# 数字电路与逻辑设计

## 第 2 章 逻辑代数与硬件描述语言基础

张江山
zhangjs@hust.edu.cn
信息工程系











1. 掌握逻辑代数常用基本定律、恒等式和规则
2. 掌握逻辑函数的基本表达式及相互转换，代数化简方法
3. 掌握逻辑函数最小项和最大项定义及性质，卡诺图化简法
4. 掌握硬件描述语言 Verilog HDL
5. 掌握正、负逻辑运算
6. 掌握逻辑符号的等效变换





- 逻辑代数：又称布尔代数
  - 是分析和设计现代数字逻辑电路的数学工具
  - 逻辑代数的定律、定理和规则，用于对表达式进行处理，以完成对逻辑电路的化简、变换、分析和设计
- 逻辑关系
  - 是事件产生的条件和结果之间的因果关系
  - 在数字电路中，将事情的条件作为输入信号，结果作为输出信号
  - 条件和结果中两种对立状态分别用逻辑 1 和 0 表示





| 基本定律 | 与 | 或 |
|---|---|---|
| 1. 0-1 律 | $A\cdot 0=0$，$A\cdot 1=A$ | $A+1=1$，$A+0=A$ |
| 2. 重叠律 | $A\cdot A=A$ | $A+A=A$ |
| 3. 互补律 | $A\cdot \overline{A}=0$ | $A+\overline{A}=1$ |
| 4. 结合律 | $A\cdot(B\cdot C)=(A\cdot B)\cdot C$ | $A+(B+C)=(A+B)+C$ |
| 5. 交换律 | $A\cdot B=B\cdot A$ | $A+B=B+A$ |
| 6. 分配律 | $A\cdot(B+C)=AB+AC$ | $A+BC=(A+B)(A+C)$ |
| 7. 反演律 | $\overline{AB}=\overline{A}+\overline{B}$ | $\overline{A+B}=\overline{A}\,\overline{B}$ |

摩根定理　　普通代数无此式





| 基本定律 | 与 | 或 |
|---|---|---|
| 8. 吸收律 | $A(A+B)=A$ | $A+AB=A$ |
| 吸收律 | $A(\overline{A}+B)=AB$（$\overline{A}$ 冗余） | $A+\overline{A}B=A+B$（$BC$ 冗余） |
| 常用恒等式 | $AB+\overline{A}C+BC=AB+\overline{A}C$ | |
| | $AB+\overline{A}C+BCD=AB+\overline{A}C$ | |

证明：$AB+\overline{A}C+BC=AB+\overline{A}C$

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

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

$=AB+\overline{A}C$

等式证明

①. 代数法

证明吸收律 $AB+A\overline{B}=A$

证：$AB+A\overline{B}=A(B+\overline{B})=A$

②. 穷举法

将等式两边分别用 $F_1$，$F_2$ 表示，

穷举输入变量所有取值的组合，

分别代入 $F_1$ 和 $F_2$ 算出相应的结果，

若都相等，则 $F_1=F_2$，否则，$F_1\neq F_2$

用真值表法证明摩根定理

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

令：$F_1=\overline{A+B}, F_2=\overline{A}\cdot\overline{B}$

| $A$ | $B$ | $F_1$ | $F_2$ |
|---|---|---|---|
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 |

得证





例：证明 $A+\overline{A}\cdot B=A+B$

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

例：试化简下列逻辑函数 $L=(A+B)(\overline{A}+B)$

$L=A\overline{A}+AB+B\overline{A}+BB$（分配率）

$=0+AB+B\overline{A}+B$ $(A\cdot\overline{A}=0, A\cdot A=A)$

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

$=B$ $(A+1=A, A\cdot 1=A)$





**1.** 代入规则

- 规则：任何含有某变量的等式，若等式中所有出现该变量的位置均代之以一个逻辑函数式，则此等式依然成立
- 作用：扩大基本公式的应用范围

例如，根据反演律 $\overline{A\cdot B}=\overline{A}+\overline{B}$（利用摩根定律；$BC$ 代替 $B$）

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

由此，摩根定律能推广到 n 个变量：

$$\overline{A_1\cdot A_2\cdot\ldots\cdot A_n}=\overline{A_1}+\overline{A_2}+\ldots+\overline{A_n}$$

$$\overline{A_1+A_2+\ldots+A_n}=\overline{A_1}\cdot\overline{A_2}\cdot\ldots\cdot\overline{A_n}$$





**2.** 反演规则

- 规则：对于任意一个逻辑函数式 $F$，做如下处理，得到的新函数式称为原函数式 $F$ 的反函数式
  - ◆若把式中的运算符 · 换成 +，+ 换成 ·
  - ◆常量 **0** 换成 **1**，**1** 换成 **0**
  - ◆原变量换成反变量，反变量换成原变量
  - ◆保持原函数的运算次序不变
- 作用：求原函数式 F 的反函数式

例 2.1.1 试求 $L=A\overline{B}+CD+0$ 的反函数

解：按照反演规则，得 $L=(\overline{A}+B)\cdot(\overline{C}+D)\cdot 1=(\overline{A}+B)(\overline{C}+D)$





注意事项

① 保持原函数的运算次序不变，必要时适当地加入括号

② 不属于单个变量上的非号有两种处理方法：

- ●非号保留，而非号下面的函数式按反演规则变换
- ●将非号去掉，而非号下的函数式保留不变

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

其反函数为 $\overline{F}=(\overline{A}+B)\cdot(A+C)\cdot\overline{B}+\overline{B}\cdot(\overline{A}+\overline{B}+\overline{C})$

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

（将 $(A+C)B$ 看成一个变量 $P$）





**3.** 对偶规则

- 对偶式：任意逻辑函数式 $F$，做如下处理，得到的新函数式称为原函数式 $F$ 的对偶式 $F'$，也称对偶函数
  - ◆若把式中的运算符 · 换成 +，+ 换成 ·
  - ◆常量 **0** 换成 **1**，**1** 换成 **0**
  - ◆保持原函数的运算次序不变
- 规则：若 $F_1=F_2$，则 $F_1'=F_2'$
- 作用：使定理公式的证明步骤减半

