

2. 逻辑代数与硬件描述语言基础

2.1 逻辑代数的基本定理和规则

2.2 逻辑函数表达式的形式

2.3 逻辑函数的代数化简法

2.4 逻辑函数的卡诺图化简法

2.5 硬件描述语言Verilog HDL基础

教学基本要求

- 1、熟悉逻辑代数常用基本定律、恒等式和规则。
- 2、掌握逻辑代数的表示方法；
- 3、掌握逻辑代数的变换和卡诺图化简法；
- 4、熟悉硬件描述语言Verilog HDL

2.1 逻辑代数的基本定理和规则

2.1.1 逻辑代数的基本定律和恒等式

2.1.2 逻辑代数的基本规则

2.1 逻辑代数的基本定理和规则

逻辑代数又称布尔代数。它是分析和设计现代数字逻辑电路不可缺少的数学工具。逻辑代数有一系列的定律、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。

逻辑关系指的是事件产生的条件和结果之间的因果关系。在数字电路中往往是将事情的条件作为输入信号，而结果用输出信号表示。条件和结果的两种对立状态分别用逻辑“1”和“0”表示。

2.1.1 逻辑代数的基本定律和恒等式

1、基本公式

0、1律: $A + 0 = A$ $A + 1 = 1$ $A \cdot 1 = A$ $A \cdot 0 = 0$

互补律: $A + \overline{A} = 1$ $A \cdot \overline{A} = 0$

交换律: $A + B = B + A$ $A \cdot B = B \cdot A$

结合律: $A + B + C = (A + B) + C$ $A \cdot B \cdot C = (A \cdot B) \cdot C$

分配律: $A (B + C) = AB + AC$ $A + BC = (A + B)(A + C)$

重叠律:

$$A + A = A$$

$$A \cdot A = A$$

反演律(摩根定理): $\overline{A + B} = \overline{A} \cdot \overline{B}$ $\overline{AB} = \overline{A} + \overline{B}$

吸收律

$$A + A \cdot B = A$$

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

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

$$(A + B) \cdot (A + C) = A + BC$$

其它常用恒等式

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

$$AB + \overline{A}C + BCD = AB + \overline{A}C$$

2、基本公式的证明

(真值表证明法)

例 证明 $A + \bar{A} \cdot B = A + B$

列出等式、右边的函数值的真值表

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

例：试化简下列逻辑函数 $L = (A + B)(\bar{A} + B)$

$$L = A\bar{A} + AB + B\bar{A} + BB \text{ (分配律)}$$

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

$$= AB + B\bar{A} + B \quad (A + 0 = A)$$

$$= B(A + \bar{A} + 1) \quad [AB + AC = A(B + C)]$$

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

2.1.2 逻辑代数的基本规则

1. 代入规则：在包含变量 A 逻辑等式中，如果用另一个函数式代入式中所有 A 的位置，则等式仍然成立。这一规则称为代入规则。

例： $B(A + C) = BA + BC$,

用 $A + D$ 代替 A ，得

$$B[(A + D) + C] = B(A + D) + BC = BA + BD + BC$$

代入规则可以扩展所有基本公式或定律的应用范围

2. 反演规则:

对于任意一个逻辑表达式 L ，若将其中所有的与（ $\cdot$ ）换成或（ $+$ ），或（ $+$ ）换成与（ $\cdot$ ）；原变量换为反变量，反变量换为原变量；将1换成0，0换成1；则得到的结果就是原函数的反函数。

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

解：按照反演规则，得

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

3. 对偶规则:

对于任何逻辑函数式, 若将其中的与 ($\cdot$) 换成或 ($+$), 或 ($+$) 换成与 ($\cdot$); 并将1换成0, 0换成1; 那么, 所得的新的函数式就是 L 的对偶式, 记作 L' 。

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

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

当某个逻辑恒等式成立时, 则该恒等式两侧的对偶式也相等。这就是对偶规则。利用对偶规则, 可从已知公式中得到更多的运算公式, 例如, 吸收律

4. 香农展开定理:

任何一个逻辑函数都可以重新表示为

$$f(x_1, x_2, \dots, x_{i-1}, x_i, x_{i+1}, \dots, x_n) = \bar{x}_i \cdot f(x_1, x_2, \dots, x_{i-1}, 0, x_{i+1}, \dots, x_n) \\ + x_i \cdot f(x_1, x_2, \dots, x_{i-1}, 1, x_{i+1}, \dots, x_n)$$

例:利用香农展开定理将3变量函数变为2变量函数, 将变量A分解出来。 $L(A,B,C) = AB+BC+AC$

$$\begin{aligned} L(A,B,C) &= \bar{A} \cdot L(0,B,C) + A \cdot L(1,B,C) \\ &= \bar{A} \cdot (0 \cdot B + BC + 0 \cdot B) + A \cdot (1 \cdot B + BC + 1 \cdot C) \\ &= \bar{A} \cdot (BC) + A \cdot (B + C) \\ &= \bar{A} \cdot L_0 + A \cdot L_1 \end{aligned}$$

式中, $L_0 = BC$, $L_1 = B + C$

可以减少函数自变量的数目, 降低函数的复杂度。

2.2 逻辑函数表达式的形式

2.2.1 逻辑函数表达式的基本形式

2.2.2 最小项与最小项表达式

2.2.3 最大项与最大项表达式

2.2 逻辑函数表达式的形式

2.2.1 逻辑函数表达式的基本形式

1、与-或表达式

若干与项进行或逻辑运算构成的表达式。由与运算符和或运算符连接起来。

$$L = A \cdot C + \bar{C} \cdot D$$

2、或-与表达式

若干或项进行与逻辑运算构成的表达式。由或运算符和与运算符连接起来。

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

通常表达式为混合形式 $L = A \cdot (B \cdot C + \bar{B} \cdot \bar{C}) + A \cdot (B \cdot \bar{C} + \bar{B} \cdot C)$

经过变换可转换为上述两种基本形式

2.2.2 最小项与最小项表达式

1. 最小项的定义和性质

(1) 定义

n 个变量 $X_1, X_2, \dots, X_n$ 的最小项是 n 个因子的乘积，每个变量都以它的原变量或非变量的形式在乘积项中出现，且仅出现一次。一般 n 个变量的最小项应有 2^n 个。

