



Verilog HDL语言规范

主讲：何宾

Email: hebin@mail.buct.edu.cn

2014.06

Verilog HDL语言要素

Verilog语言要素主要包括:

- 注释
- 间隔符
- 标识符
- 关键字
- 系统任务和函数
- 编译器命令
- 运算符
- 数字
- 字符串和属性

Verilog HDL语言要素

--注释

在Verilog HDL中有两种形式的注释。该语法规定和C语言一致。

□ 单行注释

起始于双斜杠“//”，表示该行结束，以及新的一行开始。单行注释符号“//”在块注释语句内并无特定含义。

□ 多行注释

以符号单斜杠星号“/*”作为开始标志，以星号单斜杠“*/”作为结束标志。块注释不能嵌套。

Verilog HDL语言要素

--间隔符

间隔符包括空格字符 (\b)、制表符(\t)、换行符(\n)以及换页符，这些字符除了起到与其他词法标识符相分隔的作用外，可以被忽略。

间隔符除起到分隔的作用外，在必要的地方插入相应的空格或换行符，可以使程序文本易于用户阅读与修改。

注：在字符串中，将空白和制表符认为是有意义的字符。

Verilog HDL语言要素

--标识符

Verilog HDL中的标识符可以是任意一组字母、数字、\$符号和_(下划线)符号的组合，是赋予一个对象唯一的名字。

对于标识符来说：

- 标识符的第一个字符必须是字母或者下划线。
- 标识符区分大小写。

Verilog HDL语言要素

--标识符

在Verilog HDL中，标识符分为简单标识符、转义标识符。

□ 简单标识符

简单标识符是由字母、数字、货币符号（\$）、下划线构成的任意序列。

简单标识符的第一个符号不能使用数字或\$符号，且简单标识符对大小写敏感。

Verilog HDL语言要素

--标识符

简单标识符定义的例子。

shiftreg_a

busa_index

error_condition

merge_ab

_bus3

n\$657

Verilog HDL语言要素

--标识符

□ 转义标识符

转义标识符可以在一条标识符中包含任何可打印字符。

转义标识符以\ (反斜线) 符号开头，以空白结尾。

注：空白可以是一个空格、一个制表字符或换行符。

Verilog HDL语言要素

--标识符

下面给出转义标识符的例子

`\busa+index`

`\-clock`

`***error-condition***`

`\net1\net2`

`\{a,b}`

`\a*(b+c)`

Verilog HDL语言要素

--关键字

Verilog HDL语言内部所使用的词称为关键字或保留字。

- 不能随便使用这些保留字。
- 所有的关键字都使用小写字母。

注：如果关键字前面带有转义标识符，则不再作为关键字使用。

Verilog HDL语言要素

--系统任务和函数

为了便于设计者对仿真过程进行控制，以及对仿真结果进行分析，Verilog HDL提供了大量的系统功能调用。

大致可以分为两类：

- 任务型功能调用，称为系统任务。
- 函数型功能调用，称为系统函数。

Verilog HDL语言要素

--系统任务和函数

Verilog HDL中以\$字符开始的标识符表示系统任务或系统函数。

它们的区别主要包含：

- 系统任务可以返回0个或多个值。
- 系统函数只有一个返回值。
- 系统函数在0时刻执行，即：不允许延迟。
- 系统任务可以带有延迟。

Verilog HDL语言要素

--系统任务和函数

系统任务的Verilog HDL描述例子

```
$display ("display a message");
```

```
$finish;
```

Verilog HDL语言要素

--编译器命令

同C语言中的编译预处理指令一样，Verilog HDL也提供了大量编译指令。

- 通过这些编译指令，使得EDA工具厂商用他们的工具解释Verilog HDL模型变得相当容易。
- 以`（重音符号）开始的某些标识符是编译器指令。
- 在编译Verilog HDL语言时，特定的编译器指令均有效，即：编译过程可跨越多个文件，直到遇到其它不同编译程序指令为止。

Verilog HDL语言要素

--编译器命令

编译器命令的Verilog HDL描述例子

```
`define wordsize 8
```




Verilog HDL语言要素

--运算符

Verilog提供了丰富的运算符，关于运算符的知识将在以后详细介绍。

Verilog HDL语言要素

--数字

这里主要介绍整数型常量和实数型常量。

□ 整数型常量

整数型常量可以按如下两种方式表示：

➤ 简单的十进制格式

- ◆ 这种形式的整数定义为带有一个可选的“+”（一元）或“-”（一元）操作符的数字序列。
- ◆ 在常数前的+或者-号，是一个一元的加或者减操作符。

Verilog HDL语言要素

--数字

➤ 基数表示法

这种形式的整数格式为：<size><'base_format><number>

其中：

□ <size>

定义将数字(number)转换为二进制位后，得到的位宽。该参数是一个非零的无符号十进制常数。

□ `<base_format>

撇号是指定位宽格式表示法的固有字符，不能省略。

Verilog HDL语言要素

--数字

`base_format`是用于表示数基数格式（进制）的一个字母，对大小写不敏感。在撇号后可以添加下面的基数标识：

- 字母s/S，表示该数为有符号数。
- 字母o/O，表示八进制数。
- 字母b/B，表示二进制数。
- 字母d/D，表示十进制数。
- 字母h/H，表示十六进制数。

