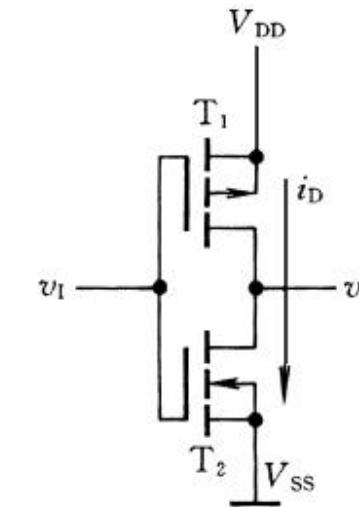
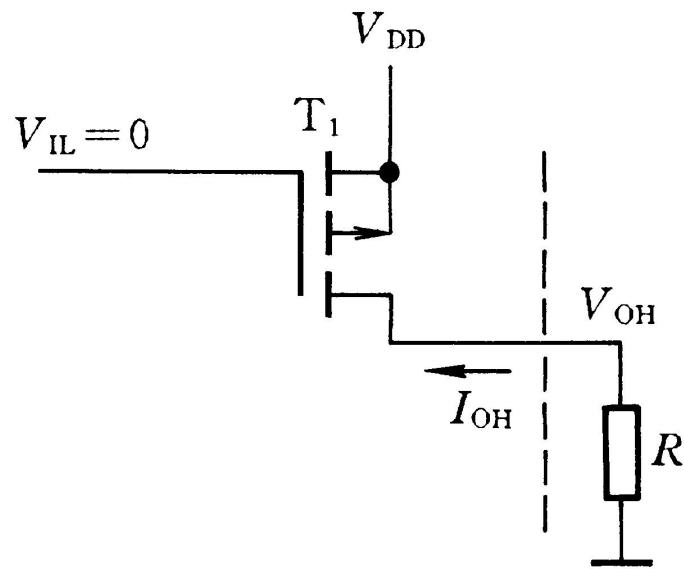
■ 输出特性

➤ 低电平输出特性

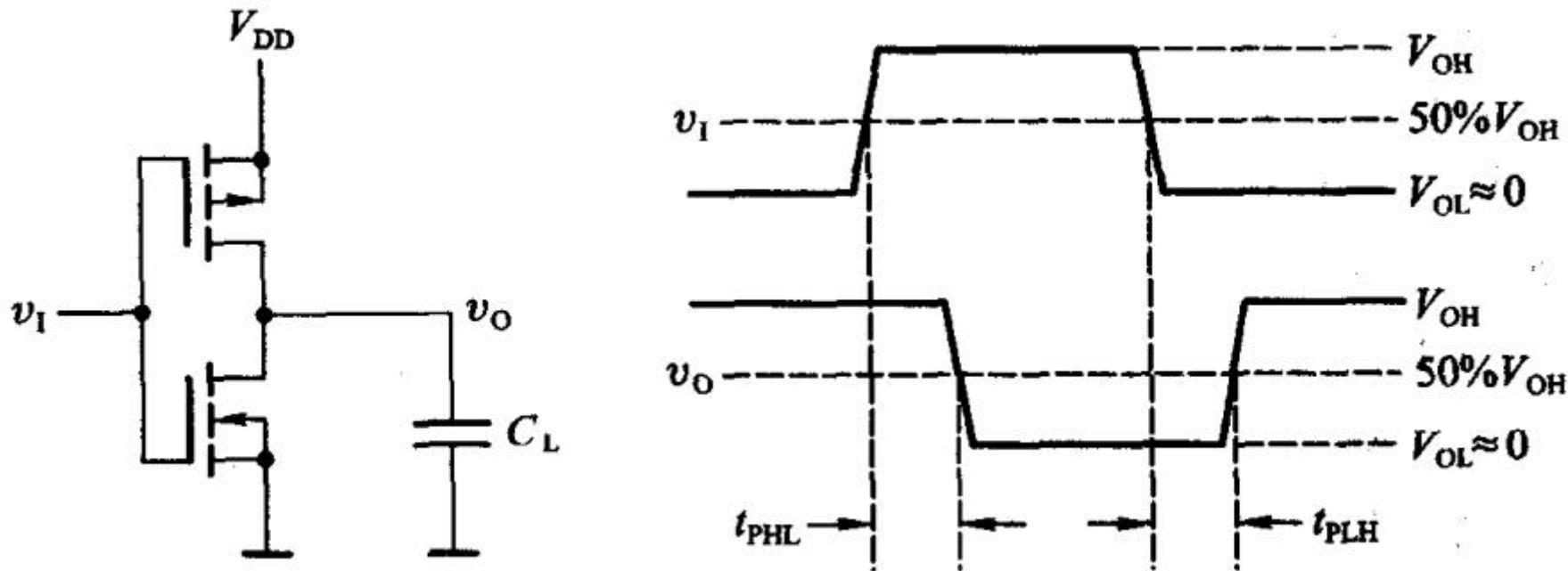


■ 输出特性

➤ 高电平输出特性



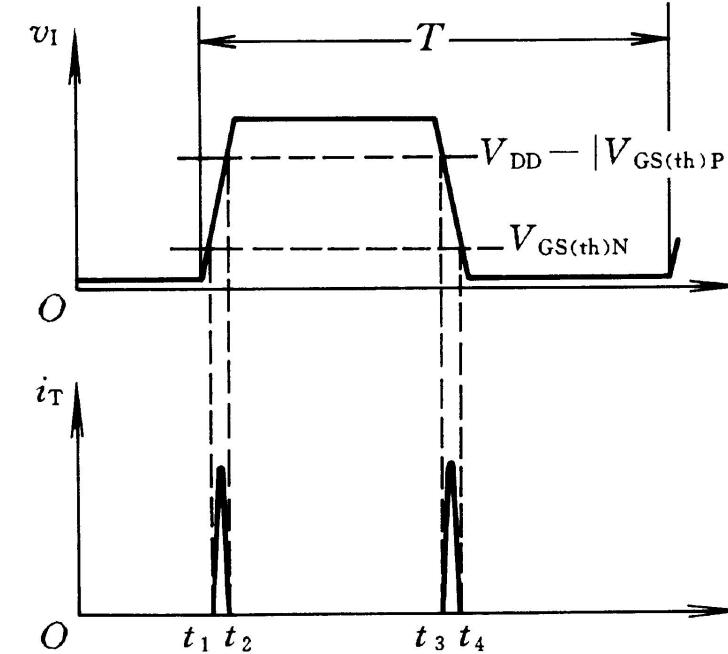
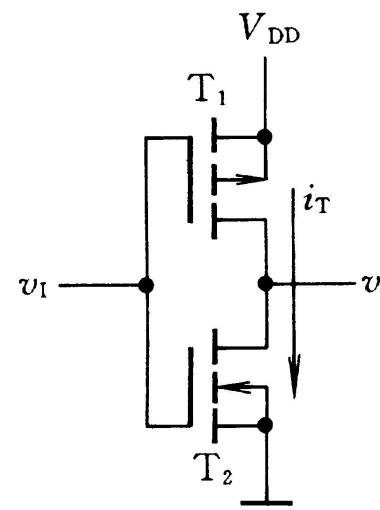
■ 动态特性：传输延迟



t_{PHL} 和 t_{PLH} 通常是相等的

§ 3.4.1 CMOS非门的结构与原理

■ 功耗



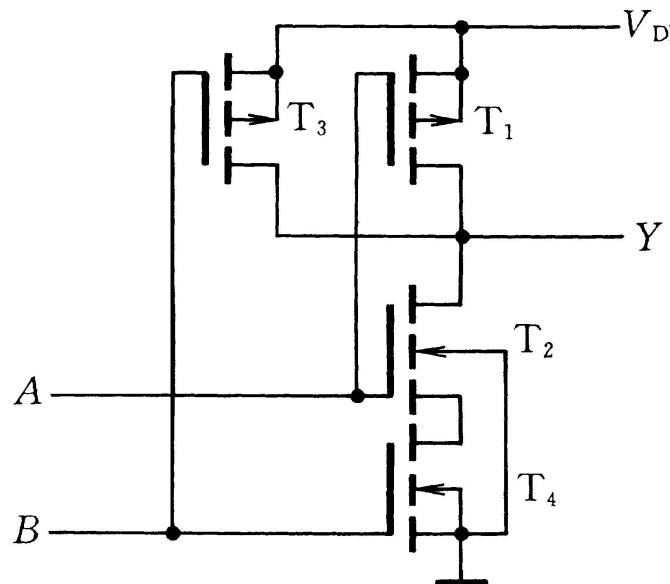
- 静态功耗小，可忽略（互补结构）
- 高频、动态情况会产生动态功耗
 - ✓ 电容充放电+导通电流
- f ：输出信号的转换频率

器件的动态导通损耗 $P_T = C_{PD} \cdot V_{DD}^2 \cdot f$
(当 f 加大时成立)

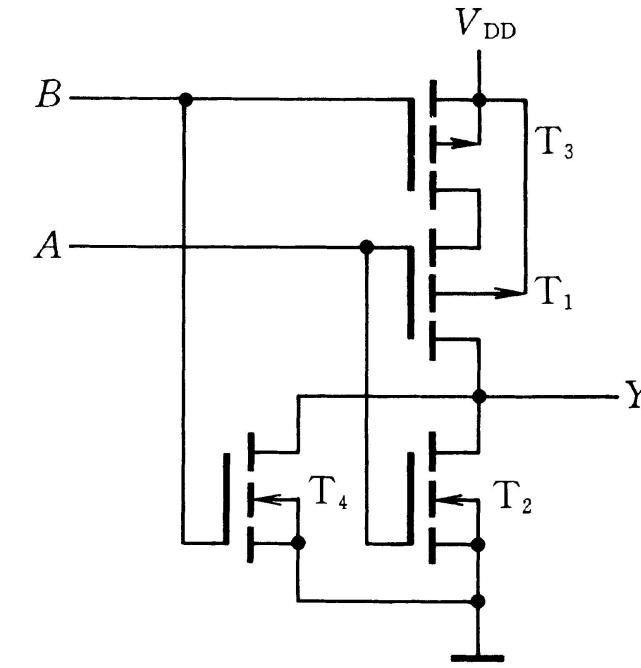
负载的充放电动态功耗 $P_L = C_L \cdot V_{DD}^2 \cdot f$

总功耗为两者之和

§ 3.4.2 CMOS逻辑门



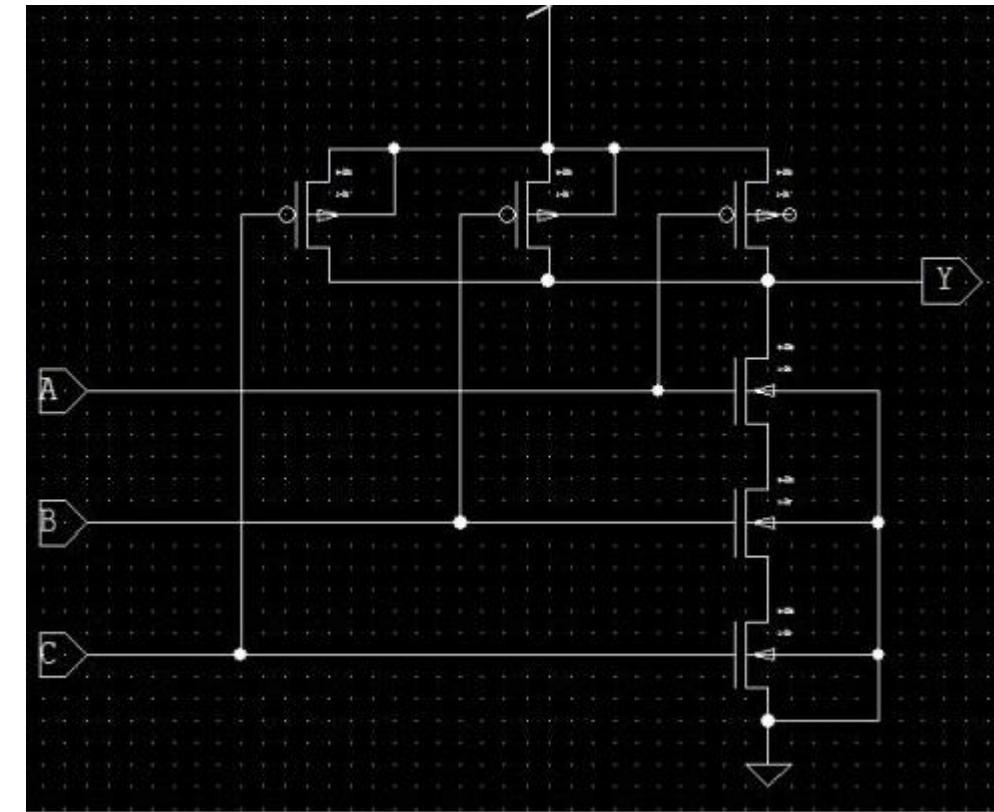
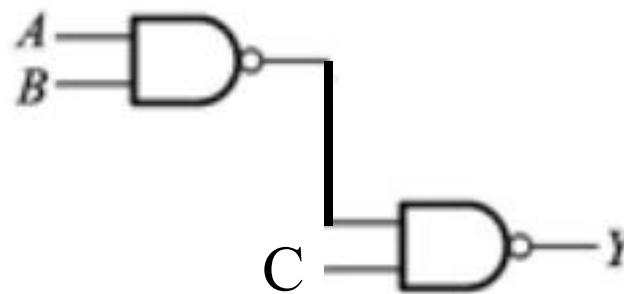
✓ 与非门



✓ 或非门

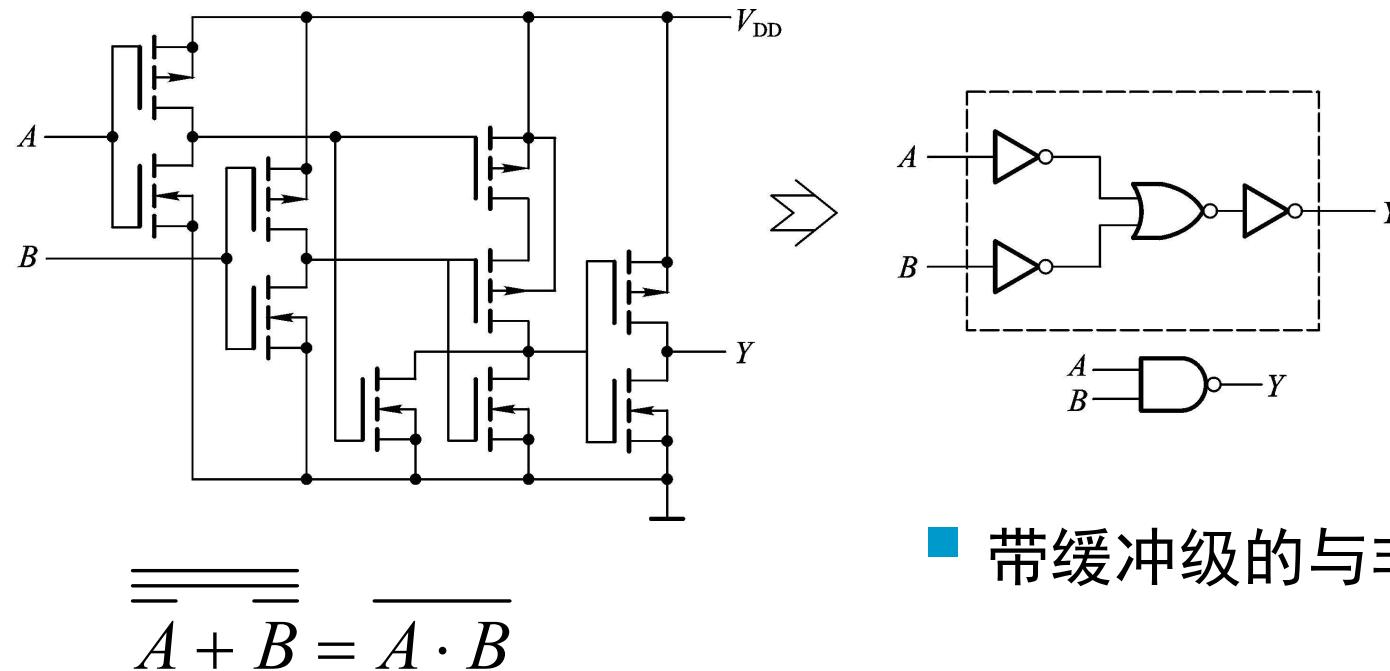
§ 3.4.2 CMOS逻辑门

- 设计一个3输入与非门



§ 3.4.2 CMOS逻辑门

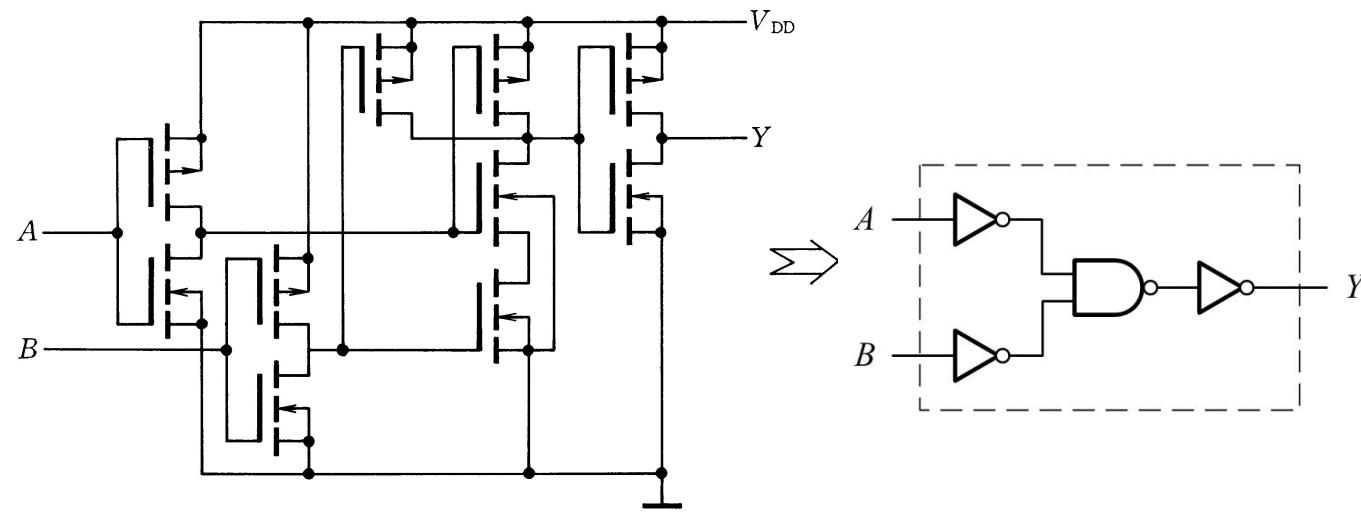
- ✓ 输入端数目增加时，“与非门”中串联的NMOS管数目要增加，引起输出低电平变高；
- ✓ 或非门串联的PMOS管数目增加，引起输出高电平变低；
- ✓ 解决方法在输入输出端分别加入反相器作缓冲级。



■ 带缓冲级的与非门

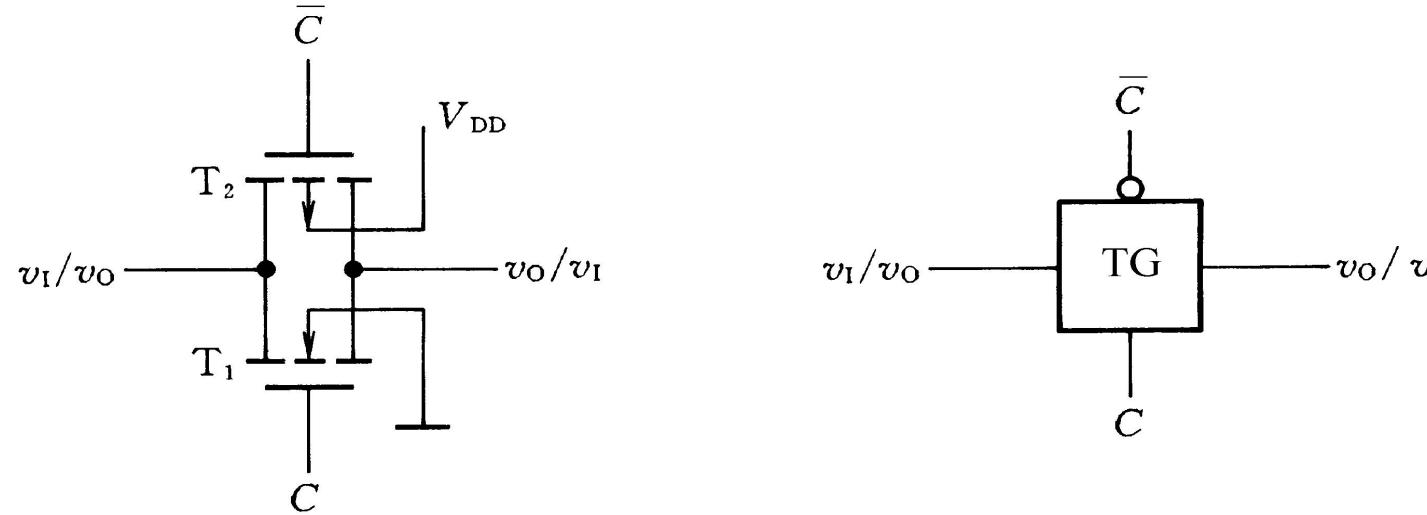
§ 3.4.2 CMOS逻辑门

■ 带缓冲级的或非门



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

§ 3.4.3 CMOS逻辑门 —— 传输门



- ✓ C 接高电平 V_{DD} ，而 \bar{C} 接0V，若 $0V \leq v_I \leq (V_{DD} - V_{GS(th)N})$ ，T₁导通；
- ✓ 若 $|V_{GS(th)P}| \leq v_I \leq V_{DD}$ ，T₂导通。
- ✓ v_I 在 $0V \sim V_{DD}$ 变化时，至少有一管导通，输出与输入之间呈低电阻， $v_O \sim v_I$ ，相当于开关闭合；
- ✓ C 接低电平0V， v_I 在 $0V \sim V_{DD}$ 的范围变化时，T₁和T₂都截止，输出呈高阻态，输入电压不能传到输出端，相当于开关断开。

§ 3.4.3 CMOS逻辑门

■ 传输门（续）

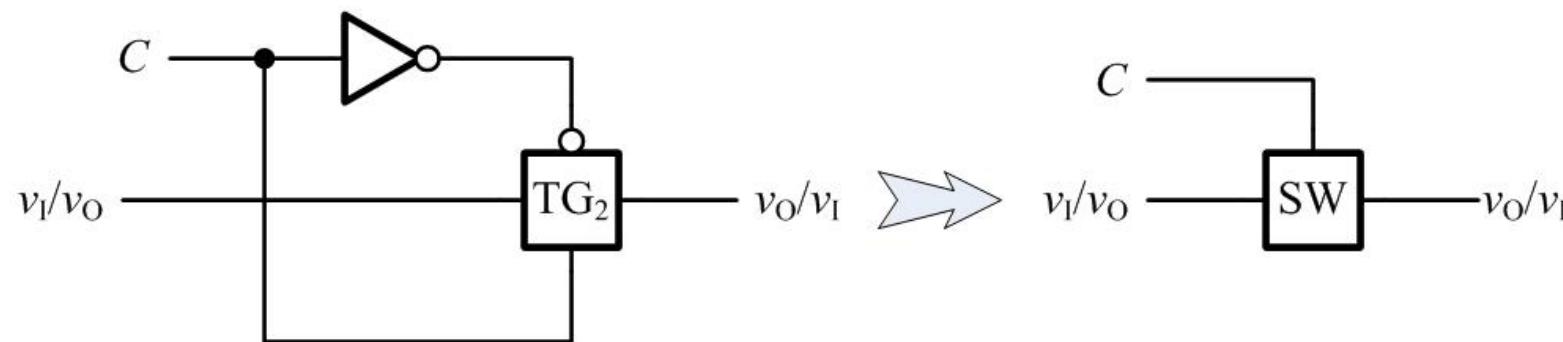
➤ 用途

- 模拟开关

- ◆ 传输连续变化的模拟信号
- ◆ 数字量控制

- 双向器件

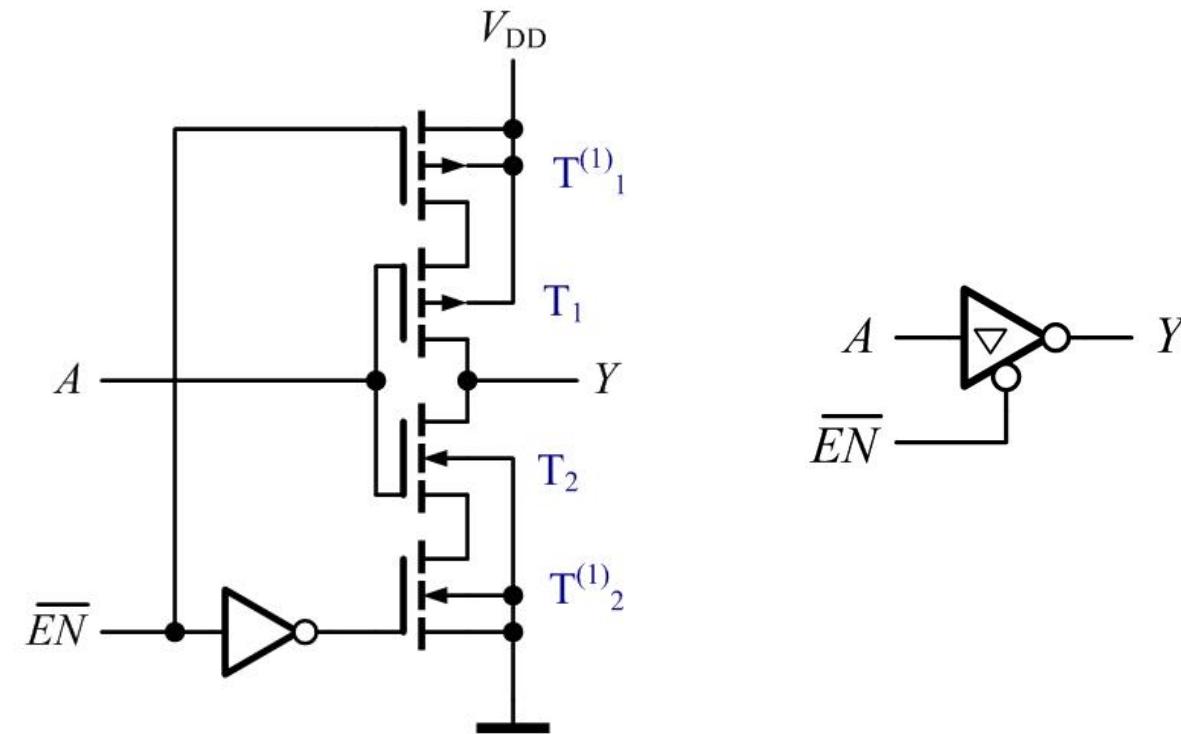
- ◆ 双向传输



§ 3.4.4 CMOS逻辑门——三态门

高电平态、低电平态、高阻态

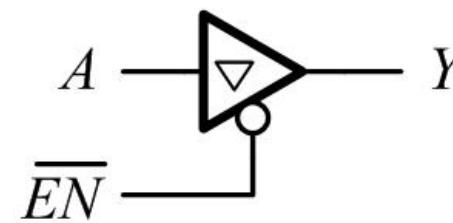
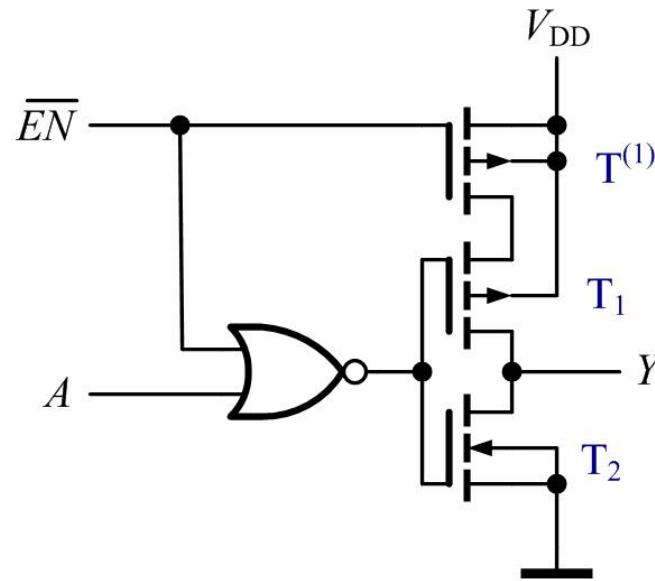
结构1：



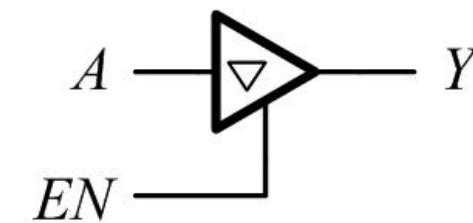
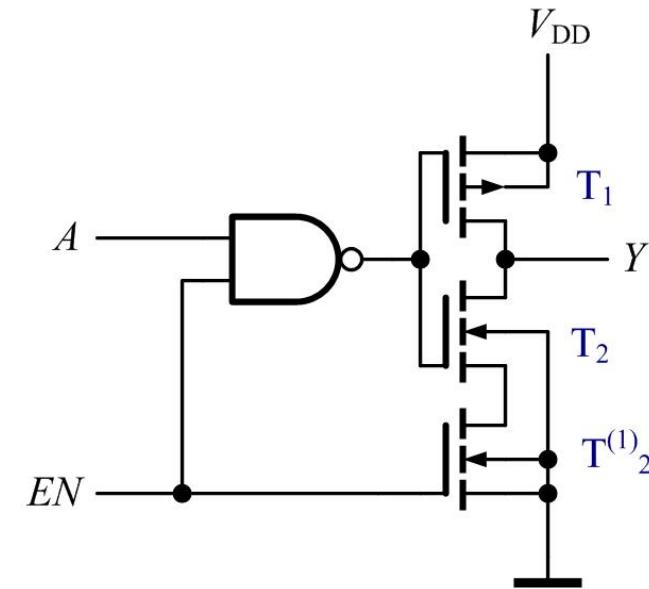
- ✓ $EN=1$ 时， $T^{(1)}_1$ 和 $T^{(1)}_2$ 同时导通， T_1 和 T_2 组成的非门正常工作
- ✓ $EN=0$ 时， $T^{(1)}_1$ 和 $T^{(1)}_2$ 同时截止，输出对地和对电源呈高阻状态

结构2：

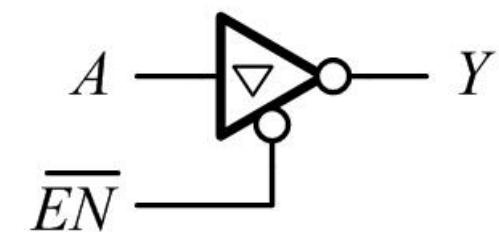
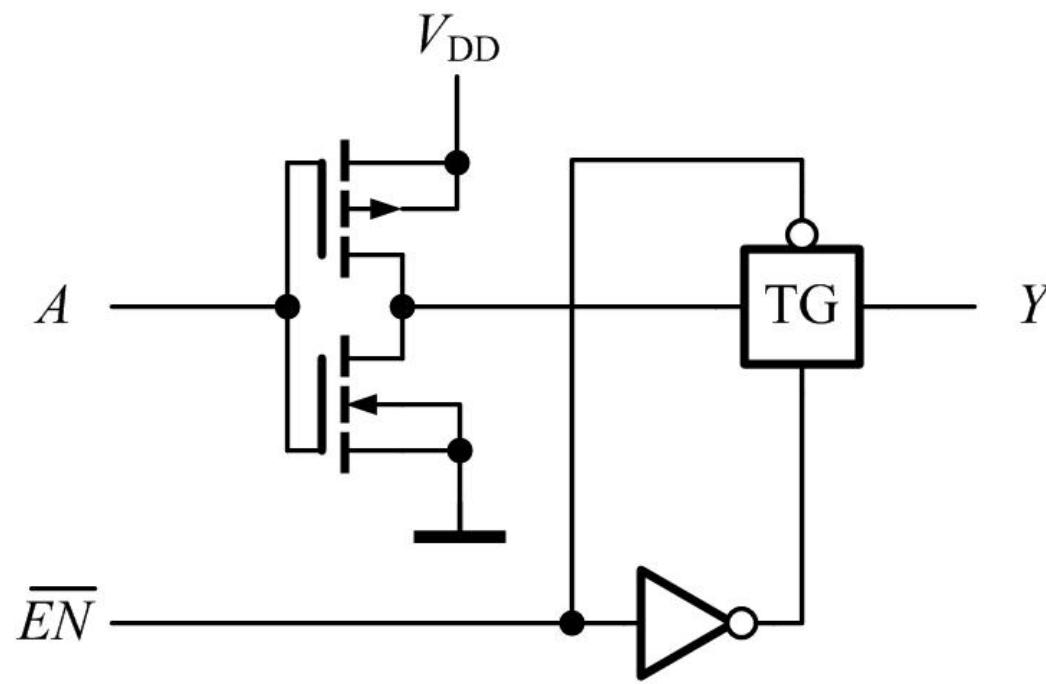
✓或非门控制三态门