例：逻辑函数 $L=(A+B)(\overline{A}+C)$ 的对偶式为 $L=AB+\overline{A}C$

注意事项：保持原函数的运算次序不变，必要时适当地加入括号

1. 例如： $F=AB+\overline{AC}+1\cdot B$

其对偶式 $F'=(A+B)\cdot(\overline{A}+C)\cdot(0+B)$

2. 已知 $A+A\cdot B=A+B$ 求证 $A\cdot(A+B)=A\cdot B$

$F_1=A+A\cdot B$ $\quad F'_1=A\cdot(A+B)$

$F_2=A+B$ $\quad F'_2=A\cdot B$

$\because\ F_1=F_2$ $\quad \therefore\ F'_1=F'_2$





逻辑函数有不同形式

与 - 或式 $\overline{AC+\overline{C}D}$

与非 - 与非式 $\overline{\overline{AC}\cdot\overline{\overline{C}D}}$ （摩根定理可证与上式等价）

或 - 与式 $(A+C)(\overline{C}+D)$

$=AC+AD+\overline{C}D=AC+\overline{C}D+AD(C+\overline{C})$

$=AC+\overline{C}D+ADC+A\overline{C}D=AC+\overline{C}D$

或非 - 或非式 $\overline{(A+C)+(\overline{C}+D)}$

与 - 或 - 非式 $\overline{AC}+\overline{C}D$

任何表达式都可变换为上述五种形式的表达式

最简与或表达式：与项数最少，且各与项中变量数最少的与 - 或表达式





1. 最小项的定义和性质

●最小项： $n$ 个变量 $X_1, X_2, \ldots, X_n$ 的最小项，是 $n$ 个因子的乘积，各变量必须都以其原或反变量的形式在乘积项中出现且仅出现一次

●$n$ 个变量的最小项应有 $2^n$ 个

◆一个变量 $A$ 有二个 ( $2^1$ ) 最小项： $\overline{A}, A$

◆二个变量 $A$ , $B$ 有四个 ( $2^2$ ) 最小项： $\overline{A}\overline{B}, \overline{A}B, A\overline{B}, AB$

◆三个变量 $A$ , $B$ , $C$ 有八个 ( $2^3$ ) 最小项：

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

对于三个变量来说， $\overline{A}B, A\overline{B}\overline{C}A, A(B+C)$ 是不是最小项？

当然不是





2. 最小项的性质

性质 1 ：任意一个最小项，只有一组变量取值使得它的值为 1

性质 2 ：不同的最小项，使得它的值为 1 的那一组变量取值也不同

性质 3 ： $m_i \cdot m_j = 0$ ( $i \neq j$ ) 性质 4 ：全部最小项之和为 1

三变量的最小项

| $A\ B\ C$ | $m_0$ $\overline{A}\overline{B}\overline{C}$ | $m_1$ $\overline{A}\overline{B}C$ | $m_2$ $\overline{A}B\overline{C}$ | $m_3$ $\overline{A}BC$ | $m_4$ $A\overline{B}\overline{C}$ | $m_5$ $A\overline{B}C$ | $m_6$ $AB\overline{C}$ | $m_7$ $ABC$ | $F=\sum_{i=0}^{2^n-1} m_i$ |
|---|---|---|---|---|---|---|---|---|---|
| 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 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 1 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| 1 0 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 1 0 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 1 1 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| 1 1 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |





3. 最小项的编号

●可用 $m_i$ 表示最小项

●下标 $i$ 的取值规则：用 1 表示最小项中的原变量，用 0 表示反变量，由此得到一个二进制数，与该二进制数对应的十进制数即下标 $i$ 的值

| 最小项 | $\overline{A}\overline{B}\overline{C}$ | $\overline{A}\overline{B}C$ | $\overline{A}B\overline{C}$ | $\overline{A}BC$ | $A\overline{B}\overline{C}$ | $A\overline{B}C$ | $AB\overline{C}$ | $ABC$ |
|---|---|---|---|---|---|---|---|---|
| 二进制数 | 000 | 001 | 010 | 011 | 100 | 101 | 110 | 111 |
| 十进制数 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 表示方法 | $m_0$ | $m_1$ | $m_2$ | $m_3$ | $m_4$ | $m_5$ | $m_6$ | $m_7$ |





4. 最小项表达式

●由若干最小项构成的与 - 或表达式，也称为标准与 - 或式

◆为“与或”逻辑表达式

◆在“与或”式中的每个乘积项都是最小项

●任何逻辑函数都可变换成唯一的最小项表达式

●转换的方法有：① . 代数转换法， ② . 真值表转换法

例 1 ：将 $L(A,B,C)=AB+\overline{A}C$ 化成最小项表达式 缺啥补啥

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

$=\underset{111}{ABC}+\underset{110}{AB\overline{C}}+\underset{011}{\overline{A}BC}+\underset{101}{A\overline{B}C}$

$=m_7+m_6+m_3+m_5$

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

①. 代数转换法

第一步：将逻辑函数转换成一般与或表示式

第二步：反复使用 $X=X(Y+\bar{Y})$ ，将所有非最小项的与项变为最小项

例 2.2.1 $L(A,B,C)=(\overline{AB+A\bar{B}+C})\cdot\overline{AB}$

摩根定理

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

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

第一步 $=(\overline{AB+A\bar{B}+C})+AB=\overline{AB}\cdot\overline{A\bar{B}}\cdot\bar{C}+AB$

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

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

第二步 $=\bar{A}B\bar{C}+A\bar{B}\bar{C}+AB(C+\bar{C})$

$=\bar{A}B\bar{C}+A\bar{B}\bar{C}+AB\bar{C}+ABC$

$=m_3+m_5+m_6+m_7=\sum m(3,5,6,7)$





②. 真值表转换法——基本思想

最小项表达式是若干最小项构成的，若某组变量值使表达式 $F=1$ ，则 $F$ 中一定有该组变量值所对应的结果为 1 的最小项

例如，已知两变量逻辑函数 $F$ 真值表如下

| $A$ | $B$ | $F$ | $m_i$ |
|---|---|---|---|
| 0 | 0 | 0 | $m_0=1$ |
| 0 | 1 | 1 | $m_1=1$ |
| 1 | 0 | 1 | $m_2=1$ |
| 1 | 1 | 0 | $m_3=1$ |

