

Verilog HDL语言入门

合肥工业大学计算机与信息学院

计算机组成原理课程组

Verilog程序的最基本结构

- ❑ 模块（**module**）是Verilog的基本描述单位，用于描述某个设计的**功能或结构**
- ❑ 一个模块可以在另一个模块中使用。
- ❑ Verilog **大小写敏感**

Verilog 模块

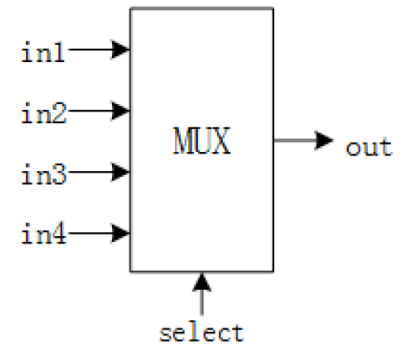
- 一个电路就是一个 **module**

```
module mux1(  
    input wire [3:0] in1, in2, in3, in4,  
    input wire [1:0] select,  
    output reg [3:0] out  
);  
  
    always@* begin  
        case (select)  
            2'b00: out = in1;  
            2'b01: out = in2;  
            2'b10: out = in3;  
            2'b11: out = in4;  
            default: out = 4'bx;  
        endcase  
    end  
  
endmodule
```

module name

ports names of module

module contents



行为描述

- 行为描述可以使用：

- `always`和`initial`语句：只有`reg`类型数据可以在这两种语句中赋值。

- 赋值语句：

- **阻塞型**过程赋值 “=” ：前一条语句没有完成赋值过程之前，后面的语句不可能被执行。

- **非阻塞型**过程赋值 “<=” ：一条非阻塞型赋值语句的执行，并不会影响块中其它语句的执行。

- **连续赋值**语句：用关键词`assign`，只要输入端操作数的值发生变化，该语句就重新计算并刷新赋值结果

```
always @( ... );
```

```
begin
```

```
.....
```

```
a <= a+1;
```

```
b <= a+1;
```

```
.....
```

```
end
```

如果执行前a的值为0，那么执行上面的过程语句后，a为1，b为1

非阻塞赋值

```
always @( ... );
```

```
begin
```

```
.....
```

```
a = a+1;
```

```
b = a+1;
```

```
.....
```

```
end
```

如果执行前a的值为0，那么执行上面的过程语句后，a为1，b为2

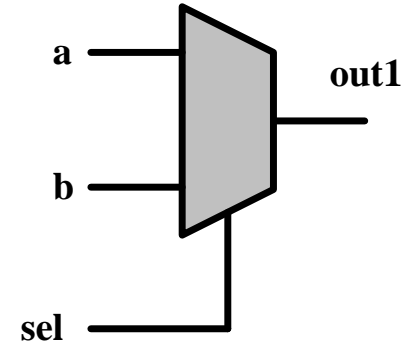
阻塞赋值

数据类型

- 线网类型（**wire**）：表示Verilog元件间的物理连线，一般使用持续赋值**assign**语句赋值
- 寄存器类型（**reg**）：表示一个抽象的数据存储单元，它只能在**always**语句和**initial**语句中被赋值
- 数组
 - `reg [15:0] reg_file[0:31];` // 32个16位的寄存器构成的寄存器堆

行为描述

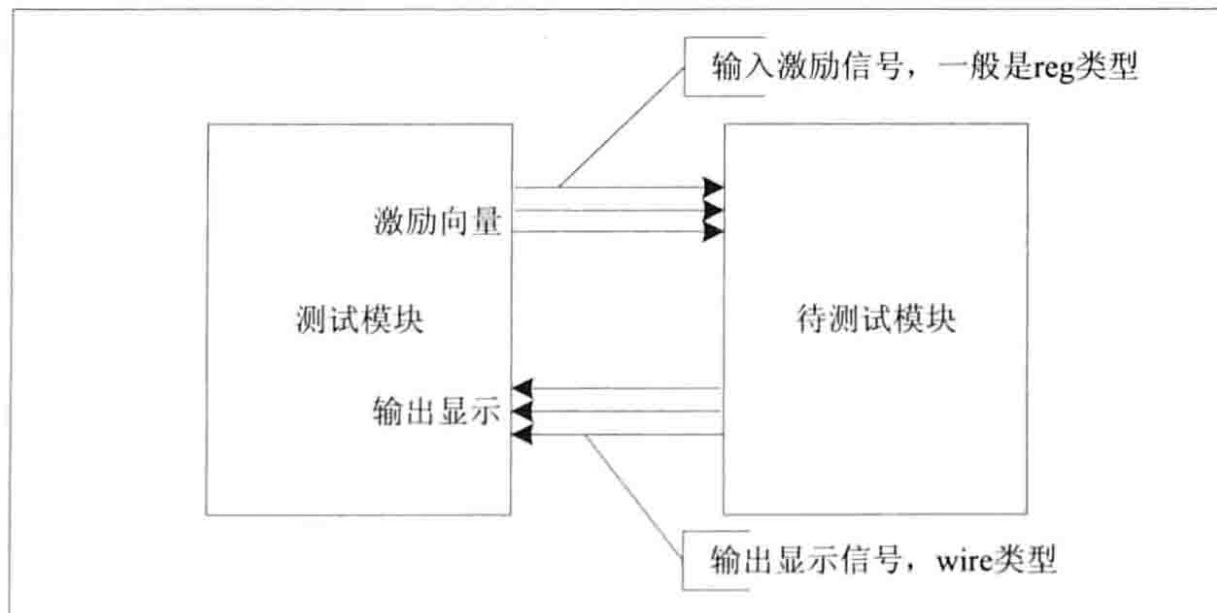
```
module mux2_1(out1, a, b, sel) ;  
    output reg out1, out2;  
    input  a, b;  
    input  sel;  
  
    assign out2 = a+b; //持续赋值语句  
  
    always @(sel or a or b)  
    begin  
        if (sel)  
            out1 = b;  
        else  
            out1 = a;  
        end  
    endmodule
```



设计验证与仿真

- 要测试一个设计块是否正确，需要一个测试模块
- 这个测试模块应包括以下三个方面的内容：
 - 测试模块中要调用到设计块，只有这样才能对它进行测试；
 - 测试模块中应包含测试的激励信号源；
 - 测试模块能够实施对输出信号的检测。

Verilog电路的仿真与验证



- Test Bench 只有模块名，没有端口列表；激励信号（输入到待测试模块的信号）必须定义为 reg 类型，以保持信号值；从待测试模块输出的信号（用户观察的信号）必须定义为 wire 类型。
- 在 Test Bench 中要调用被测试模块，也就是元件例化。
- Test Bench 中一般会使用 initial、always 过程块来定义、描述激励信号。

Testbench实例

```
module mux41_tb;

    reg [3:0] in1, in2, in3, in4;
    reg [1:0] select;
    wire [3:0] out;

    initial begin
        in1 = 4'b0001;
        in2 = 4'b0011;
        in3 = 4'b0111;
        in4 = 4'b1111;
        select = 2'b00;

        #10 select = 2'b01;

        #10 select = 2'b10;

        #10 select = 2'b11;

        #10 $stop;
    end

    mux41 uut(
        .in1(in1), .in2(in2), .in3(in3), .in4(in4),
        .select(select),
        .out(out)
    );

endmodule
```