

数字电子技术基础

By NankeDream

简单归纳课本，加上王红老师的课程中讲到的

O、绪论

1. 数字量和模拟量

- **数字量**: 在时间上和数量上都是离散的、不连续的，存在一个最小单位
- 模拟量：数字量以外的物理量

Continuous versus Discrete

- Which are “continuous”?

Color light Cars Sound Height and weight Dogs Electric current and voltage

- 数字电路和模拟电路

工作信号、研究的对象、分析/设计方法以及所用的数学工具都有显著的不同

2. 什么是电子技术

是研究电子器件及电子器件应用的一门学科

- 电子器件：通过控制器件中电子的运动而进行工作
- 电子技术的发展：集成电路、摩尔定律

3. 电子电路的作用

- 处理信息 能量转换
- 模拟电路：用连续的模拟电压/流指来表示信息
- **数字电路**：用一个连续的电压序列来表示信息

Structure 结构

- hierarchical design
- limited complexity at each level
- reusable building blocks

复用：可重复使用的构建模块

Interfaces 接口

- Key elements of system engineering
- Isolate technologies, allow evolution
- Major abstraction mechanism

What makes a good system design?

- minimal mechanism, maximal function
- reliable in a wide range of environments
- accommodates future technical improvements

4. Our plan

- Understand how things work, bottom-up
- Encapsulate(封装) our understanding using appropriate abstractions
- Study organizational principles: abstractions, interfaces, API(Application Programming Interface)
- Roll up our sleeves and design at each level of hierarchy
- Learn engineering tricks
 - History
 - Systematic approaches
 - Algorithms
 - Diagnose, fix, and avoid bugs

一、信息与编码

理想的离散是不存在的

1. What is "Information"?

(1) information:

- Knowledge, communicated or received concerning a particular fact or circumstance.

Information resolves uncertainty

- Information is simply that which cannot be predicted. The less predictable a message is the more information it conveys!

(2) Quantifying Information

Suppose you are faced with N equally probable choices, and I give you a fact that narrows it down to M choices. Then I have given you

- $\log_2(N/M)$ bits of information

(3) Encoding 编码

- Encoding describes the process of assigning representations to information
- Choosing an appropriate and efficient encoding is a real engineering challenge
- Impacts design at many levels
 - Mechanism (devices.# of components used)
 - Efficiency (bits used)
 - Reliability (noise)
 - Security (encryption)

example: 2011010909

- 数制：表述数量的规则
- 码制：表示事物的规则

(4) 我们常用到的：

十进制 D(Decimal)、二进制 B(Binary)、八进制 O(Octal)、十六进制 H(Hexadecimal)

2. 转换

(1) 二—十转换

将二进制数展开后按十进制相加，即可得到十进制数

例

$$\begin{aligned} (1011.01)_2 &= 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 + 0 \times 2^{-1} + 1 \times 2^{-2} \\ &= (11.25)_{10} \end{aligned}$$

(2) 十—二转换

对于整数部分，将十进制整数不断除以二直到得到0，可依次得到余数（0 or 1），将得到的余数从右向左书写，即可得到二进制数。

例

$$\begin{array}{r|l} 2 & 173 \\ 2 & 86 \\ 2 & 43 \\ 2 & 21 \\ 2 & 10 \\ 2 & 5 \\ 2 & 2 \\ 2 & 1 \\ \hline & 0 \end{array} \quad \begin{array}{l} \text{余数} = 1 = k_0 \\ \text{余数} = 0 = k_1 \\ \text{余数} = 1 = k_2 \\ \text{余数} = 1 = k_3 \\ \text{余数} = 0 = k_4 \\ \text{余数} = 1 = k_5 \\ \text{余数} = 0 = k_6 \\ \text{余数} = 1 = k_7 \end{array}$$

$$\text{故 } (173)_{10} = (10101101)_2.$$

对于小数部分，将十进制小数不断乘以二，直到得到1，可依次得到整数部分（0 or 1），将得到的整数部分依次排列，即得到二进制的小数。

例

$$\begin{array}{r}
 0.8125 \\
 \times \quad \quad 2 \\
 \hline
 1.6250 \quad \text{整数部分} = 1 = k_{-1}
 \end{array}$$

$$\begin{array}{r}
 0.6250 \\
 \times \quad \quad 2 \\
 \hline
 1.2500 \quad \text{整数部分} = 1 = k_{-2}
 \end{array}$$

$$\begin{array}{r}
 0.2500 \\
 \times \quad \quad 2 \\
 \hline
 0.5000 \quad \text{整数部分} = 0 = k_{-3}
 \end{array}$$

$$\begin{array}{r}
 0.5000 \\
 \times \quad \quad 2 \\
 \hline
 1.0000 \quad \text{整数部分} = 1 = k_{-4}
 \end{array}$$

故 $(0.8125)_{10} = (0.1101)_2$

(3) 二—十六转换

将整数部分与小数部分分别拆分为四个0 or 1一组的数据组，其中若整数部分的位数不是4的倍数，则高位补0，若小数部分的位数不是4的倍数，则低位补0。将每组二进制数用对应的十六进制数代替，可得到一个十六进制数据。

例

例如，将 $(01011110.10110010)_2$ 化为十六进制数时可得

$$\begin{array}{cccc}
 (0101 & 1110. & 1011 & 0010)_2 \\
 \downarrow & \downarrow & \downarrow & \downarrow \\
 = (5 & E. & B & 2)_{16}
 \end{array}$$

(4) 十六—二转换

为二—十六转换的逆过程。

例

例如，将 $(8FA.C6)_{16}$ 化为二进制数时得到

$$\begin{array}{ccccc}
 (8 & F & A. & C & 6)_{16} \\
 \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\
 = (1000 & 1111 & 1010. & 1100 & 0110)_2
 \end{array}$$

(5) 八—二与二—八转换

与十六进制的转换类似，将四位一组变为三位一组，首尾补0。

例

例如,若将 $(011110.010111)_2$ 化为八进制数,则得到

$$\begin{array}{cccc} 011 & 110. & 010 & 111 \\ \downarrow & \downarrow & \downarrow & \downarrow \\ (3 & 6. & 2 & 7)_8 \end{array}$$

反之,若将八进制数转换为二进制数,则只要将八进制数的每一位代之以等值的3位二进制数即可。例如,将 $(52.43)_8$ 转换为二进制数时,得到

$$\begin{array}{cccc} 5 & 2. & 4 & 3 \\ \downarrow & \downarrow & \downarrow & \downarrow \\ (101 & 010. & 100 & 011)_2 \end{array}$$

3. 二进制的算数运算

(1) 算数运算

加法运算

$$\begin{array}{r} 1001 \\ + 0101 \\ \hline 1110 \end{array}$$

减法运算

$$\begin{array}{r} 1001 \\ - 0101 \\ \hline 0100 \end{array}$$

乘法运算

$$\begin{array}{r} 1001 \\ \times 0101 \\ \hline 1001 \\ 0000 \\ 1001 \\ 0000 \\ \hline 0101101 \end{array}$$

除法运算

$$\begin{array}{r} 1.11\cdots \\ \hline 0101) 1001 \\ 1001 \\ \hline 0101 \\ 0101 \\ \hline 0110 \\ 0101 \\ \hline 0010 \end{array}$$

二进制数的正负号也是用0/1表示的。

在定点运算中,最高位为符号位(0为正,1为负)

如 $+5 = 0\ 0101$ $-5 = 1\ 0101$

$+5 + -5 = 0\ 0101 + 1\ 0101 = ?$?

(2) 反码、补码和补码运算

- $1011 - 0111 = 1011 + 1001 = (1)0100$

其中，1001为-0111的补码，括号里的1为进位，需舍去，对于4位二进制数，基数为16(10000)，所以1001(9)恰好是-0111(-7)对模16的补码。

对于有效数字(不包括符号位)为n位的二进制数N，它的补码(N)comp表示方法为：

$$(N)_{\text{COMP}} = \begin{cases} N & (\text{当 } N \text{ 为正数}) \\ 2^n - N & (\text{当 } N \text{ 为负数}) \end{cases}$$

即正数(符号位为0)的补码与原码相同，负数(符号位为1)的补码为 $2^n - N$ 。符号位保持不变。不过这样仍然要做减法，对于更简便的方法：当N为负数时，先求出N的反码(N)inv，即保持符号位不变，将数值位逐位求反，得到反码，然后将反码加1，即可得到二进制负数的补码。

$$(N)_{\text{COMP}} = (N)_{\text{INV}} + 1$$

例：

原 码	反 码	补 码
00011010	00011010	00011010
10011010	11100101	11100110
00101101	00101101	00101101
10101101	11010010	11010011

- 对于小数

取反加1是加在最右边，不是加数值1。

- 运算时，若需要添加符号位，则需留足运算范围，4位：-8~7；5位：-16~15；6位：-32~31

在编码取值合适的情况下，两个加数的符号位和来自最高位数位的进位相加。结果就是和的符号位。

$$\begin{array}{r}
 +13 \\
 +10 \\
 \hline
 +23
 \end{array}
 \quad
 \begin{array}{r}
 0 \ 01101 \\
 0 \ 01010 \\
 \hline
 0 \ 10111
 \end{array}
 \quad
 \begin{array}{r}
 +13 \\
 -10 \\
 \hline
 +3
 \end{array}
 \quad
 \begin{array}{r}
 0 \ 01101 \\
 1 \ 10110 \\
 \hline
 (1) 0 \ 00011
 \end{array}$$

$$\begin{array}{r}
 -13 \\
 +10 \\
 \hline
 -3
 \end{array}
 \quad
 \begin{array}{r}
 1 \ 10011 \\
 0 \ 01010 \\
 \hline
 1 \ 11101
 \end{array}
 \quad
 \begin{array}{r}
 -13 \\
 -10 \\
 \hline
 -23
 \end{array}
 \quad
 \begin{array}{r}
 1 \ 10011 \\
 1 \ 10110 \\
 \hline
 (1) 1 \ 01001
 \end{array}$$

4. 码制

用不同数码表示不同事物时遵循的规则，例如：学号、身份证号、车牌号……

- 目前，数字电路中都采用二进制
- 表示数量时称二进制
- 表示事物时称二值逻辑

(1) Fixed-length encodings 等长编码

- If all choices are equally likely (or we have no reason to expect otherwise), then a fixed-length code is often used. such a code will use at least enough bits to represent the information content.
- 8421码(BCD代码)、余3码、2421码、5211码、余3循环码

(2) 格雷码

每一位的状态变化都按一定顺序循环。编码顺序一次变化，按表中顺序变化时，相邻代码只有一位改变状态。

编 码 顺 序	二进制代码	格 雷 码
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000

应用：减少编码变化过程中产生的“噪声”，变化快。

(3) When choices are not equally probable 变长编码 Variable-length encodings

When the choices have different probabilities(π_i), you get more information when learning of an unlikely choice than when learning of a likely choice

出现频率越高的字母用短的编码表示，反之用越长的编码表示

Use shorter bit sequences for high probability choices, longer sequences for less probable choices

(4) 哈夫曼编码

如何进行哈夫曼编码？

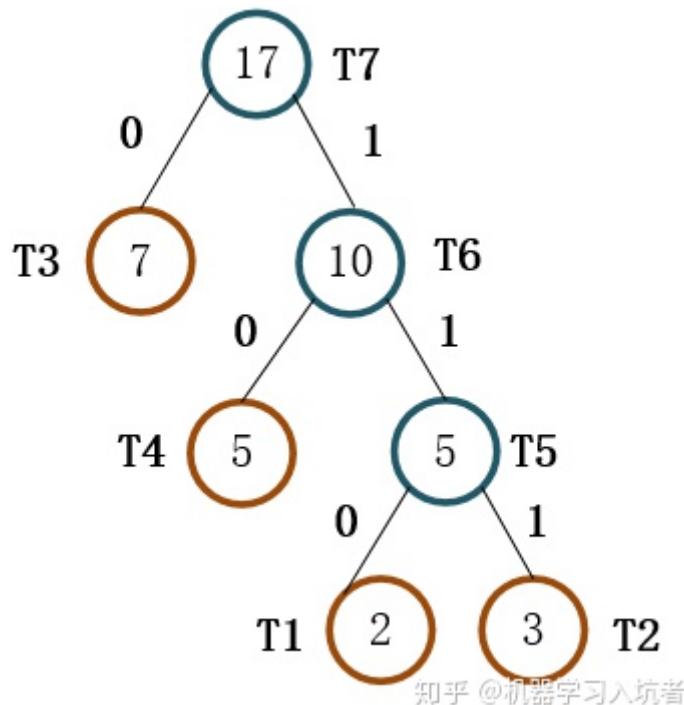
使用需要传送的字符构造字符集 $C = \{c_1, c_2, \dots, c_n\}$ ，并根据字符出现的频率构建概率集 $W = \{w_1, w_2, \dots, w_n\}$ 。哈夫曼编码的流程如下：

- 将字符集 C 作为叶子节点
- 将频率集 W 作为叶子节点的权值
- 使用 C 和 W 构造哈夫曼树
- 对哈夫曼树的所有分支，左子树分支编码为0，右子树分支编码为1

通过上述流程，即完成了哈夫曼编码。

例

设定字符集为 $C = \{T1, T2, T3, T4\}$ ，对应的频率集为 $W = \{2, 3, 7, 5\}$ ，可以构造出下面的哈夫曼树



那么， $T1 = 110$ $T2 = 111$ $T3 = 0$ $T4 = 10$

二、逻辑代数基础

基本逻辑运算、基本公式、表示方法、化简

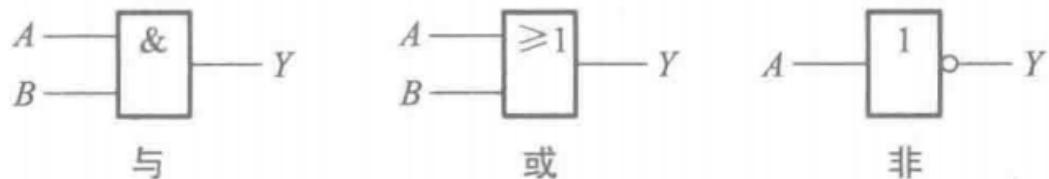
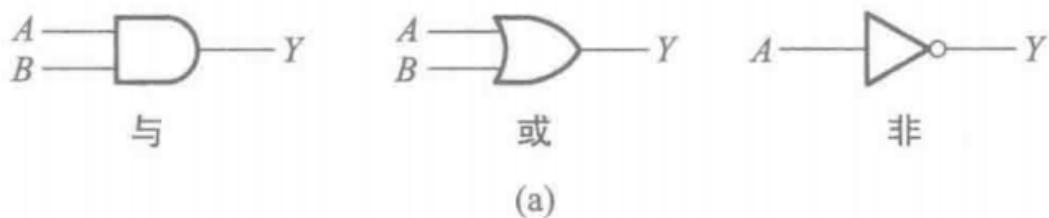
1. 概述

逻辑：事物的因果关系

逻辑运算的数学基础：逻辑代数（布尔）在二值逻辑中的变量取值：0/1

- 1854: George Boole shows that logic is math, not just philosophy!
- Boolean algebra: the mathematics of binary values
- Despite existence of relays and introduction of vacuum tube in 1906. Digital electronics did not emerge for thirty years!
- Claude Shannon notices similarities between Boolean algebra and electronic telephone switches
- Shannon's 1937 MIT Master's Thesis introduces the world to binary digital electronics

2. 逻辑代数中的三种基本运算



(1) 与 (AND)

- 条件同时具备，结果发生
- $Y = A \text{ AND } B = A \& B = A * B = AB$

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

(2) 或 (OR)

- 条件之一具备，结果发生
- $Y = A \text{ OR } B = A + B$

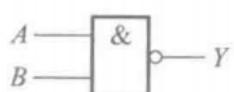
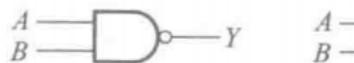
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

(3) 非(NOT)

- 条件不具备，结果发生
- $Y = A' = \text{NOT } A$

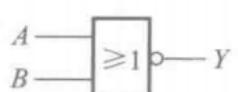
A	Y
0	1
1	0

3. 几种常用的复合逻辑运算



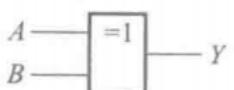
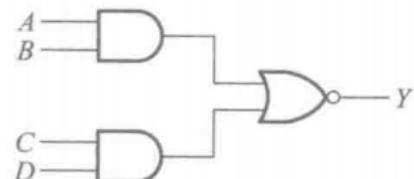
与非

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



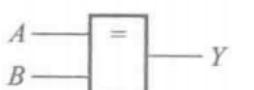
或非

$$Y = (A + B)'$$



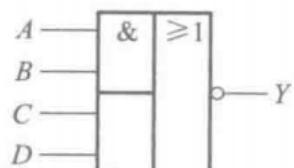
异或

$$Y = A \oplus B$$



同或

$$Y = A \odot B$$



与或非

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

(1) 与非

- 先与后非

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

(2) 或非

- 先或后非

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

(3) 与或非

- 先与在或后非

A	B	Y1	C	D	Y2	Y
0	0	0	0	0	0	1
0	0	0	0	1	0	1
0	0	0	1	0	0	1
0	0	0	1	1	1	0
0	1	0	0	0	0	1
0	1	0	0	1	0	1
0	1	0	1	0	0	1
0	1	0	1	1	1	0
1	0	0	0	0	0	1
1	0	0	0	1	0	1
1	0	0	1	1	1	0
1	1	1	0	0	0	0
1	1	1	0	1	0	0
1	1	1	1	0	0	0
1	1	1	1	1	1	0

(4) 异或

- 异就输出 1
- $A' * B + A * B'$

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

(5) 同或

- 同就输出 1
- $A' * B' + A * B$

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

4. 逻辑代数的基本公式

(1) 基本公式

序号	公 式	序号	公 式
1	$0 \cdot A = 0$	10	$1' = 0; 0' = 1$
2	$1 \cdot A = A$	11	$1 + A = 1$
3	$A \cdot A = A$	12	$0 + A = A$
4	$A \cdot A' = 0$	13	$A + A = A$
5	$A \cdot B = B \cdot A$	14	$A + A' = 1$
6	$A \cdot (B \cdot C) = (A \cdot B) \cdot C$	15	$A + B = B + A$
7	$A \cdot (B+C) = A \cdot B + A \cdot C$	16	$A + (B+C) = (A+B) + C$
8	$(A \cdot B)' = A' + B'$	17	$A + B \cdot C = (A+B) \cdot (A+C)$
9	$(A')' = A$	18	$(A+B)' = A' \cdot B'$

Important:

- $(A * B)' = A' + B'$
- $A + B * C = (A+B) * (A+C)$
- $(A + B)' = A' * B'$

公式 (17) 的证明:

$$\begin{aligned} \text{右} &= (A + B) * (A + C) = A + A * B + A * C + B * C \\ &= A (1 + B + C) + B * C \\ &= A + B * C = \text{左} \end{aligned}$$

德·摩根定理: (公式8, 18)

(2) 常用公式

序号	公 式
21	$A + A \cdot B = A$
22	$A + A' \cdot B = A + B$
23	$A \cdot B + A \cdot B' = A$
24	$A \cdot (A+B) = A$
25	$A \cdot B + A' \cdot C + B \cdot C = A \cdot B + A' \cdot C$ $A \cdot B + A' \cdot C + BCD = A \cdot B + A' \cdot C$
26	$A \cdot (A \cdot B)' = A \cdot B'; A' \cdot (AB)' = A'$

- 公式22: $A + A' * B = (A + A') * (A + B) = 1 * (A + B) = A + B$
- 公式25: $A * B + A' * C + (A + A') * B * C = A * B + A * B * C + A' * C + A' * B * C = A * B + A * C'$
- 公式26: $A * (A * B)' = A * (A' + B') = A * B'$

$$A' * (A * B)' = (A + A * B)' = A'$$

5. 逻辑代数的基本定理

(1) 代入定理

- 在任何一个包含A的逻辑等式中，若以另外一个逻辑式代入式中A的位置，则等式依然成立。
- 相当于数学中的“换元”，可以使逻辑公式千变万化。

(2) 反演定理

- 对于任意一个逻辑式Y，若将其中所有的“*”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，则得到的结果就是Y'。
- 反演定理为求取已知逻辑式的反逻辑式提供了方便。
- 仍需遵守“先括号，然后乘，最后加”的运算优先顺序。
- 不属于单个变量上的反号应保留不变。

应用举例

$$Y = A * (B + C) + C * D$$

$$Y' = (A' + B' * C') * (C' + D')$$

- 反演定理在数字电路实现过程中意义不大

(3) 对偶定理

- 若两逻辑式相等，则它们的对偶式也相等，这既是对偶定理。
- 对偶式

对于任何一个逻辑式，若将其中所有的“*”换成“+”，“+”换成“*”，0换成1，1换成0，则得到一个新的逻辑式Yd，这个Yd就称为Y的对偶式，互为对偶式。

$$\bullet Y = A * (B + C) \quad Y_d = A + B * C$$

$$Y = (A * B + C * D)' \quad Y_d = ((A + B) * (C + D))'$$

$$Y = A * B + (C + D)' \quad Y_d = (A + B) * (C * D)'$$

6. 逻辑函数及其描述方法

(1) 逻辑函数

- 逻辑函数 $Y = F(A, B, C, \dots)$

若以逻辑变量为输入，运算结果为输出，则输入变量值确定以后，输出的取值也随之而定，输入输出之间是一种函数关系。

(2) 逻辑函数的描述方法

- 真值表、逻辑式、逻辑图、波形图、卡诺图、EDA中的硬件描述语言。
- 各种方法之间可以相互转换（同一种逻辑关系在不同情况下的编码方式）。

i 描述方法

真值表

输入			输出
A	B	C	Y
0	0	0	0
0	0	1	0
0	1	0	0

- 输入变量要遍历所有可能的输入变量的取值组合，输出变量可以是多输出。

逻辑式

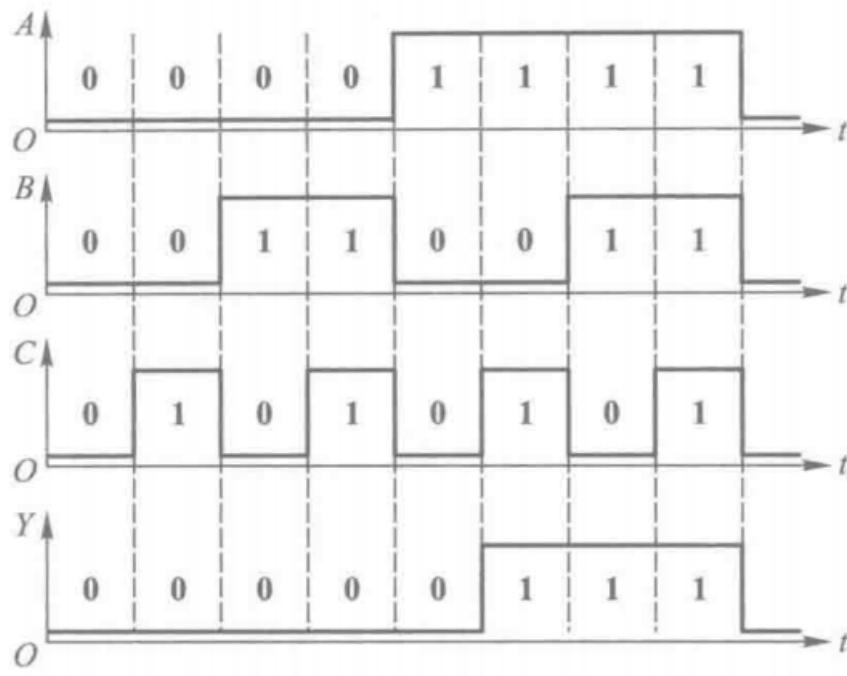
- 将输入输出之间的逻辑关系用与或非的运算式表示就得到逻辑式。

逻辑图

- 用逻辑图形符号表示逻辑运算关系，与逻辑电路的实现相对应。

波形图（时序图）

- 将输入变量所有取值可能与对应输出时间按时间顺序排列起来画成时间波形。



ii 相互转换

- 真值表——逻辑式
 - 找出真值表中使逻辑函数 $Y = 1$ 的那些输入变量取值的组合。
 - 每组输入变量取值的组合对应一个乘积项，其中取值为1的写入原变量，取值为0的写入反变量。
 - 将这些乘积项相加，即得到Y的逻辑式
- 例

A	B	C	Y
0	0	0	$1 \cdots \cdots \rightarrow A'B'C' = 1$
0	0	1	0
0	1	0	0
0	1	1	$1 \cdots \cdots \rightarrow A'BC = 1$
1	0	0	0
1	0	1	$1 \cdots \cdots \rightarrow AB'C = 1$
1	1	0	$1 \cdots \cdots \rightarrow ABC' = 1$
1	1	1	0

在输入变量取值为一下四种情况时，Y 等于 1：、

$A = 0, B = 0, C = 0; A = 0, B = 1, C = 1; A = 1, B = 0, C = 1; A = 1, B = 1, C = 0;$

因此 $Y = A' * B' * C' + A' * B * C + A * B' * C + A * B * C'$

- 逻辑式——逻辑图

- 若题目有要求，则需要对逻辑式进行等效处理。
- 用图形符号代替逻辑式中的运算逻辑符。
- 从输入到输出逐级写出每个图形符号对应的逻辑运算式。

- 波形图——真值表

要求波形图是完整的

(3) 逻辑函数的两种标准形式

最小项m

- m 是乘积项
- 包含 n 个因子
- n 个变量均以原变量和反变量的形式在 m 中出现一次
- 对于 n 变量函数有 2^n 个最小项

例

- 两变量A、B的最小项

$A' * B', A' * B, A * B', A * B$

三变量最小项的编号表

最小项	使最小项为 1 的变量取值			对应的十进制数	编号
	A	B	C		
$A' B' C'$	0	0	0	0	m_0
$A' B' C$	0	0	1	1	m_1
$A' B C'$	0	1	0	2	m_2
$A' B C$	0	1	1	3	m_3
$A B' C'$	1	0	0	4	m_4
$A B' C$	1	0	1	5	m_5
$A B C'$	1	1	0	6	m_6
$A B C$	1	1	1	7	m_7

性质

- 在输入变量的任意取值下，有且仅有一个最小项的值为1。
- 全体最小项之和为1。
- 任何两个最小项之积为0。
- 两个相邻最小项之和可以合并，消去一对因子，只留下公共因子。（相邻：仅有一个变量不同的最小项，如： $A' * B * C'$ 与 $A' * B * C$ ）

逻辑函数最小项之和的形式：

- 例： $Y(A, B, C) = A * B * C' + B * C = A * B * C' + B * C * (A + A') = A * B * C' + A * B * C + A' * B * C = \Sigma G(m(3, 6, 7))$

最大项M

- M 是相加项
- 包含n个因子
- n个变量均以原变量和反变量的形式在M中出现一次

例

- 两变量A、B的最大项
- $A' + B'$, $A' + B$, $A + B'$, $A + B$

三变量最大项的编号表

最大项	使最大项为 0 的变量取值			对应的十进制数	编号
	A	B	C		
$A + B + C$	0	0	0	0	M_0
$A + B + C'$	0	0	1	1	M_1
$A + B' + C$	0	1	0	2	M_2
$A + B' + C'$	0	1	1	3	M_3
$A' + B + C$	1	0	0	4	M_4
$A' + B + C'$	1	0	1	5	M_5
$A' + B' + C$	1	1	0	6	M_6
$A' + B' + C'$	1	1	1	7	M_7

性质

- 在输入变量的任何取值下必有一个最大项，而且只有一个最大项的值为0；
- 全体最大项之积为0；
- 任意两个最大项之和为1；
- 只有一个变量不同的最大项的乘积等与各相同变量之和。

最小项和最大项的关系

$$M_i = m_{i'}$$

例如： $m_0 = A' * B' * C'$ ， 则 $m_0' = (A' * B' * C')' = A + B + C = M_0$

7. 逻辑函数的化简方法

- 逻辑函数的最简形式

$$Y_1 = A * B * C + B' * C + A * C * D = (A * B + B') * C + A * C * D = (A + B') * C + A * C * D = A * C * (1 + D) + B' * C$$

$$Y_2 = A * C + B' * C$$

- 最简与或：

包含的乘积项已经最少，每个乘积项的因子也最少，称为最简的与或逻辑式

(1) 公式化简法

- 反复应用基本公式和常用公式，消去多余的乘积项和多余的因子，求得最简形式

1. 并项法：利用 $A * B + A * B' = A$

$$Y_1 = A(B'CD)' + AB'CD$$

$$Y_2 = AB' + ACD + A'B' + A'CD$$

$$Y_3 = A'BC' + AC' + B'C'$$

$$Y_4 = BC'D + BCD' + BC'D' + BCD$$

$$Y_1 = A((B'CD)' + B'CD) = A$$

$$Y_2 = A(B' + CD) + A'(B' + CD) = B' + CD$$

$$Y_3 = A'BC' + (A+B')C' = (A'B)C' + (A'B')C' = C'$$

$$Y_4 = B(C'D + CD') + B(C'D' + CD)$$

$$= B(C \oplus D) + B(C \oplus D)' = B$$

2. 吸收法：利用 $A + A * B = A$

$$Y_1 = ((A'B)' + C)ABD + AD$$

$$Y_2 = AB + ABC' + ABD + AB(C' + D')$$

$$Y_3 = A + (A'(BC)')'(A' + (B'C' + D')) + BC$$

$$Y_1 = ((A'B)' + C)B \cdot AD + AD = AD$$

$$Y_2 = AB + AB(C' + D + (C' + D')) = AB$$

$$Y_3 = (A + BC) + (A + BC)(A' + (B'C' + D')) = A + BC$$

3. 消项法：利用 $A * B + A' * C + B * C = A * B + A' * C$ 以及 $A * B + A' * C + B * C * D = A * B + A' * C$

$$Y_1 = AC + AB' + (B+C)'$$

$$Y_2 = AB'CD' + (AB')'E + A'CD'E$$

$$Y_3 = A'B'C + ABC + A'BD' + AB'D' + A'BCD' + BCD'E'$$

$$Y_1 = AC + AB' + B'C' = AC + B'C'$$

$$Y_2 = (AB')CD' + (AB')'E + (CD')(E)A' \\ = AB'CD' + (AB')'E$$

$$Y_3 = (A'B' + AB)C + (A'B + AB')D' + BCD'(A' + E') \\ = (A \oplus B)'C + (A \oplus B)D' + CD'(B(A' + E')) \\ = (A \oplus B)'C + (A \oplus B)D'$$

4. 消因子法：利用 $A + A' * B = A + B$

$$Y_1 = B' + ABC$$

$$Y_2 = AB' + B + A'B$$

$$Y_3 = AC + A'D + C'D$$

$$Y_1 = B' + ABC = B' + AC$$

$$Y_2 = AB' + B + A'B = A + B + A'B = A + B$$

$$\begin{aligned} Y_3 &= AC + A'D + C'D = AC + (A' + C')D = AC + (AC)'D \\ &= AC + D \end{aligned}$$

5. 配项法：

i 利用 $A + A = A$

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

ii 利用 $A + A' = 1$

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

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

(2) 卡诺图化简法

卡诺图

- **实质：将逻辑函数的最小项之和以图形的方式表示出来**
- 以 2^n 个小方块分别代表 n 变量的所有最小项，并将它们排列成矩阵，而且使几何位置相邻的两个最小项在逻辑上也是相邻的（只有一个变量不同），就得到表示 n 变量全部最小项的卡诺图

二到五变量最小项的卡诺图

(a) Karnaugh map for two variables A and B. The columns are labeled 0 and 1, and the rows are labeled 0 and 1. The minterms are: $m_0 = A'B'$, $m_1 = A'B$, $m_2 = AB'$, $m_3 = AB$.

(b) Karnaugh map for three variables A, B, and C. The columns are labeled 00, 01, 11, and 10. The rows are labeled 0 and 1. The minterms are: $m_0 = m_0$, $m_1 = m_1$, $m_2 = m_3$, $m_3 = m_2$, $m_4 = m_4$, $m_5 = m_5$, $m_6 = m_7$, $m_7 = m_6$.

(c) Karnaugh map for four variables A, B, C, and D. The columns are labeled 00, 01, 11, and 10. The rows are labeled 00, 01, 11, and 10. The minterms are: $m_0 = m_0$, $m_1 = m_1$, $m_2 = m_2$, $m_3 = m_3$, $m_4 = m_4$, $m_5 = m_5$, $m_6 = m_7$, $m_7 = m_6$, $m_8 = m_{12}$, $m_9 = m_{13}$, $m_{10} = m_{15}$, $m_{11} = m_{14}$.

(d) Karnaugh map for five variables A, B, C, D, and E. The columns are labeled 000, 001, 011, 010, 110, 111, 101, and 100. The rows are labeled 00, 01, 11, and 10. The minterms are: $m_0 = m_0$, $m_1 = m_1$, $m_2 = m_3$, $m_3 = m_2$, $m_4 = m_4$, $m_5 = m_5$, $m_6 = m_6$, $m_7 = m_7$, $m_8 = m_8$, $m_9 = m_9$, $m_{10} = m_{10}$, $m_{11} = m_{11}$, $m_{12} = m_{12}$, $m_{13} = m_{13}$, $m_{14} = m_{14}$, $m_{15} = m_{15}$, $m_{16} = m_{16}$, $m_{17} = m_{17}$, $m_{18} = m_{18}$, $m_{19} = m_{19}$, $m_{20} = m_{20}$, $m_{21} = m_{21}$, $m_{22} = m_{22}$, $m_{23} = m_{23}$, $m_{24} = m_{24}$, $m_{25} = m_{25}$, $m_{26} = m_{26}$, $m_{27} = m_{27}$, $m_{28} = m_{28}$, $m_{29} = m_{29}$, $m_{30} = m_{30}$, $m_{31} = m_{31}$.