2 变量共有 4 个最小项

$m_0=\bar{A}\bar{B}$ $m_1=\bar{A}B$

$m_2=A\bar{B}$ $m_3=AB$

$F=m_0?+m_1?+m_2?+m_3?$

$A$ , $B$ 取 0 , 0 时： $m_0=1$

而此时 $F=0$ ，故 $F$ 中不能包含 $m_0$

$A$ , $B$ 取 0 , 1 时： $m_1=1$

而此时 $F=1$ ，故 $F$ 中包含 $m_1$

…

$\therefore\ F=m_1+m_2=\bar{A}B+A\bar{B}$





在举重比赛中有三个裁判员，只有当两个或两个以上裁判员认为杠铃已经举起时，才算是成功，试写出逻辑表达式

输入： $A$ , $B$ , $C$ 0 表示认为失败， 1 表示认为成功

输出： $F$ 0 表示失败， 1 表示成功

列真值表

| $A$ | $B$ | $C$ | $F$ | |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | |
| 0 | 0 | 1 | 0 | |
| 0 | 1 | 0 | 0 | |
| 0 | 1 | 1 | 1 | $\bar{A}BC=m_3$ |
| 1 | 0 | 0 | 0 | |
| 1 | 0 | 1 | 1 | $A\bar{B}C=m_5$ |
| 1 | 1 | 0 | 1 | $AB\bar{C}=m_6$ |
| 1 | 1 | 1 | 1 | $ABC=m_7$ |

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





②. 真值表转换法——方法

第一步：将逻辑函数转换成一般与或表示式

第二步：列真值表，写出标准与或表示式

例 $F(A,B,C)=AB+B\bar{C}$ 已是一般与或式，不需要第一步

第二步：列真值表

| $A$ | $B$ | $C$ | $F$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 |

$A$ , $B$ 取 1 , 0 时： $F=1$

$B$ , $C$ 取 1 , 0 时： $F=1$

其余填 0

故 $F$ 中包含四个最小项： $m_2$ , $m_4$ , $m_5$ , $m_6$

$F=m_2+m_4+m_5+m_6=\sum m(2,4,5,6)$





1. 最大项的定义和性质

●$n$ 个变量 $X_1, X_2, \dots, X_n$ 的最大项，是 $n$ 个因子的或项，各变量都以原变量或非变量的形式在或项中出现且仅出现一次

●$n$ 个变量的最大项应有 $2^n$ 个

例如， $A$ , $B$ , $C$ 三个逻辑变量的最大项有 $2^3=8$ 个，即

$(A+B+C),(A+\bar{B}+C),(A+B+\bar{C}),(\bar{A}+B+C),$

$(A+\bar{B}+\bar{C}),(\bar{A}+B+\bar{C}),(\bar{A}+\bar{B}+C),(\bar{A}+\bar{B}+\bar{C})$





1. 最大项的定义和性质

●最大项的表示：通常用 $M_i$ 表示最大项，下标 $i$ 为最大项号

●最大项的性质：

◆对于一个最大项，只有一组变量的取值使其值为 0 ，该组值对应的十进制数是其下标编号（ 0 表示原变量， 1 表示反变量）

◆任意两个最大项的之和为 1

◆全体最大项之积为 0

2. 最小项和最大项的关系

两者之间为互补关系： $m_i=\bar{M}_i$ ，或者 $M_i=\bar{m}_i$

## 2.2.3 最大项表达式

例：逻辑电路的真值表如右，写出最小项和最大项表达式

| $A$ | $B$ | $C$ | $L$ | |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | $M_0$ |
| 0 | 0 | 1 | 0 | $M_1$ |
| 0 | 1 | 0 | 0 | $M_2$ |
| 0 | 1 | 1 | 1 | $m_3$ |
| 1 | 0 | 0 | 0 | $M_4$ |
| 1 | 0 | 1 | 1 | $m_5$ |
| 1 | 1 | 0 | 1 | $m_6$ |
| 1 | 1 | 1 | 0 | $M_7$ |

●最小项表达式：将 $L=1$ 的各个最小项相加

$$L(A,B,C)=m_3+m_5+m_6$$
$$=\sum m(3,5,6)$$
$$=\bar{A}\cdot B\cdot C+A\cdot\bar{B}\cdot C+A\cdot B\cdot\bar{C}$$

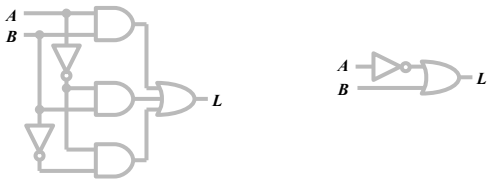
●最大项表达式：将 $L=0$ 的各个最大项相乘

$$L(A,B,C)=M_0\cdot M_1\cdot M_2\cdot M_4\cdot M_7$$
$$=\prod M(0,1,2,4,7)$$
$$=(A+B+C)\cdot(A+\bar{B}+C)\cdot(\bar{A}+B+C)\cdot(\bar{A}+\bar{B}+C)\cdot(\bar{A}+\bar{B}+\bar{C})$$



## 2.3 代数法化简

化简的目的：降低电路成本，以较少的门实现电路




两图电路逻辑功能相同

$$L=AB+\bar{A}B+A\bar{B}$$
$$=AB+A(B+\bar{B})$$
$$=AB+\bar{A}(B+1)$$
$$=AB+\bar{A}B+A=B+A$$

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



## 2.3 代数法化简

函数式变换的目的，除了化简，也可减少门电路的种类

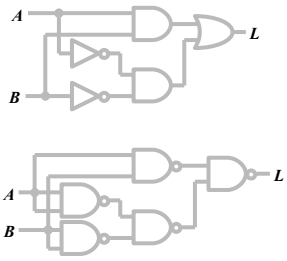
通常在一片集成电路芯片中只有一种门电路

例：已知 $L=AB\bar{D}+A\bar{B}D+ABD+\bar{A}BC\bar{D}+\bar{A}BCD$

（1）求最简的与 - 或式，并画出相应的逻辑图
（2）画出仅用与非门实现的电路

解：

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






## 2.4 卡诺图化简法 2.4.1 用卡诺图表示逻辑函数

**1.** 卡诺图的引出

卡诺图：将 $n$ 变量的全部最小项都用方格表示，并使具有逻辑相邻的最小项在几何位置上也相邻地排列起来，这样，所得到的图形叫 $n$ 变量的卡诺图

$n$ 个变量有 $2^n$ 个最小项，$n$ 个变量的卡诺图由 $2^n$ 个方格组成。每一个方格代表坐标值对应的一个最小项

逻辑相邻的最小项：如果两个最小项只有一个变量互为反变量，那么，就称这两个最小项在逻辑上相邻

例如三变量最小项：$m_6=AB\bar{C}$ 与 $m_7=AB\bar{C}$ 在逻辑上相邻。

| $m_6$ | $m_7$ |
|---|---|



## 2.4.1 用卡诺图表示逻辑函数

**2.** 卡诺图的特点：

●相邻方格对应的最小项，只有一个因子的差别，故其坐标按循环码排列

●如：00 , 01 , 11 , 10

两变量卡诺图

| $A$ \ $B$ | 0 | 1 |
|---|---|---|
| 0 | $m_0$ | $m_1$ |
| 1 | $m_2$ | $m_3$ |

三变量卡诺图

| $A$ \ $BC$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | $m_0$ | $m_1$ | $m_3$ | $m_2$ |
| 1 | $m_4$ | $m_5$ | $m_7$ | $m_6$ |