例如， A 、 B 、 C 三个逻辑变量的最小项有（ $2^3=$ ）8个，即

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

$\overline{A}B$ 、 $\overline{A}BC\overline{A}$ 、 $A(B+C)$ 等则不是最小项。

(2) 最小项的性质

三个变量的所有最小项的真值表

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

- 对于任意一个最小项，只有一组变量取值使得它的值为1；
- 任意两个最小项的乘积为0；
- 全体最小项之和为1。

(3) 最小项的编号

三个变量的所有最小项的真值表

			m_0	m_1	m_2	m_3	m_4	m_5	m_6	m_7
A	B	C	$\overline{A}\overline{B}\overline{C}$	$\overline{A}\overline{B}C$	$\overline{A}B\overline{C}$	$\overline{A}BC$	$A\overline{B}\overline{C}$	$A\overline{B}C$	$AB\overline{C}$	ABC
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

最小项的表示：通常用 m_i 表示最小项， m 表示最小项，下标 i 为最小项号。

2. 最小项表达式

由若干最小项相或构成的表达式，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。

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

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

例2 将 $L(A, B, C) = \overline{(AB + \overline{A}\overline{B} + \overline{C})\overline{A}\overline{B}}$
化成最小项表达式

a. 去掉非号 $L(A, B, C) = \overline{(AB + \overline{A}\overline{B} + \overline{C})} + AB$
 $= (\overline{AB} \cdot \overline{\overline{A}\overline{B}} \cdot \overline{\overline{C}}) + AB$
 $= (\overline{A} + \overline{B})(A + B)C + AB$

b. 去括号

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

2.2.3 最大项与最大项表达式

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 个因子或相，每个变量都以它的原变量或非变量的形式在或项中出现，且仅出现一次。一般 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

$$\begin{aligned} &(\bar{A} + \bar{B} + \bar{C}), (\bar{A} + \bar{B} + C), (\bar{A} + B + \bar{C}), (\bar{A} + B + C), \\ &(A + \bar{B} + \bar{C}), (A + \bar{B} + C), (A + B + \bar{C}), (A + B + C) \end{aligned}$$

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， M 表示最大项，下标 i 为最大项号。

最大项的性质：

- 对于任意一个最大项，只有一组变量取值使得它的值为0；
- 任意两个最大项之和为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： $m_i = \overline{M_i}$ ，或者 $M_i = \overline{m_i}$

例：逻辑电路的真值表如右，写出最小项和最大项表达式。

最小项表达式：

将 $L=1$ 的各个最小项相加

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

最大项表达式：

将 $L=0$ 的各个最大项相乘

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

A	B	C	L
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	$1 \rightarrow m_3$
1	0	0	0
1	0	1	$1 \rightarrow m_5$
1	1	0	$1 \rightarrow m_6$
1	1	1	0

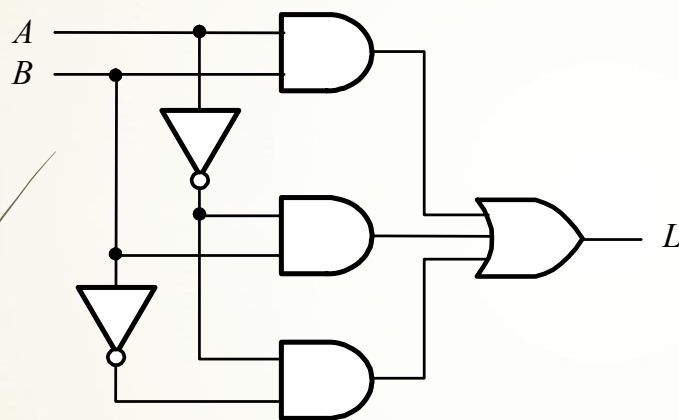
2.3 逻辑函数的代数化简法

2.3.1 逻辑函数的最简形式

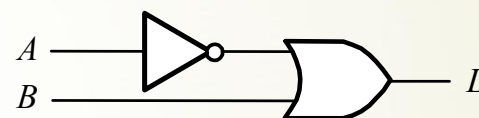
2.3.2 逻辑函数的代数化简法

2.3 逻辑函数的代数法化简

化简的目的：降低电路实现的成本，以较少的门实现电路。



(a) 标准积之和的电路

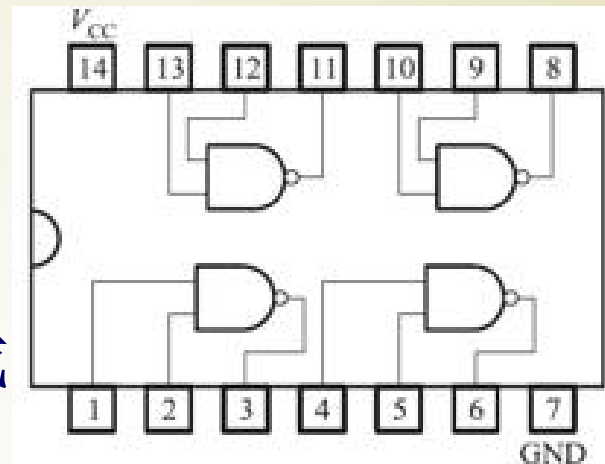


(b) 成本最低的电路

图 (a) 和图 (b) 的电路逻辑功能相同，但图 (b) 电路简单可靠性高，成本低。

2.3.1 逻辑函数的最简形式

逻辑函数有不同形式，如与-或表达式、与非-与非表达式、或-与表达式、或非-或非表达式以及与-或-非表达式等。



将其中包含的与项数最少，且每个与项中变量数最少的与-或表达式称为最简与-或表达式。

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

“与-或” 表达式

“与非-与非” 表达式

“或-与” 表达式

“或非-或非” 表达式

“与-或-非” 表达式

2.3.2 逻辑函数的代数化简法

1、逻辑函数的化简

化简的主要方法：

(1) 公式法（代数法）

(2) 图解法（卡诺图法）

代数化简法：

运用逻辑代数的基本定律和恒等式进行化简的方法。

并项法： $A + \overline{A} = 1$

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

吸收法: $A + AB = A$

$$L = \underline{\bar{A}B} + \underline{\bar{A}BCD}(E + F) = \bar{A}B$$

消去法: $A + \bar{A}B = A + B$

$$L = AB + \underline{\bar{A}C} + \underline{\bar{B}C} = AB + (\bar{A} + \bar{B})C \quad \boxed{\bar{A} + \bar{B} = \overline{AB}}$$