注意：撇号和基数标识之间不能有空格。

Verilog HDL语言要素

--数字

□ <number>

是基于基数值无符号数字序列，由基数格式所对应的数字串组成。数值x和z以及十六进制中的a到f不区分大小写。每个数字之间，可以通过‘_’符号连接。

注：

- 对于没有位宽和基数的十进制数，将其作为有符号数。
- 如果带有基数的十进制数包含了s，则认为有符号数。
- 如果只带有基数，则认为是无符号数。
- 负数应该用二进制的补码表示。

Verilog HDL语言要素

--数字

3. 非对齐宽度整数的处理

对于非对齐宽度整数的处理，遵循下面的规则：

- 当位宽小于无符号数的实际位数时，截断相应的高位部分。
- 当位宽大于无符号数的实际位数，且数值的最高位是0或1时，相应的高位部分补0或1。
- 当位宽大于无符号数的实际位数，且数值的最高位是x或z时，相应的高位部分补x或z。
- 如果未指定无符号数的位宽，那么默认的位宽至少为32位。

Verilog HDL语言要素

--数字

未指定位宽常数的例子

659 // 十进制数

'h 837FF // 十六进制数

'o 7460 // 八进制数

Verilog HDL语言要素

--数字

指定位宽常数的例子

4'b1001 // 四位二进制数

5'D3 // 五位十进制数

3'b01x // 三位数，其最低有效位未知（x表示不确定）

12'hx // 十二位数，其值不确定

16'hz // 十六位数，其值为高阻（z表示高阻状态）

Verilog HDL语言要素

--数字

带符号常数的例子

8 'd -6 //非法声明

-8 'd 6 //定义了6的二进制补码，共8位，等效于-(8'd6)

4 'shf // 定义了4位数，将其理解为-1的二进制补码，等效于-4'h 1

-4 'sd15 //这个等效于-(-4'd 1)，或者 '0001'

16'sd? //和16'sbz相同

注：?符号用于替换z。对于十六进制，设置为四位；对于八进制，设置为三位；对于二进制，设置为1位。

Verilog HDL语言要素

--数字

自动左对齐常数的例子

```
reg [11:0] a, b, c, d;
```

```
initial begin
```

```
    a = 'h x;  //生成xxx
```

```
    b = 'h 3x; //生成03x
```

```
    c = 'h z3; //生成zz3
```

```
    d = 'h 0z3; //生成0z3
```

```
end
```

Verilog HDL语言要素

--数字

```
reg [84:0] e, f, g;
```

```
e = 'h5;    // 生成{82{1'b0},3'b101}
```

```
f = 'hx;    // 生成{85{1'hx}}
```

```
g = 'hz;    // 生成{85{1'hz}}
```

带下划线的常数例子

```
27_195_000
```

```
16'b0011_0101_0001_1111
```

```
32 'h 12ab_f001
```

Verilog HDL语言要素

--数字

□ 实数型常量

在IEEE Std 754-1985中，对实数的表示进行了说明。该标准用于双精度浮点数。实数可以用十进制计数法或者科学计数法表示。

注：十进制小数点两边，至少要有一个数字。

Verilog HDL语言要素

--数字

实数常量有效表示的例子

1.2

0.1

2394.26331

1.2E12 //指数符号为e或者E

1.30e-2

0.1e-0

23E10

29E-2

236.123_763_e-12 //忽略下划线

Verilog HDL语言要素

--数字

实数常量无效表示的例子

.12

9.

4.E3

.2e-7

Verilog HDL语言要素

--数字

实数到整数的转换

Verilog HDL语言规定，通过四舍五入到最近整数的方法，将实数转换为整数。

对实数四舍五入后的表示

- 42.446和42.45转换为整数42
- 92.5和92.699转换为整数93
- -5.62转换为整数-6
- -26.22转换为整数-26

Verilog HDL语言要素

--字符串

字符串是双引号内的字符序列，用一串8位二进制ASCII码的形式表示，每一个8位二进ASCII码代表一个字符。

例如：

字符串“ab”等价于16'h5758。如果字符串用于Verilog表达式或复制语句的操作数时，字符串被当作无符号整数序列。

Verilog HDL语言要素

--字符串

字符串变量声明

字符串变量是寄存器型变量，它的位宽等于字符个数乘以8。

Verilog HDL语言要素

--字符串

字符串变量的声明。

典型地，存储12个字符的字符串“Hello China!”需要 $8*12$ （即96位）宽的寄存器。

```
reg [8*12:1] str1;
```

```
initial
```

```
begin
```

```
    str = "Hello China!";
```

```
end
```