（$B$ 跨 11、10 两列；$C$ 跨 01、11 两列）

四变量卡诺图

| $AB$ \ $CD$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | $m_0$ | $m_1$ | $m_3$ | $m_2$ |
| 01 | $m_4$ | $m_5$ | $m_7$ | $m_6$ |
| 11 | $m_{12}$ | $m_{13}$ | $m_{15}$ | $m_{14}$ |
| 10 | $m_8$ | $m_9$ | $m_{11}$ | $m_{10}$ |

（$C$ 跨 11、10 两列；$B$ 跨 01、11、10 三行；$A$ 跨 11、10 两行；$D$ 跨 00、10 两列）



## 2.4.1 用卡诺图表示逻辑函数

**2.** 卡诺图特点

相邻方格的最小项，在逻辑上也是相邻的

卡诺图的几何相邻

① 相接——紧挨的

② 相对——任一行或一列的两头

③ 相重——对折起来后位置相重

$n$ 个变量的最小项 $m_i$，有 $n$ 个相邻项相邻的最小项可合并

$$m_{12}+m_{14}=AB\bar{C}\bar{D}+AB\bar{C}D$$
$$=AB\bar{D}(C+\bar{C})=AB\bar{D}$$

去掉的是不同的，保留的是相同的

例：找出 $m_{12}$ 和 $m_2$ 的相邻项

| $L$ \ $CD$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| $AB$ 00 | $m_0$ | $m_1$ | $m_3$ | $m_2$ |
| 01 | $m_4$ | $m_5$ | $m_7$ | $m_6$ |
| 11 | $m_{12}$ | $m_{13}$ | $m_{15}$ | $m_{14}$ |
| 10 | $m_8$ | $m_9$ | $m_{11}$ | $m_{10}$ |

$$m_{12}=A\bar{B}C\bar{D}$$
$$m_4=\bar{A}B\bar{C}D$$
$$m_8=A\bar{B}\bar{C}\bar{D}$$
$$m_{13}=A\bar{B}C\bar{D}$$
$$m_{14}=A\bar{B}CD$$

**3.** 逻辑函数的卡诺图

- 当逻辑函数为最小项表达式时，将表达式中出现的最小项对应的方格，在卡诺图中标为 **1**，其余方格标 **0**（也可不标），便得到逻辑函数的卡诺图
- 任何逻辑函数都等于其卡诺图中标 **1** 方格所对应的最小项之和

例如 画出 $L(A,B,C,D)=\sum m(0,1,2,3,4,8,10,11,14,15)$ 的卡诺图

| L, AB \ CD | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | $m_0$ | $m_1$ | $m_3$ | $m_2$ |
| 01 | $m_4$ | $m_5$ | $m_7$ | $m_6$ |
| 11 | $m_{12}$ | $m_{13}$ | $m_{15}$ | $m_{14}$ |
| 10 | $m_8$ | $m_9$ | $m_{11}$ | $m_{10}$ |

1 方格

0 方格

注：函数卡诺图中 **0** 可以不填





卡诺图是真值表的一种平面几何图形表示方法

若一组变量值使逻辑函数 $F$ 为 **1**，则该组值便是其卡诺图中标 **1** 方格的坐标

例如

$$F=\sum m(1,2,3,7)=\bar{A}\bar{B}C+\bar{A}B\bar{C}+\bar{A}BC+ABC$$

函数 $F$ 真值表

| A | B | C | F |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |

| F, A \ BC | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 |

函数 $F$ 的卡诺图





逻辑函数 $F$ 的卡诺图表示步骤：

① 求逻辑函数 $F$ 的与或式

② 根据变量数确定卡诺图形式

③ 根据各与项，在卡诺图上标 **1**

例如 $F=AB+\bar{C}D+A\bar{B}\bar{C}$

① 不需要

② 根据变量数确定卡诺图形式

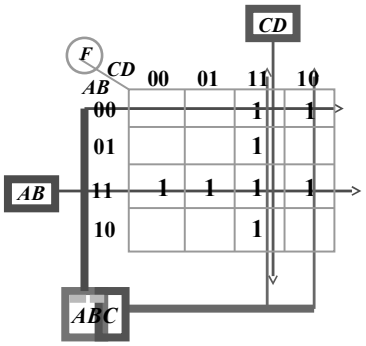
③ 根据各与项，在卡诺图上标 **1**

①$A=1, B=1, F=1$

②$C=1, D=1, F=1$

③$A=0, B=0, C=1, F=1$

其余填 **0**，也可不填，即得到 $F$ 的卡诺图








例 **2.2.3** 画出 $L$ 卡诺图

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

解：根据反演规则

$$\bar{L}=ABCD+AB\bar{C}D+A\bar{B}CD+A\bar{B}C\bar{D}+\bar{A}BC\bar{D}=\sum m(15,13,10,6,0)$$

画出 $\bar{L}$ 卡诺图

| $\bar{L}$, AB \ CD | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 1 | 0 | 0 | 0 |
| 01 | 0 | 0 | 0 | 1 |
| 11 | 0 | 1 | 1 | 0 |
| 10 | 0 | 0 | 0 | 1 |

画出 $L$ 卡诺图

| L, AB \ CD | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0 | 1 | 1 | 1 |
| 01 | 1 | 1 | 1 | 0 |
| 11 | 1 | 0 | 0 | 1 |
| 10 | 1 | 1 | 1 | 0 |

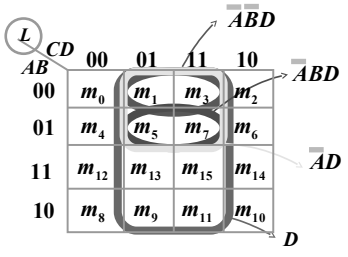
根据反函数在 $m_{15}, m_{13}, m_{10}, m_6, m_0$ 填 **0**，可直接画出 $L$ 卡诺图