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

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

配项法: $A + \bar{A} = 1$

$$L = AB + \bar{A}\bar{C} + \underline{BC} = AB + \bar{A}\bar{C} + \underline{(A + \bar{A})BC}$$

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

$$= (\underline{AB + ABC}) + (\underline{\bar{A}\bar{C} + \bar{A}CB})$$

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

2、逻辑函数形式的变化

通常在一片集成电路芯片中只有一种门电路，为了减少门电路的种类，需要对逻辑函数表达式进行变换。

例：已知 $L = ABD + \bar{A}\bar{B}\bar{D} + ABD + \bar{A}\bar{B}\bar{C}D + \bar{A}\bar{B}CD$

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

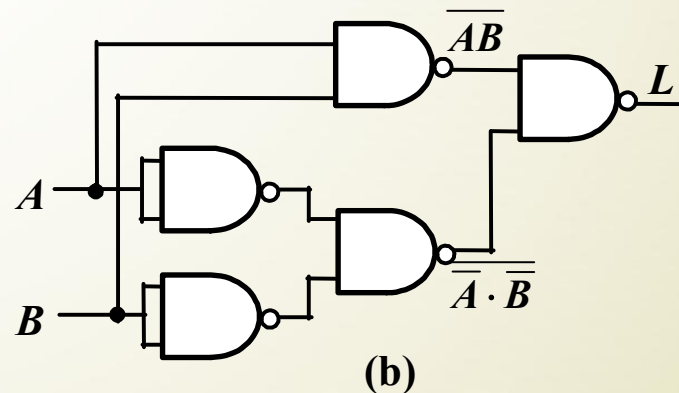
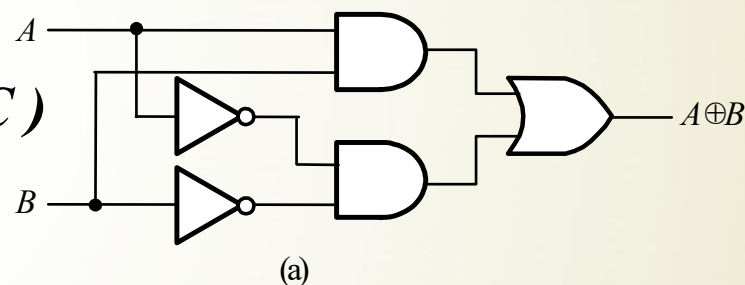
$$= AB + \bar{A}\bar{B}\bar{D} + \bar{A}\bar{B}D$$

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

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

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

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



2.4 逻辑函数的卡诺图化简法

2.4.1 用卡诺图表示逻辑函数

2.4.2 用卡诺图化简逻辑函数

代数法化简在使用中遇到的困难：

- 1.逻辑代数与普通代数的公式易混淆，化简过程要求对所有公式熟练掌握；
 - 2.代数法化简无一套完善的方法可循，它依赖于人的经验和灵活性；
 - 3.用这种化简方法技巧强，较难掌握。特别是对代数化简后得到的逻辑表达式是否是最简式判断有一定困难。
- 卡诺图法可以比较简便地得到最简的逻辑表达式。

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小方块表示，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。

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。

如最小项 $m_6=ABC\bar{C}$ 、与 $m_7=ABC$ 在逻辑上相邻

m_6	m_7
-------	-------

2、卡诺图的特点

各小方格对应于各变量不同的组合，而且上下左右在几何上相邻的方格内只有一个因子有差别，这个重要特点成为卡诺图化简逻辑函数的主要依据。

两变量卡诺图

A \ B	0	1
0	m_0	m_1
1	m_2	m_3

四变量卡诺图

AB \ CD		C			
		00	01	11	10
A	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}

三变量卡诺图

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

3、已知逻辑函数画卡诺图

当逻辑函数为最小项表达式时，在卡诺图中找出和表达式中最小项对应的小方格填上1，其余的小方格填上0（有时也可用空格表示），就可以得到相应的卡诺图。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。

例1：画出逻辑函数

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

$\textcircled{L}$ $AB \backslash CD$	00	01	11	10
00	1	1	1	1
01	1	0	0	0
11	0	0	1	1
10	1	0	1	1

例2：画出下式的卡诺图

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

解 1. 将逻辑函数化为最小项表达式

$$\bar{L} = ABCD + AB\bar{C}D + A\bar{B}C\bar{D} + \bar{A}BC\bar{D} + \bar{A}\bar{B}C\bar{D}$$

$$= \sum m(0, 6, 10, 13, 15)$$

2. 填写卡诺图

$\textcircled{L}$ 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

2.4.2 用卡诺图化简逻辑函数

1、化简的依据

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

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

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

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

$$\overline{A}\overline{B}D + ABD = AD$$

$$\overline{A}D + AD = D$$

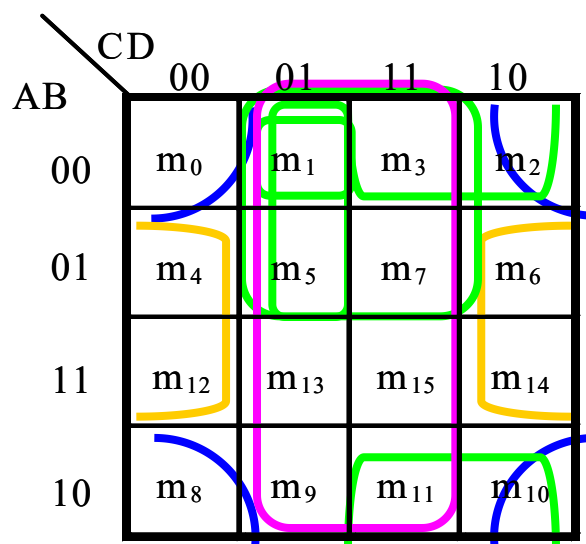
2、化简的步骤

用卡诺图化简逻辑函数的步骤如下：

- (1) 将逻辑函数写成最小项表达式
- (2) 按最小项表达式填卡诺图，凡式中包含了的最小项，其对应方格填1，其余方格填0。
- (3) 合并最小项，即将相邻的1方格圈成一组(包围圈)，每一组含 2^n 个方格，对应每个包围圈写成一个新的乘积项。本书中包围圈用虚线框表示。
- (4) 将所有包围圈对应的乘积项相加。

