Verilog HDL

第一部分 硬件描述语言Verilog HDL的基本概念和基本架构

1.硬件描述语言（HDL）概述

- 描述电子电路（特别是数字电路）的语言
- 可从门级、寄存器传输级、行为级进行描述
- 常见的语言：Verilog、VHDL

Verilog HDL语言最初是在1983年由Gateway Design Automation 公司为其模拟器产品开发的硬件建模语言。在1995年成为IEEE标准，称为IEEE Std1364－1995。目前受到了广泛的应用，还在不断的发展和完善中。

2.门原语

Verilog语言提供已经设计好的门，称为门原语（primitive，共12个），这些门可直接调用，不用再对其进行功能描述。

3.门原语调用格式

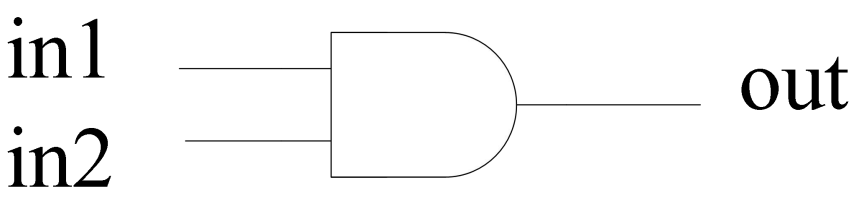
门原语名　实例名（端口连接）

注：实例名可省略，端口连接采用输出在前，输入在后。

门原语（1-6）

and （与）	or（或）	xor（异或）
nand（与非）	nor（或非）	xnor（同或）

and (out, in1, in2);

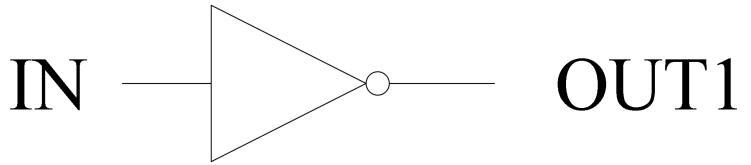


端口连接中第一个是输出，其余是输入，输入个数不限。

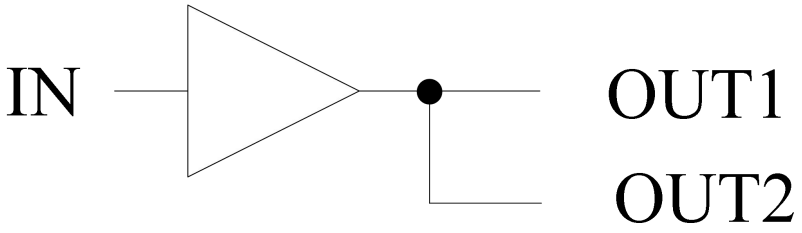
门原语（7,8）

not （非门）
buf（缓冲器）

not (OUT1, IN);



buf b1_2out(OUT1, OUT2, IN);

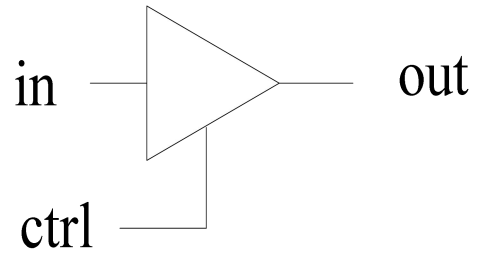


端口列表中前面是输出，最后一个是输入，输出个数不限。

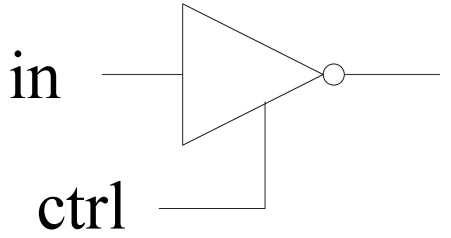
门原语（9-12）

bufif1 (控制端1有效缓冲器)	notif1 （控制端1有效非门）
bufif0 (控制端0有效缓冲器)	notif0 （控制端0有效非门）

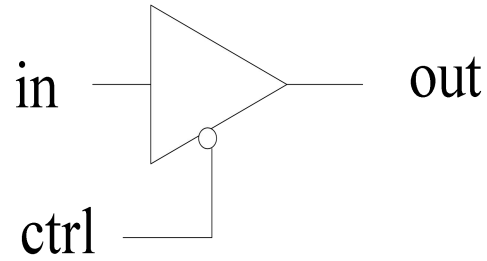
bufif1 b1 (out, in, ctrl);



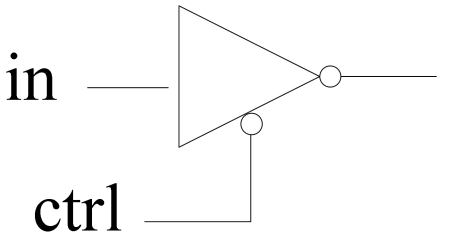
notif1 n1 (out, in, ctrl);



bufif0 b0 (out, in, ctrl);



notif0 n0 (out, in, ctrl);