**1.** 卡诺图化简的依据

依据：卡诺图中相邻方格对应的最小项在逻辑上相邻，而逻辑相邻的最小项可合并




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

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

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

去掉的是不同的，保留的是相同的

**2** 个相邻最小项合并，消去 **1** 个变量

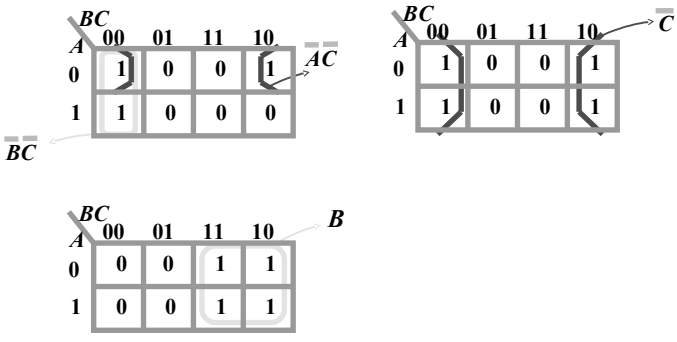
**4** 个相邻最小项合并，消去 **2** 个变量

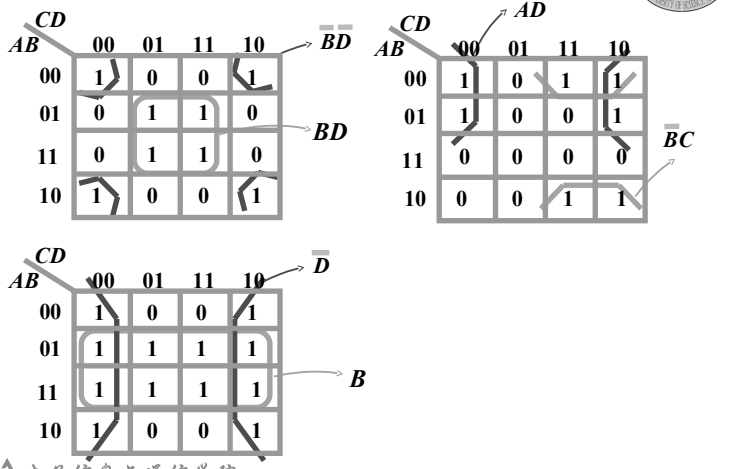
**8** 个相邻最小项合并，消去 **3** 个变量





- 任何逻辑函数都等于其卡诺图中为 **1** 的方格所对应的最小项之和
- 若将卡诺图中 **1** 方格所对应的最小项合并，则达到化简的目的










**2.** 卡诺图化简的步骤

**(1)** 将逻辑函数转换成与 - 或表达式

**(2)** 根据逻辑函数与 - 或表达式填卡诺图，得到逻辑函数的卡诺图

**(3)** 合并最小项

- 相邻的 $2^n$ 个 **1** 格圈为一个卡诺圈，各圈对应一个乘积项
- **1** 格可重复被圈，但新增圈中必须有未曾被圈过的 **1** 格
- 卡诺圈中的格数尽量多，卡诺圈的数量尽量少

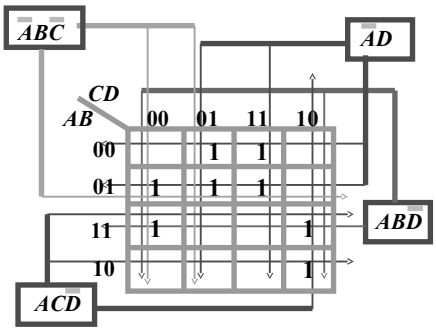
(4) 将各卡诺圈对应的乘积项相或，便得到最简与或表达式





例：用卡诺图化简 $F=\underline{AB\,D}+\underline{A\,D}+\underline{AB\,C}+\underline{AC\,D}$

**(1)** 画出逻辑函数的卡诺图




$F$ 的卡诺图

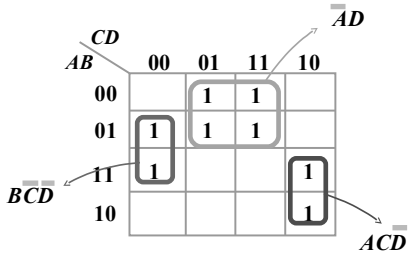




例：用卡诺图化简 $F=\underline{AB\,D}+\underline{A\,D}+\underline{AB\,C}+\underline{AC\,D}$

**(2)** 根据最小项合并规律画卡诺圈， 圈住全部 **1** 格

**(3)** 将每个卡诺圈对应的与项相或，就得到最简与或表达式




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

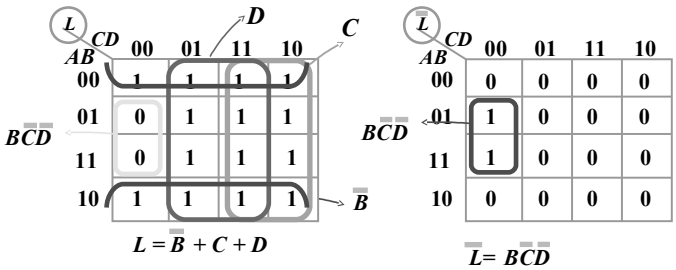




例 **2.2.6** 用卡诺图化简

$$L(A,B,C,D)=\sum m(0..3,5..11,13..15)$$

解：画出函数 $L$ 的卡诺图　　画出函数 $\overline{L}$ 的卡诺图




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

$\overline{L}=B\overline{C}\overline{D}$

$L=\overline{B\,C\,D}=B+C+D$

- 可用圈 **0** 法先求出反函数最简与或式，再求反





**3.** 含无关项的逻辑函数化简

- 一些实际问题中，输入变量之间存在相互制约或特殊限定等
- 部分变量取值与实际问题无关，即称为包含无关条件的逻辑问题
- 这些无关条件的变量取值对应的最小项称为无关项或任意项
- 卡诺图化简时，这些无关项可根据实际化简目的取任意值

## 2.4.2 用卡诺图化简逻辑函数

例 2.2.7 设计电路，能判断一位十进制数是奇数还是偶数
奇数时，电路输出为 1，为偶数时，电路输出为 0

解：
(1) 列出真值表
(2) 画卡诺图化简

| A | B | C | D | L |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | × |
| 1 | 0 | 1 | 1 | × |
| 1 | 1 | 0 | 0 | × |
| 1 | 1 | 0 | 1 | × |
| 1 | 1 | 1 | 0 | × |
| 1 | 1 | 1 | 1 | × |

$L = D$

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

必须填无关项
否则可能错当 0



## 2.5 硬件描述语言 Verilog HDL 基础

- HDL ( Hardware Description Language )
  - 类似于高级程序语言，文本形式描述数字电路结构和行为的语言
  - 可描述逻辑电路图，逻辑表达式，信号及逻辑电路功能行为
- 逻辑仿真（前仿真，行为仿真）
  - 用计算机仿真软件对数字逻辑电路的结构和行为进行预测
  - 仿真器对 HDL 描述进行解释，以文本或波形描述电路输出
- 逻辑综合
  - 根据 HDL 描述导出电路基本元件表及其连接关系（门级网表）的过程
  - 类似高级程序语言的编译过程
  - 根据网表数据库，可制作出集成电路或印刷电路板 PCB
- 时序仿真（后仿真）