✓与非门控制三态门

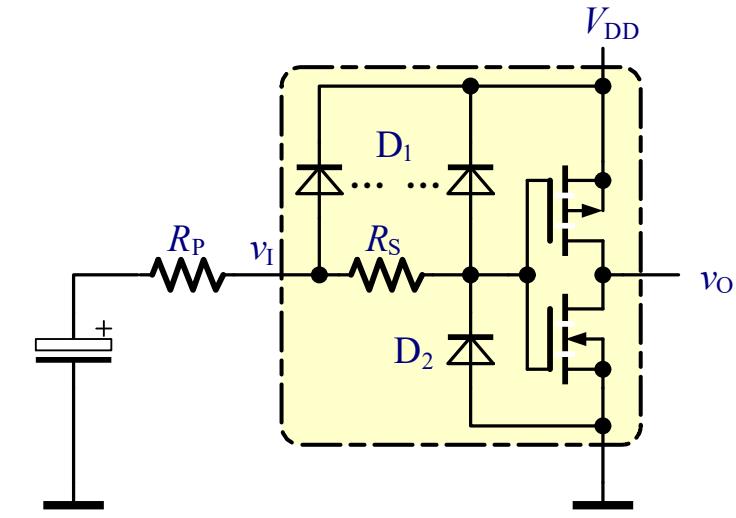


结构3：用传输门组成的三态门

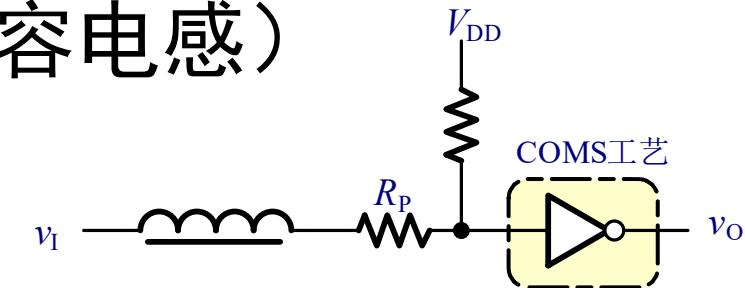


§ 3.4.6 使用CMOS门电路的注意事项

- 输入电路的防静电
——不用的引脚不能悬空



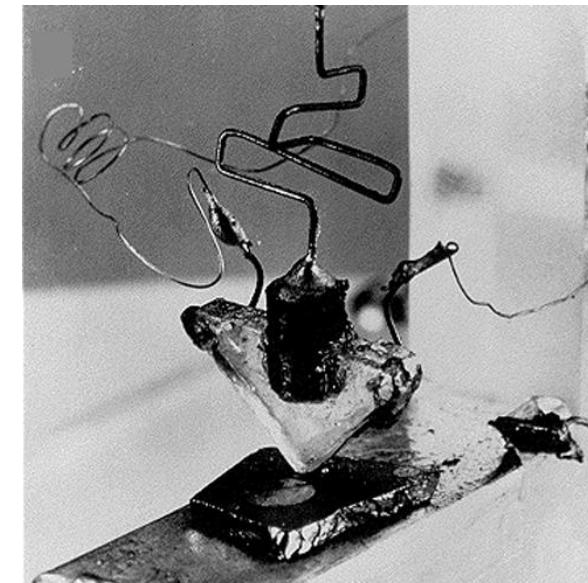
- 输入电路的过流保护
——低阻信号源 (过流)
——输入端大电容 (大瞬态电流)
——输入端接长线 (分布电容电感)



解决方法：串联保护电阻

附录3-1：三极管的发明

- 1947年12月23日，37岁的美国物理学家**肖克利**和他的合作者（**巴丁、布拉顿**）在著名的贝尔实验室向人们展示了第一个半导体电子增幅器，即最初的晶体管。



1956年，肖克利、巴丁、布拉顿三人，因发明晶体管同时荣获诺贝尔物理学奖。

仙童半导体（英特尔）与叛逆八人帮



罗伯特·诺伊斯 (Robert Noyce)

戈登·摩尔 (Gordon Moore)

金·赫尔尼 (Jean Hoerni)

朱利叶斯·布兰克 (Julius Blank)

尤金·克莱纳 (Eugene Kleiner)

杰·拉斯特 (Jay Last)

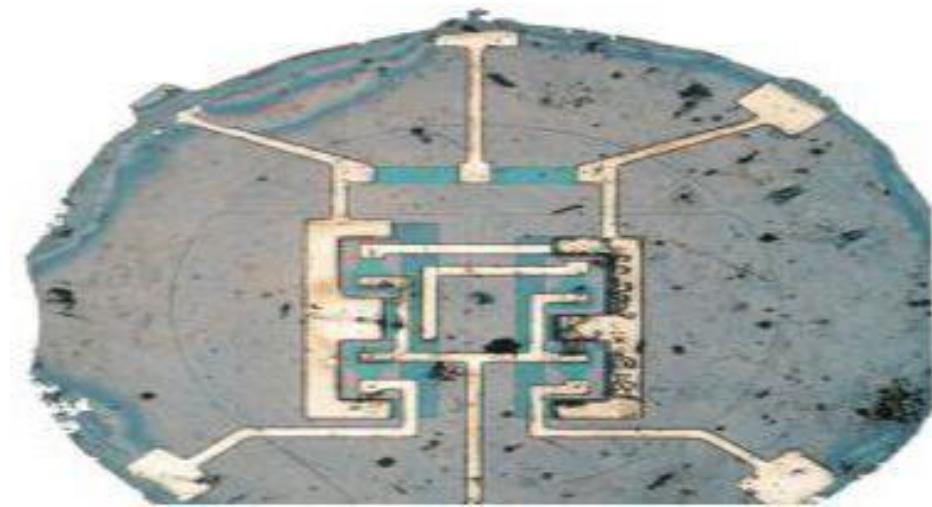
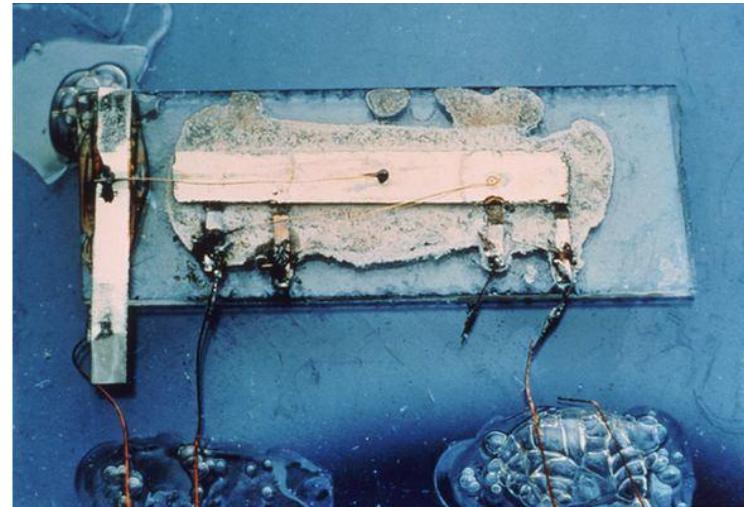
谢尔顿·罗伯茨 (Sheldon Roberts)

维克多·格里尼克 (Victor Grinnich)

1957年9月18日，八个天才出走肖克利实验室，成立仙童半导体

附录3-2：集成电路的发明

- 1958年：仙童公司Robert Noyce与德州仪器公司基尔比间隔数月分别发明了集成电路，开创了世界微电子学的历史，奠定了信息时代的基础



基尔比因为集成电路的发明2000年，获得诺贝尔奖



第三章 习题

第五版（阎石主编）

- 3.3; 3.4; 3.7;

第四章 布尔代数



数字集成电路基础

——布尔代数与逻辑函数

张悦

微电子学院

费尔北京研究院 / 自旋电子交叉学科中心

2020-10-2





目 录

2. 1 逻辑代数运算

2. 2 逻辑函数的表示方法
及其标准形式

2. 3 逻辑函数的化简



布尔代数与逻辑函数

1. 2. 二进制

- (1) 计数符号: 0, 1 .
- (2) 进位规则: 逢二进一.
- (3) 二进制数按权展开式

$$(N)_2 = \sum_{i=-m}^{n-1} a_i \times 2^i$$

数字电路中采用二进制的原因:

- 1) 数字装置简单可靠;
- 2) 二进制数运算规则简单;
- 3) 数字电路既可以进行算术运算, 也可以进行逻辑运算.



2.1 逻辑代数运算

- 逻辑变量与逻辑函数
- 逻辑运算
- 逻辑代数的公理和基本公式
- 逻辑代数的基本定理
- 逻辑代数的常用公式

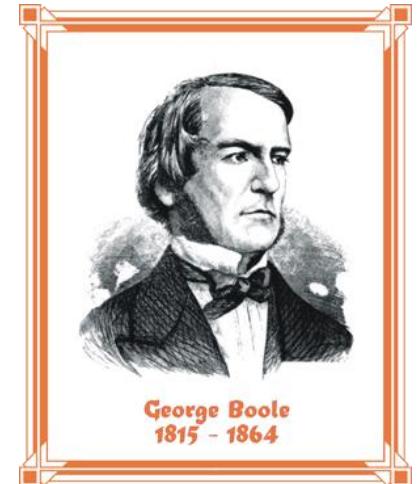
布尔代数与逻辑函数

➤ 逻辑代数

- 1848年 **George Boole** (爱尔兰/英, 1815~1864)
- 一种符号逻辑(数理逻辑)
- 又被称为: 布尔代数、开关代数

➤ 逻辑变量与逻辑函数

- 逻辑代数中的变量称为**逻辑变量**;
- 用字母**A**、**B**、**C**、...表示;
- 只能有两种可能的取值: **真或假**;
- 习惯上, 把真记作“**1**”, 假记作“**0**”;
- “**1**”和“**0**”不表示数量的大小, 表示完全对立的两种状态。
- 逻辑变量表示**数字逻辑**的状态
- 逻辑变量输入输出之间构成**函数关系**



布尔代数与逻辑函数

◆ 例：

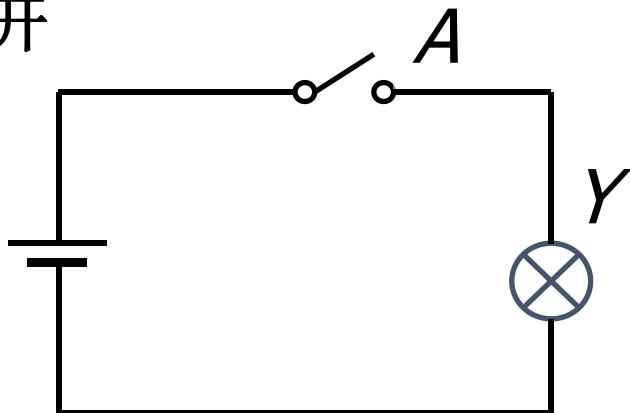
- 指示灯 Y 是否点亮取决于开关 A 是否接通
- 定义 $Y=1$ 表示灯亮， $Y=0$ 表示灯灭
- $A=1$ 表示开关接通， $A=0$ 表示开关断开
- Y 是 A 的函数，逻辑函数表达式

$$Y = A$$

- Y 和 A 都称为逻辑变量

A 称为输入逻辑变量（简称逻辑变量）

Y 称为输出逻辑变量（简称逻辑函数）





➤ 逻辑运算

通过逻辑变量的运算得到逻辑函数的值

◆ 基本逻辑运算:

逻辑与 (AND)

逻辑或 (OR)

逻辑非 (NOT)

◆ 复合逻辑运算

复合逻辑运算由基本逻辑运算组合而成
如与非、或非、同或、异或等

布尔代数与逻辑函数

◆ 基本逻辑运算

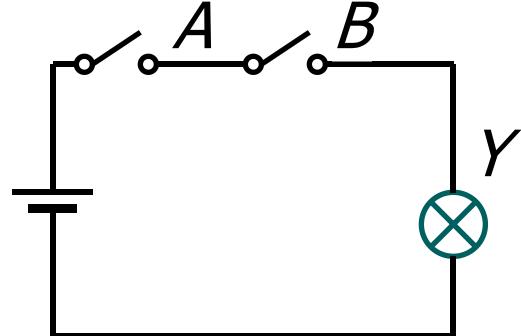
■ 与 (AND) 运算

逻辑表达式：

$$Y = A \cdot B$$

- 其中，“ \cdot ”为逻辑“与”运算符，也可以被省略
- 用真值表描述

真值表：描述各个变量取值组合和函数取值之间对应关系。



真值表

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

布尔代数与逻辑函数

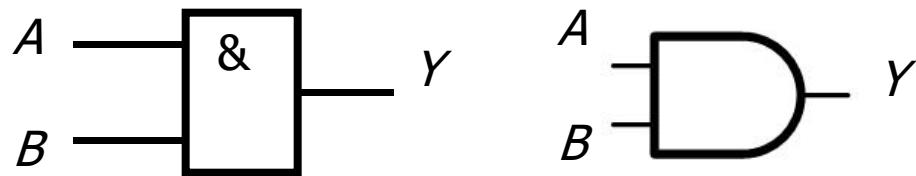
- 逻辑“与”的含义

只有当决定一件事情的所有条件都**全部**具备时，这件事情才会发生；

- 与门

在**逻辑电路**中，能够实现“与”运算的**基本单元**

- 逻辑符号



国标

美标

布尔代数与逻辑函数

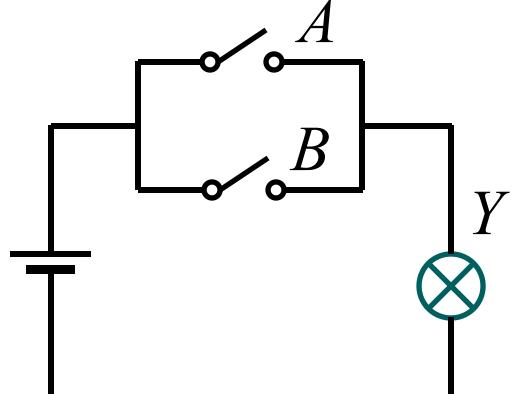
■ 或 (OR) 运算

逻辑表达式:

$$Y = A + B$$

■ 其中, “+” 为 “或” 运算符

■ 真值表



真值表

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

布尔代数与逻辑函数

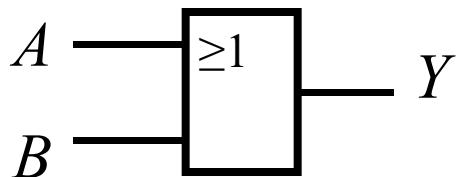
- 逻辑“或”的含义：

在决定一件事情的各条件下，只要有一个或一个以上条件具备，这件事情就发生。

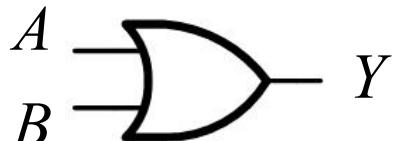
- 或门

在逻辑电路中，能够实现“或”运算的基本单元。

- 逻辑符号



国标



美标

布尔代数与逻辑函数

■ 非 (NOT) 运算

逻辑表达式：

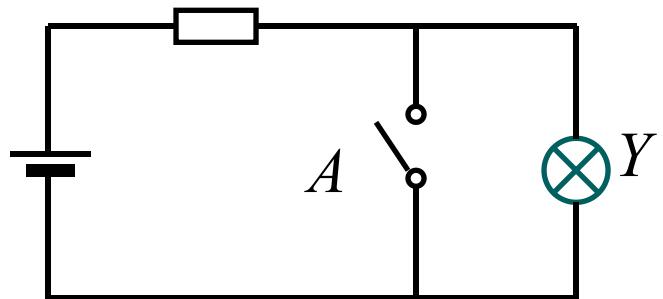
$$Y = \overline{A}$$

或

$$Y = A'$$

读作 “ A 非” 或 “非 A ”

■ 真值表



真值表

A	Y
0	1
1	0

布尔代数与逻辑函数

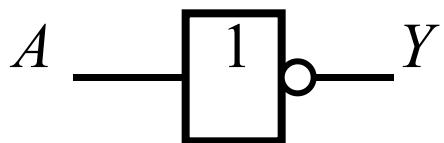
- 逻辑“非”的含义：

当条件**不具备**时，事情才会发生

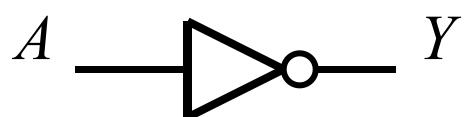
- **非门**

在**逻辑电路**中，实现“**非**”运算的**基本单元**

- 逻辑符号：



国标

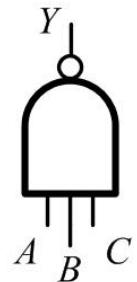
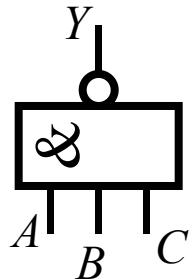


美国

布尔代数与逻辑函数

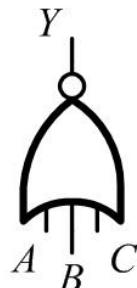
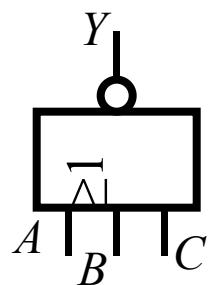
- ◆ 复合逻辑运算
- 与非运算

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



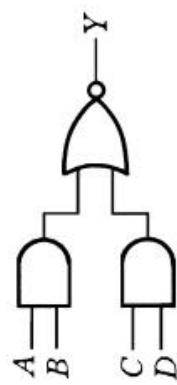
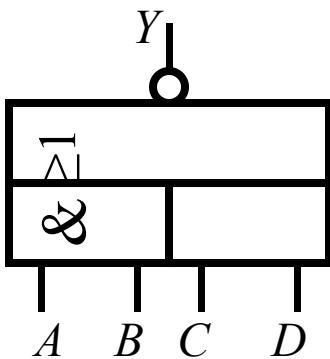
- 或非运算

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



- 与或非运算

$$Y = \overline{AB + CD}$$





◆ 复合逻辑运算

■ 逻辑运算的优先顺序：

- (1) 圆括号
- (2) 非运算
- (3) 与运算
- (4) 或运算

两种重要的复合逻辑-异或/同或

■ Exclusive-OR and Coincidence-OR

■ 异或 (XOR) 逻辑 (异或运算)

➤ 真值表

➤ 表达式: $A \oplus B = \overline{AB} + \overline{A}\overline{B}$

■ 同或 (XNOR) 逻辑

异或逻辑的非

➤ 表达式: $A \odot B = A \cdot B + \overline{A} \cdot \overline{B}$

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

模2加法

布尔代数与逻辑函数

- 逻辑符号
- 运算规则和基本公式

- 交换律

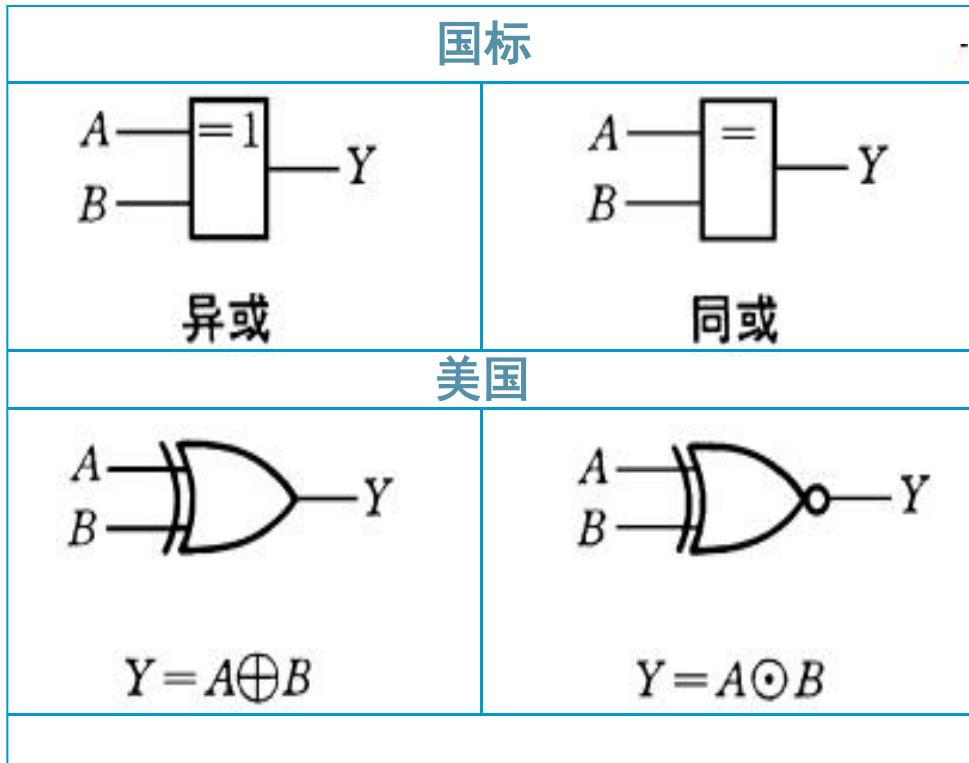
$$\begin{cases} A \oplus B = B \oplus A \\ A \odot B = B \odot A \end{cases}$$

- 反演律：

$$\begin{cases} \overline{A \oplus B} = \overline{A} \odot \overline{B} \\ \overline{A \odot B} = \overline{A} \oplus \overline{B} \end{cases}$$

- 互补律：

$$\begin{cases} A \oplus \overline{A} = 1 \\ A \odot \overline{A} = 0 \end{cases}$$



布尔代数与逻辑函数

◆ 逻辑电平

■ 正逻辑与负逻辑

- 对于一个逻辑电路，通常规定**高电平**为逻辑**1**，**低电平**为逻辑**0**，这就是**正逻辑**。反之，如果规定**高电平**为逻辑**0**，**低电平**为逻辑**1**，则称为**负逻辑**。
- 同一个逻辑电路，在不同的逻辑假定下，其逻辑功能是不同的。

A	B	F
V_L	V_L	V_L
V_L	V_H	V_L
V_H	V_L	V_L
V_H	V_H	V_H

(a) 电平关系

A B		F	A B		F
0	0	0	1	1	1
0	1	0	1	0	1
1	0	0	0	1	1
1	1	1	0	0	0

(b) 正逻辑

(c) 负逻辑

与

或



➤ 逻辑代数的公理和基本公式

◆ 逻辑代数的公理

$$(1) \bar{1} = 0 \quad \bar{0} = 1$$

$$(2) 1 \cdot 1 = 1 \quad 0 + 0 = 0$$

$$(3) 0 \cdot 0 = 0 \quad 1 + 1 = 1$$

$$(4) 1 \cdot 0 = 0 \cdot 1 = 0 \quad 0 + 1 = 1 + 0 = 1$$

(5) 如 $A \neq 0$ 则 $A = 1$, 如 $A \neq 1$ 则 $A = 0$



布尔代数与逻辑函数

◆ 逻辑代数的基本公式

组	名称	常用公式		备注
(1)	01律	$0 \cdot A = 0$	$1 + A = 1$	变量与常量
		$1 \cdot A = A$	$0 + A = A$	
(2)	重叠律	$A \cdot A = A$	$A + A = A$	同一个变量
(3)	互补律	$A \cdot \bar{A} = 0$	$A + \bar{A} = 1$	原变量与反变量之间的关系
(4)	还原律	$\bar{\bar{A}} = A$	--	
(5)	交换律	$A \cdot B = B \cdot A$	$A + B = B + A$	
(6)	结合律	$(A \cdot B) \cdot C = A \cdot (B \cdot C)$	$(A + B) + C = A + (B + C)$	
(7)	分配律	$A(B + C) = AB + AC$	$A + BC = (A + B)(A + C)$	
(8)	反演律	$\overline{A \cdot B} = \bar{A} + \bar{B}$	$\overline{A + B} = \bar{A} \cdot \bar{B}$	DeMorgan





■ 定律的证明方法

➤ 公理和法则

➤ 真值表

对于逻辑变量的所有可能的组合求逻辑函数的值。

例如：证明反演律

A	B	$\overline{A \cdot B}$	$\overline{A} + \overline{B}$	$\overline{A + B}$	$\overline{\overline{A} \cdot \overline{B}}$
0	0	1	1	1	1
0	1	1	1	0	0
1	0	1	1	0	0
1	1	0	0	0	0

得证: $\overline{A \cdot B} = \overline{A} + \overline{B}$ 得证: $\overline{A + B} = \overline{\overline{A} \cdot \overline{B}}$



布尔代数与逻辑函数

➤ 逻辑代数的基本定理

- 代入定理
- 反演定理 —— inversion theorem
- 对偶定理 —— dual theorem

■ 代入定理

在任何一个包含变量 A 的逻辑等式中，若以另外一个逻辑式代入式中的所有 A 的位置，则等式依然成立。



例 用代入定理证明**De Morgan**定理也使用于多变量的情况

已知二变量的**De Morgan**定理为

$$\overline{A + B} = \bar{A} \cdot \bar{B} \quad \overline{A \cdot B} = \bar{A} + \bar{B}$$

以 $(B+C)$ 代入左边等式中 B 的位置，同时以 $(\bar{B} \bar{C})$ 代入右边等式中 B 的位置，得

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

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

代入定理可以用来扩大定律和公式的应用范围

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

$$\overline{\underline{A_1 \cdot A_2 \cdot \dots \cdot A_n}} = \overline{\underline{A_1}} + \overline{\underline{A_2}} + \dots + \overline{\underline{A_n}} \quad \overline{\underline{A_1 + A_2 + \dots + A_n}} = \overline{\underline{A_1}} \cdot \overline{\underline{A_2}} \cdot \dots \cdot \overline{\underline{A_n}}$$

■ 反演定理

➤ 将函数 Y 式中所有的...

- “•”换成“+”，“+”换成“•”；
- “0”换成“1”，“1”换成“0”；
- 原变量换成反变量，反变量换成原变量，

则所得到的表达式是 \bar{Y} 的表达式。

注意：

1. 变换时要**保持**原式中逻辑运算的**优先顺序**；
2. 不属于单个变量上的反号应**保持不变**。



■ 反演定理（续）

➤ 例：

- DeMorgan定理是反演定理的一个特例，故被称之为“反演律”

- 已知 $Y = A \cdot [\overline{B} + C\overline{D} + \overline{E}F]$, 求 \overline{Y}

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

- 已知 $Z = A + \overline{B + \overline{C + \overline{D + \overline{E}}}}$, 求 \overline{Z}

$$\overline{Z} = \overline{A} \cdot \overline{\overline{B}} \cdot \overline{C} \cdot \overline{\overline{D}} \cdot \overline{E}$$

说明：应用反演定理中的一个细节问题

注意2：不属于单个变量上的反号应保持不变

$$Z = A + B + \overline{C} + \boxed{\overline{D} + \overline{E}}$$

$$\overline{Z} = \overline{A} \cdot \overline{B} \cdot C \cdot \boxed{\overline{D} \cdot E}$$

但在局部，根据反演律，存在：

$$\overline{\overline{D} + \overline{E}} = \overline{D} \cdot \overline{E}$$

这样考虑错误！

为何局部不等!?

说明：

对等式两端根据反演定理进行操作是整体性的“原子操作”
 不允许在进行操作的同时，对局部的逻辑项进行所谓的
 “代入”、“反演律”等操作。

可根据上式验算，证明 \overline{Z} 的表达式是正确的。



反演的应用——例题：

- 已知 $Y = A(B+C) + CD$, 求 \bar{Y}

解：由反演定理， $\bar{Y} = [\bar{A} + \bar{B} \cdot \bar{C}] \cdot (\bar{C} + \bar{D})$

展开 $\bar{A} \cdot \bar{C} + \bar{B} \cdot \bar{C} + \bar{A} \cdot \bar{D} + \bar{B} \cdot \bar{C} \cdot \bar{D} = \bar{A} \cdot \bar{C} + \bar{B} \cdot \bar{C} + \bar{A} \cdot \bar{D}$

又解： $\bar{Y} = \overline{A(B+C) + CD}$ 反复使用反演律，求“与或”表达式

$$\begin{aligned}\bar{Y} &= \overline{A(B+C)} \cdot \overline{CD} = (\bar{A} + \overline{B+C}) \cdot (\bar{C} + \bar{D}) \\ &= (\bar{A} + \bar{B} \cdot \bar{C}) \cdot (\bar{C} + \bar{D}) = \bar{A} \cdot \bar{C} + \bar{A} \cdot \bar{D} + \bar{B} \cdot \bar{C} + \bar{B} \cdot \bar{C} \cdot \bar{D} \\ &= \bar{A} \cdot \bar{C} + \bar{A} \cdot \bar{D} + \bar{B} \cdot \bar{C}\end{aligned}$$



■ 对偶定理

- 若两个逻辑式相等，则它们的**对偶式**也相等。
- **对偶式的定义**

对于一个逻辑式 Y , 将其中所有:

- “.”换成“+”，“+”换成“.”
- “1”换成“0”，“0”换成“1”
- **变量保持不变**
- 原表达式中的运算优先顺序保持不变



■ 对偶定理（续）

- 如果两个逻辑表达式相等，那么它们的对偶式也相等。

$$X = A(B + C)$$

$$X^D = A + BC$$

例如： $AB + AC = A(B + C) \leftarrow \text{对偶式} \rightarrow (A + B)(A + C) = A + BC$

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

$$Y^D = AB\overline{C}$$

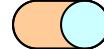
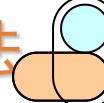
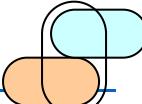
$$Z = (A + 0)(B \cdot 1)$$

$$Z^D = (A \cdot 1) + (B + 0)$$

注意：这里 X 和 X^D 是不同的逻辑函数。当等号两端都是含有逻辑变量的表达式时，对偶定理才体现出应用的意义。

布尔代数与逻辑函数

➤ 逻辑代数的常用公式

组	对偶的公式对		杜撰的助记标记/说明
(9)	$A + A \cdot B = A$	$A \cdot (A + B) = A$	吸收冗余项法 
(10)	$A + \bar{A} \cdot B = A + B$	$A \cdot (\bar{A} + B) = AB$	合并消去因子, 消元法 
(11)	$A \cdot B + \bar{A} \cdot C + B \cdot C = A \cdot B + \bar{A} \cdot C$		推广的吸收法 
(12)	$A \cdot B + \bar{A} \cdot C + B \cdot C \cdot D = A \cdot B + \bar{A} \cdot C$		推广的吸收法
(13)	利用对偶定理 写出这些公式	$\bar{A} \cdot \bar{A} \cdot B = \bar{A}$	将 $\bar{A} \cdot B$ 作DeMorgan展开后, 是另一种形式的吸收法。
(14)		$A \cdot \bar{A} \cdot B = A \cdot \bar{B}$	将 $\bar{A} \cdot B$ 作DeMorgan展开后, 是另一种形式的消元法。



提示：逻辑等式证明的方法

- 方法一、分别列出等式两边逻辑式的真值表，若真值表完全相同，则等式成立；
- 方法二、分别画出等式两边逻辑式的卡诺图（是一种邻接真值表，后续讲解），若卡诺图相同，则等式成立。

➤ 由于逻辑变量个数的增多而使用不便



提示：逻辑等式证明的方法（续）

- 方法三、若能利用逻辑代数的运算规则、公式和定理将两边转化成完全相同的形式，则等式成立；
 - 公理和法则
 - 代入定理、反演定理 和 对偶定理
 - 基本公式和常用公式
- 方法四、机器证明。

- 案例研究：逻辑代数常用公式的证明
 - 练习——通过常用公式的证明说明逻辑代数的公理、法则、定理、公式的应用方法：



逻辑代数常用公式的证明

(9) $A + A \cdot B = A$

证: $A + AB = A(1 + B) = A$ (分配律、01律)

(10) $A + \bar{A} \cdot B = A + B$

证法1: $\underbrace{A + \bar{A}B}_{(A + \bar{A})(A + B)} = A + B$

思考: 关键步骤如何得到?