- 几何相邻性，即几何位置上相邻，左右上下
- 对称相邻性，即图形中对称位置的单元是相邻的

图b是由图a以最右边为轴对称过去的，相对称的两个单元格是相邻的，同理，图c是由图b对称过去的，上下对称，图d是在图c基础上对称过去的，原本的对称关系还在，新增了一面与一面的对称搞关系，左右中间的四个也具有对称关系。

用卡诺图表示逻辑函数

- 将函数表示为最小项之和的形式 SIGEMA m_i
- 在卡诺图上与这些最小项对对应的位置上填入1，其余地方填0

例

用卡诺图表示逻辑函数

$$Y = A'B'C'D + A'BD' + ACD + AB'$$

首先将 Y 化为最小项之和的形式

$$\begin{aligned}
 Y &= A'B'C'D + A'B(C+C')D' + A(B+B')CD + AB'(C+C')(D+D') \\
 &= A'B'C'D + A'BCD' + A'BC'D' + ABCD + AB'CD + AB'CD' + AB'C'D \\
 &\quad + AB'C'D' \\
 &= m_1 + m_4 + m_6 + m_8 + m_9 + m_{10} + m_{11} + m_{15}
 \end{aligned}$$

画出四变量最小项的卡诺图，在对应于函数式中各最小项的位置上填入1，其余位置上填入0，就得到函数 Y 的卡诺图：

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

- 其实可以不用将 Y 化为最小项之和，可以直接根据原始式填卡诺图

tip: 只有0/1，千万别2

用卡诺图化简逻辑函数

- 依据：具有相邻性的最小项可合并，消去不同的因子
- 最小项的相邻性可以从图形中直观地反映出来
- 合并最小项的原则：
 - 两个相邻的最小项可合并为一项，消去一对因子
 - 四个排成矩形的相邻最小项可合并为一项，消去两对因子
 - 八个相邻最小项可合并为一项，消去三对因子
- 化简步骤：
 - 用卡诺图表示逻辑函数
 - 找出可合并的最小项
 - 合并后的项中只包含矩形中的公共因子
 - 化简后的乘积项相加（化简结果不唯一，但项数为一旦最小）
- 化简原则
 - 化简后的乘积项应包含函数式的所有最小项，即覆盖图中所有的1
 - 乘积项的数目最少，即围成的矩形最少
 - 每个乘积项因子最少，即围成的矩形最大、
 - 每个圈要有一个其专属的新鲜的独属于自己的“1”

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

(a)

	CD	AB	00	01	11	10
	00				(1)	
	01			(1)	(1)	(1)
	11		(1)	(1)		(1)
	10				(1)	

(b)

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

(c)

	CD	AB	00	01	11	10
	00		(1)		(1)	(1)
	01		(1)	(1)	(1)	(1)
	11		(1)	(1)	(1)	(1)
	10		(1)		(1)	(1)

(d)

	CD	AB	00	01	11	10
	00		(1)	(1)	(1)	(1)
	01		(1)	(1)	(1)	(1)
	11		(1)	(1)	(1)	(1)
	10		(1)		(1)	(1)

(e)

例：用卡诺图化简法将下式化简为最简与或函数式
例：

$$Y = AC' + A'C + BC' + B'C$$

画出卡诺图，并画出合并方案：

	BC	AB	00	01	11	10
	0		0	(1)	(1)	(1)
	1		(1)	(1)	0	(1)

得到结果： $Y = A * B' + A' * C + B * C'$

例：用卡诺图化简法将下式化简为最简与或逻辑式
例：

$$Y = ABC + ABD + AC'D + C'D' + AB'C + A'CD'$$

画出卡诺图，并画出合并方案：

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

得到结果： $Y = A + D'$

- 当发现卡诺图中1很多，那么我们可以把圈“1”改为圈“0”，即把“1”视为“0”，把“0”视为“1”，这样做出的结果取反（利用摩根定理）即可得到结果。

(3) Q - M法

- 卡诺图和公式法在遇到逻辑变量比较多的情况下很难操作。
- Q - M法适用于编制计算机辅助化简程序，而卡诺图法和公式法没有固定的过程，不适合计算机化简。

8. 具有关项的逻辑函数及其化简

(1) 约束项、任意项和逻辑函数式中的无关项

- 约束项：在逻辑函数中，某些输入变量的取值（取值组合）受到实际背景的限制而不能出现，那么其对应的最小项恒等于0
- 任意项：在输入变量的某些取值下，其最小项的取值为1/0不影响逻辑函数的值
- 约束项和任意项统称为逻辑函数中的无关项，可以写入可以删除

(3) 无关项在化简逻辑函数中的应用

- 加入的无关项应与函数式中尽可能多的最小项（包括原有的最小项和已写入的无关项）具有逻辑相邻性
- 合并最小项时，把卡诺图中的X作为1还是0，应以得到的相邻最小项举行组合最大，而且矩形组合数目最少为原则

例：化简具有约束的逻辑函数

$$Y = A'B'C'D + A'BCD + AB'C'D'$$

给定约束条件为

$$A'B'CD + A'BC'D + ABC'D' + AB'C'D + ABCD + ABCD' + AB'CD' = 0$$

在用最小项之和形式表示上述具有约束的逻辑函数时，也可写成如下形式

$$Y(A, B, C, D) = \sum m(1, 7, 8) + d(3, 5, 9, 10, 12, 14, 15)$$

式中以 d 表示无关项， d 后面括号内的数字是无关项的最小项编号。

解：如果不利用约束项，则 Y 已无可化简。但适当地加进一些约束项以后，可以得到

$$\begin{aligned}
 Y &= (\overbrace{A'B'C'D + A'B'CD}^{\text{约束项}}) + (\overbrace{A'BCD + A'BC'D}^{\text{约束项}}) \\
 &\quad + (\overbrace{AB'C'D' + ABC'D'}^{\text{约束项}}) + (\overbrace{ABCD' + AB'CD'}^{\text{约束项}}) \\
 &= (A'B'D + A'BD) + (AC'D' + ACD') \\
 &= A'D + AD'
 \end{aligned}$$

可见，利用了约束项以后，使逻辑函数得以进一步化简。但是，在确定该写入哪些约束项时尚不够直观。

改用卡诺图：

试化简具有无关项的逻辑函数

$$Y(A, B, C, D) = \sum m(2, 4, 6, 8) + d(10, 11, 12, 13, 14, 15)$$

画出卡诺图：

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

由图可见，若认为其中的无关项 m_{10}, m_{12}, m_{14} 为 1，而无关项 m_{11}, m_{13}, m_{15} 为 0，则可将 m_4, m_6, m_{12} 和 m_{14} 合并为 BD' ，将 m_8, m_{10}, m_{12} 和 m_{14} 合并为 AD' ，将 m_2, m_6, m_{10} 和 m_{14} 合并为 CD' ，于是得到

$$Y = BD' + AD' + CD'$$

9. 多输出逻辑函数的化简

- 在化简一组具有相同输入变量的逻辑函数时，可以在卡诺图分组时，尽量多地画出各个卡诺图中相同的公共项，这样得到的逻辑函数可能不是最简的，但在实现逻辑电路时，可以使输出有共同的输入，减少门电路和连线的数目。

$$Y_1(A, B, C, D) = \sum(1, 4, 5, 6, 7, 10, 11, 12, 13, 14, 15)$$

例： $Y_2(A, B, C, D) = \sum(1, 3, 4, 5, 6, 7, 12, 14,)$

$$Y_3(A, B, C, D) = \sum(3, 7, 10, 11,)$$

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

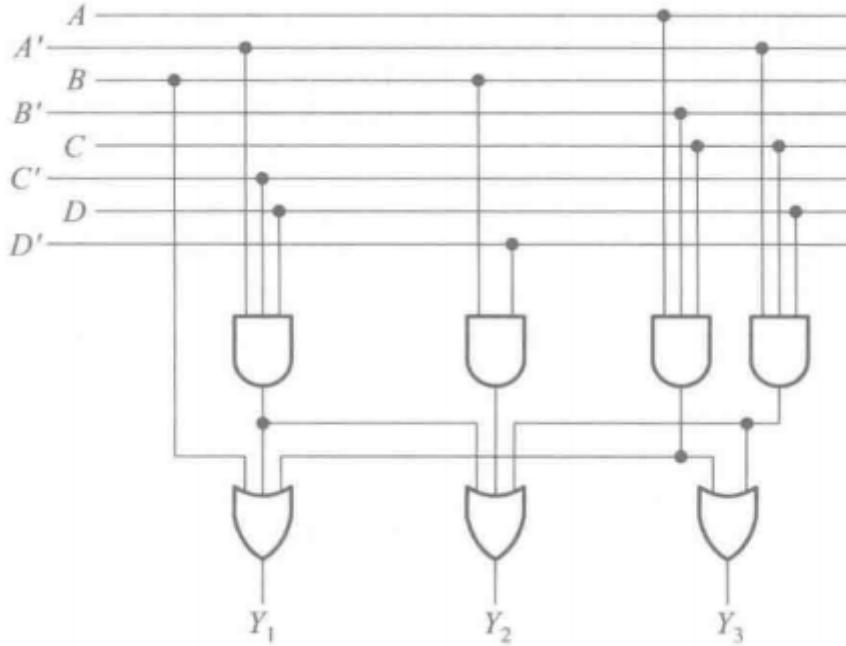
$$Y_1 = B + AB'C + A'C'D$$

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

$$Y_2 = A'C'D + A'CD + BD'$$

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

$$Y_3 = A'CD + AB'C$$



三、门电路

1. 概述

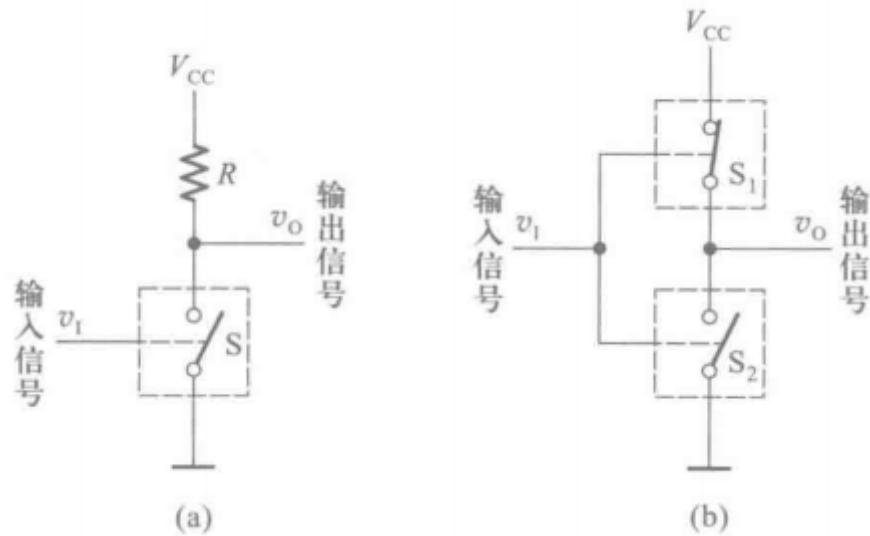
- Why digital?

Because it keeps the contracts simple! The price we pay for this robustness:

All the information that we transfer between modules is only 1 crummy bit!

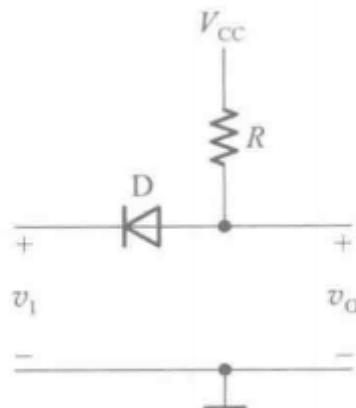
- Keep in mind that the world is not digital, we would simply like to engineer it to behave that way. Furthermore, we must use real physical phenomena to implement digital designs!
- Using Voltages "Digitally"
 - Key idea: don't allow "0" to be mistaken for a "1" or vice versa
 - Use the same "uniform representation convention" for every component and wire in our digital system to implement devices with high reliability, we outlaw "close calls" via a representation convention which forbids a range of voltages between "0" and "1"
- The digital abstraction
 - Making bits concrete
 - What makes a good bit
 - Getting bits under contract

- 门电路：实现基本运算、复合运算的单元电路，如与门、与非门、或门……
 - 正逻辑：高电平表示1，低电平表示0
 - 负逻辑：高电平表示0，低电平表示1
- 获得高、低电平的基本原理：高/低电平都允许有一定的变化范围

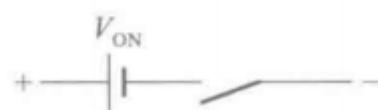
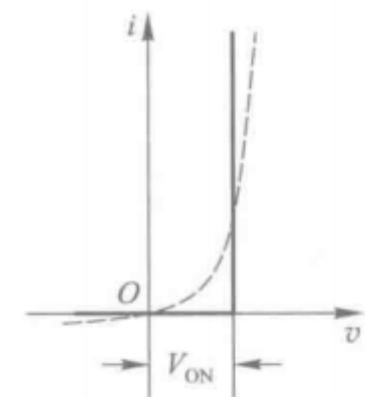


2. 半导体二极管门电路

- 二极管开关电路：

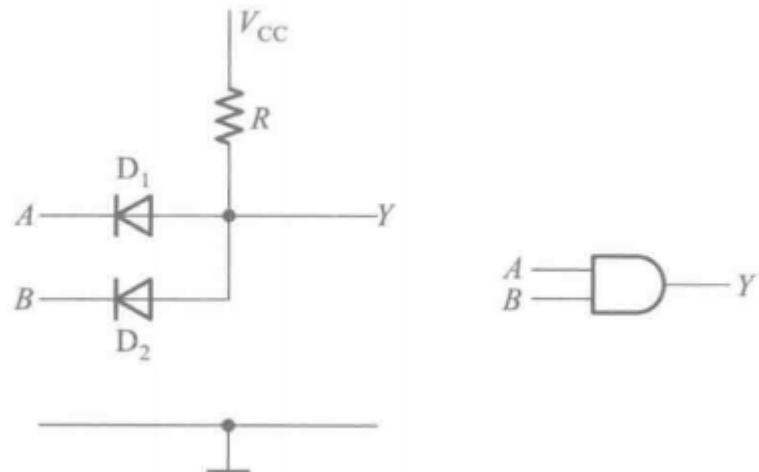


- 数字电路中二极管的伏安特性及近似方法：



(1) 二极管与门

- 电路图: ($V_{CC} = 5V$)



- 逻辑电平及真值表

A/V	B/V	Y/V
0	0	0.7
0	3	0.7
3	0	0.7
3	3	3.7

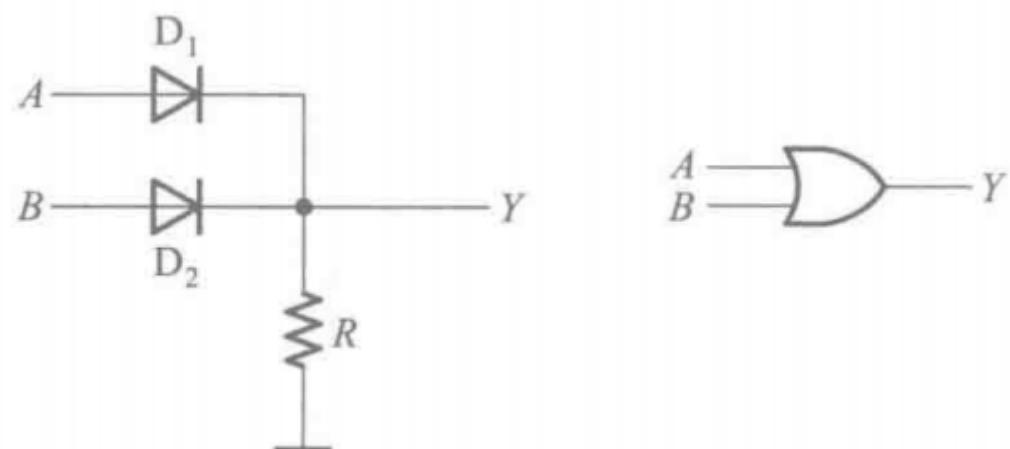
A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

- 缺点:

- 输出的高电平和输入的高电平不相等，不能将输出作为下一个的输入
- 当输出端对地接上负载电阻时，负载电阻的改变有时会影响输出的高电平，带负载能力差

(3) 二极管或门

- 电路图:



- 逻辑电平及真值表

A/V	B/V	Y/V
0	0	0
0	3	2.3
3	0	2.3
3	3	2.3

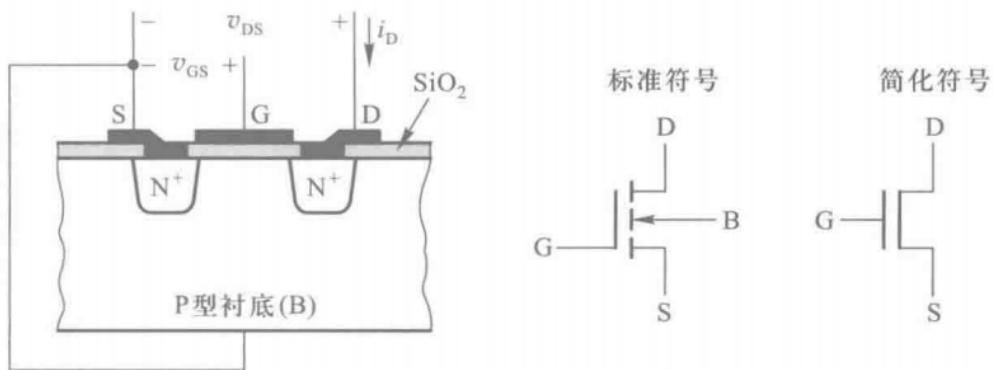
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

- 二极管或门同样存在着输出点评偏移的问题，只适用于集成电路内部的逻辑单元，无法制作具有标准化输出电平的集成电路

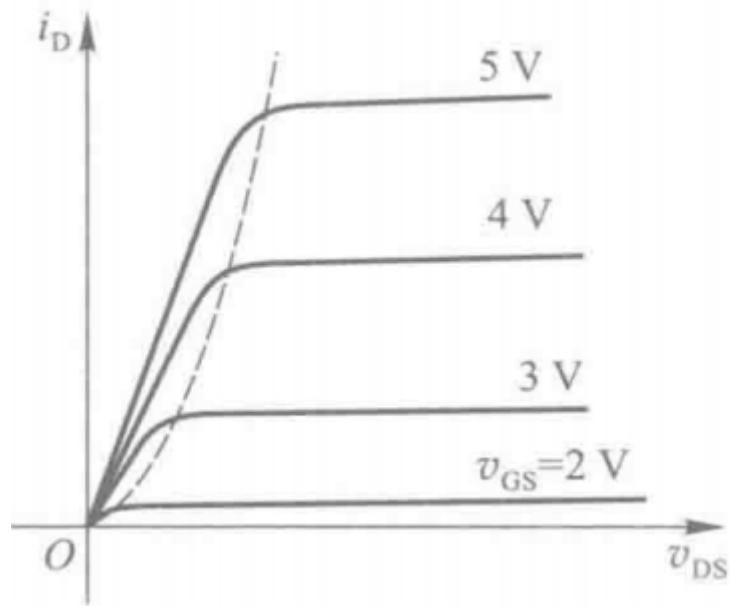
3. CMOS门电路 (C: Complementary)

(1) MOS管复习

- MOS管的结构和符号（以N沟道增强型为例）

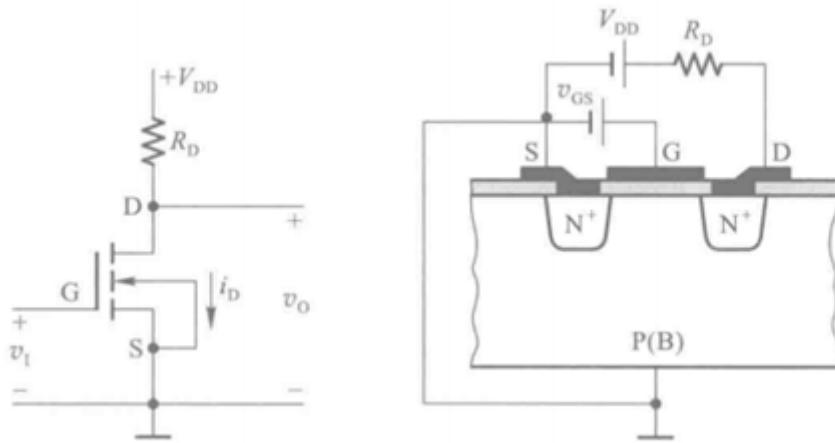


- 源极S(Source)、漏极(Drain)、栅极(Gate)
- MOS管的输入输出特性 (N沟道增强型, 共源)
 - 漏极-源极间为输入回路, 加上电压Vgs后不会有栅极电流。
 - 输出特性:
虚线左边为可变电阻区, Vgs一定时, Id与Vds之比近似为常数
虚线右边为恒流区, Id大小基本由Vgs决定



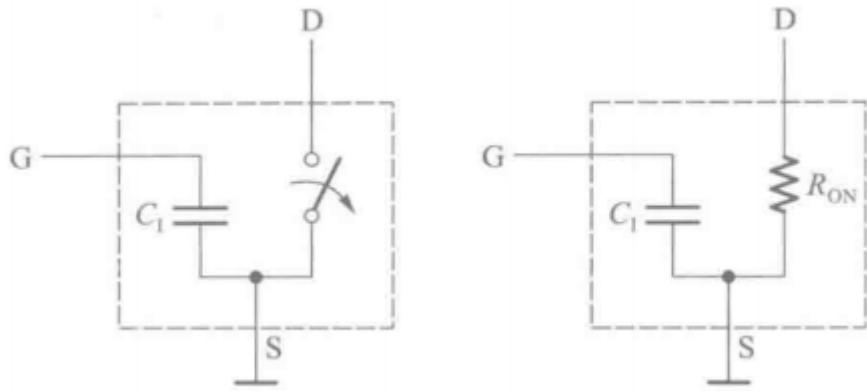
- MOS管的基本开关电路（以N沟道增强型为例）

- $V_1 = V_{GS} < V_{Gsth}$ 时，MOS工作在截止区，只要负载电组 R_d 远小于MOS截止内阻，输出即为高电平 V_{Ohi} ，约为 V_{DD} ，相当于断开的开关
- $V_1 > V_{Gsth}$ 且 V_{DS} 较高时，MOS工作在恒流区，电路工作在放大状态
- V_1 继续升高，只要 $R_d \gg R_{on}$ ，输出端将为低电平 V_{Vol} ，约为 0，MOS管的D-S间相当于闭合开关
- 只要电路参数选择合理，就可以做到输入为低电平时MOS截止，开关电路输出高电平，输入为高电平时，MOS导通，开关电路输出低电平



- MOS管的开关等效电路

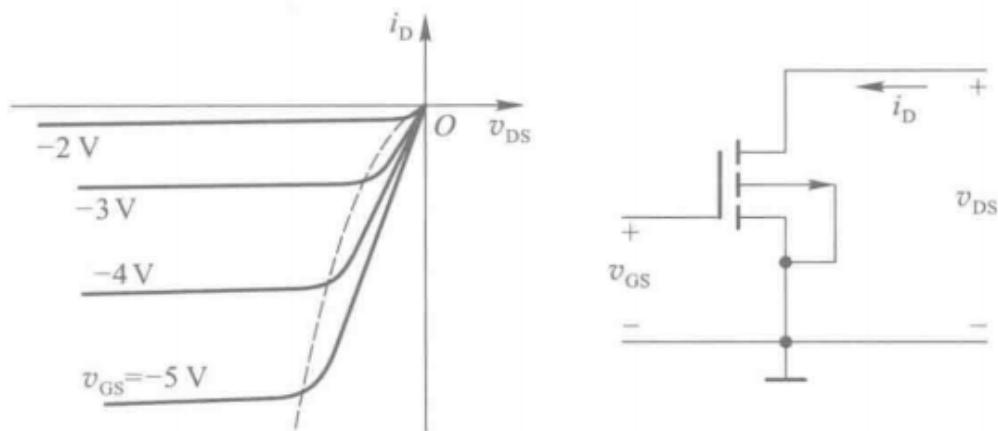
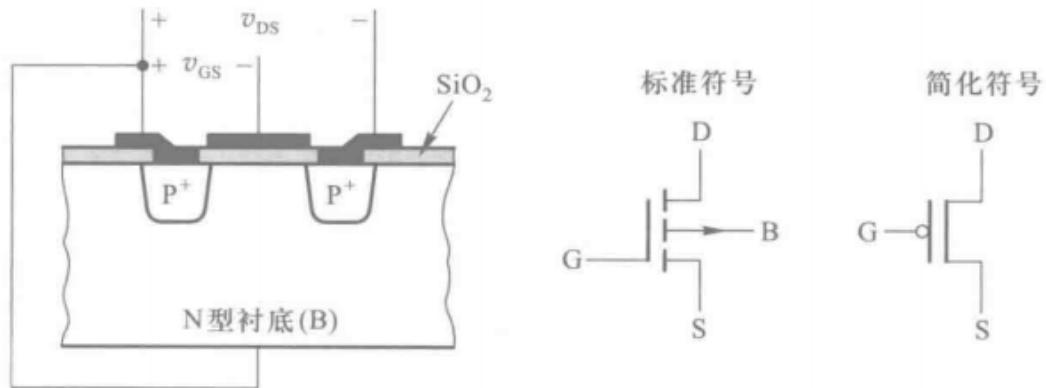
- MOS截止时D-S内阻非常大，视为断路
- MOS导通时D-S内阻很小，1kΩ以内，不能不计
- C1几皮法，负载电容不可避免

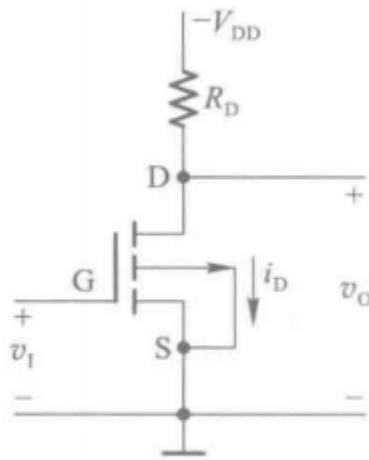


- MOS管的四种类型

1. N沟道增强型

2. P沟道增强型，工作时使用负电源，开启电压 V_{gsth} 为负值。当 $V_1 = 0$ 时，MOS管不导通，输出低电平 V_{ol} ，只要 R_d 远小于MOS管截止内阻，则 V_{ol} 约为 $-V_{dd}$ ；当 $V_1 < V_{gsth}$ 时，MOS管导通，输出为高电平 V_{oh} ，只要 R_d 远大于MOS管的导通内阻 R_{on} ，则 V_{oh} 约为0



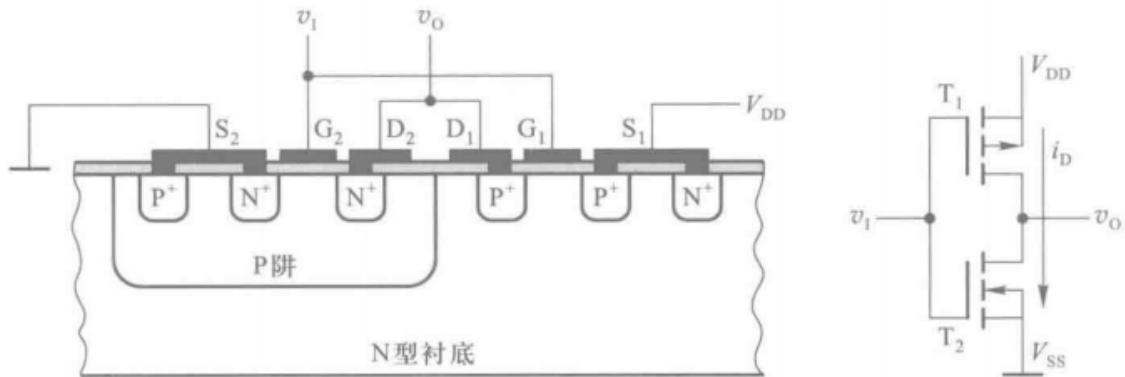


3. N沟道耗尽型，在 $V_{GS} = 0$ 时到导电沟道就已经存在， V_{GS} 为正时沟道变宽， I_D 增大， V_{GS} 为负时沟道变窄， I_D 减小，直到 V_{GS} 小于某一负电压值 V_{GSOFF} 时，导电沟道才消失，MOS截止， V_{GSOFF} 称为夹断电压
4. P沟道耗尽型，在 $V_{GS} = 0$ 时到导电沟道就已经存在， V_{GS} 为负时沟道变宽， $|I_D|$ 的绝对值增大， V_{GS} 为正时沟道变窄， $|I_D|$ 的绝对值减小，当 V_{GS} 的正电压大于夹断电压 V_{GSOFF} 时，导电沟道消失，管子截止。

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		-	-	+		
P 沟道耗尽型	N 型	P 型		+	+	-		

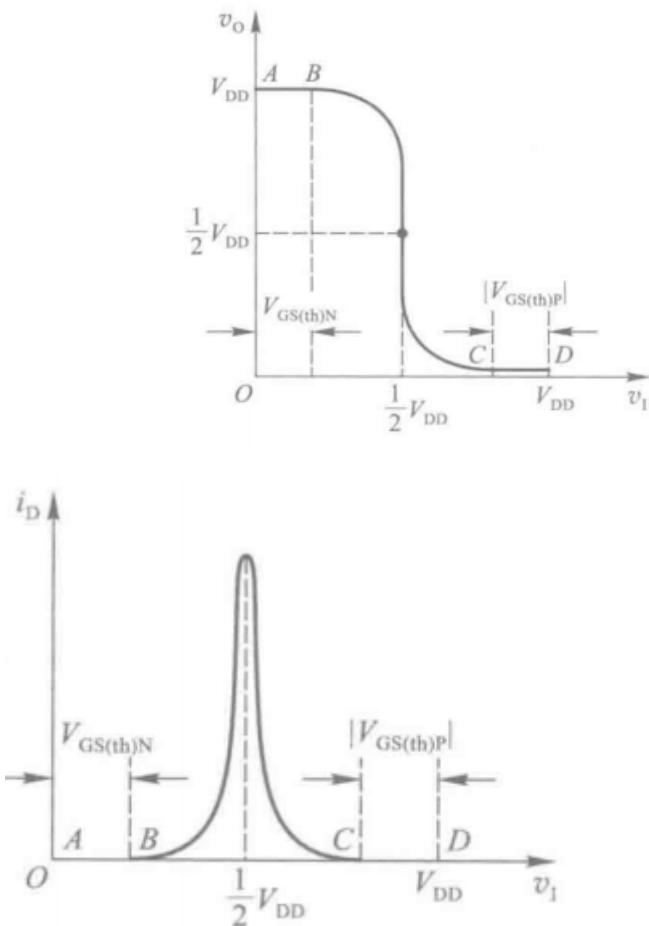
(2) CMOS 反相器的电路结构和工作原理

- CMOS反相器结构与电路图



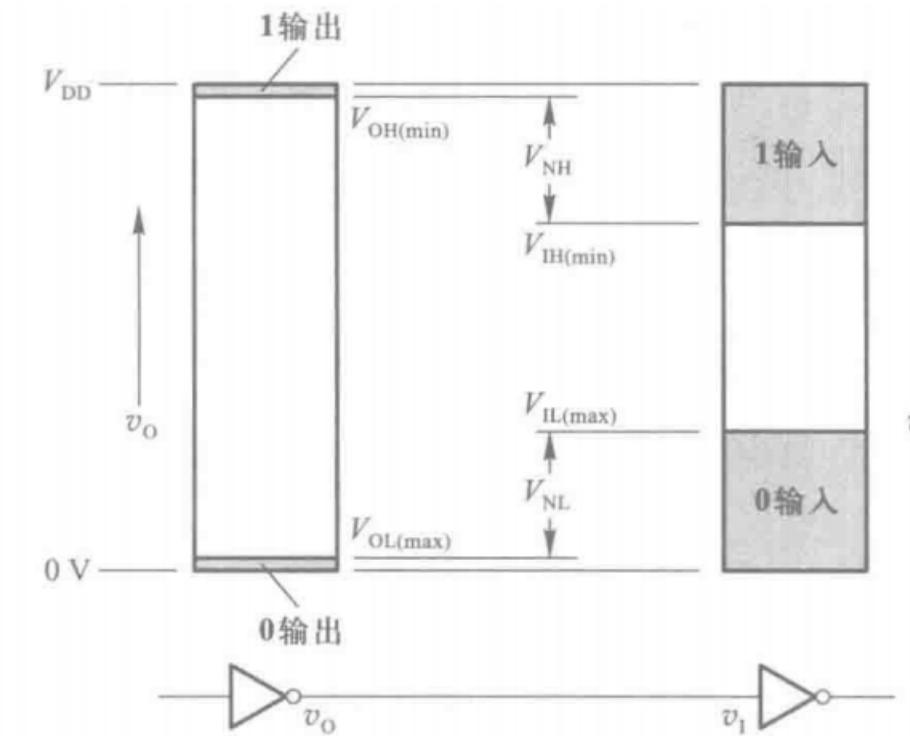
当 $V_1 = 0$ 时, T_1 导通, 导通内阻很低, T_2 截止, 内阻很高, 输出为高电平 $V_{oh} = V_{dd}$; 当 $V_1 = V_{dd}$ 时, T_1 截止而 T_2 导通, 输出为低电平 $V_{ol} = 0$ 。输入输出为逻辑非的关系, 称为非门或反相器。

