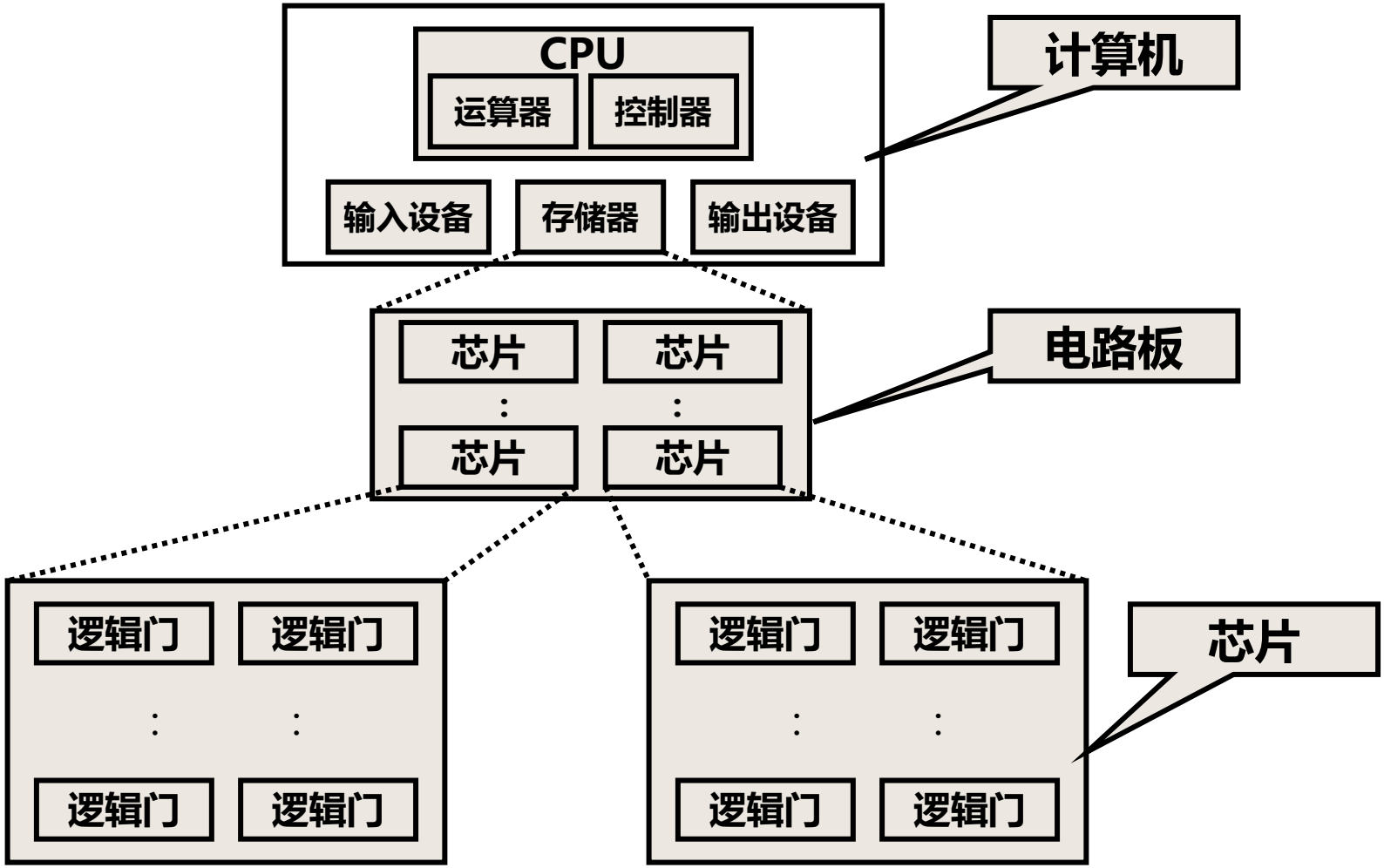
2. 数字电路基础

计算机中的逻辑电路


CPU
运算器
控制器
输入设备
存储器
输出设备
计算机
芯片
芯片
芯片
芯片
电路板
逻辑门
逻辑门
逻辑门
逻辑门
逻辑门
逻辑门
逻辑门
逻辑门
芯片

概览

- 2.1 数字电路和数字逻辑概述
  - 2.1.1 逻辑代数基础
  - 2.1.2 门电路
- 2.2 计算机中的数字电路
  - 2.2.1 组合逻辑：加法器，ALU，译码器，数据选择器， ROM
  - 2.2.2 时序逻辑：触发器，计数器，寄存器和RAM

2.1 数字电路与数字逻辑概述

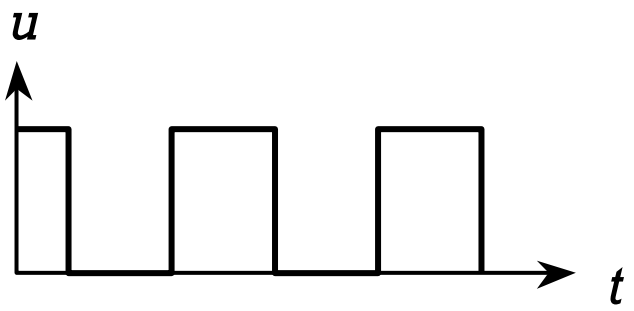
■数字电路的特点

① 工作信号是二进制的数字信号，在时间上和数值上是离散的（不连续），反映在电路上就是低电平和高电平两种状态（即0和1两个逻辑值）。

② 和模拟电路相比，组成数字电路的元器件的精度要求不高，只要在工作时能够可靠地区分0和1两种状态即可。

③ 数字量更便于加工、处理、传输、存储等，可靠，抗干扰能力强。

④ 在数字电路中，研究的主要问题是电路的逻辑功能，即输入信号的状态和输出信号的状态之间的关系。



数字信号波形

2.1.1 逻辑代数基础

- 事物往往存在两种对立的状态，在逻辑代数中可以抽象地表示为 0 和 1 ，称为**逻辑0状态**和**逻辑1状态**。

- **逻辑常量**：0和1，不表示数量的大小，而是表示两种对立的逻辑状态。
- **逻辑变量：**逻辑代数中的变量，取值只逻辑0和逻辑1两种

- 逻辑代数：按一定的逻辑关系进行运算的代数，是分析和设计数字电路的数学工具。在逻辑代数中，只有 **0** 和 **1** 两种逻辑值，**与、或、非**三种基本逻辑运算，还有**与或、与非、与或非、异或**等几种导出逻辑运算。

逻辑运算-与逻辑

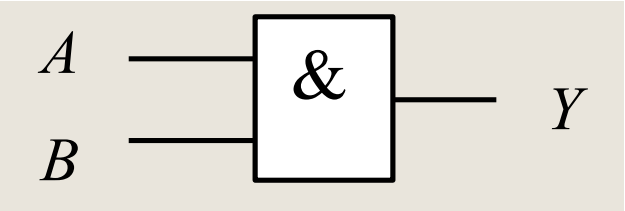
- 与逻辑的定义：仅当决定事件（Y）发生的所有条件（A，B，C，…）均满足时，事件（Y）才能发生。表达式为：

Y = A B C…

把所有可能的条件组合，及其对应结果一一列出来的表格叫做真值表。

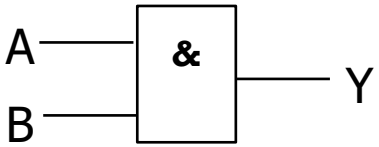
A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

逻辑符号

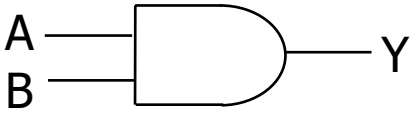


Y = A B

国标



国际常用



逻辑运算-或逻辑

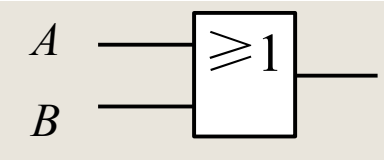
或逻辑的定义：当决定事件（Y）发生的各种条件（A，B，C，...)中，只要有一个或多个条件具备，事件（Y）就发生。表达式为：

$$Y = A + B + C + \ldots$$

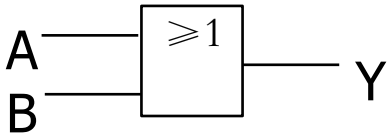
真值表

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

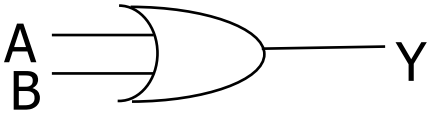
逻辑符号



Y=A+B



国标



国际常用

逻辑运算-非逻辑

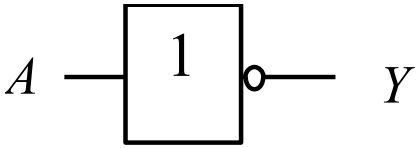
非逻辑：指的是逻辑的否定。当决定事件（Y）发生的条件（A）满足时，事件不发生；条件不满足，事件反而发生。表达式为：

$$Y = \overline{A}$$

真值表

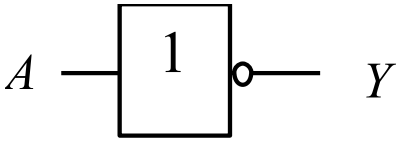
A	Y
0	1
1	0

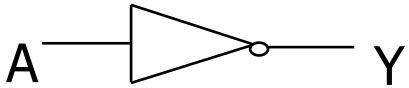
逻辑符号



$Y=\overline{A}$

国标





国际常用

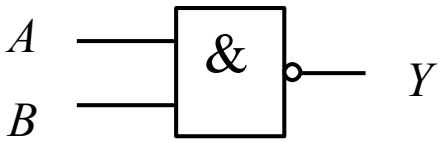
其他常用的逻辑运算

（1）与非运算：逻辑表达式为：

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

真值表

$$Y = \overline{AB}$$



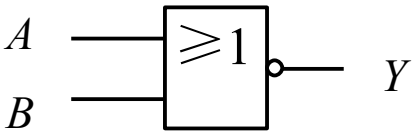
与非门的逻辑符号

（2）或非运算：逻辑表达式为：

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

真值表

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



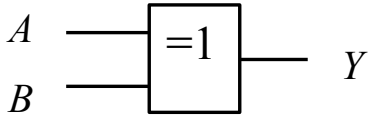
或非门的逻辑符号

（3）异或运算：逻辑表达式为：

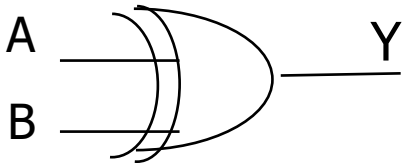
$$Y = \overline{A}B + A\overline{B} = A \oplus B$$

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

真值表



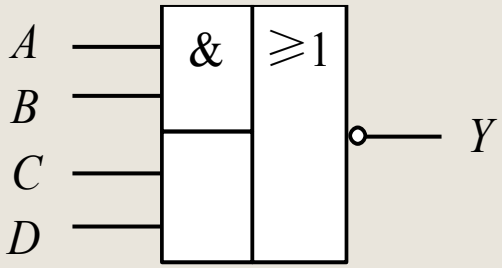
异或门的逻辑符号



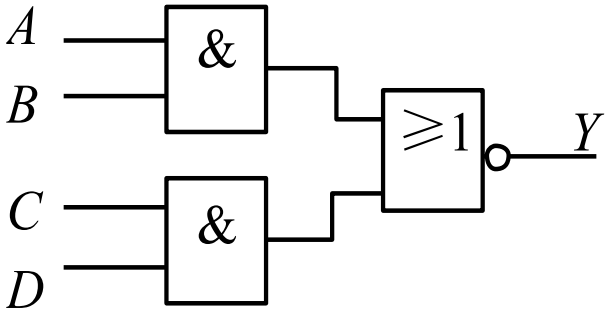
国际常用

（4） 与或非运算：逻辑表达式为：

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



与或非门的逻辑符号



与或非门的等效电路

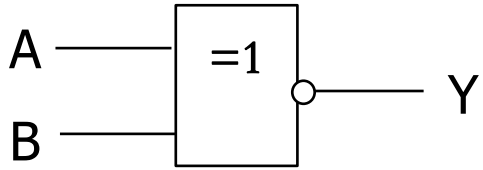
(5) 同或运算

- 逻辑表达式： $Y = AB + \overline{A}\,\overline{B} = A \odot B$

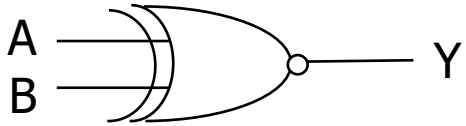
同或逻辑真值表

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

同或逻辑符号



国标



国际常用

名称	国标符号	曾用符号	国外流行符号
与	A B & Y	A B Y	A B Y
或	A B ≥1 Y	A B + Y	A B Y
非	A 1 Y	A Y	A Y
与非	A B & Y	A B Y	A B Y
或非	A B ≥1 Y	A B + Y	A B Y
与或非	A B C D & ≥1 Y	A B C D + Y	A B C D Y
异或	A B =1 Y	A B ⊕ Y	A B Y
同或	A B = Y	A B ⊙ Y	A B Y

逻辑函数

- 逻辑表达式：由逻辑变量和与、或、非3种运算符连接起来所构成的式子，称为逻辑表达式。
- 在逻辑表达式中，字母上面没有非运算符的叫做原变量，有非运算符的叫做反变量。
- 逻辑函数：如果对应于输入逻辑变量A、B、C、…的每一组确定值，输出逻辑变量 *Y*就有唯一确定的值，则称Y是A、B、C、…的逻辑函数。记为

$$Y = f(A, B, C, \cdots)$$

逻辑函数的相等

设有两个逻辑函数 $Y_1 = f(A, B, C, \cdots)$ $\quad Y_2 = g(A, B, C, \cdots)$

如果对应于输入变量A、B、C、...的任何一组变量取值，输出变量Y_1和Y_2的值都相同，则称Y_1和Y_2是相等的，记为$Y_1=Y_2$。

若两个逻辑函数相等，则它们的真值表一定相同；反之，若两个函数的真值表完全相同，则这两个函数一定相等。

因此，要证明两个逻辑函数是否相等，只要分别列出它们的真值表，看看它们的真值表是否相同即可。

证明等式：

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

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

逻辑代数的公式与定理

- 常量之间的关系定理
- 常量和变量的关系定理
- 与普通代数相似的定理
- 逻辑代数的一些特殊定理
- 关于等式的三个规则

常量之间的关系

1) $0 \cdot 0 = 0$

2) $0 \cdot 1 = 0$

3) $1 \cdot 1 = 1$

4) $\overline{0} = 1$

1') $1 + 1 = 1$

2') $1 + 0 = 1$

3') $0 + 0 = 0$

4') $\overline{1} = 0$

变量与常量之间的关系

5) $A \cdot 1 = A$

6) $A \cdot 0 = 0$

7) $A \cdot \overline{A} = 0$

5') $A + 0 = A$

6') $A + 1 = 1$

7') $A + \overline{A} = 1$

与普通代数类似的定理

交换律

8) $A \cdot B = B \cdot A$ 8') $A + B = B + A$

结合律

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

分配律

10) $A \cdot (B + C) = A \cdot B + A \cdot C$ 10') $A + B \cdot C = (A + B) \cdot (A + C)$

逻辑代数特殊的定理

同一律

11) $A \cdot A = A$　　11') $A + A = A$

德摩根定理

12) $\overline{A \cdot B} = \overline{A} + \overline{B}$　　12') $\overline{A + B} = \overline{A} \cdot \overline{B}$

还原律　13) $\overline{\overline{A}} = A$

其他公式

14) $\overline{A}B+AB=B$

15) $A + AB=A$

16) $A + \overline{A}B=A + B$

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

18) $\overline{A\overline{B} + \overline{A}B} = AB + \overline{A}\overline{B}$

关于等式的三个规则

- 代入规则：任何一个含有变量A的等式，如果将所有出现A的位置都用同一个逻辑函数代替，则等式仍然成立。这个规则称为代入规则。

例如，已知等式 $\overline{AB} = \overline{A} + \overline{B}$ ，用函数Y =AC代替等式中的A，根据代入规则，等式仍然成立，即有：

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

关于等式的三个规则

- 反演规则：对于任何一个逻辑表达式Y，如果将表达式中的所有“•”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**，那么所得到的表达式就是函数Y的反函数$\overline{Y}$（或称补函数）。这个规则称为反演规则。例如：

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

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

注意：反演规则遵循的两个规则

1. 遵守“先括号、然后与、最后或”的运算优先次序；
2. 不属于单个变量上的反号应保留不变。

关于等式的三个规则

- 对偶规则：对于任何一个逻辑表达式Y，如果将表达式中的所有“·”换成“+”，“+”换成“·”，“0”换成“1”，“1”换成“0”，而**变量保持不变**，则可得到的一个新的函数表达式Y′，Y′称为函数Y的对偶函数。例如：

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

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

对偶规则的意义：**如果两个函数相等，则它们的对偶函数也相等**。例如：

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

$$A(B+C) = AB + AC \longrightarrow A + BC = (A+B)(A+C)$$

逻辑函数的化简

- 逻辑函数的标准与或式
- 公式法化简
- 图形法化简
- 含约束项的逻辑函数化简

逻辑函数的标准与或式

逻辑函数的最小项：如果一个函数的某个乘积项，包含了函数的全部变量，其中每个变量都以原变量或反变量的形式出现，且作为因子仅出现一次，这样的乘积项就叫做最小项。n变量函数有2^n个最小项。

例如：3个变量A、B、C可组成8个最小项：

$$\overline{A}\,\overline{B}\,\overline{C}、\overline{A}\,\overline{B}C、\overline{A}B\overline{C}、\overline{A}BC、A\overline{B}\,\overline{C}、A\overline{B}C、AB\overline{C}、ABC$$

最小项的表示方法：通常用符号m_i来表示最小项。下标 i 的确定：确定变量顺序，反变量当做0，原变量当做1，则与这个二进制数相对应的十进制数，就是这个最小项的下标i。例如：3个变量A、B、C的8个最小项可以分别表示为

$$m_0=\overline{A}\,\overline{B}\,\overline{C}、m_1=\overline{A}\,\overline{B}C、m_2=\overline{A}B\overline{C}、m_3=\overline{A}BC$$

$$m_4=A\overline{B}\,\overline{C}、m_5=A\overline{B}C、m_6=AB\overline{C}、m_7=ABC$$

逻辑函数的标准与或式

任何一个逻辑函数都可以表示成唯一的一组最小项之和，称为**标准与或表达式**，也称为**最小项表达式**。

对于不是最小项表达式的与或表达式，可利用公式 $A+\overline{A}=1$ 和 $A(B+C)=AB+AC$，来配项展开成最小项表达式。

$$
\begin{aligned}
Y &= \overline{A} + BC \\
&= \overline{A}(B+\overline{B})(C+\overline{C}) + (A+\overline{A})BC \\
&= \overline{A}BC + \overline{A}B\overline{C} + \overline{A}\overline{B}C + \overline{A}\overline{B}\overline{C} + ABC + \overline{A}BC \\
&= \overline{A}\overline{B}\overline{C} + \overline{A}\overline{B}C + \overline{A}B\overline{C} + \overline{A}BC + ABC \\
&= m_0 + m_1 + m_2 + m_3 + m_7 \\
&= \sum m(0,1,2,3,7)
\end{aligned}
$$

如果列出了函数的真值表，则只要将函数值为1的那些最小项相加，便是函数的最小项表达式。

A	B	C	Y	最小项
0	0	0	0	m_0
0	0	1	1	m_1
0	1	0	1	m_2
0	1	1	1	m_3
1	0	0	0	m_4
1	0	1	1	m_5
1	1	0	0	m_6
1	1	1	0	m_7

$m_1 = \bar{A}\bar{B}C$

$m_2 = \bar{A}B\bar{C}$

$m_3 = \bar{A}BC$

$m_5 = A\bar{B}C$

$$Y = m_1 + m_2 + m_3 + m_5 = \sum m(1,2,3,5)$$

$$= \bar{A}\bar{B}C + \bar{A}B\bar{C} + \bar{A}BC + A\bar{B}C$$

逻辑函数的最简与或式

最简与或式：乘积项最少、并且每个乘积项中的变量也最少的与或表达式。

$$
\begin{aligned}
Y &= \overline{A}B\overline{E} + \overline{A}B + A\overline{C} + A\overline{C}E + B\overline{C} + B\overline{C}D \\
&= \overline{A}B + A\overline{C} + B\overline{C} \\
&= \overline{A}B + A\overline{C} \longrightarrow \text{最简与或表达式}
\end{aligned}
$$

逻辑函数的公式法化简

逻辑函数的公式化简法就是运用逻辑代数的基本公式、定理和规则来化简逻辑函数，这里主要针对与或式的化简。

① 并项法

利用公式 $A+\overline{A}=1$，将两项合并为一项，并消去一个变量。

运用分配律

$$Y_1 = ABC + \overline{A}BC + B\overline{C} = (A+\overline{A})BC + B\overline{C}$$