(让我们记住: “或”对“与”的分配律)

证法2: 根据 对偶定理, 证明该式的对偶式, 即:

$$A \cdot (\bar{A} + B) = AB \quad (\text{根据 分配律 互补律 易证})$$



逻辑代数常用公式的证明

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

证: $= AB + \bar{A}C + BC(A + \bar{A}) = (AB + ABC) + (\bar{A}C + \bar{A}CB)$
 $= AB(1 + C) + \bar{A}C(1 + B) = AB + \bar{A}C$

(互补律、分配律、01律; 或者: 互补律、吸收法)

(12) $A \cdot B + \bar{A} \cdot C + B \cdot C \cdot D = A \cdot B + \bar{A} \cdot C$

证:

同上, 互补律、吸收法 (含: 代入定理)

或者, 反用(11), 然后用吸收法, 最后再正用(11)

2.2 逻辑函数的表示方法及其标准形式

➤ 逻辑函数的表示方法

■ 逻辑函数表达式

- 组成：逻辑变量、逻辑常量，逻辑运算符号。
- 例： $Y = AB + \bar{A}C$

■ 真值表

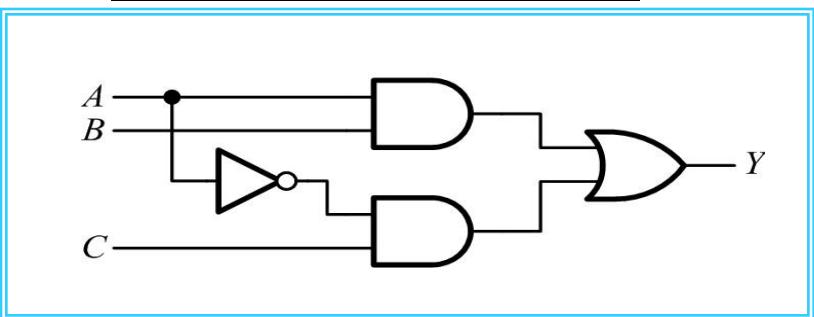
■ 卡诺图

- 一种特殊的真值表。

■ 逻辑图

- 用逻辑门符号构成的逻辑函数关系图形；
- 物理实现的原理图。

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





■ 波形图

- 将逻辑函数输入变量每一种可能出现的取值与对应的输出取值按时间顺序排列起来，就得到了表示该逻辑函数的波形图。
- 也称为时序图。
- 如：逻辑分析仪——通过实验观察波形检验逻辑功能。



➤ 表示方法之间的相互转换 ➤

■ 由逻辑表达式列出真值表

- 将输入变量取值的所有组合状态逐一代入逻辑式求出函数值，列成表，即得真值表；
- 输入变量取值的组合一般按自然二进制数递增的顺序排列。



布尔代数与逻辑函数

例：列出 $Y = A + \overline{B}C + \overline{A}\overline{B}\overline{C}$ 真值表

A	B	C	$\overline{B}C$	$\overline{A}\overline{B}\overline{C}$	Y
0	0	0	0	0	0
0	0	1	1	0	1
0	1	0	0	1	1
0	1	1	0	0	0
1	0	0	0	0	1
1	0	1	1	0	1
1	1	0	0	0	1
1	1	1	0	0	1

为了避免差错，可以将表达式中部分的项算出，再最终计算逻辑函数的值



布尔代数与逻辑函数

■ 由真值表写出逻辑表达式

- 找出使逻辑函数 Y 为1的变量取值组合；
- 每个使函数 Y 为1的变量取值组合对应一个乘积项
(即：“与项”)，其中取值为1的写入原变量，取值为0的写入反变量；
- 将这些乘积项相或，即得到 Y 的逻辑表达式。

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

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

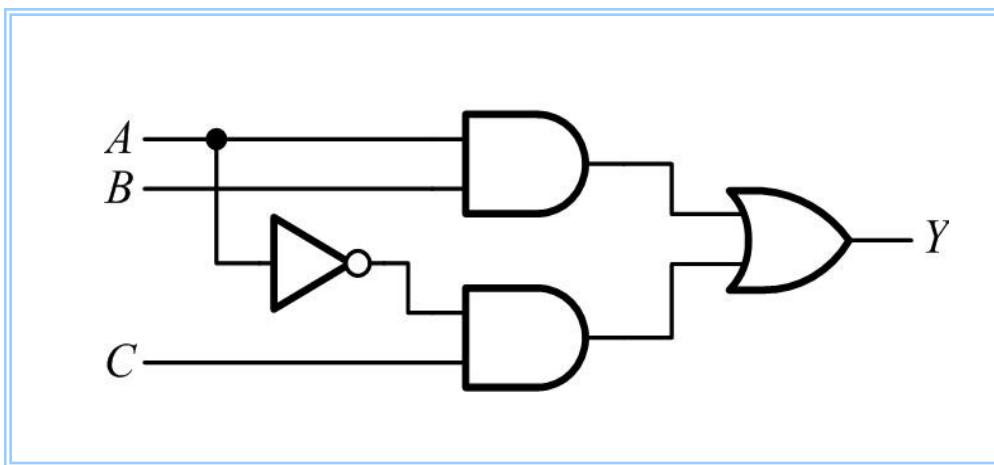
$\overline{A} \overline{B} C$

$A \overline{B} C$

■ 由逻辑式画出逻辑图

- 用图形符号代替逻辑式中的运算符号，并按运算的优先顺序将它们连接起来。

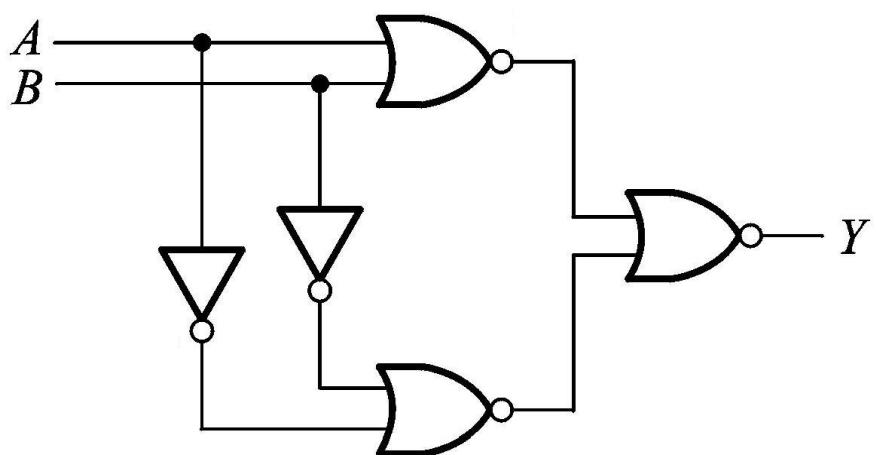
$$Y = AB + \overline{AC}$$



布尔代数与逻辑函数

■ 由逻辑图写出逻辑式

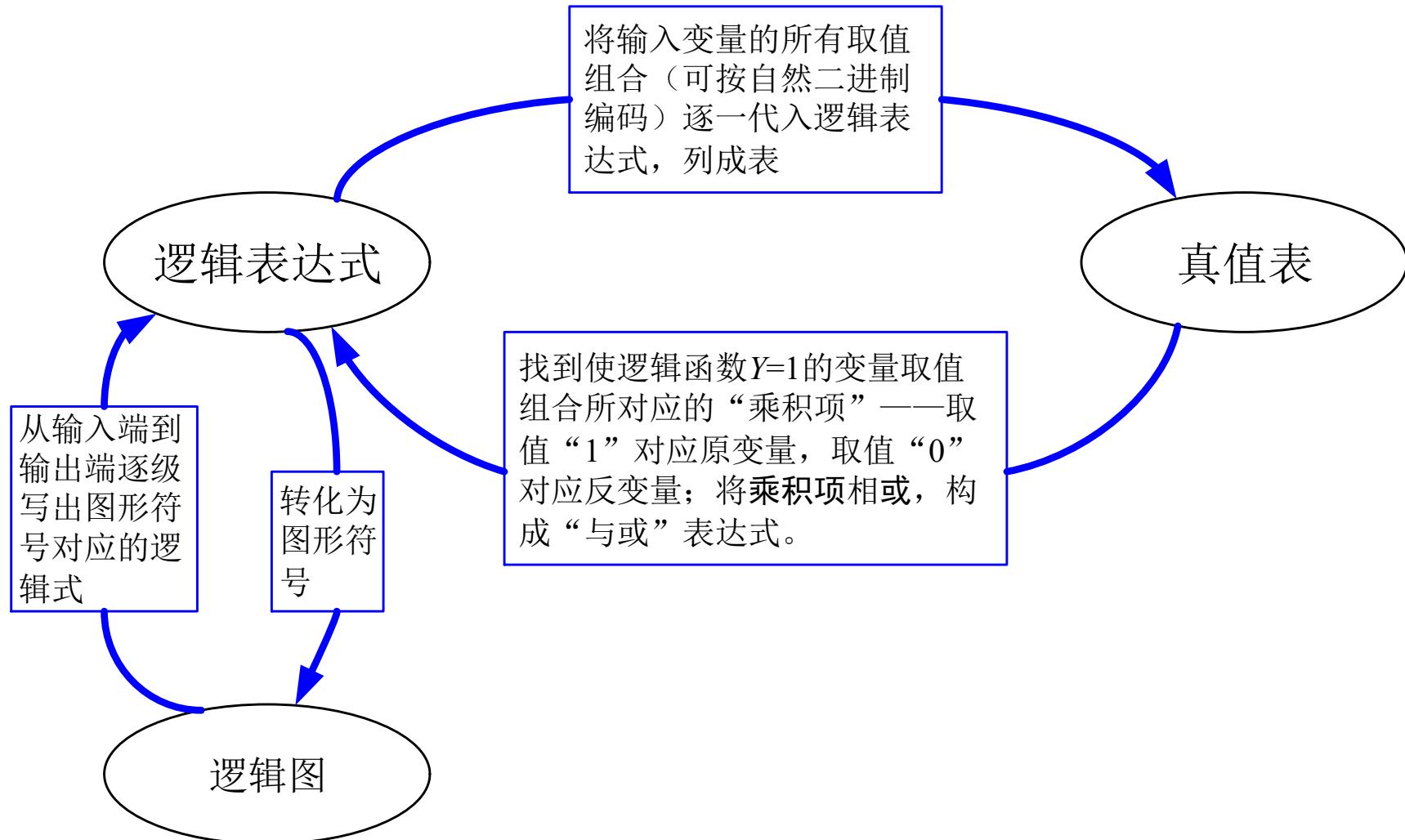
- 从输入端到输出端逐级写出图形符号对应的逻辑式



$$\begin{aligned} Y &= \overline{\overline{A+B}} + \overline{\overline{A}} + \overline{\overline{B}} \\ &= (A+B)(\overline{A} + \overline{B}) \\ &= A\overline{B} + \overline{A}B \\ &= A \oplus B \end{aligned}$$

布尔代数与逻辑函数

小结——逻辑函数表示方法之间的转换





➤ 逻辑函数的表示方法

- 标准“与或”表达式（最小项之和）
- 标准“或与”表达式（最大项之积）

■ 函数的最小项及其性质

◆ 最小项

- 在一个有 n 个变量的逻辑函数中，包含全部 n 个变量的乘积项称为最小项，其中每个变量必须而且只能以原变量或反变量的形式出现一次
- 最小项有时也称为全积项或者标准乘积项



三变量最小项及其编号

最小项	使最小项为1的变量取值			十进制	编号
	A	B	C		
$\overline{A} \overline{B} \overline{C}$	0	0	0	0	m_0
$\overline{A} \overline{B} C$	0	0	1	1	m_1
$\overline{A} B \overline{C}$	0	1	0	2	m_2
$\overline{A} BC$	0	1	1	3	m_3
$A \overline{B} \overline{C}$	1	0	0	4	m_4
$A \overline{B} C$	1	0	1	5	m_5
$AB \overline{C}$	1	1	0	6	m_6
ABC	1	1	1	7	m_7

◆ 最小项的性质

- 每一个最小项与变量的一组取值相对应，只有该组取值才使其为1

❖ 例如： $\overline{A} \overline{B} \overline{C}$ $\Leftrightarrow 0 \ 1 \ 0$

- 全体最小项之和恒为1

❖ 即：
$$\sum_{i=0}^{2^n-1} m_i \equiv 1$$

- 任意两个不同的最小项的乘积恒为0

❖ 例如： $(\overline{A} \overline{B} \overline{C})(\overline{A} \overline{B} C) \equiv 0$

■ 标准与或表达式

- 每个与项都是最小项的“与或”表达式，称为**标准与或表达式**，也称为**最小项之和表达式**

■ 从真值表求标准与或表达式

- 1) 找出使逻辑函数Y为1的变量取值组合
- 2) 写出使函数Y为1的变量取值组合相对应的最小项
- 3) 将这些最小项相“或”，即得到**标准与或表达式**



布尔代数与逻辑函数

例：

ABC	Y
000	0
001	0
010	0
011	1
100	0
101	1
110	1
111	1

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

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

$$Y = \sum m(3,5,6,7)$$



布尔代数与逻辑函数

■ 从一般与或表达式求标准与或表达式

- 方法：利用基本公式 $A + \bar{A} = 1$ （互补律）补全与项中的变量。
- 例如：

$$\begin{aligned}Y &= AB + BC + AC \\&= AB(C + \bar{C}) + BC(A + \bar{A}) + AC(B + \bar{B}) \\&= ABC + \bar{A}BC + A\bar{B}C + ABC \\&= m_3 + m_5 + m_6 + m_7 = \sum m(3,5,6,7)\end{aligned}$$

对于任何一个逻辑函数，它的真值表是唯一的，因而它的标准与或表达式（不考虑顺序）也是唯一的



■ 函数的最大项及其性质

◆ 最大项

- 在一个有 n 个变量的逻辑函数中，包含 **全部 n 个变量的和项**（确切地说，是“或项”）称为最大项，其中每个变量必须而且只能以原变量或反变量的形式出现一次
- 最大项有时也称为**全和项**或者**标准和项**

布尔代数与逻辑函数

三变量最大项及其编号

最大项	使最大项为0的 变量取值			十进制	编号
	A	B	C		
$A + B + C$	0	0	0	0	M_0
$A + B + \bar{C}$	0	0	1	1	M_1
$A + \bar{B} + C$	0	1	0	2	M_2
$A + \bar{B} + \bar{C}$	0	1	1	3	M_3
$\bar{A} + B + C$	1	0	0	4	M_4
$\bar{A} + B + \bar{C}$	1	0	1	5	M_5
$\bar{A} + \bar{B} + C$	1	1	0	6	M_6
$\bar{A} + \bar{B} + \bar{C}$	1	1	1	7	M_7



布尔代数与逻辑函数

➤ 最大项的性质

- 每一个最大项与变量的一组取值对应，即只有这一组取值才使该最大项为0。

例如：

$$\overline{A} + B + \overline{C} \Leftrightarrow 101$$

- 全体最大项之积恒为0。

$$\prod_{i=0}^{2^n-1} M_i \equiv 0$$

- 任意两个不同的最大项之和恒为1。

$$M_i + M_j \equiv 1 \quad \forall i, j; i \neq j$$

例如：

$$(\overline{A} + B + \overline{C}) + (A + \overline{B} + \overline{C}) = 1$$

- 最大项和最小项之间的关系： $M_i = \overline{m}_i$

例如：

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





■ 标准或与表达式

- 每个或项都是最大项的或与表达式称为标准或与表达式，也称为最大项之积表达式

■ 从函数真值表求标准或与表达式

- 1) 在真值表中找出使逻辑函数 Y 为0的行；
- 2) 对于 $Y=0$ 的行，写出对应的最大项；
- 3) 将所得到的最大项相“与”；
- 4) 由最大项“原”、“反”变量与“0”、“1”取值对应关系，确定最大项编号，可写成 $\prod M(\dots)$ 形式。

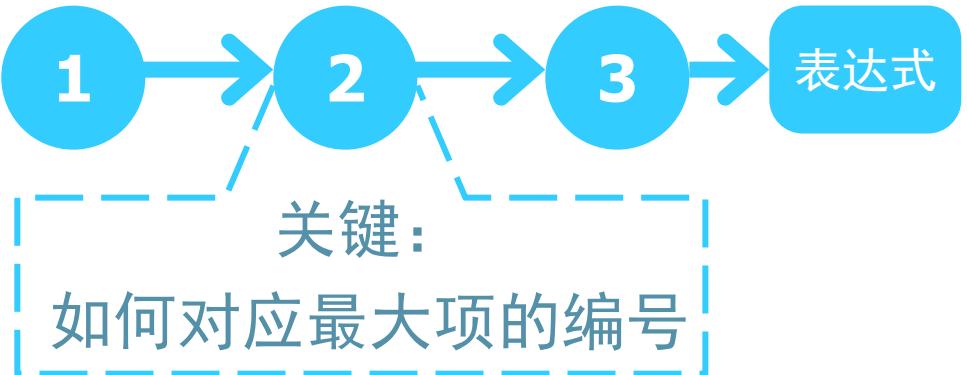
例题和说明



布尔代数与逻辑函数

例：

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



方法一、由最大项的定义，根据最大项变量取值与最大项编号的对应关系

$$Y = \prod (0,1,2,4) \quad Y = M_0 \cdot M_1 \cdot M_2 \cdot M_4$$

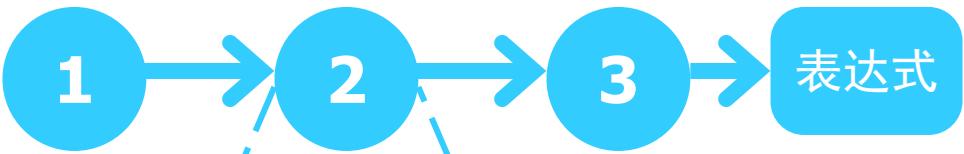
注意：

最大项编号 / 变量取值 的对应关系。

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

布尔代数与逻辑函数

	A	B	C	Y
$\overline{m_0}$	0	0	0	0
$\overline{m_1}$	0	0	1	0
$\overline{m_2}$	0	1	0	0
$\overline{m_3}$	0	1	1	1
$\overline{m_4}$	1	0	0	0
$\overline{m_5}$	1	0	1	1
$\overline{m_6}$	1	1	0	1
$\overline{m_7}$	1	1	1	1



方法二、注意到...

在以 A, B, C 原变量列出的真值表中， $Y=0$ 的 $\sum \overline{m_i}$ ；反演展开后利用 $M_i = \overline{m_i}$ 的关系，对应得到最大项 M_i 的编号。

$$Y = \prod (0, 1, 2, 4) \quad Y = M_0 \cdot M_1 \cdot M_2 \cdot M_4$$

这样，也可以先确定所含最大项的编号，再根据最大项编号和变量取值的对应关系，写出以逻辑变量表达的最大项之积表达式→

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

布尔代数与逻辑函数

■ 标准与或表达式 和 标准或与表达式

➤ 如果函数的标准与或表达式为：

$$Y = \sum_i m_i$$

➤ 函数的标准或与表达式则为：

$$Y = \prod_{k \neq i} M_k$$

例如： $Y = \sum m(3,5,6,7)$ $Y = \prod M(0,1,2,4)$

ABC	Y
000	0
001	0
010	0
011	1
100	0
101	1
110	1
111	1



布尔代数与逻辑函数

■ 推导:

$$Y = \sum_i m_i$$

由最小项性质: $1 \equiv \sum m_i$

则:

$$1 = Y + \overline{Y} = \sum_i m_i + \sum_{k \neq i} m_k$$

DeMorgan定理
(反演律) : $Y = \overline{\sum_{k \neq i} m_k} = \prod_{k \neq i} \overline{m_k} = \prod_{k \neq i} M_k$

$$M_k = \overline{m_k} \quad m_k = \overline{M_k}$$

可以认为是最小/最大项的一个性质

➤ 所以, 可以从与或表达式求或与表达式





布尔代数与逻辑函数

■ 作业:

第六版

- 2.1-(2,6,7); 2.13-(2,5,8,9)
- 2.18-(c,d); 2.10-(2,4);
2.11-(4,5)
- 2.16-(6,7); 2.16-(8);
2.17-(2,5); 2.2-(2,3)
2.20-(1,4)

第五版

- 2.1-(2,6,7); 2.15-(2,5,8,9)
- 2.20-(c,d); 2.10-(2,4);
2.11-(4,5)
- 2.18-(6,7); 2.18-(8); 2.19-(2,5);
2.2-(2,3)
2.12-(1,3); 2.13-(2,3); 2.22-(1,4)

如果采用第四版教科书

- 1.5-(1,2,3); 1.7-(2,6,7); 1.8-(2,5,8,9)
- 1.9-(c,d); 1.10-(2,5);
1.11-(2,4); 1.12-(4,5)
- 1.13-(6,7,9); 1.14-(2,5); 1.15-(2,3);
1.16-(1,3); 1.17-(2,3); 1.20(1,4)



2.3 逻辑函数的化简

- 逻辑函数的最简形式
- 公式法化简逻辑函数
- 卡诺图法化简逻辑函数
 - 卡诺图
 - 卡诺图化简法（化简为最简**与或**表达式）
 - 用卡诺图化简法 求 最简**或与**表达式
 - 具有无关项的逻辑函数的化简
- 逻辑函数形式的转换



➤ 逻辑函数的最简形式

- 同一个逻辑函数可以写成各种不同形式的表达式
 - 表达式越简单，所表示的逻辑关系越明显
 - 表达式越简单，一般说来，就可以用最少的电子器件来实现
- 注意：确切地说，不同形式的逻辑器件对应着不同形式的最简逻辑表达式
- 需要通过化简的方法找出逻辑函数的最简形式



◆ 最简与或表达式

- 最常用的是 **与或表达式**，由它容易推导出其它表达形式

◆ 判别 **与或表达式** 是否为最简的条件：

- 乘积项（与项）最少
- 每个乘积项中因子（逻辑变量）最少



➤ 公式法化简逻辑函数

- 根据逻辑代数的公理、定律、定理、公式等，消去逻辑函数式中多余的乘积项和多余的因子，进行化简。
- 公式法化简**没有固定的步骤**，而要根据具体问题具体应用不同的方法，这些方法大致包括：
- 并项法、吸收法、消因子法、消项法、配项法等。
- 化简的方法不是唯一的。



■ 并项法

- 利用互补律: $\bar{A} + A = 1$, 将两项合并为一项, 合并时消去一个逻辑变量 (一个原变量、一个反变量)

- 例:
$$\begin{aligned} & A\bar{B}C + ABC\bar{C} + A \cdot \bar{B} \cdot \bar{C} + ABC \\ &= A(\bar{B}C + B\bar{C}) + A(\bar{B} \cdot \bar{C} + BC) \\ &= A(B \oplus C) + A(\overline{B \oplus C}) = A \end{aligned}$$

实际上这道例题, 对于 B, C , 已经是 $\sum_{i=0}^3 m_i^{(B,C)} \equiv 1$

布尔代数与逻辑函数

■ 吸收法

- 利用公式: $A + AB = A$, 吸收掉冗余的乘积项。
- 例:

$$\begin{aligned} & A + \overline{\overline{A} \cdot \overline{BC}} (\overline{A} + \overline{\overline{BC}} + D) + BC \\ &= \underline{A} + BC + (\underline{A + BC}) (\overline{A} + \overline{\overline{BC}} + D) \\ &= A + BC \end{aligned}$$

↓
吸收项



■ 消因子法

- 利用公式: $A + \overline{A}B = A + B$, 消去多余的因子
- 例:

$$AB + \overline{A}C + \overline{B}C$$



■ 消项法

◆ 利用常用公式：

- $AB + \bar{A}C + BC = AB + \bar{A}C$
- $AB + \bar{A}C + BCD = AB + \bar{A}C$

消去多余的乘积项

● 例：

$$Y_1 = AC + A\bar{B} + \overline{B+C} = AC + A\bar{B} + \overline{B} \cdot \overline{C} = AC + \overline{B} \cdot \overline{C}$$

$$\begin{aligned} Y_2 &= A\bar{B}\bar{C}\bar{D} + \overline{\bar{A}\bar{B}}E + \overline{AC}\bar{D}E \\ &= (\bar{A}\bar{B})\bar{C}\bar{D} + (\overline{\bar{A}\bar{B}})E + (\bar{C}\bar{D})(E)\bar{A} = A\bar{B}\bar{C}\bar{D} + \overline{\bar{A}\bar{B}}E \end{aligned}$$

■ 配项法

- 根据重叠律 $A+A=A$, 在式中重复某项, 再化简;
- 根据互补率 $A+\bar{A}=1$, 式中某项乘以 $A+\bar{A}$, 再化简。

$$\begin{aligned} Y &= \overline{ABC} + \overline{ABC} + ABC \\ &= \overline{ABC} + \underline{\overline{ABC}} + \underline{\overline{ABC}} + ABC \\ &= \overline{AB}(\overline{C} + C) + (\overline{A} + A)BC \\ &= \overline{AB} + BC \end{aligned}$$

说明：本例只是用来演示，实际上先对后两项并项，然后再消因子，更加直观一些。



➤ 卡诺图法化简逻辑函数

提纲

- 卡诺图
- 卡诺图化简法
- 具有无关项的逻辑函数的化简



■ 卡诺图 (Karnaugh Map)

- ◆ 定义和历史
- ◆ 卡诺图的构成与特点
- ◆ 根据逻辑函数填写卡诺图
- ◆ 由卡诺图得到标准与或表达式



■ 卡诺图 (Karnaugh Map)

◆ 定义和历史

- 卡诺图是由美国工程师卡诺 (Karnaugh, M) 首先提出的一种用来描述逻辑函数的特殊**方格图***。
- 在这个方格图中，
 - 每一个小方格代表逻辑函数的一个**最小项***
 - 而且几何位置“**相邻**”的小方格具有**逻辑相邻性**，
 - ❖ 即：两个相邻的小方格所代表的最小项只有一个变量取值不同。



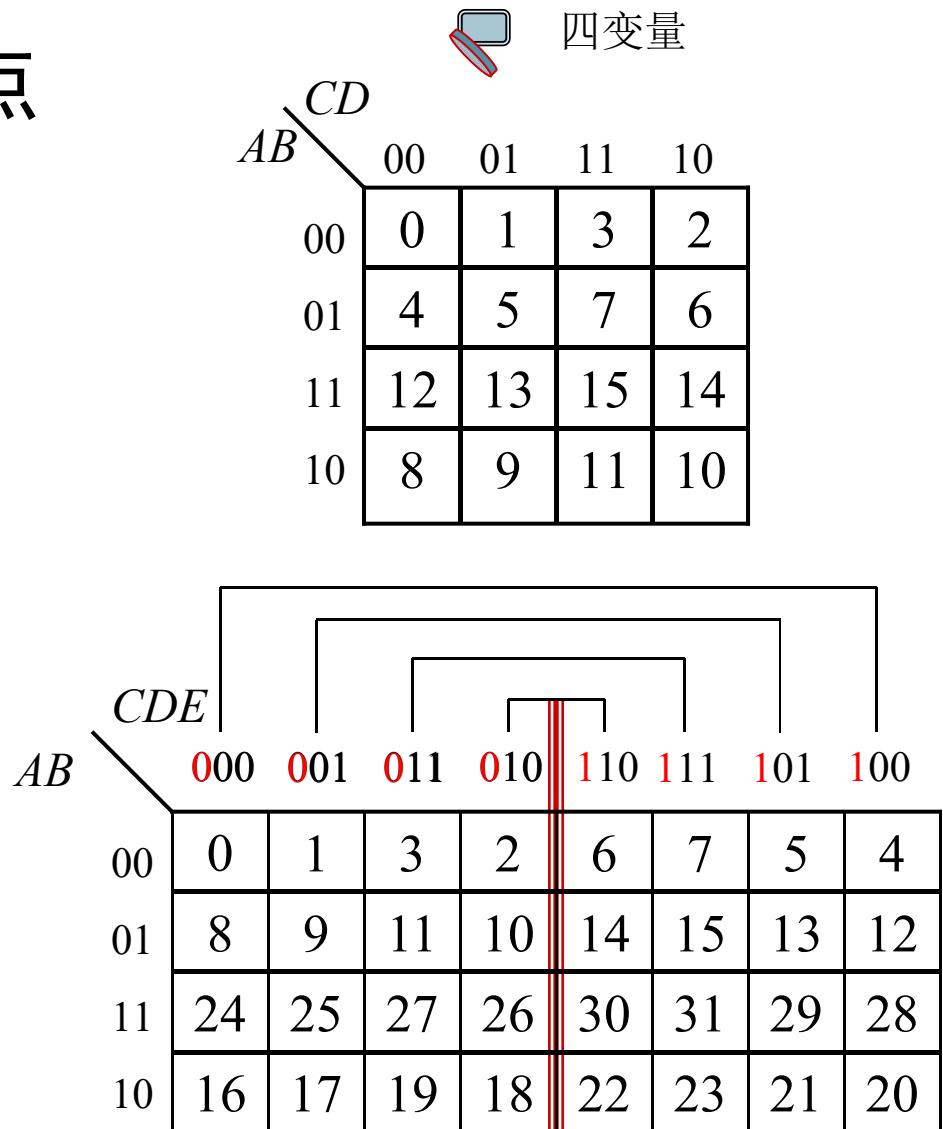
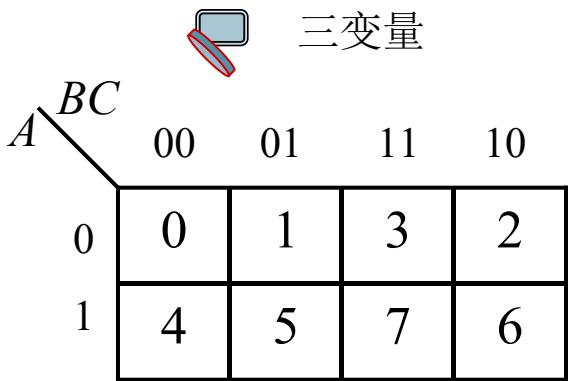
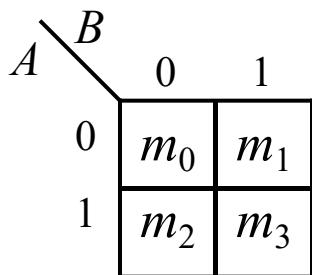
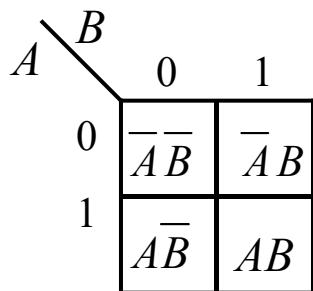
➤ 确切地说：

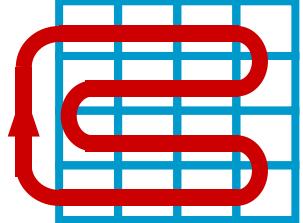
- * 卡诺图是由美国工程师维奇 (Veitch) 和卡诺 (Karnaugh) 分别从不同角度提出的；
 - 1953年, Karnaugh, Manrice (美国贝尔实验室)
 - “The Map Method for Synthesis of Combinational Logic”, Trans. of Amer. Inst. of Electrical Engineers, 1970.1.1.
- * 卡诺图也是一种特殊的真值表——邻接真值表；
 - “**几何相邻**的小方格具有**逻辑相邻性**”
- * 也存在每一小格代表最大项的卡诺图。

布尔代数与逻辑函数

◆ 卡诺图的构成与特点

➤ 卡诺图的构成





➤ 卡诺图的特点

- 卡诺图中的小方格数等于**最小项**总数，若逻辑函数的变量数为 n ，则小方格数为 2^n 个
 - 最小项的卡诺图纵横两列标注的“0”、“1”取值组合表示使方格对应内最小项为1时的变量取值；
 - 变量取值组合“0”、“1”的自然二进制数值就是对应最小项的编号。
- 任何一个 n 变量逻辑函数可以用 **n 变量最小项卡诺图**表示
 - 逻辑函数等于在卡诺图中填入“1”的小格（“**1格**”）所对应的最小项之和。**（但由于K-map是二维图，最多 $n=5$ ）**
- 卡诺图是“邻接**真值表**”，
 - 在卡诺图中，变量的取值按格雷循环码排列；因此，
 - 几何位置“相邻”的最小项具有逻辑相邻性。

◆ 卡诺图的特点（续）

➤ 卡诺图几何位置“相邻”的方格之间的逻辑关系：

- 逻辑相邻性；
- 闭合（上下或左右）

		CDE								
		000	001	011	010		110	111	101	100
AB	00	0	1	3	2		6	7	5	4
		8	9	11	10		14	15	13	12
11	24	25	27	26		30	31	29	28	
10	16	17	19	18		22	23	21	20	

- 对于五个变量，仅仅用二维空间的相邻性已经不够，
——（轴）对称位置的逻辑相邻性，

➤ 变量应按最小项编码的顺序分组

- $1 \times 1, 1 \times 2, 2 \times 2, 2 \times 3$
- 更高维不适用！



◆ 根据逻辑函数填写卡诺图

- 得到 **标准与或表达式**

- 若已知逻辑函数的表达式，可首先把函数写成**最小项之和**的形式；

- 填写 **卡诺图**

- 在卡诺图上与这些最小项对应的位置上填入1，在其余位置上填入0，这样就可得到表示该逻辑函数的卡诺图。

例：

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

布尔代数与逻辑函数

例：

$$\begin{aligned}
 Y &= \overline{(A \cdot B + \overline{A} \cdot \overline{B} + \overline{C}) A \cdot \overline{B}} \\
 &= (\overline{A} + \overline{B})(A + B)C + AB \\
 &= \overline{A}\overline{B}C + A\overline{B}C + AB \\
 &= \overline{A}\overline{B}C + A\overline{B}C + A\overline{B}\overline{C} + ABC \\
 Y(A,B,C) &= m3 + m5 + m6 + m7
 \end{aligned}$$



	A	B	C	
	00	01	11	10
0	0	1	3	2
1	4	5	7	6

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

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

布尔代数与逻辑函数

例：

$$\begin{aligned}
 Y &= \overline{(AB + \overline{AB} + \overline{C})\overline{AB}} \\
 &= (\overline{A} + \overline{B})(A + B)C + AB \\
 &= \overline{ABC} + A\overline{BC} + AB
 \end{aligned}$$

与或表达式



熟练后，亦可根据 **与或表达式** 直接填入（提倡）



		BC	00	01	11	10
		A	0			
0	0				1	
	1			1	1	1

布尔代数与逻辑函数

根据 与或表达式 直接填入，又例：

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



1

try
it...