端口列表中前面是输出，中间是输入，最后是使能端，输出个数不限。

第二部分 使用Verilog HDL语言设计常见的MSI组合电路

模块（module）

- 模块是Verilog的基本描述单位，用于描述某个设计的功能或结构及其与其它模块通信的外部接口

- 模块中，可以采用下述方式描述一个设计：
  - 数据流方式
  - 行为方式
  - 结构方式
  - 上述方式的混合

- 每个模块先要进行端口的定义，并说明输入（input)、输出（output)和双向（inout)，然后对模块功能进行描述。

•模块（module）

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, output, inout);
        参数定义(可选)；
        数据类型定义(wire, reg等)；

        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；
endmodule
```

说明部分

逻辑功能描述部分，其顺序是任意的

•模块（module）

几种描述方式小结：

•结构描述（门级描述）方式：

一般使用Primitive（内部元件）、自定义的下层模块对电路描述。主要用于层次化设计中。

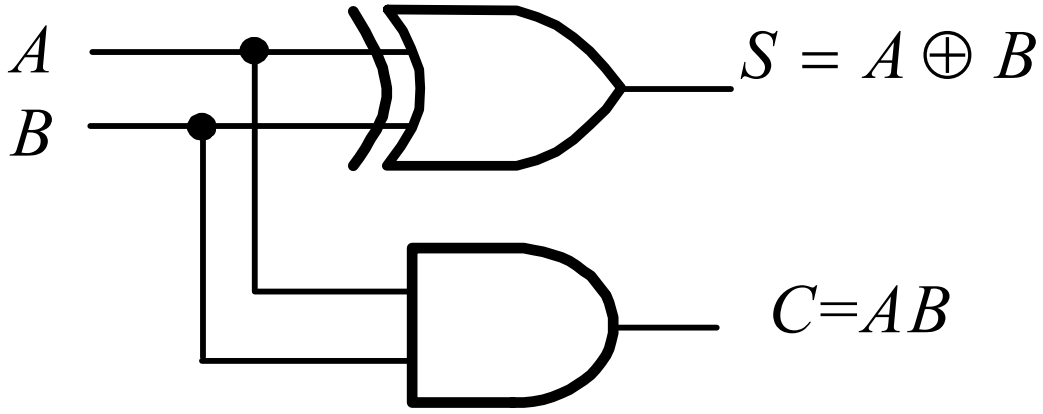
•数据流描述方式：

一般使用assign语句描述，主要用于对组合逻辑电路建模。

•行为描述方式：

一般使用下述语句描述，可以对组合、时序逻辑电路建模。

1）initial 语句
2）always 语句



半加器逻辑图

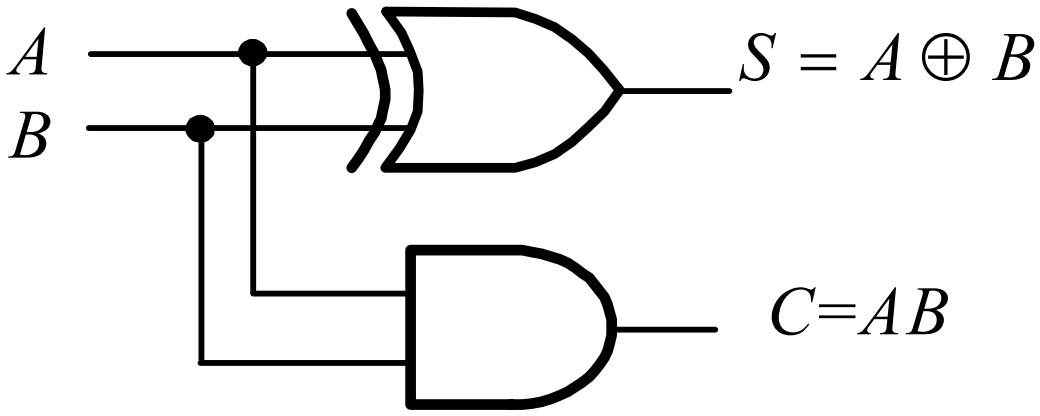
模块名

```
/* Gate-level description of a half adder */
module HalfAdder_GL(A, B, S, C);
   input    A, B;     //输入端口声明
   output   S, C;     //输出端口声明
   wire     A, B, S, C;