画包围圈时应遵循的原则：

- (1) 包围圈内的方格数一定是 2^n 个，且包围圈必须呈矩形。
- (2) 循环相邻特性包括上下底相邻，左右边相邻和四角相邻。
- (3) 同一方格可以被不同的包围圈重复包围多次，但新增的包围圈中一定要有原有包围圈未曾包围的方格。
- (4) 一个包围圈的方格数要尽可能多，包围圈的数目要可能少。

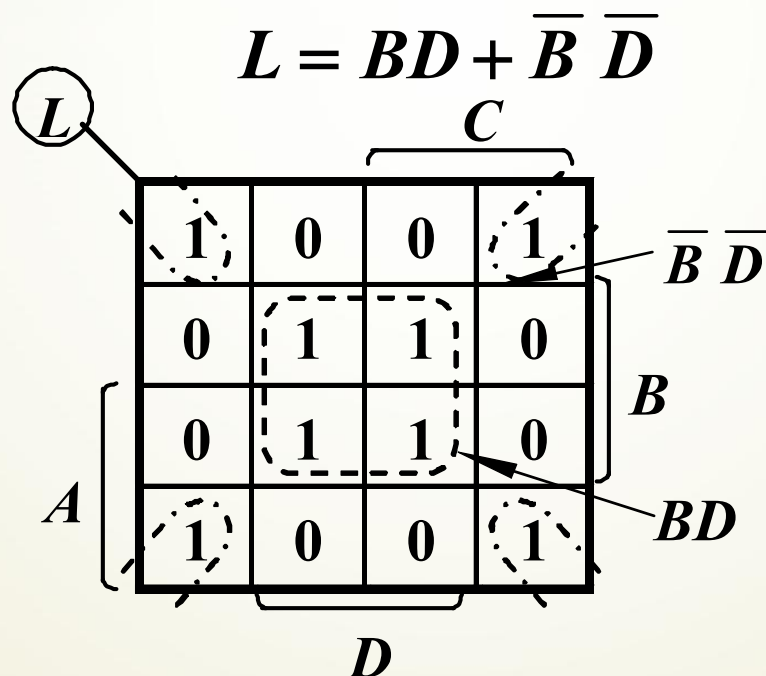


例 3：用卡诺图法化简下列逻辑函数

$$L(A, B, C, D) = \sum m(0, 2, 5, 7, 8, 10, 13, 15)$$

解：(1) 由 L 画出卡诺图

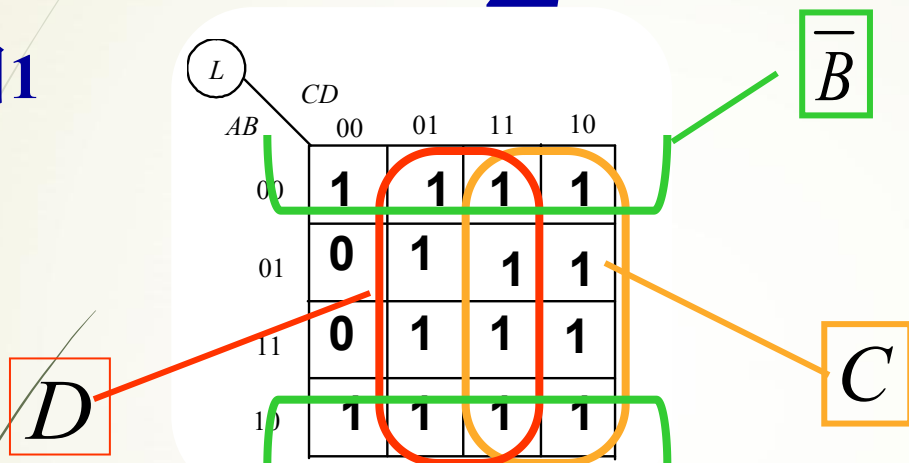
(2) 画包围圈合并最小项，得最简与-或表达式



例4：用卡诺图化简

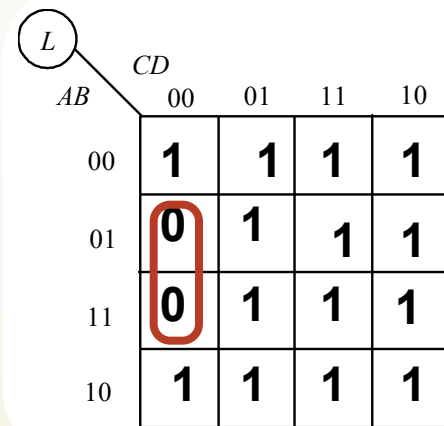
$$L(A, B, C, D) = \sum m(0 \sim 3, 5 \sim 7, 8 \sim 11, 13 \sim 15)$$

圈1



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

圈0



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

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

3、具有无关项的化简

(1) 什么叫无关项:

在真值表内对应于变量的某些取值下，函数的值可以是任意的，或者这些变量的取值根本不会出现，这些变量取值所对应的最小项称为无关项或任意项。

在含有无关项逻辑函数的卡诺图化简中，它的值可以取0或取1，具体取什么值，可以根据使函数尽量得到简化而定。

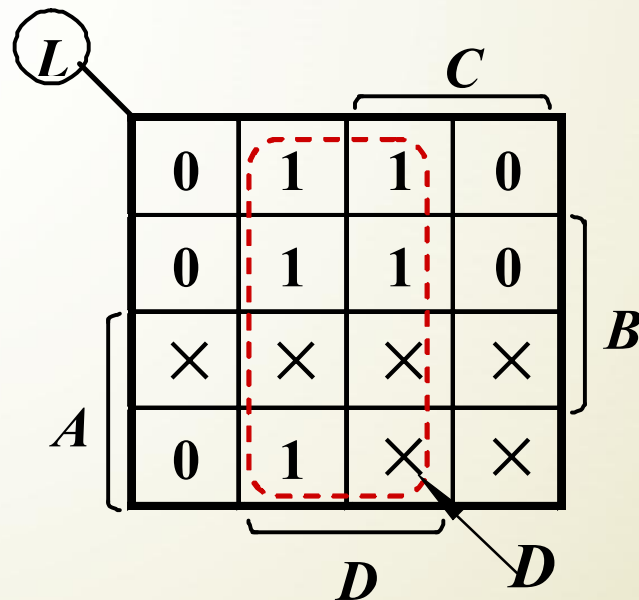
例5：试写出判断1位十进制数(以8421BCD码表示)奇偶性电路的输出逻辑函数的最小项表达式。当十进制数为奇数时，电路输出为1；否则输出为0。并求简化的逻辑函数表达式。