- 电压传输特性和电流传输特性

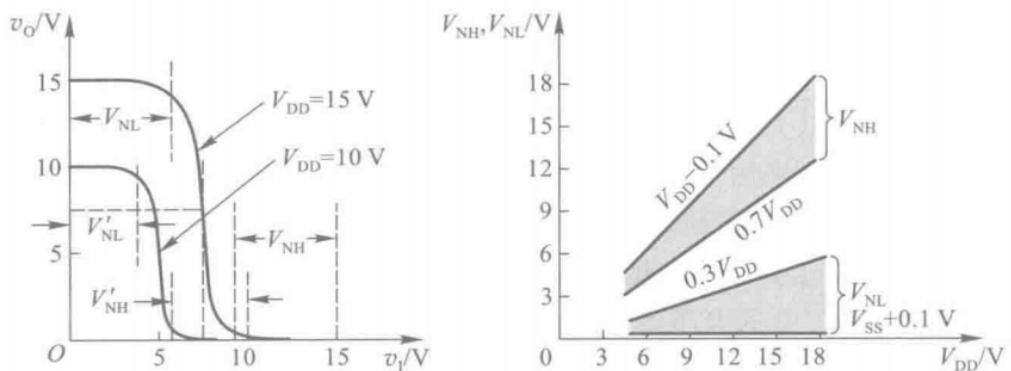


很明显, 我们一般利用其电压传输特性, 更接近理想的开关特性, 而不利用电流传输特性, 在BC段工作时, 器件可能会因功耗过大而损坏

- 输入噪声容限: 输入允许的0/1的差距



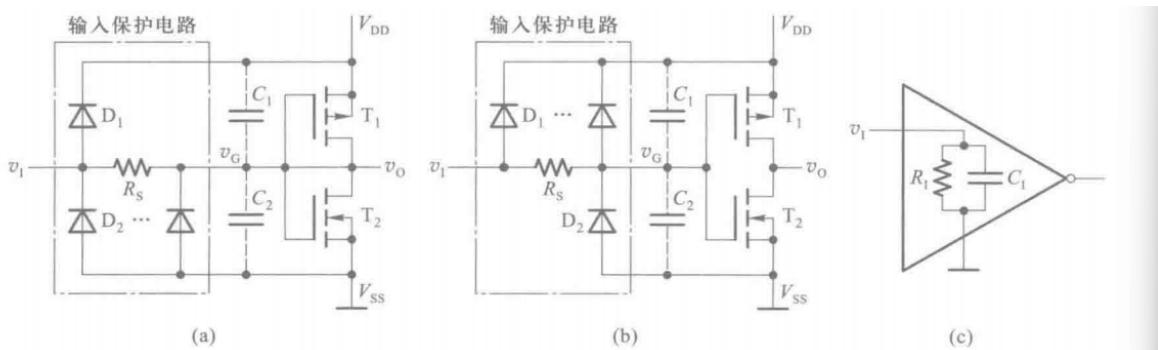
在CMOS门电路中，当负载为另外的门电路的情况下（相当于空载）， $V_{oh\min} = V_{dd} - 0.1V$ ， $V_{ol} = V_{ss} + 0.1V = 0.1V$ ， V_{ss} 表示源极电位，为0。实验得 $V_{nh} = V_{nl} = 30\%V_{dd}$ ，因此可通过提高 V_{dd} 来提高噪声容限，但功耗会增加，不能随意增大



(3) CMOS反相器的静态输入特性和输出特性

- 输入特性：反相器输入端看进去的输入电压和电流的关系（看课本）

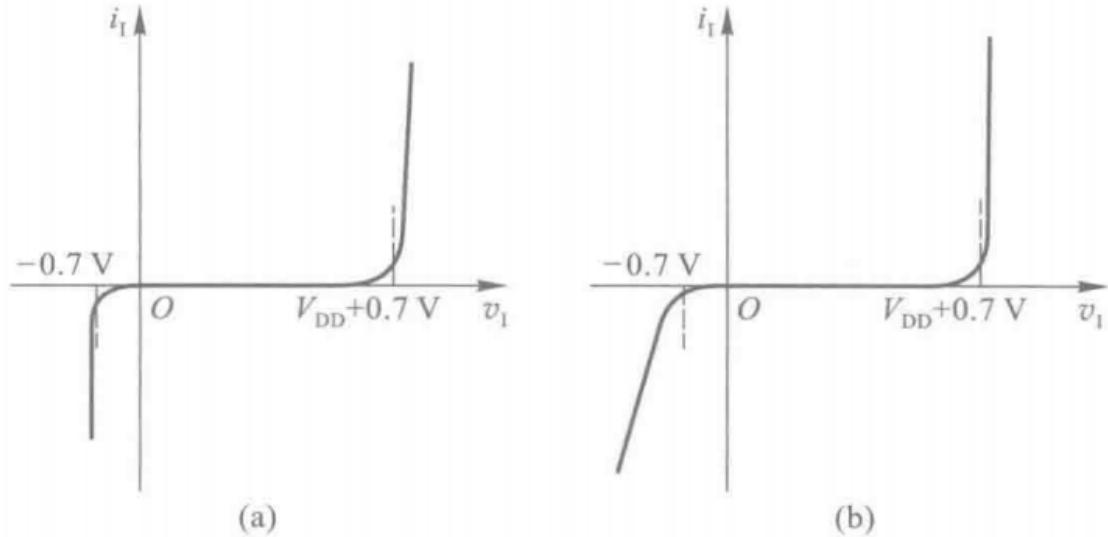
因MOS管的栅极与衬底间的绝缘介质非常薄，易被击穿，因此采用保护电路，常用的保护电路有两种：



在74HC系列的CMOS器件中，多采用类似于图a所示的保护电路，双极型二极管D1、D2的正向导通压降 $V_{df} = 0.5\sim0.7V$ ，反向击穿电压约为30V，D2是输入端的N型扩散电阻区和P型衬底间自然形成的，为分布式二极管结构，用虚线和两端的二极管表示，C1、C2分别表示T1和T2的栅极等效电容。

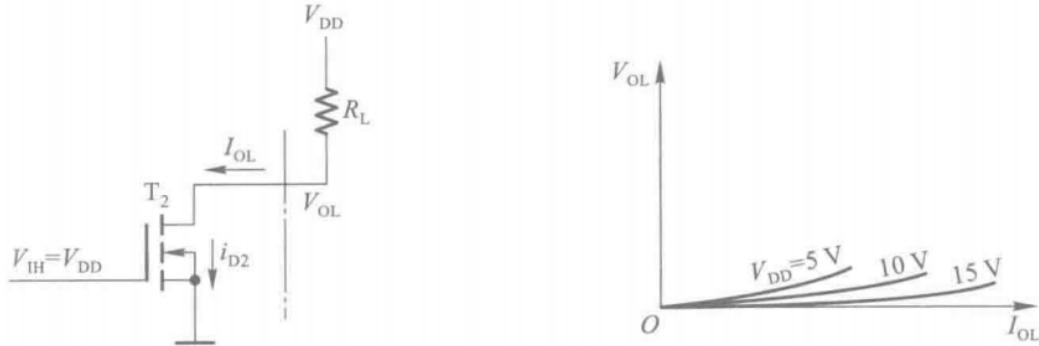
输入端的等效电路可以用图c表示，C1的典型数值为5pF，R1的数值在10Mohm以上

a、b的输入输出特性曲线

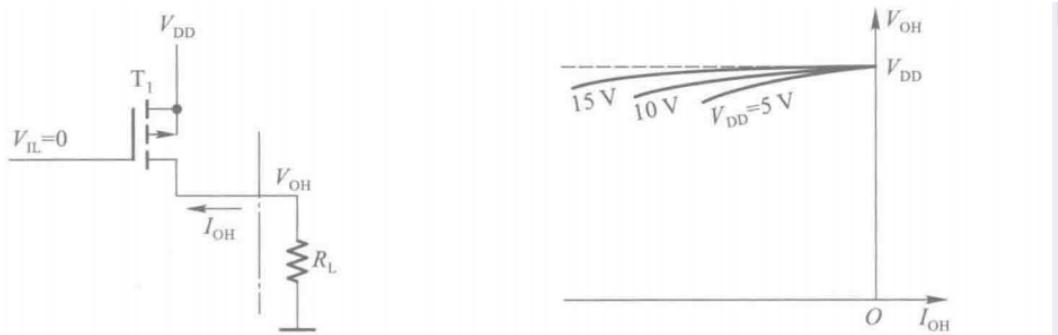


- 输出特性：从反相器的输出端看进去的输出电压与输出电流的关系

低电平输出特性：输出为低电平，P沟道截止，N沟道导通，等效电路如下，负载电流 I_{OL} 从负载电路注入T2，使输出电平升高。 V_{DD} 越大， V_{OL} 越低。



高电平输出特性：输出为高电平，P沟道导通，N沟道截止，等效电路如下，负载电流 I_{oh} 从门电路的输出端流出，拉低了输出电平， V_{dd} 越高， V_{oh} 越高。

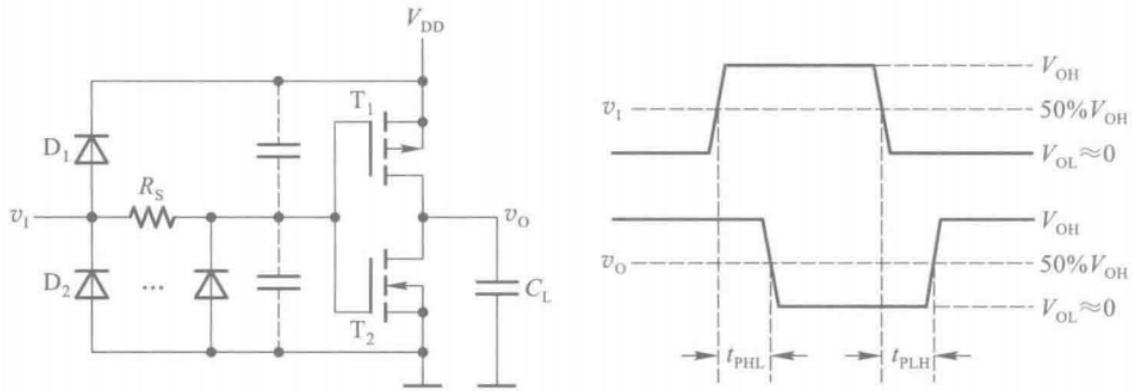


(4) CMOS反相器的动态特性

动态特性讨论的是当电路状态转换过程中所表现出来的一些性质

- 传输延时时间

延迟产生原因：输入和负载都存在电容，输入信号发生跳变时，输出电压的变化必然滞后于输入电压的变化，从而产生延迟。

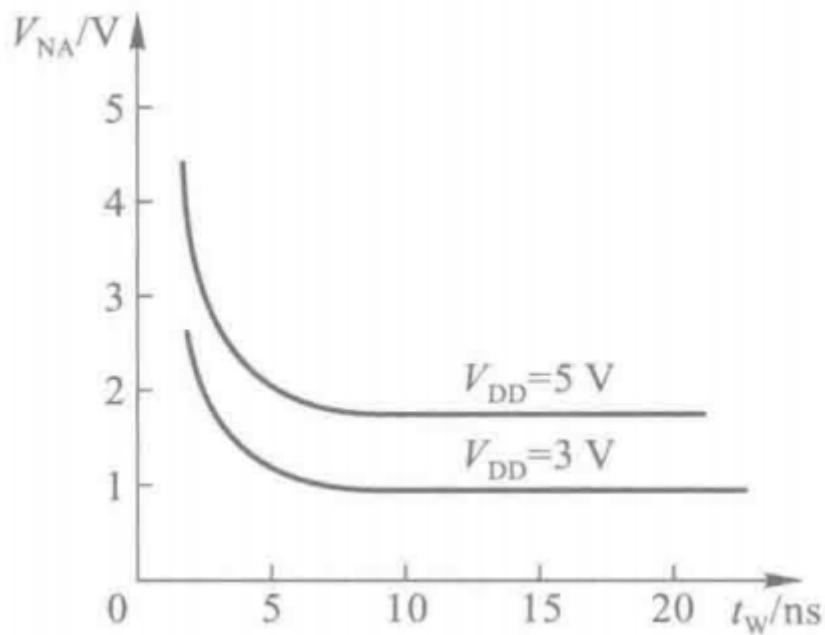


tphl与tplh通常是相等的，以tpd表示。为缩短延迟，必须减小负载电容和MOS管的导通电阻，所以应尽可能的提高电源电压和输入信号的高电平。

- 交流噪声容限

由于负载电容和MOS管寄生电容的存在，输入信号状态变化时必须有足够的变化幅度和作用时间才能使输出状态改变。当输入信号为窄脉冲，而且脉冲宽度接近门电路传输延迟时间，为使输出状态改变所需要的脉冲信号幅度将远大于直流输入信号的幅度。因此，反相器对这类窄脉冲的噪声容限——交流噪声容限远高于直流噪声容限，传输延迟时间越长，交流噪声容限也越大。

交流噪声容限Vna也受电源电压和负载电容的影响，tw表示噪声电压的持续时间



- 动态功耗

指CMOS反相器从一种稳定工作状态突然转变到另一种稳定状态的过程中，将产生附加的功耗，称之为动态功耗，其由对负载电容的充放电功耗和MOS管导通功耗组成

充放电功耗：

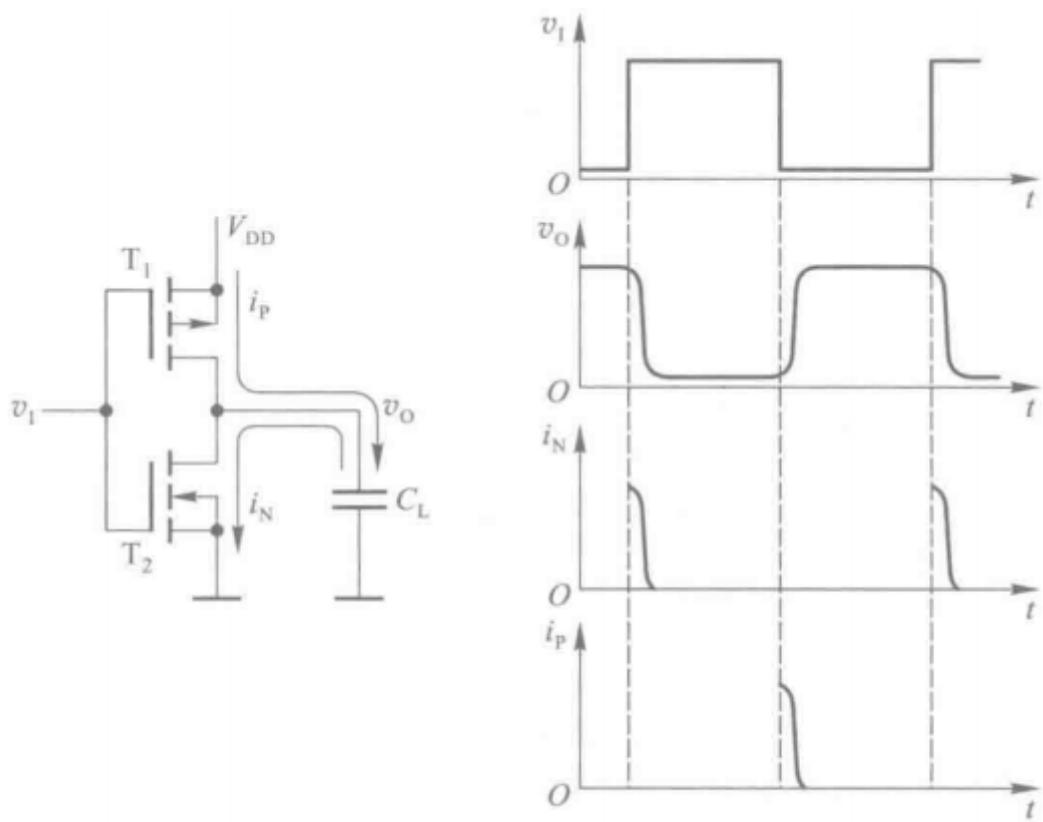


图 3.3.24 CMOS 反相器对负载电容的充、放电电流波形

输入电压在高低电平间转换时，产生充电电流*i_N*和放电电流*i_P*，可计算的平均功耗：

$$P_C = \frac{1}{T} \left[\int_0^{T/2} i_N v_O dt + \int_{T/2}^T i_P (V_{DD} - v_O) dt \right]$$

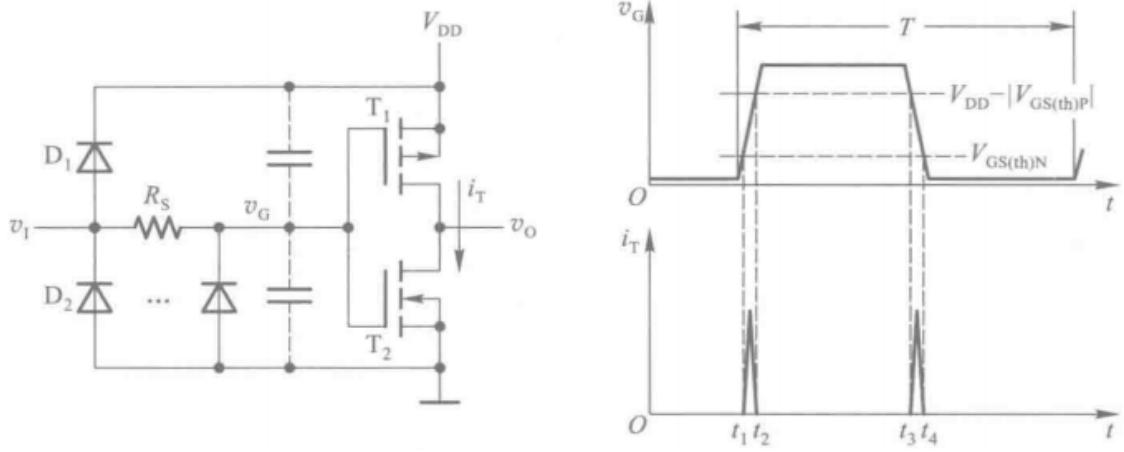
$$i_N = -C_L \frac{dv_O}{dt}$$

$$i_P = C_L \frac{dv_O}{dt} = -C_L \frac{d(V_{DD} - v_O)}{dt}$$

$$\begin{aligned} P_C &= \frac{1}{T} \left[C_L \int_{V_{DD}}^0 -v_O dv_O + C_L \int_{V_{DD}}^0 -(V_{DD} - v_O) d(V_{DD} - v_O) \right] \\ &= \frac{C_L}{T} \left[\frac{1}{2} V_{DD}^2 + \frac{1}{2} V_{DD}^2 \right] \\ &= C_L f V_{DD}^2 \end{aligned}$$

导通功耗：

在很短的时间内，T1，T2同时导通，有瞬时导通电流*i_T*流过T1、T2：



瞬时导通功耗 P_t 、电源电压 V_{DD} 、输入信号 v_1 的重复频率：

$$P_t = C_{PD} f V_{DD}^2$$

总的功耗 P_d 应为 P_c 和 P_t 之和：

$$\begin{aligned} P_d &= P_c + P_t \\ &= (C_L + C_{PD}) f V_{DD}^2 \end{aligned}$$

在工作频率较高的情况下，CMOS反相器的动态功耗要比静态功耗大得多，这时静态功耗可以忽略。

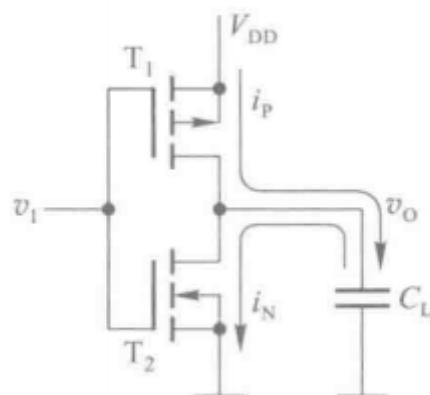
- 扇出（看课本）

“扇出”是以数字表示一个电路的输出端能够驱动的同类型负载电路输入端的数目。理想是无限。

(5) 其他类型的CMOS门电路

其他逻辑功能的CMOS门电路

For CMOS inverter:



Pullup: make this connection when Vin near 0 so that Vout = Vdd

Pulldown: make this connection when Vin near Vdd so that Vout = 0

Now we want **complementary** pullup and pulldown logic, the pulldown should be "on" when the pullup is "off" and vice versa.

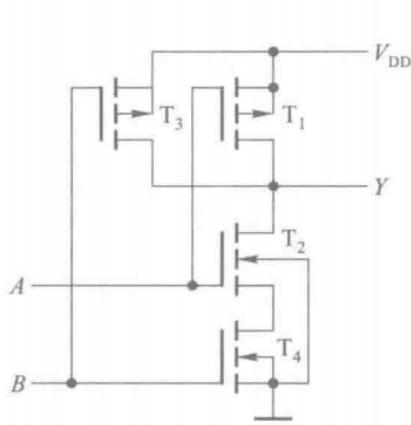


图 3.3.28 CMOS 与非门

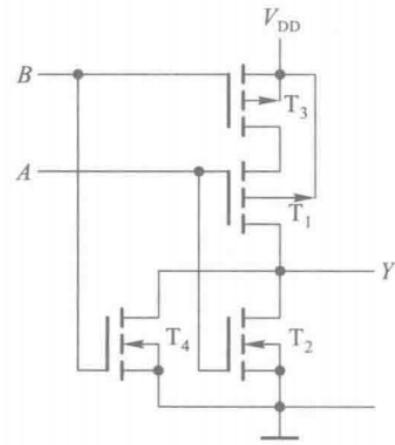


图 3.3.29 CMOS 或非门

(忽略输入端保护电路)

CMOS 与非门：N 沟道串联，P 沟道并联，只要 A、B 中有一个是低电平，输出即为高电平。当 A = 1 B = 0 时，T3 导通，T4 截止，Y = 1；A = 0 B = 1 时，T1 导通，T2 截止，Y = 1；只有 A = B = 1 时，T1 和 T3 同时截止，T2 和 T4 同时导通，Y = 0。

CMOS 或非门：N 沟道并联，P 沟道串联，只要 A、B 中有一个是高电平，输出就是低电平。当 A = 1 B = 0 时，T2 导通，T1 截止，Y = 0；A = 0 B = 1 时，T4 导通，T3 截止，Y = 0；只有 A = B = 0 时，T2 和 T4 同时截止，T1 和 T3 同时导通，Y = 1。

利用与非门、或非门和反相器有可组成与门、或门、与或非门、异或门等（根据字面意义组合电路即可）

缓冲级：在门电路的每个输入端、输出端各增设一级反相器，此时反相器被称为缓冲器。带有缓冲级的门电路的输出电阻输出的高低电平以及电压传输特性将不再受输入端状态的影响。需要注意的是，添加缓冲器后，与非电路变或非电路，或非电路变与非电路。

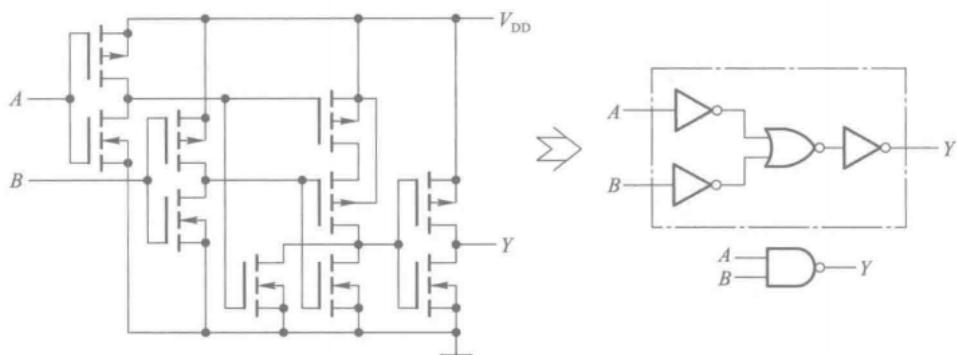


图 3.3.30 带缓冲级的 CMOS 与非门电路

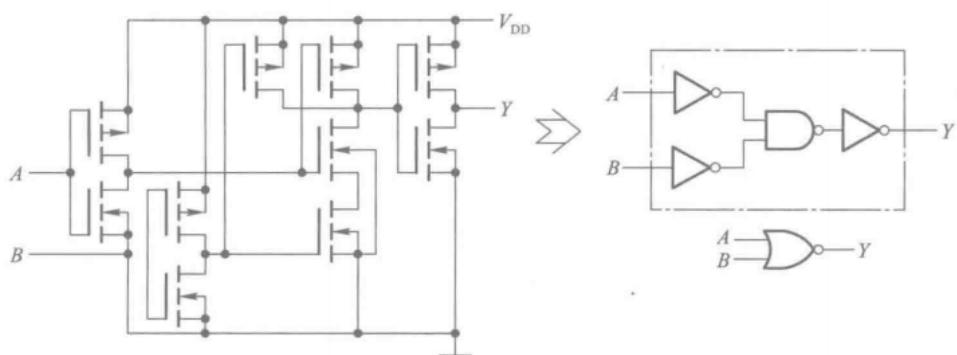


图 3.3.31 带缓冲级的 CMOS 或非门电路

漏极开漏的门电路

线与逻辑：即两个输出端（包括两个以上）直接互连就可以实现“AND”的逻辑功能。但不能直接使用，否则会引起器件烧毁。

在CMOS电路中为了满足输出电平转换，吸收大负载电流以及实现线与连接需要，有时将输出级电路结构改为一个漏极开路输出的MOS管，构成漏极开路输出门电路（Open-Drain Output, OD门）。

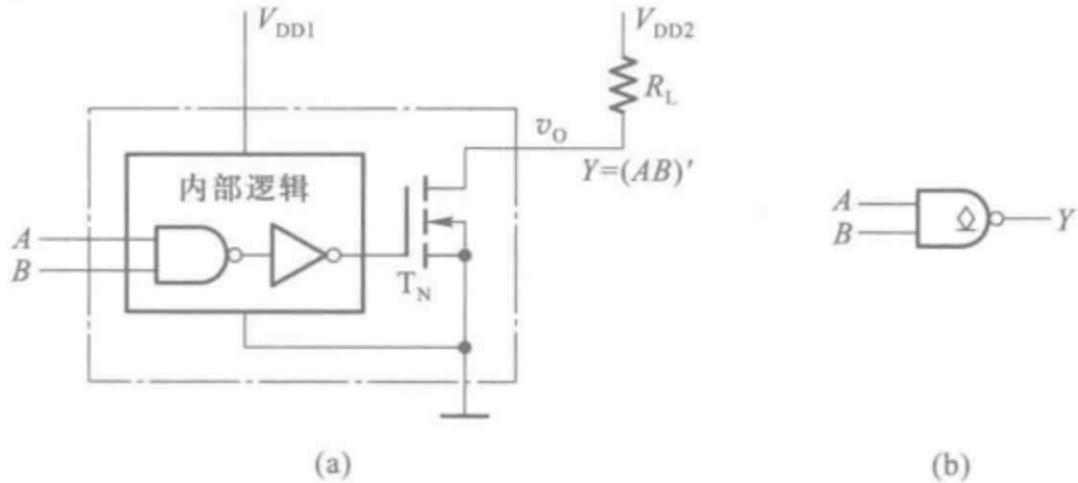
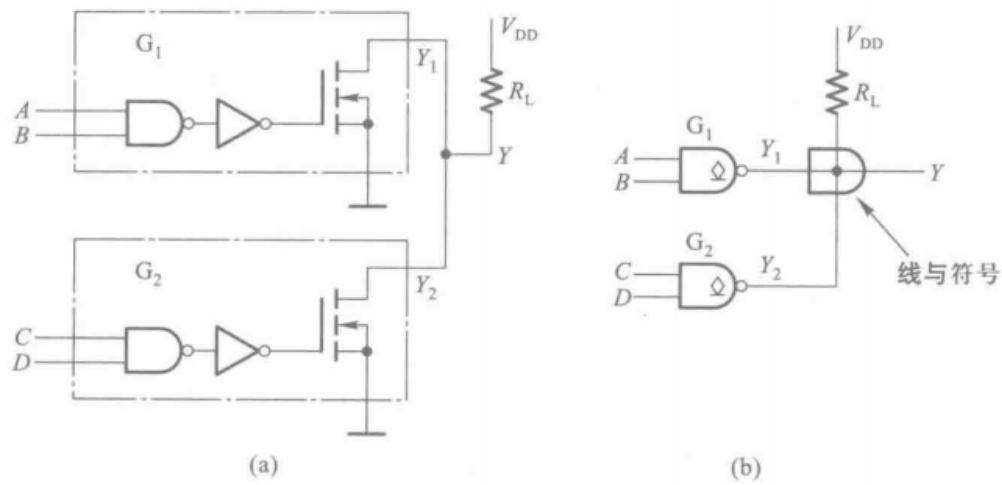


图 3.3.32 OD 输出的与非门

(a) 电路结构 (b) 逻辑符号

OD门工作时必须将输出端经上拉电阻RL接到电源上，设TN的截止内阻和导通内阻分别为Roff和Ron，则只要满足Roff ≫ RL ≫ Ron，就能使输出高电平为Vdd2，从而实现输出高电平从Vdd1变为Vdd2。OD门的另一个作用是实现线与逻辑，如图b，只有Y1、Y2均为高电平时，Y才为高电平。



外接电阻的计算方法：

若RL取值过大，则输出的高电平可能会低于阈值，由此可计算RLmax。若每个OD门输出管截止时的漏电流为Ioh，负载门每个输入端的高电平输入电流为Iih，要求输出高电平不低于Voh，可得到：

$$V_{DD} - (nI_{OH} + mI_{IH}) R_L \geq V_{OH}$$

$$R_L \leq (V_{DD} - V_{OH}) / (nI_{OH} + mI_{IH}) = R_{L(max)}$$

式中的n是并联OD门的数目，m是负载门电路高电平输入电流的数目。

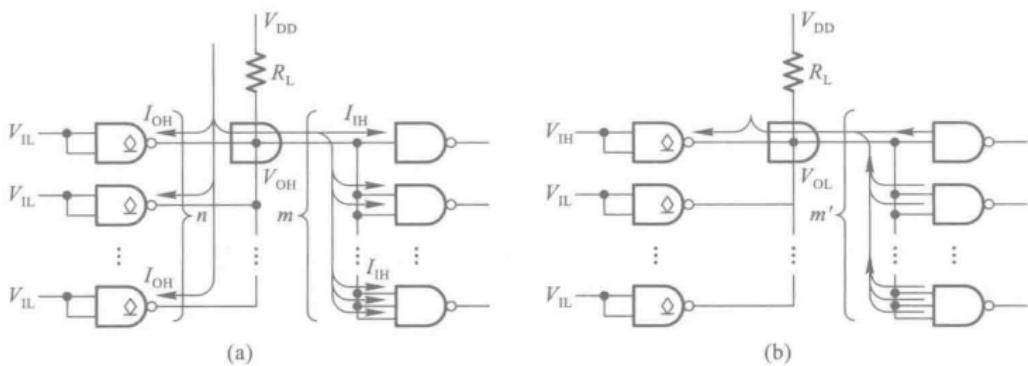


图 3.3.34 OD 门外接上拉电阻的计算

(a) R_L 最大值的计算 (b) R_L 最小值的计算

若只有一个OD门输出MOS导通，输出为低电平，当若RL取值过小，则负载电流会过大，可能会超出该导通MOS管所允许的最大电流，据此可计算 $R_{L\min}$ ，若OD门允许的最大负载电流为 $I_{OL(\max)}$ 、负载门的每个输入端的低电平输入电流输入电流为 I_{IL} ，此时的输出低电平为 V_{OL} ，则：

$$(V_{DD} - V_{OL}) / R_L + m' |I_{IL}| \leq I_{OL(\max)}$$

$$R_L \geq (V_{DD} - V_{OL}) / (I_{OL(\max)} - m' |I_{IL}|) = R_{L(\min)}$$

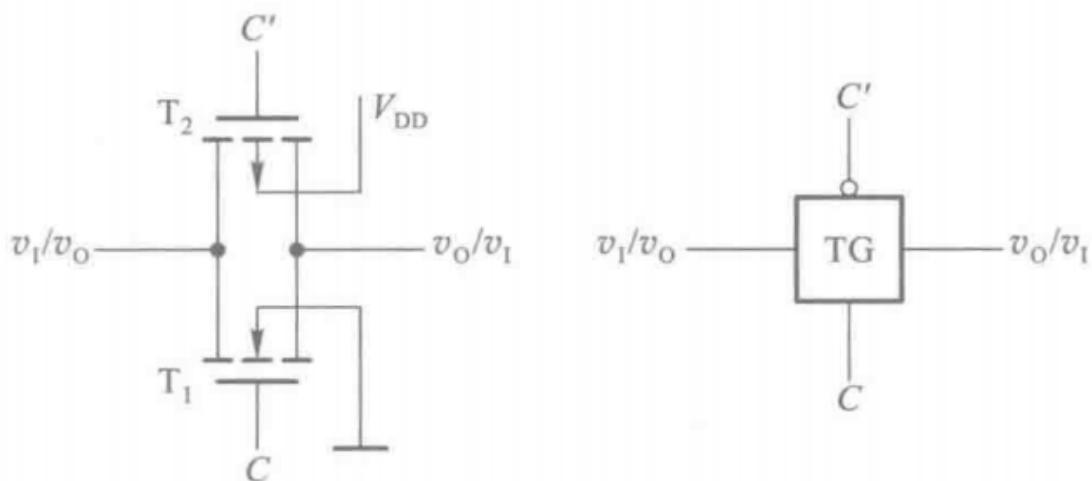
m' 为负载门电路低电平输入电流的数目，在负载为CMOS门电路的情况下， m 和 m' 相等。

那么：

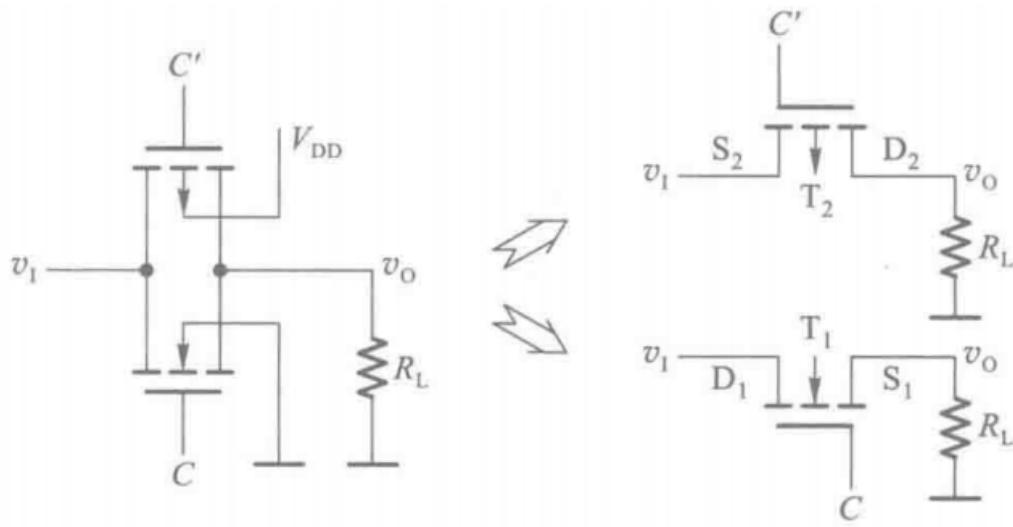
$$R_{L(\max)} \geq R_L \geq R_{L(\min)}$$