   xor X1 (S, A, B );
   and A1 (C, A, B);
endmodule
```

端口类型说明

数据类型说明

功能描述

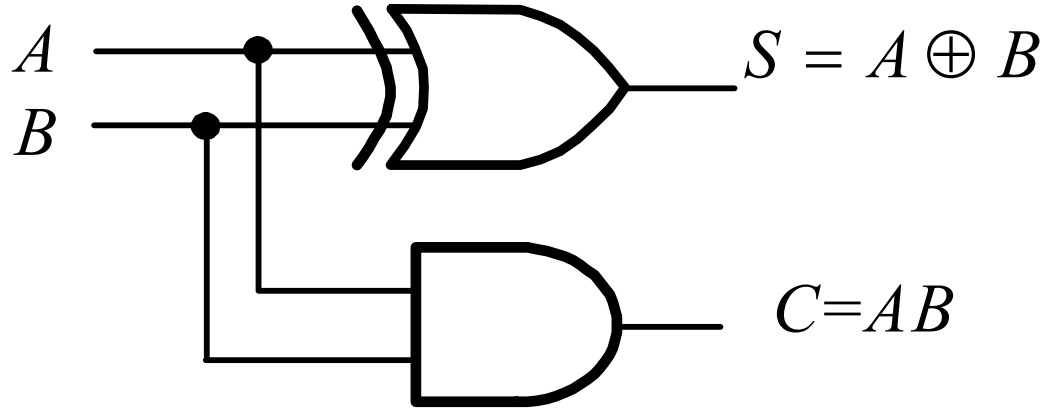


半加器逻辑图

```
/* Dataflow description of a half adder */
module HalfAdder_DF(A, B, S, C);
  input   A, B;
  output  S, C;
  wire    A, B, S, C;
  assign   S = A ^ B;    //赋值
  assign   C = A & B; //赋值
 endmodule
```

```
/* Gate-level description of a ha
module HalfAdder_GL(A, B, S, C);
  input   A, B;  //输入端口声明
  output  S, C; //输出端口声明
  wire    A, B, S, C;

  xor X1 (S, A, B );
  and A1 (C, A, B);
endmodule
```

```
/* Behavioral description of a half adder */
module HalfAdder_BH(A, B, S, C);
  input   A, B;
  output  S, C;
  reg     S, C;  //声明端口数据类型为寄存器
 always @(A or B)    // always语句：循环重复执行
    begin
      S = A ^ B;  //用过程赋值语句描述逻辑功能
      C = A & B;
    end
endmodule
```



半加器逻辑图

不同时期的语法规则有所不同!

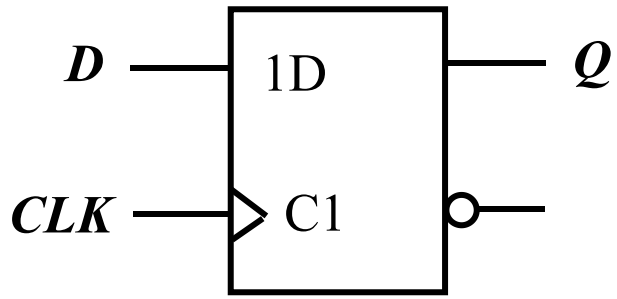
```
// module half_adder (S, C, x, y);   1995版
// output   S, C;
// input    x, y;

module half_adder (output S, C, input x, y);   2001、2005版
// Instantiate primitive gates
  xor (S, x, y);
  and (C, x, y);
endmodule
```

第三部分 使用Verilog HDL语言设计常用触发器

一、行为建模描述DFF

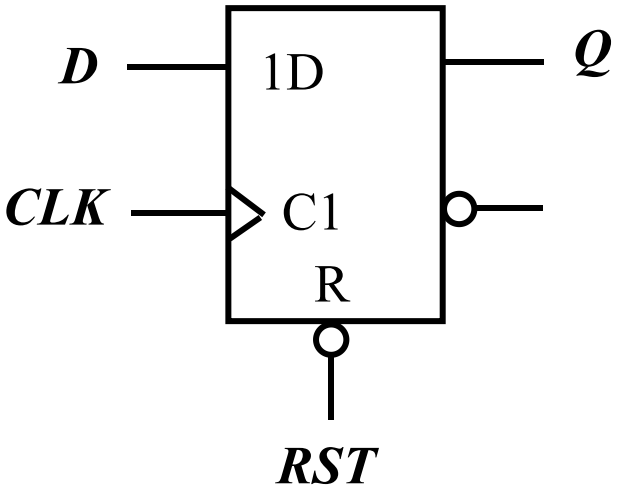
1. 没有异步清零端DFF

```
//D flip-flop without reset
module DFF (Q, D, CLK);
output  Q;
input D, CLK;
reg Q;
always @ (posedge CLK )
begin
Q <= D;
end
endmodule
```



2. 有异步清零端DFF

```
//D flip-flop with asynchronous reset ( V2001,V2005)
module DFF (output reg Q, input D, CLK, RST);
always @ (posedge CLK , negedge RST)
begin
if (!RST)  Q <= 1'b0;    //same as: if (RST == 0)
else  Q <= D;
end
endmodule
```



3. 有异步清零、置位端DFF

```
module dff_rs_async(clk,r,s,d,q);
input clk,r,s,d;
output q;
reg q;
always@(posedge clk or posedge r or posedge s)
begin
if(r) q<=1'b0;
else if(s) q<=1'b1;
else q<=d;
end
endmodule
```