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

运用分配律

$$Y_2 = ABC + A\overline{B} + A\overline{C} = ABC + A(\overline{B}+\overline{C})$$

$$= ABC + A\overline{BC} = A(BC+\overline{BC}) = A$$

运用摩根定律

② 吸收法

利用公式A + A B = A，消去多余的项。

$$Y_1 = \overline{A}B + \overline{A}BCD(E+F) = \overline{A}B$$

运用摩根定律

$$Y_2 = A + \overline{\overline{B} + \overline{CD}} + \overline{\overline{AD}\,\overline{B}} = A + BCD + AD + B$$

$$= (A + AD) + (B + BCD) = A + B$$

如果一个乘积项是另外一个乘积项的因子，则另外一个乘积项是多余的。

③ 消去法

利用公式16 $A+\overline{A}B=A+B$，消去乘积项中多余的因子。

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

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

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

④ 配项消去法

(1)利用公式17(推论) $AB+\overline{A}C+BC=AB+\overline{A}C$，将 BC 项消去。

$$
\begin{aligned}
Y_1 &= A\overline{B}+AC+ADE+\overline{C}D \\
&= A\overline{B}+(AC+\overline{C}D+\cancel{ADE}) \\
&= A\overline{B}+AC+\overline{C}D
\end{aligned}
$$

$$
\begin{aligned}
Y_2 &= AB+\overline{B}C+AC(\cancel{DE+FG}) \\
&= AB+\overline{B}C
\end{aligned}
$$

（2）利用公式11′ $A+A=A$，为某项配上其所能合并项。

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

逻辑函数的图形法化简

逻辑函数的图形化简法：将逻辑函数用卡诺图来表示，利用卡诺图来化简。

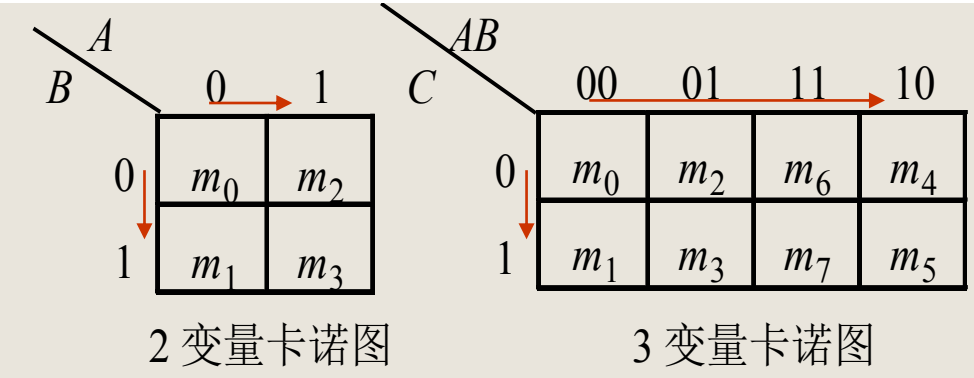
- 表示最小项的卡诺图：将n变量的全部最小项各用一个小方块表示，并使具有逻辑相邻性的最小项在几何位置上也相邻地排列起来，所得到的图形叫n变量最小项的卡诺图。（相邻项是指两个最小项，只有一个因子互为反变量，其余因子均相同，又称为逻辑相邻项）。

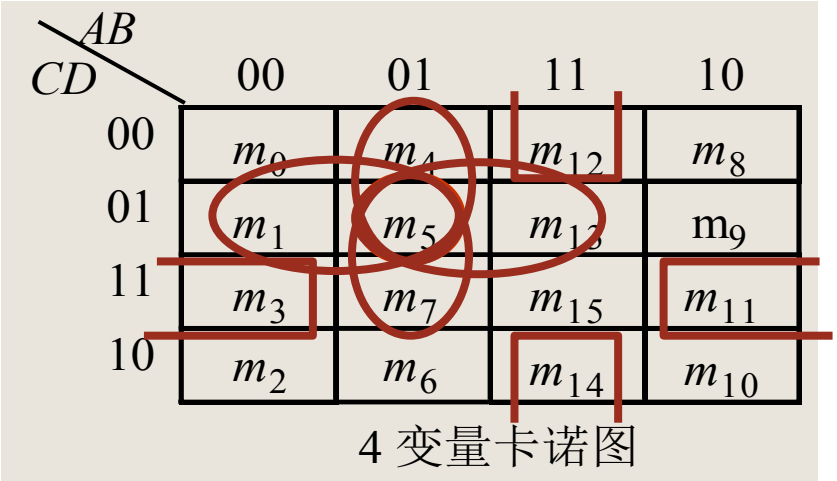
C \ AB	00	01	11	10
0	m_0	m_2	m_6	m_4
1	m_1	m_3	m_7	m_5

3 变量卡诺图

逻辑函数的卡诺图

卡诺图：将逻辑函数的最小项排列成矩阵形式，并且使矩阵的横方向和纵方向的逻辑变量的取值，按照格雷码(循环码)的顺序排列。





卡诺图的特点是：任意两个逻辑相邻的最小项，在图中也是相邻的。

- 逻辑函数是以真值表或者以最小项表达式给出：在卡诺图中，那些使真值表中函数值为1的最小项、或给定逻辑函数中的最小项，相对应的方格内填入1，其余的方格内填入0。

$$Y(A,B,C,D)=\sum m(1,3,4,6,7,11,14,15)$$

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

m_1 m_3 m_4 m_{11} m_7 m_6 m_{15} m_{14}

- 逻辑函数以一般的逻辑表达式给出：先将函数变换为与或表达式（不必变换为最小项之和的形式），然后在卡诺图上，找出包含每一个乘积项的那些最小项（该乘积项就是这些最小项的公因子），相对应的方格内填入1，其余的方格内填入0。

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

变换为与或表达式

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

说明：如果求得了函数Y的反函数$\overline{Y}$，则对$\overline{Y}$中所包含的各个最小项，在卡诺图相应方格内填入0，其余方格内填入1，即可得Y的卡诺图。

$\overline{A}\,\overline{D}$的公因子

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

$\overline{B}C$的公因子

利用卡诺图化简

（1）任何两个（2^1个）标1的相邻最小项，可以合并为一项，并消去一个变量（消去互为反变量的因子，保留公因子）。

$C \backslash AB$	00	01	11	10
0	1	0	0	1
1	0	1	1	0

$$\cancel{\bar{A}}\bar{B}\bar{C} + \cancel{A}\bar{B}\bar{C} = \bar{B}\bar{C}$$

$$\cancel{\bar{A}}BC + \cancel{A}BC = BC$$

$CD \backslash AB$	00	01	11	10
00	0	1	0	0
01	0	0	0	1
11	0	0	0	1
10	0	1	0	0

$$A\bar{B}\cancel{\bar{C}}D + A\bar{B}\cancel{C}D = A\bar{B}D$$

$$\bar{A}B\cancel{\bar{C}}\bar{D} + \bar{A}B\cancel{C}\bar{D} = \bar{A}B\bar{D}$$

（2）任何4个（2^2个）标1的相邻最小项，可以合并为一项，并消去2个变量。（消去互为反变量的因子，保留公因子）。

$C \backslash AB$	00	01	11	10
0	1	1	1	1
1	0	1	1	0

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

$$\bar{A}B\bar{C}+\bar{A}BC+AB\bar{C}+ABC=(\bar{A}\bar{C}+\bar{A}C+A\bar{C}+AC)B=B$$

$CD \backslash AB$	00	01	11	10
00	0	1	0	0
01	1	1	1	1
11	0	1	1	0
10	0	1	0	0

$\bar{C}D$

$\bar{A}B$

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

$\overline{B}\,D$

$B\,\overline{D}$

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

$B\,D$

$\overline{B}\,\overline{D}$

（3）任何8个（2^3个）标1的相邻最小项，可以合并为一项，并消去3个变量。（消去互为反变量的因子，保留公因子）。

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

D

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

$\overline{B}$

相邻最小项的数目必须为2^i个才能合并为一项，并消去 i 个变量。包含的最小项数目越多，即由这些最小项所形成的圈越大，消去的变量也就越多，从而所得到的逻辑表达式就越简单。这就是利用卡诺图化简逻辑函数的基本原理。

逻辑函数图形法化简步骤

逻辑表达式或真值表

$$Y(A,B,C,D)=\sum m(3,5,7,8,11,12,13,15)$$

1

1

卡诺图

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

2

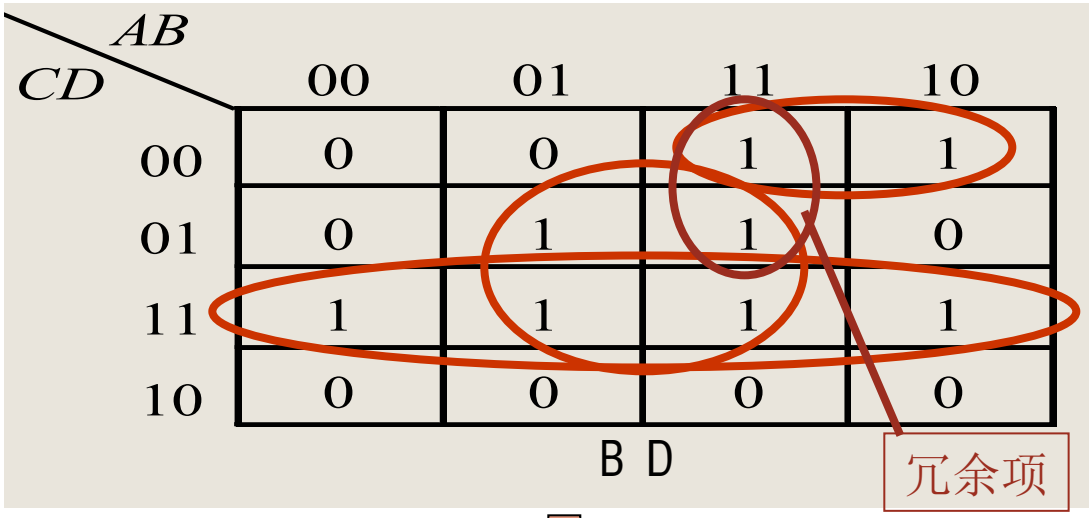
①圈越大越好，但每个圈中标1的方格数目必须为2^i个。②同一个方格可同时画在几个圈内，但每个圈都要有新的方格，否则它就是多余的。③不能漏掉任何一个标1的方格。

合并最小项

3

最简与或表达式

2



3

将代表每个圈的乘积项相加

$$Y(A,B,C,D) = BD + CD + A\bar{C}\bar{D}$$

两点说明:

① 在有些情况下，最小项的圈法不只一种，得到的各个乘积项组成的与或表达式各不相同，哪个是最简的，要经过比较、检查才能确定。

圈的个数越少越简

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

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

不是最简

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

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

最简

② 在有些情况下，不同圈法得到的与或表达式都是最简形式。即一个函数的最简与或表达式不是唯一的。

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

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

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

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

圈的个数相同，每个圈越大越简

包含约束项的逻辑函数化简

约束项：不会出现的变量取值，所对应的最小项。

- 由最小项的性质知道，只有在函数的与或式中出现的最小项，其值才会为1。而约束项是不出现的最小项，所以其值总是等于0。
- **约束条件**：由约束项加起来所构成的值为0的逻辑表达式。因为约束项的值恒为0，所以约束条件是一个值恒为0的条件等式。

例：判断一位十进制数是否为偶数。

A B C D	Y	A B C D	Y	说 明
0 0 0 0	1	1 0 0 0	1	
0 0 0 1	0	1 0 0 1	0	
0 0 1 0	1	1 0 1 0	×	不会出现
0 0 1 1	0	1 0 1 1	×	不会出现
0 1 0 0	1	1 1 0 0	×	不会出现
0 1 0 1	0	1 1 0 1	×	不会出现
0 1 1 0	1	1 1 1 0	×	不会出现
0 1 1 1	0	1 1 1 1	×	不会出现

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

输入变量A，B，C，D取值为0000-1001时，逻辑函数Y有确定的值，根据题意，偶数时为1，奇数时为0。

$$Y(A,B,C,D)=\Sigma m(0,2,4,6,8)$$

A，B，C，D取值为1010 -1111的情况不会出现或不允许出现，对应的最小项属于约束项。约束项之和构成的逻辑表达式叫做**约束条件**，用一个值恒为 0 的条件等式表示。

$$\Sigma d(10,11,12,13,14,15)=0$$

含有约束条件的逻辑函数可以表示成如下形式：

$$F(A,B,C,D)=\Sigma m(0,2,4,6,8)+\Sigma d(10,11,12,13,14,15)$$

在逻辑函数的化简中，充分利用约束项可以得到更加简单的逻辑表达式，其相应的逻辑电路也更简单。

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

不利用约束项的化简结果为：

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

利用约束项的化简结果为：

$$Y = \overline{D}$$

逻辑函数的表示方法

- 真值表
- 卡诺图
- 逻辑表达式
- 逻辑图
- 波形图

① 真值表

真值表：是由变量的所有可能取值组合及其对应的函数值所构成的表格。

真值表列写方法：每一个变量均有0、1两种取值，n个变量共有2^n种不同的取值，将这2^n种不同的取值按顺序（一般按二进制递增规律）排列起来，同时在相应位置上填入函数的值，便可得到逻辑函数的真值表。

例如：当A=B=1、或B=C=1时，函数Y=1；否则Y=0。

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	0
1	1	0	1
1	1	1	1

② 卡诺图

卡诺图：是由表示变量的所有可能取值组合的小方格所构成的图形。

逻辑函数卡诺图的填写方法：在那些使函数值为1的变量取值组合所对应的小方格内填入1，其余的方格内填入0，便得到该函数的卡诺图。

C \ AB	00	01	11	10
0	0	0	1	0
1	0	1	1	0

③ 逻辑表达式

逻辑表达式：是由逻辑变量和与、或、非3种运算符连接起来所构成的式子。

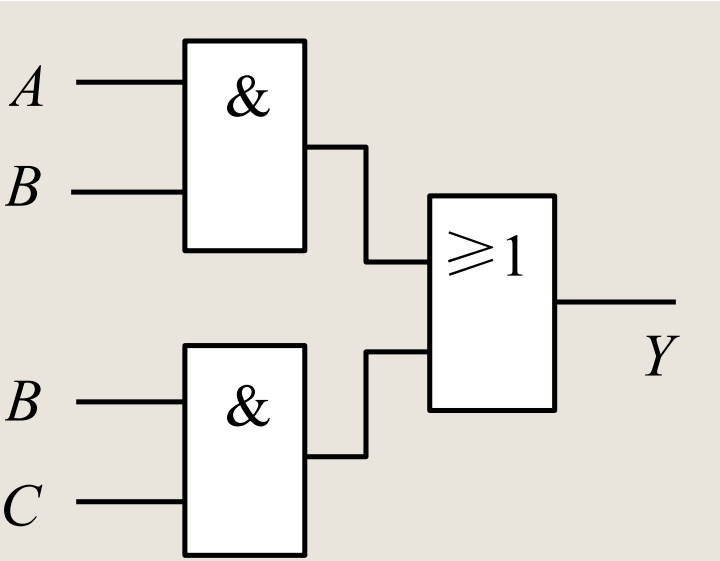
函数的标准与或表达式的列写方法：将函数的真值表中那些使函数值为1的最小项相加，便得到函数的标准与或表达式。

$$Y = \overline{A}BC + AB\overline{C} + ABC$$
$$= \sum m(3,6,7)$$

④ 逻辑图

逻辑图：是由表示逻辑运算的逻辑符号所构成的图形。

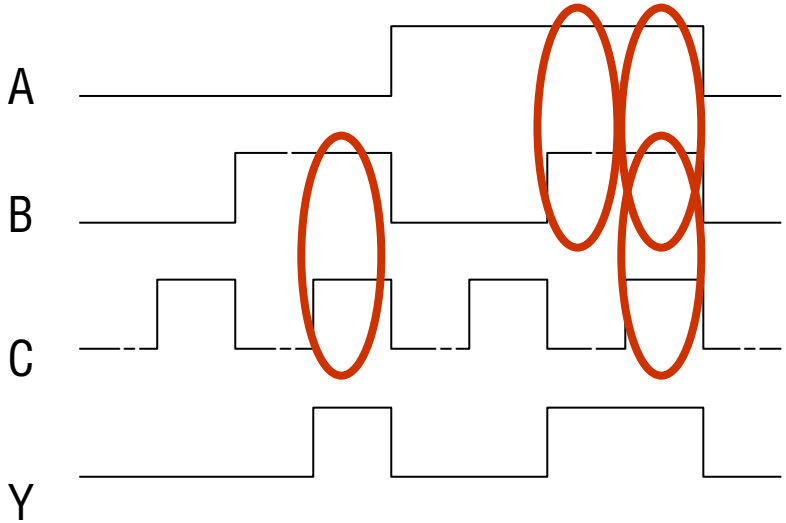
$$Y = AB + BC$$



⑤ 波形图

波形图：是由输入变量的所有可能取值组合的高、低电平及其对应的输出函数值的高、低电平所构成的图形。

$$Y = AB + BC$$



2.1.2 门电路

半导体的导电特性

原子是由带正电的原子核和负电的电子组成，电子分几层围绕原子核不停的运动。在同一原子中，原子核对内层电子吸引力较大，对外层电子吸引力较小。影响物质电性能的主要是外层电子。

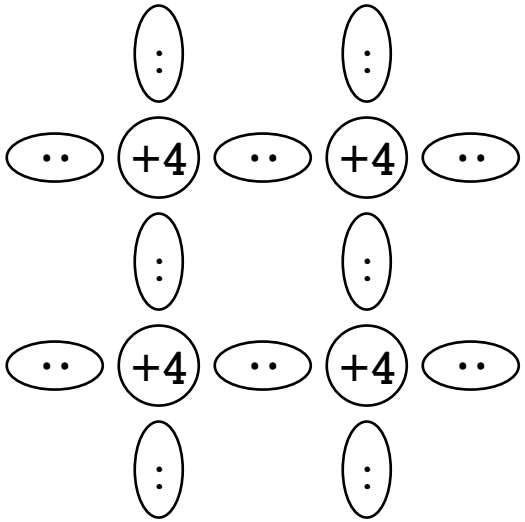
导体：金属的外层电子受原子核的束缚力最小，因此有大量电子能够挣脱束缚成为自由电子。这些自由电子成为载流子，它们在外电场的作用下作定向运动形成电流。所以金属是导体。

绝缘体：绝缘体中，原子的外层电子受原子核的束缚力很大，不容易挣脱出来。由于载流子很少，所以绝缘体导电性能很差。

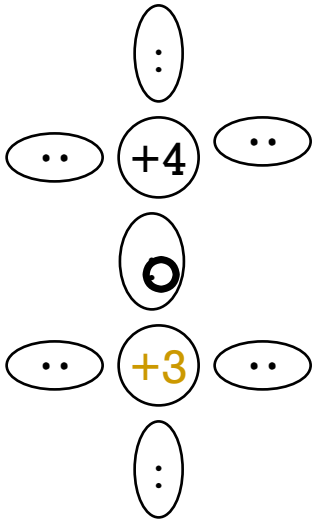
半导体：半导体的结构比较特殊，外层电子既不像导体那样容易挣脱，也不象绝缘体那样束缚很紧。其导电性能界于导体和绝缘体之间。

本征半导体：

本征半导体是指那些四价元素，例如硅或锗，它们的核外有四个电子，可以构成稳定的晶格结构。每相邻两原子都共用一对电子，电子对中的任一电子既围绕自身原子核运动，也时常出现在相邻原子核所属的轨道上。这样的组合称为“共价键结构”。

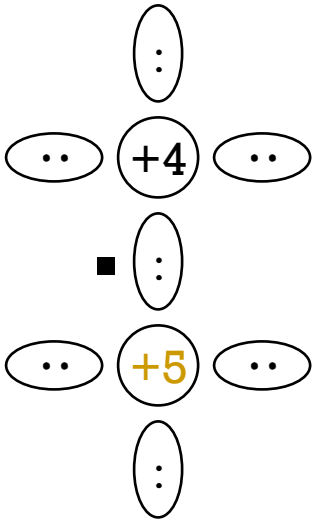


本征半导体本身并没有使用价值。但是如果向本征半导体中有目的掺入某些杂质，就可以大大改变它的导电性能。例如，分别掺入三价元素硼和五价元素磷，就可以分别得到P型半导体和N型半导体。