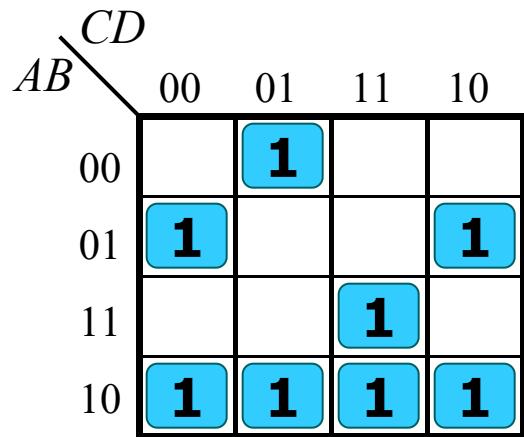
		CD		
		00	01	11
AB	00			
	01			
11	00			
	10			

布尔代数与逻辑函数

◆ 根据卡诺图写出逻辑表达式：

- 既可写出 **标准“与或”表达式**（简易）
- 也可写出 **标准“或与”表达式**（也较方便，见后...）

练习：由卡诺图写出标准与或表达式





■ 卡诺图化简法（化简为最简与或表达式）

◆ 用卡诺图化简逻辑函数的依据

- 由于卡诺图上几何位置的相邻性与逻辑上的相邻性是一致的，因而从卡诺图上能直观地找出那些具有相邻性的最小项，并将其合并化简
- 几何相邻的两个方格（包括“闭合”与“轴对称”）所代表的最小项只有一个变量不同
- 根据互补律，当方格为1（简称“1格”），且两个“1格”相邻时，对应的最小项就可以加以合并，消去一对原变量与反变量，合并后只剩公共因子

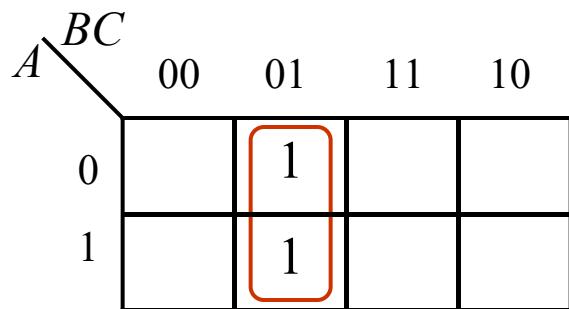
- 例如： $ABC + \bar{A}\bar{B}C = AC(B + \bar{B}) = AC$

布尔代数与逻辑函数

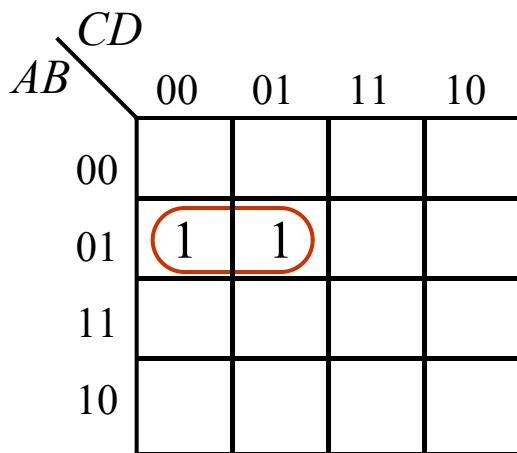
■ 如何“直观地”找到可以合并的最小项呢？

◆ 最小项卡诺图逻辑化简规则

➤ 规则1：卡诺图中两个相邻 “1格” 的最小项可以合并成一个与项，并消去一个变量



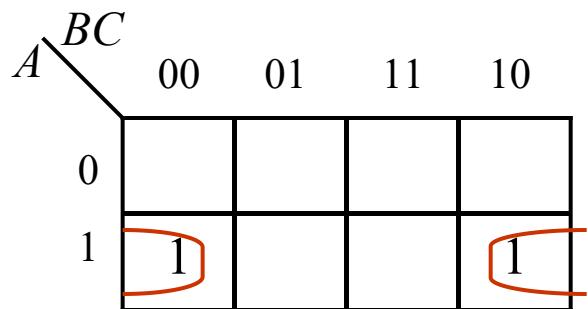
$$Y = \overline{A} \cdot \overline{B} \cdot C + A \cdot \overline{B} \cdot C = \overline{B}C$$



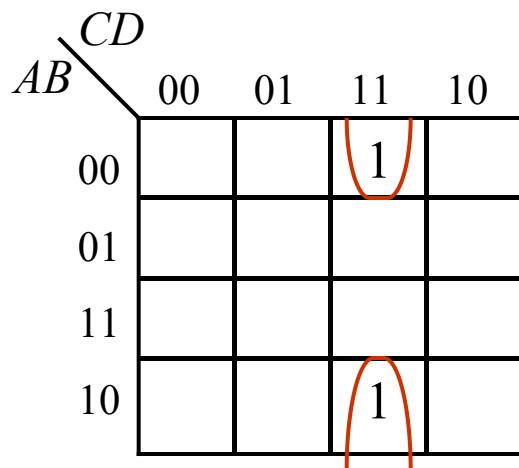
$$\overline{A}\overline{B}\overline{C}$$

布尔代数与逻辑函数

➤ 规则1：



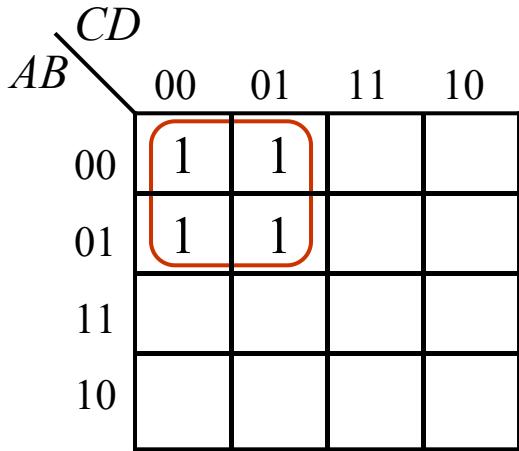
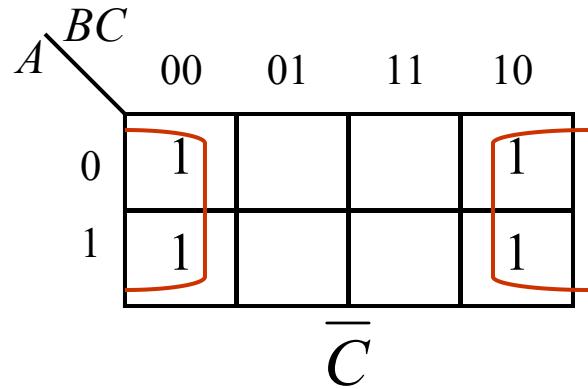
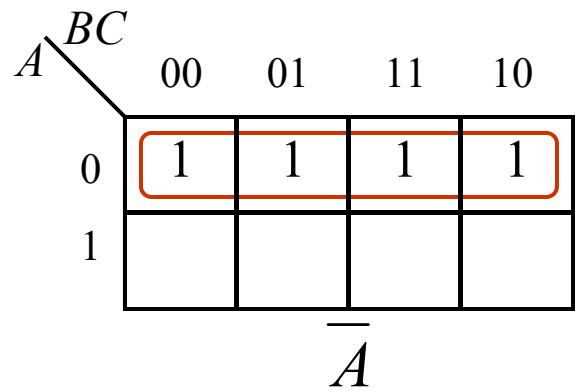
$$Y = A \cdot \overline{B} \cdot \overline{C} + A \cdot B \cdot \overline{C} = A \cdot \overline{C}$$



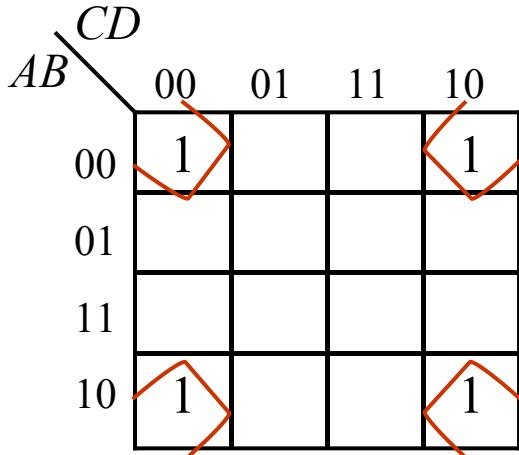
$$\overline{B}CD$$

布尔代数与逻辑函数

➤ 规则2：卡诺图中四个相邻“1格”的最小项可以合并成一个与项，并消去两个变量。



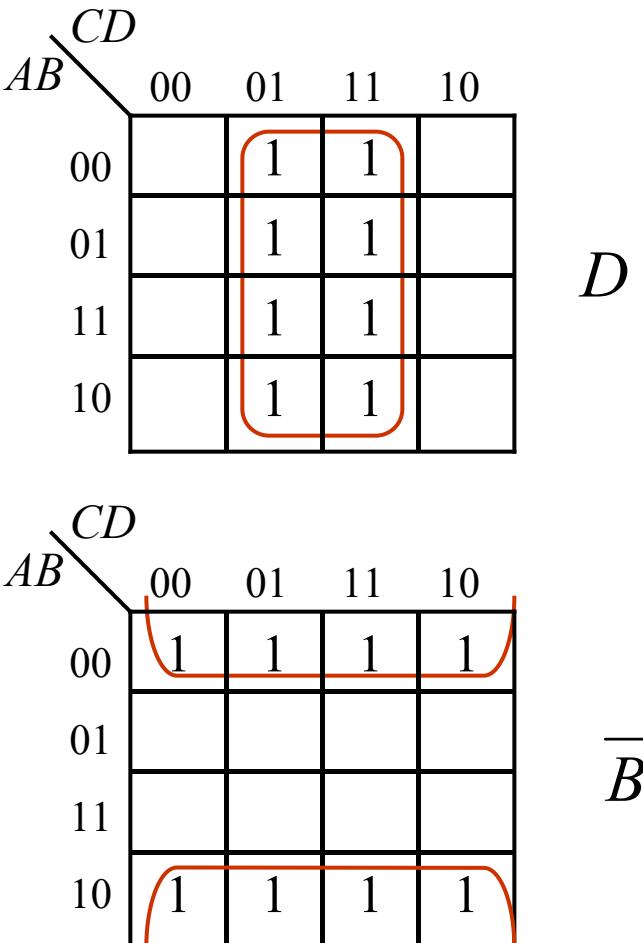
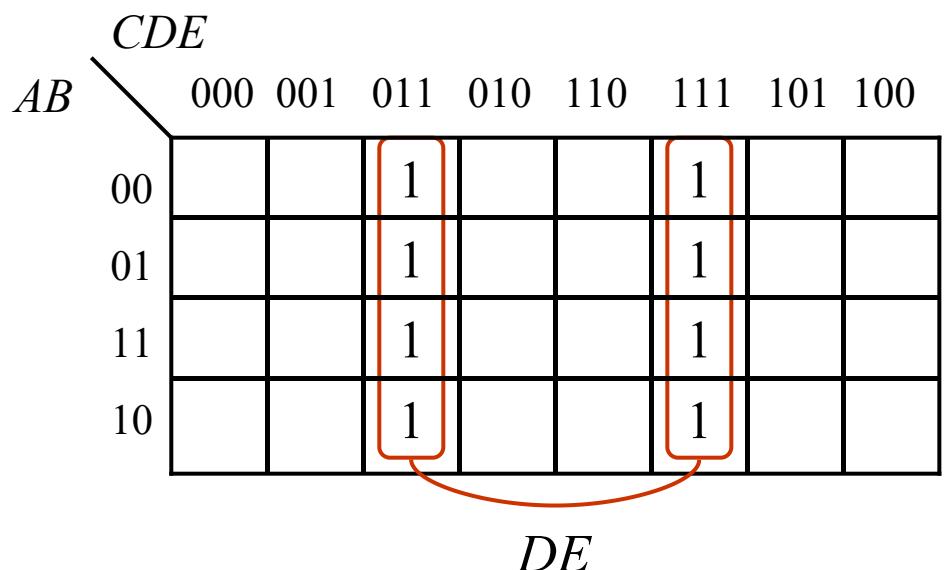
$$\overline{A} \cdot \overline{C}$$



$$\overline{B} \cdot \overline{D}$$

布尔代数与逻辑函数

➤ 规则3：卡诺图中八个相邻“1格”的最小项可以合并成一个与项，并消去三个变量





■ 用最小项卡诺图化简法求 最简与或表达式

步骤：

- (1) 建立逻辑函数的卡诺图；
- (2) 合并最小项——反复利用互补律化简；
- (3) 写出 **最简与或表达式**。

■ 问题在于：如何选择可合并的**最小项**，以达到最简？

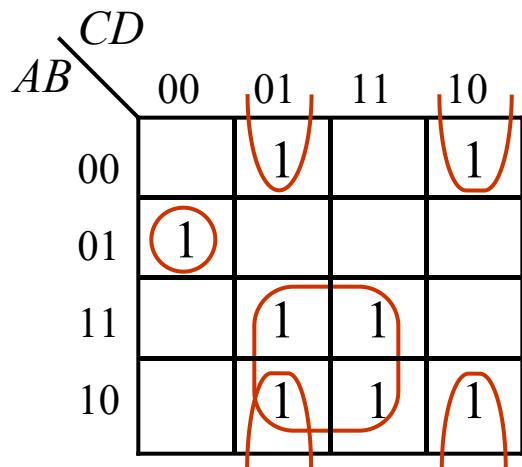
● 技巧：**选择卡诺圈的技巧！**

● 理论：找到**实质蕴涵项**

布尔代数与逻辑函数

练习

$$Y(A, B, C, D) = \sum m(1, 2, 4, 9, 10, 11, 13, 15)$$



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



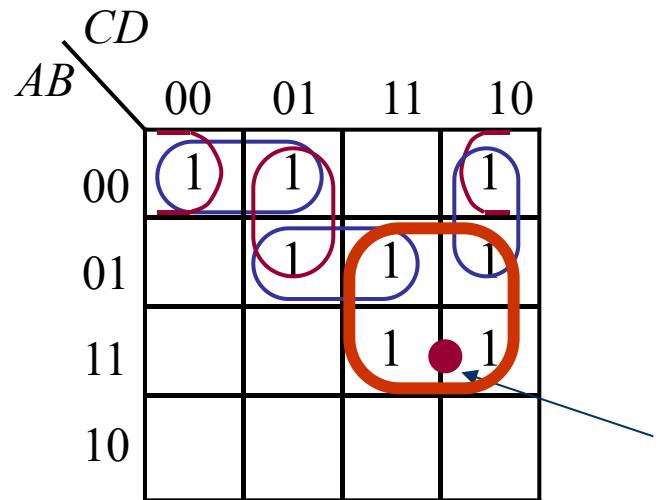
■ 画卡诺圈和选择卡诺圈的技巧

五个原则：

- 1) “1格”不能漏圈； 卡诺圈覆盖所有“1格”
- 2) “1格”允许被一个以上的圈所包围；
- 3) 圈的个数尽可能少； 最小覆盖
- 4) 圈的面积尽可能大； 本原蕴含项
- 5) 每个圈至少应包含一个新的“1格”。实质蕴含项
 - 这个“新的‘1格’”——学名“实质最小项”

布尔代数与逻辑函数

练习



“新的‘1’
格”

圈的个数尽可能少

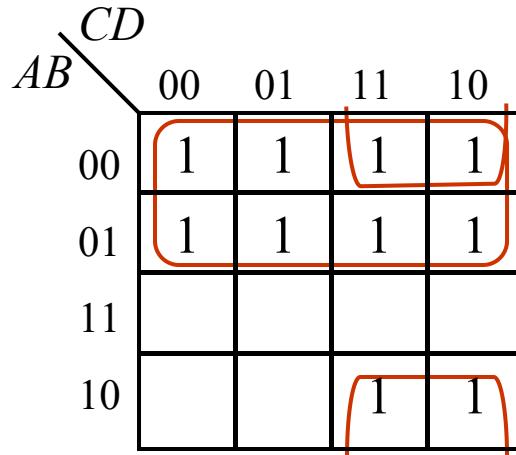
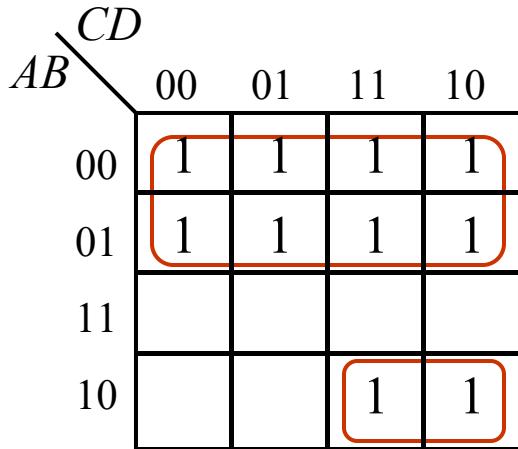
	CD \ AB	00	01	11	10
00	1	1		1	
01		1	1	1	
11			1	1	
10					

	CD \ AB	00	01	11	10
00	1	1		1	
01		1	1	1	
11			1	1	
10					

OK

布尔代数与逻辑函数

练习



圈的面积尽可能大

布尔代数与逻辑函数

练习

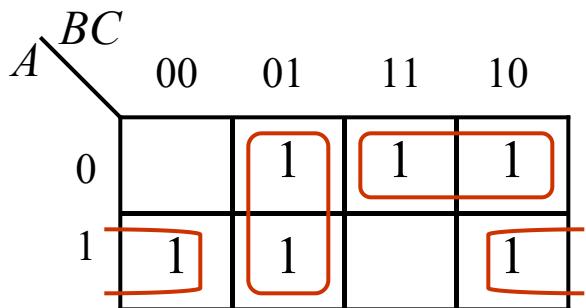
	CD	00	01	11	10
AB	00	1			
00	01	1	1	1	
01	11	1	1	1	
11	10			1	

	CD	00	01	11	10
AB	00	1			
00	01	1	1	1	1
01	11	1	1	1	
11	10			1	

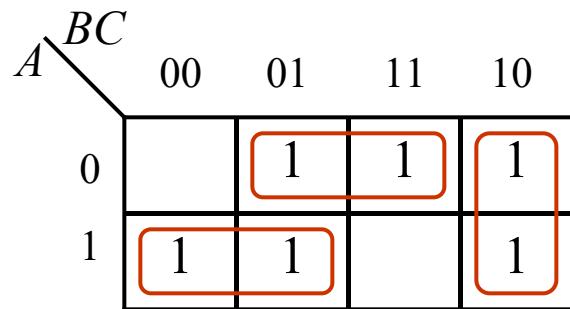
每个圈至少应包含一个新的“1格”

布尔代数与逻辑函数

■ 注意：卡诺图化简得到的最简式一定是唯一的吗？



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



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



■ 用卡诺图化简法求 **最简或与表达式**

- 方法： 合并反函数的最小项
- 注意： 反函数可以用真值表或者卡诺图中 $Y=0$ 对应的的最小项之和来表示。
 - 1) 画出逻辑函数 Y 的卡诺图；
 - 2) **合并0方格**（俗称“0格”）求得**反函数的 最简与或表达式**；
 - 3) 对反函数的最简“与或”式进行**反演变换**（DeMorgan公式），得函数的最简“或与”式。

布尔代数与逻辑函数

练习：求 最简或与表达式

$$Y(A, B, C, D) = \sum m(0, 1, 2, 5, 8, 9, 10)$$

$$\bar{Y} = AB + CD + B\bar{D}$$

$$Y = \overline{AB + CD + B\bar{D}}$$

$$= \overline{A \cdot B} \cdot \overline{C \cdot D} \cdot \overline{B \cdot \overline{D}}$$

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

AB	CD	00	01	11	10
00		1	1	0	1
01		0	1	0	0
11		0	0	0	0
10		1	1	0	1



■ 具有无关项的逻辑函数的化简

◆ 无关项

- 约束项：输入逻辑变量的某些取值组合禁止出现
- 任意项：一些取值组合出现时，输出逻辑值可以是任意的
- 这些取值组合对应的最小项称为**约束项**或**任意项**，统称为**无关项**
- 在卡诺图的方格中，常使用符号“×”（或“ ϕ ”）表示

◆ 无关项在化简逻辑函数中的应用

- 合理利用无关项，一般可得到更加简单的化简结果
- 在卡诺图中，无关项“×”可以被作为1，也可以被作为0
- 目的：参与化简操作的无关项应该与函数式中尽可能多的最小项具有逻辑相邻性

例：带有无关项的逻辑函数和卡诺图

$$Y(A, B, C, D) = \sum m(1, 3, 7, 11, 15) + \sum d(0, 2, 5)$$

$$Y = \overline{A}D + CD$$

合并包含无关项的最小项

- 将无关项“×”作为1——认为函数式包含此无关项；还是
- 将无关项“×”作为0——认为函数式不包含此无关项？

原则：应该使...

➤相邻最小项矩形组合（“卡诺圈”）最大，

➤并且，组合（“卡诺圈”）数目最少。

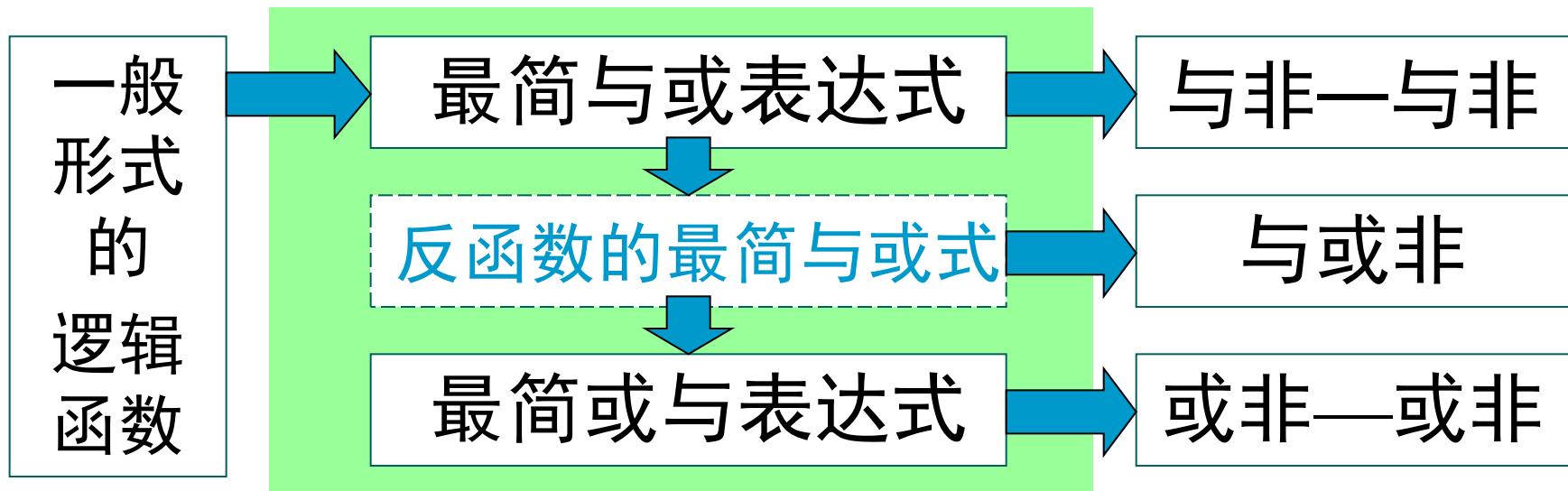
无关项化简不唯一，并且有可能不同化简结果不相等！

		CD	00	01	11	10
AB		00	x	1	1	x
		01	x	1		
		11			1	
		10				1

布尔代数与逻辑函数

► 逻辑函数形式的转换

- 最简与或表达式可用 **与门** 和 **或门** 来实现，但在数字电路系统中，广泛使用的有 **与非门**、**与或非门** 及与 **或非门** 等。



■ 与或 → 与非—与非

- 只要将 与或表达式 两次求反，再使用一次 DeMorgan公式，就可以得到“与非—与非” 表达式

$$\begin{aligned} Y &= AC + \overline{AB} \\ &= \overline{\overline{AC} + \overline{\overline{AB}}} \\ &= \overline{\overline{A} \cdot \overline{C} \cdot \overline{\overline{A}} \cdot \overline{B}} \end{aligned}$$

用两级 与非门 即可实现

布尔代数与逻辑函数

■ 与或 → 与或非

- 先求其 反函数 的 最简与或表达式，然后再 求反 即可得到 “与或非” 表达式。

$$Y = AC + \overline{A}B$$

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

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

✓ 用一个 “与或非” 门即可实现

		BC	00	01	11	10
		A	0	1	1	1
0	0		1	1		
	1	1	1	1	1	

布尔代数与逻辑函数

■ 与或 \rightarrow (或与) \rightarrow 或非—或非

- 1) 作出原函数卡诺图，用合并“0格”的方法先求出其 反函数 的 最简与或表达式
- 2) 对所得“与或”表达式 求反 得到原函数的 最简或与表达式
- 3) 两次求反，并利用DeMorgan公式，便可得到原函数的——“或非—或非” 表达式

$$Y = AC + \overline{A}B$$

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

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

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

DeMorgan

		BC	00	01	11	10
		A	0	1	1	1
0	1	0	1	1		
		1	1	1	1	1

$$\overline{A + B} + \overline{\overline{A} + C}$$

DeMorgan

布尔代数与逻辑函数

■ 与或 \rightarrow 或非一或非

- 1) 作出原函数卡诺图, 用合并“0”格的方法先求出其 反函数 的 最简与或表达式;
- 2) 反函数表达式两端分别求反;
- 3) 再对各个乘积项应用DeMorgan公式。 (内层“反用” DeMorgan)

$$Y = AC + \overline{A}B$$

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

$$\begin{aligned} Y &= \overline{\overline{A} \cdot \overline{B}} + \overline{\overline{A} \cdot \overline{C}} \\ &= \overline{\overline{A} + \overline{B}} + \overline{\overline{A} + \overline{C}} \\ &= \overline{A} + B + \overline{A} + C \end{aligned}$$

		BC	00	01	11	10
		A	0	1	1	1
0	0		1	1		
	1		1	1	1	1

练习：用最简的“或非”逻辑实现逻辑函数

■ 例： $Y = \overline{A} \cdot \overline{D} + \overline{A} \cdot \overline{B} \cdot C + A \cdot \overline{C} \cdot D$

(外部) 约束条件： $A \cdot C \equiv 0$

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

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

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

$AB \backslash CD$	00	01	11	10
00	1	1	1	1
01	1	1		1
11		1	X	X
10		1	X	X

第五章 组合逻辑



2020

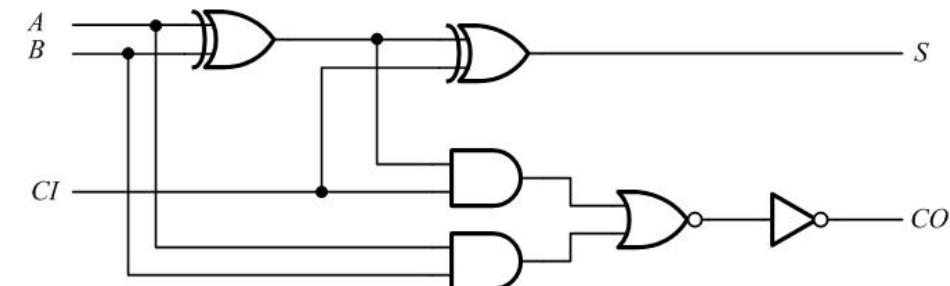
数字电路与系统

第四章、组合逻辑电路



第四章 组合逻辑电路

- ❖ 数字系统由数字电路模块构成，这些模块可分为两大类：
 - 组合逻辑电路；
 - 时序逻辑电路。
- ❖ 组合逻辑电路 — 功能上无记忆，结构上无反馈；
 - 电路任一时刻的输出状态只取决于该时刻各输入状态的组合，而与电路的原状态无关。



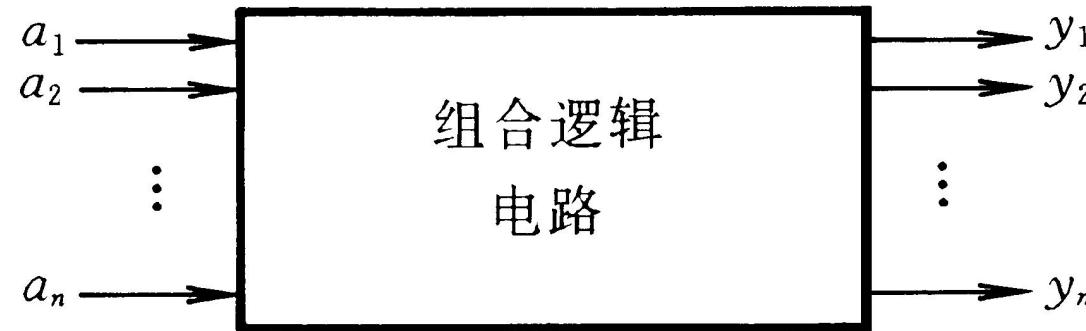
$$S = (A \oplus B) \oplus CI$$
$$CO = (A \oplus B)CI + AB$$

单比特加法器



第四章 组合逻辑电路

- 对于任何一个多输入多输出组合逻辑电路，可以用框图或者一组逻辑函数来表示



$$y_1 = f_1(a_1, a_2, \dots, a_n)$$

$$y_2 = f_2(a_1, a_2, \dots, a_n)$$

$$\mathbf{Y} = F(\mathbf{A})$$

...