## 2.5.1 Verilog HDL 语言的基本语法规则

- 建模：用规定的语法规则描述数字电路的过程
- 间隔符：分隔文本
  - 如：空格符（\b）、TAB 键（\t）、页符
- 注释符：改善可读性，不影响编译
  - 多行注释符（用于写多行注释）： /* --- */
  - 单行注释符：以 // 开始到行尾结束为注释文字
- 标识符：用字符串标识对象（如模块、输入／输出端口、变量）
  - 以英文字母或下划线开始
  - 如 clk , counter8 , _net , bus_A
- 关键词：用来定义语言结构的特殊词汇
  - 关键词都是小写，关键词不能作为标识符使用
  - 如 module , endmodule , input , output , wire , reg , and 等



## 2.5.1 Verilog HDL 语言的基本语法规则

- 逻辑值集合： Verilog 语言规定了 4 种基本的逻辑值

| | |
|---|---|
| 0 | 逻辑 0 ， 逻辑假 |
| 1 | 逻辑 1 ， 逻辑真 |
| x 或 X | 不确定的值（未知状态） |
| z 或 Z | 高阻态 |

常量
- 整数型
  - 十进制数的形式的表示方法：表示有符号常量
    例如： 30 , -2
  - 带基数的形式的表示方法：表示常量
    格式为： <+/-><位宽>'<基数符号><数值>
    例如： 3'b101 , 5'o37 , 8'he3 , 8'b1001_0011
- 实数型常量
  - 十进制记数法 如： 0.1 , 2.0 , 5.67
  - 科学记数法 如 : 23_5.1e2 = $235.1\times10^2$
    5e-4 = $5\times10^{-4}$



## 2.5.1 Verilog HDL 语言的基本语法规则

- Verilog 参数定义语句可定义标识符代表常量，称为符号常量
- 格式为：

```
parameter 参数名 1 =常量表达式 1 , 参数名 2 =常量表达式 2 , ……;
```

如： `parameter BIT = 1, BYTE = 8, PI = 3.14;`

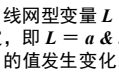
- 字符串：字符串是双撇号内的字符序列



## 2.5.2 变量的数据类型

- 线网型：用于标识连线，一般指电路中的物理连接

例：线网型变量 *L* 的值由与门的驱动信号 *a* 和 *b* 所决定，即 $L = a \,\&\, b$
*a* , *b* 的值发生变化，线网 *L* 的值会立即跟着变化




常用的线网类型由关键词 `wire` 定义
`wire` 型变量的定义格式如下：

```
wire [n-1:0] 变量名 1 , 变量名 2 , …, 变量名 n ;
```

变量位宽度

```
例: wire a, b, L;          // 将输出信号 L 声明为线网型变量
    wire [7:0] data_bus;   // 声明 8-bit 宽的线网型总线变量
```

## 2.5.2 变量的数据类型

●寄存型：用于行为描述时，暂存临时状态或中间结果

寄存型变量只能在 **initial** 或 **always** 语句块内被赋值

行为级描述是电路功能行为的抽象描述，并不描述电路的具体组成结构

4 种寄存器类型的变量

| 寄存器类型 | 功能说明 |
|---|---|
| **reg** | **1** 位寄存型变量 |
| **integer** | **32** 位带符号的整数寄存型变量 |
| **real** | **64** 位带符号的实数寄存型变量 |
| **time** | **64** 位无符号的时间寄存型变量 |

例：

```
reg clock;            // 定义一个 1 位寄存型变量
reg [3:0] counter;    // 定义一个 4 位寄存型变量
```



## 2.5.3 运算符及其优先级

**1.** 运算符

| | | |
|---|---|---|
| {} {{}} | Concatenation, replication | 拼接，复制 |
| unary + unary - | Unary operators | 单目正负 |
| + - * / ** | Arithmetic | 算术 |
| % | Modulus | 模 |
| > >= < <= | Relational | 关系 |
| ! | Logical negation | 逻辑非 |
| && | Logical and | 逻辑与 |
| \|\| | Logical or | 逻辑或 |
| == | Logical equality | 逻辑相等 |
| != | Logical inequality | 逻辑不等 |
| === 全等 | Case equality **(including x and z)** | |
| !== 不全等 | Case inequality **(including x and z)** | |

| | | |
|---|---|---|
| ~ | Bitwise negation | 位非 |
| & | Bitwise and | 位与 |
| \| | Bitwise inclusive or | 位或 |
| ^ | Bitwise exclusive or | 位异或 |
| ^~ or ~^ | Bitwise equivalence | 位同或 |
| & | Reduction and | 单目缩位与 |
| ~& | Reduction nand | 单目缩位与非 |
| \| | Reduction or | 单目缩位或 |
| ~\| | Reduction nor | 单目缩位或非 |
| ^ | Reduction xor | 单目缩位异或 |
| ~^ or ^~ | Reduction xnor | 单目缩位同或 |
| << | Logical left shift | 逻辑左移 |
| >> | Logical right shift | 逻辑右移 |
| <<< | Arithmetic left shift | 算术左移 |
| >>> | Arithmetic right shift | 算术右移 |
| ? : | Conditional | 条件 |



## 2.5.3 运算符及其优先级

位拼接运算符

将两个或多个信号的某些位拼接起来成为一个新的操作数

设 `A = 1'b1 , B = 2'b10 , C = 2'b00`

`{B, C}` = `4'b10_00`

`{A, B[1], C[0]}` = `3'b1_1_0`

`{A, B, C, 3'b101} = 8'b1_10_00_101`

对同一个操作数的重复拼接还可以双重大括号构成的运算符 `{ { } }`

例如 `{4{A}} = 4'b1111`

`{{2{A}}, {2{B}}, C} = 8'b11_1010_00`



## 2.5.3 运算符及其优先级

位运算与缩位运算

***A***： **4'b1010**
***B***： **4'b1111**

```
A: 1010
B: 1111
   &
1010
```

**A & B**： **A** 与 **B** 进行位运算

**& A**： **A** 进行缩位运算，`1 & 0 & 1 & 0 = 0`



## 2.5.3 运算符及其优先级

**2.** 运算符的优先级

| | |
|---|---|
| + - ! ~ & ~& \| ~\| ^ ~^ ^~ (unary) | Highest precedence |
| ** | |
| * / % | |
| + - (binary) | |
| << >> <<< >>> | |
| < <= > >= | |
| == != === !== | |
| & (binary) | |
| ^ ^~ ~^ (binary) | |
| \| (binary) | |
| && | |
| \|\| | |
| ?: (conditional operator) | |
| {} {{}} | Lowest precedence |