P型半导体
向四价的本征半导体(硅或锗)注入三价原子(硼或铝)，就构成P型半导体。

空穴多，吸引电子

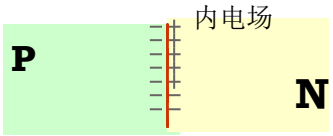


N型半导体
向四价的本征半导体(硅或锗)注入五价原子(砷或磷)，就构成N型半导体。

电子多

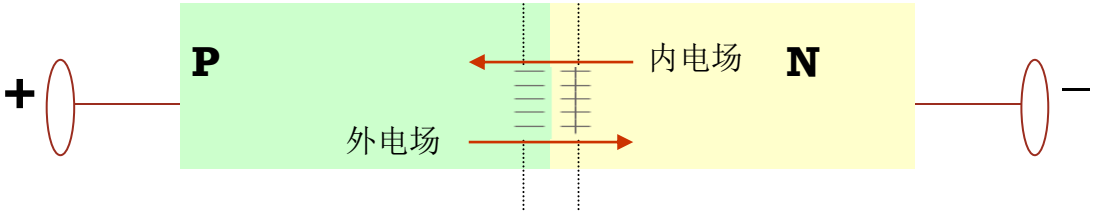
PN结的单向导电性

PN结结构：一块硅片上，用不同的掺杂工艺使其一边形成N型半导体，另一边形成P型半导体，两种半导体的交界面附近的区域为PN结。

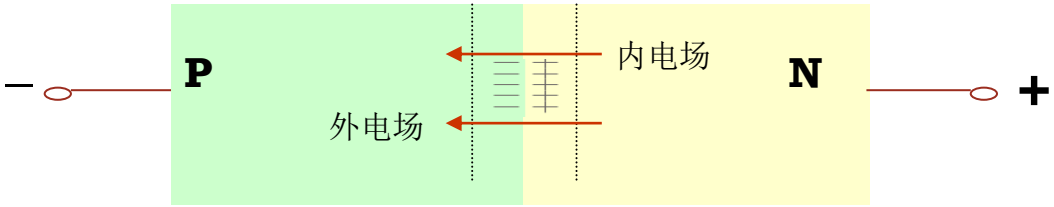


在PN结两端适当加以外部电压，可以实现单向导电。

加正向电压时PN结导通：加正向电压时，内外电场方向相反，阻挡层变窄，相当于PN结的电阻很小。

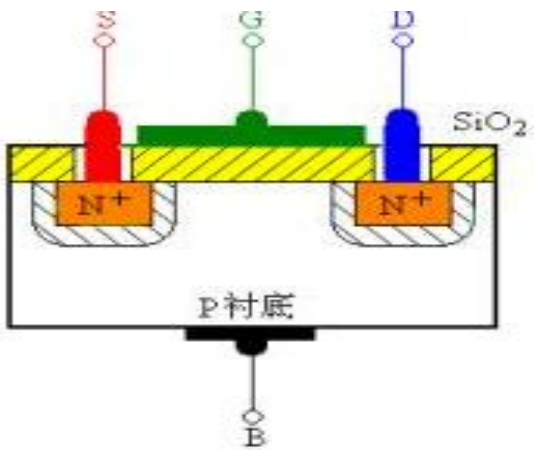


加反向电压时PN结截止：加反向电压时，内外电场方向相同，阻挡层变宽，相当于PN结的电阻很大。

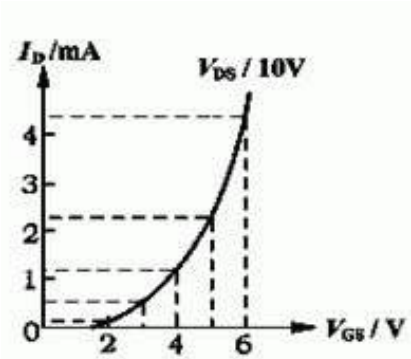


MOS管：由金属-氧化物-半导体构成的，分为N沟道和P沟道MOS管

- N沟道MOS管：在P型衬底上，利用光刻、扩散等方法，制作出两个高掺杂浓度N^+型区，并引出电极，源极S和漏极D，同时在源极和漏极之间的二氧化硅SiO_2绝缘层上，制作一个金属电极栅极G。



箭头向内.N沟道



MOS管是通过栅极电压u_{GS}控制其工作状态。

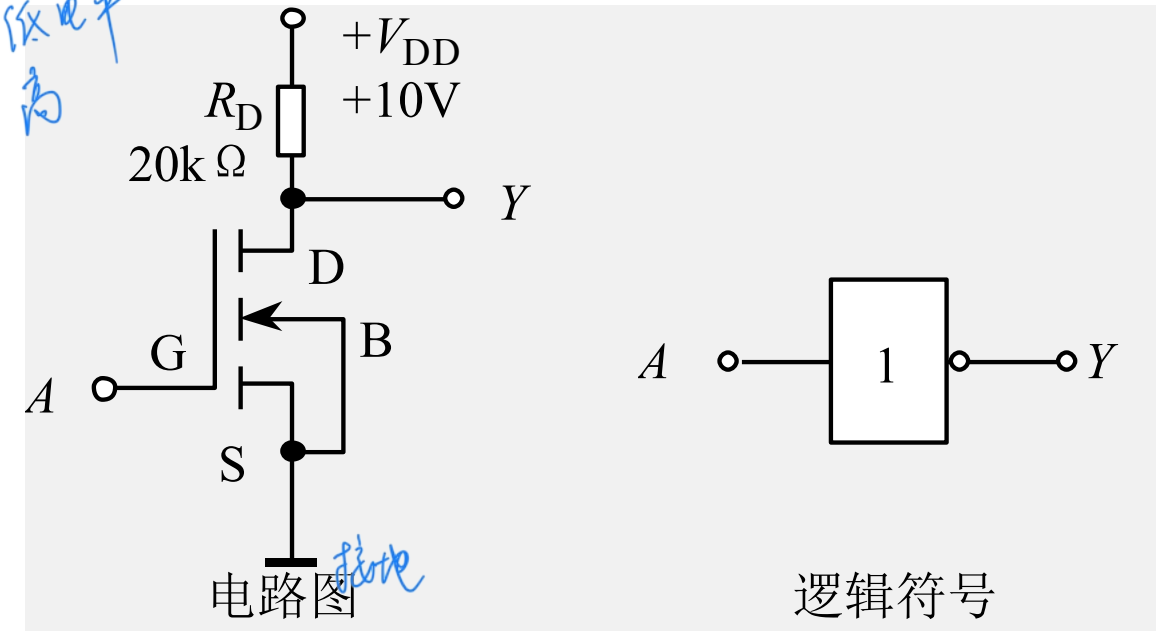
- 截止状态：$U_{GS} \le U_{TN}$, $i_D=0$
- 导通状态：$U_{GS} > U_{TN}$, $i_D>0$

- P沟道MOS管：与N沟道增强型MOS管有明显的对偶关系。其衬底是N型硅半导体，漏极和源极是两个高浓度掺杂的P^+区，而且它的u_{GS}、u_{DS}、开启电压U_{TP}都是负值。

MOS管非门

输入高电平，输出低电平
低 高

N沟道



N沟道　　D-S间无电流

①当$u_A = 0V$时，由于$u_{GS} = u_A = 0V$，小于开启电压U_{TN}，所以MOS管截止。输出电压为$u_Y = V_{DD} = 10V$。

Y与S极断开

②当$u_A = 10V$时，由于$u_{GS} = u_A = 10V$，大于开启电压U_{TN}，所以MOS管导通，导通电阻很小，输出电压为$u_Y \approx 0V$。

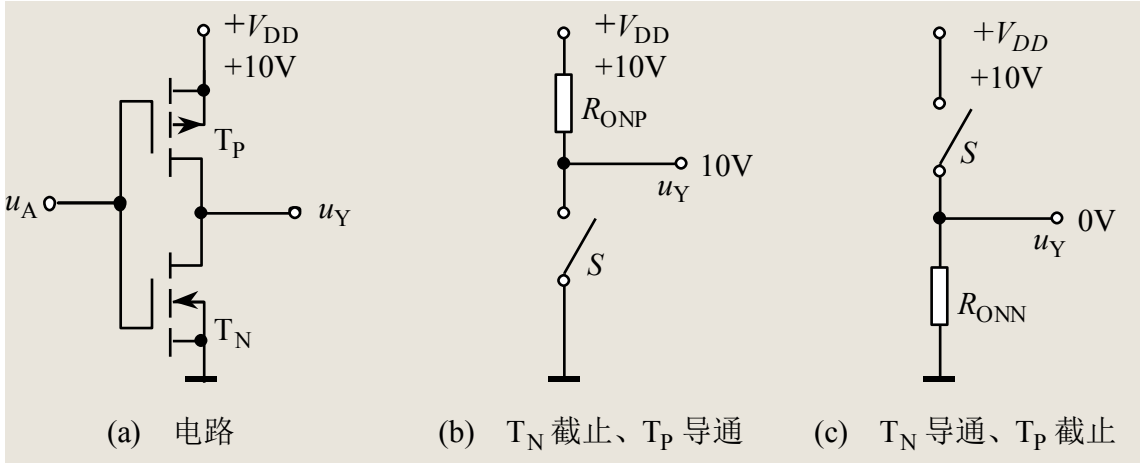
$$Y = \overline{A}$$

CMOS集成电路

- CMOS（Complementary Metal Oxide Semiconductor）是互补对称式金属-氧化物-半导体电路的缩写，CMOS具有电压控制、功耗极小、连接方便等一系列优点，是目前应用最广泛的集成电路之一。
- 用P沟道增强型MOS管和N沟道增强型MOS管，按照互补对称形式连接构成。

CMOS非门

$$Y = \overline{A}$$



(a) 电路　(b) T_N 截止、T_P 导通　(c) T_N 导通、T_P 截止

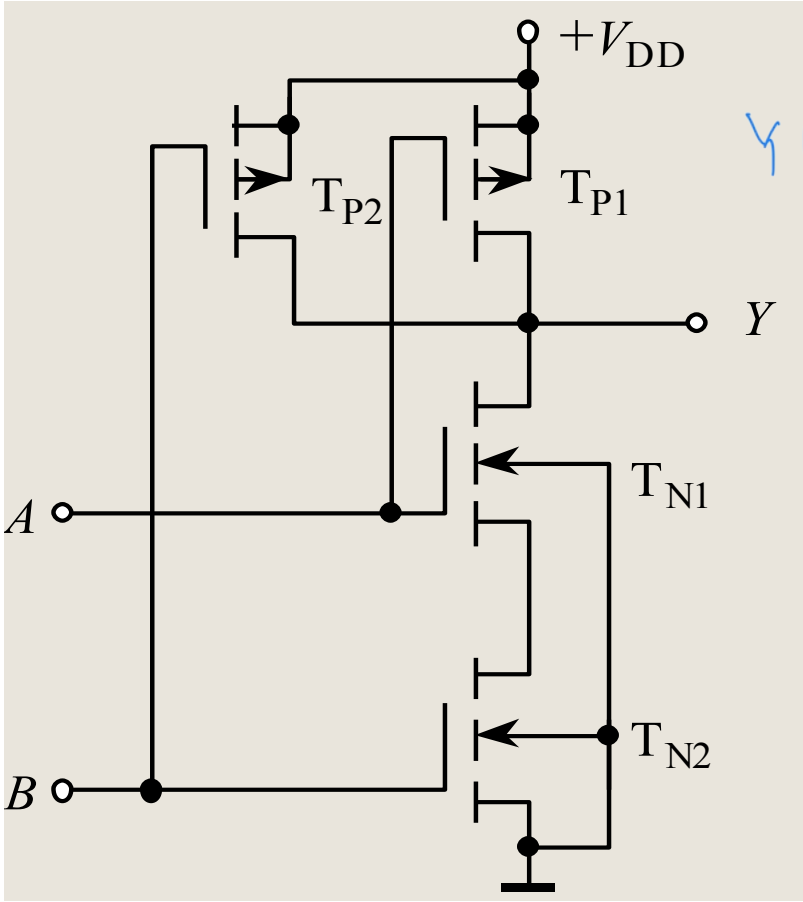
N: 0V　P: -10V

（1）$u_A = 0V$时，T_N截止，T_P导通。输出电压$u_Y = V_{DD} = 10V$。

（2）$u_A = 10V$时，T_N导通，T_P截止。输出电压$u_Y = 0V$。

N: 10V

CMOS与非门



Y他断开

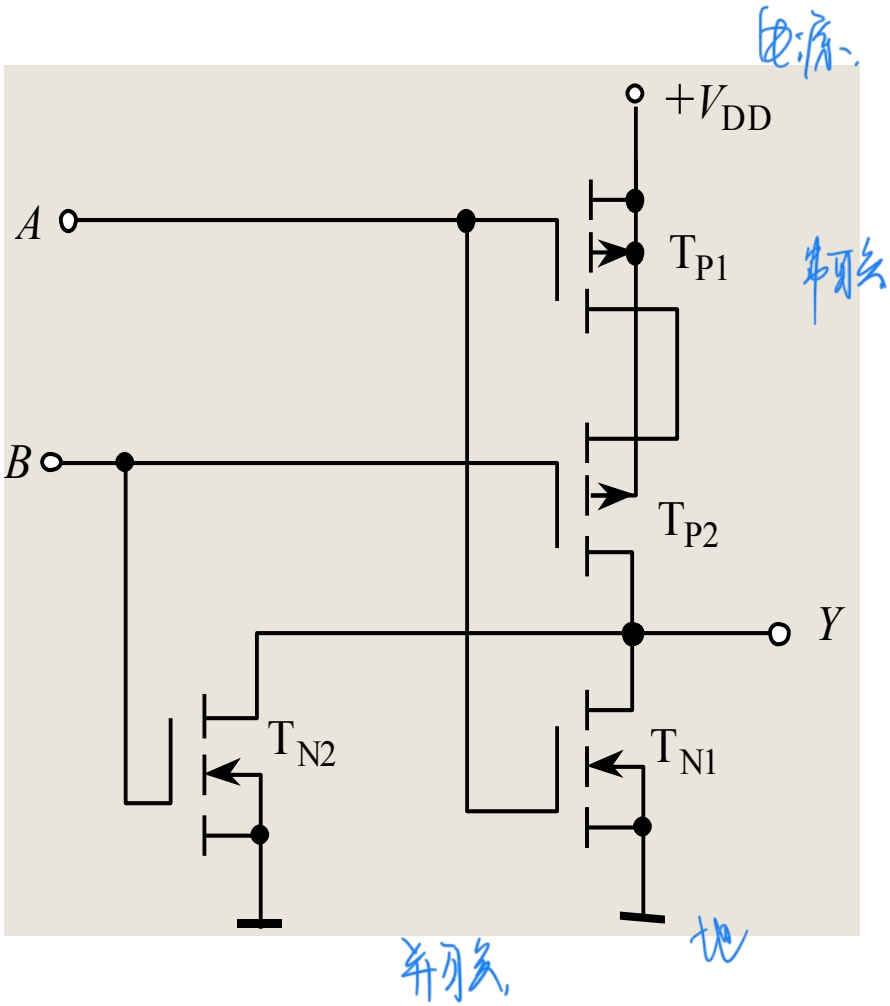
Y U_{DD} 导通

①*A*、*B*当中有一个或全为低电平时，T_{N1}、T_{N2}中有一个或全部截止，T_{P1}、T_{P2}中有一个或全部导通，输出*Y*为高电平。

②只有当输入*A*、*B*全为高电平时，T_{N1}和T_{N2}才会都导通，T_{P1}和T_{P2}才会都截止，输出*Y*才会为低电平。

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

CMOS或非门

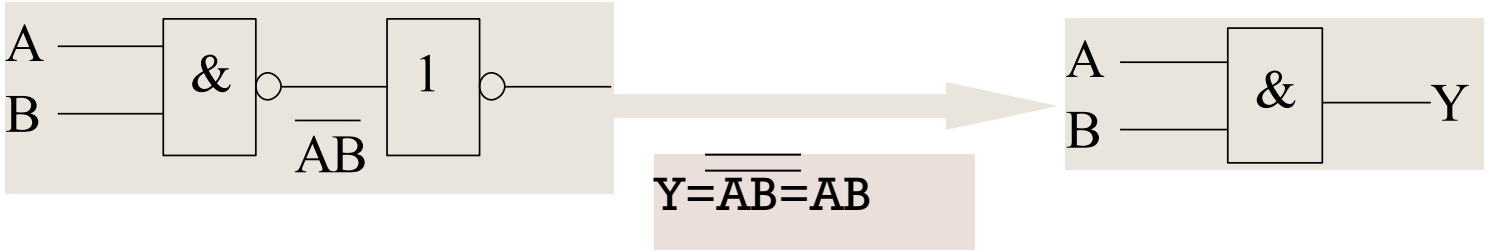


①只要输入*A*、*B*当中有一个或全为高电平，T_{P1}、T_{P2}中有一个或全部截止，T_{N1}、T_{N2}中有一个或全部导通，输出*Y*为低电平。

②只有当*A*、*B*全为低电平时，T_{P1}和T_{P2}才会都导通，T_{N1}和T_{N2}才会都截止，输出*Y*才会为高电平。

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

与门

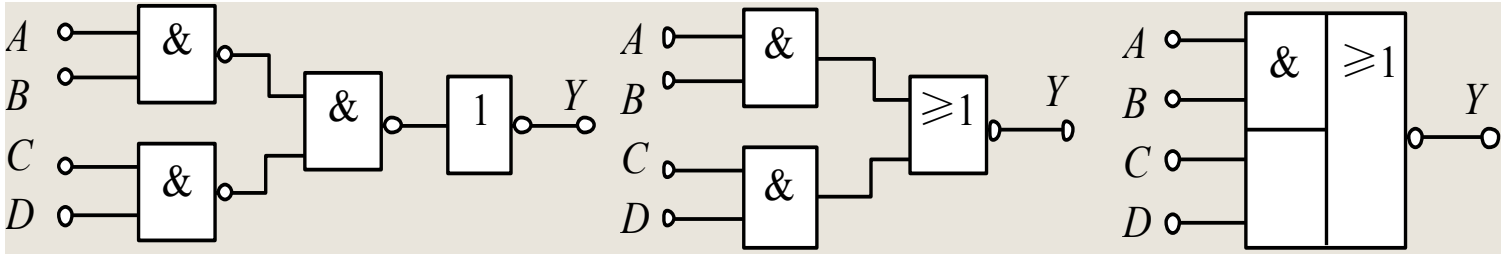


$$Y=\overline{\overline{AB}}=AB$$

或门

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

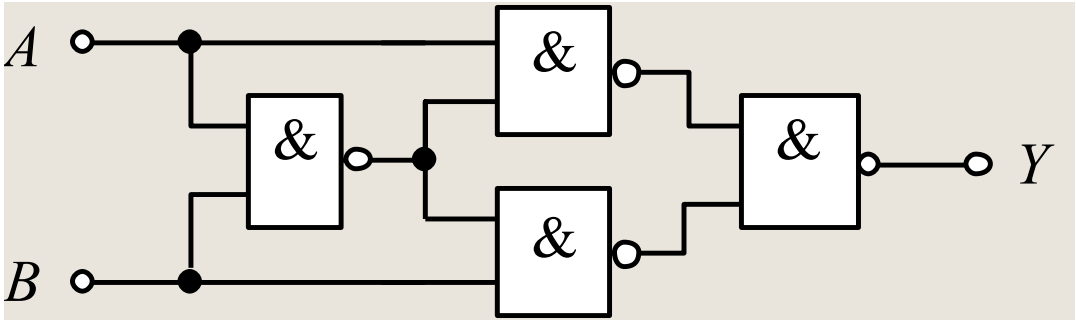
CMOS与或非门



(a) 由与非门和反相器构成　(b) 由与门和或非门构成　(c) 逻辑符号

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

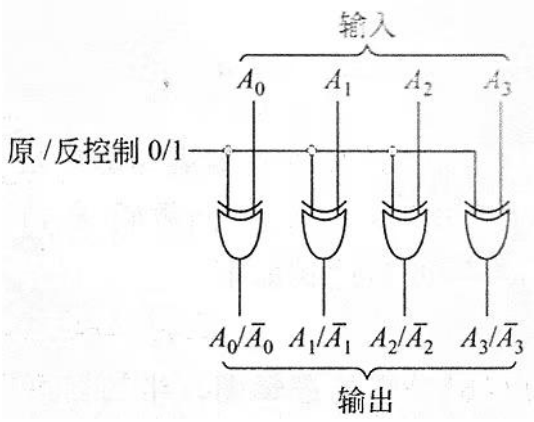
CMOS异或门



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

异或门的简单应用

- 可控原/反码输出电路



^0 原

^1 反

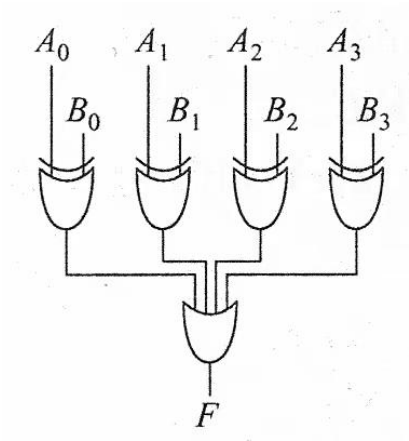
1^0 =1

0^0 =0

0^1 =1

1^1 =0

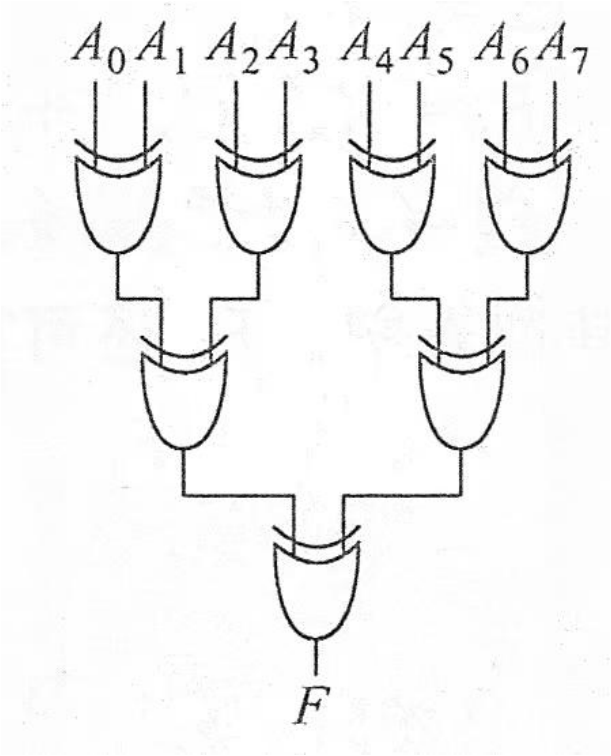
- 数码比较器



- 奇偶检测电路

1的个数是奇数个 → 1

(偶数) → 0



2.2 计算机中的数字逻辑电路

- 计算机的基本组成（冯诺依曼体系结构）
  - 控制器、运算器、存储器、输入设备、输出设备
  - 每部分都是由集成电路板组成，每个电路板上是由许多集成电路芯片组成，每个集成电路的基础电路是逻辑门电路（gate）。
- 数字逻辑电路可以分为
  - 组合逻辑电路
  - 时序逻辑电路