CMOS传输门

也是构成各自逻辑电路的基本单元电路



C 和 C' 是互补控制信号，高低电平为 V_{dd} 和0，若 $C = 0, C' = 1$ ，只要输入信号在0到 V_{dd} 之间，则 $T1$ 和 $T2$ 同时截止，传输门截止；反之，则当 $0 < V_1 < V_{dd} - V_{gsth}$ 时 $T1$ 导通， $|V_{gsth}p| < V_1 < V_{dd}$ 时， $T2$ 导通， $T1$ 与 $T2$ 至少有一个是导通的。CMOS传输门是双向器件，输入和输出、源极漏极可互易使用。



利用传输门和反相器构建异或门：

当 $A = 1, B = 0$ 时, TG_1 截止而 TG_2 导通, $Y = B' = 1$;

当 $A = 0, B = 1$ 时, TG_1 导通而 TG_2 截止, $Y = B = 1$;

当 $A = B = 0$ 时, TG_1 导通而 TG_2 截止, $Y = B = 0$;

当 $A = B = 1$ 时, TG_1 截止而 TG_2 导通, $Y = B' = 0$ 。

因此, Y 与 A, B 之间是异或逻辑关系, 即 $Y = A \oplus B$ 。

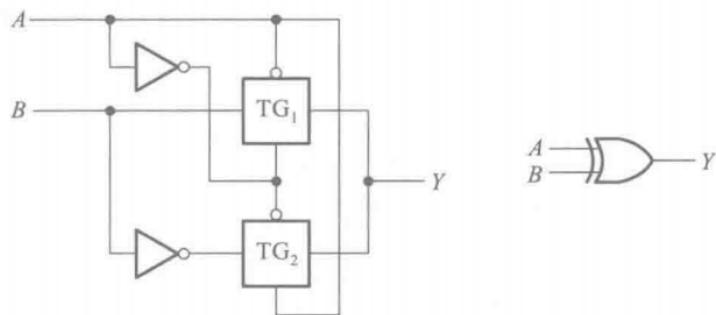


图 3.3.38 用反相器和传输门构成的异或门电路

利用传输门作模拟开关：传输连续变化的模拟电压信号。 (应该不重要)

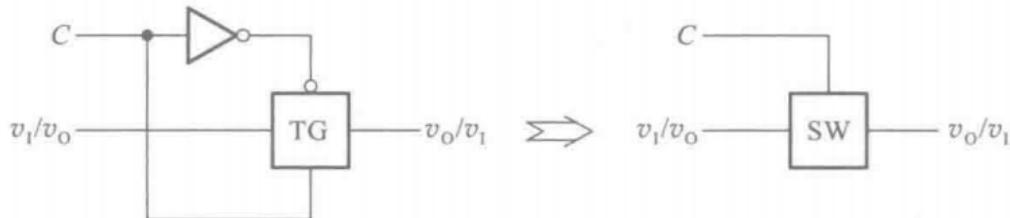


图 3.3.39 CMOS 双向模拟开关的电路结构和符号

三态输出的CMOS门电路

高电平、低电平、高阻态

三态输出反相器：

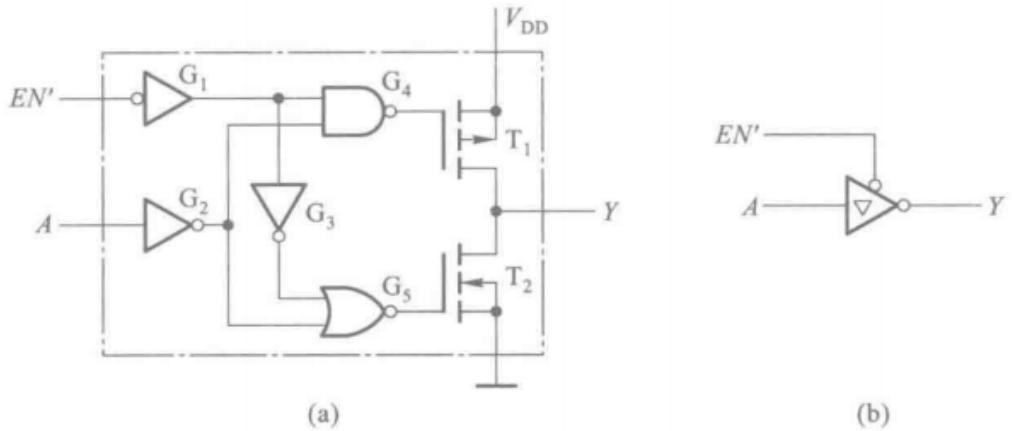


图 3.3.41 三态输出的 CMOS 反相器

(a) 电路结构 (b) 逻辑符号

$EN' = 0$ 时，电路为反相器，输出 $Y = A'$ ； $EN' = 1$ 时，不管 A 如何， T_1 和 T_2 同时截止，输出呈现高阻态。

利用三态输出可以实现总线结构，利用总线分时传输若干个门电路的输出信号。三态输出还可实现数据的双向传输， $EN = 1$ 时，数据传输到总线， $EN = 0$ 时，数据从总线传输到电路。

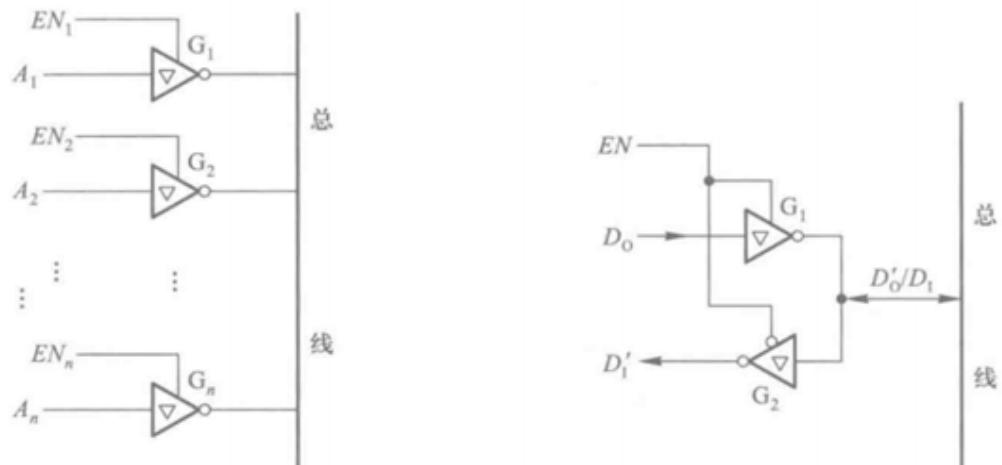


图 3.3.42 用三态输出反相器接成
总线结构

图 3.3.43 用三态输出反相器实现
数据双向传输

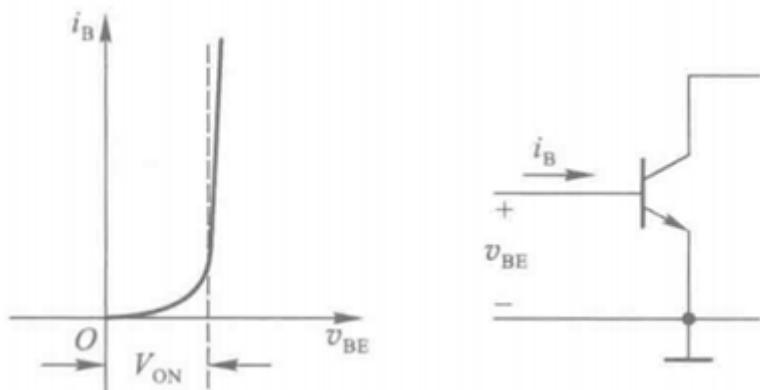
4. TTL门电路：简要看了，没细学，摆！

TTL：三极管——三极管逻辑 (Transistor-Transistor)

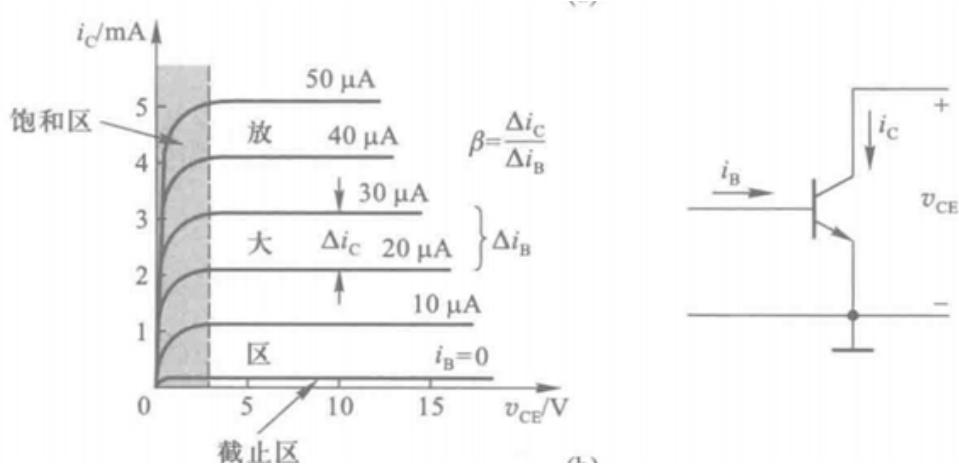
(1) 双极性三极管BJT复习

概念回顾：基级，集电极，发射极、NPN、PNP、

输入特性曲线：

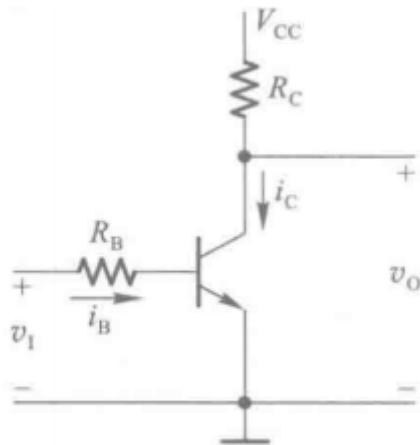


输出特性曲线：



饱和区、放大区、截止区

(2) 三极管反相器



输入电压V1为0时（或V1<Von）， $v_{BE} = 0$ ，三极管截止，三极管输出高电平

当V1 > Von后，三极管进入放大区，模电内容

V1继续升高， R_C 上的压降也随之增大，压降接近Vcc时，三极管饱和，三极管输出低电平，饱和基极电流 $i_{BS} = V_{CC}/(\beta \cdot R_C)$ 。

(3) 三极管反相器的开关等效电路

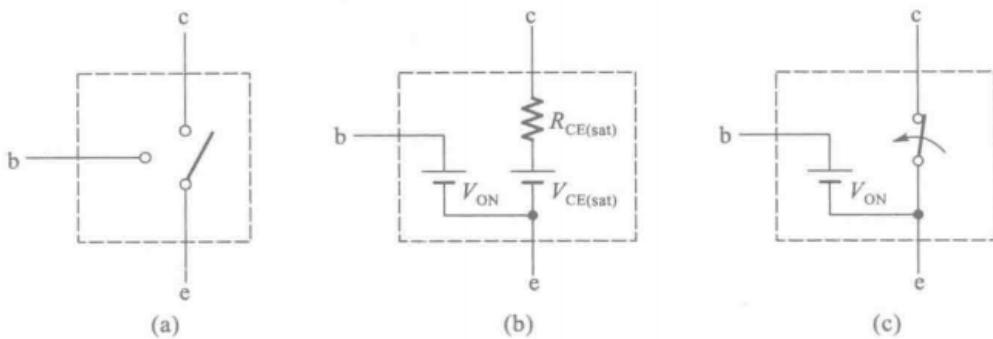


图 3.4.5 双极型三极管的开关等效电路

(a) 截止状态 (b)、(c) 饱和导通状态

当输入为低电平，并且 $V_{il} < V_{on}$ ，三极管处于截止状态，输出为高电平。当输入为高电平 $V_{ih} > V_{on}$ ，并且 $i_b > I_{bs}$ ，三极管处于饱和导通状态，输出低电平

动态开关特性：

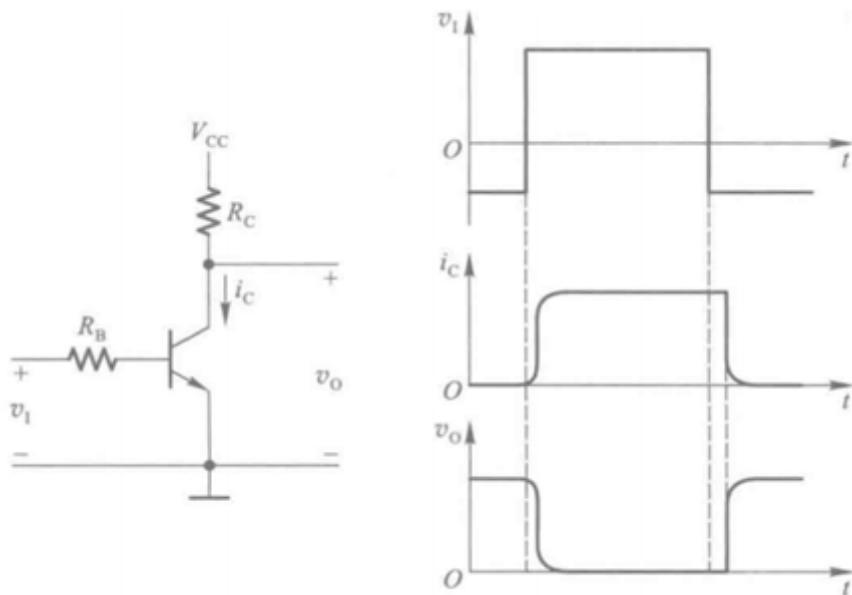


图 3.4.7 双极型三极管的动态开关特性

由于结电容的存在，电压的变化会出现滞后效应

(4) TTL反相器的电路结构和工作原理

- 电路结构

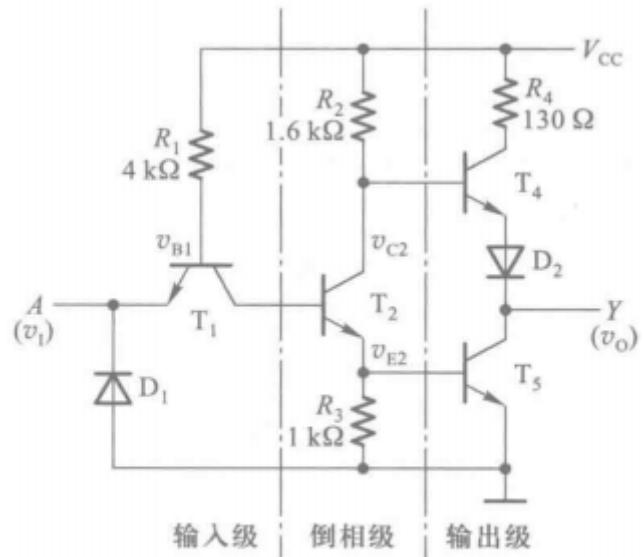


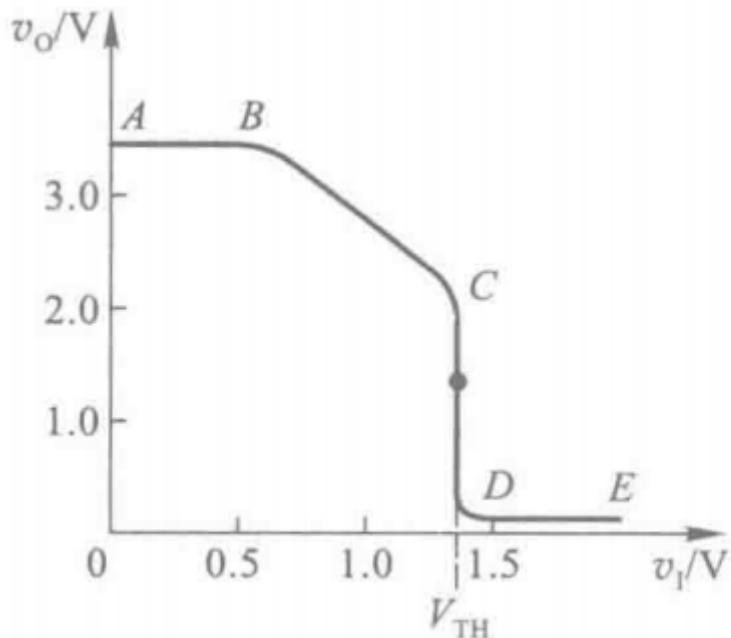
图 3.4.9 TTL 反相器的典型电路

T1、R1、D1组成输入级，T2、R2、R3组成倒相级，T4、T5、D2、R4组成输出级。D1起保护作用。

分析看课本

输出 $Y = A'$

- 电压传输特性



0~0.6 截止区 大于1.4 饱和区

- 输入端噪声容限

与CMOS反相器类似, $V_{nh} = V_{ohmin} - V_{ihmin}$ $V_{nl} = V_{ilmax} - V_{olmax}$

(5) TTL反相器的静态输入特性和输出特性

- 输入特性

仅考虑输入信号时高电平和低电平

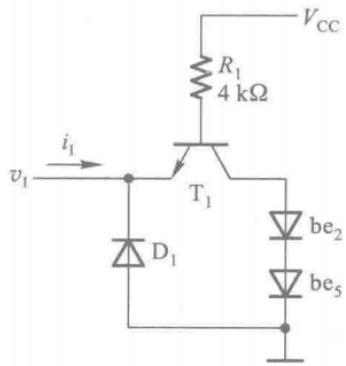


图 3.4.11 TTL 反相器的输入端等效电路

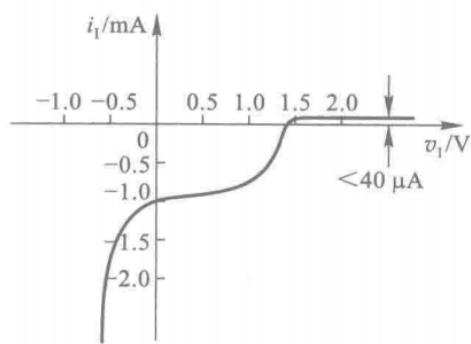


图 3.4.12 TTL 反相器的输入特性

- 输出特性

高电平输出特性：

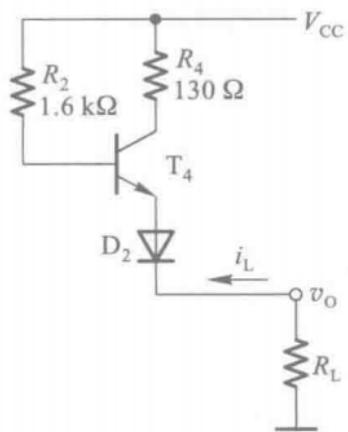


图 3.4.13 TTL 反相器高电平输出等效电路

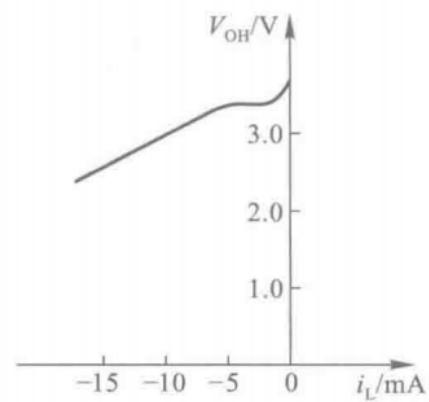


图 3.4.14 TTL 反相器高电平输出特性

低电平输出特性：

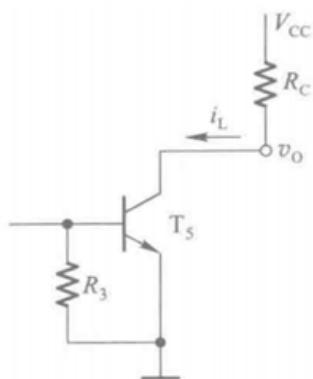


图 3.4.15 TTL 反相器低电平输出等效电路

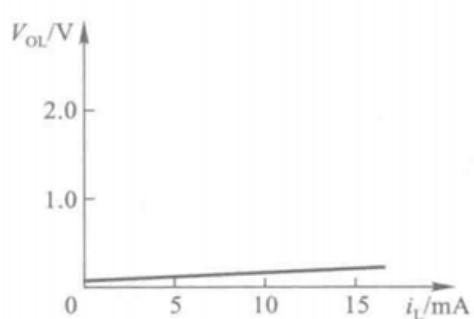


图 3.4.16 TTL 反相器低电平输出特性

- 输入端负载特性

在输入端与地之间接入电阻 R_p 。算了摆烂。。其实早就摆了

(6) TTL反相器的动态特性

- 传输延迟时间

摆了。。

四、组合逻辑电路

1. 概述

- 组合逻辑电路的特点
 - 从功能上：任意时刻的输出仅取决于该时刻的输入
 - 从电路结构上：不含记忆存储元件
- 逻辑功能的描述

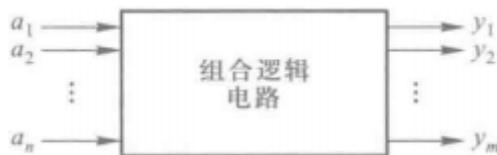


图 4.1.2 组合逻辑电路的框图

A combinational device is a circuit element that has:

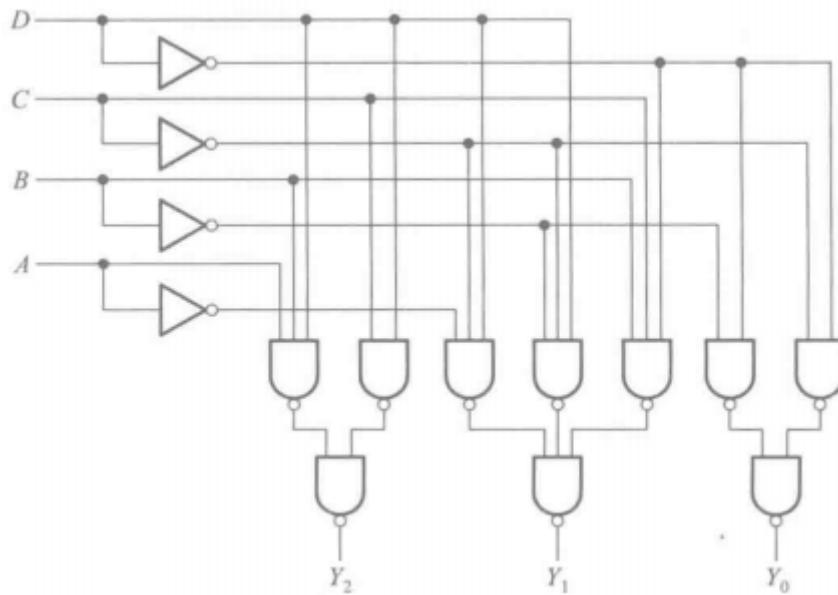
1. one or more digital inputs
2. one or more digital outputs
3. a functional specification that details the value of each output for every possible combination of valid input values
4. a timing specification consisting(at minimum) of an upper bound tpd on the required time for the device to compute the specified output values from an arbitrary set of stable, valid input values

2. 组合逻辑电路分析方法

所谓分析一个给定的组合逻辑电路，就是要通过分析找出电路的逻辑功能，通常采用的分析方法是从电路的输入到输出逐级写出逻辑函数式，然后用公式化简法或卡诺图化简法将得到的函数式化简或变换，以使逻辑关系简单明了，有时可以列出真值表。

根据给出的逻辑图可写出 Y_2, Y_1, Y_0 和 D, C, B, A 之间关系的逻辑式

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



3. 组合逻辑电路的基本设计方法

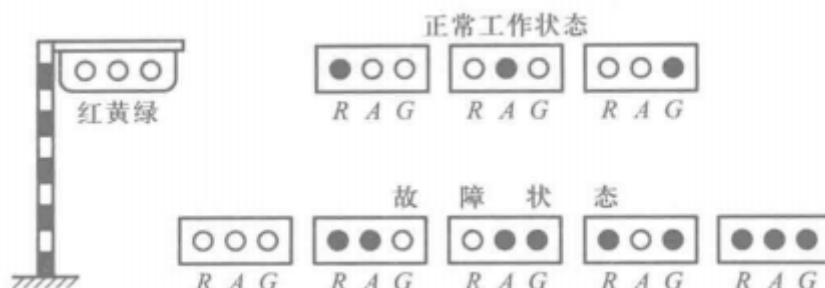
设计目标：完成实现这一逻辑功能的最简逻辑电路

- 逻辑抽象
 - 分析因果关系，确定输入输出变量
 - 定义逻辑状态的含义，赋值
 - 列出真值表
- 写出逻辑函数式
- 选定器件类型
 - 门电路、中规模集成的常用组合逻辑器件、大规模集成的可编程逻辑器件
- 根据所选器件
 - 对逻辑式进行化简：门
 - 变换：MSI
 - 进行相应的描述：PLD
- 画出逻辑电路图，或下载到PLD
- 工艺设计

设计举例

- 设计一个监视交通信号灯状态的逻辑电路

出现故障状态时，要求发出故障信号：



进行逻辑抽象：红R，黄Y，绿G，灯亮为1，不亮为0，正常状态下Z为0，发生故障为1.

列出真值表：

R	A	G	Z
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

根据真值表列出逻辑函数式：

$$Z = R' * A' * G' + R' * A * G + R * A' * G + R * A * G' + R * A * G$$

化简：

$$Z = R' * A' * G' + R * A + R * G + A * G$$

画出逻辑电路

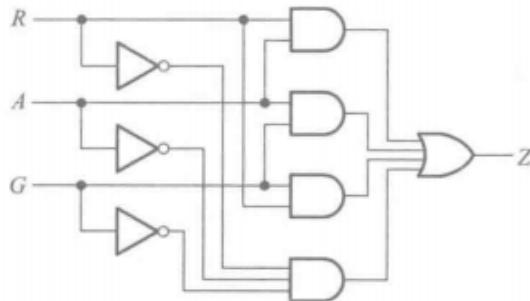


图 4.3.3 例 4.3.1 的逻辑图之一

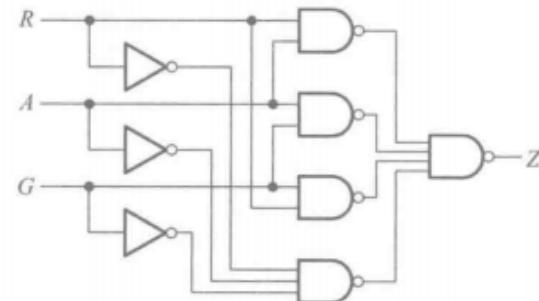


图 4.3.4 例 4.3.1 的逻辑图之二

4. 若干常用组合逻辑电路

(1) 编码器

- 编码：将输入的每个高低电平信号变成一个对应的二进制编码
- 普通编码器

同一时间只允许输入一个编码信号

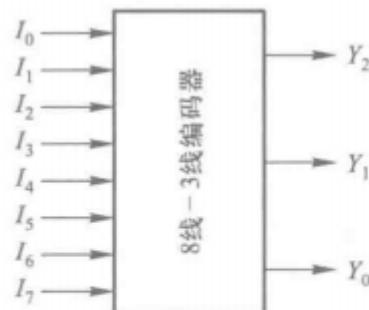


图 4.4.1 3 位二进制
(8 线-3 线) 编码器的框图

真值表：

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

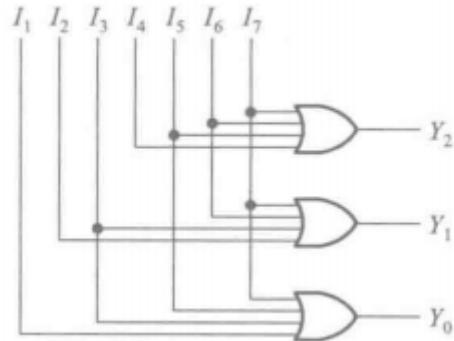
直接得出逻辑函数式

$$\left\{ \begin{array}{l} Y_2 = I'_0 I'_1 I'_2 I'_3 I'_4 I'_5 I'_6 I'_7 + I'_0 I'_1 I'_2 I'_3 I'_4 I'_5 I'_6 I'_7 \\ \quad + I'_0 I'_1 I'_2 I'_3 I'_4 I'_5 I'_6 I'_7 + I'_0 I'_1 I'_2 I'_3 I'_4 I'_5 I'_6 I'_7 \\ Y_1 = I'_0 I'_1 I'_2 I'_3 I'_4 I'_5 I'_6 I'_7 + I'_0 I'_1 I'_2 I'_3 I'_4 I'_5 I'_6 I'_7 \\ \quad + I'_0 I'_1 I'_2 I'_3 I'_4 I'_5 I'_6 I'_7 + I'_0 I'_1 I'_2 I'_3 I'_4 I'_5 I'_6 I'_7 \\ Y_0 = I'_0 I'_1 I'_2 I'_3 I'_4 I'_5 I'_6 I'_7 + I'_0 I'_1 I'_2 I'_3 I'_4 I'_5 I'_6 I'_7 \\ \quad + I'_0 I'_1 I'_2 I'_3 I'_4 I'_5 I'_6 I'_7 + I'_0 I'_1 I'_2 I'_3 I'_4 I'_5 I'_6 I'_7 \end{array} \right.$$

逻辑函数式化简

$$\left\{ \begin{array}{l} Y_2 = I_4 + I_5 + I_6 + I_7 \\ Y_1 = I_2 + I_3 + I_6 + I_7 \\ Y_0 = I_1 + I_3 + I_5 + I_7 \end{array} \right.$$

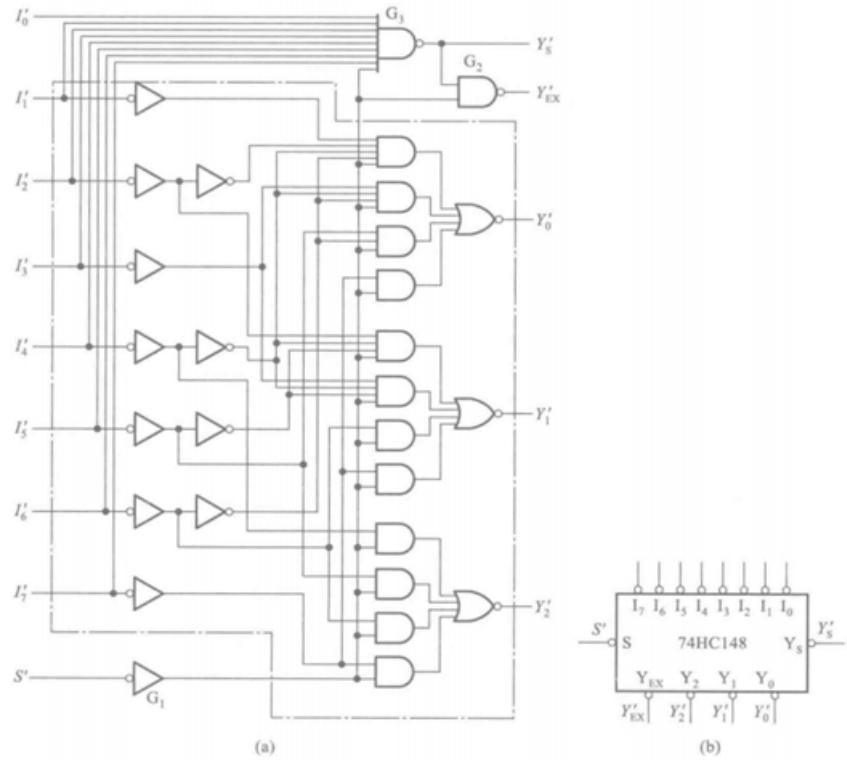
逻辑电路图



• 优先编码器

允许同时输入两个以上的编码信号，设计优先编码器时已经将所有的输入信号按优先顺序排了队，当几个输入信号同时出现时，只对其中优先权中最高的一个进行编码

虚线框外的部分为附加控制电路



真值表：

S'	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	Y'_S	Y'_EX
1	x	x	x	x	x	x	x	x	1	1	1	1	1
0	1	1	1	1	1	1	1	1	1	1	1	0	1
0	x	x	x	x	x	x	x	0	0	0	0	1	0
0	x	x	x	x	x	x	0	1	0	0	1	1	0
0	x	x	x	x	x	0	1	1	0	1	0	1	0
0	x	x	x	x	0	1	1	1	0	1	1	1	0
0	x	x	x	0	1	1	1	1	1	0	0	1	0
0	x	x	0	1	1	1	1	1	1	0	1	1	0
0	x	0	1	1	1	1	1	1	1	1	0	1	0
0	0	1	1	1	1	1	1	1	1	1	1	1	0

写出逻辑式：

$$\begin{cases} Y'_2 = ((I_4 + I_5 + I_6 + I_7)S)' \\ Y'_1 = ((I_2 I'_4 I'_5 + I_3 I'_4 I'_5 + I_6 + I_7)S)' \\ Y'_0 = ((I_1 I'_2 I'_4 I'_6 + I_3 I'_4 I'_6 + I_5 I'_6 + I_7)S)' \end{cases}$$

附加虚线外由门G1、G2、G3组成的控制电路，S'为选通输入端，只有在S' = 0的情况下，编码器才能正常工作，在S' = 0时，所拥有的输出端均为高电平

选通输出端YS'和扩展端Yex'用于扩展编码功能，由图可知：

$$Y'_S = (I'_0 I'_1 I'_2 I'_3 I'_4 I'_5 I'_6 I'_7 S)'$$

表示：电路工作，但无编码输入

$$\begin{aligned}
 Y'_{\text{EX}} &= ((I'_0 I'_1 I'_2 I'_3 I'_4 I'_5 I'_6 I'_7 S)' S)' \\
 &= ((I_0 + I_1 + I_2 + I_3 + I_4 + I_5 + I_6 + I_7) S)'
 \end{aligned}$$

表示：电路工作，且有编码输入

优先：在 $S' = 0$ 时，电路正常工作，允许7个输入中同时有几个输入端为低电平， $I7'$ 的优先权最高， $I0'$ 的优先权最低，当 $I7' = 0$ 时，无论其他输入如何，输出端只给 $I7'$ 的编码，即 $Y2'Y1'Y0' = 000$ 。当 $I7' = 2, I6' = 0$ 时，无论其余输入端有无输入信号，只对 $I6'$ 编码，输出为 $Y2'Y1'Y0' = 001$ 。……。