解：(1) 输入变量用 A 、 B 、 C 、 D 表示,输出用 L 表示。当输入为1、3、5、7、9时, L 为1；当输入为0、2、4、6、8时, L 为0。其余10~15六种为无关项，用 d 表示。

$$L = \sum m(1,3,5,7,9) + \sum d(10,11,12,13,14,15)$$

(2) 卡诺图化简

$$L = D$$



2.5 硬件描述语言Verilog HDL基础

2.5.1 VerilogHDL的基本结构

2.5.2 逻辑功能的仿真与测试

2.5.3 基本语法规则

2.5.4 数据类型

2.5.5 运算符及其优先级

2.5.6 Verilog内部的基本门级元件

2.5 硬件描述语言Verilog HDL基础

硬件描述语言HDL(Hardware Description Language)

类似于高级程序设计语言. 它是一种以文本形式来描述

数字系统硬件的结构和行为的语言, 用它可以表示

逻辑电路图、逻辑表达式, 复杂数字逻辑系统完成的

逻辑功能。HDL是高层次自动化设计的起点和基础。

计算机对HDL的处理：

逻辑仿真 是指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，以文本形式或时序波形图形式给出电路的输出。在仿真期间如发现设计中存在错误，可以对HDL描述进行及时的修改。

逻辑综合 是指从HDL描述的数字逻辑电路模型中导出电路基本元件列表以及元件之间的连接关系（常称为门级网表）的过程。类似对高级程序设计进行编译产生目标代码的过程。产生门级元件及其连接关系的数据库，根据这个数据库可以制作出集成电路或印刷电路板PCB。

2.5.1 Verilog程序的基本结构

模块是Verilog描述电路的基本单元。对数字电路建模时，用一个或多个模块。不同模块之间通过端口进行连接。

- 1、每个模块以关键词`module`开始，以`endmodule`结束。
- 2、每个模块先要进行端口的定义，并说明输入（`input`）和输出（`output`），然后对模块功能进行描述。
- 3、除了`endmodule`语句外，每个语句后必须有分号。
- 4、可以用`/* --- */`和`//.....`对程序的任何部分做注释。
- 5、逻辑功能的描述方式有三种不同风格：结构描述方式（门级描述方式）数据流描述方式，行为描述方式。

模块定义的一般语法结构如下：

module 模块名 (端口名1, 端口名2, 端口名3, …) ;

端口类型说明(input, outout, inout);

参数定义(可选);

数据类型定义(wire, reg等);

} 说明部分

实例化低层模块和基本门级元件;

连续赋值语句(assign);

过程块结构(initial和always)

行为描述语句;

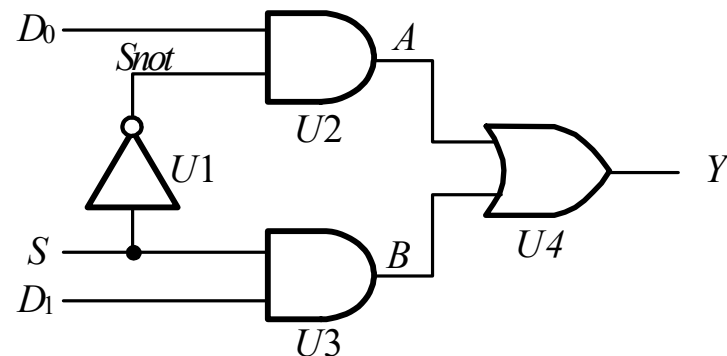
} 逻辑功能描述部分, 其顺序是任意的

endmodule

例 用门级描述建立模型

模块名

```
module mux2to1(D0, D1, S, Y );  
  input D0, D1, S; //定义输入信号  
  output Y; //定义输出信号  
  wire Snot, A, B ; //定义内部节点信号数据类型  
  //下面对电路的逻辑功能进行描述  
  not U1(Snot, S);  
  and U2(A, D0, Snot);  
  and U3(B, D1, S);  
  or U4(Y, A, B);  
endmodule
```



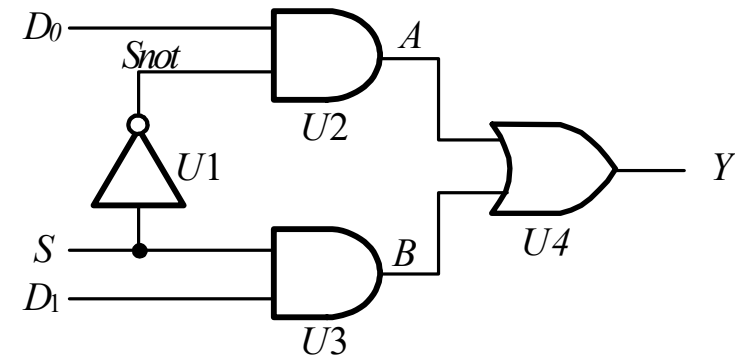
端口类型说明

数据类型说明

电路结构描述

例 用数据流描述方式建立模型

$$Y = D_0 \cdot \overline{S} + D_1 \cdot S$$



```
module mux2to1_df(  
  input D0, D1, S,  
  output wire Y  
);
```

修订版本Verilog-2001/2005标准的写法

数据类型
说明

//电路功能描述

```
  assign Y = (~S & D0) | (S & D1); //表达式左边Y必须是wire型  
endmodule
```

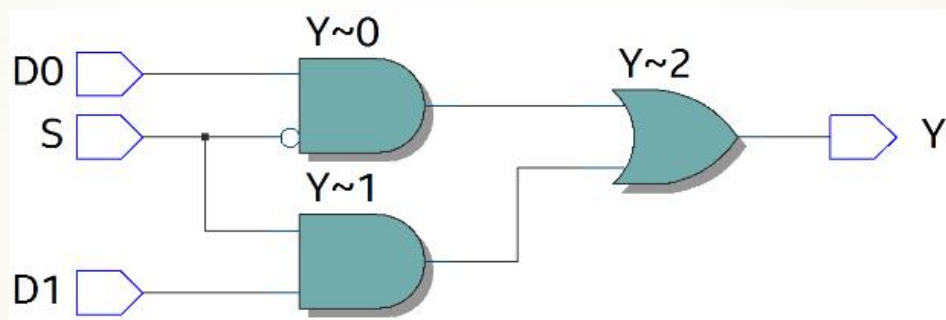
电路结构描述

注意，在assign语句中，左边变量的数据类型必须是wire型。

逻辑综合

将数据流描述的代码送到逻辑综合软件进行综合，得到如图2.5.5所示的逻辑图

这是IntelFPGA开发软件Quartus Prime 17.1进行逻辑综合的结果。



数据流描述代码逻辑综合的结果

2.5.2 逻辑功能的仿真与测试

逻辑电路的设计块完成后，就要测试这个设计块描述的逻辑功能是否正确。为此必须在输入端口加入测试信号，而从其输出端口检测其结果是否正确，这一过程常称为搭建测试平台。根据仿真软件的不同，搭建测试平台的方法也不同。