2.2.1 组合逻辑电路

输出不会作为输入反馈

- 加法器
- 译码器 地址 → 选存储单元
- 数据选择器
- 只读存储器（ROM）
- 算术逻辑单元（ALU） 运算器

组合逻辑电路

组合逻辑电路：输出仅由输入决定，与电路当前状态无关；电路结构中无反馈环路（无记忆）

$$Y_0=F_0(I_0,I_1,\ldots I_{n-1})$$
$$Y_1=F_1(I_0,I_1,\ldots I_{n-1})$$
$$\ldots$$
$$Y_{m-1}=F_{m-1}(I_0,I_1,\ldots I_{n-1})$$

$$Y(t_n)=F[I(t_n)]$$

t_n时刻的稳定输出$Y(t_n)$,仅取决于t_n时刻的输入$I(t_n)$

组合逻辑电路的分析

- 电路分析：已知逻辑电路，寻找输出与输入之间的逻辑关系，确定电路功能的过程。
- 组合逻辑电路的分析步骤：
  - 根据给定的逻辑图，从输入到输出逐级写出逻辑函数表达式；
  - 用公式法或卡诺图法化简逻辑函数表达式；
  - 根据化简的逻辑函数表达式列出真值表 (可省略）；
  - 对逻辑函数表达式或真值表进行分析，概括出给定组合逻辑电路的逻辑功能。

组合逻辑电路的基本分析方法

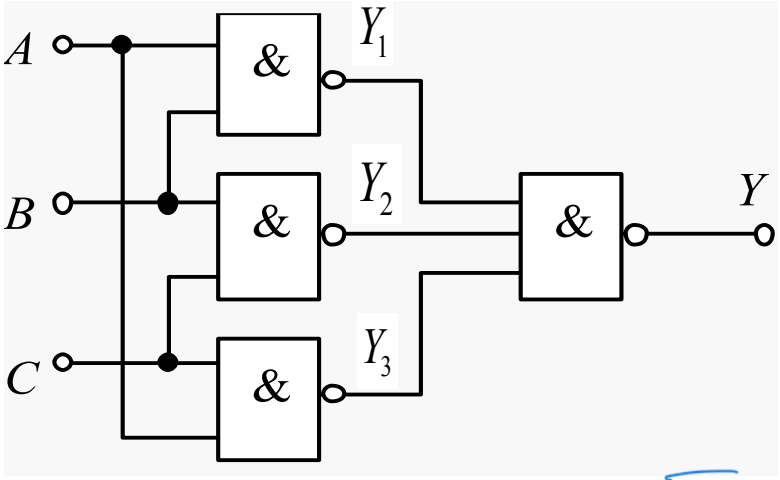
逻辑图

1 从输入到输出逐级写出

逻辑表达式

化简 2

最简与或表达式



1

$$Y_1 = \overline{AB}$$
$$Y_2 = \overline{BC}$$
$$Y_3 = \overline{CA}$$

$$Y = \overline{Y_1 Y_2 Y_3} = \overline{\overline{AB}\ \overline{BC}\ \overline{AC}}$$

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

$\overline{AB} = \bar{A}\bar{B}$

2

$$Y = AB + BC + CA$$

$$Y = AB + BC + CA$$

最简与或表达式

3

真值表

4

说明电路的功能

3

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

4

当输入A、B、C中有2个或3个为1时，输出Y为1，否则输出Y为0。

可以认为电路是一种3人表决用的组合电路：只要有2票或3票同意，表决就通过。

组合逻辑电路的设计

- 电路设计：根据所要求实现的逻辑功能，设计出相应的逻辑电路。
  - 通常以电路简单、所用器件最少为目标。
- 设计步骤：
  1. 分析设计要求，对问题逻辑抽象，确定输入、输出变量。
  2. 对输入和输出变量赋予0、1值，根据输入输出之间的因果关系，列出输入输出对应关系表，即真值表。
  3. 根据真值表写出逻辑表达式或卡诺图，并根据设计要求进行化简和变换。
  4. 根据化简式画出逻辑电路图。

加法器——半加器和全加器

半加器

能对两个1位二进制数进行相加，而求得和及进位的逻辑电路称为**半加器**。

半加器真值表

本位的和

加数

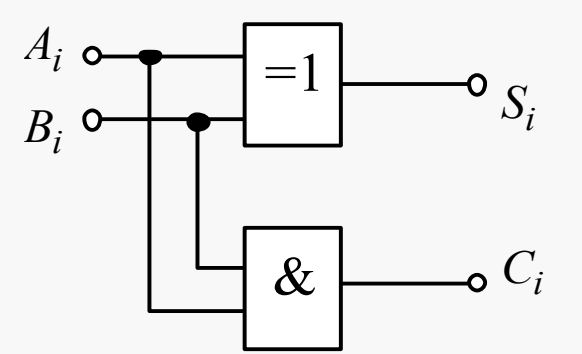
向高位的进位

A_i	B_i	S_i	C_i
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

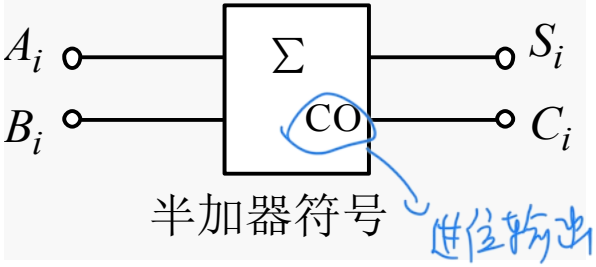
真值表→逻辑表达式

$$S_i = \overline{A}_i B_i + A_i \overline{B}_i = A_i \oplus B_i$$ 异或

$$C_i = A_i B_i$$ 都为1



半加器电路图



半加器符号

全加器

能对两个1位二进制数进行相加并考虑低位来的进位，即相当于3个1位二进制数相加，求得和及进位的逻辑电路称为**全加器**。

A_i	B_i	C_{i-1}	S_i	C_i
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

C_{i-1} \ A_iB_i	00	01	11	10
0	0	1	0	1
1	1	0	1	0

S_i 的卡诺图

$$S_i = m_1 + m_2 + m_4 + m_7$$

C_{i-1} \ A_iB_i	00	01	11	10
0	0	0	1	0
1	0	1	1	1

C_i 的卡诺图

A_iB_i B_iC_{i-1} A_iC_{i-1}

A_i、B_i：加数，C_{i-1}：低位来的进位，S_i：本位的和，C_i：向高位的进位。

$$C_i=A_iB_i+A_iC_{i-1}+B_iC_{i-1}$$

$$C_i=A_iB_i+m_5+m_3$$

全加器的逻辑图和逻辑符号

$$S_i = m_1 + m_2 + m_4 + m_7 = \overline{A}_i\overline{B}_iC_{i-1} + \overline{A}_iB_i\overline{C}_{i-1} + A_i\overline{B}_i\overline{C}_{i-1} + A_iB_iC_{i-1}$$

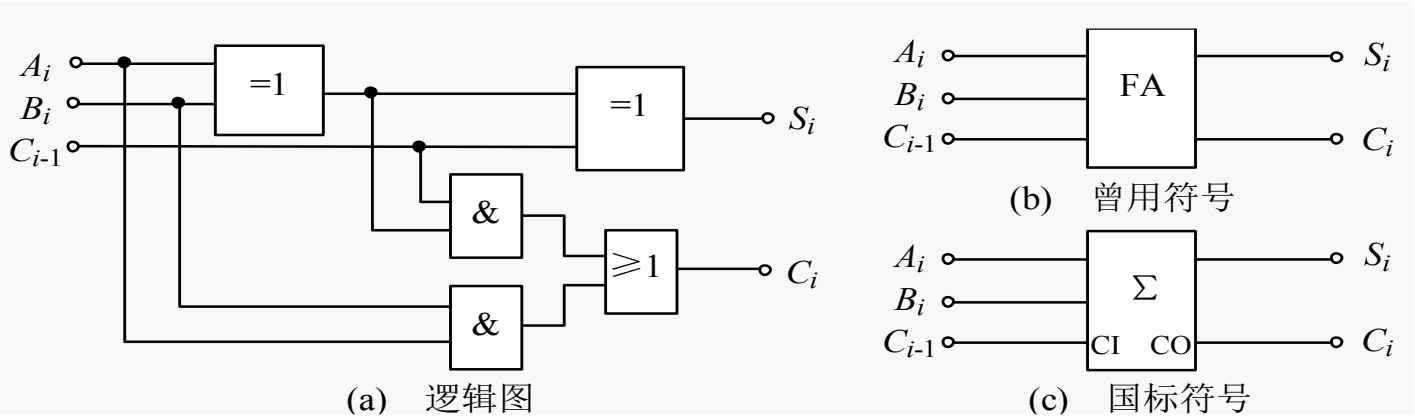
$$= \overline{A}_i(\overline{B}_iC_{i-1} + B_i\overline{C}_{i-1}) + A_i(\overline{B}_i\overline{C}_{i-1} + B_iC_{i-1}) = \overline{A}_i(B_i \oplus C_{i-1}) + A_i\overline{(B_i \oplus C_{i-1})}$$

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

奇数位是偶数个1

$$C_i = m_3 + m_5 + A_iB_i = \overline{A}_iB_iC_{i-1} + A_i\overline{B}_iC_{i-1} + A_iB_i = (\overline{A}_iB_i + A_i\overline{B}_i)C_{i-1} + A_iB$$

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

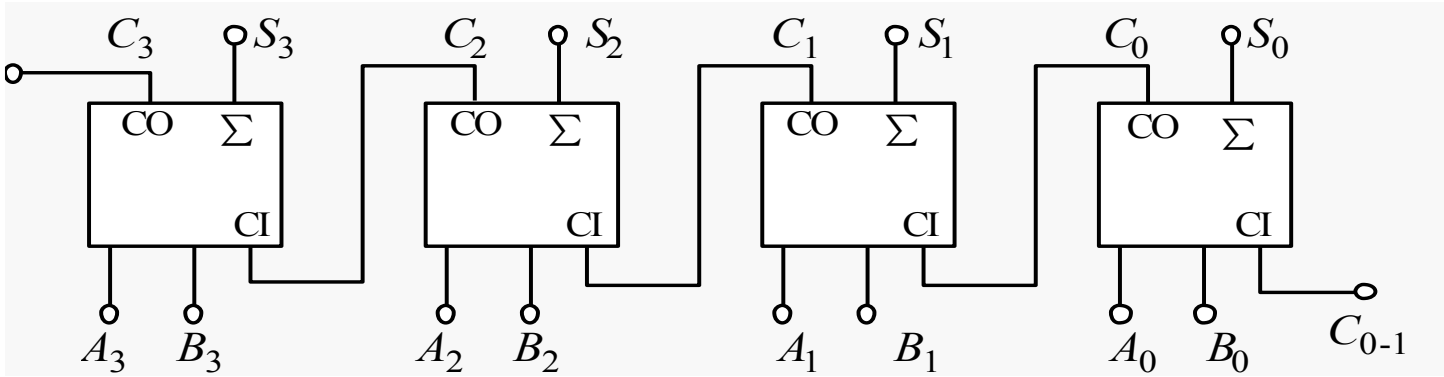


加法器

实现多位二进制数相加的电路称为**加法器**。

4位串行进位加法器

构成：把 n 个全加器串联起来，低位全加器的进位输出连接到相邻的高位全加器的进位输入。



特点：进位信号是由低位向高位逐级传递的，速度不高。

超前进位加法器（并行进位加法器）

超前进位加法器：作加法运算时，进位输入信号由二进制加数直接产生，而不需要等待低位加数相加后的进位，这样的加法器称之为超前进位加法器。

以4位二进制加法器为例：根据全加器的逻辑功能，我们可以得到4位二进制($D_3D_2D_1D_0$)加法器中，D_{i+1}(*i*= 0-3)位全加器的进位输入信号C_i的表达式为：

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

进位生成项 $G_i = A_iB_i$

低位进位传递条件 $P_i = A_i \oplus B_i$

进位表达式 $C_i = A_iB_i + (A_i \oplus B_i)C_{i-1} = G_i + P_iC_{i-1}$

和表达式 $S_i = A_i \oplus B_i \oplus C_{i-1} = P_i \oplus C_{i-1}$

进位生成项 $G_i = A_i B_i$

低位进位传递条件 $P_i = A_i \oplus B_i$

进位表达式 $C_i = A_i B_i + (A_i \oplus B_i) C_{i-1} = G_i + P_i C_{i-1}$

和表达式 $S_i = A_i \oplus B_i \oplus C_{i-1} = P_i \oplus C_{i-1}$

4位超前进位加法器递推公式

$$\begin{cases} S_0 = P_0 \oplus C_{0-1} \\ C_0 = G_0 + P_0 C_{0-1} \end{cases}$$

$$\begin{cases} S_1 = P_1 \oplus C_0 = P_1 \oplus (G_0 + P_0 C_{0-1}) \\ C_1 = G_1 + P_1 C_0 = G_1 + P_1 G_0 + P_1 P_0 C_{0-1} \end{cases}$$

$$\begin{cases} S_2 = P_2 \oplus C_1 \\ C_2 = G_2 + P_2 C_1 = G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_{0-1} \end{cases}$$

$$\begin{cases} S_3 = P_3 \oplus C_2 \\ C_3 = G_3 + P_3 C_2 = G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 G_0 + P_3 P_2 P_1 P_0 C_{0-1} \end{cases}$$

只要$A_3A_2A_1A_0$、$B_3B_2B_1B_0$和C_{0-1}给出后，便可以按上式直接确定$C_3C_2C_1C_0$的值，提高运算速度，因为高位的运算不需再等待低位的进位。

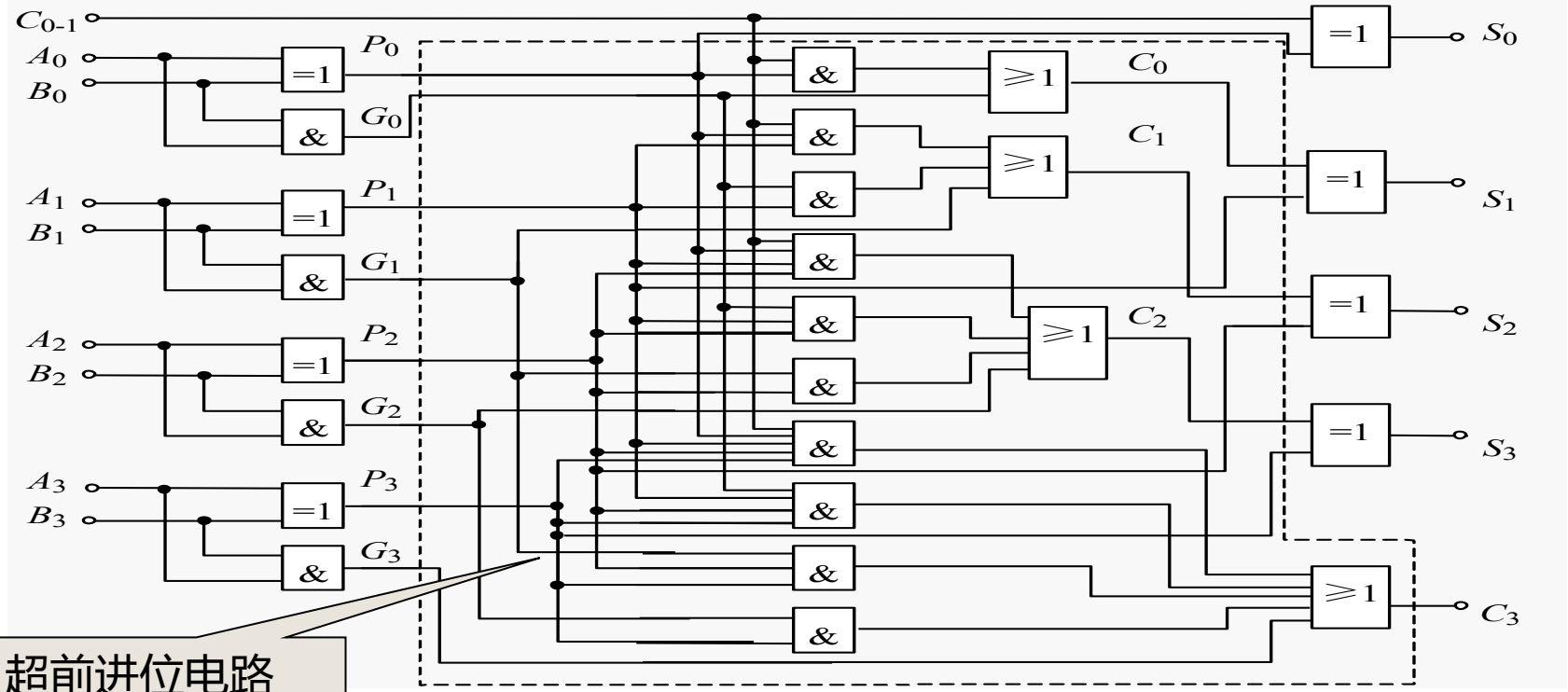
4位超前进位加法器

$$\begin{cases} S_0 = P_0 \oplus C_{0-1} \\ C_0 = G_0 + P_0 C_{0-1} \end{cases}$$

$$\begin{cases} S_1 = P_1 \oplus C_0 \\ C_1 = G_1 + P_1 C_0 = G_1 + P_1 G_0 + P_1 P_0 C_{0-1} \end{cases}$$

$$\begin{cases} S_2 = P_2 \oplus C_1 \\ C_2 = G_2 + P_2 C_1 = G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_{0-1} \end{cases}$$

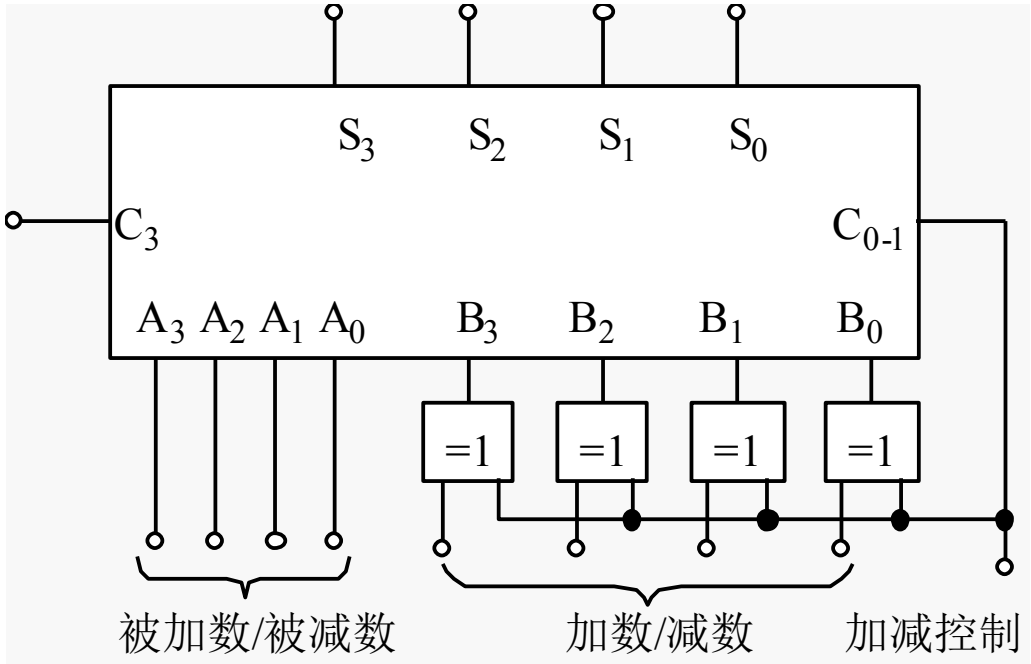
$$\begin{cases} S_3 = P_3 \oplus C_2 \\ C_3 = G_3 + P_3 C_2 = G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 G_0 + P_3 P_2 P_1 P_0 C_{0-1} \end{cases}$$



超前进位电路

加法实现减法运算

加减法运算



不影响　　A+B+0

$C_{0-1}=0$时，$B\oplus 0=B$，电路执行A+B运算；

$C_{0-1}=1$时，$B\oplus 1=\overline{B}$，电路执行A - B=A+$\overline{B}$+1运算。　　补码，取反+1

$A+\overline{B}+C_{0-1}=A+\overline{B}+1$

译码器

指令 → Controller

- 将二进制代码所表示的信息翻译成对应输出的高低电平信号的过程称为译码。实现译码功能的电路称为译码器（Decoder）。
- 地址译码，指令译码，显示译码等应用。