表中出现的三种 $Y2' * Y1' * Y01 = 111$ 的情况可以用 YS' 和 YEX' 的不同状态加以区分

(2) 译码器

译码器的逻辑功能是将每个输入的二级制代码译成对应的输出高、低电平信号或另外一个代码，译码是编码的反操作。

- 二进制译码器：输入二进制代码，输出高低电平信号
- 38译码器

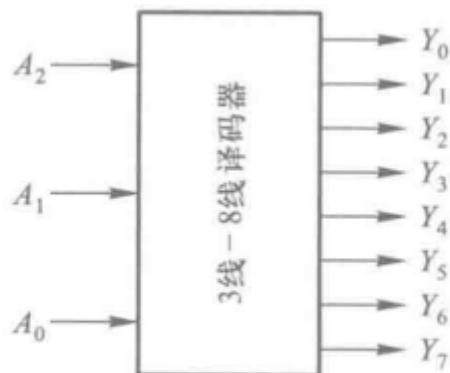


图 4.4.5 3 位二进制

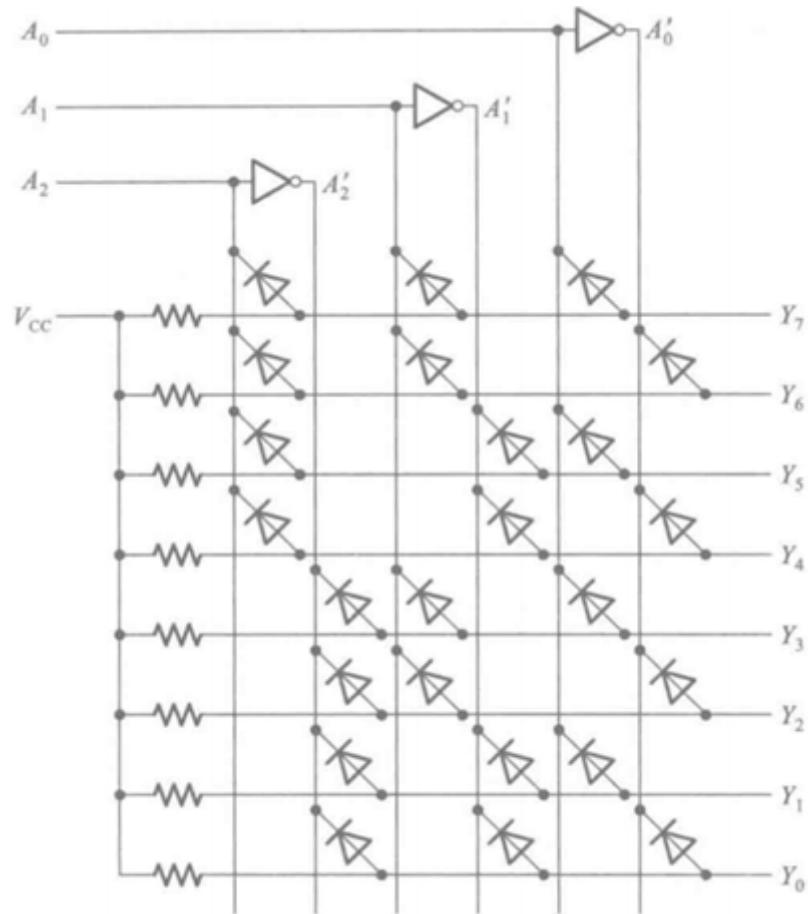


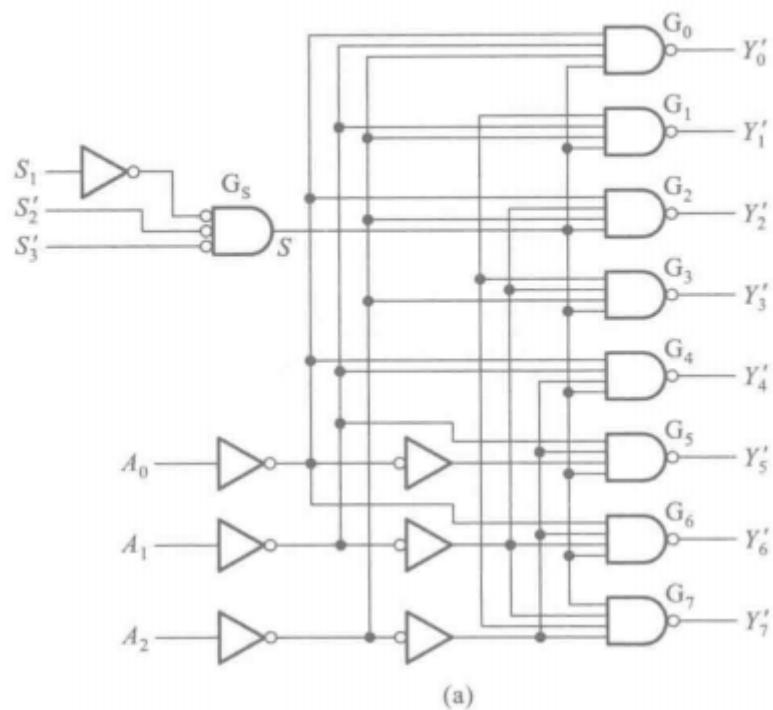
图 4.4.6 用二极管与门阵列组成的 3 线-8 线译码器

当某一输出端的三个输入值中有一个值为0时，相当于接地，使该输出端输出为0，当某一输出端的三个输入值均为1时，输出高电平3.7V，而输入高电平为3V，输出的高低电平会发生偏移。输入电阻低，输出电阻高。

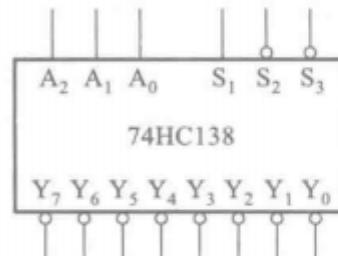
真值表：

输入			输出							
A_2	A_1	A_0	Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0

与非门38译码器



(a)



(b)

图 4.4.7 用与非门组成的 3 线-8 线译码器 74HC138

(a) 内部逻辑图 (b) 逻辑框图

输入					输出							
S_1	$S'_2 + S'_3$	A_2	A_1	A_0	Y'_0	Y'_1	Y'_2	Y'_3	Y'_4	Y'_5	Y'_6	Y'_7
0	x	x	x	x	1	1	1	1	1	1	1	1
x	1	x	x	x	1	1	1	1	1	1	1	1
1	0	0	0	0	0	1	1	1	1	1	1	1
1	0	0	0	1	1	0	1	1	1	1	1	1

1	0	0	1	0	1	1	0	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1	1	1
1	0	1	0	0	1	1	1	1	0	1	1	1
1	0	1	0	1	1	1	1	1	1	0	1	1
1	0	1	1	0	1	1	1	1	1	1	0	1
1	0	1	1	1	1	1	1	1	1	1	1	0

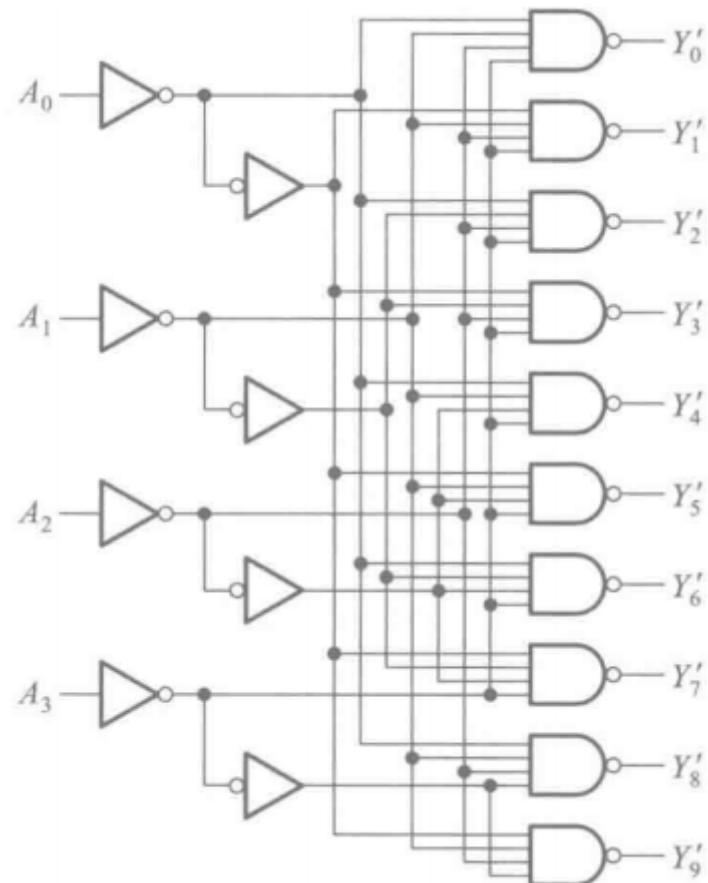
三个S端是附加的控制端 $S = 1$ 时，译码器处于工作状态，否则译码器被禁止，所以输出均为高电平，正常工作时：

$$\left\{ \begin{array}{l} Y'_0 = (A'_2 A'_1 A'_0)' = m'_0 \\ Y'_1 = (A'_2 A'_1 A_0)' = m'_1 \\ Y'_2 = (A'_2 A_1 A'_0)' = m'_2 \\ Y'_3 = (A'_2 A_1 A_0)' = m'_3 \\ Y'_4 = (A_2 A'_1 A'_0)' = m'_4 \\ Y'_5 = (A_2 A'_1 A_0)' = m'_5 \\ Y'_6 = (A_2 A_1 A'_0)' = m'_6 \\ Y'_7 = (A_2 A_1 A_0)' = m'_7 \end{array} \right.$$

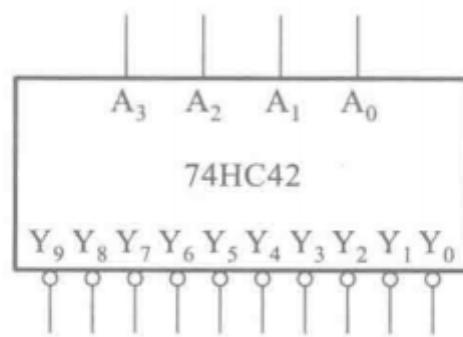
可以看出，Y0'~Y1'是A2、A1、A0的三个变量的最小项的译码输出，所以也叫最小项译码器

- 二—十进制译码器

将输入的BCD码的是个代码译成10个高低电平输出信号



(a)



(b)

根据逻辑图可得到：

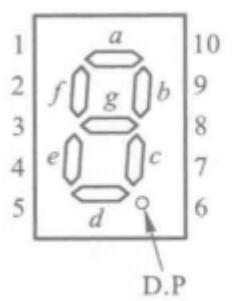
$$\begin{cases} Y'_0 = (A'_3 A'_2 A'_1 A'_0)' & Y'_5 = (A'_3 A_2 A'_1 A_0)' \\ Y'_1 = (A'_3 A'_2 A'_1 A_0)' & Y'_6 = (A'_3 A_2 A_1 A'_0)' \\ Y'_2 = (A'_3 A'_2 A_1 A'_0)' & Y'_7 = (A'_3 A_2 A_1 A_0)' \\ Y'_3 = (A'_3 A'_2 A_1 A_0)' & Y'_8 = (A_3 A'_2 A'_1 A'_0)' \\ Y'_4 = (A'_3 A_2 A'_1 A'_0)' & Y'_9 = (A_3 A'_2 A'_1 A_0)' \end{cases}$$

列出真值表：

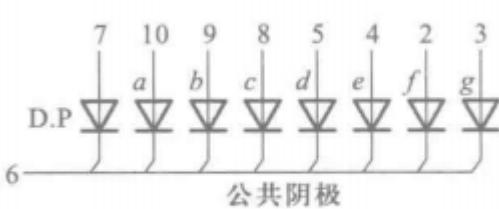
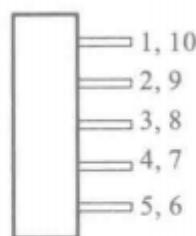
序号	输入				输出									
	A_3	A_2	A_1	A_0	Y'_0	Y'_1	Y'_2	Y'_3	Y'_4	Y'_5	Y'_6	Y'_7	Y'_8	Y'_9
0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
1	0	0	0	1	1	0	1	1	1	1	1	1	1	1
2	0	0	1	0	1	1	0	1	1	1	1	1	1	1
3	0	0	1	1	1	1	1	0	1	1	1	1	1	1
4	0	1	0	0	1	1	1	1	0	1	1	1	1	1
5	0	1	0	1	1	1	1	1	1	0	1	1	1	1
6	0	1	1	0	1	1	1	1	1	1	0	1	1	1
7	0	1	1	1	1	1	1	1	1	1	1	0	1	1
8	1	0	0	0	1	1	1	1	1	1	1	1	0	1
9	1	0	0	1	1	1	1	1	1	1	1	1	1	0
伪 码	1	0	1	0	1	1	1	1	1	1	1	1	1	1
	1	0	1	1	1	1	1	1	1	1	1	1	1	1
	1	1	0	0	1	1	1	1	1	1	1	1	1	1
	1	1	0	1	1	1	1	1	1	1	1	1	1	1
	1	1	1	0	1	1	1	1	1	1	1	1	1	1
	1	1	1	1	1	1	1	1	1	1	1	1	1	1

对于BCD以外的伪码，译码器不产生低电平信号，拒绝翻译

- 七段字符显示器



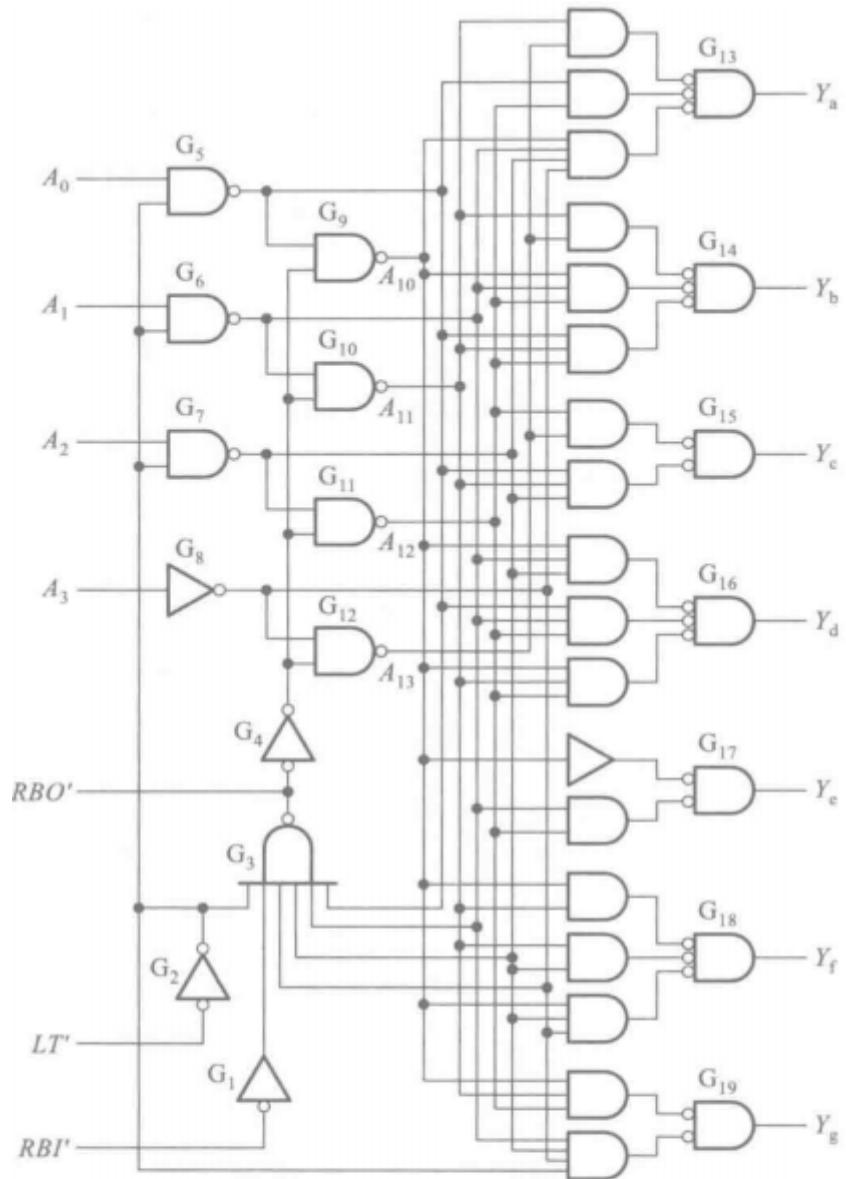
(a)



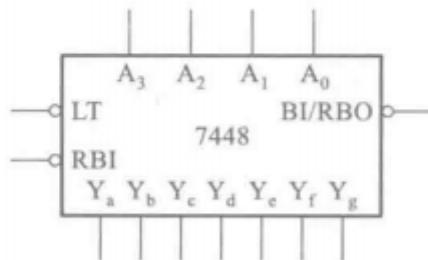
(b)

输入					输出							
数字	A_3	A_2	A_1	A_0	Y_a	Y_b	Y_c	Y_d	Y_e	Y_f	Y_g	字形
0	0	0	0	0	1	1	1	1	1	1	0	□
1	0	0	0	1	0	1	1	0	0	0	0	一
2	0	0	1	0	1	1	0	1	1	0	1	二
3	0	0	1	1	1	1	1	1	0	0	1	三
4	0	1	0	0	0	1	1	0	0	1	1	四
5	0	1	0	1	1	0	1	1	0	1	1	五
6	0	1	1	0	0	0	1	1	1	1	1	六
7	0	1	1	1	1	1	1	0	0	0	0	七
8	1	0	0	0	1	1	1	1	1	1	1	八
9	1	0	0	1	1	1	1	0	0	1	1	九

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



(a)



(b)

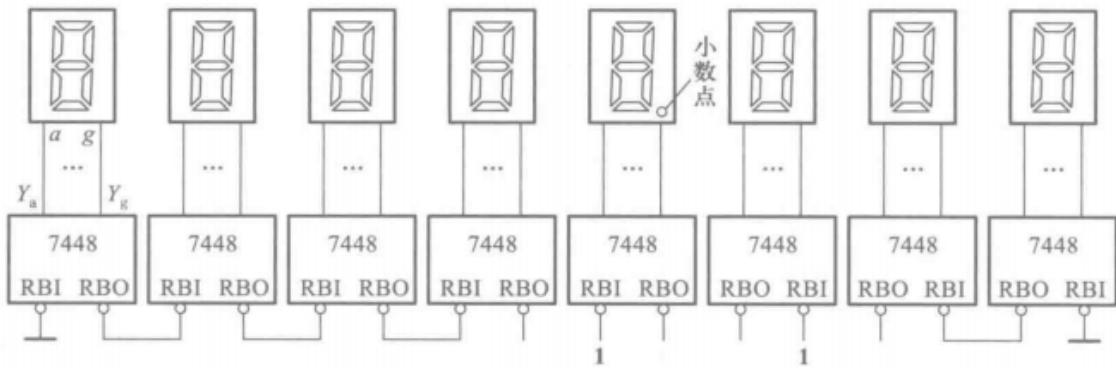
图 4.4.14 BCD-七段显示译码器 7448 的逻辑图

(a) 内部逻辑图 (b) 逻辑框图

$LT' = 0$ 时，所有输出均为高电平，使七段数码管同时点亮，以检查该数码管各段能否正常发光

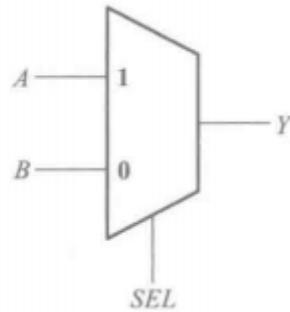
$RBI' = 0$ ，且输入的A均为0时本应该输出的数字0熄灭，实现0熄灭

BI'/RBO'是灭零输入控制端/灭零输出端，将灭零输入控制端和灭零输出端配合使用即可实现多位数码管显示系统的灭零控制



(3) 数据选择器

二选一数据选择器，从两路输入中选择一路输出

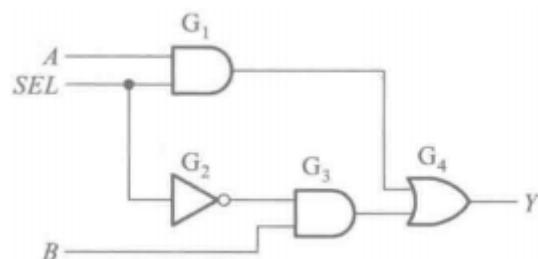


真值表：

SEL	A	B	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

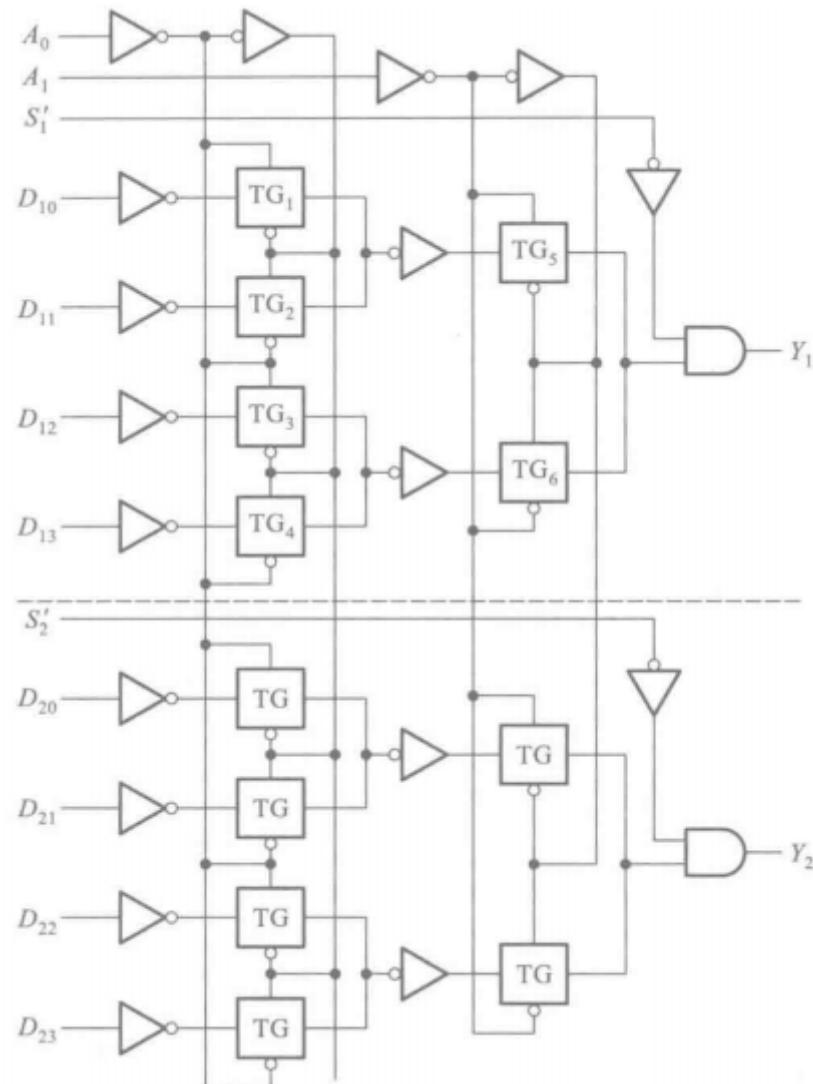
二选一数据选择器表达式： $Y = SEL * A + SEL' * B$

电路图：



(双) 四选一数据选择器

从四个输入端中选择一个输出



上半部分：当 $A_0 = 0$ 时，TG1和TG3导通， $A_0 = 1$ 时，TG2和TG4导通，当 $A_1 = 0$ 时，TG5导通， $A_1 = 1$ 时，TG6导通，因此前四路输入中，只有一路可以输出

下半部分同理对于 S' ，当 $S' = 0$ 时，数据选择器工作， $S' = 1$ 时，数据选择器被禁止工作

逻辑式：

$$Y_1 = [D_{10}(A'_1 A'_0) + D_{11}(A'_1 A_0) + D_{12}(A_1 A'_0) + D_{13}(A_1 A_0)] \cdot S_1$$

(4) 加法器

两个二进制数之间的算数运算无论是加减乘除，目前在数字计算机中都是化做若干步加法运算进行的，因此，加法器是构成算术运算器的基本单元

- 一位加法器

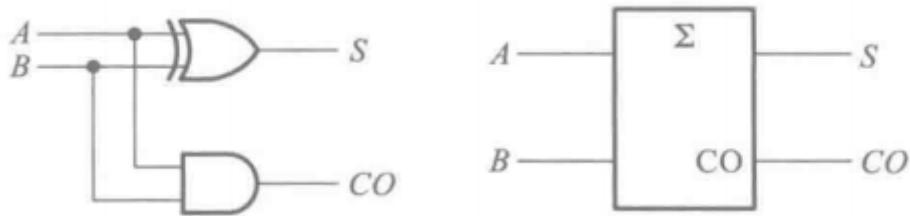
半加器：不考虑来自低位的进位将两个1为二进制数相加，称为半加。

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

逻辑式：

$$\begin{cases} S = A'B + AB' = A \oplus B \\ CO = AB \end{cases}$$

逻辑电路：



全加器：

在将两个多位二进制相加时，除了最低位以外，每一位都应该考虑来自低位的进位，即将两个对应位的加数和来自低位的进位三个数相加，称为全加

输入			输出	
CI	A	B	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

逻辑式：

$$\begin{cases} S = (A'B'CI' + AB'CI + A'BCI + ABCI')' \\ CO = (A'B' + B'CI' + A'CI')' \end{cases}$$

逻辑电路：

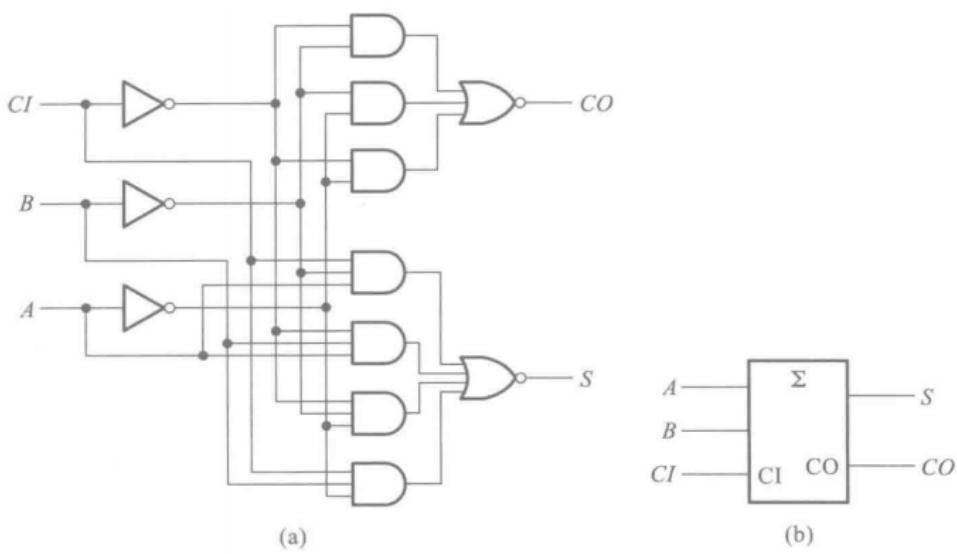


图 4.4.23 双全加器 74LS183

(a) $\frac{1}{2}$ 逻辑图 (b) 图形符号

- 多位加法器

串行进位加法器，两个多位数相加时每一位都是带进位相加的因而必须使用全加器。只要依次将低位全加器的进位输出端CO接到高位全加器的进位输出端CI就可以构成多位加法器了。

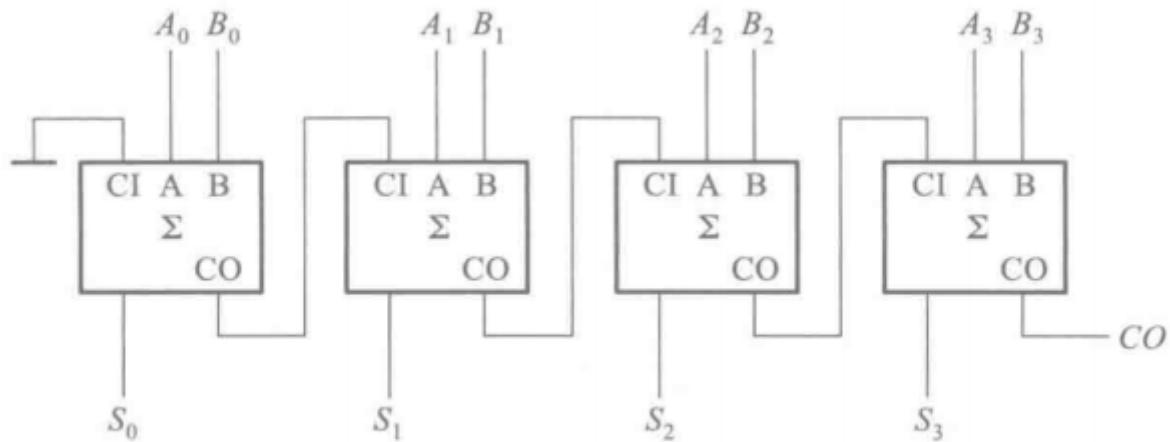


图 4.4.24 4 位串行进位加法器

缺点，传输延迟时间长，不适合高速运算，但电路结构与简单。

超前进位加法器：

通过逻辑电路事先得出每一位全加器的进位输入信号，而无需再从最低位开始向高位逐次传递进位信号，有效提高了运算速度

每一位的进位信号可表示为：

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

可改写为：

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

展开：

$$\begin{aligned} (CO)_i &= G_i + P_i (CI)_i \\ &= G_i + P_i [G_{i-1} + P_{i-1} (CI)_{i-1}] \\ &= G_i + P_i G_{i-1} + P_i P_{i-1} [G_{i-2} + P_{i-2} (CI)_{i-2}] \\ &\quad \vdots \\ &= G_i + P_i G_{i-1} + P_i P_{i-1} G_{i-2} + \cdots + P_i P_{i-1} \cdots P_1 G_0 \\ &\quad + P_i P_{i-1} \cdots P_0 (CI)_0 \end{aligned}$$

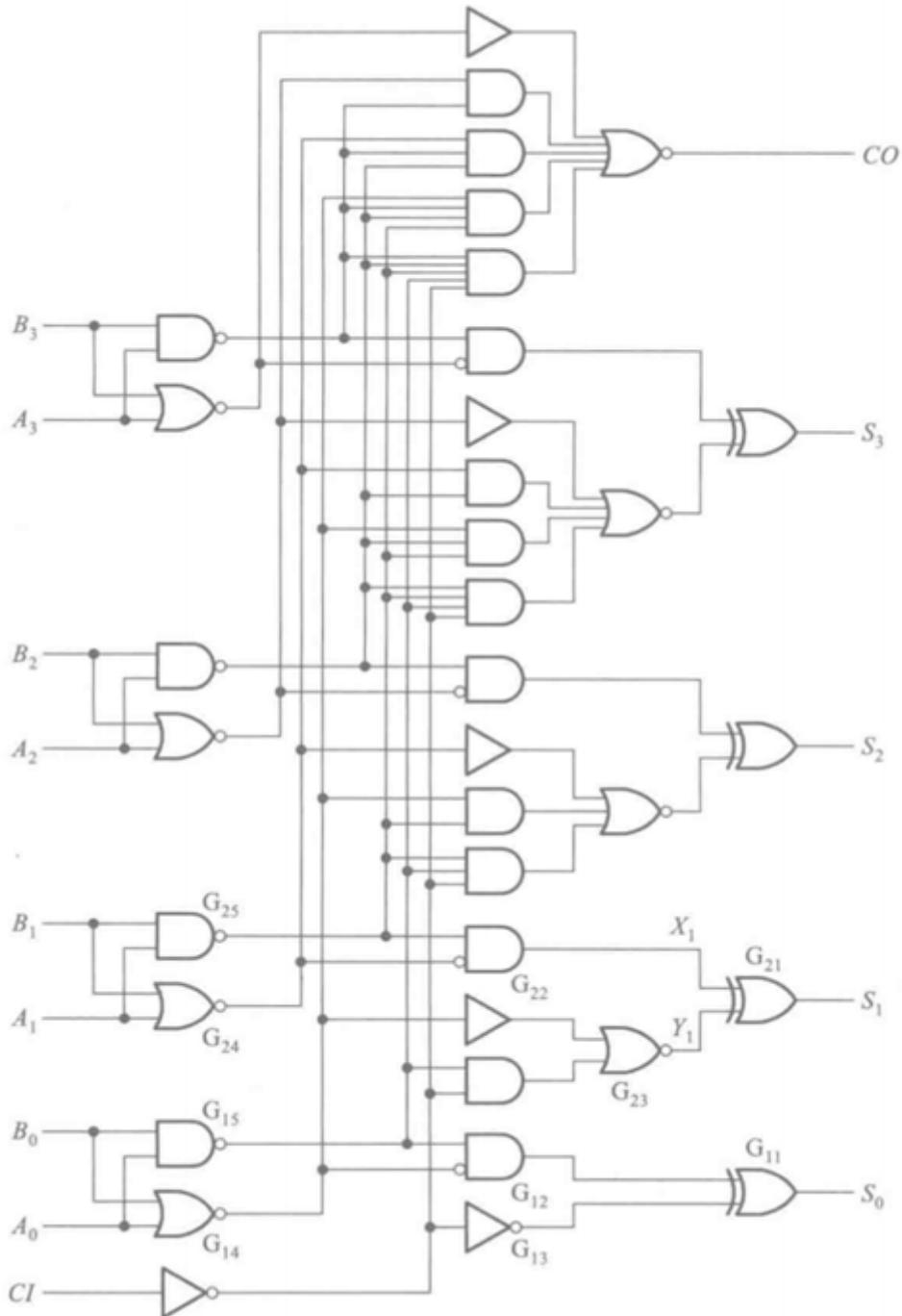
根据全加器真值表：

$$S_i = A_i B'_i (CI)'_i + A'_i B_i (CI)'_i + A'_i B'_i (CI)_i + A_i B_i (CI)_i$$