$$y_m = f_m(a_1, a_2, \dots, a_n)$$

组合逻辑电路共同特点：不包含存储单元



第四章 组合逻辑电路

§ 4.1 组合逻辑电路的分析与设计

➤组合逻辑电路的分析

➤组合逻辑电路的设计

§ 4.2 组合逻辑电路模块及其应用

§ 4.3 组合逻辑电路中的竞争与冒险

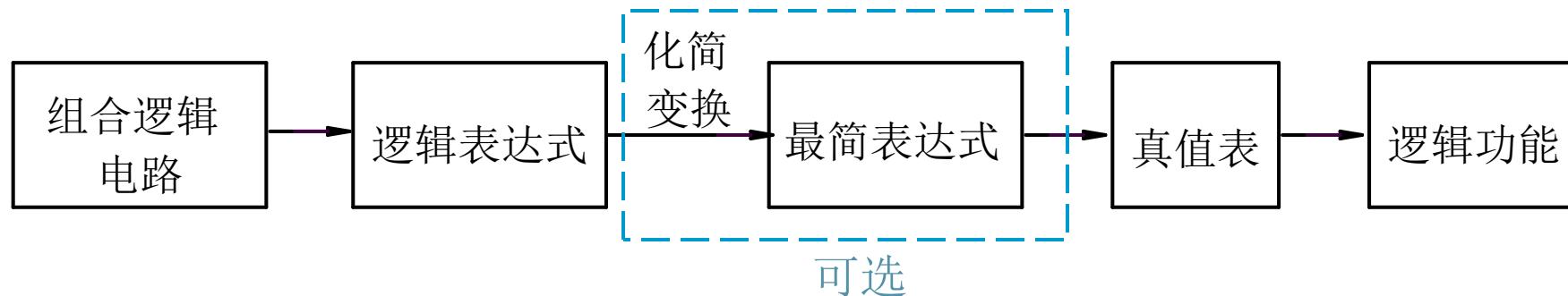


§ 4.1.1 组合逻辑电路分析

■ 组合逻辑电路的分析

➤ 组合逻辑电路的分析是用逻辑函数来**描述**已知的电路，找出输入、输出之间的关系，化简，得到真值表，从而判断电路**功能**；

➤ 分析步骤



§ 4.1.1 分析

- ✓ 写逻辑表达式

$$P = \overline{ABC}$$

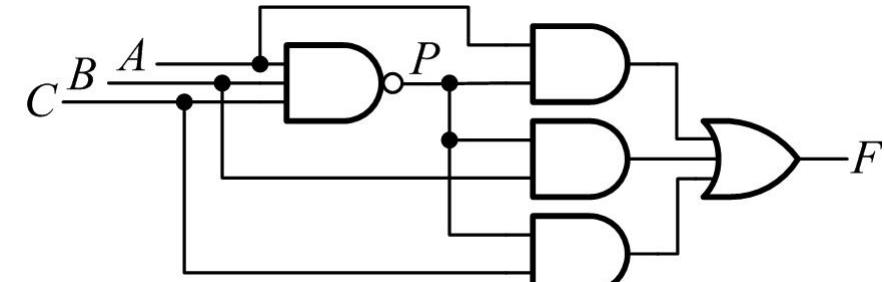
$$F = AP + BP + CP = \overline{A}\overline{ABC} + \overline{B}\overline{ABC} + \overline{C}\overline{ABC}$$

- ✓ 化简与变换

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

- ✓ 列出真值表

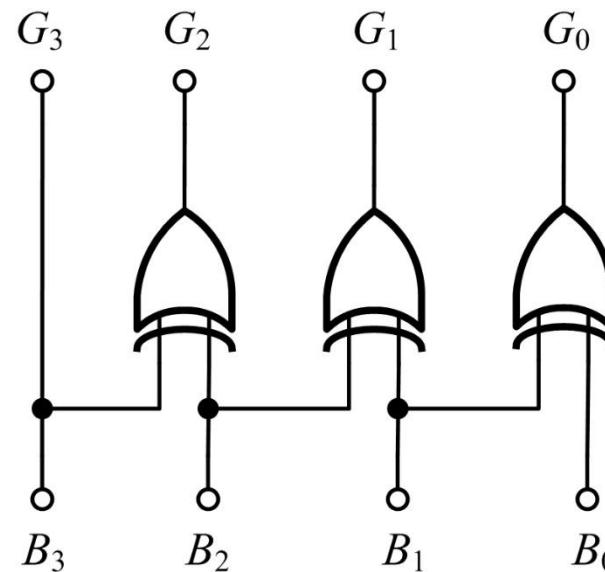
- ✓ 分析逻辑功能：“不一致电路”



A	B	C	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

§ 4.1.1 分析

✓ 自然二进制码至格雷码的转换电路



$$\begin{cases} G_3 = B_3 \\ G_2 = B_3 \oplus B_2 \\ G_1 = B_2 \oplus B_1 \\ G_0 = B_1 \oplus B_0 \end{cases}$$

B_3	B_2	B_1	B_0	G_3	G_2	G_1	G_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	1	1	1
1	1	0	0	1	0	1	0
1	1	0	1	1	0	1	1
1	1	1	0	1	0	0	1
1	1	1	1	1	0	0	0

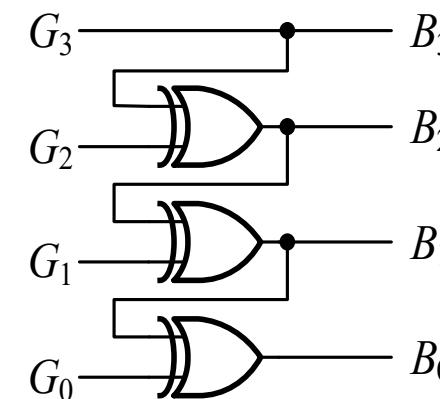


(续)

■ 思考：从格雷码转换到自然二进制码的公式和电路？

$$\begin{cases} G_3 = B_3 \\ G_2 = B_3 \oplus B_2 \\ G_1 = B_2 \oplus B_1 \\ G_0 = B_1 \oplus B_0 \end{cases}$$

$$\begin{cases} B_3 = G_3 \\ B_2 = B_3 \oplus G_2 = G_3 \oplus G_2 \\ B_1 = B_2 \oplus G_1 = G_3 \oplus G_2 \oplus G_1 \\ B_0 = B_1 \oplus G_0 = G_3 \oplus G_2 \oplus G_1 \oplus G_0 \end{cases}$$





第四章 组合逻辑电路

§ 4.1 组合逻辑电路的分析与设计

- 组合逻辑电路的分析
- 组合逻辑电路的设计

§ 4.2 组合逻辑电路模块及其应用

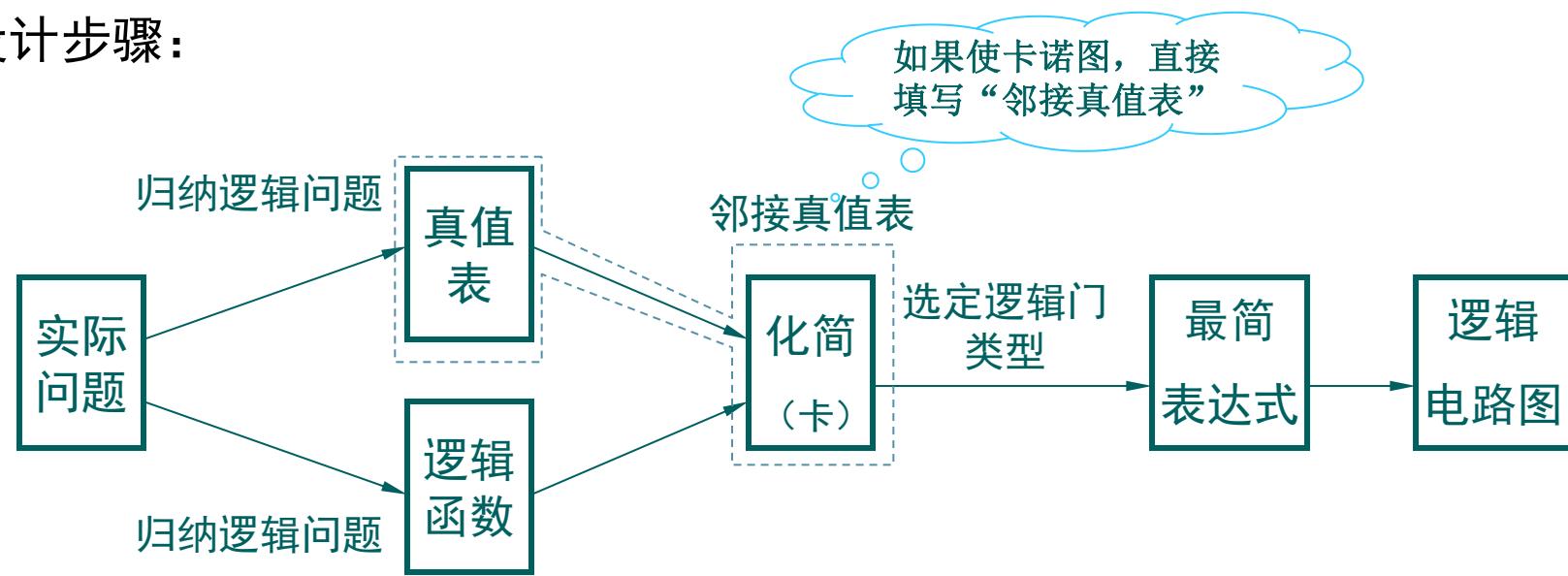
§ 4.3 组合逻辑电路中的竞争与冒险



§ 4.1.2 组合逻辑电路设计

组合逻辑电路的设计方法

- 组合逻辑电路的设计——即：根据给定的逻辑问题，得出能实现设计要求的逻辑电路
- 实现手段：
 - 采用门电路
 - 采用可编程逻辑器件
- 设计步骤：

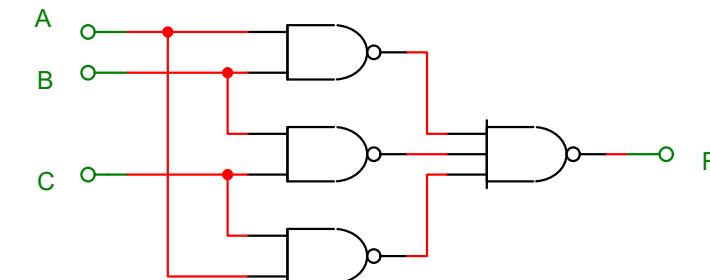
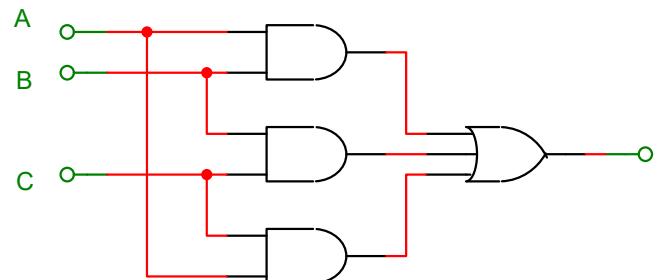


§ 4.1.2 组合逻辑电路设计

- 练习：按“少数服从多数”原则设计三人表决电路

A	B	C	F
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

$$\begin{aligned} F &= \overline{A}\overline{B}C + A\overline{B}\overline{C} + A\overline{B}\overline{C} + ABC \\ &= AB + BC + AC \\ &= \overline{\overline{AB} \cdot \overline{BC} \cdot \overline{AC}} \end{aligned}$$





§ 4.1.2 组合逻辑电路设计

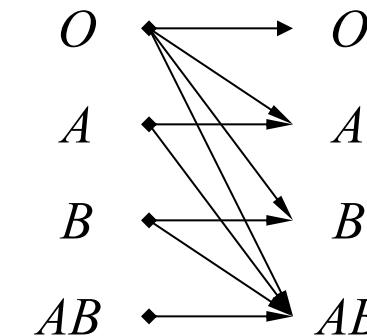
■ 练习：用与非门设计一输血与受血者血型关系检测电路

- 用 A 、 B 两个变量的4种组合表示献血者的血型
- 用 A_1 、 B_1 两个变量的4种组合表示受血者的血型
- 输出变量 F 表示两者之间的关系是否符合配合原则

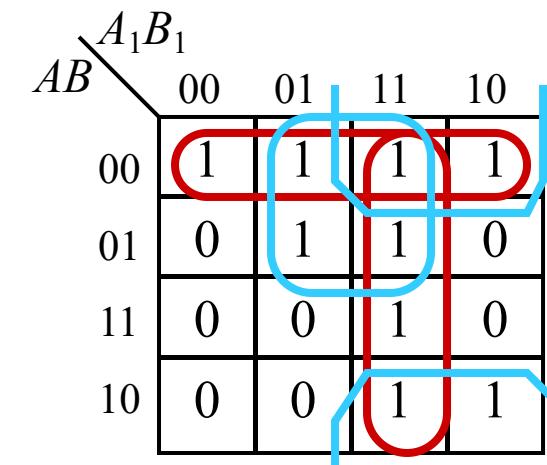
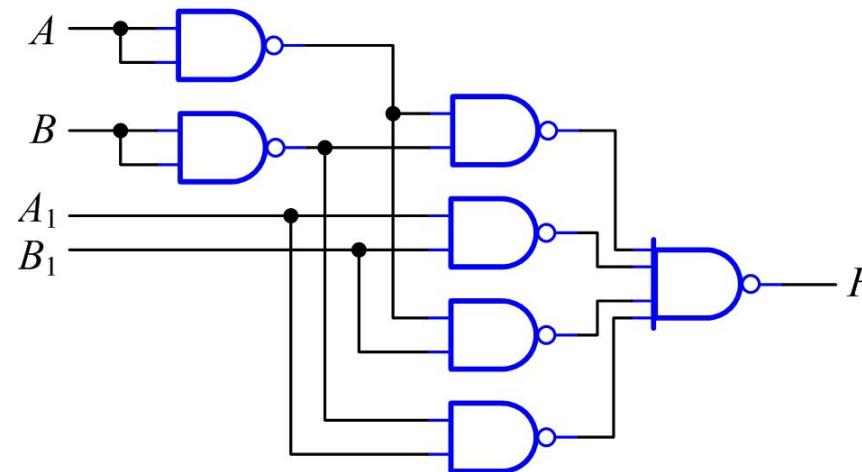
输入变量的表示

血型	献血者		受血者	
	A	B	A_1	B_1
O	0	0	0	0
A	1	0	1	0
B	0	1	0	1
AB	1	1	1	1

血型配合原则



§ 4.1.2 设计



$$\begin{aligned}F &= \overline{\overline{AB}} + A_1B_1 + \overline{\overline{AB}_1} + \overline{\overline{BA}_1} \\&= \overline{\overline{AB}} \cdot \overline{\overline{A_1B_1}} \cdot \overline{\overline{AB}_1} \cdot \overline{\overline{BA}_1}\end{aligned}$$

卡诺图中两个相邻（包括闭合、轴对称）“1格”的最小项可以合并成一个与项，消去一个原变量和反变量

反复利用合并项法则，保留相同变量，消掉相反变量

§ 4.1.2 设计

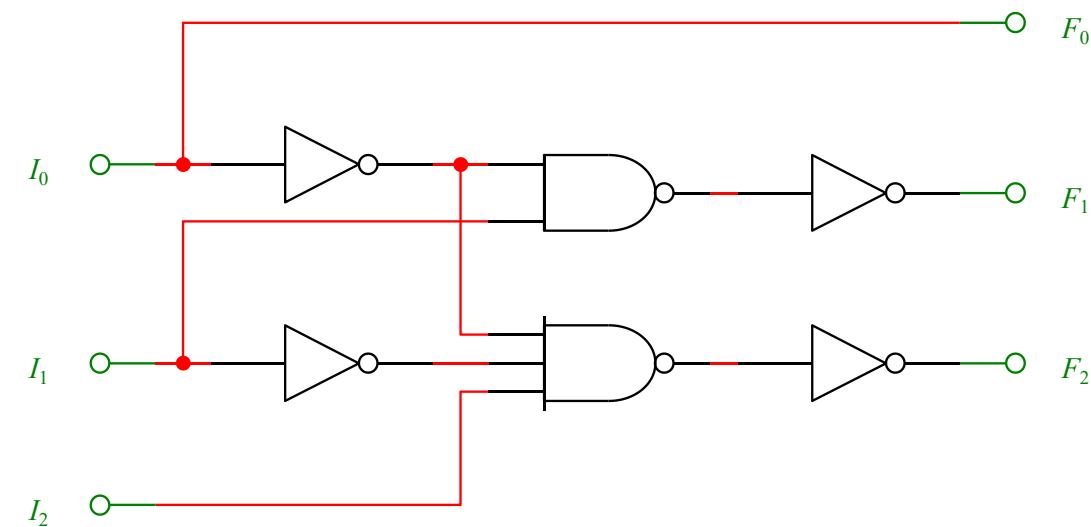
- 练习：设计一个电话信号控制电路：电路有 I_0 （火警）、 I_1 （盗警）和 I_2 （日常业务）三种输入信号，按优先次序分别从 F_0 、 F_1 、 F_2 输出，在同一时间只能有一个信号通过。要求用与非门实现。

输入输出关系表

输入			输出		
I_0	I_1	I_2	F_0	F_1	F_2
0	0	0	0	0	0
1	X	X	1	0	0
0	1	X	0	1	0
0	0	1	0	0	1

$$F_0 = I_0 \quad F_2 = \overline{I_0} \cdot \overline{I_1} \cdot I_2$$

$$F_1 = \overline{I_0} \cdot I_1$$





第四章 组合逻辑电路

§ 4.1 组合逻辑电路的分析与设计

§ 4.2 组合逻辑电路模块及其应用

§ 4.3 组合逻辑电路中的竞争与冒险



§ 4.2 组合逻辑电路模块及其应用

- 编码器
- 译码器
- 数据选择器
- 数值比较器
- 加法器



§ 4.2.1 组合逻辑电路模块 之 编码器

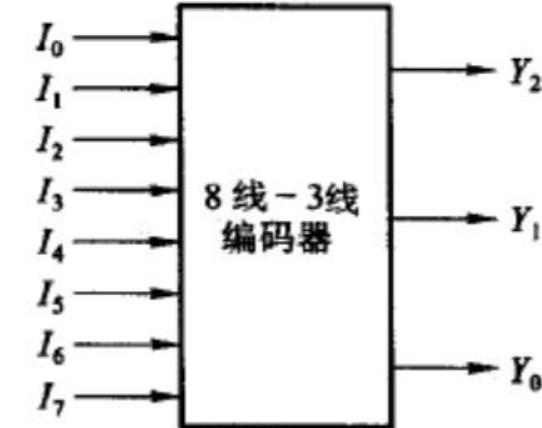
■ 编码器

- 编码器将字母、数字等信息符号编成一组二进制代码；
- 编码器对每一个**有效的**输入信号，都确定地产生的一组二进制代码与之对应；
- 编码器是一种**多输入多输出组合逻辑电路**；
- 通常 m 个输入信号，需要 n 位二进制编码， m 应不大于 2^n 。
- 常用的编码器有二进制编码器。

§ 4.2.1 编码器

■ 二进制编码器

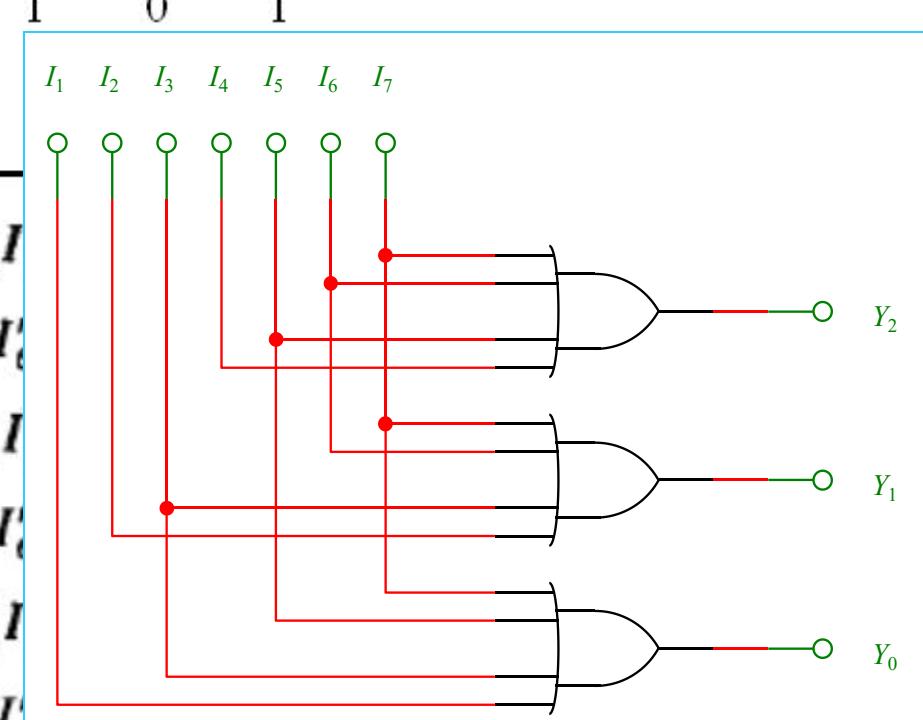
- 用 n 位二进制代码对 2^n 个信号进行编码
- 三位二进制编码器 **8线—3线编码器**



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

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

✓ 电路图



$$\checkmark Y_2 \text{ 表达式 } I_1' I_2' I_3' I_4' I_5' I_6' I_7' + I_0' I_1' I_2' I_3' I_4' I_5' I_6' I$$

输入变量仅限表中取值，其

它为约束项，化简：

$$Y_1 = I_0' I_1' I_2' I_3' I_4' I_5' I_6' I_7' + I_0' I_1' I_2' I_3' I_4' I_5' I_6' I$$

$$+ I_0' I_1' I_2' I_3' I_4' I_5' I_6' I_7' + I_0' I_1' I_2' I_3' I_4' I_5' I_6' I$$

$$Y_0 = I_0' I_1' I_2' I_3' I_4' I_5' I_6' I_7' + I_0' I_1' I_2' I_3' I_4' I_5' I_6' I$$

$$+ I_0' I_1' I_2' I_3' I_4' I_5' I_6' I_7' + I_0' I_1' I_2' I_3' I_4' I_5' I_6' I$$

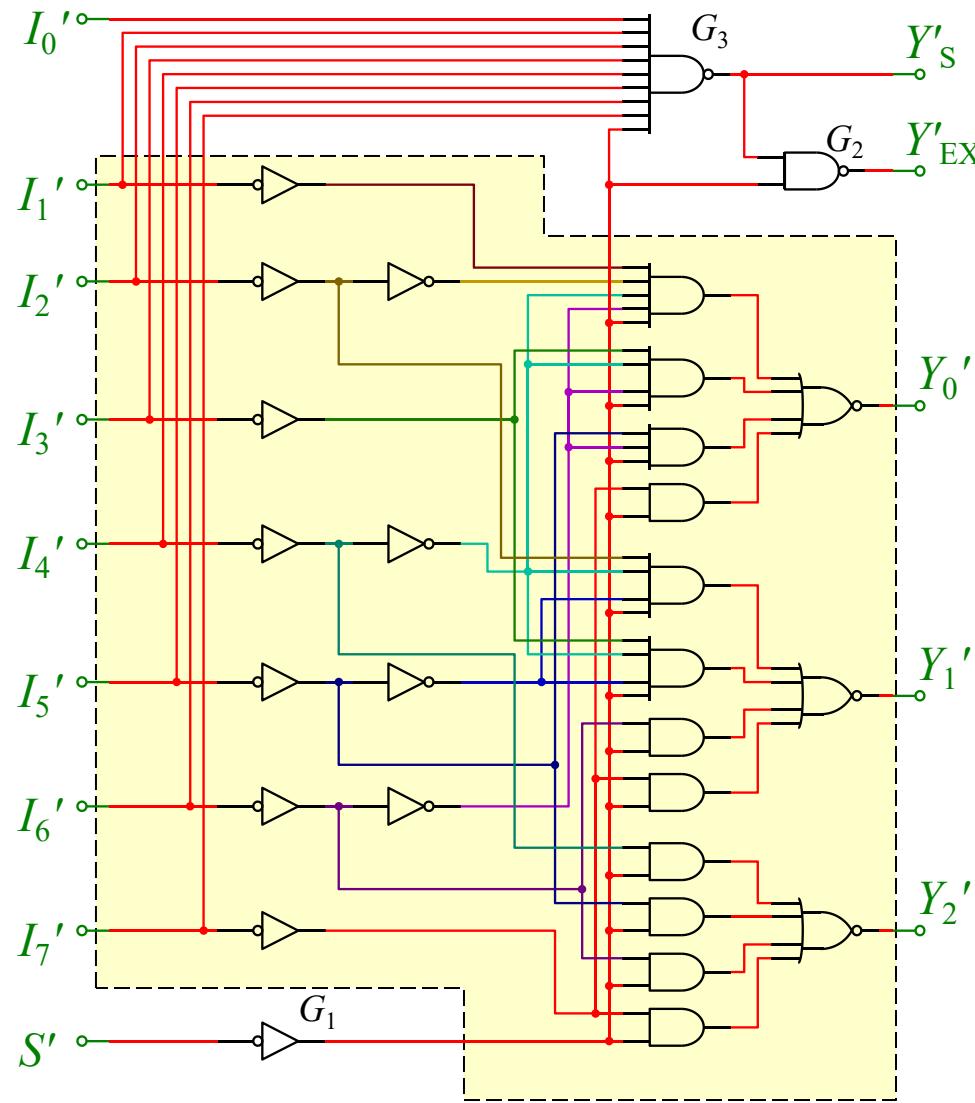


§ 4.2.1 编码器

■ 优先编码器

- 允许同时输入两个以上的编码信号，输入信号规定了优先顺序，当多个输入信号同时出现时，只对**优先级最高的**信号进行编码

- 常用的 8线—3线 优先编码器 74HC**148**
 - 除输入输出信号外，增加了输入、输出使能信号和输出扩展信号



例：74HC148

✓ 使能输入端 \bar{S}

✓ 虚框内为编码器主电路

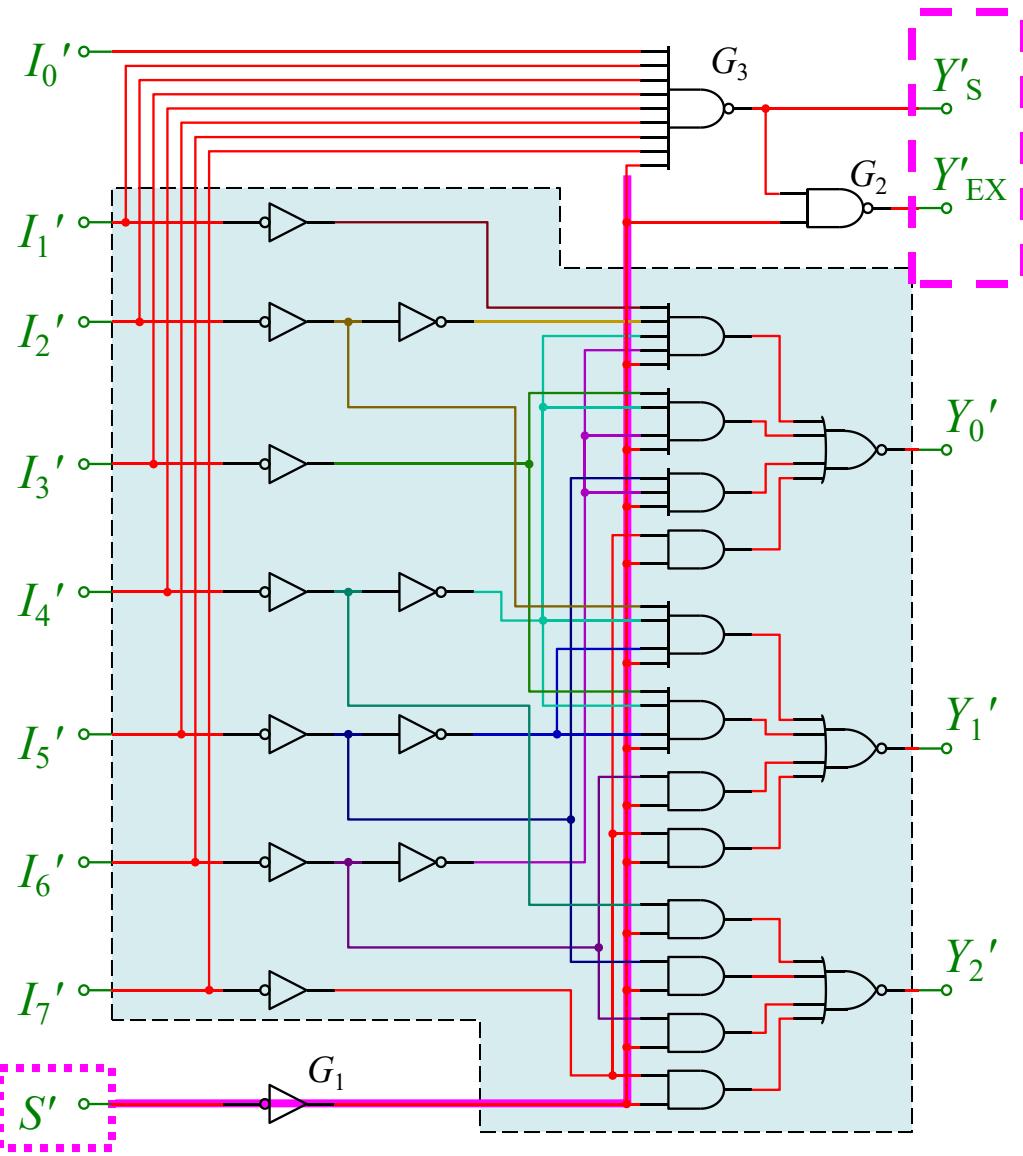
$$\bar{Y}_2 = \overline{(I_4 + I_5 + I_6 + I_7)} \cdot S$$

$$\bar{Y}_1 = \overline{(I_2 \bar{I}_4 \bar{I}_5 + I_3 \bar{I}_4 \bar{I}_5 + I_6 + I_7)} \cdot S$$

$$\bar{Y}_0 = \overline{(I_1 \bar{I}_2 \bar{I}_4 \bar{I}_6 + I_3 \bar{I}_4 \bar{I}_6 + I_5 \bar{I}_6 + I_7)} \cdot S$$

低电平为有效输入

为了扩展电路功能与增加灵活性，还增加了两个门



✓ 片选输入端 S' 低电平有效
高电平封锁

✓ 使能输出端 \bar{Y}_S

$$\bar{Y}_S = \overline{\bar{I}_0 \cdot \bar{I}_1 \cdots \bar{I}_7 \cdot S}$$

只有所有输入为高且 $S=1$ 时，才为0；为低电平时表示电路正常，但无信号输入

✓ 扩展输出端 \bar{Y}_{EX}

$$\bar{Y}_{EX} = \overline{\bar{Y}_S \cdot S}$$

$$= \overline{(I_0 + I_1 + \cdots + I_7) \cdot S}$$

只要任意一个输入为低，且 $S=1$ 时，才为0；低电平表示电路工作且有信号输入

■ 优先编码器（续）

低电平为有效输入

I7优先级最高，容许多个同时输入

可以俗称为

“有信号输入” 和 “无信号输入” 指示

✓ 功能表

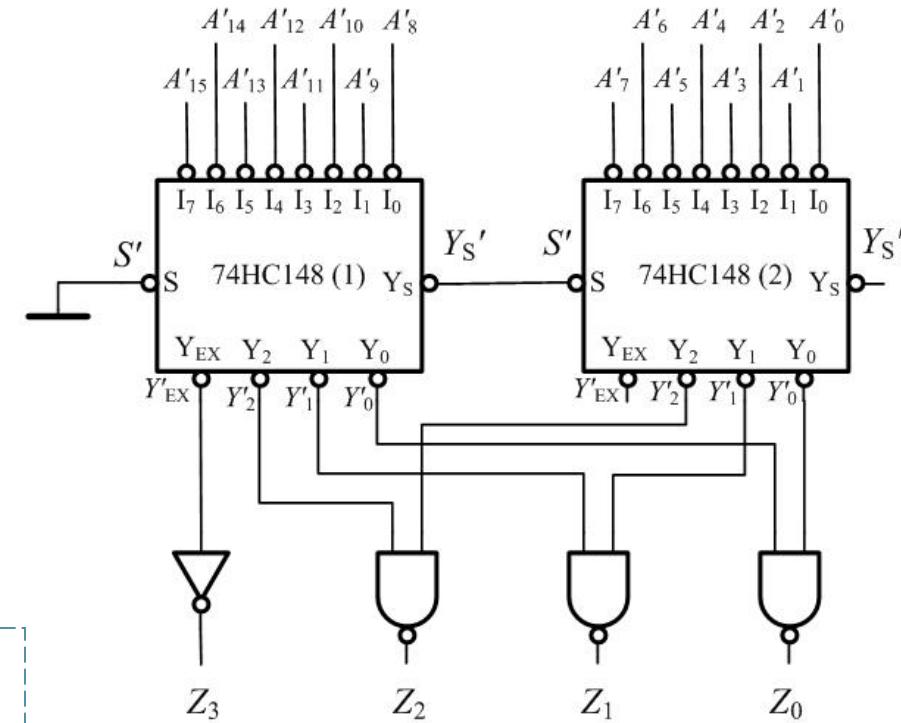
§ 4.2.1 编码器

■ 优先编码器（续）

➤ 编码器的扩展 →

- 16线-4线优先编码器

- ✓ 按优先级顺序，把高优先级
 - (1) 片的“无信号输入”端接低优先级(2)片的使能端；只有A15-A8均无输入信号时，才容许(2)的A7-A0编码
- ✓ 高优先级 (1) 片的“有信号输入”端作为编码输出最高位；
- ✓ 低三位为两片编码器输出信号的与非