地址→存储单元

3位二进制译码真值表

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

输入：3位二进制代码

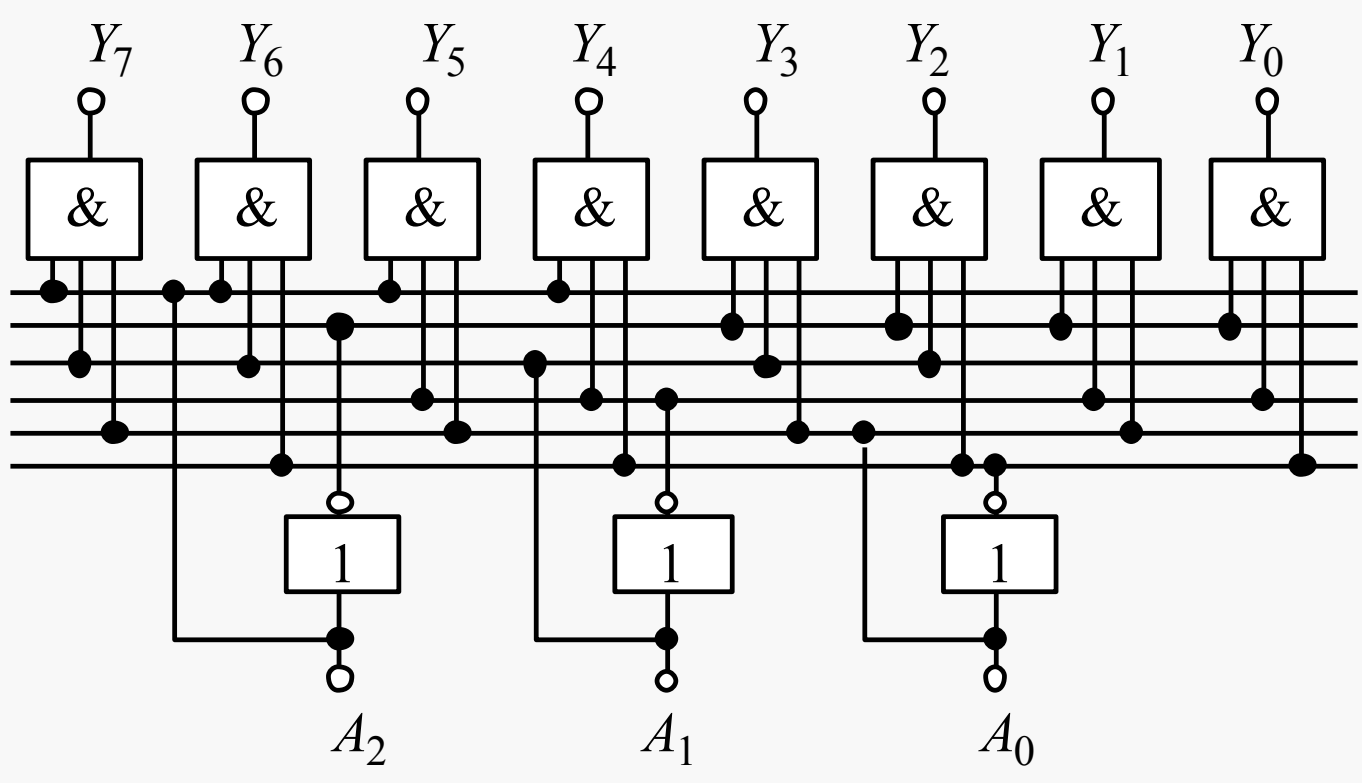
输出：8个互斥的信号

3 线-8 线译码器

逻辑表达式

$$
\begin{cases}
Y_0 = \overline{A}_2\overline{A}_1\overline{A}_0 \\
Y_1 = \overline{A}_2\overline{A}_1 A_0 \\
Y_2 = \overline{A}_2 A_1\overline{A}_0 \\
Y_3 = \overline{A}_2 A_1 A_0 \\
Y_4 = A_2\overline{A}_1\overline{A}_0 \\
Y_5 = A_2\overline{A}_1 A_0 \\
Y_6 = A_2 A_1\overline{A}_0 \\
Y_7 = A_2 A_1 A_0
\end{cases}
$$

逻辑图

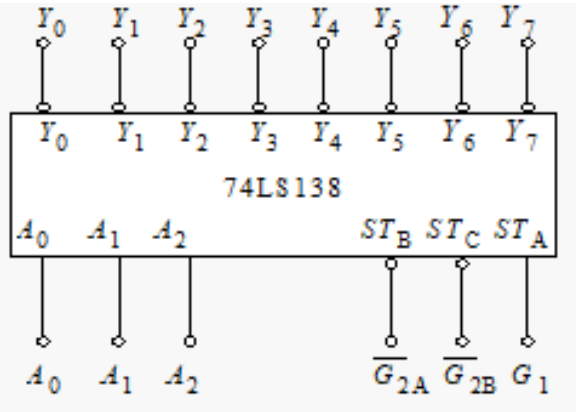


电路特点：与门组成的阵列

集成3线-8线译码器

$$\overline{G}_2 = \overline{G}_{2A} + \overline{G}_{2B}$$

输入					输出							
使能		选择										
G_1	$\overline{G}_2$	A_2	A_1	A_0	$\overline{Y}_7$	$\overline{Y}_6$	$\overline{Y}_5$	$\overline{Y}_4$	$\overline{Y}_3$	$\overline{Y}_2$	$\overline{Y}_1$	$\overline{Y}_0$
×	1	×	×	×	1	1	1	1	1	1	1	1
0	×	×	×	×	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	1	1	1	1	1	1



输入：自然二进制码

输出：低电平有效

数据选择器

- 在多路数据传送过程中，能够根据需要将其中任一路挑选出来的电路，叫做数据选择器，或多路选择器（MUX）。
- 多路选择器可应用于数据通路，包括ALU中操作数的选择，寄存器选择输入数据来源等。

4选1数据选择器

真值表

地址变量

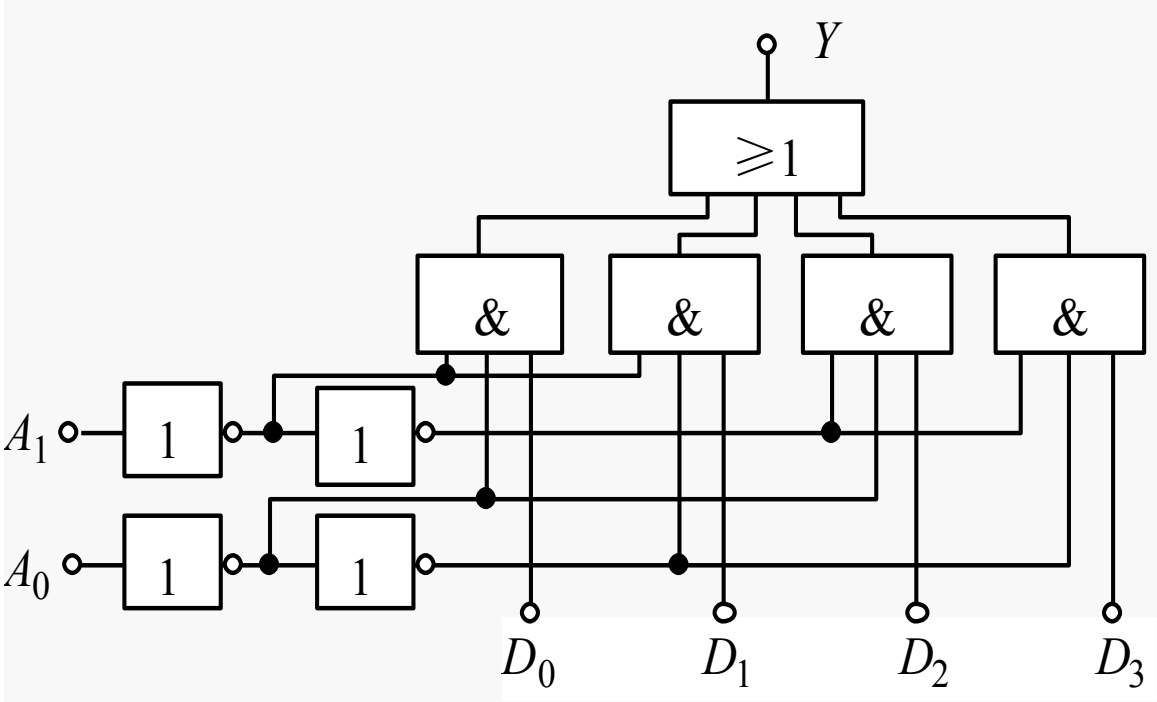
输入数据

输入			输出
D	A_1	A_0	Y
D_0	0	0	D_0
D_1	0	1	D_1
D_2	1	0	D_2
D_3	1	1	D_3

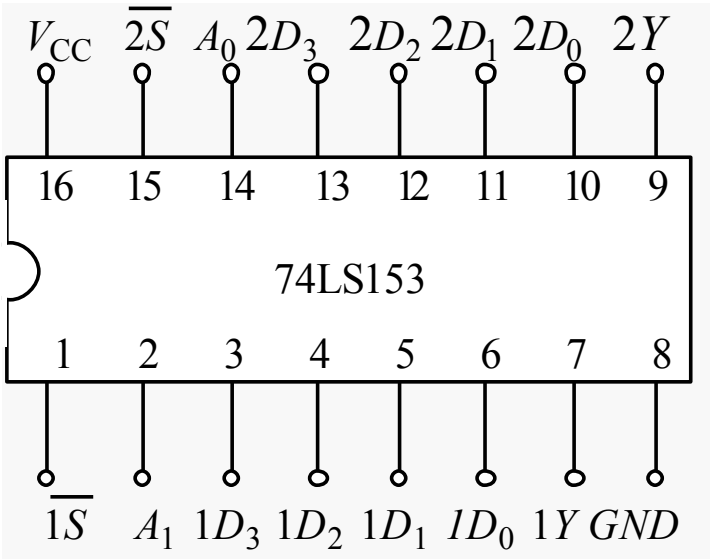
由地址码决定从 4 路输入中选择哪 1 路输出。

$$Y = D_0\overline{A}_1\overline{A}_0 + D_1\overline{A}_1A_0 + D_2A_1\overline{A}_0 + D_3A_1A_0 = \sum_{i=0}^{3} D_i m_i$$

逻辑图



集成双4选1数据选择器74LS153



输入				输出
S	D	A_1	A_0	Y
1	×	×	×	0
0	D_0	0	0	D_0
0	D_1	0	1	D_1
0	D_2	1	0	D_2
0	D_3	1	1	D_3

选通控制端S为低电平有效，即$\overline{S}$=0时芯片被选中，处于工作状态；$\overline{S}$=1时芯片被禁止，Y≡0。

存储器

存储器的分类：

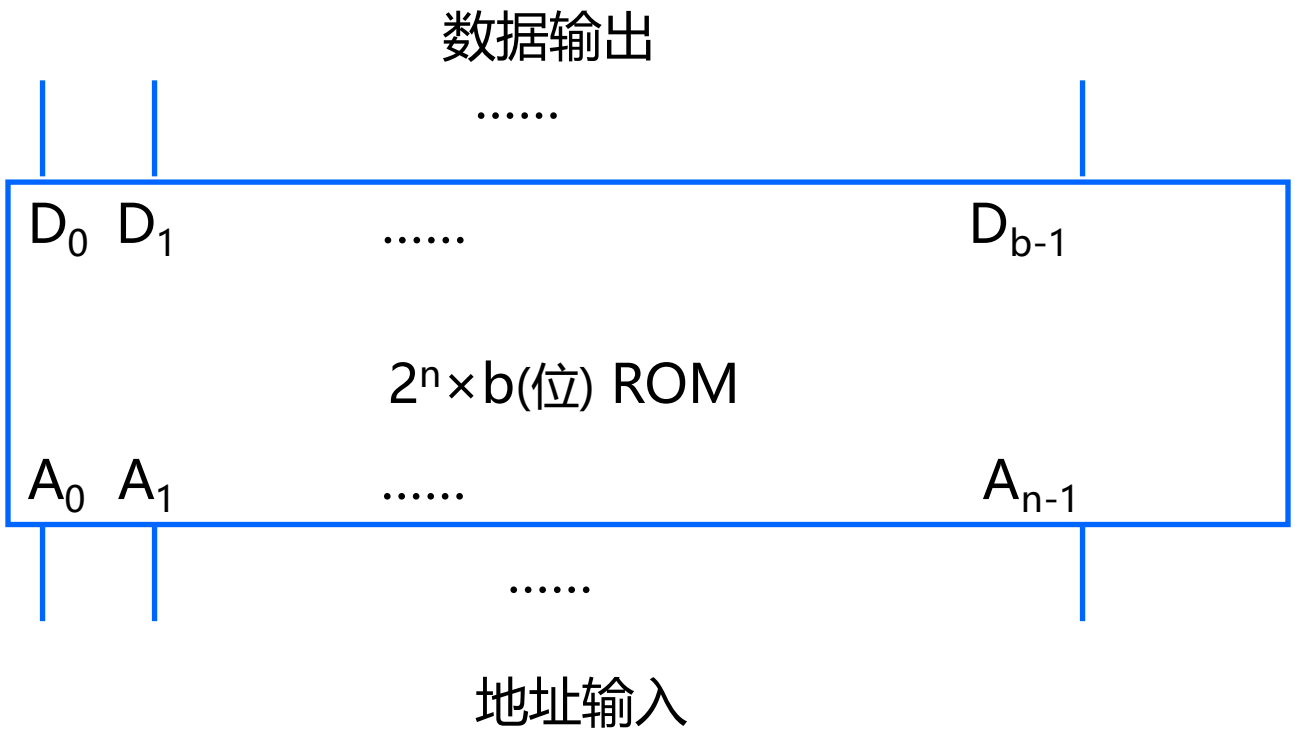
Random Access Memory

- RAM：在工作时既能从中读出（取出）信息，又能随时写入（存入）信息，但断电后所存信息消失。

Read Only Memory

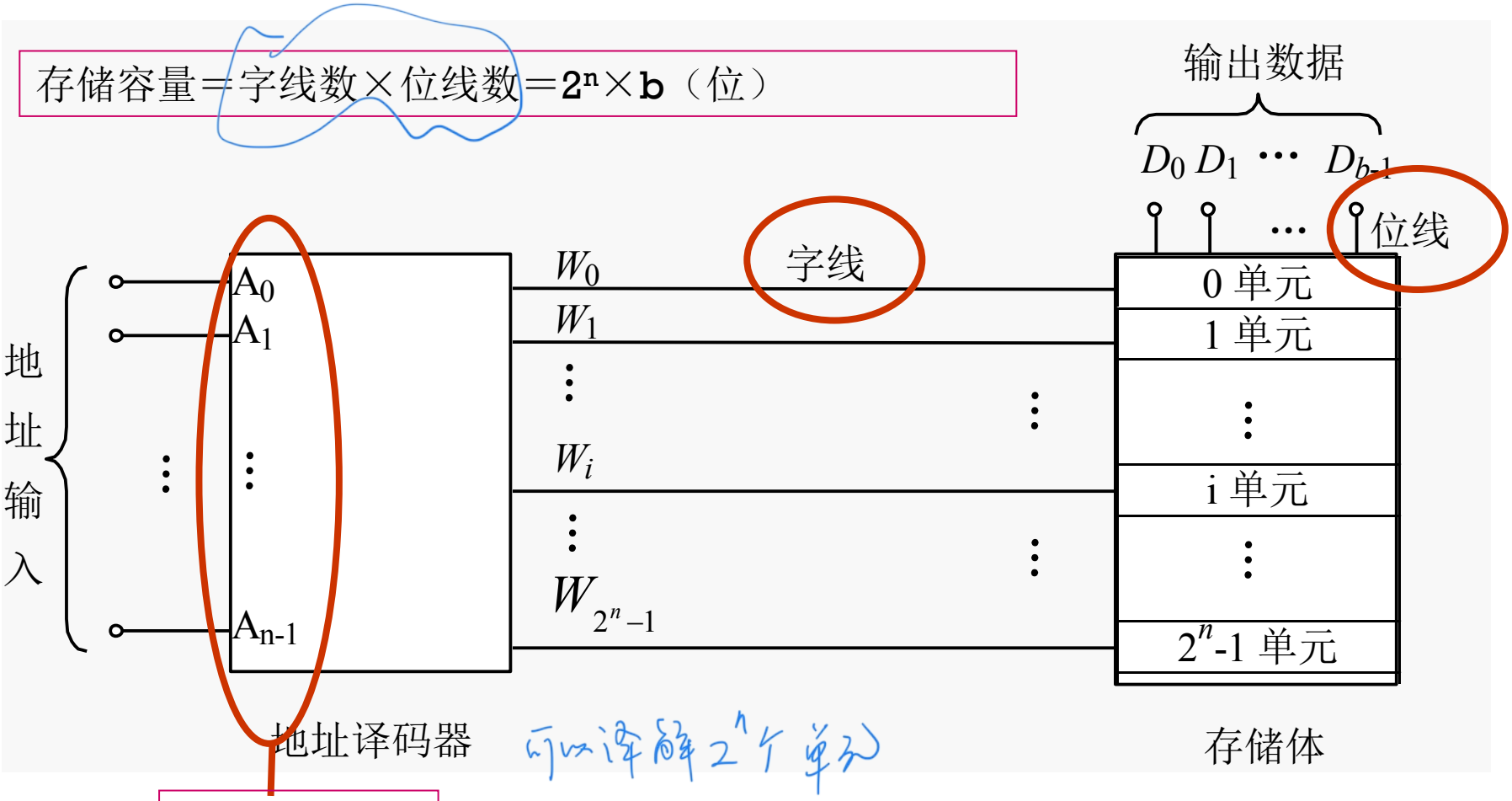
- ROM：在工作时只能从中读出信息，不能写入信息，且断电后其所存信息仍能保持。
  - 掩膜ROM：不能改写。
  - PROM：只能改写一次。
  - EPROM：可以改写多次。

ROM的基本结构



ROM内部结构示意图

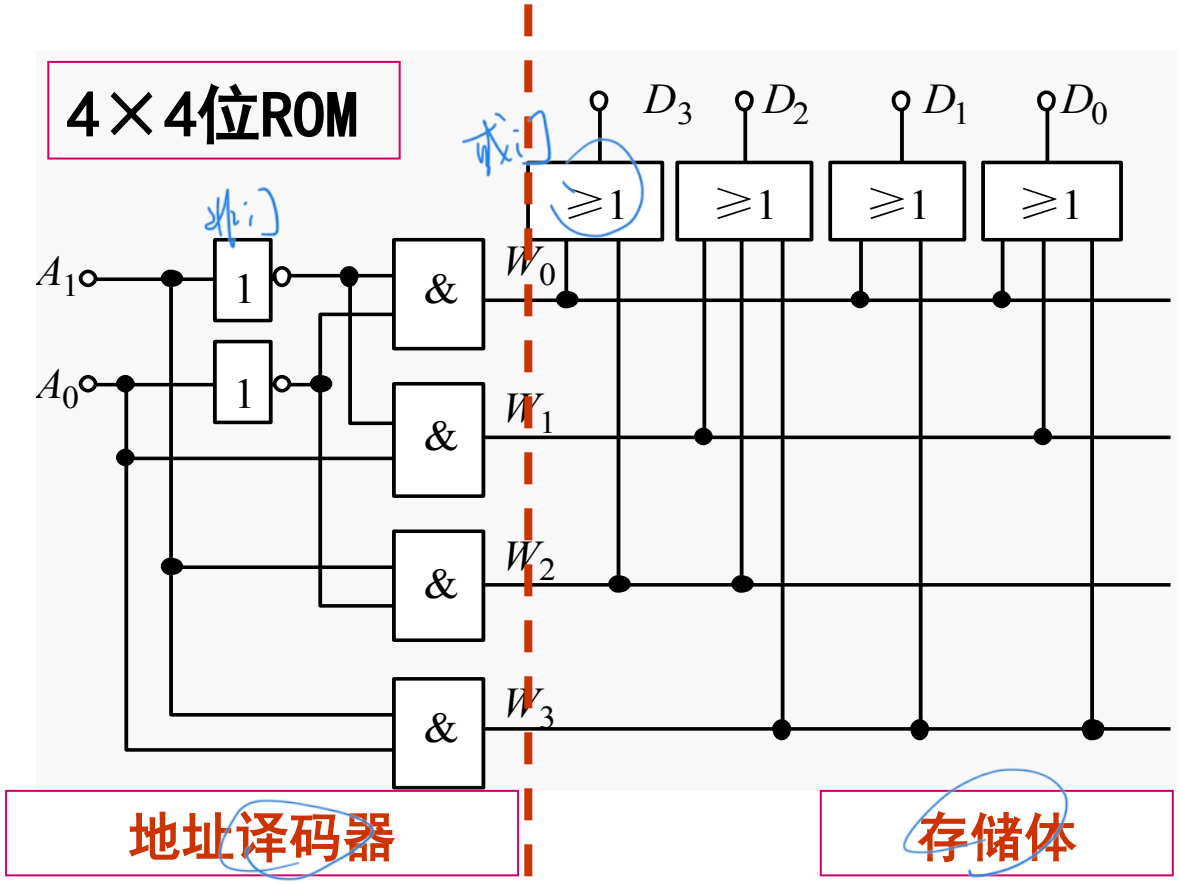
存储容量＝字线数×位线数＝$2^n \times b$（位）



可以译解2^n个单元

存储单元地址

ROM的基本工作原理



$$W_0 = m_0 = \overline{A}_1\overline{A}_0$$

$$W_1 = m_1 = \overline{A}_1 A_0$$

$$W_2 = m_2 = A_1\overline{A}_0$$

$$W_3 = m_3 = A_1 A_0$$

$$D_3 = W_0 + W_2 = m_0 + m_2$$

$$D_2 = W_1 + W_2 + W_3 = m_1 + m_2 + m_3$$

$$D_1 = W_0 + W_3 = m_0 + m_3$$

$$D_0 = W_0 + W_1 + W_3 = m_0 + m_1 + m_3$$

存储内容

地址		字线				存储内容			
A_1	A_0	W_0	W_1	W_2	W_3	D_3	D_2	D_1	D_0
0	0	1	0	0	0	1	0	1	1
0	1	0	1	0	0	0	1	0	1
1	0	0	0	1	0	1	1	0	0
1	1	0	0	0	1	0	1	1	1