变换为异或函数：

$$\begin{aligned}
 S_i &= (A_i B'_i + A'_i B_i) (CI)'_i + (A_i B_i + A'_i B'_i) (CI)_i \\
 &= (A_i \oplus B_i) (CI)'_i + (A_i \oplus B_i)' (CI)_i \\
 &= A_i \oplus B_i \oplus (CI)_i
 \end{aligned}$$

逻辑电路图：



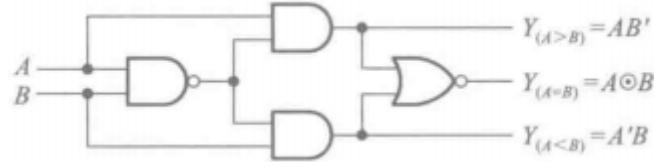
从两个加数送到输入端到完成加法运算只需三级门电路的传输延迟时间，而获得进位输出信号仅需一级反相器和以及与或非门的传输延迟时间。然而必须指出，运算时间缩短是用电路复杂程度的代价换取的，当加法器的位数增加时，电流的复杂程度也随之急剧上升

(5) 数值比较器

比较两个数值的大小

- 1位数值比较器

- $A > B$ ($A = 1, B = 0$), $A' * B = 1$, 则用 $A * B'$ 作为 $A > B$ 的输出信号
- $A < B$ ($A = 0, B = 1$), $A' * B = 1$, 则用 $A' * B$ 作为 $A < B$ 的输出信号
- $A = B$, 则用 A 同或 B 作为 $A = B$ 的输出信号



- 多位数值比较器

从高到低逐位比较, 当高位相等时, 才继续比较低位

对于四位数值比较器

可得逻辑函数式:

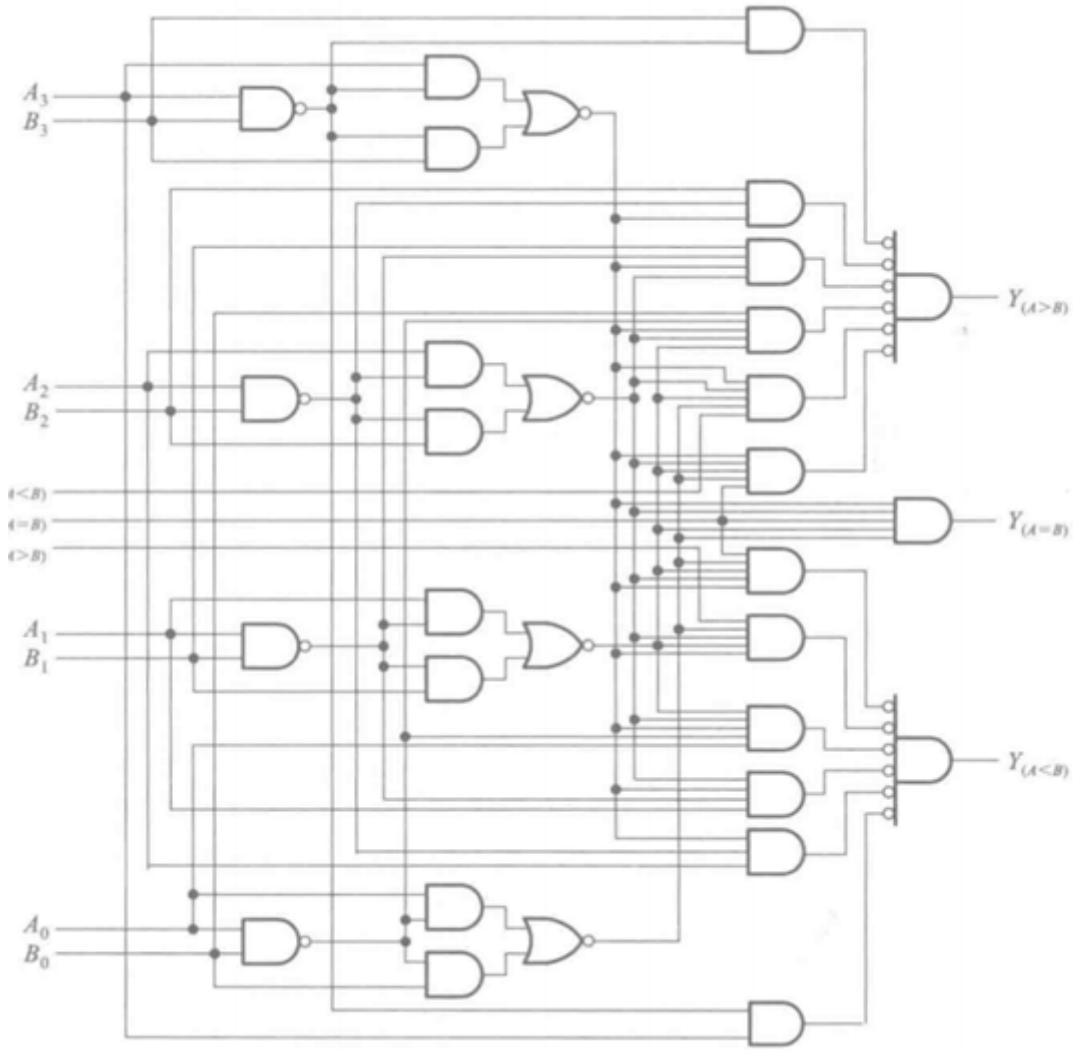
$$\begin{aligned}Y_{(A>B)} &= A_3 B'_3 + (A_3 \odot B_3) A_2 B'_2 + (A_3 \odot B_3) (A_2 \odot B_2) A_1 B'_1 \\&\quad + (A_3 \odot B_3) (A_2 \odot B_2) (A_1 \odot B_1) A_0 B'_0 \\&\quad + (A_3 \odot B_3) (A_2 \odot B_2) (A_1 \odot B_1) (A_0 \odot B_0) I_{(A>B)} \\Y_{(A<B)} &= A'_3 B_3 + (A_3 \odot B_3) A'_2 B_2 + (A_3 \odot B_3) (A_2 \odot B_2) A'_1 B_1 \\&\quad + (A_3 \odot B_3) (A_2 \odot B_2) (A_1 \odot B_1) A'_0 B_0 \\&\quad + (A_3 \odot B_3) (A_2 \odot B_2) (A_1 \odot B_1) (A_0 \odot B_0) I_{(A<B)} \\Y_{(A=B)} &= (A_3 \odot B_3) (A_2 \odot B_2) (A_1 \odot B_1) (A_0 \odot B_0) I_{(A=B)}\end{aligned}$$

其中 I 是来自低位的比较结果, 仅用于比较位数大于4位的两个数值, 比较位数为4位时, 三个 I 分别取0, 0, 1;

由于比较结果只有三种可能:

$$\begin{aligned}Y_{(A>B)} &= (Y_{(A<B)} + Y_{(A=B)})' \\Y_{(A<B)} &= (Y_{(A>B)} + Y_{(A=B)})'\end{aligned}$$

逻辑电路图:



5. 层次化和模块化的设计方法

层次化是指自上而下地对整个设计任务进行分层和分块的划分，降低每层的复杂度，简化每个模块的功能，或自底向上地对每个有限复杂度的模块进行实现或调用。模块化是指将经过设计和验证的能完成一定功能的逻辑电路封装成为模块，在后续的设计中都可反复使用。

核心是将电路逐级分解为若干个简单的模块，然后将这些模块设计好并连接起来。

具体例子参考课本

6. 可编程逻辑器件

Programmable Logic Device PLD

其逻辑功能由用户通过器件编程来设定，有些PLD集成度很高，足以满足设计一般数字系统的需要，这样就可以由设计人员自行编程而将一个数字系统“集成”在一片PLD上，作成“片上系统”（System on Chip，简称SoC）

PLD的内部结构

采用如下逻辑符号：

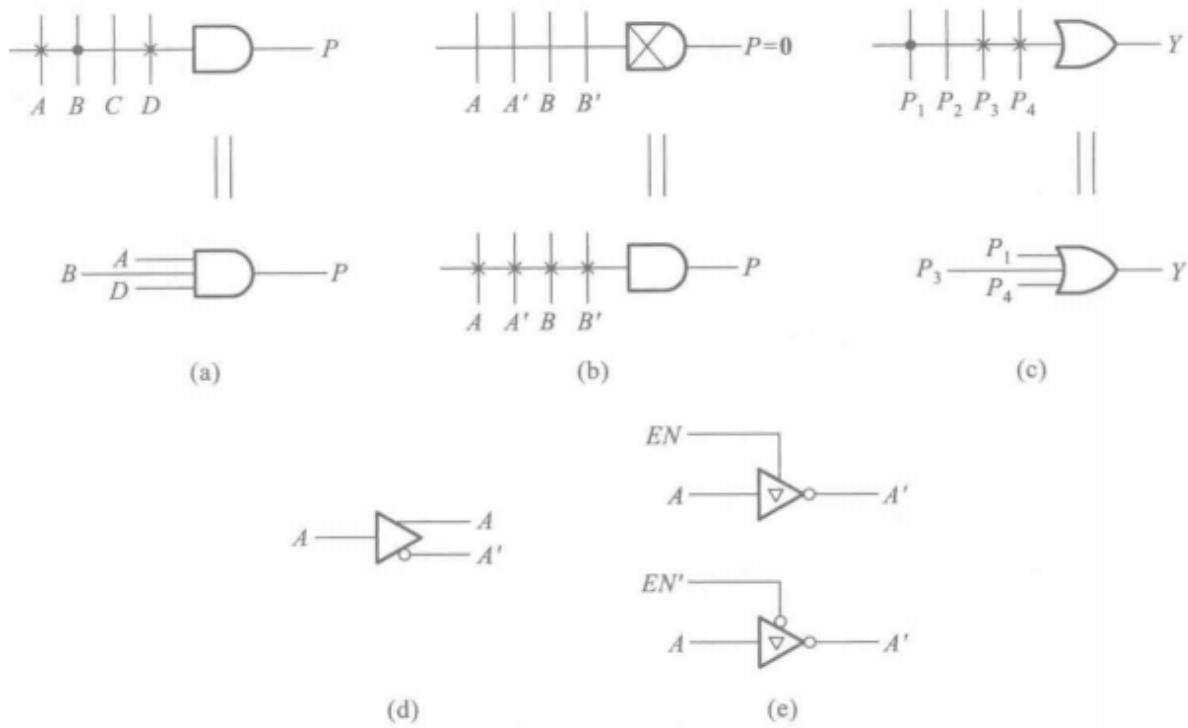


图 4.6.1 PLD 电路中门电路的惯用画法

(a) 与门 (b) 输出恒等于 0 的与门 (c) 或门 (d) 互补输出的缓冲器 (e) 三态输出的缓冲器

任何一个逻辑式都可以变换为与或表达式，因而任何一个逻辑函数都可以用一级与逻辑电路和一级或逻辑电路来实现。

现场可编程逻辑阵列 (Programmable Logic Array, PLA)，最早使用PLD

PLA由可编程的与逻辑阵列和可编程的或逻辑阵列以及输出缓冲器组成。图中的与逻辑阵列最多可产生8个可编程的乘积项，或逻辑阵列最多能产生4个组合逻辑函数。PLA的基本结构提供了一定规模的与阵列和或阵列，在生产时并未定义逻辑功能，只是为实现一定规模的逻辑运算提供了资源和可能，用户根据设计需要对与阵列和或阵列进行编程设定，从而得到某种特定的逻辑功能，PLA是PLD电路结构中最简单的一种。

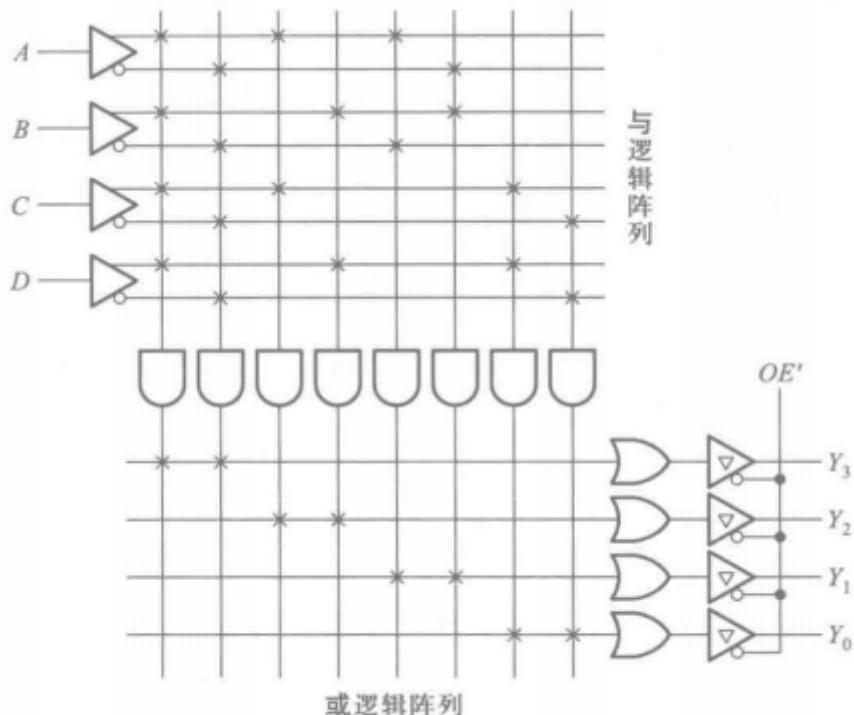


图 4.6.2 PLA 的基本电路结构

7. 硬件描述语言

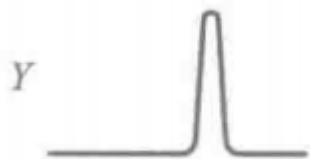
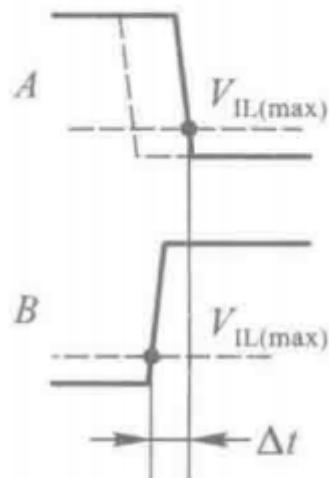
8. 用可编程通用模块设计组合逻辑电路

以上两节另外作专门的系统学习

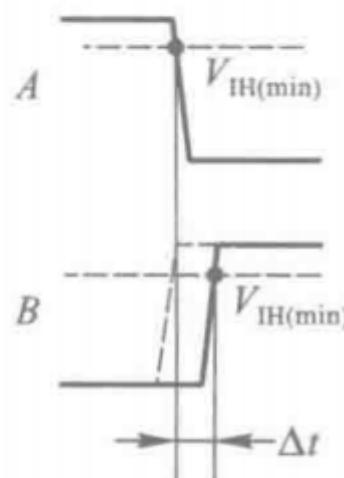
9. 组合逻辑电路中的竞争—冒险

(1) 竞争—冒险现象及成因

- 竞争：两个输入“同时向相反的逻辑电平变化”，称存在“竞争”，只要存在竞争现象，输出就有可能出现违背稳态下逻辑关系的尖峰脉冲



(a)



(b)

图 4.9.1 由于竞争而产生的尖峰脉冲

- 由于竞争而在电路输出端可能产生尖峰脉冲的现象就称为竞争—冒险

产生的尖峰脉冲可能会使负载电路产生误动作，应在设计时采取措施加以避免

(2) 检查竞争—冒险现象的方法

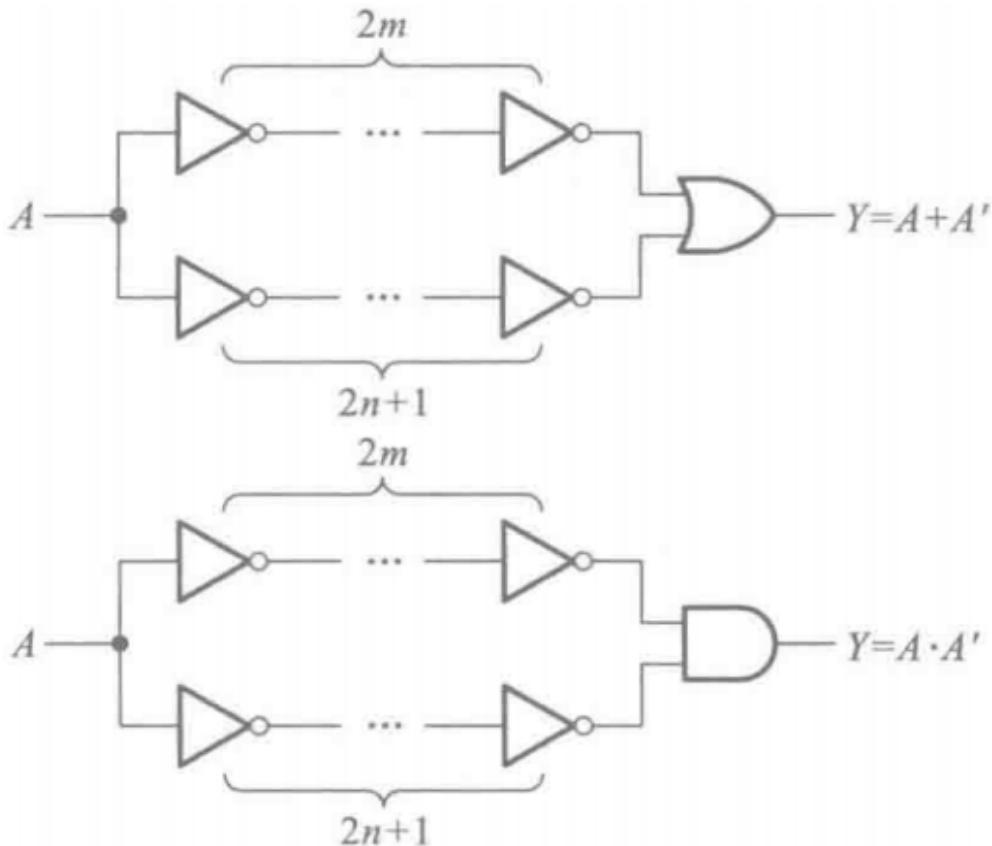
在输入变量每次只有一个改变状态的简单情况下，可以通过逻辑函数式判断组合逻辑电路中是否有竞争—冒险现象存在

如果输出端门电路的两个输入信号A和A'是输入变量经过两个不同的传输途径而来的，那么当输入变量A的状态发生突变时输出端便有可能产生尖端脉冲，因此，只要输出端的逻辑函数在一定条件下能简化成

$$Y = A + A' \text{ 或 } Y = A \cdot A'$$

则可判定存在竞争冒险现象

例如：



输出端换为或非门、与非门同样存在竞争冒险现象

局限性：当输入变量的数目很多，很难从逻辑函数式上找出所有产生竞争—冒险现象的情况。

可以通过计算机辅助手段，模拟仿真数字电路，迅速查出电路是否会存在竞争—冒险现象

(3) 消除竞争—冒险现象的方法

- 接入滤波电容

尖峰脉冲很窄，用很小的电容就可将尖峰削弱到 V_{th} 以下

优点是简单易行，缺点是增加了输出电压波形的上升时间和下降时间，使波形变坏

- 引入选通脉冲

在电路中引入一个选通脉冲 p ，取选通脉冲作用时间， p 的高电平出现在在电路达到温稳定之后，所以在尖峰脉冲出现的时候每个G的输出端都不会出现尖峰脉冲， p 的高电平输出使G的输出避开了尖峰脉冲的影响时间。

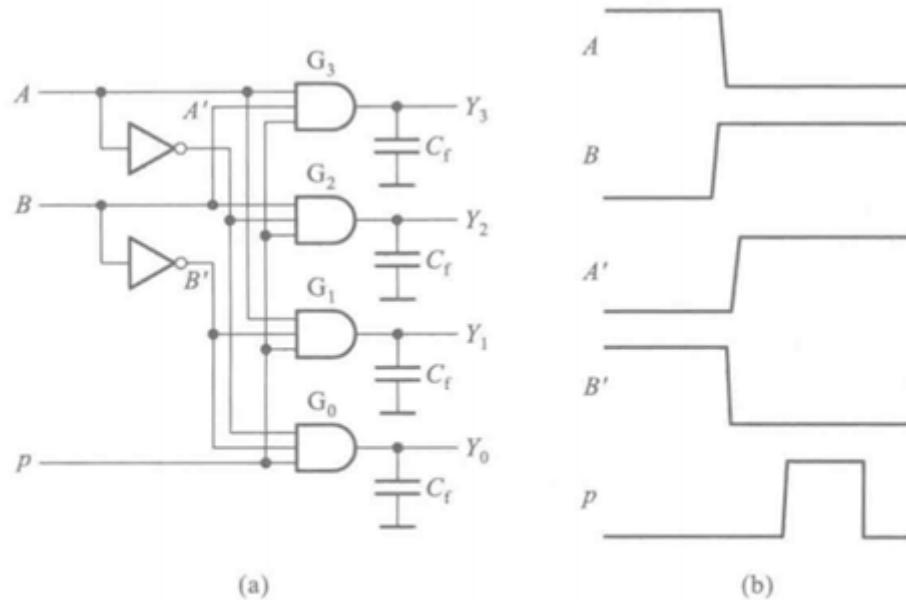


图 4.9.5 消除竞争-冒险现象的几种方法

(a) 电路接法 (b) 电压波形

- 修改逻辑设计

例: $Y = A * B + A' * C$

在 $B = C = 1$ 的条件下, $Y = A + A'$, 稳态下 $Y = 1$, 当 A 改变状态时存在竞争-冒险

根据逻辑代数常用公式可知:

$$Y = A * B + A' * C = A * B + A' * C + B * C$$

在增加了 BC 项以后, 在 $B = C = 1$ 时无论 A 如何改变, 输出始终保持 $Y = 1$, 因此 A 的变化不再会引起竞争-冒险现象, 这种方法称为“增加冗余项”

不过这种方法适用性有限, 增加冗余项之后仅仅消除了在 $B=C=1$ 时, 由于 A 的改变而产生的竞争冒险现象

五、半导体存储电路

1. 概述

在复杂电路中, 不仅需要对各种数字信号进行算数运算和逻辑运算, 而且还需要在运算过程中不断地将运算数据和运算结果保存起来, 因此, 存储电路就成为计算机以及所有复杂数字系统不可缺少的组成部分。

将用于存储一组数据的存储电路叫做寄存器 Register, 将用于存储大量数据的存储电路叫做存储器 Memory, 寄存器和半导体存储器中都包含了许多存储单元。

- 存储单元可分为静态存储单元和动态存储单元。

静态存储单元由门电路组成, 包括各种结构的锁存器和触发器, 只要不断电, 其状态就能保持下去。

动态存储单元利用电容的电荷存储效应来存储数据, 由于电容的充放电需要时间, 因此其速度会低于静态存储单元。由于电容上的电荷会泄露, 因此必须定期刷新电荷。将原来的数据重新写入, 保证数据不会丢失。

- 寄存器由一组触发器组成，每个触发器的输入和输出都有引出端，可以直接和周围电路进行数据交换，由n个触发器组成的寄存器可以存储一组n位的二值数据
- 存储器的基本结构形式都是由存储矩阵和读/写控制电路两部分组成。
- 可将存储器分为随机存储器（Random Access Memory, RAM）和只读存储器（Read-Only Memory, ROM）两类。

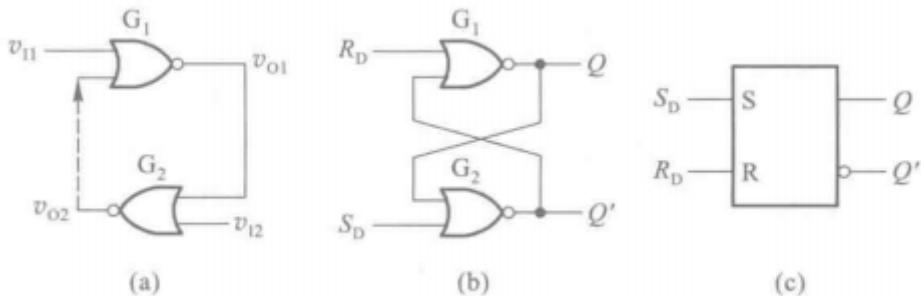
RAM的工作特点是可以随时从其中快速地读出写入数据，RAM的又可分为静态随机存储器（Static Random Access Memory, SRAM）和动态随机存储器（Dynamic Random Access Memory, DRAM），SRAM采用的是静态存储单元，DRAM采用的是动态存储单元。

ROM在正常的读/写状态下，只能从中读出所存储的数据，因此ROM一般都用来存储一些固定的数据。ROM可分为“掩模ROM”（Mask Read-Only Memory）、“可编程ROM”（Programmable Read-Only Memory, PROM）和“可擦除的可编程ROM”（Erasable Programmable Read-Only Memory, EPROM）。掩模ROM中的数据在制作芯片时已经确定，无法更改，PROM中的数据可以有用户写入，但写入后就无法更改，EPROM中的数据较为灵活，可写入可修改，例如，闪存（Flash Memory）。但擦除修改的速度仍较慢，通常将其用作只读存储器。

2. SR锁存器

最基本最简单的静态存储单元，由两个或非门或者与非门组成

(1) 或非门锁存器



Q 和 Q' 称为输出端，定义 $Q = 1$ 和 $Q' = 0$ 为锁存器的1状态， $Q = 0$ 和 $Q' = 1$ 为锁存器的0状态， S_D 称为置位端或置1输入端， R_D 称为复位端或置0输入端。

当 $S_D = 1$, $R_D = 0$ 时, $Q = 1$, $Q' = 0$, 在 $S_D = 1$ 信号消失后, 由于 Q 端仍然输出高电平, 所以 Q' 端仍然输出0, 保持1状态

当 $S_D = 0$, $R_D = 1$ 时, $Q = 0$, $Q' = 1$, 在 $R_D = 1$ 信号消失后, 由于 Q' 端仍然输出高电平, 所以 Q' 端仍然输出0, 保持0状态

当 $S_D = R_D = 0$ 时, 电路维持原来的状态不变

当 $S_D = R_D = 1$ 时, $Q = Q' = 0$, 这既不是1状态也不是0状态, 不允许输入 $S_D = R_D = 1$ 的信号

锁存器的新状态 Q^* 称为次态。与新的输入和初态 Q 有关

列出真值表：

S_D	R_D	Q	Q^*
0	0	0	0
0	0	1	1
1	0	0	1
1	0	1	1
0	1	0	0
0	1	1	0
1	1	0	0 ^①
1	1	1	0 ^②

① S_D, R_D 的 1 状态同时消失后状态不定。

(2) 与非门寄存器

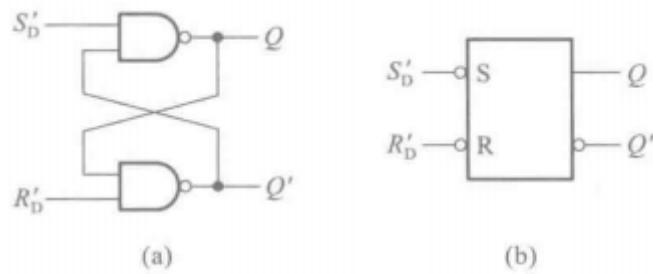


图 5.2.2 用与非门组成的 SR 锁存器

(a) 电路结构 (b) 图形符号

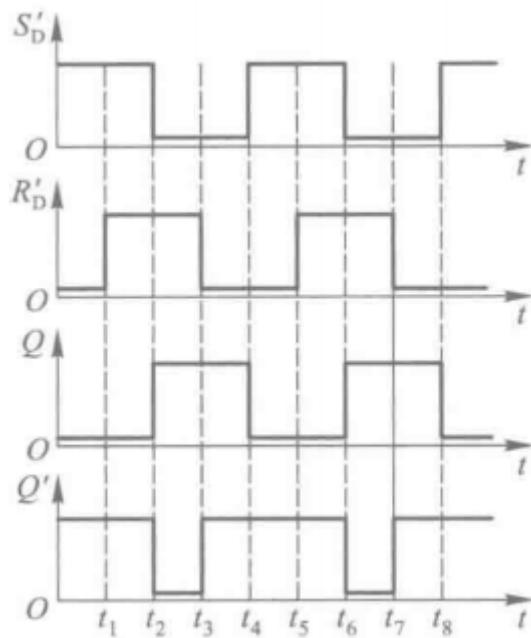
S'_D 和 R'_D 表示置1输入端和置0输入端。特性表如下：

S'_D	R'_D	Q	Q^*
1	1	0	0
1	1	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
0	0	0	1 ^①
0	0	1	1 ^②

① S'_D, R'_D 的 0 状态同时消失以后状态不定。

同样的，不应以 $S'_D = R'_D = 1$ 为输入信号

例：在与非门SR锁存器中，已知 S'_D 和 R'_D 波形如下，试画出 Q 和 Q' 端对应的波形



3. 触发器

触发器与锁存器的不同之处在于，它增加了一个触发信号输入端，只有当触发信号到来时，触发器才能按照输入的置1、置0信号置成相应状态。将这个触发信号称为时钟信号CLPCK记作CLK，当系统中有多个触发器需要同时动作时，就可以用同一个时钟信号作为同步控制信号了。

触发信号的工作方式可以分为电平触发，边沿触发和脉冲触发三种，在不同的触发方式下，触发器的动作过程各具有不同的动作特点。

逻辑功能和触发方式是触发器的两个最重要的特性，逻辑功能是指触发器的次态和初态与输入之间的关系，而触发方式是指触发器动态翻转过程中的动作特点。

(1) 电平触发的触发器

- 电路结构和工作原理

由与非门SR锁存器和与非门输入控制电路

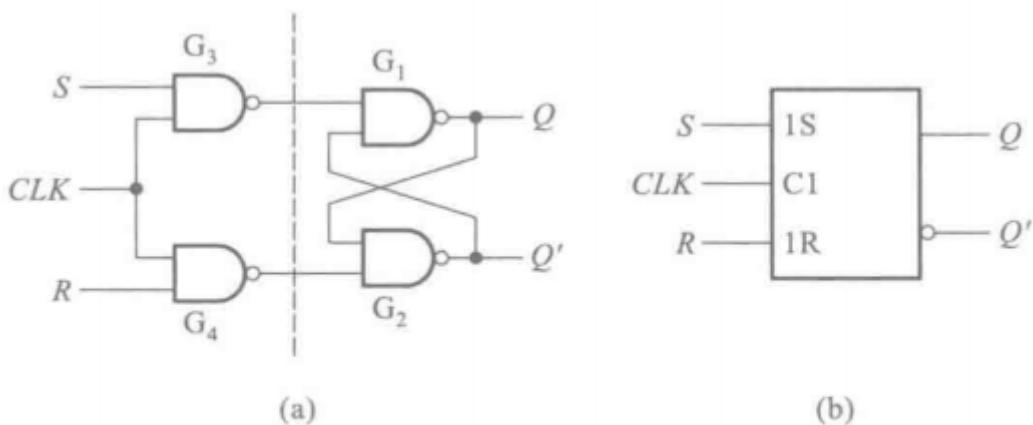


图 5.3.1 电平触发 SR 触发器(门控 SR 锁存器)

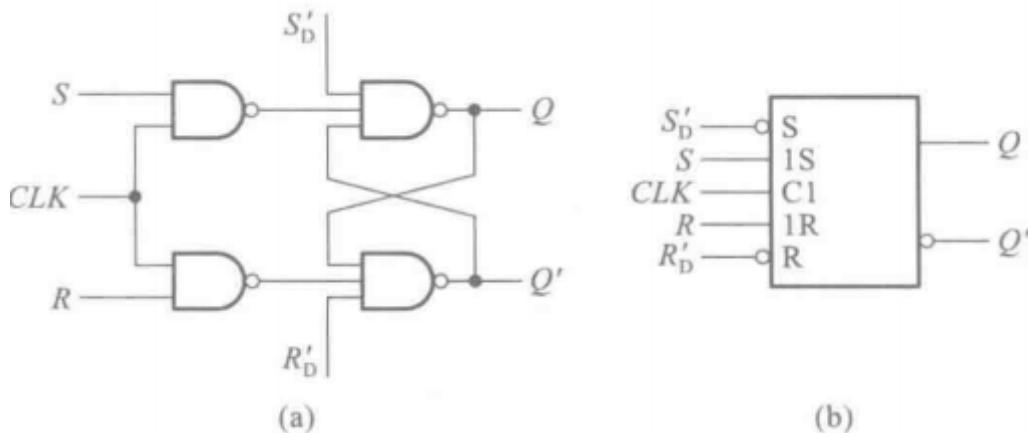
(a) 电路结构 (b) 图形符号

当 $CLK = 0$ 时， S 、 R 信号无法通过，故输出保持原来的状态不变，只有将 CLK 置为高电平， S 、 R 信号才能通过 G_4 、 G_3 ，锁存器才能正常工作。

特性表如下：

CLK	S	R	Q	Q'
0	x	x	0	0
0	x	x	1	1
1	0	0	0	0
1	0	0	1	1
1	1	0	0	1
1	1	0	1	1
1	0	1	0	0
1	0	1	1	0
1	1	1	0	1 ⁽¹⁾
1	1	1	1	1 ⁽¹⁾

在某些应用下，需要在 CLK 的有效电平到达之前预先将触发器置成指定的状态，所以在实用的电路中设置异步置1输入端 S_d' 和异步置0输入端 R_d' ，如图所示：



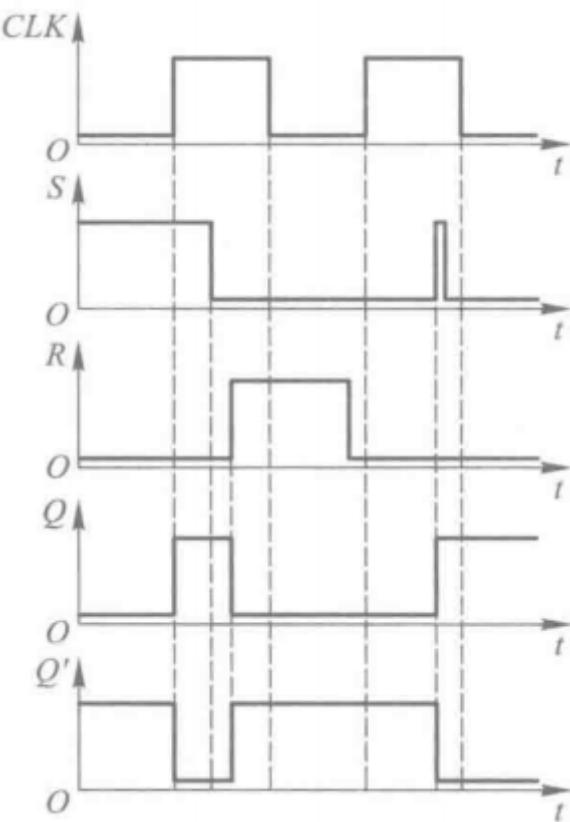
只要在 S_d' 或 R_d' 加入低电平，即可立即将触发器置1或置0，不受时钟信号的控制，触发器在时钟信号控制下正常工作时应使 S_d' 和 R_d' 处于高电平，此外，用 S_d' 或 R_d' 将触发器或复位应当在 $CLK = 0$ 的状态下进行，否则在 S_d' 或 R_d' 返回高点平以后预置的状态不一定能保存下来。