A15优先级最高

注意：在中规模电路中，逻辑框图外部输入或输出端加小圆圈，同时在外部标注的输入或输出端信号加上非号，1、以低电平作为有效信号；2、门电路输入端加小圆圈有时候表示信号经反向后输入。需具体情况具体分析（通常会说明）。



§ 4.2.2 组合逻辑电路模块 之 译码器

- 与编码器相反，译码器将输入代码转换成特定的输出信号（**最常用的地址译码器**）
- 两种常用的译码器：
 - 二进制译码器
 - 数字显示译码器



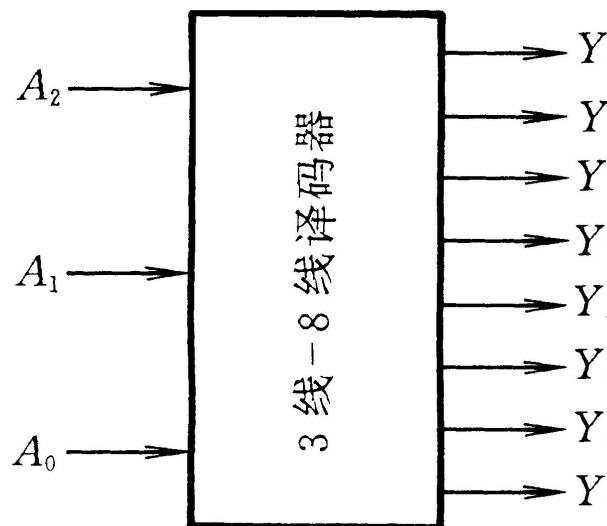
§ 4.2.2 译码器

■ 二进制译码器

- 译码器有 n 个输入信号和 m 个输出信号
- $m=2^n$ 二进制全译码器
 - 3线-8线译码器

§ 4.2.2 译码器

3线-8线译码器



$$Y_0 = \overline{A}_2 \overline{A}_1 \overline{A}_0$$

$$Y_1 = \overline{A}_2 \overline{A}_1 A_0$$

$$Y_2 = \overline{A}_2 A_1 \overline{A}_0$$

$$Y_3 = \overline{A}_2 A_1 A_0$$

$$Y_4 = A_2 \overline{A}_1 \overline{A}_0$$

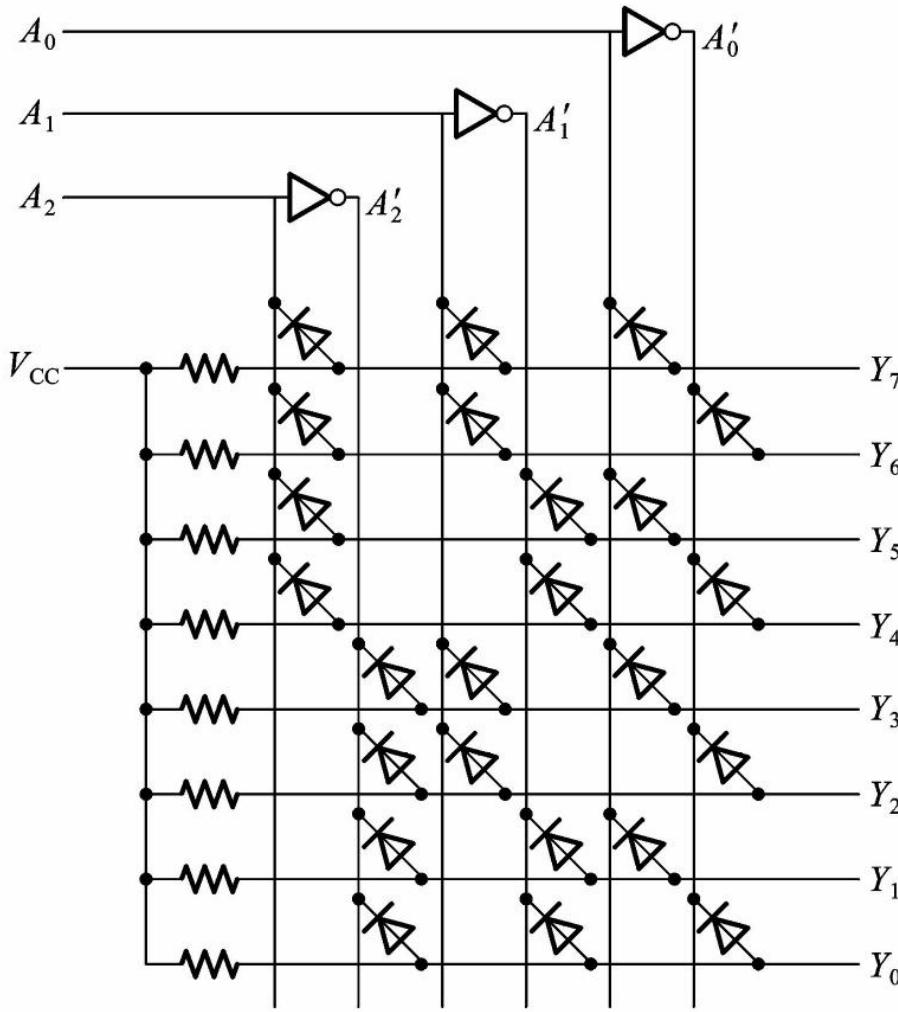
$$Y_5 = A_2 \overline{A}_1 A_0$$

$$Y_6 = A_2 A_1 \overline{A}_0$$

$$Y_7 = A_2 A_1 A_0$$

§ 4.2.2 译码器

用二极管与门阵列组成
3线-8线译码器



$$Y_0 = \overline{A_2} \overline{A_1} \overline{A_0}$$

$$Y_1 = \overline{A_2} \overline{A_1} A_0$$

$$Y_2 = \overline{A_2} A_1 \overline{A_0}$$

$$Y_3 = \overline{A_2} A_1 A_0$$

$$Y_4 = A_2 \overline{A_1} \overline{A_0}$$

$$Y_5 = A_2 \overline{A_1} A_0$$

$$Y_6 = A_2 A_1 \overline{A_0}$$

$$Y_7 = A_2 A_1 A_0$$



§ 4.2.2 译码器

- 集成译码器74HC138
- 常用的3线-8线译码器

S_1 、 \overline{S}_2 和 \overline{S}_3

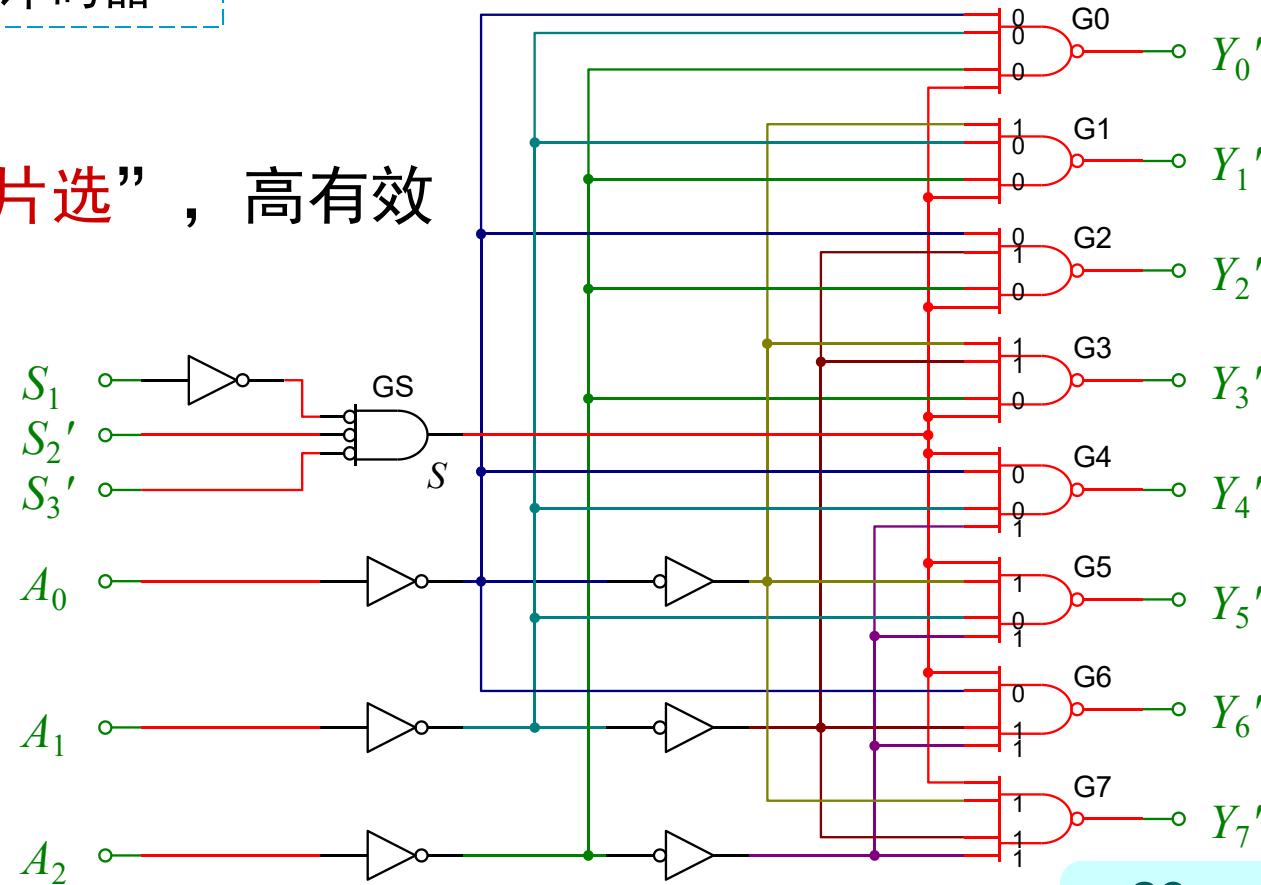
为使能端，又叫作“片选”，高有效

$$S = S_1 \overline{S}_2 + \overline{S}_3$$

作用：

- ✓ $S=1$ 时，工作
- ✓ $S=0$ 时，禁止，所有输出封锁为高电平

输入					输出							
S_1	$\overline{S}_2 + \overline{S}_3$	A_2	A_1	A_0	\overline{Y}_7	\overline{Y}_6	\overline{Y}_5	\overline{Y}_4	\overline{Y}_3	\overline{Y}_2	\overline{Y}_1	\overline{Y}_0
0	×	×	×	×	1	1	1	1	1	1	1	1
×	1	×	×	×	1	1	1	1	1	1	1	1
1	0	0	0	0	1	1	1	1	1	1	1	0
1	0	0	0	1	1	1	1	1	1	1	0	1
1	0	0	1	0	1	1	1	1	1	0	1	1
1	0	0	1	1	1	1	1	1	0	1	1	1
1	0	1	0	0	1	1	1	0	1	1	1	1
1	0	1	0	1	1	1	0	1	1	1	1	1
1	0	1	1	0	1	0	1	1	1	1	1	1
1	0	1	1	1	0	1	0	1	1	1	1	1



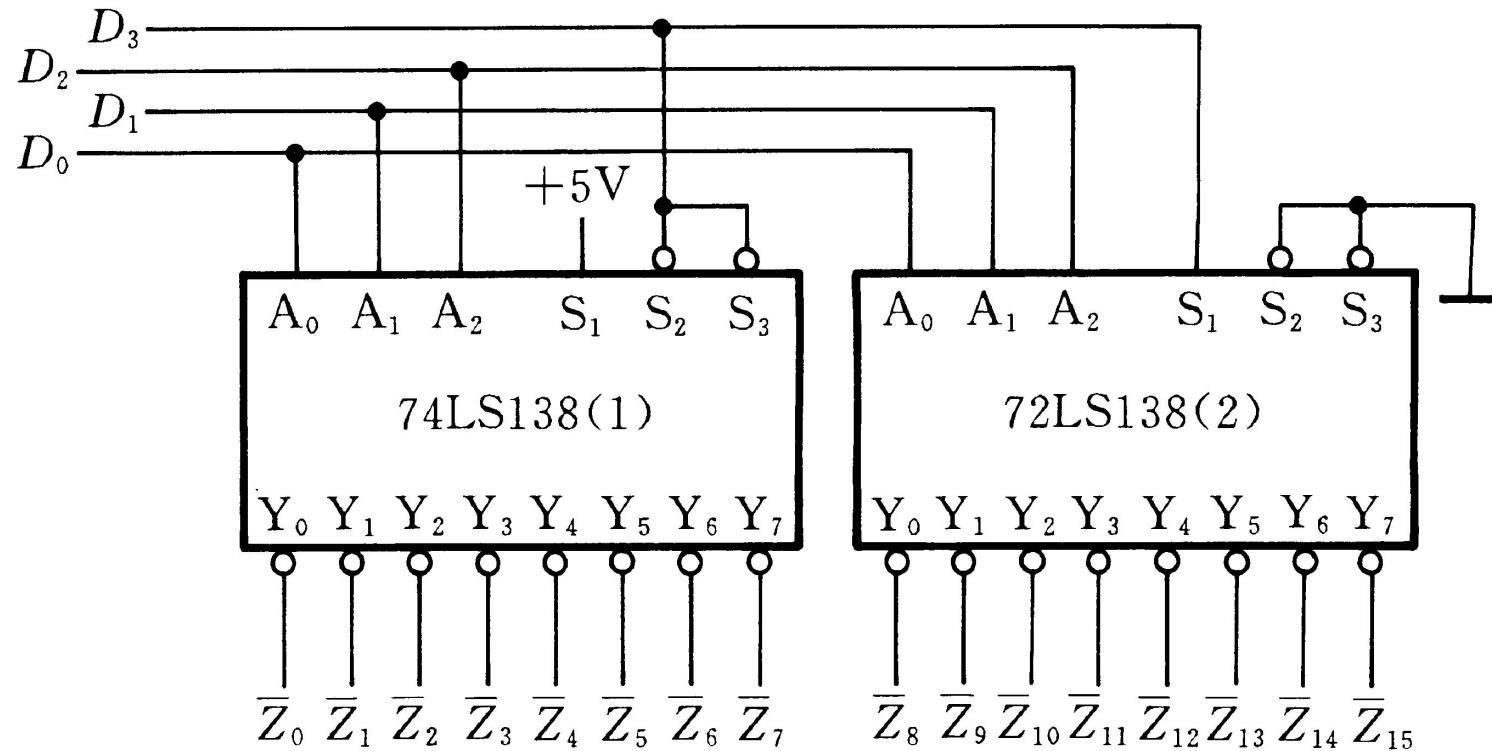
$$S = S_1 \overline{S_2} + \overline{S_3}$$

§ 4.2.2 译码器

思考：如果需要 6线-64线译码器，
需要几枚74138芯片？

■ 译码器的扩展

➤ 例题：将两片74HC138扩展为4线—16线译码器

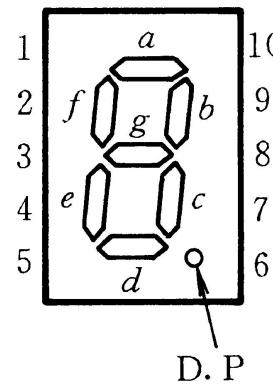


§ 4.2.2 译码器

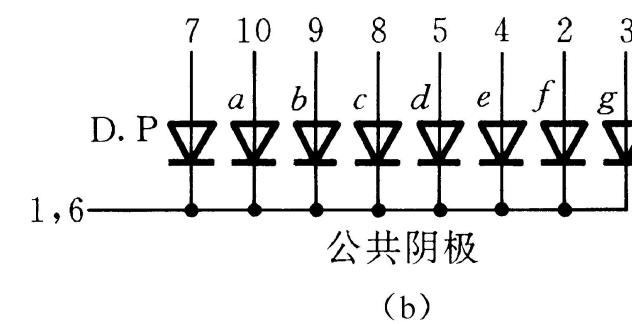
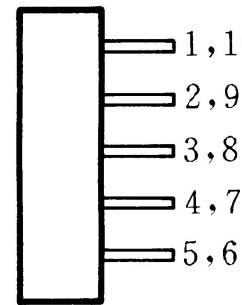
■ 数字显示译码器 (随处可见)

- 能够显示数字、字母或符号的器件称为**数字显示器**
- 能把数字量翻译成数字显示器所能识别的信号的译码器称为**数字显示译码器**。

■ 七段数字显示器 (半导体数码管BS201A, 含小数点八段)



(a)



§ 4.2.2 译码器

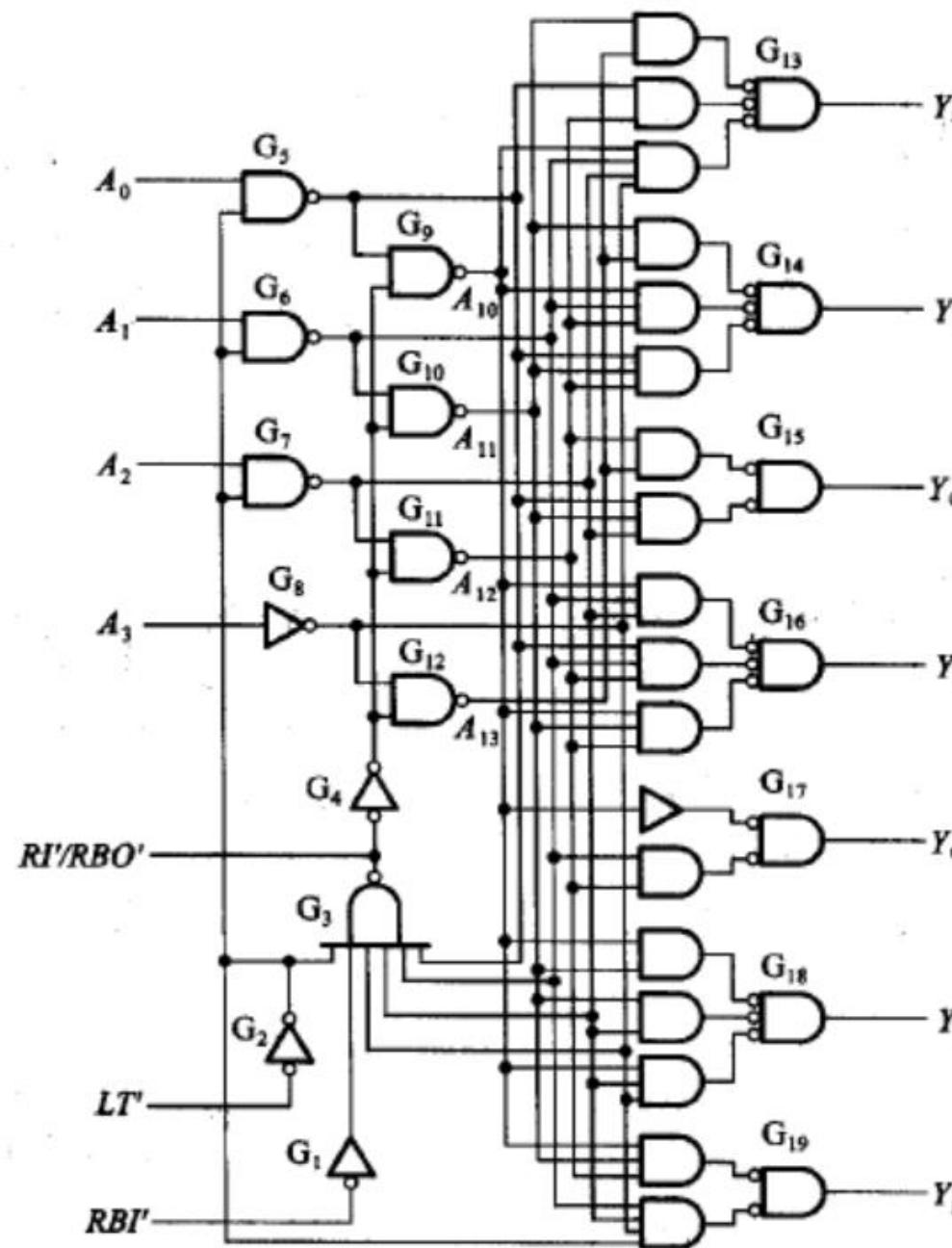
七段数字 显示译码

每个输入代码
多个输出有效



$$\left\{ \begin{array}{l} Y_a = (A'_3 A'_2 A'_1 A_0 + A_3 A_1 + A_2 A'_0)' \\ Y_b = (A_3 A_1 + A_2 A_1 A'_0 + A_2 A'_1 A_0)' \\ Y_c = (A_3 A_2 + A'_2 A_1 A'_0)' \\ Y_d = (A_2 A_1 A_0 + A_2 A'_1 A'_0 + A'_2 A'_1 A_0)' \\ Y_e = (A_2 A'_1 + A_0)' \\ Y_f = (A'_3 A'_2 A_0 + A'_2 A_1 + A_1 A_0)' \\ Y_g = (A'_3 A'_2 A'_1 + A_2 A_1 A_0)' \end{array} \right.$$

七段显示译码器7448逻辑公式

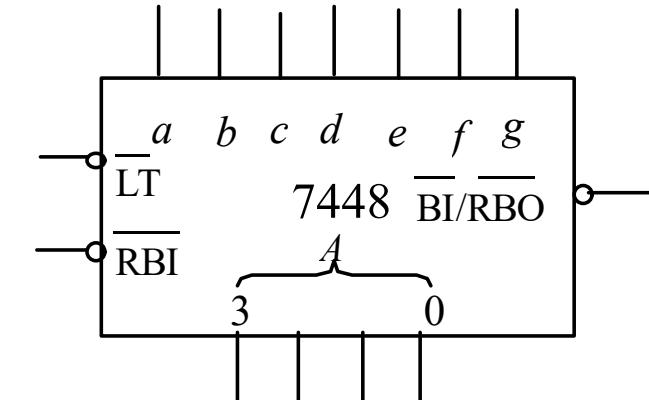


七段显示译码器7448逻辑图

§ 4.2.2 译码器

➤ 七段显示译码器7448

- ✓ 试灯 $\overline{LT}=0$ 所有灯亮
- ✓ 灭零 $\overline{RBI}=0$ 把不希望显示的零熄灭
- ✓ 正常译码显示 $\overline{LT}=1, \overline{RBI}=1$
- ✓ 控制端 $\overline{RI}/\overline{RBO}$ 可作输入端/输出端



输入 $\overline{RI}=0$, 数码管全灭, \overline{RI} 为灭灯输入端;

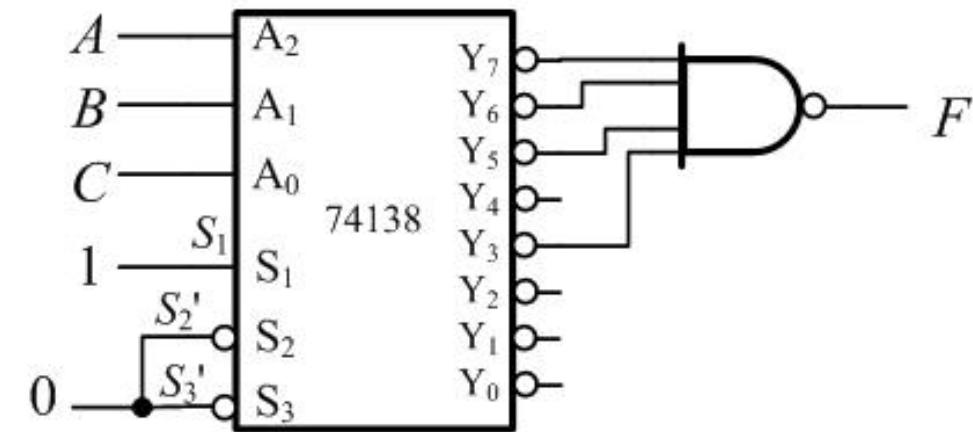
作为输出端使用时, 称为灭零输出端, 如
当 $RBI'=0$, 输入为0000时, RBO' 输出0,
指示该片处于灭零状态

- ✓ $\overline{BI}/\overline{RBO}$ 和 \overline{RBI} 配合使用, 可以实现多位数显示时的“无效0消隐”功能

§ 4.2.2 译码器

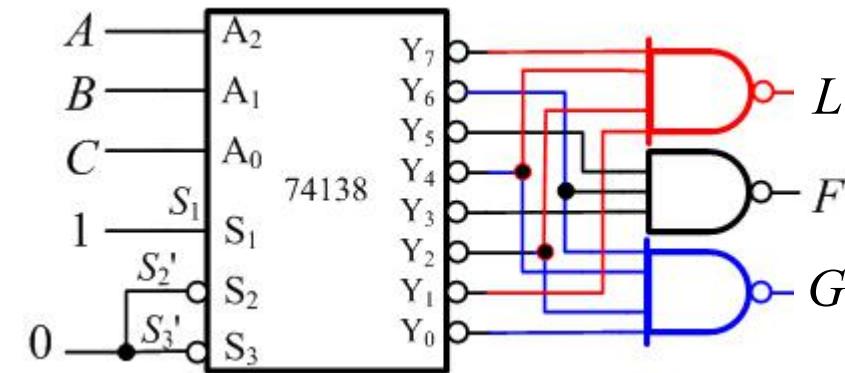
- 二进制全译码器的应用——实现组合逻辑函数
 - 译码器的每个输出端分别与一个最小项相对应

$$\begin{aligned} F &= AB + BC + AC \\ &= \overline{ABC} + A\overline{B}C + A\overline{B}\overline{C} + ABC \\ &= m_3 + m_5 + m_6 + m_7 \\ &= \overline{m_3} \cdot \overline{m_5} \cdot \overline{m_6} \cdot \overline{m_7} \end{aligned}$$



■ 实现组合逻辑函数

输入			输出		
A	B	C	L	F	G
0	0	0	0	0	1
0	0	1	1	0	0
0	1	0	1	0	1
0	1	1	0	1	0
1	0	0	1	0	1
1	0	1	0	1	0
1	1	0	0	1	1
1	1	1	1	0	0



$$L = \overline{A} \overline{B} C + \overline{A} B \overline{C} + A \overline{B} \overline{C} + ABC = m_1 + m_2 + \overline{\overline{m}_4 + m_7} = \overline{\overline{m}_1 \cdot \overline{m}_2 \cdot \overline{m}_4 \cdot \overline{m}_7}$$

$$F = \overline{A} BC + A \overline{B} C + AB \overline{C} = m_3 + m_5 + m_6 = \overline{\overline{m}_3 \cdot \overline{m}_5 \cdot \overline{m}_6}$$

$$G = \overline{A} \overline{B} \overline{C} + \overline{A} B \overline{C} + A \overline{B} \overline{C} + AB \overline{C} = m_0 + m_2 + m_4 + m_6 = \overline{\overline{m}_0 \cdot \overline{m}_2 \cdot \overline{m}_4 \cdot \overline{m}_6}$$

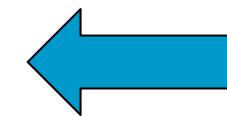
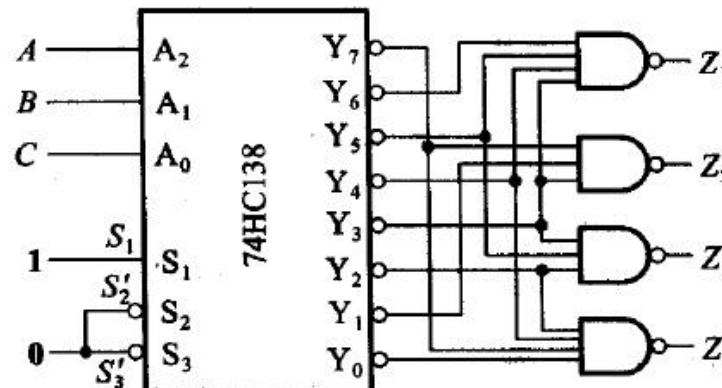


■ 用译码器实现组合逻辑函数

$$\begin{cases} Z_1 = AC' + A'BC + AB'C \\ Z_2 = BC + A'B'C \\ Z_3 = A'B + AB'C \\ Z_4 = A'BC' + B'C' + ABC \end{cases}$$



$$\begin{cases} Z_1 = ABC' + AB'C' + A'BC + AB'C = m_3 + m_4 + m_5 + m_6 \\ Z_2 = ABC + A'BC + A'B'C = m_1 + m_3 + m_7 \\ Z_3 = A'BC + A'BC' + AB'C = m_2 + m_3 + m_5 \\ Z_4 = A'BC' + AB'C' + A'B'C' + ABC = m_0 + m_2 + m_4 + m_7 \end{cases}$$

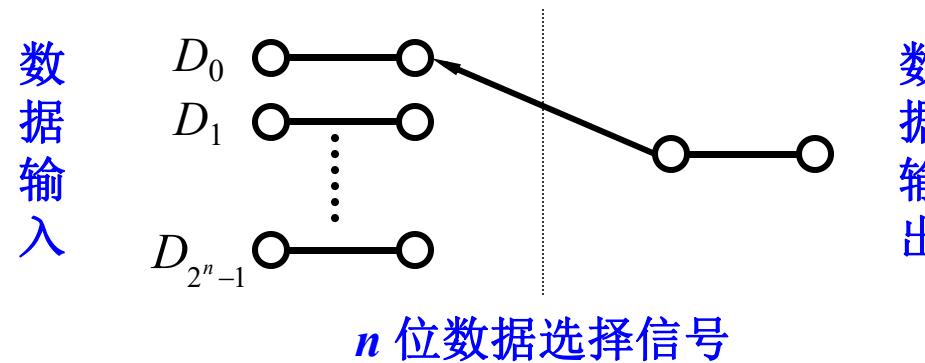


$$\begin{cases} Z_1 = (m'_3 \cdot m'_4 \cdot m'_5 \cdot m'_6)' \\ Z_2 = (m'_1 \cdot m'_3 \cdot m'_7)' \\ Z_3 = (m'_2 \cdot m'_3 \cdot m'_5)' \\ Z_4 = (m'_0 \cdot m'_2 \cdot m'_4 \cdot m'_7)' \end{cases}$$

§ 4.2.3 组合逻辑电路模块 之 数据选择器

■ 数据选择器

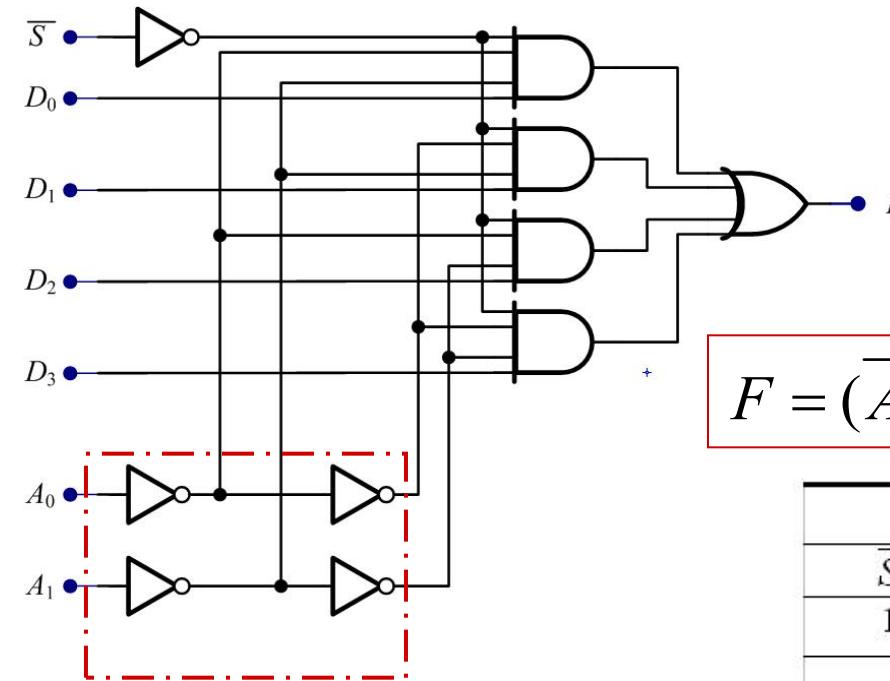
- 根据地址选择码从多路输入数据中选择一路输出



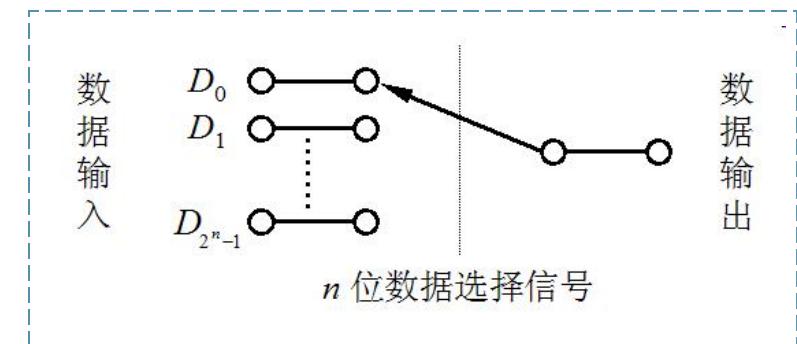
- ✓ 注意：选择信号位数n与数据输入信号个数满足的关系
- ✓ 常用的数据选择器有4选1、8选1、16选1等多种类型

