## 逻辑门

逻辑门（Logic Gates)是在[集成电路](https://baike.baidu.com/item/%E9%9B%86%E6%88%90%E7%94%B5%E8%B7%AF/108211" \t "https://baike.baidu.com/item/%E9%80%BB%E8%BE%91%E9%97%A8/_blank)(Integrated Circuit)上的基本[组件](https://baike.baidu.com/item/%E7%BB%84%E4%BB%B6/6902128" \t "https://baike.baidu.com/item/%E9%80%BB%E8%BE%91%E9%97%A8/_blank)。简单的逻辑门可由晶体管组成。这些晶体管的组合可以使代表两种信号的高[低电平](https://baike.baidu.com/item/%E4%BD%8E%E7%94%B5%E5%B9%B3/6946314" \t "https://baike.baidu.com/item/%E9%80%BB%E8%BE%91%E9%97%A8/_blank)在通过它们之后产生高电平或者低电平的信号。高、低电平可以分别代表[逻辑](https://baike.baidu.com/item/%E9%80%BB%E8%BE%91/543" \t "https://baike.baidu.com/item/%E9%80%BB%E8%BE%91%E9%97%A8/_blank)上的“真”与“假”或[二进制](https://baike.baidu.com/item/%E4%BA%8C%E8%BF%9B%E5%88%B6/361457" \t "https://baike.baidu.com/item/%E9%80%BB%E8%BE%91%E9%97%A8/_blank)当中的1和0，从而实现[逻辑运算](https://baike.baidu.com/item/%E9%80%BB%E8%BE%91%E8%BF%90%E7%AE%97/7224729" \t "https://baike.baidu.com/item/%E9%80%BB%E8%BE%91%E9%97%A8/_blank)。

### 概念

#### 基本单元

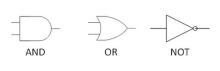
[IMG_256](https://baike.baidu.com/pic/é»è¾é¨/5141155/0/a8ad94138dcc4ac3f6039e24?fr=lemma%26ct=single)

逻辑门又称“数字逻辑电路基本单元”。执行“或”、“与”、“非”、“或非”、“与非”等[逻辑运算](https://baike.baidu.com/item/%E9%80%BB%E8%BE%91%E8%BF%90%E7%AE%97" \t "https://baike.baidu.com/item/%E9%80%BB%E8%BE%91%E9%97%A8/_blank)的电路。任何复杂的[逻辑电路](https://baike.baidu.com/item/%E9%80%BB%E8%BE%91%E7%94%B5%E8%B7%AF" \t "https://baike.baidu.com/item/%E9%80%BB%E8%BE%91%E9%97%A8/_blank)都可由这些逻辑门组成。广泛用于计算机、通信、控制和数字化仪表。

#### 作用

通过控制高、低电平（分别代表逻辑上的“真”与“假”或二进制当中的“1”和“0”），从而实现逻辑运算。

### 逻辑门的种类

[](https://baike.baidu.com/pic/é»è¾é¨/5141155/0/a2cc7cd98d1001e961ea3d1fbf0e7bec55e797cf?fr=lemma%26ct=single)

常见的逻辑门包括“与”门，“或”门，“非”门，“[异或](https://baike.baidu.com/item/%E5%BC%82%E6%88%96" \t "https://baike.baidu.com/item/%E9%80%BB%E8%BE%91%E9%97%A8/_blank)”（也称：互斥或）等等。逻辑门可以组合使用实现更为复杂的逻辑运算。

#### 或门

或门（英文：Or gate）又称或电路。如果几个条件中，只要有一个条件得到满足，某事件就会发生，这种关系叫做“或”逻辑关系。具有“或”逻辑关系的电路叫做或门。或门有多个输入端，一个输出端，多输入或门可由多个2输入或门构成。只要输入中有一个为高电平时（逻辑1），输出就为高电平（逻辑1）；只有当所有的输入全为低电平时，输出才为低电平。

#### **与门**

与门（英语：AND gate）又称“与电路”。是执行“与”运算的基本逻辑门电路。有多个输入端，一个输出端。当所有的输入同时为高电平（逻辑1）时，输出才为高电平，否则输出为低电平（逻辑0）。

#### **非门**

非门（英文：NOT gate）又称反相器，是[逻辑电路](https://baike.baidu.com/item/%E9%80%BB%E8%BE%91%E7%94%B5%E8%B7%AF" \t "https://baike.baidu.com/item/%E9%80%BB%E8%BE%91%E9%97%A8/_blank)的基本单元，非门有一个输入和一个输出端。逻辑符号中输出端的圆圈代表反相的意思。当其输入端为高电平（逻辑1）时输出端为低电平（逻辑0），当其输入端为低电平时输出端为高电平。也就是说，输入端和输出端的电平状态总是反相的。