## 2.5.4 Verilog 内部的基本门级元件

门级建模：将逻辑电路图用 **HDL** 规定的文本语言表示出来

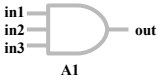
基本门级元件模型

| | | | | |
|---|---|---|---|---|
| 多输入门 | **and** | 与门 | **nand** | 与非门 |
| | **or** | 或门 | **nor** | 或非门 |
| | **xor** | 异或门 | **xnor** | 异或非门 |
| 多输出门 | **buf** | 缓冲器 | **not** | 反相器 |
| 三态门 | **bufif1** | 高电平有效控制三态缓冲器 | **notif1** | 高电平有效控制三态反相器 |
| | **bufif0** | 低电平有效控制三态缓冲器 | **notif0** | 低电平有效控制三态反相器 |

**1.** 多输入门

只允许有一个输出，但可以有多个输入

```
and A1(out, in1, in2, …);
    调用名
or  O1(out, in1, in2, …);
xor X1(out, in1, in2, …);
```




**and**

| in1 | in2 | out |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 0 | x | 0 |
| 0 | z | 0 |
| 1 | 1 | 1 |
| 1 | x | x |
| 1 | z | x |
| x | x | x |
| x | z | x |
| z | z | x |

**or**

| in1 | in2 | out |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 0 | x | x |
| 0 | z | x |
| 1 | 1 | 1 |
| 1 | x | 1 |
| 1 | z | 1 |
| x | x | x |
| x | z | x |
| z | z | x |

**xor**

| in1 | in2 | out |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 0 | x | x |
| 0 | z | x |
| 1 | 1 | 0 |
| 1 | x | x |
| 1 | z | x |
| x | x | x |
| x | z | x |
| z | z | x |

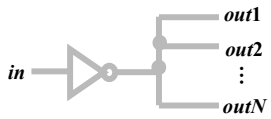


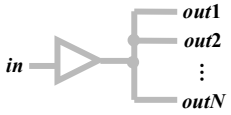


**2.** 多输出门

允许有多个输出，但只有一个输入

```
not N1(out1, out2, …, in);
buf B1(out1, out2, …, in);
```




**not**

| in | out |
|---|---|
| 0 | 1 |
| 1 | 0 |
| x | x |
| z | x |

**buf**

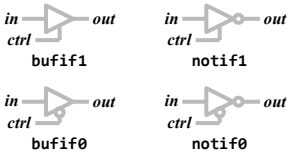
| in | out |
|---|---|
| 0 | 0 |
| 1 | 1 |
| x | x |
| z | x |





**3.** 三态门

有输出、数据输入和输入控制端各 **1** 个，若控制信号无效，则输出高阻态 **z**




**bufif1**

| ctrl | in | out |
|---|---|---|
| 0 | 0 | z |
| 0 | 1 | z |
| 0 | x | z |
| 0 | z | z |
| 1 | 0 | 0 |
| 1 | 1 | 1 |
| 1 | x | x |
| 1 | z | x |

**notif1**

| ctrl | in | out |
|---|---|---|
| 0 | 0 | z |
| 0 | 1 | z |
| 0 | x | z |
| 0 | z | z |
| 1 | 0 | 1 |
| 1 | 1 | 0 |
| 1 | x | x |
| 1 | z | x |





- 模块是 Verilog 描述电路的基本单元

```
module ABC(port1, port2, ...); // 各模块通过端口相互连接
  inout/input/output ... ;     // 端口类型说明
  reg/wire/ ... ;              // 变量类型说明
  parameter ... ;              // 参数定义，逗号分割多个参数

  and/or/ ... ;                // 结构描述方式，逻辑门或底层模块
  assign ... ;                 // 数据流描述方式，连续赋值语句
  always/initial               // 行为描述方式，过程块结构
  begin                        // 多条语句组成的行为描述块
    if ... else ... ;          // 条件描述语句
    case                       // 分支描述语句
      ...
    endcase                    // 无实际语义的语句无分号
  end
endmodule
```





- Verilog 2005 标准新增的参数和端口定义形式

```
module ABC #(parameter WIDTH = 32)
            (output reg port1, inout port2, ...);
...
endmodule
```

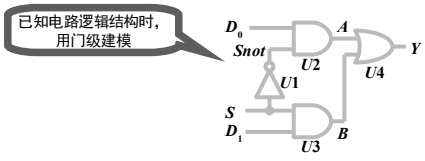
- 新增的模块使用形式

```
ABC #(8) abc1(.port2(x), .port1(y), ...);
```





例 用结构描述方式建立门电路 Verilog 模型




模块名

```
module mux2to1(output Y, input[1:0] D, input S);
  wire Snot, A, B; // 定义内部节点信号数据类型
  not U1(Snot, S);
  and U2(A, D[0], Snot);
  and U3(B, D[1], S);
  or U4(Y, A, B);
endmodule
```

数据类型说明

用门级建模方式
描述电路结构

## 2.5.5 Verilog 程序的基本结构

例 用数据流描述方式建立模型

$$Y = D_0 \cdot S + D_1 \cdot S$$

已知逻辑表达式时，用数据流建模

```
module mux2to1(output Y, input[1:0] D, input S);
  assign Y = (~S & D[0]) | (S & D[1]);
endmodule
```

- assign 语句，左边变量的数据类型必须是 wire 型
- 端口变量默认为 wire 型



## 2.5.5 Verilog 程序的基本结构

例 用行为描述方式建模，实现二选一选择器电路

已知电路行为功能要求时，用行为级建模

```
module mux2to1(output reg Y, input[1:0] D, input S);
  always @(S, D)
    if(S == 1) Y = D[1];
    else Y = D[0];
endmodule
```