ModelSim软件为例，首先建立激励块
激励块大致由三部分组成：

- 一、实例引用被测试的模块（即设计块）。
- 二、给输入信号赋值，即激励信号。
- 三、指定测试结果的显示格式，并指定输出文件名。

2.5.2 逻辑功能的仿真与测试

激励块代码

```
//文件名: test_mux2to1_df.v
`timescale 1ns/1ns           //时间单位为 1 ns, 精确度为 1ns
module test_mux2to1_df;      //激励块, 没有端口列表
    reg PD0, PD1, PS;        //声明输入信号
    wire PY;                  //声明输出信号

//实例引用设计块
mux2to1_df t_mux (PD0, PD1, PS, PY); //按照端口位置连接
initial begin                  //激励信号
    PS = 0; PD1 = 0; PD0 = 0;    //语句1
    #5 PS = 0; PD1 = 0; PD0 = 1; //语句2 (#5 代表延迟 5 ns)
    #5 PS = 0; PD1 = 1; PD0 = 0; //语句3
    ....(这里省略了语句4-8, 写完整)
    #5 PS = 0; PD1 = 0; PD0 = 0; //语句9
    #5 $stop;                     //语句10
endmodule
```

2.5.2 逻辑功能的仿真与测试

激励块代码（续）

```
initial begin //输出部分
```

```
    $monitor($time, ":\tS = %b\tD1 = %b\tD0 = %b\tY =  
    %b",PS,PD1,PD0,PY);
```

```
    end
```

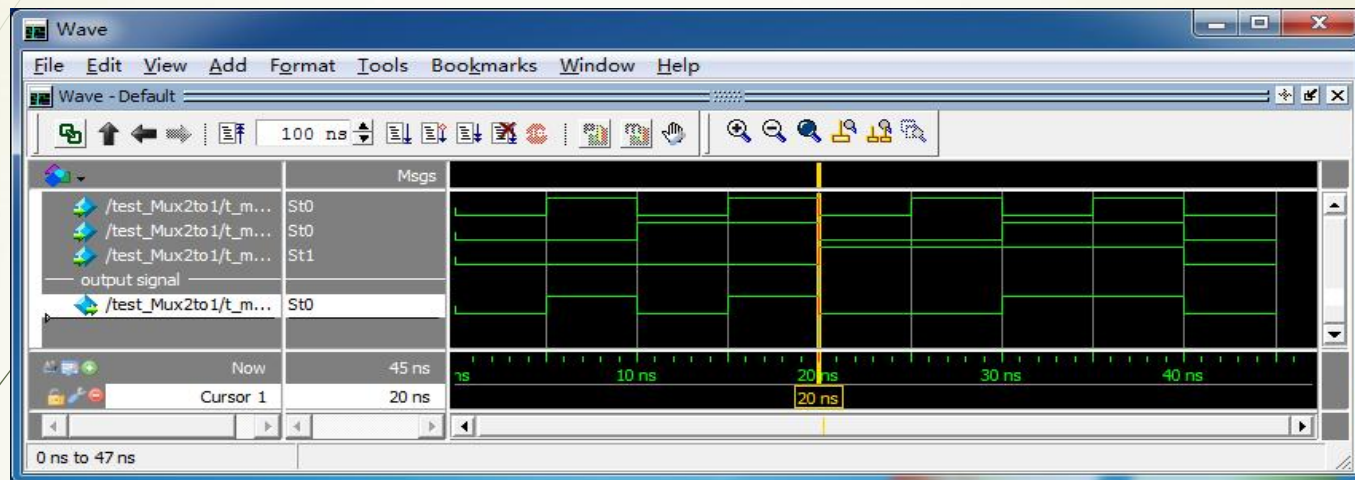
```
endmodule
```

其中，

- \$monitor、\$time和\$stop为系统任务
- \$monitor将信息以指定的格式输出到屏幕上，双引号内是要显示的内容，
- %b代表它后面的信号用二进制格式显示，
- \t 代表水平制表符，
- \$time 将返回当前的仿真时间，
- \$stop为停止仿真，但不退出仿真环境。附录A中将介绍。

2.5.2 逻辑功能的仿真与测试

建立工程项目，编译、仿真，输出波形或文本结果



The Transcript window shows the simulation results. The command `VSIM 3> run -all` was executed. The output shows the state of signals S, D1, D0, and Y at various time intervals. A note indicates that the simulation stopped at 45 ns due to a break in the module `test_Mux2to1` at line 20.

```
VSIM 3> run -all
#           0: S = 0 D1 = 0 D0 = 0 Y = 0
#           5: S = 0 D1 = 0 D0 = 1 Y = 1
#          10: S = 0 D1 = 1 D0 = 0 Y = 0
#          15: S = 0 D1 = 1 D0 = 1 Y = 1
#          20: S = 1 D1 = 0 D0 = 0 Y = 0
#          25: S = 1 D1 = 0 D0 = 1 Y = 0
#          30: S = 1 D1 = 1 D0 = 0 Y = 1
#          35: S = 1 D1 = 1 D0 = 1 Y = 1
#          40: S = 0 D1 = 0 D0 = 0 Y = 0
# ** Note: $stop      : D:/myIntelFPGA17/task_2.p71/sim/t_Mux2to1_df.v(20)
#   Time: 45 ns   Iteration: 0   Instance: /test_Mux2to1
# Break in Module test_Mux2to1 at D:/myIntelFPGA17/task_2.p71/sim/t_Mux2to1_df.v line 20
VSIM 4>
```