$$D_3 = W_0 + W_2 = m_0 + m_2 \qquad D_2 = W_1 + W_2 + W_3 = m_1 + m_2 + m_3$$

$$D_1 = W_0 + W_3 = m_0 + m_3 \qquad D_0 = W_0 + W_1 + W_3 = m_0 + m_1 + m_3$$

对于给定的地址，相应一条字线输出高电平，与该字线相连接的或门输出为1，未连接的或门输出为0。

ROM应用举例：用ROM作函数运算表

例：用ROM构成能实现函数$y = x^2$的运算表电路。

设x的取值范围为0 ~ 15的正整数，则对应的是4位二进制正整数，用$B = B_3B_2B_1B_0$表示。

根据$y = x^2$可算出y的最大值是$15^2 = 225$，可以用8位二进制数 $Y=Y_7Y_6Y_5Y_4Y_3Y_2Y_1Y_0$ 表示。

由此可列出$Y = B^2$ 即$y = x^2$的真值表。

逻辑表达式

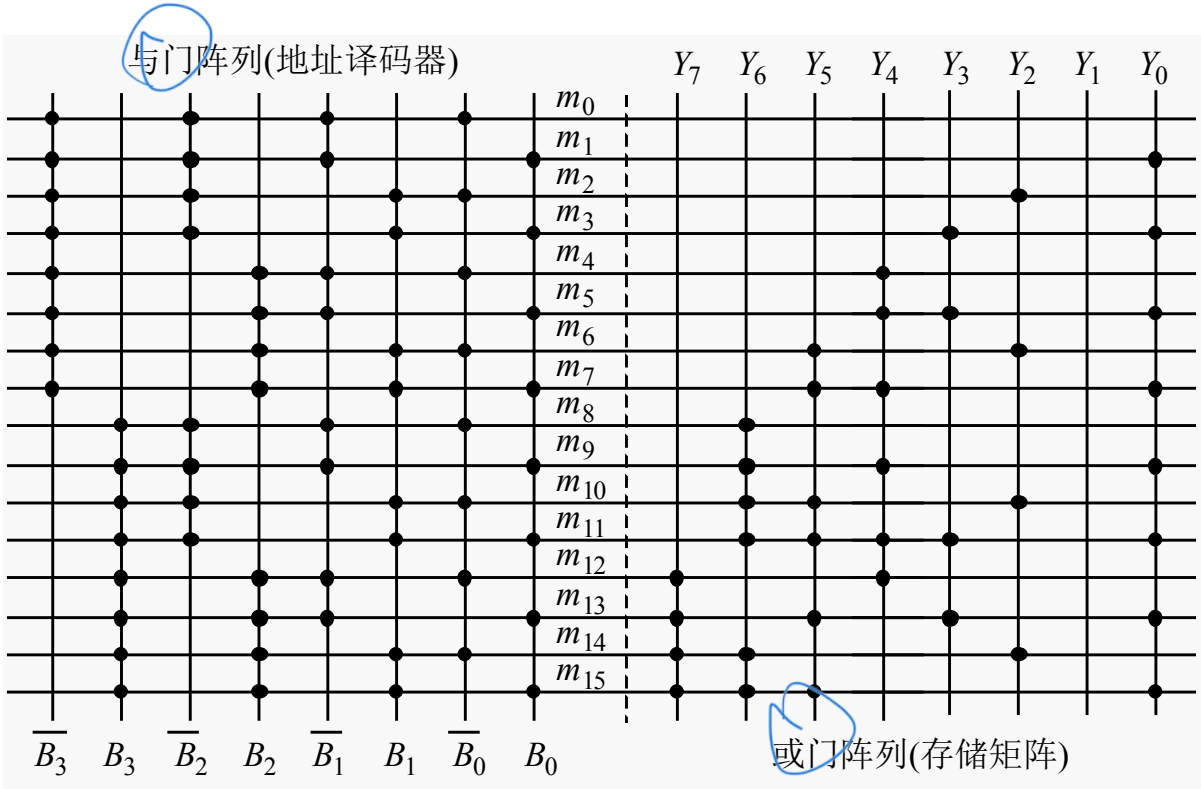
$$
\begin{cases}
Y_7 = \sum m(12,13,14,15) \\
Y_6 = \sum m(8,9,10,11,14,15) \\
Y_5 = \sum m(6,7,10,11,13,15) \\
Y_4 = \sum m(4,5,7,9,11,12) \\
Y_3 = \sum m(3,5,11,13) \\
Y_2 = \sum m(2,6,10,14) \\
Y_1 = 0 \\
Y_0 = \sum m(1,3,5,7,9,11,13,15)
\end{cases}
$$

真值表

输入				输出								注
B_3	B_2	B_1	B_0	Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	Y_0	十进制数
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0	0	1	1
0	0	1	0	0	0	0	0	0	1	0	0	4
0	0	1	1	0	0	0	0	1	0	0	1	9
0	1	0	0	0	0	0	1	0	0	0	0	16
0	1	0	1	0	0	0	1	1	0	0	1	25
0	1	1	0	0	0	1	0	0	1	0	0	36
0	1	1	1	0	0	1	1	0	0	0	1	49
1	0	0	0	0	1	0	0	0	0	0	0	64
1	0	0	1	0	1	0	1	0	0	0	1	81
1	0	1	0	0	1	1	0	0	1	0	0	100
1	0	1	1	0	1	1	1	1	0	0	1	121
1	1	0	0	1	0	0	1	0	0	0	0	144
1	1	0	1	1	0	1	0	1	0	0	1	169
1	1	1	0	1	1	0	0	0	1	0	0	196
1	1	1	1	1	1	1	0	0	0	0	1	225

阵列图

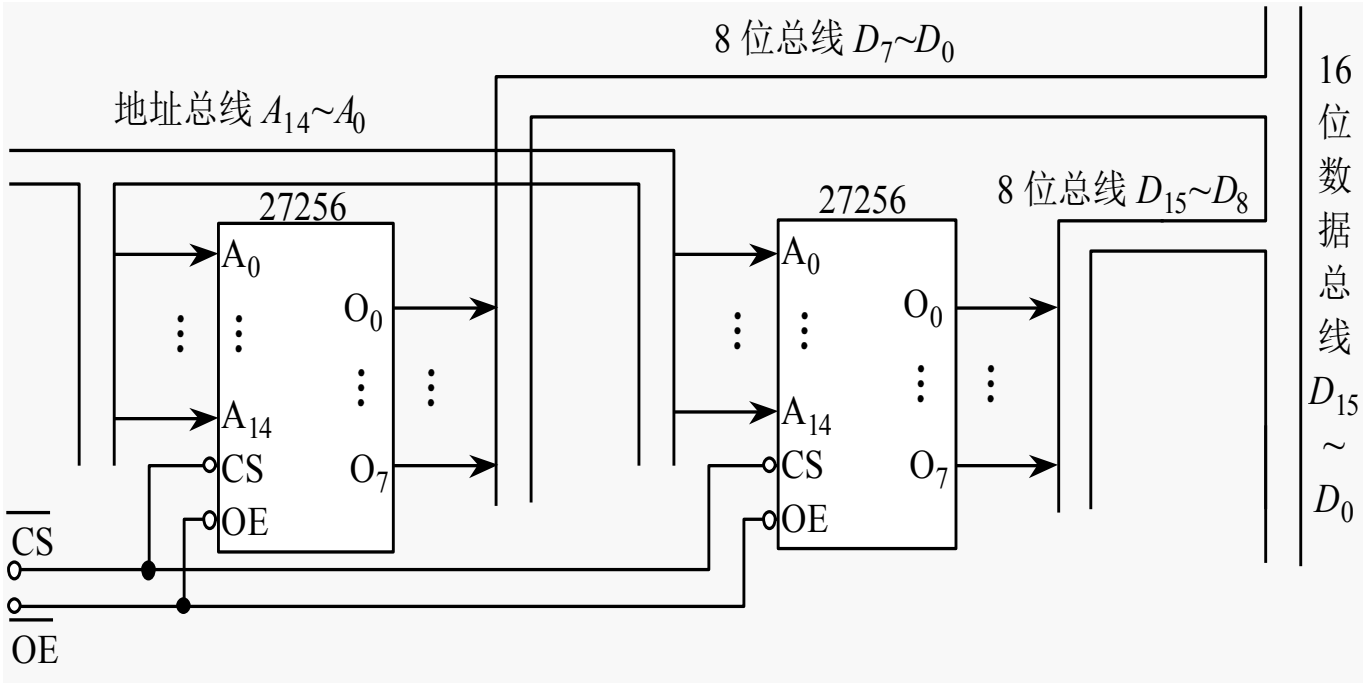
$$
\begin{cases}
Y_7 = \sum m(12,13,14,15) \\
Y_6 = \sum m(8,9,10,11,14,15) \\
Y_5 = \sum m(6,7,10,11,13,15) \\
Y_4 = \sum m(4,5,7,9,11,12) \\
Y_3 = \sum m(3,5,11,13) \\
Y_2 = \sum m(2,6,10,14) \\
Y_1 = 0 \\
Y_0 = \sum m(1,3,5,7,9,11,13,15)
\end{cases}
$$



ROM的容量扩展

1、位扩展（字长的扩展）

用两片27256（32k × 8）扩展成32k×16位EPROM



8+8
字长并联

地址信号，片选控制也要并联

地址线及控制线分别并联

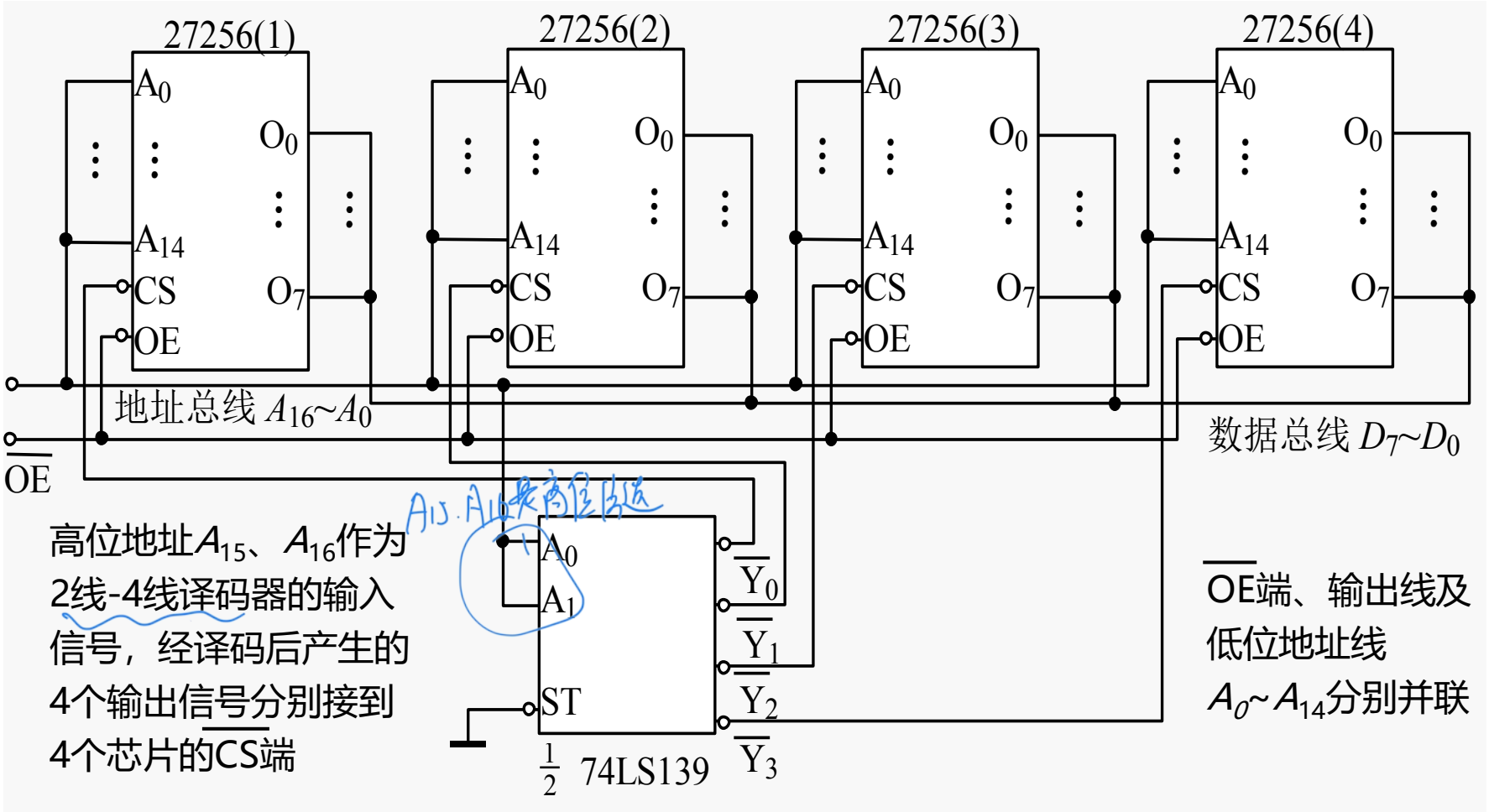
输出一个作为高8位，另一个作为低8位

任何时候只有一片输出数据，用高位做片选

2、字扩展（字数扩展，地址码扩展）

$2^2 \to 2$位片选

用4片27256扩展成4×32k×8位EPROM



可靠性代码

可靠性代码：代码传输时不易出错，或出现错误时容易被发现并进行校正，常见的有格雷码、奇偶校验码等。

1. 格雷码

格雷码：任何两个相邻代码之间只有一位码元不同，且0和最大数对应的格雷码之间也只有一位不同，首尾相连。

$$G_i = B_i \oplus B_{i+1}$$

典型4位格雷码　4位自然二进制码

	G3	G2	G1	G0	B3	B2	B1	B0
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	1	0	0	1	0
3	0	0	1	0	0	0	1	1
4	0	1	1	0	0	1	0	0
5	0	1	1	1	0	1	0	1
6	0	1	0	1	0	1	1	0
7	0	1	0	0	0	1	1	1
8	1	1	0	0	1	0	0	0
9	1	1	0	1	1	0	0	1
10	1	1	1	1	1	0	1	0
11	1	1	1	0	1	0	1	1
12	1	0	1	0	1	1	0	0
13	1	0	1	1	1	1	0	1
14	1	0	0	1	1	1	1	0
15	1	0	0	0	1	1	1	1

2. 奇偶校验码

◆ **奇偶校验码(Parity Check)**：一种校验代码传输正确性的方法，由两部分组成：

① 需要传输的信息本身，n位二进制代码。

② 1位奇偶校验位：增加1位0或1使二进制代码中1的个数为奇数或偶性。

- 奇校验：1的个数为奇数
- 偶校验：1的个数为偶数

特点：

① 只能发现奇数个数传输错误，没有错误定位和纠错能力，发现错误后，只能重传。

② 实现简单，广泛应用。

十进制数	8421 奇校验码		8421 偶校验码	
	信息码	校验位	信息码	校验位
0	0 0 0 0	1	0 0 0 0	0
1	0 0 0 1	0	0 0 0 1	1
2	0 0 1 0	0	0 0 1 0	1
3	0 0 1 1	1	0 0 1 1	0
4	0 1 0 0	0	0 1 0 0	1
5	0 1 0 1	1	0 1 0 1	0
6	0 1 1 0	1	0 1 1 0	0
7	0 1 1 1	0	0 1 1 1	1
8	1 0 0 0	0	1 0 0 0	1
9	1 0 0 1	1	1 0 0 1	0

改进的奇偶校验码

双向偶校验：

垂直水平两个维度校验

0110100	1
1011010	0
0010110	1
1110101	1
1001011	0
1000110	1

0110100	1
1011010	0
0010111	1
1110101	1
1001011	0
1000110	1

(可纠错)

可纠正1位错误

0110100	1
1011010	0
0110110	1
1110101	1
1001001	0
1000110	1

(可检错)

可检测出某行(列)上的奇数位

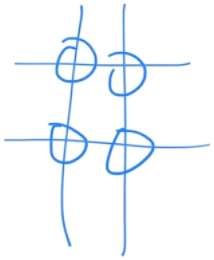
0110100	1
1011010	0
0110100	1
1110101	1
1001011	0
1000110	1

(可检错)

可检出一部分偶数位错误

1110101	1
1011010	0
0010110	1
1110101	1
1001011	0
0000111	1

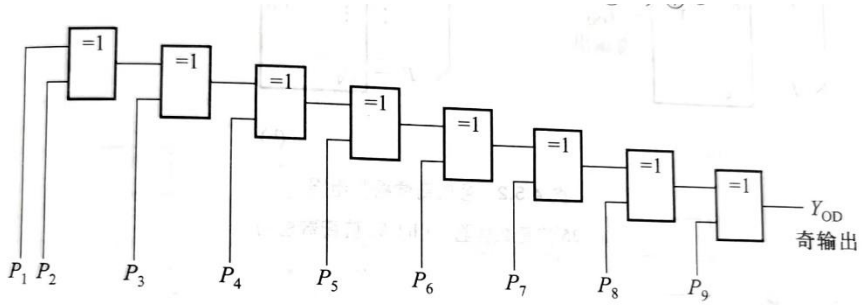
(不能检错)

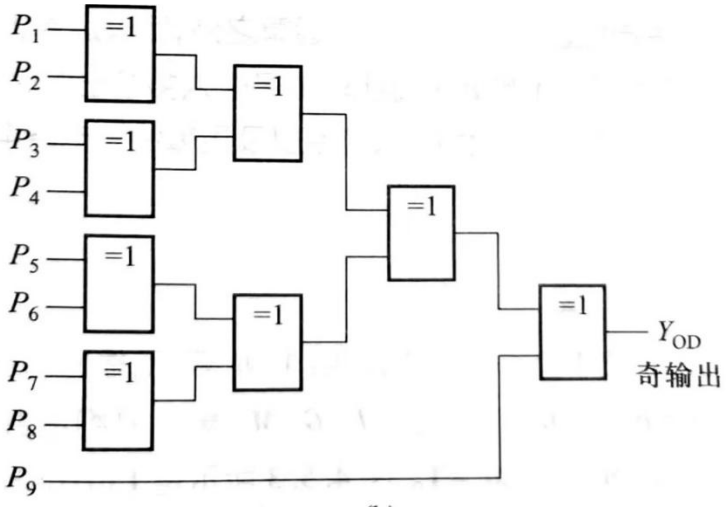


不能检测出错码分布在矩形4个顶点上的错误

奇偶检验电路

$$Y_{OD} = P_1 \oplus P_2 \oplus P_3 \oplus P_4 \oplus P_5 \oplus P_6 \oplus P_7 \oplus P_8 \oplus P_9$$

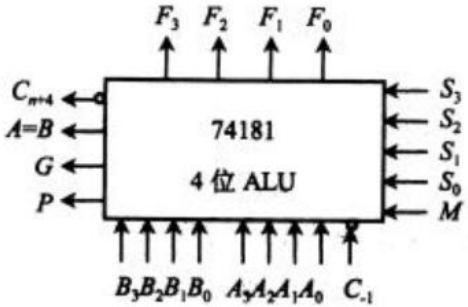




异或运算满足交换律和结合律，两种电路功能上等价

算术逻辑单元（ALU）

- 算术逻辑单元：简称ALU，是一种功能较强的组合逻辑电路，能进行多种算术运算和逻辑运算。
- ALU的一个基本逻辑结构是超前进位加法器。



SN74181型四位ALU功能表中

- 能执行16种算术运算和16种逻辑运算；
- M：状态控制端，当M=H，执行逻辑运算；M=L执行算术运算；
- $S_0 \sim S_3$：运算选择控制端，它决定电路执行哪种算术运算或哪种逻辑运算；
- $A_3 \sim A_0$，$B_3 \sim B_0$：参加运算的两个数；

SN74181型ALU功能表

功能表				
工作方式选择输入 $S_3S_2S_1S_0$	负逻辑输入或输出		正逻辑输入或输出	
	逻辑运算 (M=1)	算术运算 (M=0)（C_{-1}=0）	逻辑运算 (M=1)	算术运算 (M=0)（C_{-1}=1）
0000	$\overline{A}$	A 减 1	$\overline{A}$	A
0001	$\overline{AB}$	AB 减 1	$\overline{A+B}$	$A+B$
0010	$\overline{A}+B$	$A\overline{B}$ 减 1	$\overline{A}B$	$A+\overline{B}$
0011	逻辑 1	减 1	逻辑 0	减 1
0100	$\overline{A+B}$	A 加（$A+\overline{B}$）	$\overline{AB}$	A 加 $A\overline{B}$
0101	$\overline{B}$	AB 加（$A+\overline{B}$）	$\overline{B}$	($A+B$) 加 $A\overline{B}$
0110	$\overline{A\oplus B}$	A 减 B 减 1	$A\oplus B$	A 减 B 减 1
0111	$A+\overline{B}$	$A+\overline{B}$	$A\overline{B}$	$A\overline{B}$ 减 1
1000	$\overline{A}B$	A 加（$A+B$）	$\overline{A}+B$	A 加 AB
1001	$A\oplus B$	A 加 B	$\overline{A\oplus B}$	A 加 B
1010	B	$A\overline{B}$ 加（$A+B$）	B	（$A+\overline{B}$）加 AB
1011	$A+B$	$A+B$	AB	AB 减 1
1100	逻辑 0	A 加 A^*	逻辑 1	A 加 A^*
1101	$A\overline{B}$	AB 加 A	$A+\overline{B}$	（$A+B$）加 A
1110	AB	$A\overline{B}$ 加 A	$A+B$	（$A+\overline{B}$）加 A
1111	A	A	A	A 减 1