- always 和 initial 语句块中，被赋值变量类型必须是寄存型



## 2.5.6 逻辑功能的仿真与测试

1. 编写 Testbench 测试向量

```
`include "mux2to1.v"                // 包含被测试模块描述文件
`timescale 1us/100ns                // 定义单位时间 / 精度
module testbench;                   // 测试模块
  reg[1:0] D;                       // 定义测试信号
  reg S;
  wire Y;
  mux2to1 test(Y, D, S);            // 放置被测模块，将测试信号接入端口
  initial begin                     // 初始化（不可综合的行为描述）
    // 根据指定信号事件，格式化输出字符串（系统任务或函数以 $ 标识）
    $monitor($time, ": D0 D1 S Y -> %d %d %d %d", D[0], D[1], S, Y);
    $dumpfile("testbench.vcd");     // 设置信号输出文件
    $dumpvars(0, testbench);        // 指定输出的信号范围（范围，模块）
    $display("Running testbench");  // 格式化输出字符串，类似 printf
    #20 $stop;                      // 20 个单位时间后，结束仿真
  end
  initial begin                     // 初始化（各描述块是独立并行的）
    D = 2'b00;
    S = 0;
    #10 S = 1;                      // 10 个单位时间后改变 S 信号
  end
  always #1 D[0] = ~D[0];           // 周期性行为，各描述块是独立并行的
  always #5 D[1] = ~D[1];           // D[0] 周期 2us，D[1] 周期 10us
endmodule
```



## 2.5.6 逻辑功能的仿真与测试

1. 安装 iverilog

   •sudo apt install iverilog, gtkwave

2. 编译

   iverilog testbench.v -o testbench.o

3. 行为仿真（前仿真）

   vvp -n testbench.o  # 非单步仿真

4. 波形分析

   gtkwave testbench.vcd

5. 综合：用支持所选器件的综合工具输出网表数据

6. 时序仿真（后仿真）：不同器件时延参数不同，可能影响结果

7. 下载网表数据至所选器件



## 3.7 逻辑描述中的几个问题 3.7.1 正负逻辑问题

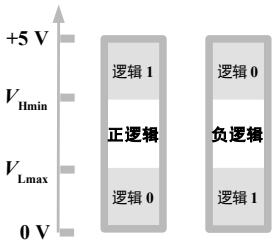
1. 正负逻辑的规定

在数字系统中，可用两种逻辑体制表示电路输入和输出的高、低电平

正逻辑体制：将高电平用逻辑 1 表示，低电平用逻辑 0 表示

负逻辑体制：将高电平用逻辑 0 表示，低电平用逻辑 1 表示

若无特别说明，一般采用正逻辑体制






## 3.7.1 正负逻辑问题

2. 正负逻辑等效变换

逻辑内涵一致（电路功能一致）的正、负逻辑表达式是等效的

输入 / 输出电平关系

| A | B | F |
|---|---|---|
| L | L | H |
| L | H | H |
| H | L | H |
| H | H | L |

正逻辑

| A | B | F |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

$F = A \cdot B$

负逻辑

| A | B | F |
|---|---|---|
| 1 | 1 | 0 |
| 1 | 0 | 0 |
| 0 | 1 | 0 |
| 0 | 0 | 1 |

$F = A + B$

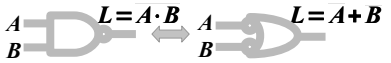
与非 ⟺ 或非

与 ⟺ 或

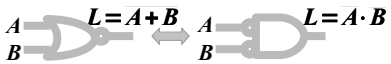
**1.** 基本逻辑门电路的等效符号

同一逻辑体制下，逻辑内涵一致的逻辑门电路符号，称为等效符号

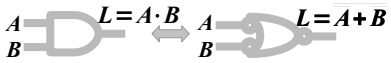
与非门及等效符号如图



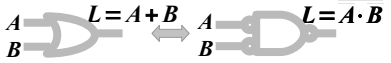

或非门及等效符号如图




与门及等效符号如图




或门及等效符号如图



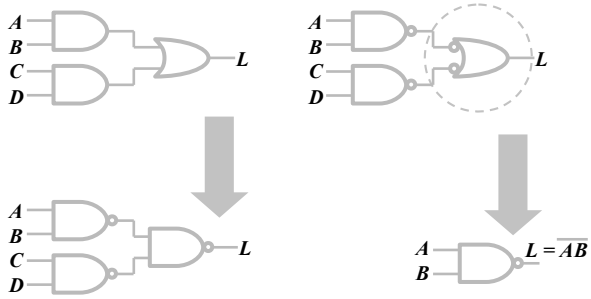





**2.** 逻辑门等效符号的应用

利用逻辑门等效符号，可直接变换逻辑电路

$L = AB + CD = \overline{AB} + \overline{CD} = \overline{AB \cdot CD}$　　$L = A + B = \overline{A \cdot B}$








**3.** 逻辑门等效符号可强调有效电平，便于理解信号高低电平作用

低电平有效：信号低电平时，电路履行预期功能，用反变量形式表示

高电平有效：信号高电平时，电路履行预期功能，用原变量形式表示

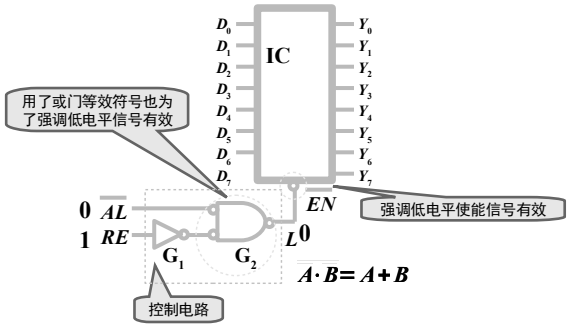
例如：高电平有效的三态非门，$EN = 1$ 时，具有非门功能，否则输出为高阻状态

低电平有效的三态非门，$\overline{EN} = 0$ 时，具有非门功能，否则输出为高阻状态






**3.** 逻辑门等效符号可强调有效电平，便于理解信号高低电平作用




当 $RE = 1$，$\overline{AL} = 0$，$G_2$ 门两个输入均有效，输出有效，$\overline{EN}$ 有效





**3.** 逻辑门等效符号可强调有效电平，便于理解信号高低电平作用

可根据不同控制信号要求，用等效符号描述有效控制信号

| 要求 | 要求 | 要求 |
|---|---|---|
| $RE = 1, AL = 1$ | $RE = 0, \overline{AL} = 0$ | $RE = 1, AL = 1$ |
| $L = 0$ | $L = 1$ | $L = 1$ |






| 康华光教材 7 版 | 罗杰教材 |
|---|---|
| 2.1.1 | 2.1.1 (5, 6) |
| 2.1.2 | 2.1.2 |
| 2.2.7 | 2.3.5 |
| 2.3.4 | 2.2.4 |
| 2.4.1 | 2.4.1 |
| 2.4.3 | 2.4.3 |
| 2.5.6 | 6.4.1 |