§ 4.2.3 数据选择器

• 四选一数据选择器



全地址译码



逻辑表达式

$$F = (\overline{A_1} \overline{A_0} D_0 + \overline{A_1} A_0 D_1 + A_1 \overline{A_0} D_2 + A_1 A_0 D_3) \cdot S$$

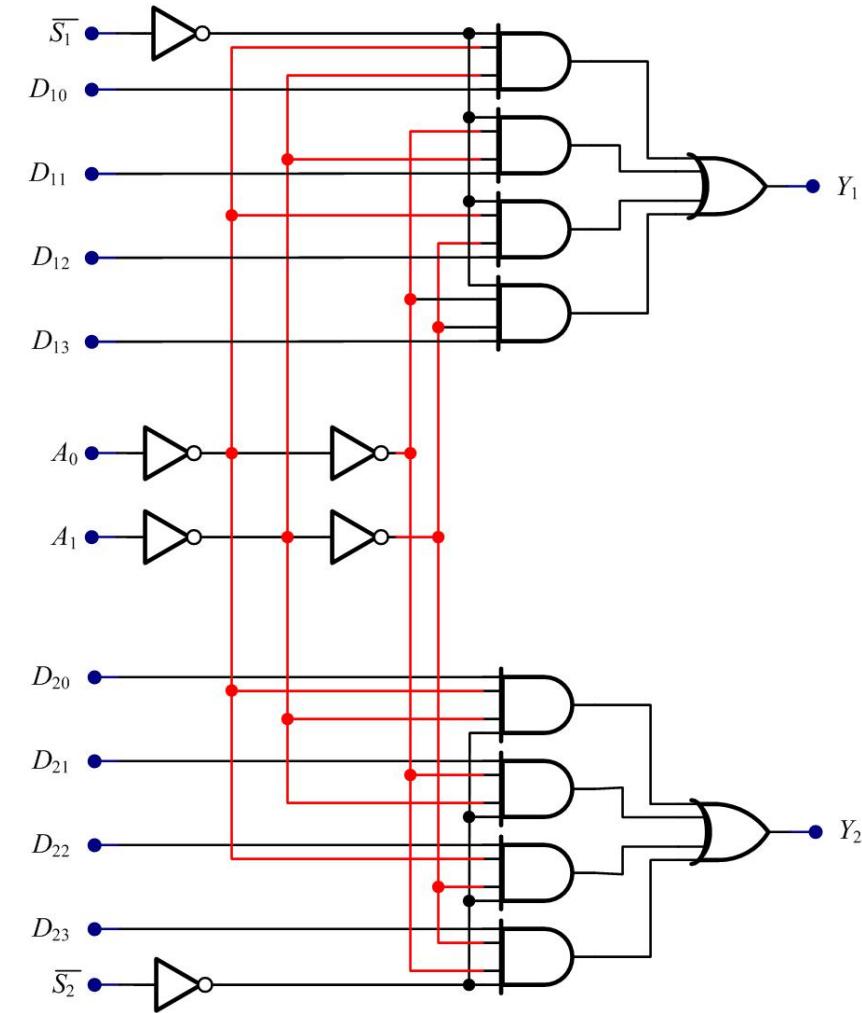
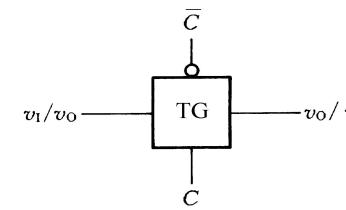
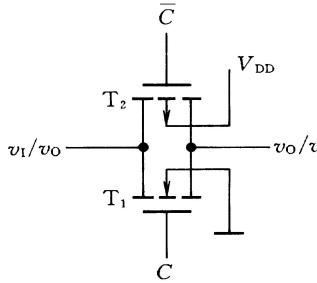
输入						输出	
\bar{S}	A_1	A_0	D_3	D_2	D_1	D_0	F
1	\times	\times	\times	\times	\times	\times	0
0	0	0	\times	\times	\times	0	0
			\times	\times	\times	1	1
	0	1	\times	\times	0	\times	0
		1	\times	\times	1	\times	1
	1	0	\times	0	\times	\times	0
			\times	1	\times	\times	1
1	1	\times	0	\times	\times	\times	0
		1	\times	\times	\times	\times	1



§ 4.2.3 数据选择器

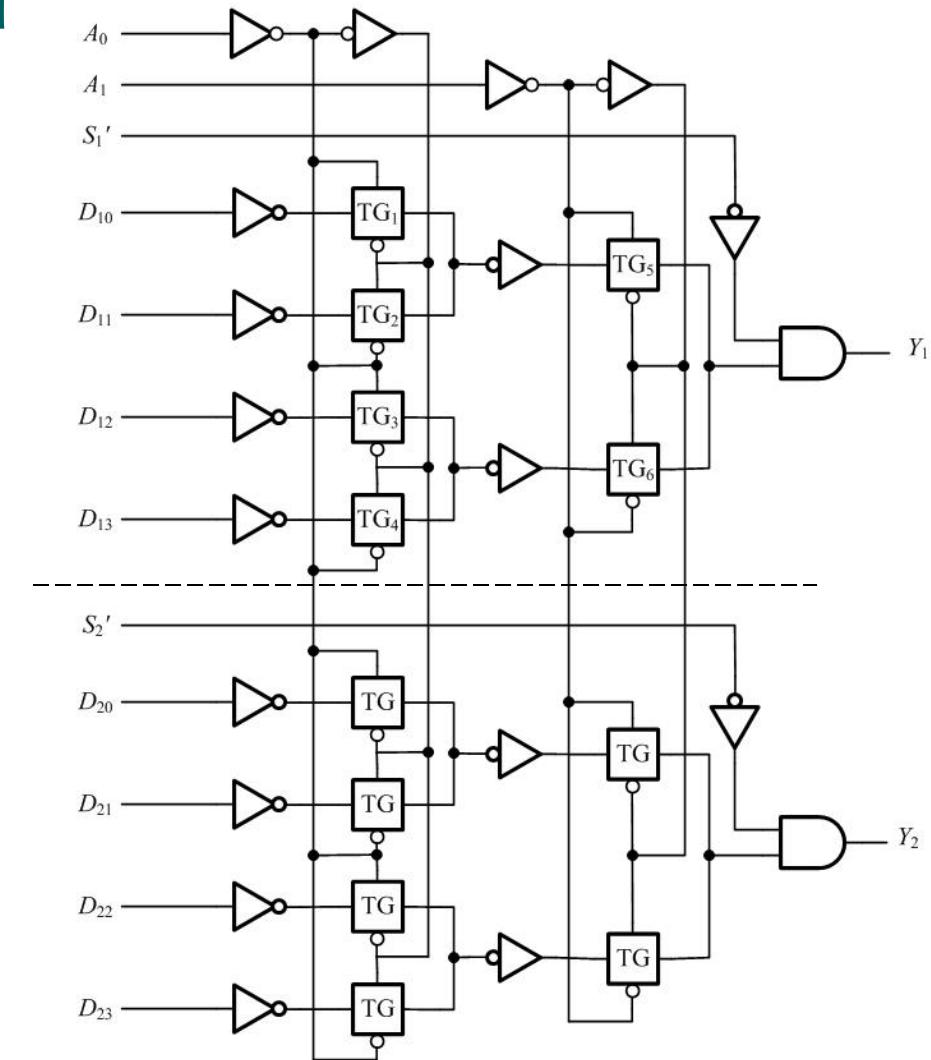
- 双4选1数据选择器
74LS153

问题：利用我们之前学的哪个门电路可以很方便的实现数据选择器？



§ 4.2.3 数据选择器

- 用传输门构成的双4选1数据选择器
- ◆ 74HC153



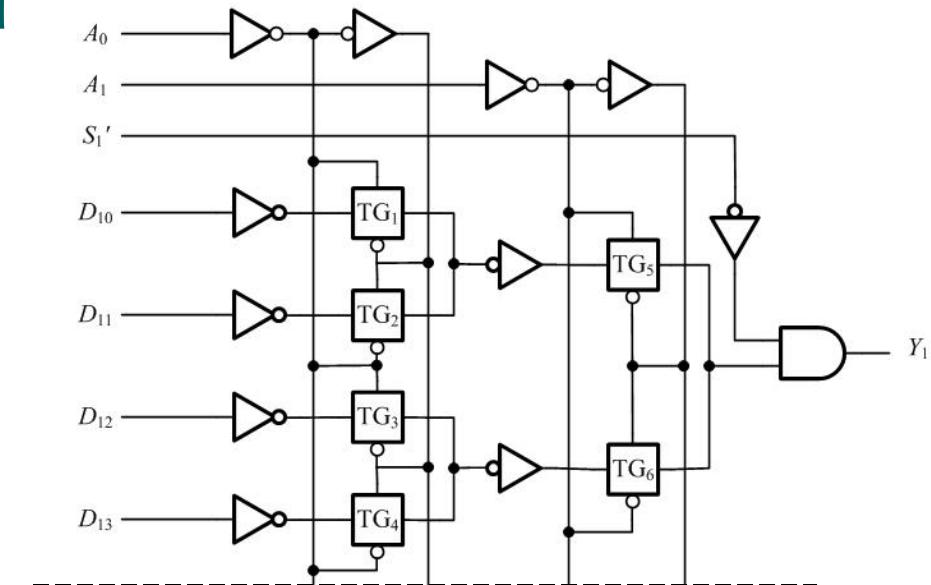


§ 4.2.3 数据选择器

- 用传输门构成的双4选1数据选择器

◆ 74HC153

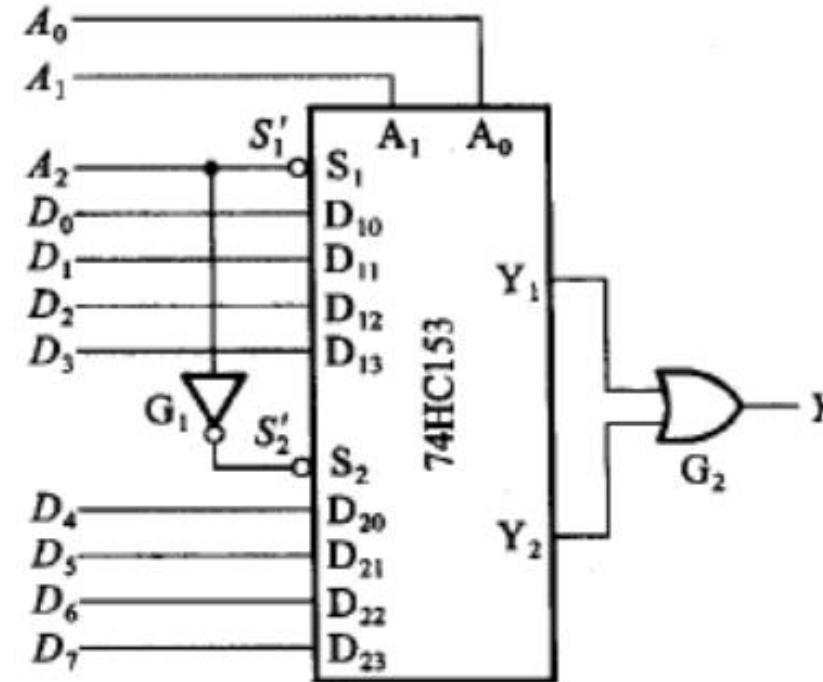
S_1'	A_1	A_0	Y_1
1	X	X	0
0	0	0	D_{10}
0	0	1	D_{11}
0	1	0	D_{12}
0	1	1	D_{13}



$$Y_1 = S_1[D_0(\overline{A}_1 \cdot \overline{A}_0) + D_1(\overline{A}_1 \cdot A_0) + D_2(A_1 \cdot \overline{A}_0) + D_3(A_1 \cdot A_0)]$$

§ 4.2.3 数据选择器

利用74HC153 两个4选1扩展成8选1?



$$\begin{aligned}Y = & (A_2'A_1'A_0')D_0 + (A_2'A_1'A_0)D_1 + (A_2'A_1A_0')D_2 + (A_2'A_1A_0)D_3 \\& + (A_2A_1'A_0')D_4 + (A_2A_1'A_0)D_5 + (A_2A_1A_0')D_6 + (A_2A_1A_0)D_7\end{aligned}$$

§ 4.2.3 数据选择器

■ 数据选择器的应用

➤ 实现组合逻辑函数

- 例题：用8选1数据选择器 74151 实现组合逻辑函数 F

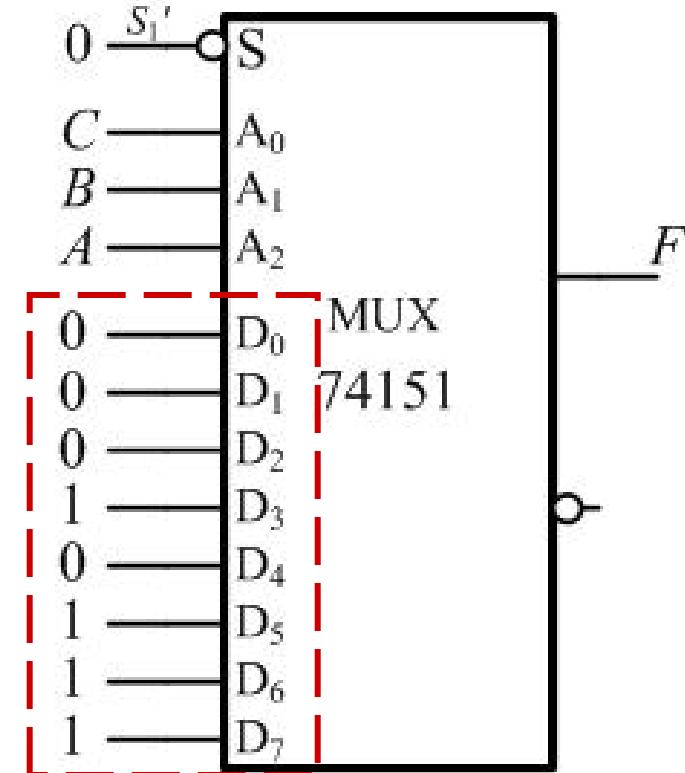
$$\bullet F = AB + BC + AC$$

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

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

❖ 本例题中：逻辑函数的变量和地址输入变量个数相同

$$F = \overline{\overline{A}\overline{B}\overline{C}}D_0 + \overline{\overline{A}\overline{B}C}D_1 + \overline{\overline{A}B\overline{C}}D_2 + \overline{\overline{A}BC}D_3 + \overline{A\overline{B}\overline{C}}D_4 + \overline{A\overline{B}C}D_5 + A\overline{B}\overline{C}D_6 + ABCD_7$$





§ 4.2.3 数据选择器

- 例题：用4选1数据选择器实现逻辑函数：

$$F = AB + AC + BC$$

❖ 逻辑函数的变量**多于地址输入变量个数**

对比

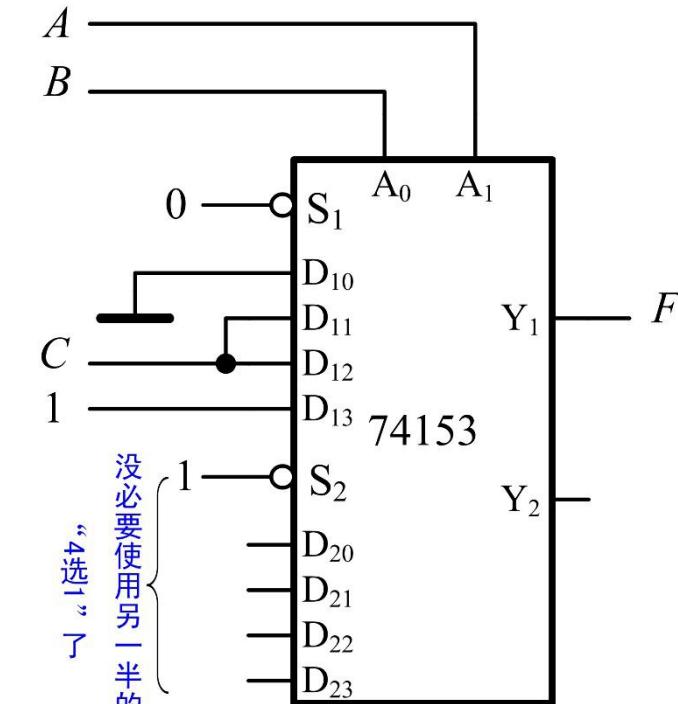
$$\begin{aligned} F &= AB + AC + BC \\ &= A\bar{B}\bar{C} + \bar{A}BC + \bar{A}\bar{B}C + ABC \\ &= \bar{A}\bar{B}C + \bar{A}\bar{B}C + AB(\bar{C} + C) \end{aligned}$$

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

❖ 4选1选择器的输出表达式

$$F = \bar{A}_1 \bar{A}_0 D_0 + \bar{A}_1 A_0 D_1 + A_1 \bar{A}_0 D_2 + A_1 A_0 D_3$$

$$A = A_1, \quad B = A_0 \quad D_0 = 0 \quad D_1 = D_2 = C \quad D_3 = 1$$





§ 4.2.4 组合逻辑电路模块 之 数值比较器

■ 数值比较器

- 对两个位数相同的二进制整数进行数值比较

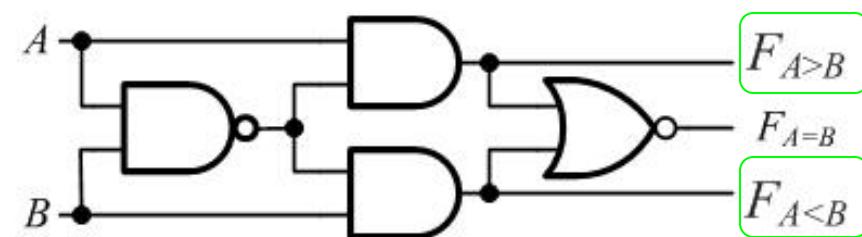
- 例如：1位数值比较器

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

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

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

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





- 多位比较器 (如 $A_1A_0 \text{ vs } B_1B_0$) (考虑优先级)

数值输入		级联输入			输出		
$A_1\ B_1$	$A_0\ B_0$	$I_{A>B}$	$I_{A<B}$	$I_{A=B}$	$F_{A>B}$	$F_{A<B}$	$F_{A=B}$
$A_1 > B_1$	$\times\ \times$	\times	\times	\times	1	0	0
$A_1 < B_1$	$\times\ \times$	\times	\times	\times	0	1	0
$A_1 = B_1$	$A_0 > B_0$	\times	\times	\times	1	0	0
$A_1 = B_1$	$A_0 < B_0$	\times	\times	\times	0	1	0
$A_1 = B_1$	$A_0 = B_0$	1	0	0	1	0	0
$A_1 = B_1$	$A_0 = B_0$	0	1	0	0	1	0
$A_1 = B_1$	$A_0 = B_0$	0	0	1	0	0	1

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

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

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

- 多位比较器 —— 以2位为例

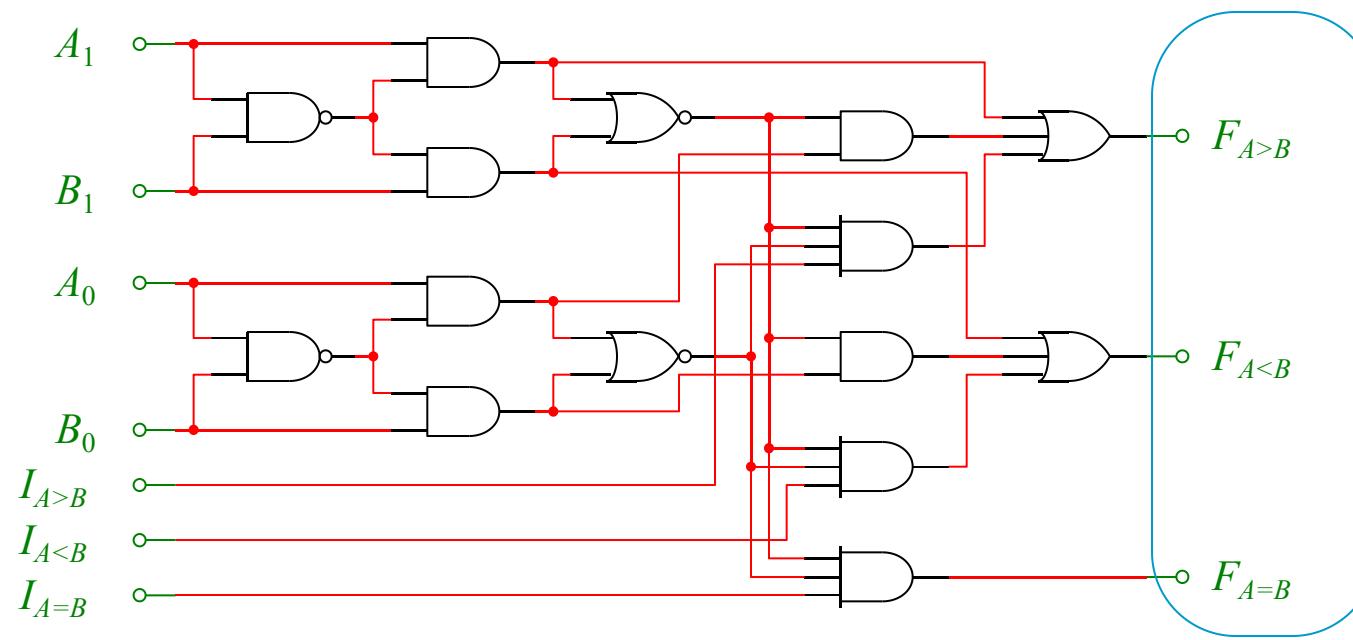
$$F_{A>B} = (A_1 > B_1) + (A_1 = B_1) \cdot (A_0 > B_0) + (A_1 = B_1) \cdot (A_0 = B_0) \cdot I_{A>B}$$

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

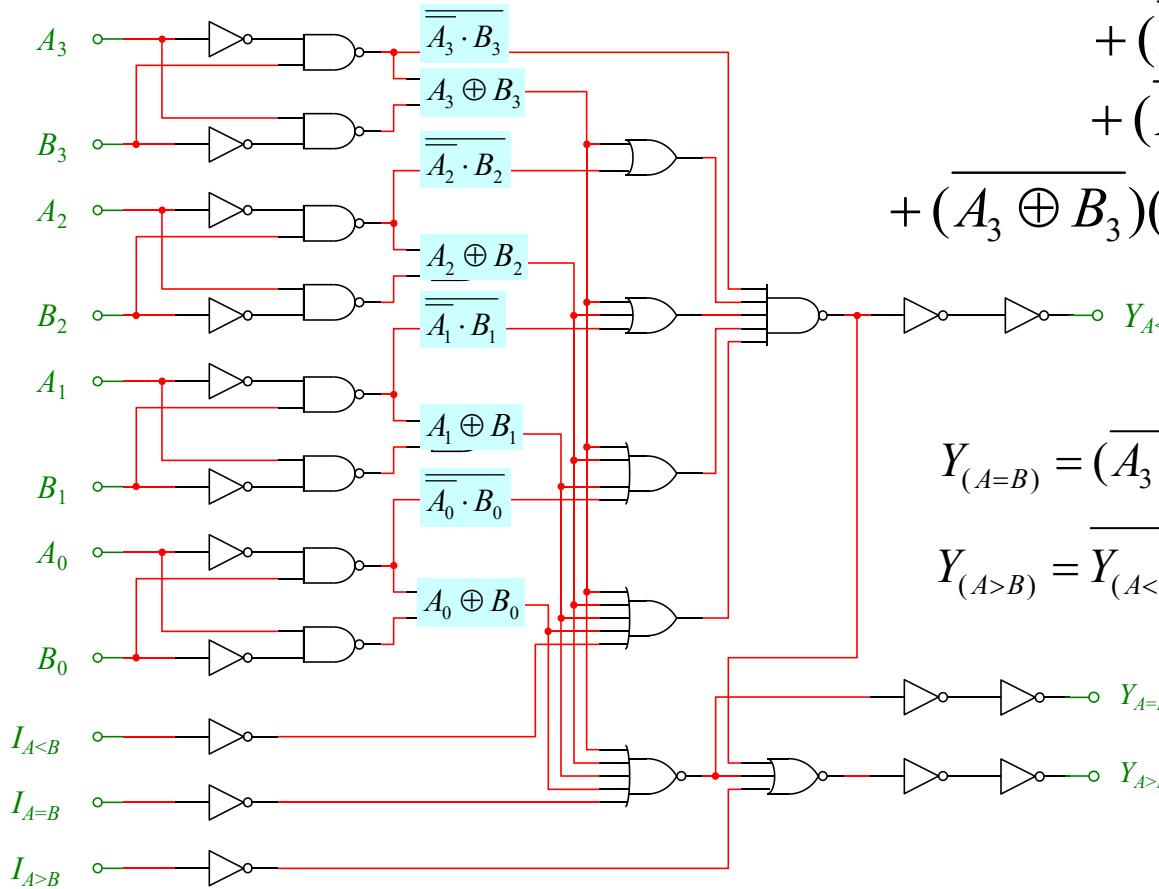
$$F_{A=B} = (A_1 = B_1) \cdot (A_0 = B_0) \cdot I_{A=B}$$

组合逻辑设计

简化：注意到 $A>B$, $A<B$ 和 $A=B$ 并不是互不相关的



- 多位比较器
—— 例如：4位数码比较器



$$\begin{aligned} Y_{(A < B)} &= \overline{\overline{A_3}B_3 + (\overline{A_3} \oplus B_3)\overline{A_2}B_2} \\ &+ (\overline{A_3} \oplus B_3)(\overline{A_2} \oplus B_2)\overline{A_1}B_1 \\ &+ (\overline{A_3} \oplus B_3)(\overline{A_2} \oplus B_2)(\overline{A_1} \oplus B_1)\overline{A_0}B_0 \\ &+ (\overline{A_3} \oplus B_3)(\overline{A_2} \oplus B_2)(\overline{A_1} \oplus B_1)(\overline{A_0} \oplus B_0)I_{(A < B)} \end{aligned}$$

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

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

$$Y_{A=B}$$

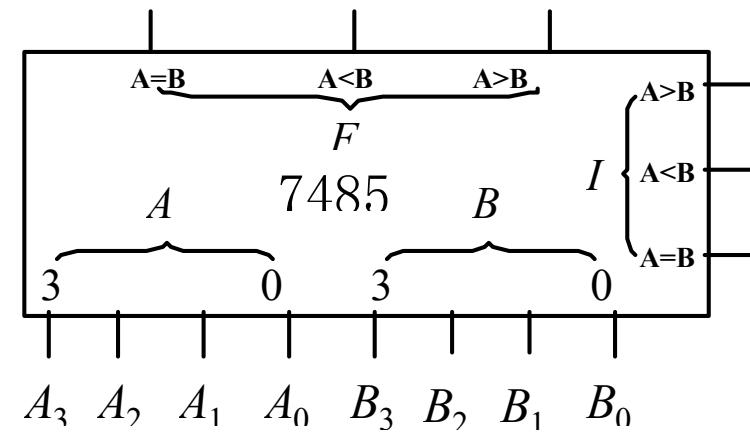
$$Y_{A>B}$$



§ 4.2.4 数值比较器

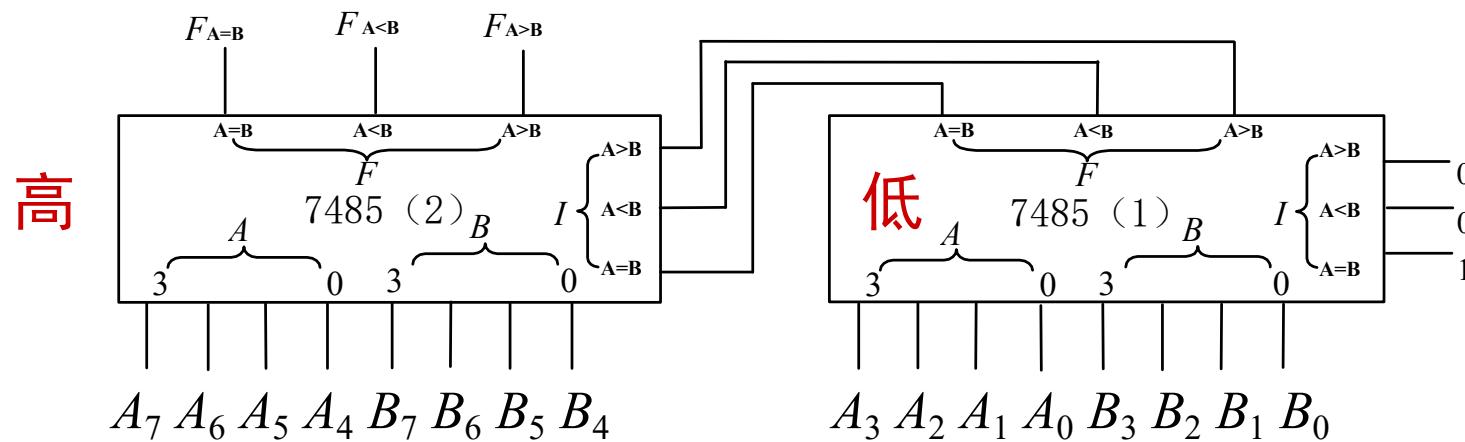
■ 数值比较器的扩展

- 例题：利用集成数值比较器7845进行
 - 比较器的位数串联扩展



§ 4.2.4 数值比较器

- 比较器的位数串联扩展
——通用的串联扩展方法



低位模块将比较结果馈入高位模块的级联输入端

§ 4.2.5 组合逻辑电路模块之加法器

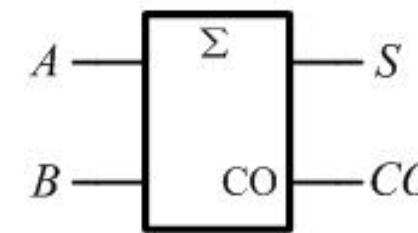
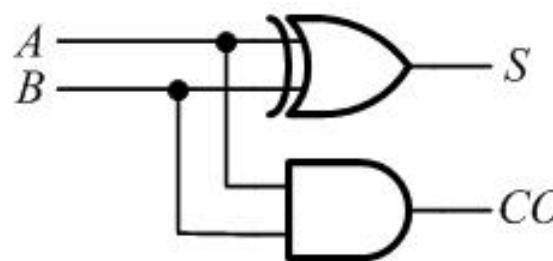
■ 加法器

➤ 半加器

✓ 仅考虑加数和被加数的运算

$$S = \overline{A}B + A\overline{B} = A \oplus B \quad CO = AB$$

输入		输出	
被加	加数	和数	进位
A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



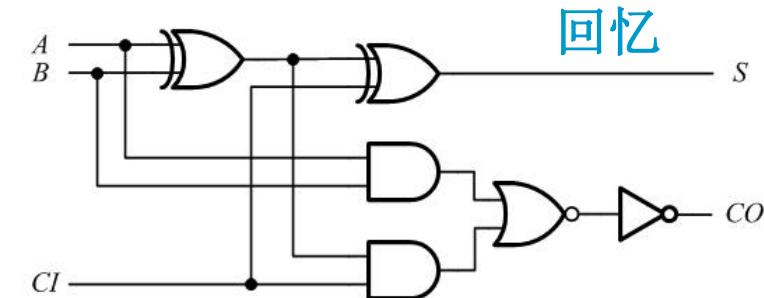
§ 4.2.5 加法器

➤ 全加器

✓ 考虑加数、被加数和相邻低位的进位的运算

全加器的真值表

输入			输出	
A	B	CI	S	CO
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



$$S = A \oplus B \oplus CI$$

$$CO = AB + (A \oplus B) \cdot CI$$

对比...