（1） 1=高电平；0=低电平；（2）*表示每一位均移到下一个更高位，即 $A^*=2A$

实验要求实现的ALU

指令名称	Op码[5:0]	指令功能	补充说明
ADD	100000	有符号数相加	使用超前进位加法器，注意根据结果置Over标志。
ADDU	100001	无符号数相加	使用超前进位加法器，Over标志恒为0。
SUB	100010	有符号数相减	使用超前进位加法器，注意根据结果置Over标志。
SUBU	100011	无符号数相减	使用超前进位加法器，Over标志恒为0。
SLL	000000	逻辑左移	A[4:0]为移位位数，在低位补零，Over标志恒为0。
SRL	000010	逻辑右移	A[4:0]为移位位数，在高位补零，Over标志恒为0。
SRA	000011	算术右移	A[4:0]为移位位数，Over标志恒为0。 若B操作数最高位为0，则高位补零，否则高位补1。
AND	100100	位与	Over标志恒为0。
OR	100101	位或	Over标志恒为0。
XOR	100110	位异或	Over标志恒为0。
NOR	100111	位或非	Over标志恒为0。

2.2.2 时序逻辑电路

时序电路: 在任何时刻的稳定输出，不仅与该时刻的输入信号有关，而且还与电路原来的状态有关。

触发器概述

在数字电路中，基本的工作信号只有0和1两种取值，作为存放二值逻辑信号的单元电路——**触发器**，对它的基本要求，即触发器的基本特征是：

- 有两个稳定的状态：0状态和1状态；
- 在不同的输入情况下，它可以被置成0状态或1状态；
- 当输入信号消失后，所置成的状态能够保持不变。

触发器的现态和次态：触发器接收输入信号之前的状态叫做现态，用Q^n表示。触发器接收输入信号之后的状态叫次态，用Q^{n+1}表示。现态和次态是两个相邻离散时间里触发器输出端的状态。

触发器的类型

- 根据逻辑功能不同分为

  *RS*触发器　　*D*触发器　　*JK*触发器　　*T*触发器　　*T′* 触发器

- 根据触发方式不同分为

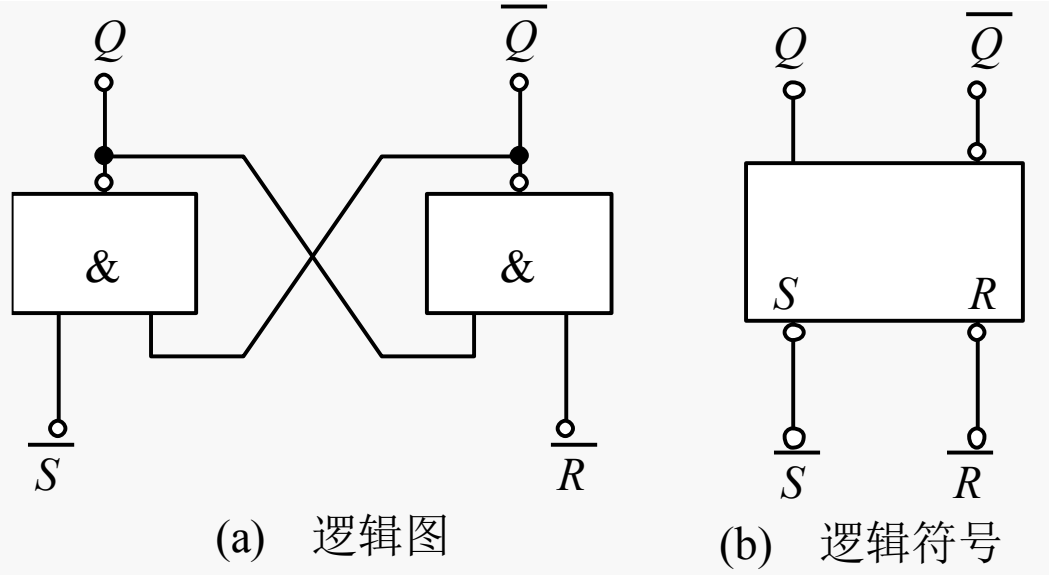
  电平触发器　　边沿触发器

- 根据电路结构不同分为

  基本 *RS*触发器　　同步触发器　　边沿触发器

用与非门组成的基本RS触发器

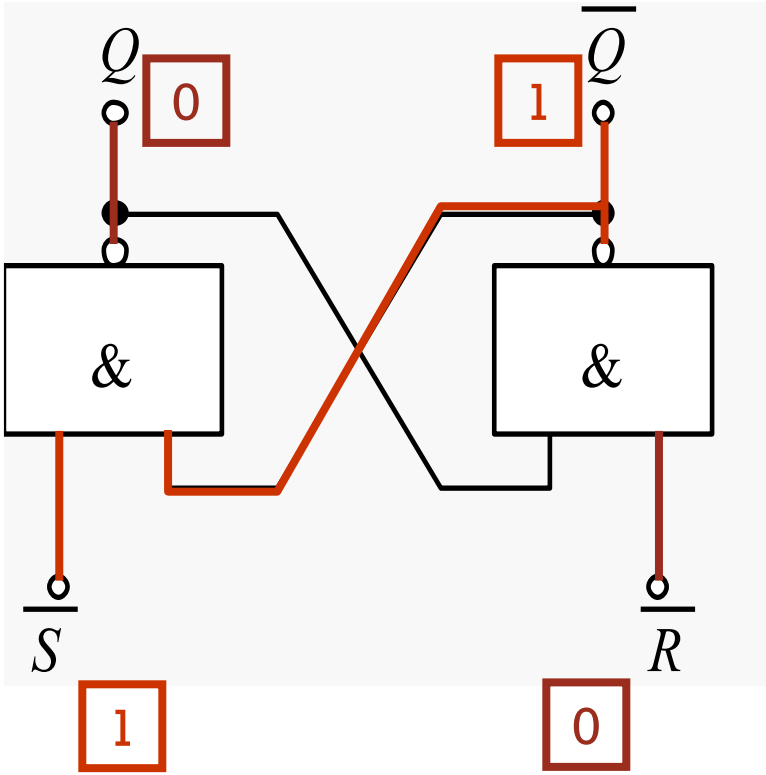
电路组成和逻辑符号



信号输入端，低电平有效。

信号输出端，Q=0、 $\bar{Q}$ =1的状态称0状态，Q=1、 $\bar{Q}$ =0的状态称1状态。

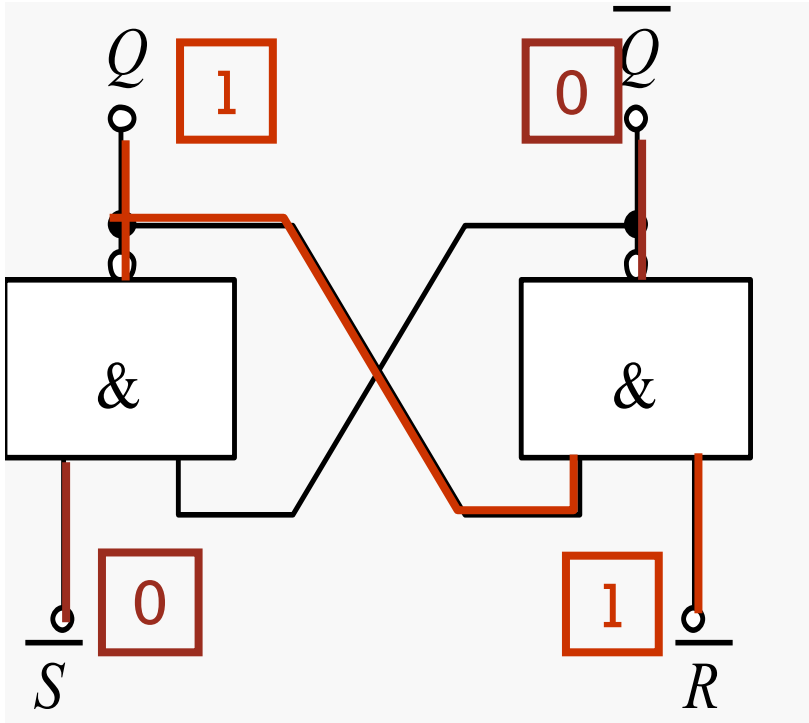
工作原理



$\overline{R}$	$\overline{S}$	Q
0	1	0

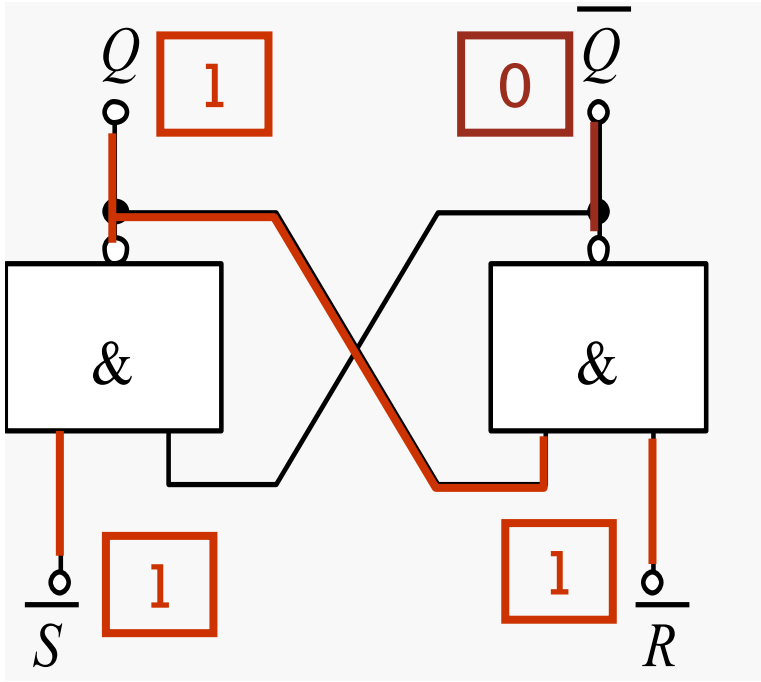
① $\overline{R}=0$、$\overline{S}=1$时：由于$\overline{R}=0$，不论原来Q为0还是1，都有$\overline{Q}=1$；再由$\overline{S}=1$、$\overline{Q}=1$可得$Q=0$。即不论触发器原来处于什么状态都将变成0状态，这种情况称将触发器置0或复位。 $\overline{R}$端称为触发器的置0端或复位端(即Reset)。

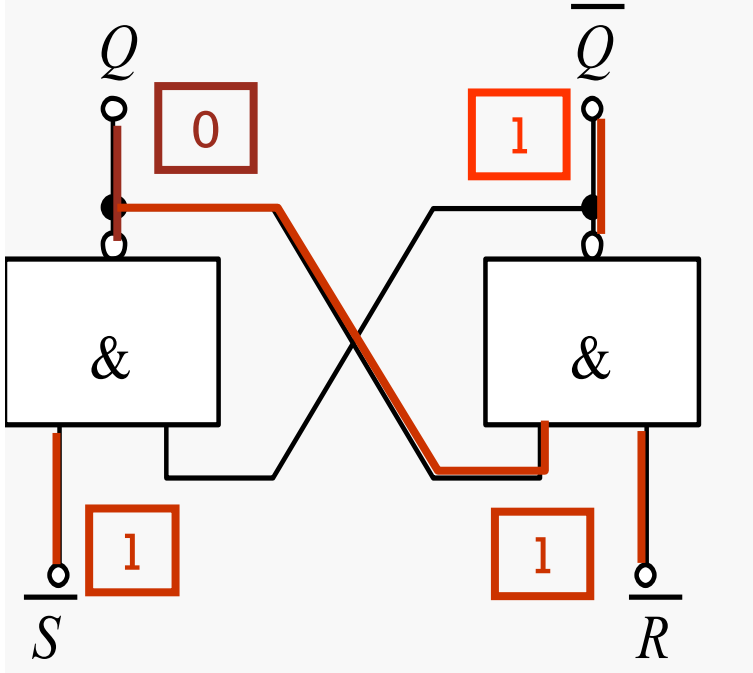
输出稳定



$\overline{R}$	$\overline{S}$	Q
0	1	0
1	0	1

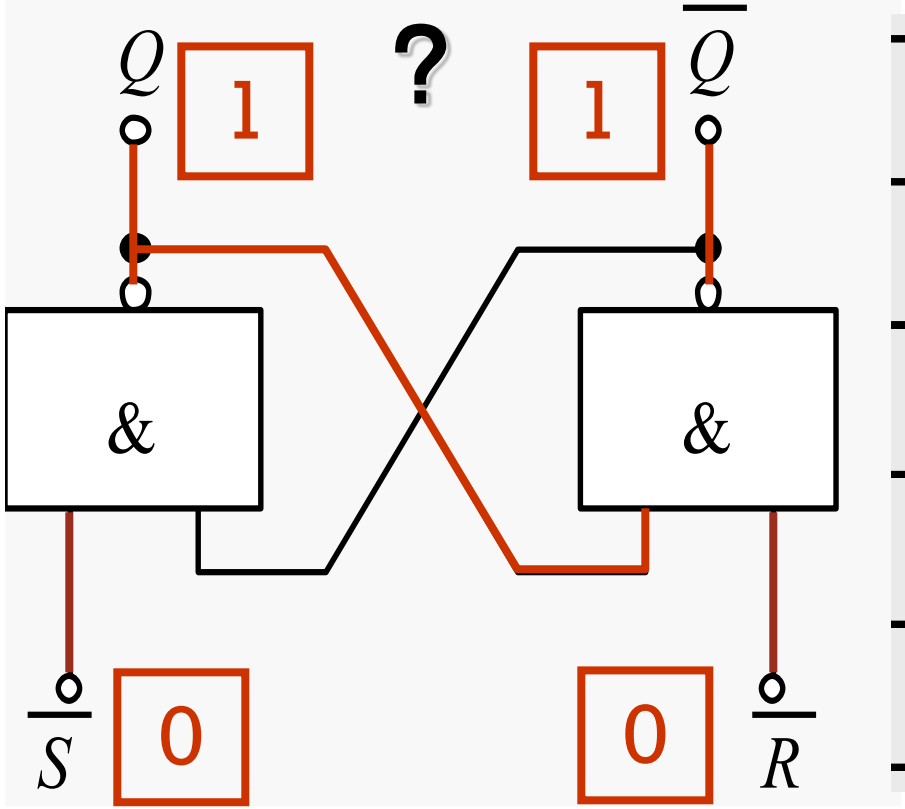
② $\overline{R}$ =1、 $\overline{S}$ =0时：由于$\overline{S}$ =0，不论原来Q为0还是1，都有Q=1；再由$\overline{R}$ =1、Q=1可得$\overline{Q}$ =0。即不论触发器原来处于什么状态都将变成1状态，这种情况称将触发器置1或置位。 $\overline{S}$端称为触发器的置1端或置位端(即Set)。


Q
1
0
$\overline{Q}$
&
&
1
1
$\overline{S}$
$\overline{R}$



$\overline{R}$	$\overline{S}$	Q
0	1	0
1	0	1
1	1	不变

③ $\overline{R}=1$、 $\overline{S}=1$时：根据与非门的逻辑功能不难推知，触发器保持原有状态不变，即原来的状态被触发器存储起来，这体现了触发器具有记忆能力。



$\overline{R}$	$\overline{S}$	Q
0	1	0
1	0	1
1	1	不变
0	0	不允许

④ $\overline{R}$ =0、$\overline{S}$=0时： $\overline{Q}$ =Q=1，不符合触发器的逻辑关系。并且由于与非门延迟时间不可能完全相等，在两输入端的0同时撤除后，将不能确定触发器是处于1状态还是0状态。所以触发器不允许出现这种情况，这就是基本*RS*触发器的约束条件。

特性表（真值表）

现态：触发器接收输入信号之前的状态，也就是触发器原来的稳定状态。

$\overline{R}$	$\overline{S}$	Q^n	Q^{n+1}	功能
0	0	0	不用	不允许
0	0	1	不用	
0	1	0	0	$Q^{n+1}=0$
0	1	1	0	置 0
1	0	0	1	$Q^{n+1}=1$
1	0	1	1	置 1
1	1	0	0	$Q^{n+1}=Q^n$
1	1	1	1	保持

次态：触发器接收输入信号之后所处的新的稳定状态。

$\bar{R}\bar{S}$

01 置0

10 置1

11 保持

次态Q^{n+1}的卡诺图

Q^n \ $\bar{R}\bar{S}$	00	01	11	10
0	×	0	0	1
1	×	0	1	1

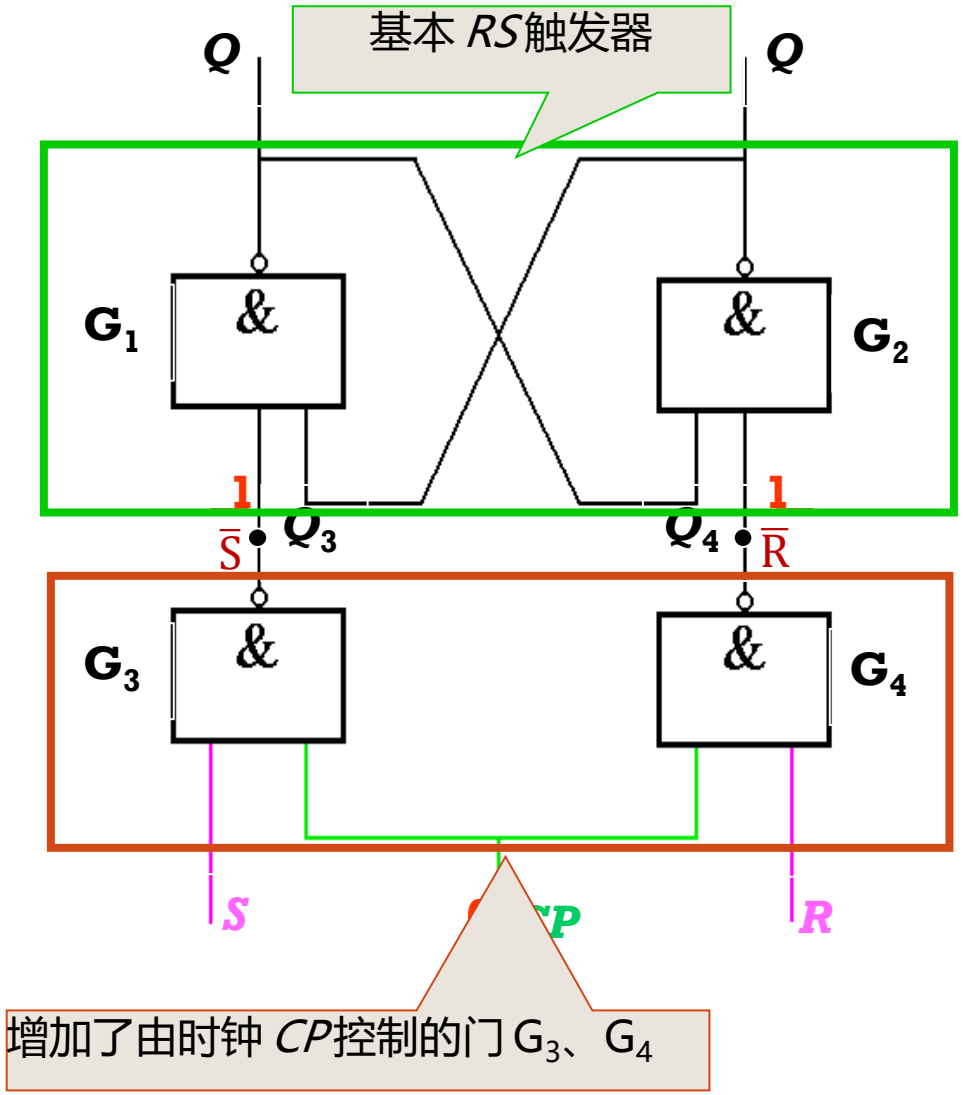
特性方程

$$\begin{cases} Q^{n+1} = \overline{(\bar{S})} + \bar{R}Q^n = S + \bar{R}Q^n \\ \bar{\bar{R}}\bar{\bar{S}} = 0 \qquad (RS = 0) \qquad \text{约束条件} \end{cases}$$

触发器的特性方程：触发器次态Q^{n+1}与输入及现态Q^n之间的逻辑关系式

同步RS触发器

1. 电路结构与工作原理



工作原理

- $CP = 0$ 时，G_3、G_4 被封锁，输入信号 R、S 不起作用。基本 RS 触发器的输入均为 1，触发器状态保持不变。

- $CP = 1$ 时，G_3、G_4 解除封锁，将输入信号 R 和 S 取非后送至基本 RS 触发器的输入端。

特性表

CP	R	S	Q^n	Q^{n+1}	功能
0	×	×	×	Q^n	$Q^{n+1}=Q^n$ 保持
1	0	0	0	0	$Q^{n+1}=Q^n$ 保持
1	0	0	1	1	
1	0	1	0	1	$Q^{n+1}=1$ 置 1
1	0	1	1	1	
1	1	0	0	0	$Q^{n+1}=0$ 置 0
1	1	0	1	0	
1	1	1	0	不用	不允许
1	1	1	1	不用	