• 电平触发方式的动作特点

- 只有当 CLK 变为有效电平的时候，触发器才能接收输入信号，并按照输入信号将触发器的输出置成相应状态
- 在 $CLK = 1$ 时， S 和 R 的状态变化都可以引起输出状态的改变，在 CLK 回到0以后，触发器保存的是 CLK 回到0以前的瞬间状态

如果在 $CLK = 1$ 期间 S 、 R 的状态发生多次变化，那么触发器输出的状态也将发生多次翻转，降低了触发器的抗干扰能力

例：画出 CLK 、 S 、 R 波形已知情况下， Q 和 Q' 对应的波形：



• D触发器

为适应单端输入信号都需要，可得到如下所示D触发器

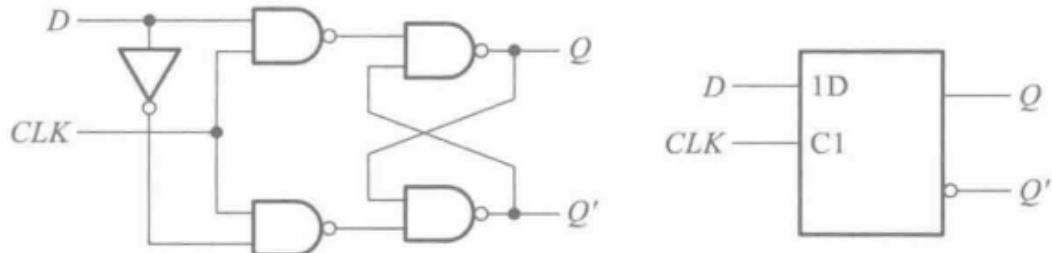


图 5.3.4 电平触发 D 触发器(D型锁存器)

特性表如下：

CLK	D	Q	Q'
0	x	0	0
0	x	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

在CMOS电路中，经常利用CMOS传输门组成电平触发D触发器，当 $CLK = 1$ 时，传输门TG1导通、TG2截止， $Q = D$ 。在 $CLK = 1$ 的全部时间里， Q 端的状态始终跟随D端的状态而改变。 $CLK = 0$ 后，TG2导通，TG1截止，由于反相器G1输入电容的存储效应，短时间内G1输入端仍然保持为TG1截止以前瞬间的状态，而且这时反相器G1、G2和传输门TG2形成了状态自锁的闭合回路， Q 和 Q' 的状态被保存下来。

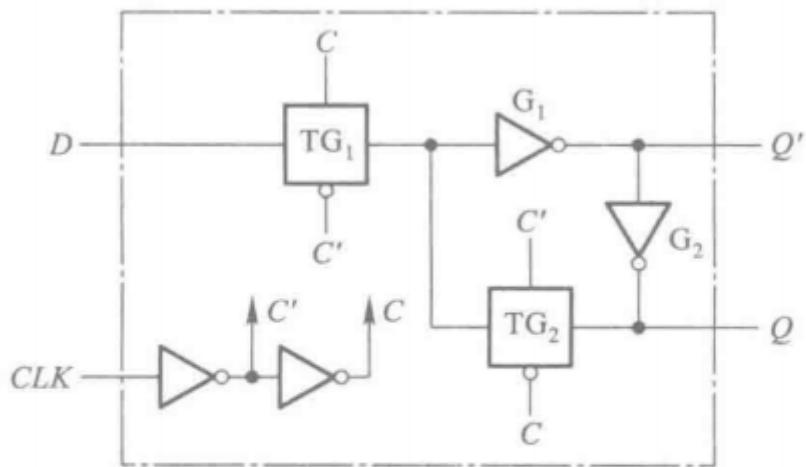
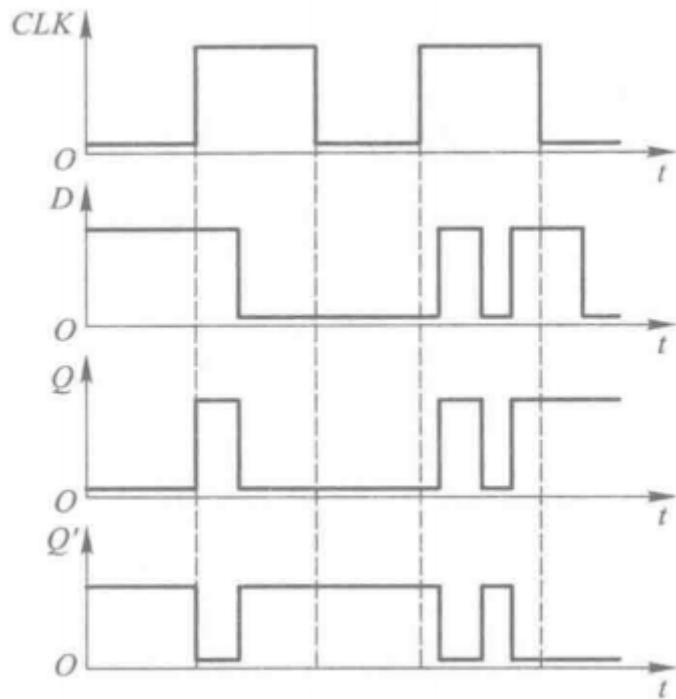


图 5.3.5 利用 CMOS 传输门组成的电平触发
D 触发器(透明 D 型锁存器)

例，根据CLK和D的波形画出Q和Q'的波形

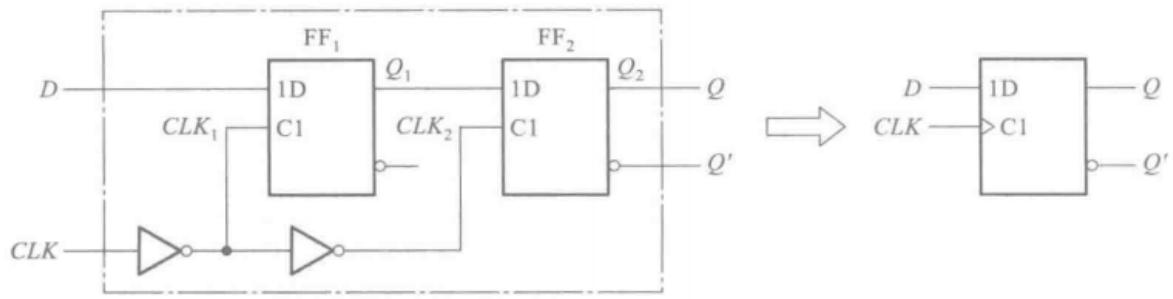


(2) 边沿触发的触发器

- 电路结构和工作原理

为提高触发器的抗干扰能力，希望触发器的次态仅仅取决于CLK信号下降沿（或上升沿）到达时刻输入信号的状态，而在此之前和之后输入状态的变化对触发器的次态没有影响。

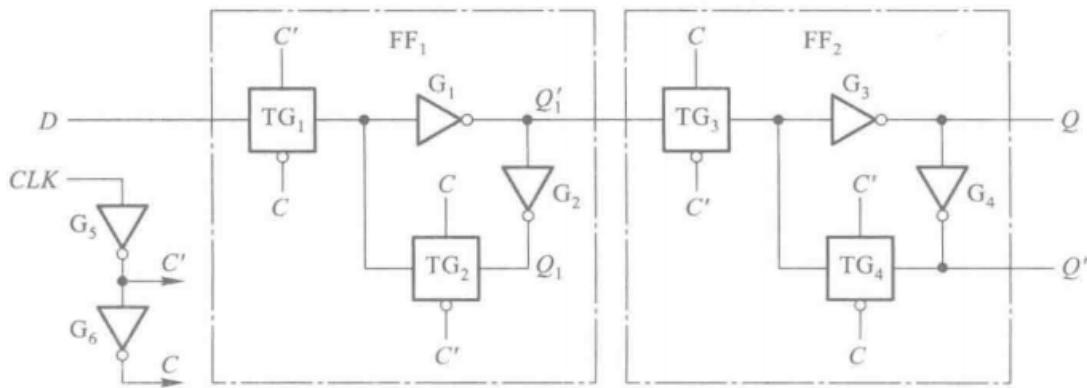
用两个电平触发D触发器组成边沿触发D触发器，如图，



当 $CLK = 0$ 时， $CLK_1 = 1$ ，所以 FF_1 的输出随 D 变化， $Q_1 = D$ ， CLK_2 为低电平， FF_2 的输出 Q_2 保持不变。

当 CLK 由0跳为1时， CLK_1 变为低电平， Q_1 保持为 CLK 上升沿到达瞬间输入端 D 的状态，**此后不再随 D 改变**，与此同时， CLK_2 跳变为高电平，使 Q_2 与其 Q_1 的状态相同，所以输出端 Q 便被置成了与 CLK 上升沿到达前瞬间 D 端相同的状态，而与以前和以后 D 端的状态无关。从而完成边沿触发

在CMOS集成电路中，采用如下电路做边沿触发器：



当 $CLK = 0$ 时， $C = 0$ ， $C' = 1$ ， TG_1 导通， TG_2 截止， $Q_1 = D$ ，由于 TG_3 截止， TG_4 导通， FF_2 保持原来的状态不变。

当 CLK 的上升沿到达时， $C = 1$ ， $C' = 0$ ， TG_1 变为截止， TG_2 变为导通，由于反相器 G_1 输入电容的存储效应， G_1 输入端的电压不会立刻改变，同时，随着 TG_4 截止， TG_3 导通， Q_1 的状态通过 TG_3 、 G_3 、 G_4 到达输出端，使 $Q^* = D$ ，从而构成上升沿触发的D触发器

如果将 CLK 输入端的一个反相器去掉，则变成下降沿触发，应该 CLK 输入端加画一个小圆圈

CLK	D	Q	Q^*
\times	\times	\times	Q
\uparrow	0	0	0
\uparrow	0	1	0
\uparrow	1	0	1
\uparrow	1	1	1

实现异步置位、复位

引入 S_d 、 R_d 信号，因为 S_d 、 R_d 是以高电平作为置1和置0输入信号的，所以将4个反相器改为或非门，如图：

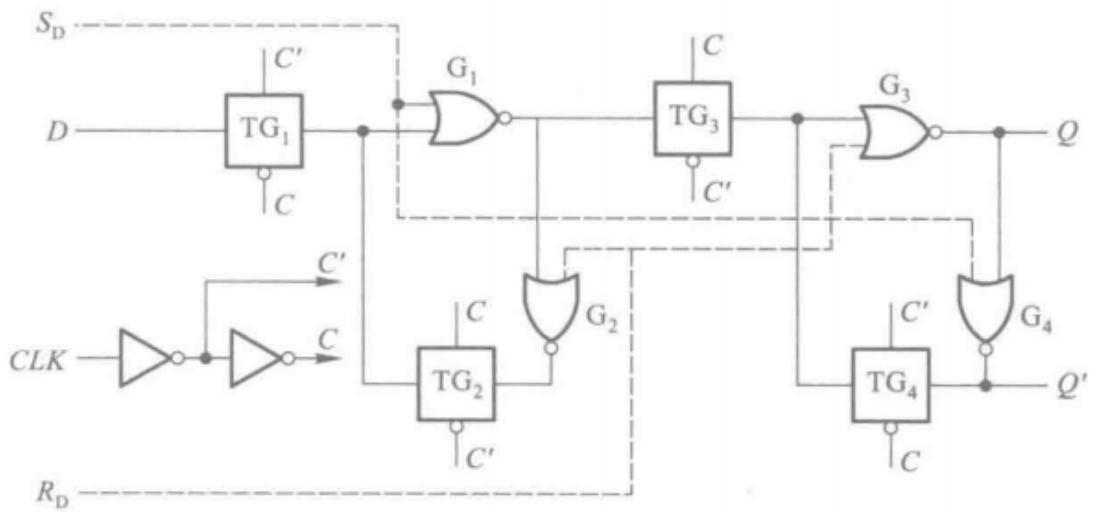
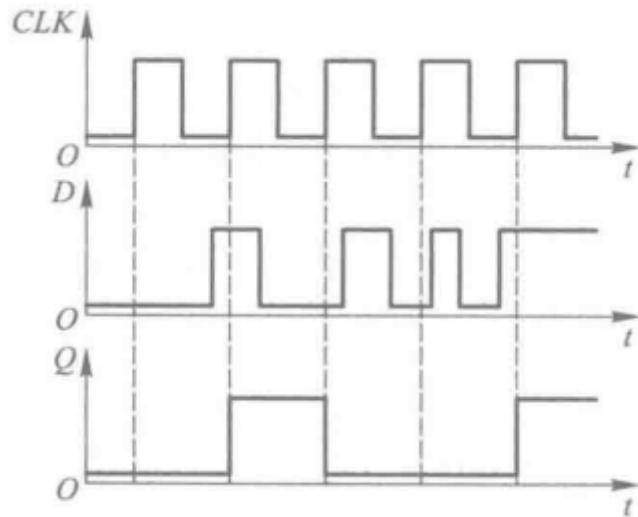


图 5.3.8 带有异步置位、复位端的 CMOS 边沿触发 D 触发器

例：上升沿触发器，由 CLK 和 D 波形画出 Q 的波形



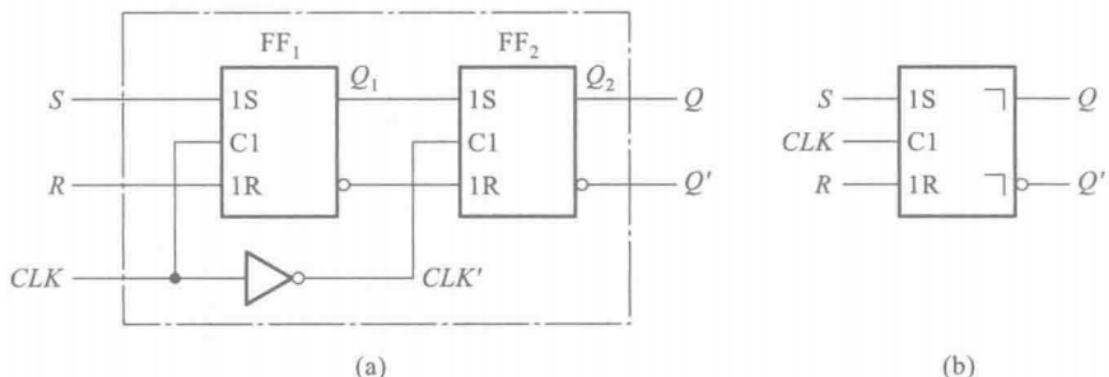
- 边沿触发方式的动作特点

触发器的次态仅仅取决于时钟信号的上升沿（正边沿）或下降沿（负边沿）到达输入时的逻辑状态，而在这以前或以后，输入信号的变化对触发器的输出状态没有影响，提高了触发器的抗干扰能力。

(3) 脉冲触发的触发器

- 电路结构和工作原理

将边沿触发器里的两个电平触发的 D 触发器换成电平触发的 SR 触发器：



FF1为主触发器，FF2为从触发器，当 $CLK = 0$ 时，FF1保持原状态不变，在 CLK 变为高点平之后， $CLK = 1$ ， $CLK' = 0$ ，主触发器的输出 Q_1 将按照S和R的输入端信号被置成相应状态，而从触发器的状态保持不变，当 CLK 回到低电平，即下降沿到来时，从触发器的输出 Q_2 被置成与此刻 Q_1 相同的状态，而主触发器保持不变。

由于在 CLK 高电平期间主触发器的输出状态可能随S和R的状态变化而变化，输出端的状态不可能始终与输入端保持一致，因此在脉冲SR触发器中，不可能像边沿触发器那样，仅仅根据 CLK 下降沿到来时刻输入端S和R状态确定输出端Q的状态，而必须考察全部 $CLK = 1$ 期间主触发器的状态变化。

例如，当 $CLK = 1$ 期间输入信号先是 $S = 0$ 、 $R = 1$ ，主触发器被置成 $Q_1 = 0$ ，随后又变为 $S = 1$ ， $R = 0$ ，于是触发器被置成了 $Q_1 = 1$ ，而在 CLK 下降沿到来之前输入又变成了 $S = 0$ ， $R = 0$ ，这时主触发器将保持 $Q_1 = 1$ 不变，这样在 CLK 下降沿到来时，输出便被置成 $Q = Q_2 = 1$

特性表：

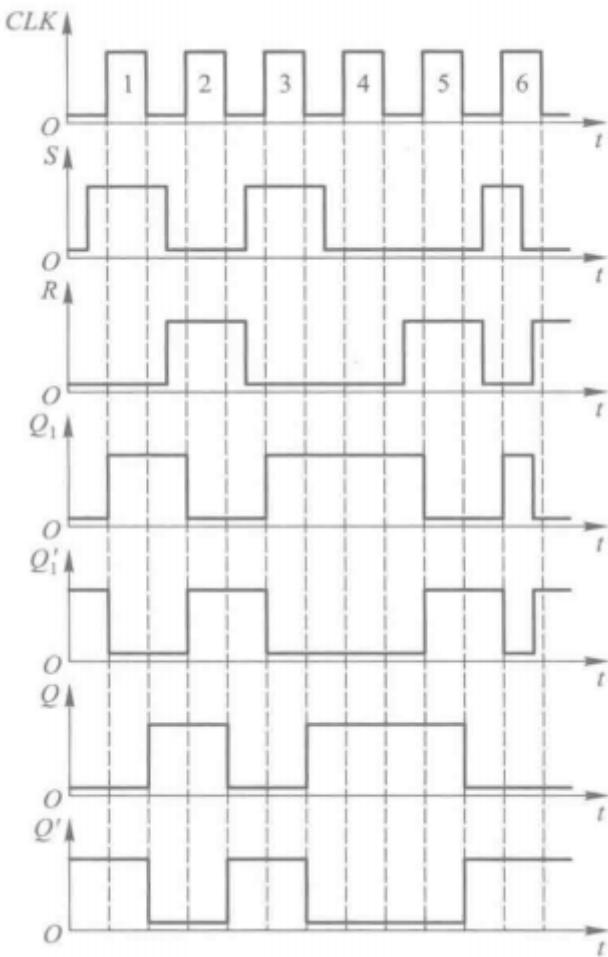
CLK	S	R	Q	Q^*
\times	\times	\times	\times	Q
	0	0	0	0
	0	0	1	1
	1	0	0	1
	1	0	1	1
	0	1	0	0
	0	1	1	0
	1	1	0	1 ⁽¹⁾
	1	1	1	1 ⁽¹⁾

(1) CLK 回到低电平后输出状态不定。

若在 CLK 输入端增加一个反相器，则电路将变为低电平有效信号，这时输出状态的变化将发生在 CLK 的上升沿，在功能表 CLK 一栏中用“”

表示，同时，在图形符号中 CLK 输入端处增画一个小圆圈

例：根据 CLK 、 S 、 R 的波形求出 Q 和 Q' 的波形，设触发器初始 $Q = 0$ ，保证 $S * R = 0$ ，若 $S = R = 1$ ，则规定 $Q^* = Q'$ ，则可接触 $S * R = 0$ 的约束



- JK触发器

在SR触发器的基础上，如果当 $S = R = 1$ 时，将 Q 和 Q' 接回到输入端，用 Q 替代 S 端的输入信号，用 Q' 替代 R 端的输入信号，即可实现 $Q^* = Q'$ ，为与 SR 触发器作出区分，称其为 JK 触发器

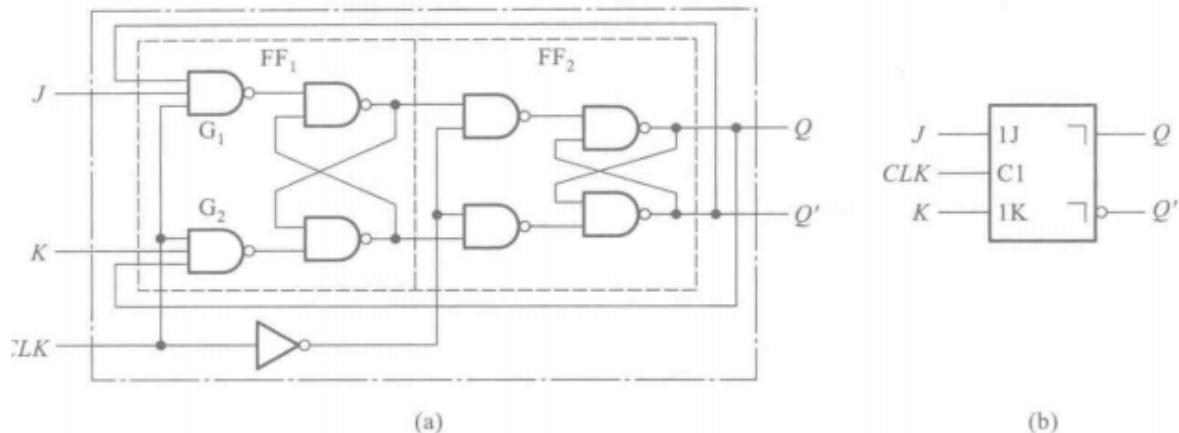


图 5.3.12 正脉冲触发的 JK 触发器

(a) 电路结构图 (b) 图形逻辑符号

若 $J = 1, K = 0$ ，则 $CLK = 1$ 时， FF_1 置 1，待 $CLK = 0$ 后 FF_2 随之置 1，即 $Q^* = 1$

若 $J = 0, K = 1$ ，则 $CLK = 1$ 时， FF_1 置 0，待 $CLK = 0$ 后 FF_2 随之置 0，即 $Q^* = 0$

若 $J = K = 0$ ，则由于门 G_1, G_2 被封锁，触发器保持原状态不变， $Q^* = Q$

若 $J = K = 1$ ，若 $Q = 0$ ， G_2 被封锁， $CLK = 1$ 时仅 G_1 能输出低电平信号，故 FF_1 置 1， $CLK = 0$ 后， FF_2 随之置 1，即 $Q^* = 1$ ；若 $Q = 1$ ， G_1 被封锁， $CLK = 1$ 时仅 G_2 能输出低电平信号，故 FF_1 置 0， $CLK = 0$ 后 FF_2 随之置 0，故 $Q^* = 0$ ；因此 $Q^* = Q'$

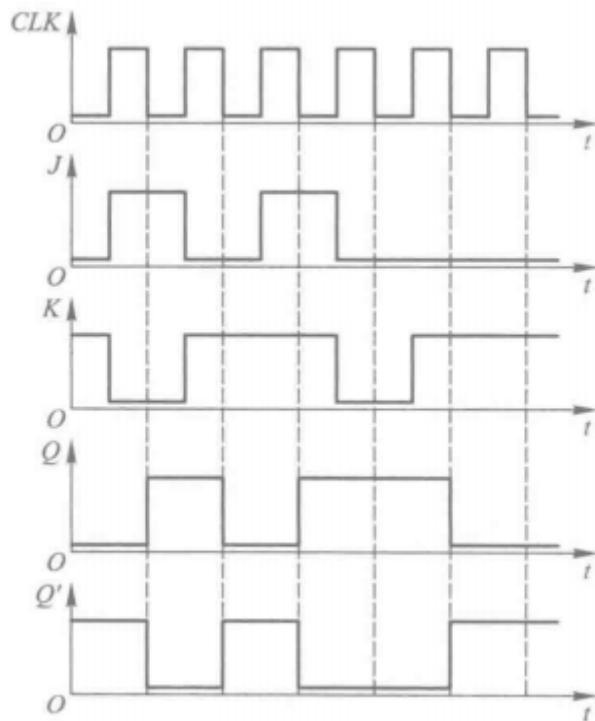
特性表如下：

表 5.3.5 脉冲触发 JK 触发器的特性表

CLK	J	K	Q	Q'
\times	\times	\times	\times	Q
\square	0	0	0	0
\square	0	0	1	1
\square	1	0	0	1
\square	1	0	1	1
\square	0	1	0	0
\square	0	1	1	0
\square	1	1	0	1
\square	1	1	1	0

在有些集成电路触发器产品中，输入端的J和K不止一个，在这种情况下J和K时逻辑与的关系用 $J_1 * J_2$ 表示原本的 J

例根据 CLK、J、K 的波形画出 Q 和 Q' 的波形



• 脉冲触发方式的动作特点

- 触发器的翻转分两步动作，第一步，当 $CLK = 1$ 时，FF1接收输入端的信号，被置成相应状态，而FF2不动，第二步， CLK 下降沿到来时FF2根据FF1的输出进行状态改变，所以 Q 、 Q' 状态的改变发生在 CLK 的下降沿。
- FF1本身是一个电平触发的SR锁存器，当 $CLK = 1$ 期间内，输入信号都会对FF1起控制作用

在 $CLK = 1$ 期间输入信号发生过变化以后， CLK 下降沿到达时从触发器的状态不一定能按此刻输入信号的状态来确定，必须考虑整个 $CLK = 1$ 期间内输入信号的变化过程才能确定触发器的次态

只有在 $CLK = 1$ 期间内输入状态始终未变的情况下，用 CLK 下降沿到达时输入状态决定触发器的次态才肯定是对的，否则必须考虑 $CLK = 1$ 期间输入状态变化全过程，才能确定 CLK 下降沿到达时触发器的次态

例：已知 CLK 、 J 、 K ，画出 Q 、 Q'

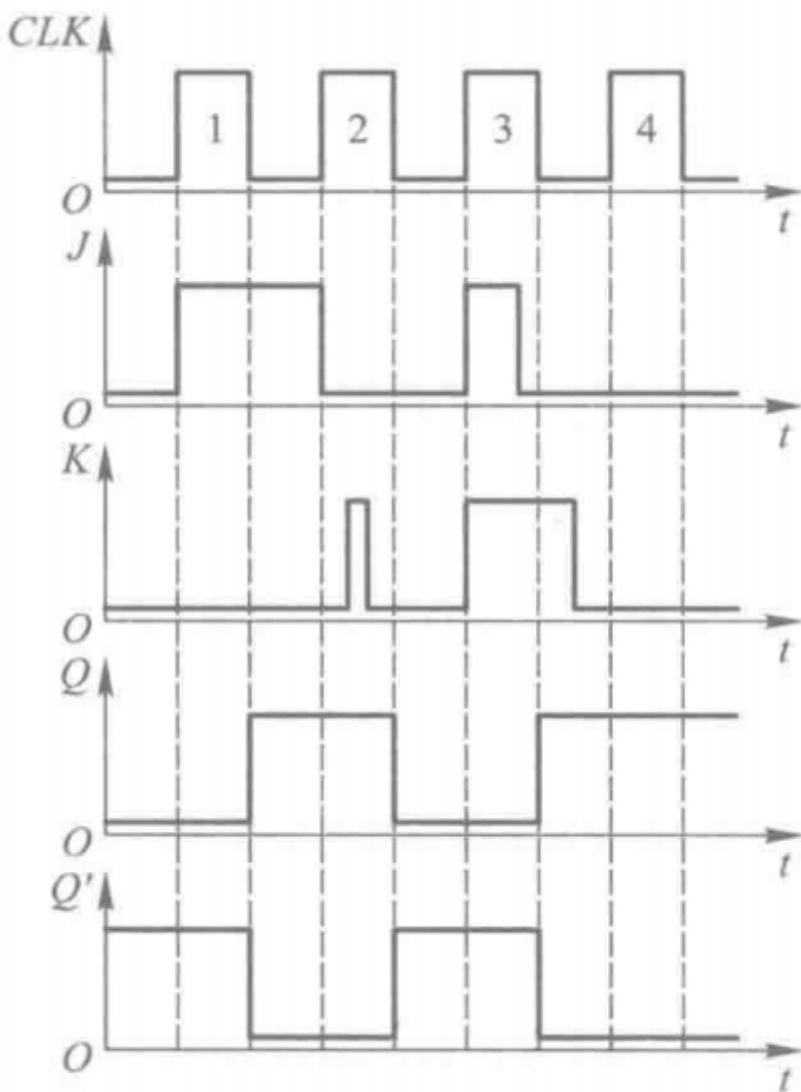


图 5.3.15 例 5.3.6 的电压波形图

第二个CLK高电平期间K端状态发生过变化，因而不能简单地以CLK下降沿到达时、K的状态来决定触发器的次态，因为在CLK高点平期间曾出现过短时间的J = 0、K = 1的状态，此时FF1便被置0，所以虽然CLK下降沿到达时输入状态回到了J = K = 0，但FF2仍按FF1的状态被置0，即Q* = 0

第三个CLK下降沿到达时，J = 0、K = 1，如果以这时的输入状态决定触发器次态，应保持Q* = 0，但由于CLK高电平期间曾出现过J = K = 1状态，CLK下降沿到达之前主触发器以被置1，所以CLK下降沿到达后从触发器被置1

(4) 触发器按逻辑功能分类

- SR触发器

前文已介绍

根据真值表写出逻辑函数式：

$$\begin{cases} Q^* = S'R'Q + SR'Q' + SR'Q = SR' + S'R'Q \\ [SR = 0] \quad (\text{约束条件}) \end{cases}$$

利用约束条件化简：

$$\begin{cases} Q^* = S + R'Q \\ SR = 0 \quad (\text{约束条件}) \end{cases}$$

- JK触发器

前文已介绍

根据真值表写出逻辑函数式：

$$Q^* = J * Q' + K' * Q$$

- T触发器

当控制信号T = 1时每来一个时钟信号触发器的状态就翻转一次，当T = 0时，时钟信号到达后触发器的逻辑功能不变。成为T触发器

特性表：

T	Q	Q^*
0	0	0
0	1	1
1	0	1
1	1	0

逻辑符号：

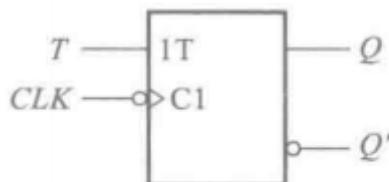


图 5.3.16 T 触发器的图形

逻辑符号

只要将JK触发器的两个输入端连在一起作为T端，就可以构成T触发器

当T = 1时， $Q^* = Q'$

- D触发器

上文已介绍

$$Q^* = D$$

- 触发器逻辑总结

JK触发器的逻辑功能最强，包含了SR触发器和T触发器的所有逻辑功能，在使用SR触发器时，只要将JK触发器的J、K端当作S、R端使用，就可以实现SR触发器的功能，在需要T触发器时，只要将J、K连在一起当作T端使用，既可以实现T触发器的功能

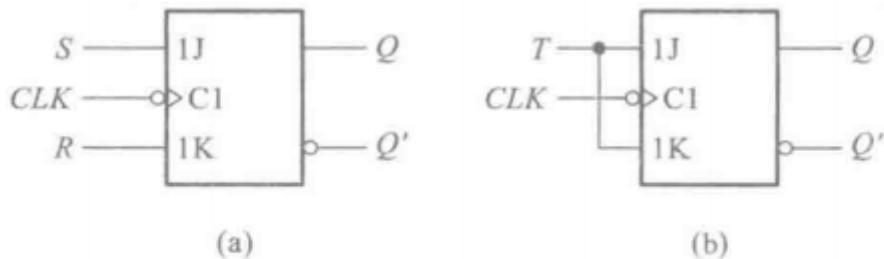


图 5.3.17 将 JK 触发器用作 SR、T 触发器

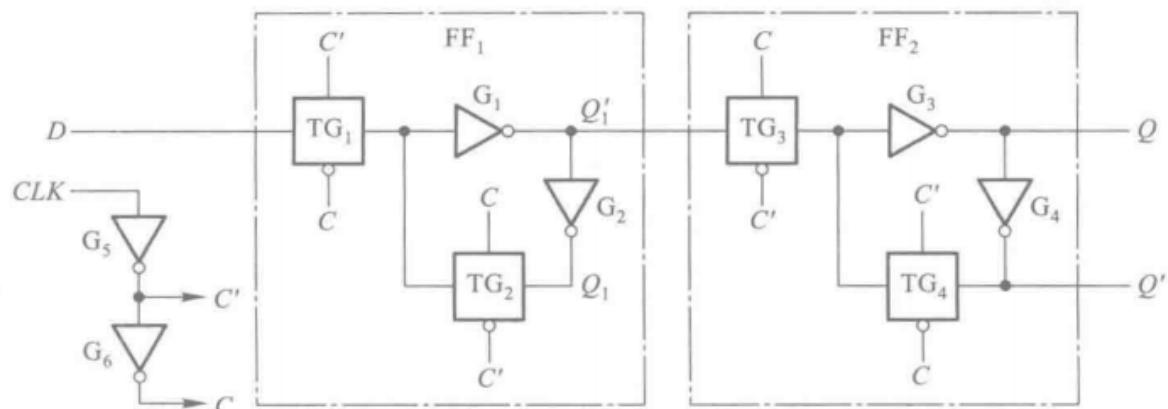
(a) 用作 SR 触发器 (b) 用作 T 触发器

逻辑功能和触发方式是触发器的两个最重要的特性，逻辑功能是指触发器的次态和初态与输入之间的关系，而触发方式是指触发器动态翻转过程中的动作特点

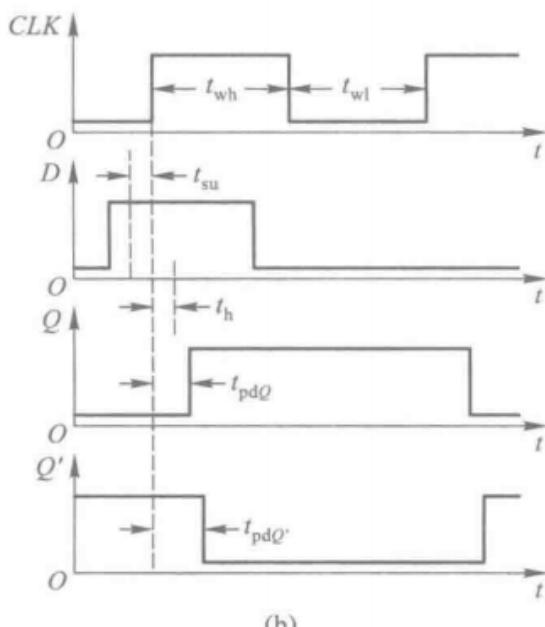
(5) 触发器的动态特性

分析触发器的动态翻转过程，从而找出对输入信号、时钟信号以及两者相互配合关系的要求，通常用建立时间、保持时间、传输延迟时间以及最高时钟频率等几个参数具体描述触发器的动态特性。