2.5.3 Verilog HDL的基本语法规则

为对数字电路进行描述（常称为建模），Verilog语言规定了一套完整的语法结构。

1. 间隔符: Verilog 的间隔符主要起分隔文本的作用，可以使文本错落有致，便于阅读与修改。

间隔符包括空格符（\b）、TAB 键（\t）、换行符（\n）及换页符。

2. 注释符: 注释是为了改善程序可读性, 在编译时不起作用。

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。

3. 标识符和关键词

标识符:给对象(如模块名、电路的输入与输出端口、变量等)取名所用的字符串。以英文字母或下划线开始

如, `clk`、`counter8`、`_net`、`bus_A`。

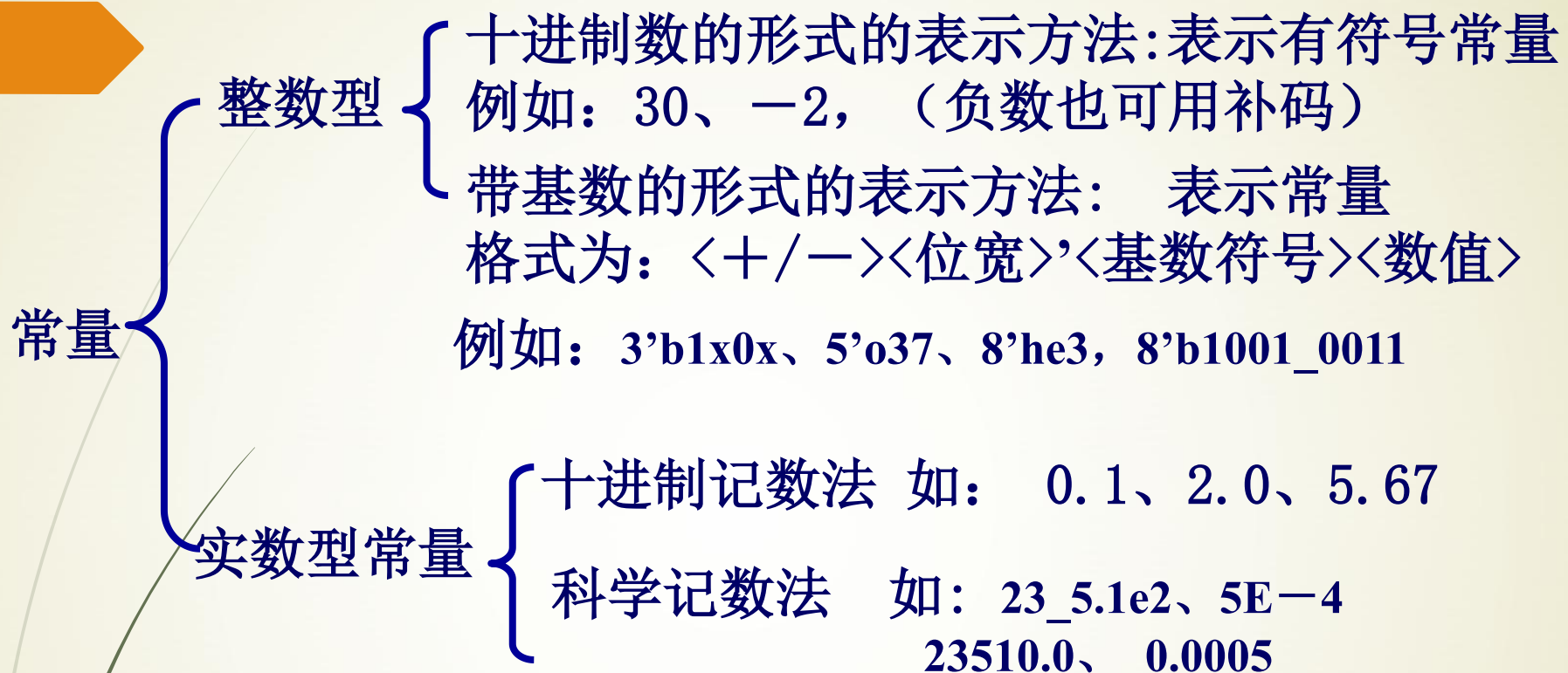
关键词:是Verilog语言本身规定的特殊字符串,用来定义语言的结构。例如, `module`、`endmodule`、`input`、`output`、`wire`、`reg`、`and`等都是关键词。关键词都是小写,关键词不能作为标识符使用。

4. 逻辑值集合

为了表示数字逻辑电路的逻辑状态, 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

5. 常量及其表示



Verilog允许用参数定义语句, 定义一个标识符来代表一个常量, 称为符号常量。定义的格式为:

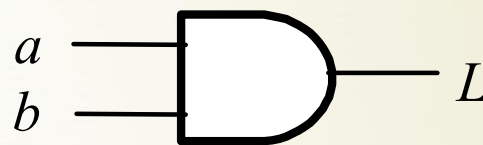
parameter 参数名1=常量表达式1, 参数名2=常量表达式2, ...; 如 **parameter** BIT=1, BYTE=8, PI=3.14;

6. 字符串: 字符串是双撇号内的字符序列

2.5.4 数据类型

1、**线网类型**:是指输出始终根据输入的变化而更新其值的变量,它一般指的是硬件电路中的各种物理连接.

例:网络型变量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。



常用的网络类型由关键词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;
变量宽度

例:wire L; //将上述电路的输出信号L声明为网络型变量
wire [7:0] data bus; //声明8-bit宽的网络型总线变量

2、变量类型（之前称寄存器型）

它对应的是具有状态保持作用的电路元件, 如触发器寄存器。

寄存器型变量只能在initial或always内部被赋值。

4种变量类型

抽象描述,
不对应具
体硬件

变量类型	功能说明
reg	用于行为描述, 表示存储逻辑值变量
integer	32位带符号的整数型变量
real	64位带符号的实数型变量,
time	64位无符号的时间变量