00

01

10

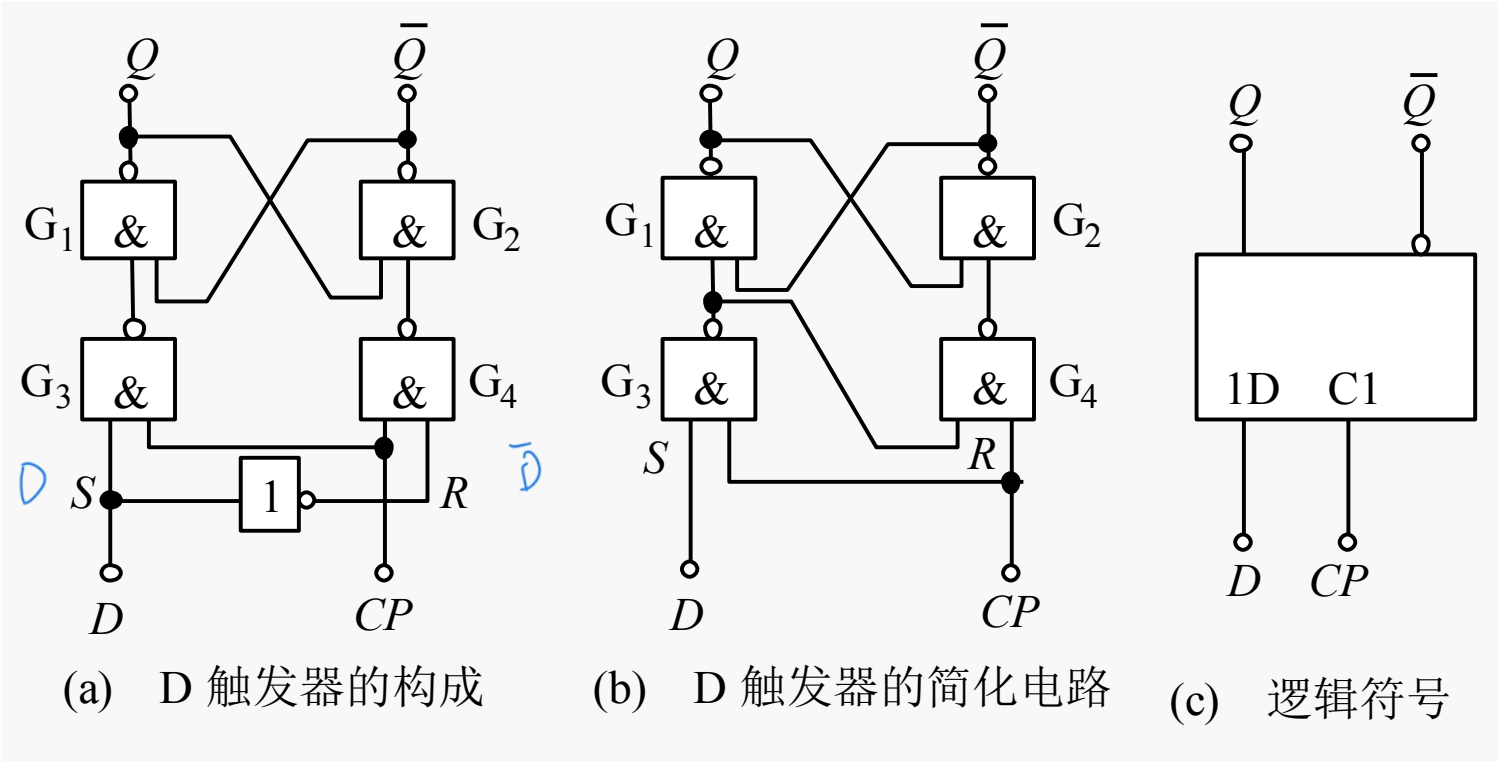
特性方程

$$\begin{cases} Q^{n+1}=S+\overline{R}Q^n \\ RS=0 \end{cases}$$ (RS不能同时为1)

CP=1期间有效

输入从 $\overline{R}\overline{S}$ → RS

同步D触发器



(a) D 触发器的构成　(b) D 触发器的简化电路　(c) 逻辑符号

将S=D、R= $\overline{D}$ 代入同步RS触发器的特性方程，得同步D触发器的特性方程：

$$Q^{n+1} = S + \overline{R}Q^n = D + \overline{\overline{D}}Q^n = D$$

CP=1期间有效

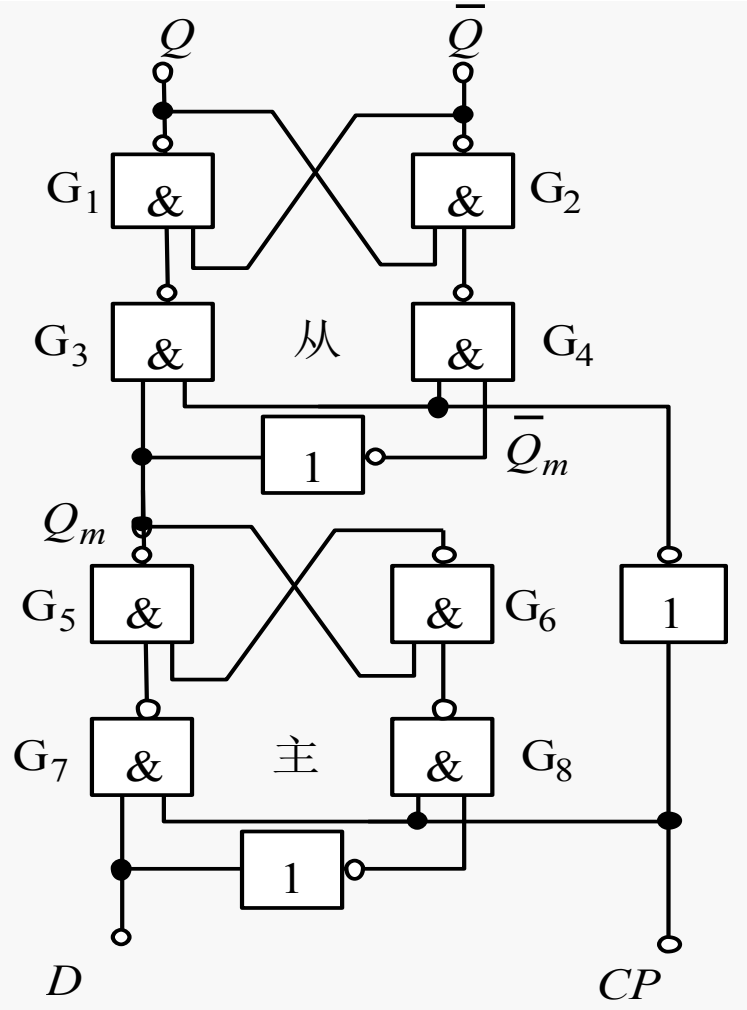
无有置0,1功能

无空翻触发器——边沿触发器

工作特点:

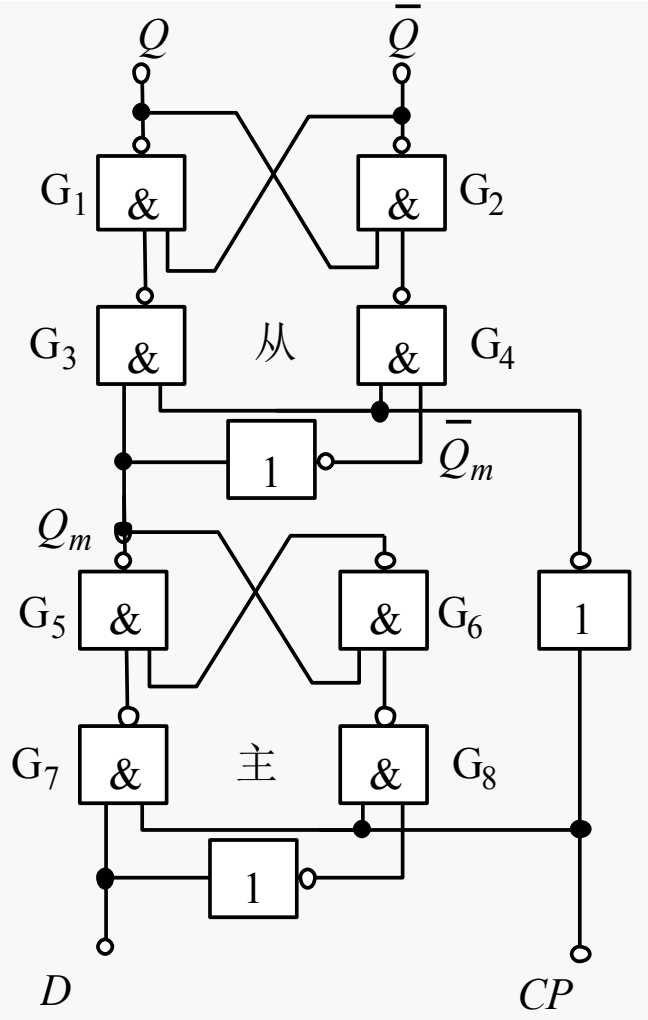
- 只能在 *CP* 上升沿(或下降沿)时刻接收输入信号
- 电路状态只能在 *CP* 上升沿(或下降沿)时刻翻转
- 克服了空翻，可靠性和抗干扰能力强，应用范围广

边沿D触发器



工作原理

- *CP*= 0时，门G_7、G_8被封锁，门G_3、G_4打开，从触发器的状态取决于主触发器:$Q=Q_m$、$\overline{Q} = \overline{Q_m}$，输入信号*D*不起作用。
- *CP*= 1时，门G_7、G_8打开，门G_3、G_4被封锁，从触发器状态不变，主触发器的状态跟随输入信号*D*的变化而变化，即在*CP*= 1期间始终都有Q_m=D。

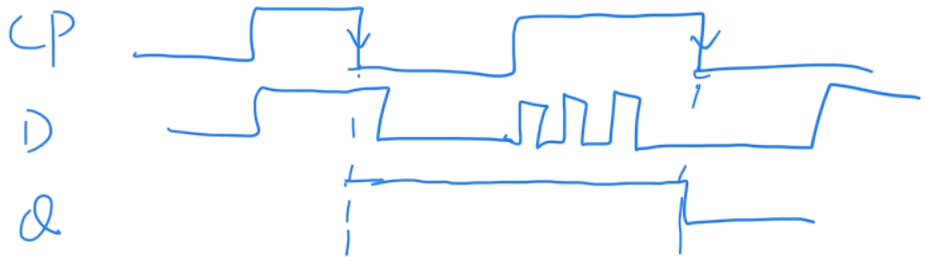


- *CP*下降沿到来时，封锁门G_7、G_8，打开门G_3、G_4，主触发器锁存*CP*下降时刻*D*的值，即Q_m=D，随后将该值送入从触发器，使Q=D、$\overline{Q}=\overline{D}$。
- *CP*下降沿过后，主触发器锁存的*CP*下降沿时刻*D*的值被保存下来，而从触发器的状态也将保持不变。

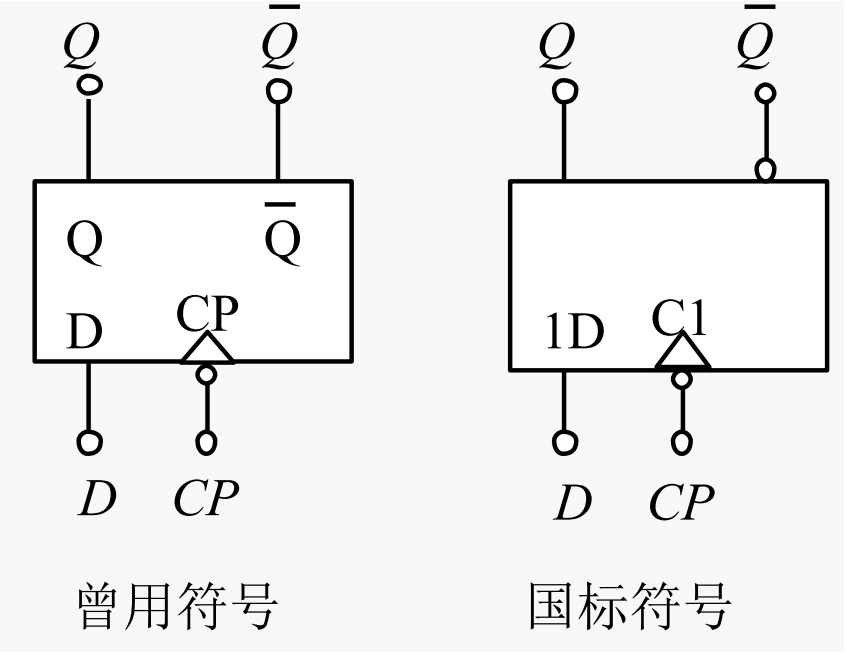
综上所述，边沿*D*触发器的特性方程为：

$$Q^{n+1}=D$$ 下降沿时刻有效

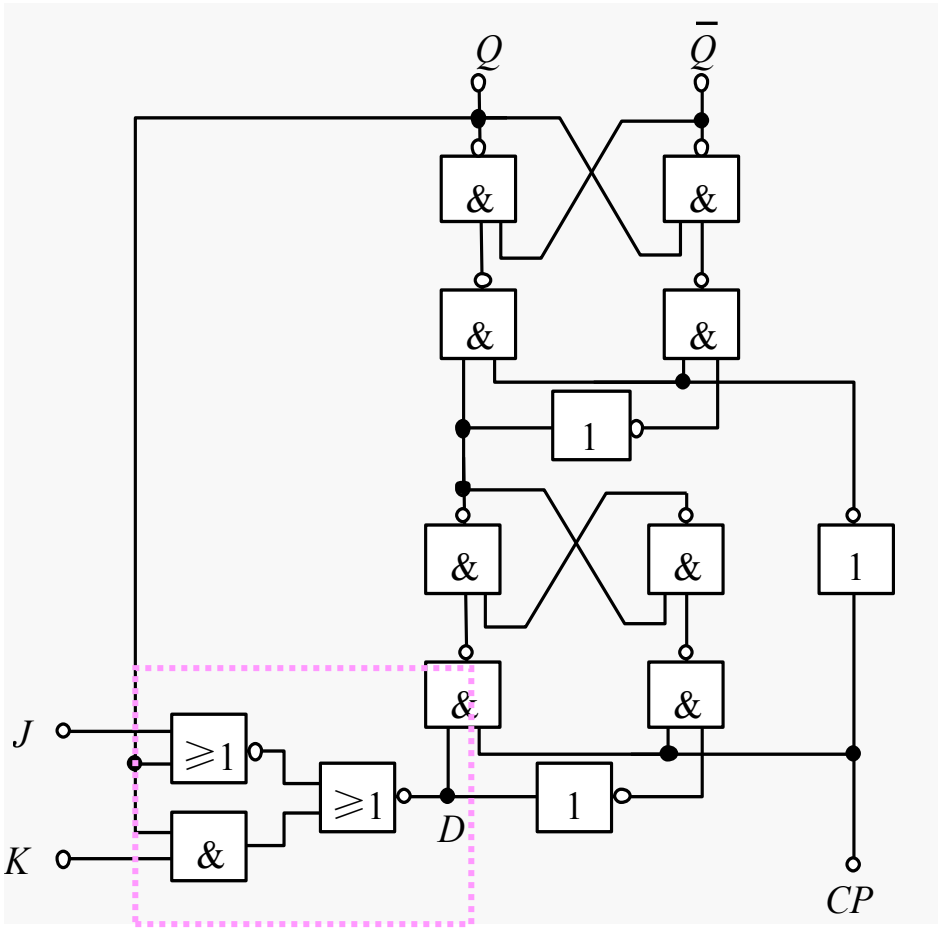
只看下降沿之前D是什么信号



逻辑符号


Q
$\overline{Q}$
Q
$\overline{Q}$
D
CP
D
CP
曾用符号
Q
$\overline{Q}$
1D
C1
D
CP
国标符号

边沿JK触发器



$$D = \overline{\overline{J + Q^n} + KQ^n}$$

$$= (J + Q^n) \cdot \overline{KQ^n}$$

$$= (J + Q^n)(\overline{K} + \overline{Q}^n)$$

$$= J\overline{Q}^n + \overline{K}Q^n + J\overline{K}$$

$$= J\overline{Q}^n + \overline{K}Q^n$$

$$Q^{n+1} = D$$

$$= J\overline{Q}^n + \overline{K}Q^n$$ *CP*下降沿时刻有效

在边沿D触发器的基础上，增加了三个门，并将Q反馈给输入，便构成了边沿JK触发器。

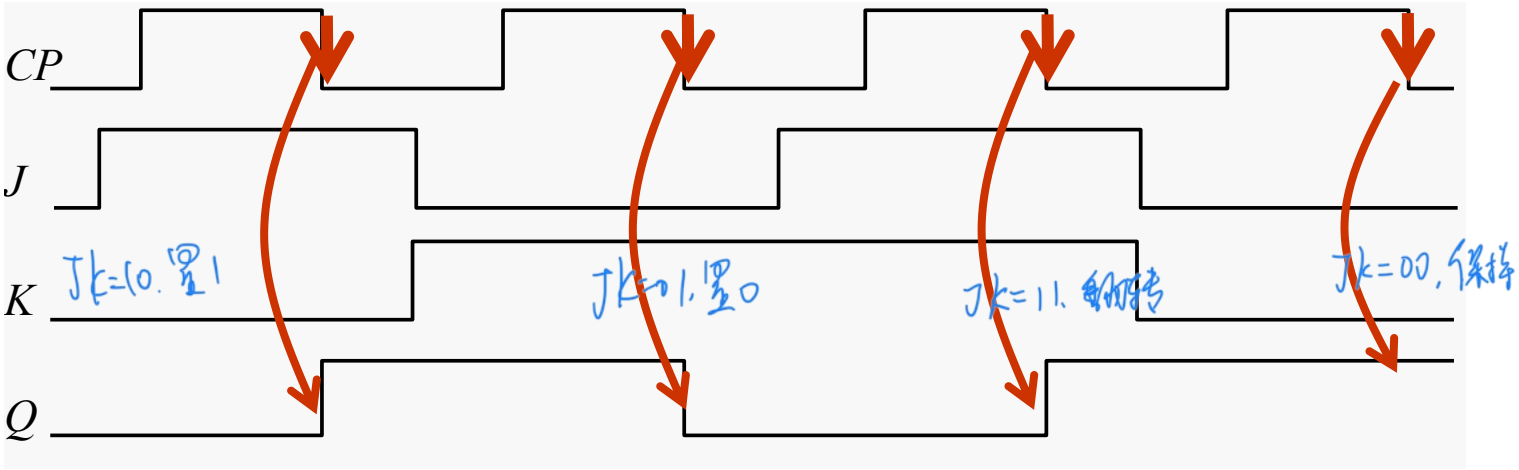
特性表

和RS是反的

$$Q^{n+1} = J\bar{Q}^n + \bar{K}Q^n$$

J	K	Q^n	Q^{n+1}	功能	
0	0	0	0	$Q^{n+1} = Q^n$ 保持	JK=00
0	0	1	1		
0	1	0	0	$Q^{n+1} = 0$ 置 0	JK=01
0	1	1	0		
1	0	0	1	$Q^{n+1} = 1$ 置 1	JK=10
1	0	1	1		
1	1	0	1	$Q^{n+1} = \overline{Q}^n$ 翻转	JK=11
1	1	1	0		

时序图



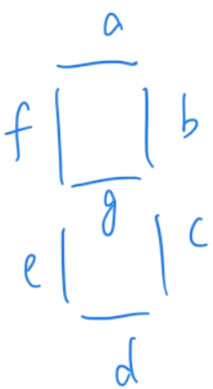
小结

- 触发器是数字电路的极其重要的基本单元。
- 触发器有两个稳定状态，在外界信号作用下，可以从一个稳态转变为另一个稳态；无外界信号作用时状态保持不变。
- 触发器可以作为二进制存储单元使用。
- 各种不同逻辑功能的触发器的特性方程为：

RS触发器：$Q^{n+1}=S+\overline{R}Q^{n}$，其约束条件为： $RS=0$

JK触发器： $Q^{n+1}=J\overline{Q^{n}}+\overline{K}Q^{n}$

D触发器： $Q^{n+1}=D$



输入：0~9　　输出：显示字型

找出 O_e 的逻辑表达式

	I_3	I_2	I_1	I_0	O_a	O_b	O_c	O_d	O_e	O_f	O_g
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	1	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	1	0	1	1
>9 (非法)					0	0	0	0	0	0	0

$$\bar{I}_0\bar{I}_1\bar{I}_2\bar{I}_3+\bar{I}_0I_1\bar{I}_2\bar{I}_3$$

$$+\bar{I}_0I_1I_2\bar{I}_3+\bar{I}_0\bar{I}_1\bar{I}_2I_3$$

$I_1I_0 \backslash I_3I_2$	00	01	11	10
00	1			1
01				
11				
10	1		1	

$\bar{I}_0\bar{I}_1\bar{I}_2$

$\bar{I}_0 I_1 \bar{I}_3$

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