假定传输门从控制信号C和C'跳变到它的输出状态改变的延迟时间、反相器的传输延迟时间都是 t_d



(a)



(b)

图 5.3.19 边沿触发 D 触发器动态特性的分析

(a) 电路图 (b) 电压波形图

建立时间 (setup) 是指在触发器的时钟信号上升沿到来以前，数据稳定不变的时间，如果建立时间不够，数据将不能在这个时钟上升沿被打入触器。

保持时间 (hold) 是指在触发器的时钟信号上升沿到来以后，数据稳定不变的时间，如果保持时间不够，数据同样不能被打入触发器。

- 建立时间 (Setup time) t_{su}

指输入信号应当先于时钟信号CLK动作沿到达的时间。

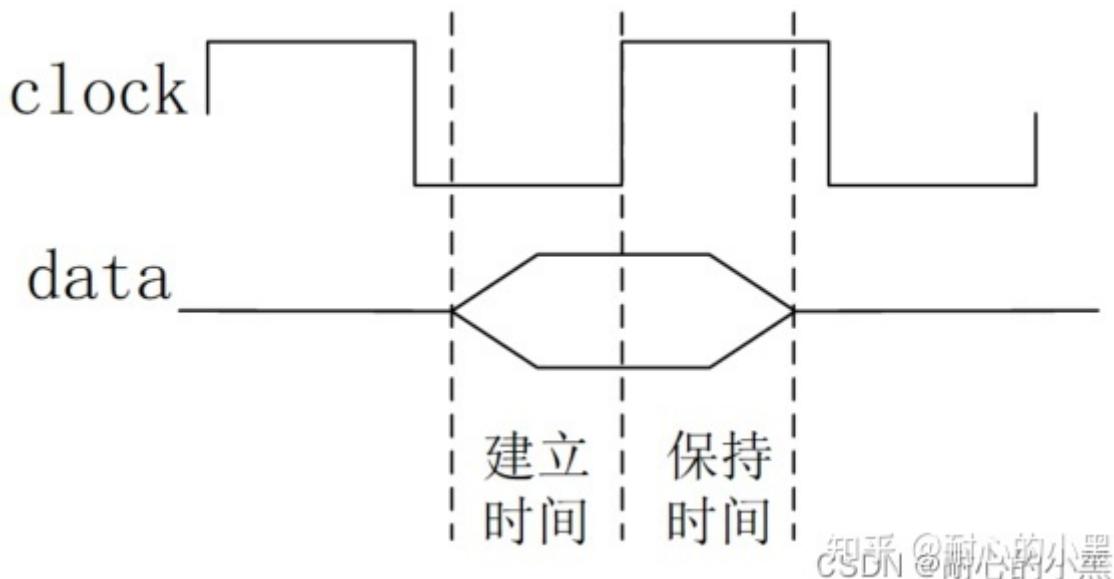
为了保证触发器可靠地翻转，在C和C'状态改变以前FF1和Q1的状态必须稳定地建立起来，使 $Q_1 = D$ ，由于加到D端的输入信号需要经过传输门TG1和反相器G1、G2的传输延迟时间才能到达Q1端，而在CLK的上升沿到达后，只需经过反相器G5的传输延迟时间C'的状态即开始改变，因此D端的输入信号必须先于CLK的上升沿至少 $2td$ 的时间才能到达，即 $t_{su} = 2td$

- 保持时间 (Hold time) t_h

时钟信号CLK动作沿到达后，输入信号仍然需要保持不变的时间，在C和C'改变状态使TG1变为截止，TG2变为导通之前，D端的输入信号应当保持不变，为此，至少在CLK上升沿到达后 $2td$ 时间内输入信号应当保持不变，即保持时间应当为 $t_h = 2td$

可以通俗的理解为：时钟到来之前，数据需要提前准备好；时钟到来之后，数据还要稳定一段时间。建立时间和保持时间组成了数据稳定的窗口，如下图所示。

如果数据在传输中不满足建立时间或保持时间，则会处于[亚稳态](#)，导致传输出错。



- 传输延迟时间 (Propagation delay time) tpd

传输延迟时间是指从CLK动作沿到达开始，直到触发器输出的新状态稳定建立所需要的时间，FF2输出端Q的新状态需要经过C、C'、TG3和G3的传输延迟以后才能建立起来，所以输出端Q的传输延迟时间 $tpdq = 4td$ ，而Q'端还要再经过G4的延迟才能建立起来。因而输出端Q的传输延迟时间 $tpdq = 5td$

- 最高时钟频率 (Maximum clock frequency) f_{max}

指触发器在连续、重复翻转的情况下，时钟信号可以达到的最高重复频率。

为了保证触发器能够可靠地翻转，CLK的低电平持续时间 twl 必须大于建立时间，所以 $twl_{min} = 2td$

CLK变为高电平后，直到Q'新状态建立起来以前，TG3必须保持导通状态，因而C和C'状态不能改变，考虑到需要经过G5的传输延迟时间 td 以后C和C'的状态才能改变，所以CLK高电平期间 twh 必须大于 $tpdq' - td$ ，故 twh 的最小值应为 $rwh_{min} = 4td$

最高时钟频率为：

$$f_{max} = 1/(twl_{min} + twh_{min}) = 1/6td$$

若CLK波形占空比为50%，则应取 $twl = twh = 4td$ ，这时最高时钟频率将是 $1/(8td)$

需要注意的是：在以上的分析过程中，我们假定所有门电路的传输延迟时间都是相等的，但实际上每个门电路的传输延迟时间是各不相同的，上述分析只能定性分析，具体数值需要实验来测定

4. 寄存器

寄存器（Register）能够存储一组二值代码，他被广泛应用于数字计算机和各种复杂的数字系统当中

由于一个触发器能够存储一位二值代码，所以用N个触发器组成的寄存器能够组成一组N位的二进制代码。每个触发器的输入端和输出端都能被直接引出

对寄存器中的触发器，只要求能够置1和置0，所以各种触发方式的触发器都能组成触发器

四位寄存器：

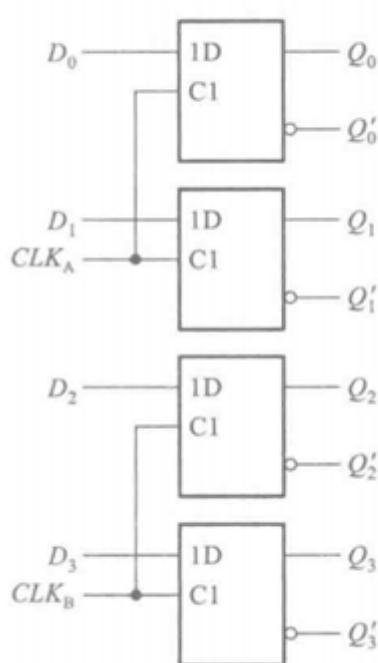


图 5.4.1 74LS75 的逻辑图

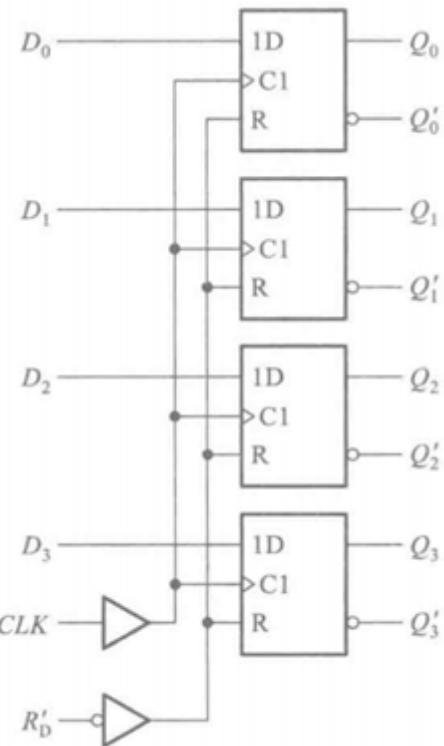


图 5.4.2 74HC175 的逻辑图

74LS74，在CLK高电平期间，Q端的状态跟随D端状态而变，在CLK变成低电平以后，Q端将保持CLK变为低电平时刻D端的状态

74HC175，用CMOS边沿触发器组成4位寄存器，触发器的状态仅仅取决于CLK上升沿到达时刻D端的状态

为了增加使用的灵活性，在有些寄存器电路中还附加了一些控制电路，使寄存器又增加了异步置0，输出三态控制，和“保持”等功能，“保持”是指CLK信号到达时触发器不随D端信号而改变状态，保持原来的状态不变

这两种寄存器接收数据时所有各位代码同时输入的，而且触发器中的数据同时地出现在输出端的，因此将这种输入输出方式称为并行输入、并行输出方式

5. 存储器

存储器是指能够存储大量二值信息的器件

存储容量和存取速度是衡量存储器件性能的两个最重要的指标

由于存储器存储单元庞大而引脚有限，所以在电路结构上不可能像寄存器那样把每个存储单元的输入和输出直接引出，因此在存储器中给每个存储单元编了一个地址，只有被输入地址代码指定的那些存储单元才能与公共的输入/输出引脚接通，进行数据的读入与写出

RAM与ROM的区别在于，RAM在正常工作状态下就可以随时快速地访问存储器里写入数据或从中读出数据。SRAM结构简单，集成度高，存取速度不如DRAM

(1) SRAM

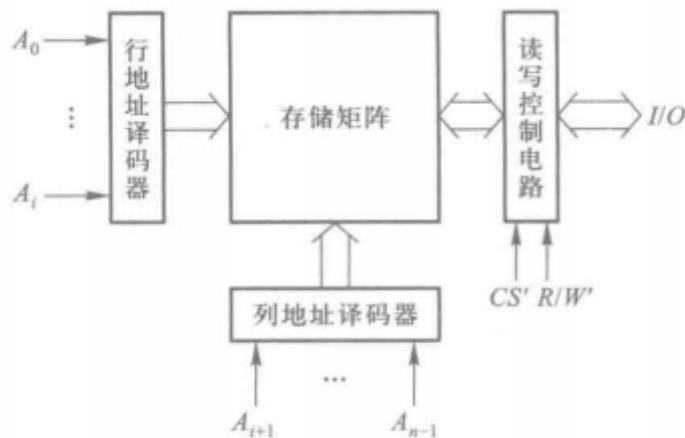


图 5.5.1 SRAM 的结构框图

- SRAM的结构和工作原理

SRAM通常由存储矩阵，地址译码器和读/写控制电路三部分组成

存储矩阵由许多存储单元组成，每个存储单元能够存储1位二值数据地址译码器一般都分成行地址译码器和列地址译码器两部分。行地址译码器将输入地址代码的若干位译成某一条字线的输出高、低电平信号，从存储矩阵中选中一行存储单元；列地址译码器将输入地址代码的其余几位译成某一根输出线上的高、低电平信号，从字线选中的一行存储单元中再选1位(或几位)，使这些被选中的单元经读/写控制电路与输入/输出端接通，以便对这些单元进行读、写操作

读/写控制电路用于对电路的工作状态进行控制。当读/写控制信号 $R/W'=1$ 时，执行读操作，将存储单元里的数据送到输入/输出端上。当 $R/W'=0$ 时，执行写操作，将输入/输出端上的数据写入存储单元中。

CS' 为片选输入端，其为0时RAM正常工作，其为1时，RAM的所有输入输出均为高阻态，不能对RAM进行读/写操作

在多位数据并行输出的存储器中，将并行输出的一组数据叫做一个“字”(word)，存储器的每个地址中存放一个字。存储器的容量用存储单元的数量表示，写成“(字数)×(每个字的位数)”的形式

例：1024*4位SRAM结构框图

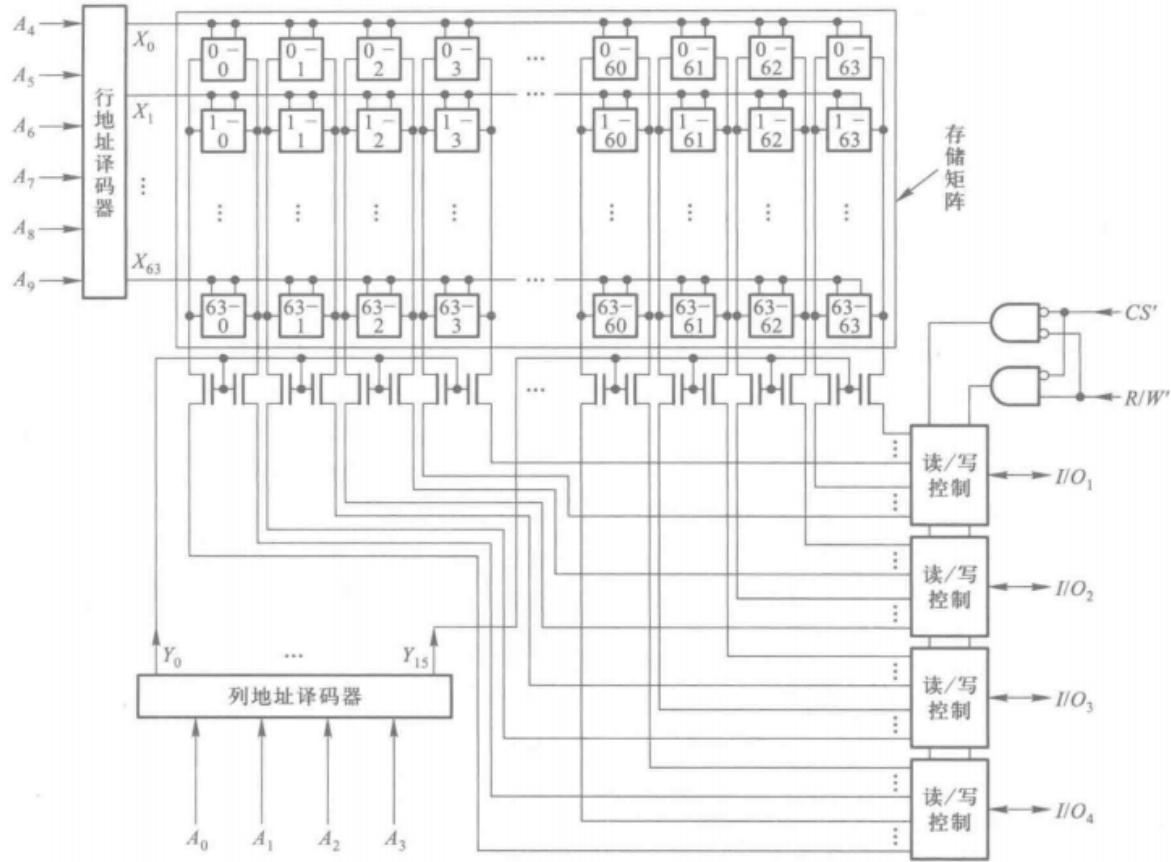


图 5.5.2 1024×4 位 SRAM 的结构框图

4096个存储单元被排列成 64×64 的矩阵，

10位输入地址代码被分为两组译码，A4~A9这6位地址码加到行地址译码器上，用它的输出信号从64行存储单元中选出指定的一行，另外4位地址码加到列地址译码器上，利用它的输出信号再从1行里挑出要进行读/写的4个存储单元

I/O既是数据输入端又是数据输出端， $CS' = 0$ ， $R/W' = 1$ 时，读/写控制电路工作在读出状态，这时被译码器选中的4个存储单元中的数据被送到I/O口， $CS' = 0$ 且 $R/W' = 0$ 时，执行写入操作，工作在写入状态，加到I/O口的数据便被写入指定的存储单元中

若 $CS' = 1$ ，则所有I/O均处于禁止状态。

- SRAM的静态存储单元

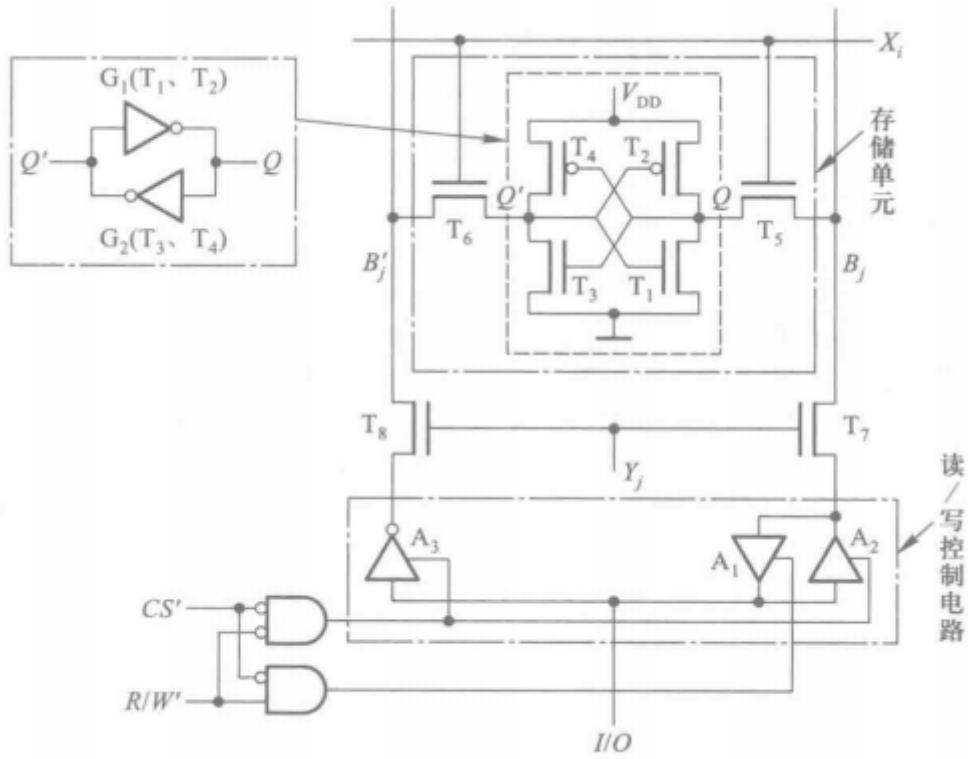


图 5.5.3 六管 CMOS 静态存储单元

静态存储单元是在SR锁存器的基础上附加门控管而构成的，是靠锁存器的自保功能存储数据的

T1~T4组成SR锁存器，用于记忆1位二值代码，T1、T2和T3、T4分别接成了反相器G1、G2，将G1与G2首位相连，可以将写入写出结合在一起，T5、T6是门控管，作模拟开关使用，以控制锁存器的Q、Q'和位线Bj、Bj'之间的联系。Xi = 1时T5、T6导通，锁存器与位线接通，Xi = 0时，T5、T6截止，锁存器与位线之间的联系被切断，T7、T8是每一列存储单元公用的两个门控管，连接读/写缓冲放大器，T7、T8的开关状态由列地址译码器的输出Yj来控制，Yj = 1时导通，Yj = 0时截止。

存储单元所在的一行和所在的一列同时被选中后，Xi = 1、Yj = 1，T5、T6、T7、T8均处于导通状态，Q和Q'与Bj和Bj'接通，如果这时CS' = 0、R/W' = 1，则读/写缓冲放大器的A1导通、A2和A3截止，Q端的状态经A1送到I/O端口，实现数据输出。若此时CS' = 0，R/W' = 0，则A1截止、A2和A3导通，加到I/O端口的数据被写入存储单元中

(2) DRAM

- DRAM的动态存储单元

其是利用MOS电容可以存储电荷的原理制成的，结构简单，在大规模高集成的RAM中普遍应用。由于电容的充放电需要时间，因此其速度会低于静态存储单元。由于电容上的电荷会泄露，因此必须定期刷新电荷。将原来的数据重新写入，保证数据不会丢失。称为刷新或再生。

早期采用的动态存储单元为四管电路或三管电路，而单管动态存储单元是所有存储单元中电路结构最简单的一种。

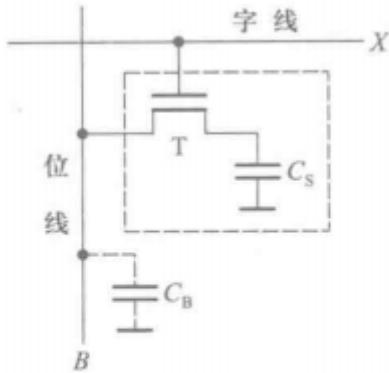


图 5.5.4 单管动态
MOS存储单元

单管动态MOS存储电路的电路结构如图，存储单元由一只N沟道增强型MOS管T和一个电容Cs组成。在进行写操作时，字线给出高电平，使T导通，位线上的数据便经过T被存入Cs中，在进行读操作时，字线同样应给出高电平，并使T导通，这时C经T位线上的电容Cb提供电荷，使位线获得读出的信号电平。设Cs上原来存有正电荷。电压Vcs为高电平，而位线电位Vb = 0，则执行读操作以后位线电平将上升为

$$v_b = \frac{C_s}{C_s + C_b} v_{cs}$$

因为在实际的存储器电路中位线上总是同时接有很多存储单元，使 $C_b \gg C_s$ ，所以位线上读出的电压信号很小。

- DRAM的总体结构

为了提高集成度的同时减少器件引脚数目，目前大容量的DRAM多半采用1位输入，1位输出和地址分时输入，亦称，地址多路复用

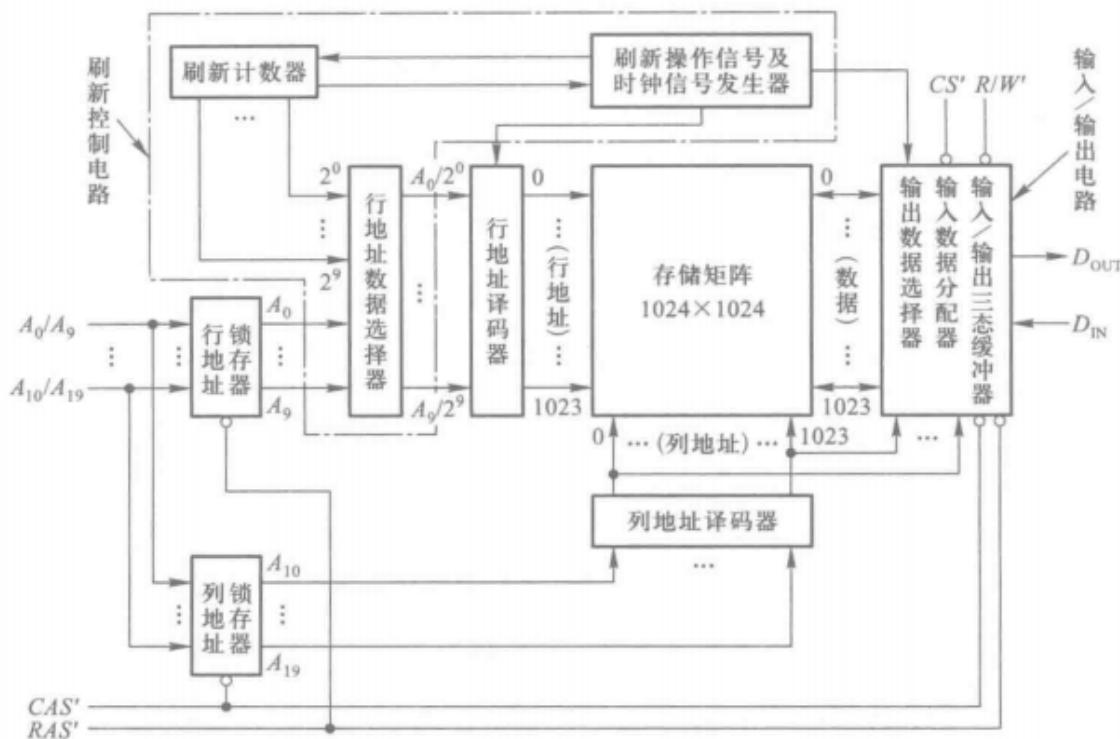


图 5.5.5 1 M×1 位 DRAM 的结构框图

从总体上讲，它除了包含存储矩阵、地址译码器和输入/输出电路三个部分组成以外，还增加了一套刷新控制电路。

(3) ROM

在存储器的某些应用场合中，要求存储的是一些固定不变的数据，例如计算机中的字库，正常工作状态下，这些数据只供读出使用，不需要随时进行修改，为了适应这种需要，产生了ROM。

ROM的工作特点在于：所存储的数据是固定的，预先写好的，正常工作时，这些数据只能读出不能随时写入或修改。

只要将只读存储器中每个存储单元的输出接到固定的高电平或者低电平就行了，大大简化了存储单元的电路结构，不需要用锁存器或者触发器作为存储单元。

- ROM的结构和工作原理

ROM的电路结构包含存储矩阵、地址译码器和输出缓冲器三个组成部分，如图：

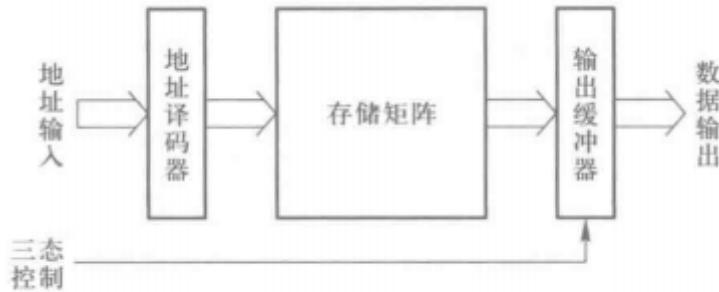


图 5.5.6 ROM 的电路结构框图

存储矩阵由许多存储单元排列而成，每个排列单元可存放1位二值代码，每一个或每一组存储单元有一个对应的地址代码。通常ROM都采用多位数据并行输出的结构形式，每一个输入地址同时选中一组存储单元。

地址译码器的作用是将输入的地址代码译成相应的控制信号，利用这个控制信号从存储矩阵中将指定的单元选出，并把其中的数据送到输出缓冲器。

输出缓冲器的作用有两个，一是能提高存储器的带负载能力，二是实现对输出状态的三态控制，以便与系统的总线连接。

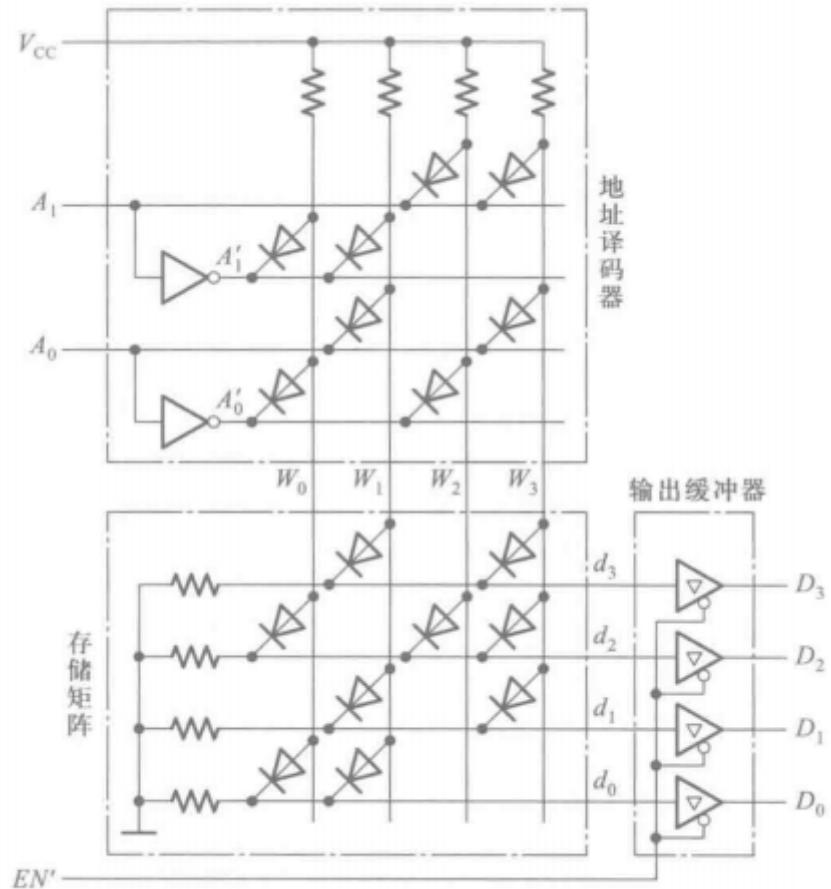


图 5.5.7 二极管 ROM 的电路结构图

图为具有2位地址输入码和4位数据输出的ROM电路，其存储单元由二极管构成，它的地址码由4个二极管与门组成，2位地址码A₁A₀能给出4个不同的地址，地址译码器将这4个地址代码分别译成W₀~W₃这4根线上的高电平信号。存储矩阵实际上由4个二极管或门组成的译码器，当W₀~W₃每根线上给出高电平信号时，都会在d₃~d₀这4根线上输出一个4位二值代码。将每个输出代码称为一个“字”，并将W₀~W₃称为字线，将d₀~d₃称为位线（或数据线），而A₁、A₀称为地址线，输出端的缓冲器用来提高带负载能力，并将输出的高低电平变换为标准的逻辑电平。同时，通过给定EN'信号实现对输出的三态控制。

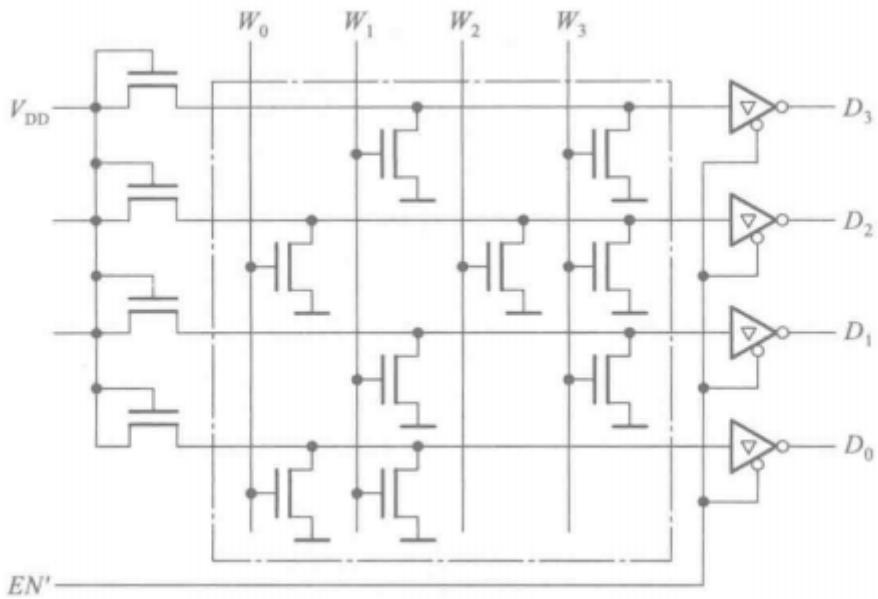
在读取数据时，只要输入指定的地址代码并令EN' = 0，则指定地址内存个存储单元所存的数据便会出现在线路上。数据表如下：

表 5.5.1 图 5.5.7 ROM 中的数据表

地址		数 据			
A ₁	A ₀	D ₃	D ₂	D ₁	D ₀
0	0	0	1	0	1
0	1	1	0	1	1
1	0	0	1	0	0
1	1	1	1	1	0

子线和位线的每个交叉点都是一个存储单元，交点处接由二极管时相当于存1，没有接二极管时相当于存0，交叉点的数目也就是存储单元数。

下为用MOS管构成的存储矩阵：



字线与位线的交叉点上接有MOS管时相当于存1，没有接MOS管时相当于存0。

- ROM的分类

掩模只读存储器(Mask ROM)

掩模ROM在出厂时内部存储的数据就已经固化在里面了，制作周期长，制作版图成本高，多用于大批量、定型电子产品中

可编程只读存储器(PROM)

PROM的总体结构与掩模ROM一样，由存储矩阵、地址译码器、输出电路组成，在出厂时所有存储单元都存入了1，内部由熔丝结构，在写入数据时只要将存入0的那些存储单元上的熔丝烧断即可。可见，PROM的内容一经写入以后，就不可能修改了，只能写入一次。

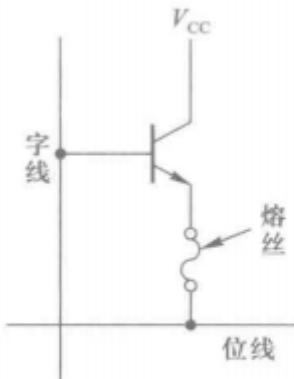


图 5.5.9 熔丝型 PROM
的存储单元

用电信号擦除的可编程只读存储器——闪存 (Flash Memory)

闪存的存储单元是一只浮栅MOS管，在浮栅MOS管中，除了控制栅Gc以外，还在控制栅和衬底之间增加了一个浮置栅Gf。

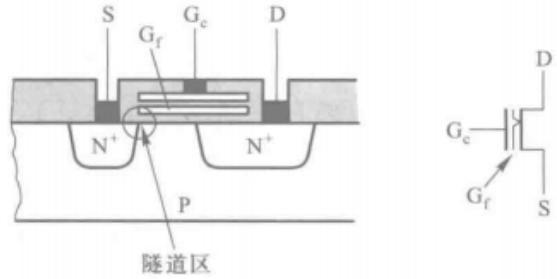


图 5.5.11 闪存中的浮栅 MOS 管

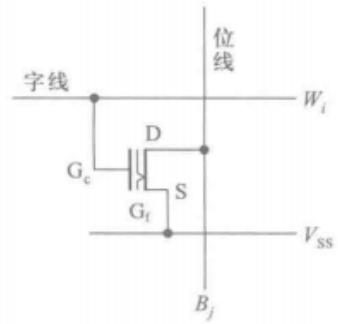


图 5.5.12 闪存的存储单元