



## 奋斗的小孩之 altera 系列

### 第八篇 verilog 基础语法

在 Verilog 语言中，可综合的语句（可以被综合成电路）并不多，本篇中着重介绍可综合语句，不可综合语句后续再来介绍。

模块（module）是 verilog HDL 设计当中的基本组成单元，每个设计都是由一个或者多个模块构成，为了能更好地完成设计，我们先来学习模块的写法。

注意：在设计当中，应该在英文状态下输入，否则将会出现语法错误。Verilog HDL 是区分大小写的。

模块必须以关键字 module 开始，以关键字 endmodule 结束，例：

```
module <模块名> (<端口名称, ..., 端口名称>);
```

```
    <端口方向> <端口类型> <端口位宽> <端口名称>;
```

```
    .....
    .....
endmodule
```

端口的方向：input（输入），output（输出），inout（双向端口）。

端口类型：wire（可以省略不写），reg。

端口的位宽：[X -1:0]（定义一个位宽为 X 的总线）。



中间寄存器或中间连线，定义方法如下：

```
wire [X-1:0] <名称>;
```

```
reg [X-1:0] <名称>;
```

下面介绍 verilog 建模的方法：

### 1. 结构建模

在学习结构建模之前，我们首先先来学习一下，怎么样调用模块实例。

假设我们已经写好了一个二选一多路选择器的代码，如下：

```
module mux21 (dataa, datab, outdata, sel);  
    input [2:0] dataa;  
    input [2:0] datab;  
    input sel;  
    outdata [2:0] outdata;  
    .....  
    .....  
endmodule
```

那么我们应该怎么样去调用它呢？如下：

```
<模块名><实例名>(  
    .端口（连线）,  
    .端口（连线）  
);
```



下面是用结构建模的方式实现三选一：

```
module mux31 (dataa, datab, datac, sel1, sel2, outdata);

    input [2:0] dataa;

    input [2:0] datab;

    input [2:0] datac;

    input sel1;

    input sel2;

    output [2:0] outdata;

    wire [2:0] data;

    mux21 mux21_dut1 (
        .dataa(dataa),
        .datab(datab),
        .outdata(data),
        .sel(sel1)
    );

    mux21 mux21_dut2 (
        .dataa(data),
        .datab(datac),
```



```
.outdata(outdata),  
  
.sel(sel2)  
  
);  
  
endmodule
```

上述建模是利用了两个二选一多路选择器来构成了一个三选一多路选择器。

## 2. 数据流建模。

使用 assign 语句进行赋值，例如：

```
assign outdata = indata1 + indata2;
```

上述赋值执行的方式：当等号右端的数据发生变化时，右端的表达式就会被重新计算，然后赋给等号左侧的变量。

## 3. 行为建模。

使用 always 进行建模，例如：

```
always @ (posedge clk)  
if (rst)  
  
    outdata <= 0;  
else  
  
    outdata <= indata;
```

1) always 语句是重复执行的。

2) 括号里面的内容为敏感变量，当敏感变量有变化时，

always 语句执行一次。



- 3) 当敏感列表里面是 posedge (上升沿) 或者 negedge (下降沿) 时, 用来描述时序逻辑, 直接为某个电平时, 用来表示组合逻辑。
- 4) 时序逻辑用 <= (非阻塞) 赋值, 组合逻辑用 = (阻塞) 赋值。
- 5) 凡是在 always 模块当中被赋值了, 都应该定义成 reg 的形式。

上述只是很简单的介绍了语法, 如果还是不明白的小伙伴可以直接看笔者后续的内容, 在应用中, 慢慢去体会这些语法。

制作人: 奋斗的小孩

fpga 交流群: 282124839