#### **与非门**

由与门与非门组合而成。

#### **或非门**

由或门和非门组合而成。

其中，非门和或非门在数字电路中较为常见。

### 符号表

设其中A和B为输入变量

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 类型 | 又名/短释 | 逻辑函数表示 | 真值表 | |
| **NOT** | 非门╱反相器：  逆转输入的高低状态。 | A' | **输入**  A  0  1 | **输出**  An  1  0 |
| **AND** | 与门╱且门：  所有输入为高时，  才会有输出高。 | A\*B | **输入**  A B  0 0  0 1  1 0  1 1 | **输出**  AandB  0  0  0  1 |
| **NAND** | 与非门：  与与门相反。  所有输入为高时，  才会有输出低。 | (A\*B)' | **输入**  A B  0 0  0 1  1 0  1 1 | **输出**  AnandB  1  1  1  0 |
| **OR** | 或门：  所有输入为低时，  才会有输出低。 | A + B | **输入**  A B  0 0  0 1  1 0  1 1 | **输出**  AorB  0  1  1  1 |
| **NOR** | 或非门：  与或门相反。  所有输入为低时，  才会有输出高。 | (A + B)' | **输入**  A B  0 0  0 1  1 0  1 1 | **输出**  AnorB  1  0  0  0 |
| **XOR** | 异或门：  输入相同时输出为低，  否则为高。 | A'\*B + A\*B' | **输入**  A B  0 0  0 1  1 0  1 1 | **输出**  AxorB  0  1  1  0 |
| **XNOR** | 同或门：  与异或门相反。  输入相同时输出为高，  否则为低。 | A\*B + A'\*B' | **输入**  A B  0 0  0 1  1 0  1 1 | **输出**  AxnorB  1  0  0  1 |

### 逻辑门电路

把基本逻辑运算的电子电路称之为[逻辑门电路](https://baike.baidu.com/item/%E9%80%BB%E8%BE%91%E9%97%A8%E7%94%B5%E8%B7%AF/6008840" \t "https://baike.baidu.com/item/%E9%80%BB%E8%BE%91%E9%97%A8/_blank)。在数字电路关系应用中，逻辑门电路中的门代表着基本逻辑关系的电路。

通过对逻辑门内部电路的源器件的差异，我们可以将其分为三大类，比较常用的就是CMOS 的逻辑门电路。

这种CMOS 逻辑门电路具备良好的应用效益，首先其功耗比较低，具备较低的应用成本，其电源电压的范围比较宽，逻辑度比较高，具备较强的抗干扰能力，其输入阻抗比较高。MOS 门电路是由一系列的单极型MOS 管构成，其具备比较简单的制造工艺，其功耗水平比较低，具备较高的集成度，其抗干扰能力强，比较适合进行大规模集成电路的应用。

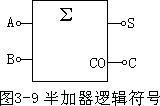
在实践过程中，MOS 门电路按照其MOS 管的应用不同，进行不同类型的划分。CMOS 电路的应用优点比较特殊，其静态功耗比较低，抗干扰能力强，工作具备较高的稳定性，其开关速度也是比较高的，因此其应用性比较广泛。

在数字电路应用中，逻辑门电路是一种基本的逻辑元件。逻辑门的中就是一种开关，在一定条件的建立下，其决定信号的通过或者不通过。在实际运作中，我们可以看到门电路输入及其输出存在密切的因果关系，我们把门电路称之为逻辑门电路，其基本逻辑关系是非、或、与关系。 [1]

## 半加器

半加器电路是指对两个输入[数据位](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E4%BD%8D/3441892" \t "https://baike.baidu.com/item/%E5%8D%8A%E5%8A%A0%E5%99%A8/_blank)相加，输出一个结果位和[进位](https://baike.baidu.com/item/%E8%BF%9B%E4%BD%8D/5989952" \t "https://baike.baidu.com/item/%E5%8D%8A%E5%8A%A0%E5%99%A8/_blank)，没有进位输入的[加法器](https://baike.baidu.com/item/%E5%8A%A0%E6%B3%95%E5%99%A8/9374198" \t "https://baike.baidu.com/item/%E5%8D%8A%E5%8A%A0%E5%99%A8/_blank)电路。 是实现两个一位二进制数的加法运算电路。

### 简介

[](https://baike.baidu.com/pic/%E5%8D%8A%E5%8A%A0%E5%99%A8/7653973/0/91ae68c691298f089c163d7e?fr=lemma%26ct=single)

半加器是实现两个一位二进制数加法运算的器件。它具有两个输入端(被加数A和加数B)及输出端Y。 [1]