例: `reg clock;` //定义一个1位寄存器变量
`reg [3:0] counter;` //定义一个4位寄存器变量

2.5.5 运算符及其优先级

1. 运算符

运算符分为算术运算符、逻辑运算符、关系运算符、移位运算符等

类型	符号	功能说明	类型	符号	功能说明
算术运算符	$+$ $-$ $-$ 2的补码 $*$ $/$ $\%$ $**$	二进制加 二进制减 2的补码 二进制乘 二进制除 取余 指数幂	关系运算符 (双目运算符)	$>$ $<$ $>=$ $<=$ $==$ $!=$ $===$ $!==$	大于 小于 大于或等于 小于或等于 相等 不相等 Case中的相等 Case中的不等
位运算符 (双目运算符)	$\sim$ $\&$ $ $ $\wedge$ $\wedge \sim$ 或 $\sim \wedge$	按位取反 按位与 按位或 按位异或 按位同或	缩位运算符 (单目运算符)	$\&$ $\sim \&$ $ $ $\sim $ $\wedge$ $\wedge \sim$ 或 $\sim \wedge$	缩位与 缩位与非 缩位或 缩位或非 缩位异或 缩位同或
逻辑运算符 (双目运算符)	$!$ $\&\&$ $\parallel$	逻辑非 逻辑与 逻辑或	移位运算符 (双目运算符)	$>>$ $<<$ $>>>$ $<<<$	逻辑右移 逻辑左移 算术右移 算术左移
位拼接运算符	$\{\}$ $\{\{\}$	将多个操作数 拼接成为一个 操作数	条件运算符 (三目运算符)	$?:$	根据条件表达式是否成立, 选择表达式

位拼接运算符

作用是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，进行运算操作。

设 $A=1'b1$, $B=2'b10$, $C=2'b00$

则 $\{B,C\}=4'b1000$

$\{A,B[1],C[0]\}=3'b110$

$\{A,B,C,3'b101\}=8'b11000101$ 。

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， $\{4\{A\}\}=4'b1111$, $\{2\{A\},2\{B\},C\}=8'b11101000$ 。

位运算符与缩位运算的比较

A: 4'b1010 、

B: 4'b1111,

位运算	$\sim A = 0101$ $\sim B = 0000$	$A \& B =$ 1010	$A B =$ 1111	$A \wedge B =$ 0101	$A \sim \wedge B =$ 1010
缩位运算	$\& A = 1 \&$ $0 \& 1 \& 0 = 0$	$\sim \& A = 1$ $\& B = 1$	$ A = 1$ $\sim B = 0$	$\wedge A = 0$ $\wedge B = 0$	$\sim \wedge A = 1$ $\sim \wedge B = 1$

2. 运算符的优先级

优先级的顺序从下向上依次增加。

类型	符号	优先级别	
取反	! ~ -(求2的补码)	最高优先级	
算术	* / + -		
移位	>> << >>> <<<		
关系	< <= > >=		
等于	== != === !==		
缩位	& ~& ^ ^~ ~		
逻辑	&& 		
条件	?:	最低优先级	

条件运算符

是三目运算符，运算时根据条件表达式的值选择表达式。

一般用法：

`condition_expr?expr1:expr2;`

首先计算第一个操作数`condition_expr`的值：

- 结果为逻辑1，则选择第二个操作数`expr1`的值作为结果返回。
- 结果为逻辑0，选择第三个操作数`expr2`的值作为结果返回。

2.5.6 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

三态门

多输出门

多输入门

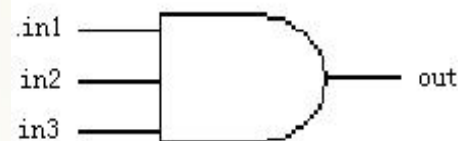
元件符号	功能说明	元件符号	功能说明
and	多输入端的与门	nand	多输入端的与非门
or	多输入端的或门	nor	多输入端的或非门
xor	多输入端的异或门	xnor	多输入端的异或非门
buf	多输出端的缓冲器	not	多输出端的反相器
bufif1	控制信号高电平有效的三态缓冲器	notif1	控制信号高电平有效的三态反相器
bufif0	控制信号低电平有效的三态缓冲器	notif0	控制信号低电平有效的三态反相器

1、多输入门

只允许有一个输出，但可以有多多个输入。

调用名

and A1 (out, in1, in2, in3) ;



and真值表

and		输入1			
		0	1	x	z
输入 2	0	0	0	0	0
	1	0	1	x	x
	x	0	x	x	x
	z	0	x	x	x

x- 不确定状态 z- 高阻态

如果有一个输入为x或者z，
则输出为x。

or真值表

or		输入1			
		0	1	x	z
输入 2	0	0	1	x	x
	1	1	1	1	1
	x	x	1	X	X
	z	x	1	X	X

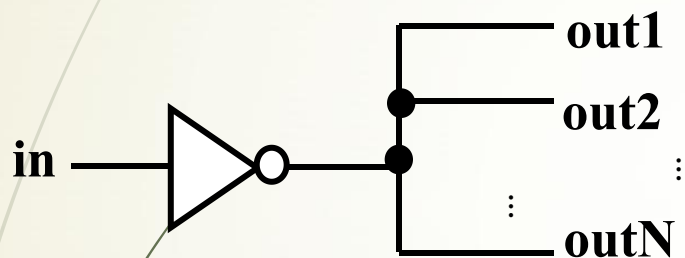
xor真值表

xor		输入1			
		0	1	x	z
输入 2	0	0	1	x	x
	1	1	0	x	x
	x	x	x	x	x
	z	x	x	x	x

2、多输出门

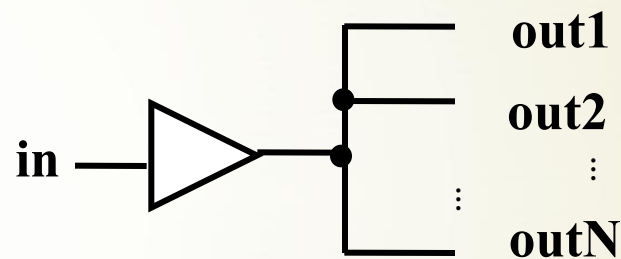
允许有多个输出，但只有一个输入。

not N1 (out1, out2, ..., in); buf B1 (out1, out2, ..., in);



not真值表

not	输 入			
	0	1	x	z
输 出	1	0	x	x



buf真值表

buf	输 入			
	0	1	x	z
输 出	0	1	x	x

3、三态门

有一个输出、一个数据输入和一个输入控制。
如果输入控制信号无效，则三态门的输出为高阻态 z 。



图 4.6.3 三态门元件模型
(a) bufif1 (b) notif1

bufif1真值表

bufif1		控制输入			
		0	1	x	z
数据输入	0	z	0	0/z	0/z
	1	z	1	1/z	1/z
	x	z	x	x	x
	z	z	x	x	x

notif1真值表

notif1		控制输入			
		0	1	x	z
数据输入	0	z	1	1/z	1/z
	1	z	0	0/z	0/z
	x	z	x	x	x
	z	z	x	x	x