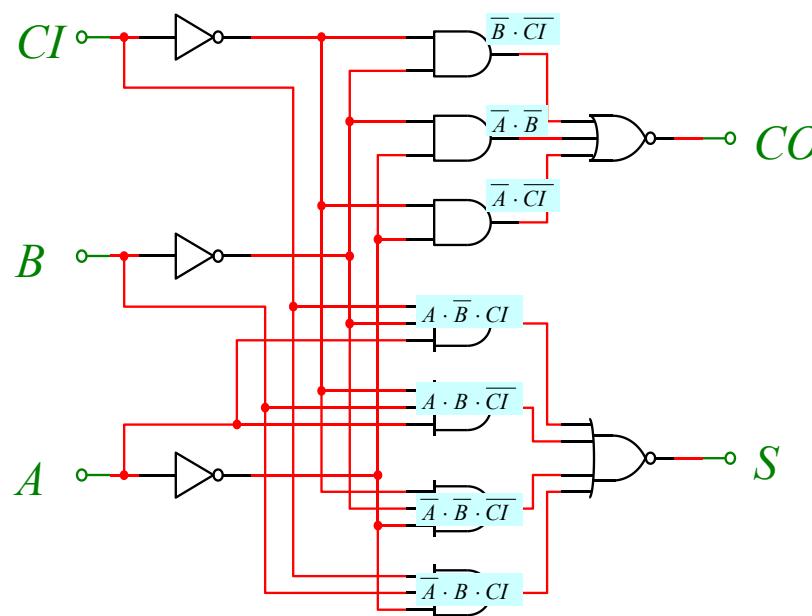
$$S = \overline{\overline{A} \cdot \overline{B} \cdot \overline{CI}} + \overline{A} \cdot B \cdot CI + A \cdot \overline{B} \cdot CI + A \cdot B \cdot \overline{CI}$$

$$CO = \overline{\overline{A} \cdot \overline{B}} + \overline{B} \cdot \overline{CI} + \overline{A} \cdot \overline{CI}$$

➤ 全加器

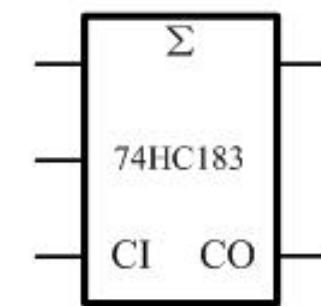
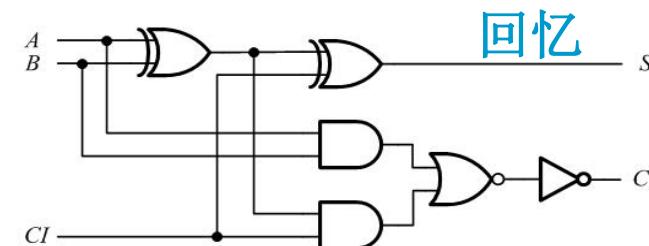
$$S = \overline{\overline{A} \cdot \overline{B} \cdot \overline{CI}} + \overline{ABC}I + A\overline{B}CI + ABC\overline{I}$$

$$CO = \overline{\overline{A} \cdot \overline{B}} + \overline{B} \cdot \overline{CI} + \overline{A} \cdot \overline{CI}$$



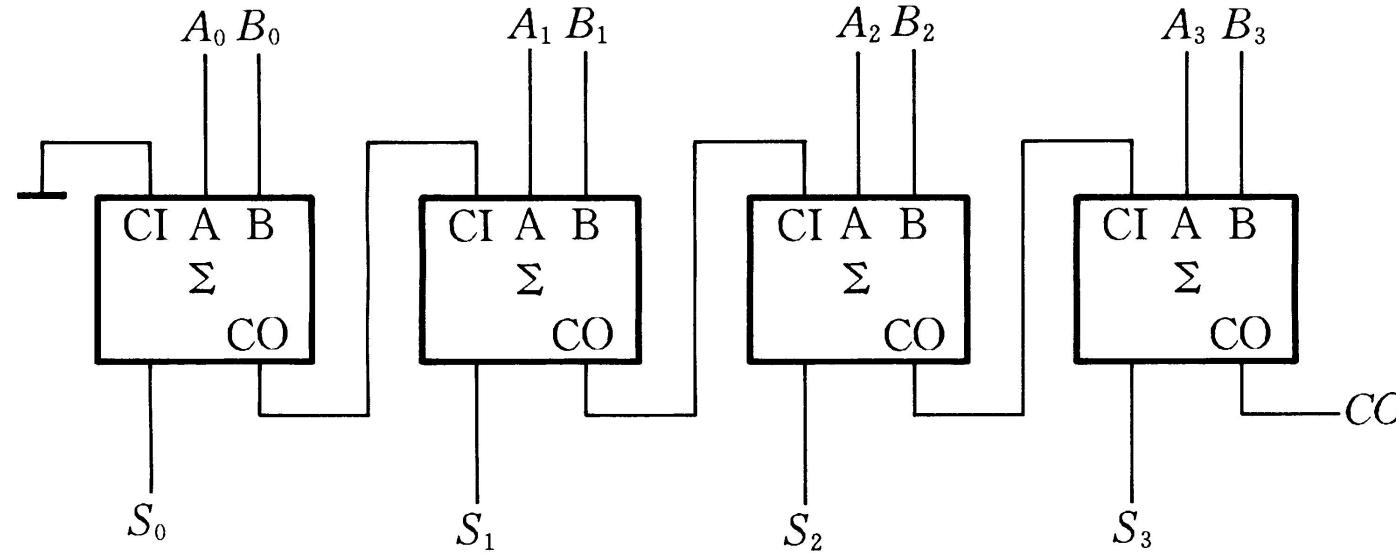
全加器的真值表

输入			输出	
A	B	CI	S	CO
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



§ 4.2.5 加法器

➤ 多位数串行进位加法器



- ✓ 缺点：运算速度慢
- ✓ 事实上：第 i 位进位信号，是 $\{A_0, A_1, \dots, A_{i-1}\}$, $\{B_0, B_1, \dots, B_{i-1}\}$ 的组合逻辑函数；
- 超前进位加法器：Carry Look-ahead



§ 4.2.5 加法器

➤ 快速进位集成4位加法器74283

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

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

令 $G_i = A_i B_i$

$$C_1 = \overline{A_1 + B_1} + \overline{A_1 B_1} \cdot \overline{A_0 + B_0} + \overline{A_1 B_1} \cdot \overline{A_0 B_0} \cdot \overline{CI}$$

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

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

例如：

$$C_0 = \overline{A_0 + B_0} + \overline{A_0 B_0} \cdot \overline{CI}$$

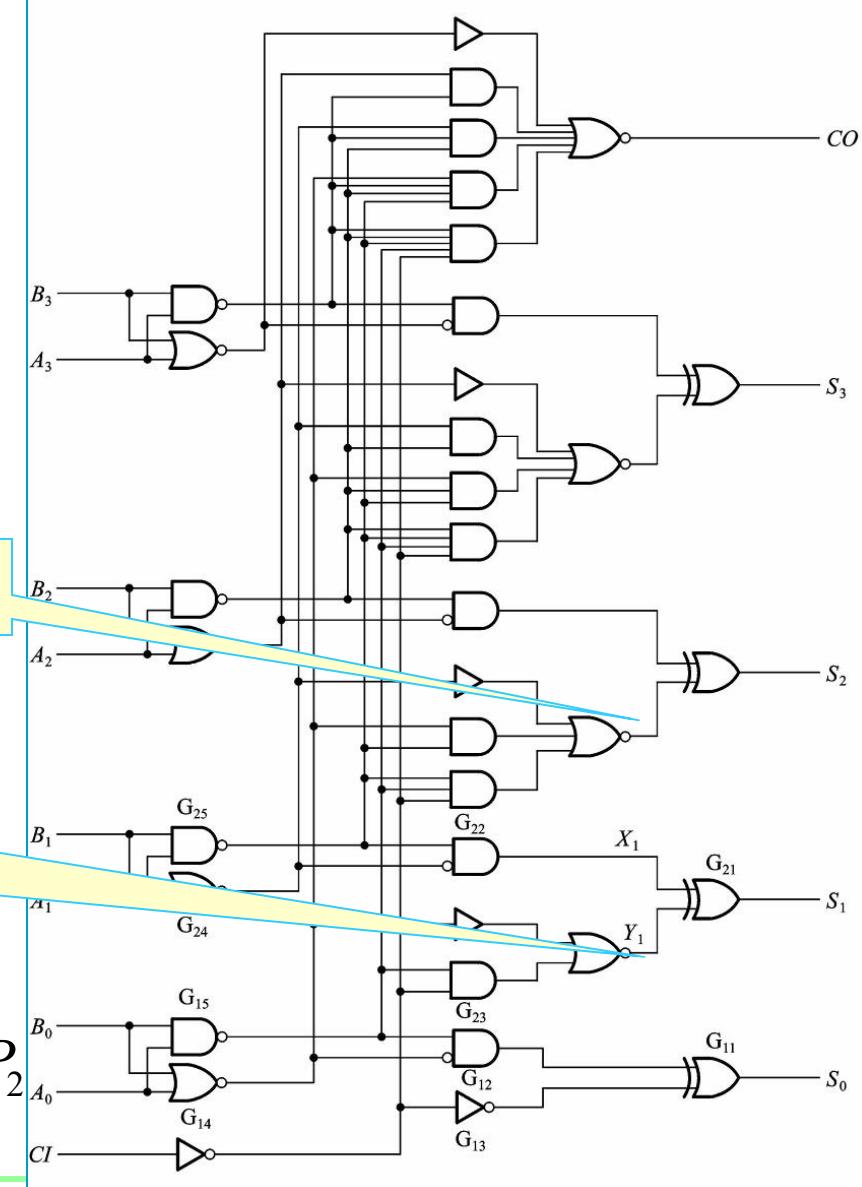
$$C_0 = G_0 + P_0 CI$$

$$C_1 = G_1 + P_1 C_0 = G_1 + P_1 G_0 + P_1 P_0 CI$$

$$C_2 = G_2 + P_2 C_1 = G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2$$

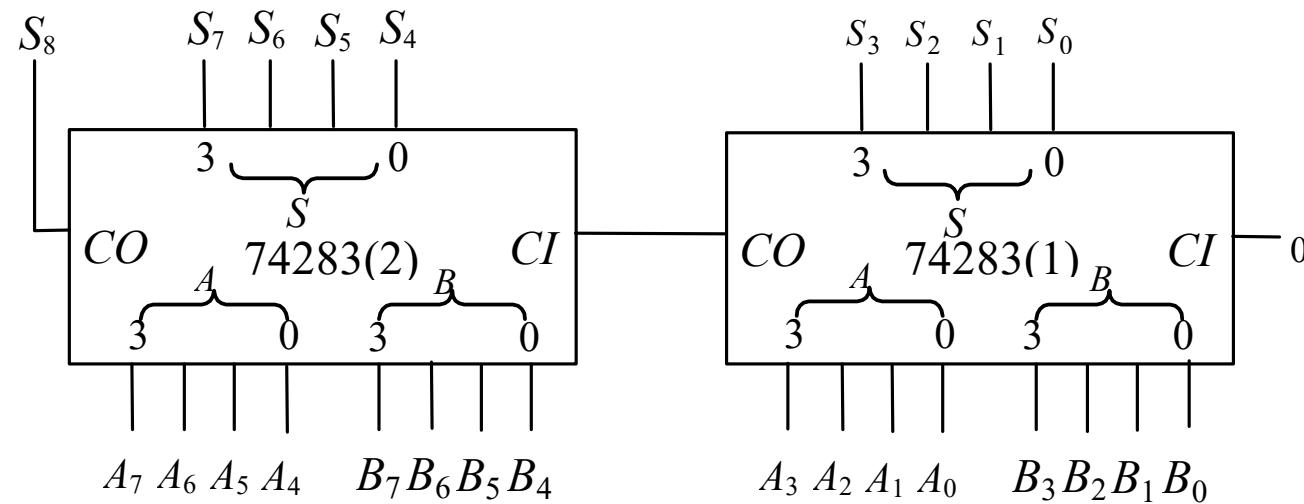
.....

$$CO = G_3 + P_3 C_2 = G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 G_0 + P_3 P_2 P_1 P_0 CI$$



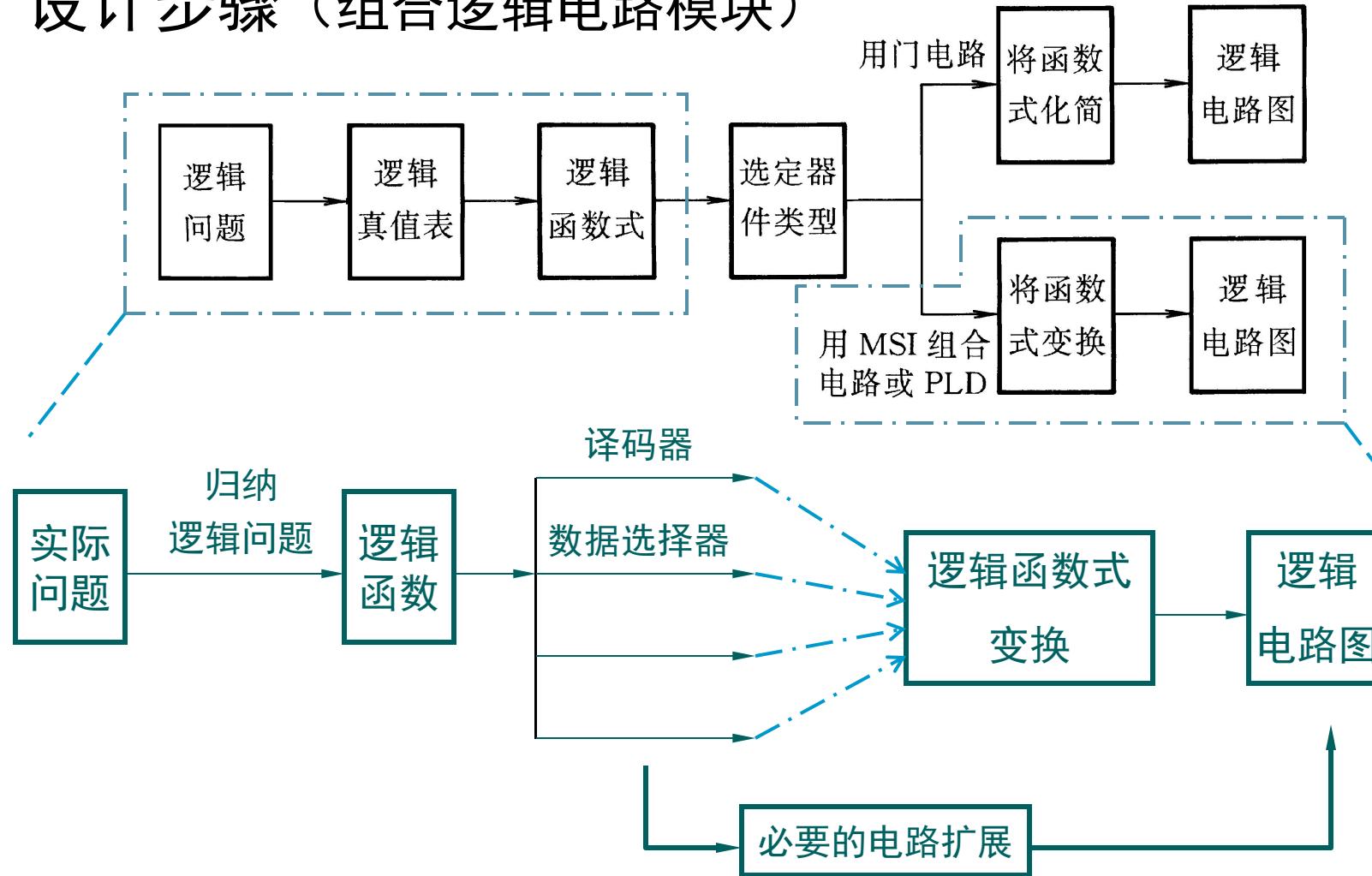
§ 4.2.5 加法器

■ 加法器的级联



归纳：组合逻辑电路设计

■ 设计步骤（组合逻辑电路模块）





第四章 组合逻辑电路

§ 4.1 组合逻辑电路的分析与设计

§ 4.2 组合逻辑电路模块及其应用

§ 4.3 组合逻辑电路中的竞争与冒险



§ 4.3 组合逻辑电路中的竞争与冒险现象

在前面的组合逻辑分析与设计中，考虑稳态的输入情况；而组合逻辑电路的工作正确性，还要考虑逻辑转换时的正确性...

- 产生竞争冒险的原因
- 冒险现象的识别
- 冒险现象的消除

§ 4.3 竞争——冒险现象

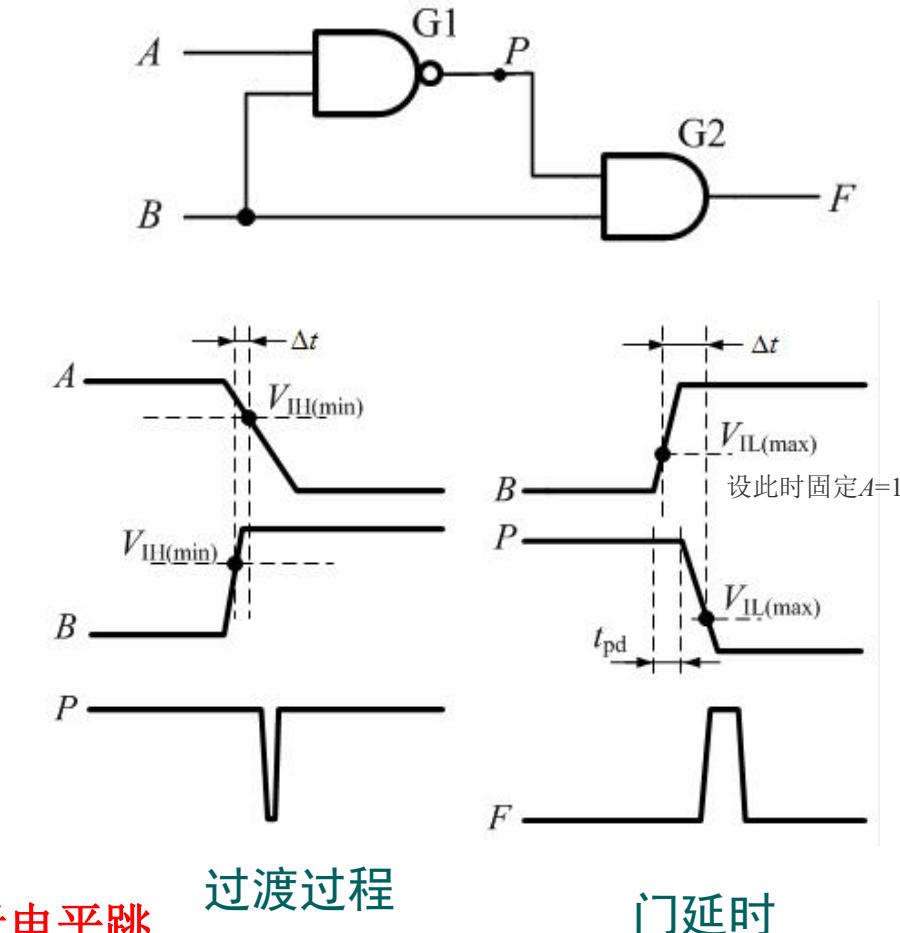
■ 产生竞争冒险的原因

➤ 例如：

$$P = \overline{AB} \quad F = P \cdot B = \overline{AB} \cdot B$$

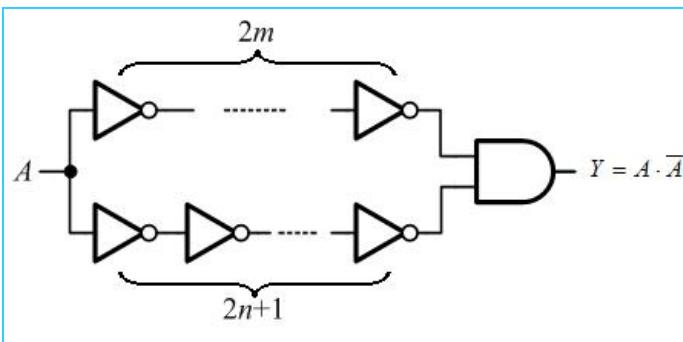
- ✓ 稳态下，正常工作；但是
- ✓ 信号变化时的**过渡过程不一致**和门电路的**传输延时**的存在，使电路产生瞬间错误输出。该现象称为**竞争-冒险**。

门电路两个输入信号“同时”向相反的逻辑电平跳变（一个从1变0，另一个从0变1）的现象叫作竞争



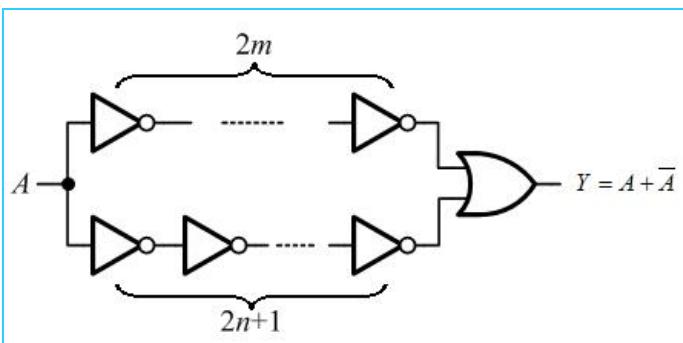
§ 4.3 竞争——冒险现象

- 冒险现象的识别 因为竞争在输出端产生尖峰脉冲的现象
- 某些逻辑变量取特定值时，表达式能转换为：



$$Y = A \bar{A}$$

✓ 存在“1”型冒险

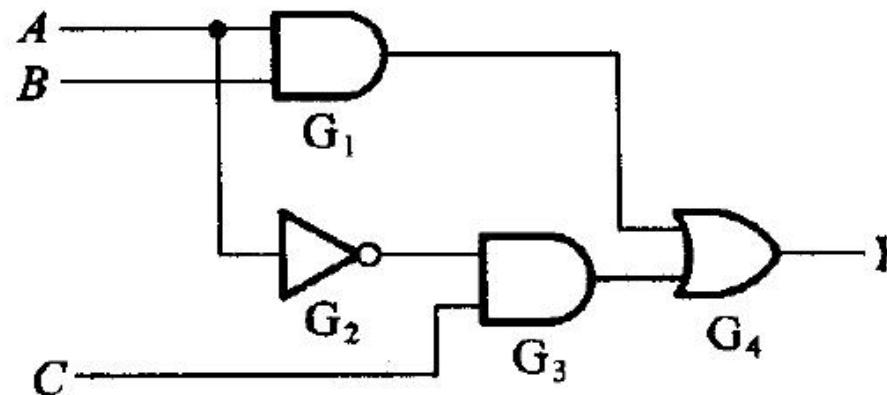


$$Y = A + \bar{A}$$

✓ 存在“0”型冒险

§ 4.3 竞争——冒险现象

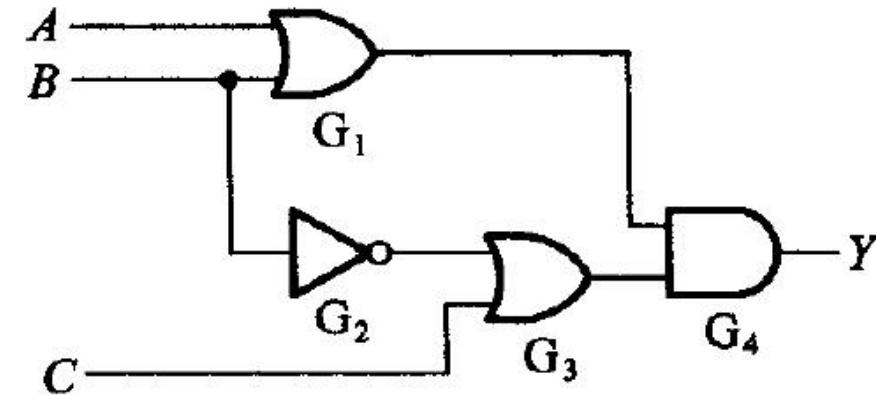
■ 冒险现象的识别



$$Y = AB + A'C$$

当 $B = C = 1$ 时, 上式将成为

$$Y = A + A'$$



$$Y = (A + B) \cdot (B' + C)$$

在 $A = C = 0$ 的条件下, 上式简化为

$$Y = B \cdot B'$$

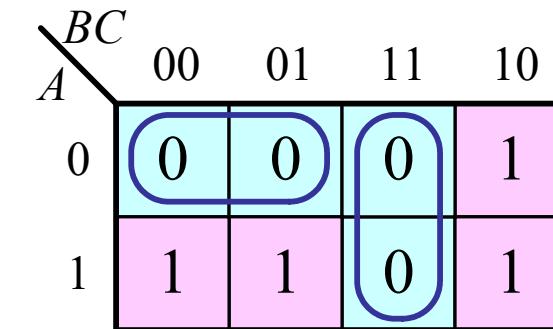
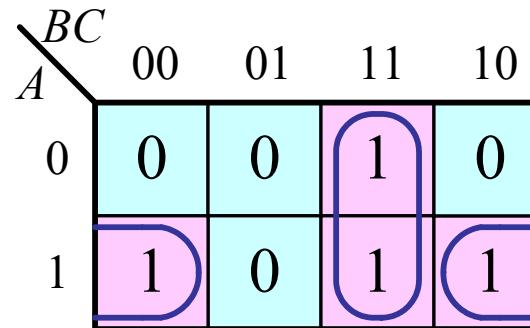
§ 4.3 竞争——冒险现象

■ 冒险现象的识别

$$F = A\bar{C} + BC \quad A=B=1 \text{ 时}, \quad F = C + \bar{C} \quad \checkmark \text{ 存在冒险}$$

$$F = (A+B)(\bar{B}+\bar{C}) \quad A=0, \quad C=1 \text{ 时}, \quad F = B \cdot \bar{B} \quad \checkmark \text{ 存在冒险}$$

\checkmark 通过卡诺图识别冒险



\checkmark 卡诺图上的圈相切，且相切处无其它圈包含



§ 4.3 竞争——冒险现象

■ 冒险现象的消除

- ✓ 增加冗余项

$$F = A\bar{C} + BC + AB$$

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

		BC	00	01	11	10
		A	0	0	1	0
0	0	0	0	1	0	
	1	1	0	1	1	

		BC	00	01	11	10
		A	0	0	0	1
0	0	0	0	0	1	
	1	1	1	0	1	

- ✓ 变换逻辑式

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

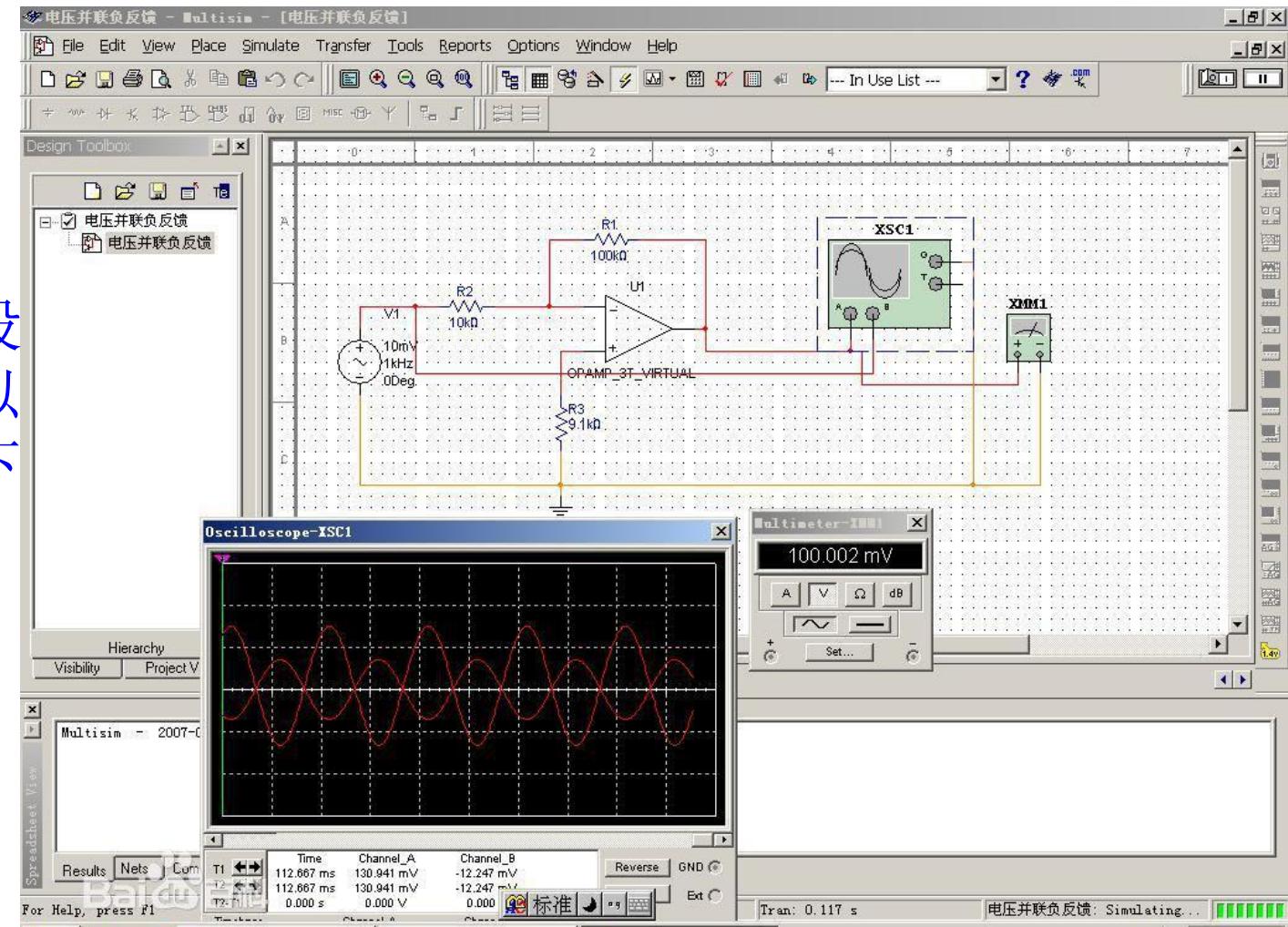
- ✓ 增加选通信号

- ✓ 增加输出滤波电容



补充：用Multisim分析组合逻辑电路

- Multisim是美国国家仪器有限公司推出的以Windows为基础的仿真工具，适用于板级的模拟/数字电路板的设计工作。它包含了电路原理图的图形输入、电路硬件描述语言输入方式，具有丰富的仿真分析能力。



有兴趣参加电子设计竞赛的同学可以课余时间学习一下



本章小结

- 组合逻辑电路的特点：输出只决定于该时刻各输入组合，与电路的原状态无关，电路中无记忆单元，没有反馈通路。
- 分析步骤：写出各输出端的逻辑表达式→化简和变换逻辑表达式→列出真值表→确定功能；
- 设计步骤：根据设计要求列出真值表→写出逻辑表达式（或填写卡诺图）→逻辑化简和变换→画出逻辑图。
- 用译码器和数据选择器实现组合逻辑功能。
- 常用的中规模组合逻辑模块包括：编码器、译码器、数据选择器、数值比较器、加法器…，为了增加使用的灵活性和便于功能扩展，设置了输入、输出使能端或扩展端。
- 组合电路中存在竞争和冒险现象，可通过表达式和卡诺图识别冒险现象，并应采取措施消除冒险现象。



第四章 课程作业

阎石 老师 第五版

- 4.3;
- 4.6;
- 4.8; 4.10
- 4.12; 4.17
- 4.28;
- 4.32 ;