是数据输入被加数A、加数B，[数据输出](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E8%BE%93%E5%87%BA" \t "https://baike.baidu.com/item/%E5%8D%8A%E5%8A%A0%E5%99%A8/_blank)S和数(半加和)、[进位](https://baike.baidu.com/item/%E8%BF%9B%E4%BD%8D" \t "https://baike.baidu.com/item/%E5%8D%8A%E5%8A%A0%E5%99%A8/_blank)C0。

A和B是相加的两个数，S是半加和数，C是进位数。

所谓半加就是不考虑进位的加法，它的真值表如下 (见表)： [2]

|  |  |  |  |
| --- | --- | --- | --- |
| 被加数A | 加数B | 和数S | 进位数C |
| 0  0  1  1 | 0  1  0  1 | 0  1  1  0 | 0  0  0  1 |

表中：

IMG_257

IMG_258

### 分析

由逻辑状态表可写出逻辑式，按组合数字电路的分析方法和步骤进行。

1．写出输出逻辑表达式

2．列出真值表

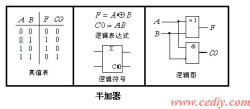
半加器的真值表见下表。表中两个输入是加数A0和B0，输出有一个是和S0，另一个是[进位](https://baike.baidu.com/item/%E8%BF%9B%E4%BD%8D" \t "https://baike.baidu.com/item/%E5%8D%8A%E5%8A%A0%E5%99%A8/_blank)C0。

|  |  |  |  |
| --- | --- | --- | --- |
| 输入 | | 输出 | |
| A | B | C | S |
| 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 |

3．给出逻辑说明

半加器是实现两个一位二进制码相加的电路，因此只能用于两个[二进制码](https://baike.baidu.com/item/%E4%BA%8C%E8%BF%9B%E5%88%B6%E7%A0%81" \t "https://baike.baidu.com/item/%E5%8D%8A%E5%8A%A0%E5%99%A8/_blank)最低位的相加。

因为高位二进制码相加时，有可能出现

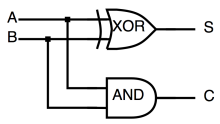
[](https://baike.baidu.com/pic/%E5%8D%8A%E5%8A%A0%E5%99%A8/7653973/0/5d212aa803a5a6aacb130cb2?fr=lemma%26ct=single)

低位的[进位](https://baike.baidu.com/item/%E8%BF%9B%E4%BD%8D" \t "https://baike.baidu.com/item/%E5%8D%8A%E5%8A%A0%E5%99%A8/_blank)，因此两个加数相加时还要计算低位的进位，需要比半加器多进行一次相加运算。

能计算低位进位的两个一位二进制码的相加电路，即为[全加器](https://baike.baidu.com/item/%E5%85%A8%E5%8A%A0%E5%99%A8" \t "https://baike.baidu.com/item/%E5%8D%8A%E5%8A%A0%E5%99%A8/_blank)。

### 实现

半加器不考虑低位向本位的[进位](https://baike.baidu.com/item/%E8%BF%9B%E4%BD%8D" \t "https://baike.baidu.com/item/%E5%8D%8A%E5%8A%A0%E5%99%A8/_blank)，因此它不属于时序逻辑电路，有两个输入端和两个输出。

[](https://baike.baidu.com/pic/%E5%8D%8A%E5%8A%A0%E5%99%A8/7653973/0/96dda144ad345982c60529400df431adcaef8451?fr=lemma%26ct=single)半加器的逻辑实现

设加数（输入端）为A、B ；和为S ；向高位的进位为Ci+1

**逻辑表达式**：

IMG_261

；

IMG_262

。

### 输入和输出

**半加器**有两个输入和两个输出，输入可以标识为A、B，输出通常标识为求和（**S**um）和进位（**C**arry）。输入经异或（XOR）运算后即为S，经和（AND）运算后即为C。

**半加器**有两个二进制的输入，其将输入的值相加，并输出结果到和（**S**um）和进位（**C**arry）。半加器虽能产生进位值，但半加器本身并不能处理进位值。

### 与全加器区别

半加器没有接收进位的输入端，全加器有进位输入端，在将两个多位二进制数相加时，除了最低位外，每一位都要考虑来自低位的进位，半加器则不用考虑，只需要考虑两个输入端相加即可。

## 全加器

全加器英语名称为full-adder，是用[门电路](https://baike.baidu.com/item/%E9%97%A8%E7%94%B5%E8%B7%AF/10796427" \t "https://baike.baidu.com/item/%E5%85%A8%E5%8A%A0%E5%99%A8/_blank)实现两个二进制数相加并求出和的组合线路，称为一位全加器。一位全加器可以处理低位进位，并输出本位加法进位。多个一位全加器进行级联可以得到多位全加器。常用二进制四位全加器74LS283。

### 应用举例

#### 真值表

一位全加器的真值表如下图，其中Ai为被加数，Bi为加数，相邻低位来的进位数为Ci-1，输出本位和为Si。向相邻高位进位数为Ci [1]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 输入 | | | 输出 | |
| Ci-1 | Ai | Bi | Si | Ci |
| 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 |

### 描述

一位全加器的表达式如下：

Si=Ai⊕Bi⊕Ci-1

IMG_256

第二个表达式也可用一个异或门来代替或门对其中两个输入信号进行求和：

IMG_257

[硬件](https://baike.baidu.com/item/%E7%A1%AC%E4%BB%B6" \t "https://baike.baidu.com/item/%E5%85%A8%E5%8A%A0%E5%99%A8/_blank)描述语言Verilog 对一位全加器的三种建模方法：

### 结构化描述方式（Verilog）

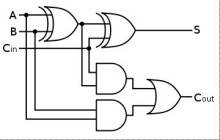
module FA\_struct (A, B, Cin, Sum, Count);

input A;

input B;

input Cin;

output Sum;

[](https://baike.baidu.com/pic/%E5%85%A8%E5%8A%A0%E5%99%A8/9791810/0/08f790529822720e825e758a74cb0a46f21fab2a?fr=lemma%26ct=single)

output Count;

wire S1, T1, T2, T3;

// -- statements -- //

xor x1 (S1, A, B);

xor x2 (Sum, S1, Cin);

and A1 (T3, A, B );

and A2 (T2, B, Cin);

and A3 (T1, A, Cin);

or O1 (Count, T1, T2, T3 );

endmodule

该实例显示了一个全加器由两个异或门、三个与门、一个或门构成 (或者可以理解为两个半加器与一个或门的组合)。S1、T1、T2、T3则是门与门之间的连线。代码显示了用纯结构的建模方式，其中xor 、and、or 是Verilog HDL 内置的门器件。以 xor x1 (S1, A, B) 该例化语句为例：xor 表明调用一个内置的异或门，器件名称xor ，代码实例化名x1（类似原理图输入方式）。括号内的S1，A，B 表明该器件管脚的实际连接线（信号）的名称，其中 A、B是输入，S1是输出。

### 数据流描述方式

`timescale 1ns/100ps

module FA\_flow(A,B,Cin,Sum,Count);

input A,B,Cin;

output Sum, Count;

wire S1,T1,T2,T3;

assign # 2 S1 = A ^ B;

assign # 2 Sum = S1 ^ Cin;

assign #2 T3 = A & B;

assign #2 T1 = A & Cin;

assign #2 T2 = B & Cin ;

assign #2 Count=T1 | T2 | T3;

endmodule

注意在各assign 语句之间，是[并行执行](https://baike.baidu.com/item/%E5%B9%B6%E8%A1%8C%E6%89%A7%E8%A1%8C" \t "https://baike.baidu.com/item/%E5%85%A8%E5%8A%A0%E5%99%A8/_blank)的，即各语句的执行与语句之间的顺序无关。如上，当A有个变化时，S1、T3、T1 将同时变化，S1的变化又会造成Sum的变化。

### 行为描述方式

module FA\_behav(A, B, Cin, Sum, Cout );

input A,B,Cin;

output Sum,Cout;

reg Sum, Cout;

reg T1,T2,T3;

always@ ( A or B or Cin )

begin

Sum = (A ^ B) ^ Cin ;

T1 = A & Cin;

T2 = B & Cin ;

T3 = A & B;

Cout = (T1| T2) | T3;

end

endmodule

### 全加器的VHDL描述

library ieee;

use ieee.std\_logic\_1164.all;

Entity full\_add is

port(a,b,c:in std\_logic;

sum,count:out std\_logic);

end entity full\_add;

architecture art of full\_add is

begin

process(a,b,c) is

begin

if(a='0' and b='0' and c='0') then

sum<='0';count<='0';

elsif(a='1' and b ='0' and c='0') then

sum<='1';count<='0';

elsif(a='0' and b='1' and c= '0') then

sum<='1';count<='0';

elsif(a='1' and b='1' and c= '0') then

sum<='0';count<='1';

elsif(a='0' and b='0' and c= '1') then

sum<='1';count<='0';

elsif(a='1' and b='0' and c= '1') then

sum<='0';count<='1';

elsif(a='0' and b='1' and c= '1') then

sum<='0';count<='1';

else

sum<='1';count<='1';

end if;

end process;

end architecture art;