Verilog HDL语言要素

--字符串

字符串操作

可以使用Verilog HDL的操作符对字符串进行处理，被操作符处理的数据是8位ASCII码的序列。

对于非对齐宽度的情况，采用下面的方式进行处理：

- 在操作过程中，如果声明的字符串变量位数大于字符串实际长度，则在赋值操作后，字符串变量的左端（即高位）补0。这一点与非字符串值的赋值操作是一致的。
- 如果声明的字符串变量位数小于字符串实际长度，那么截断字符串的左端，这样就丢失了高位字符。

Verilog HDL语言要素

--字符串操作的例子

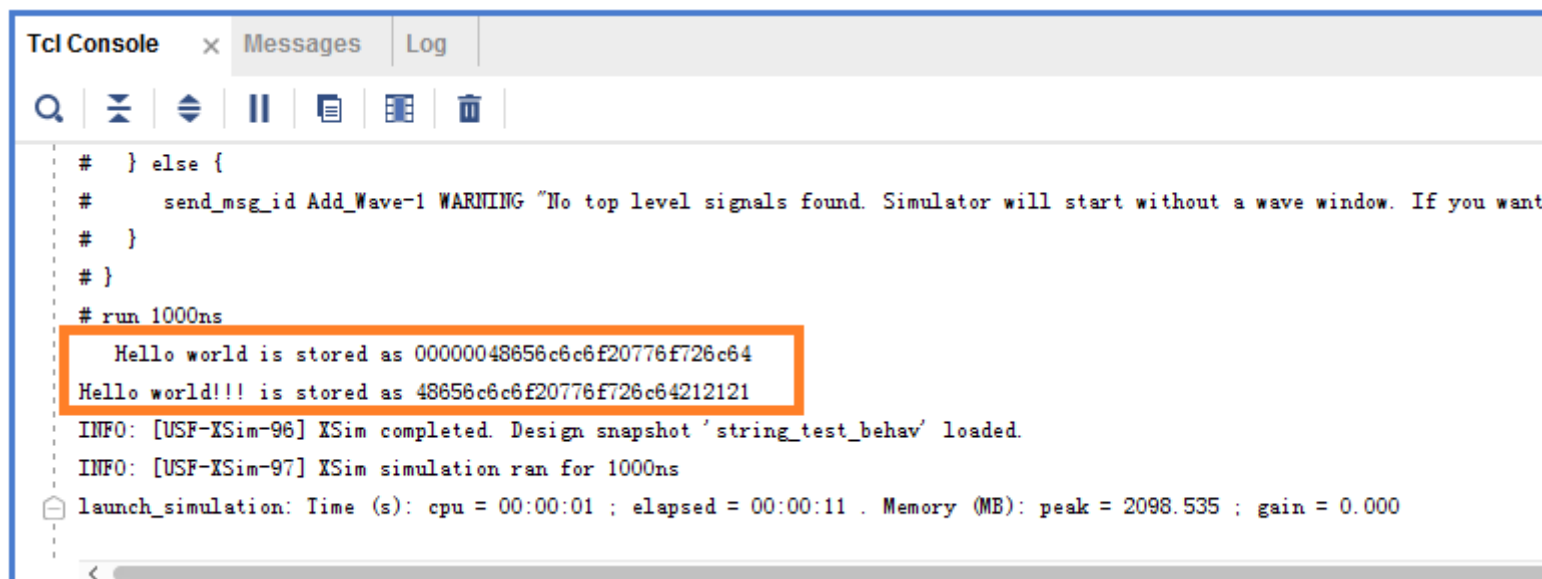
```
module string_test;  
    reg [8*14:1] stringvar;  
    initial begin  
        stringvar = "Hello world";  
        $display("%s is stored as %h", stringvar,stringvar);  
        stringvar = {stringvar,"!!!"};  
        $display("%s is stored as %h", stringvar,stringvar);  
    end  
endmodule
```

该设计保存在本书配套资源的\eda_verilog\attribute_1目录下

Verilog HDL语言要素

--字符串操作的例子

测试结果



The screenshot shows a simulation console window with tabs for 'Tcl Console', 'Messages', and 'Log'. The 'Tcl Console' tab is active, displaying a series of commands and their outputs. Two lines of output are highlighted with an orange box: 'Hello world is stored as 00000048656c66f20776f726c64' and 'Hello world!!! is stored as 48656c66f20776f726c64212121'. Below these, there are informational messages from the XSim simulator and a final status line for the simulation launch.

```
Tcl Console x Messages Log
[Search] [Zoom In] [Zoom Out] [Full Screen] [Print] [Close] [Delete]

# } else {
#     send_msg_id Add_Wave-1 WARNING "No top level signals found. Simulator will start without a wave window. If you want
# }
# }
# run 1000ns
Hello world is stored as 00000048656c66f20776f726c64
Hello world!!! is stored as 48656c66f20776f726c64212121
INFO: [USF-XSim-96] XSim completed. Design snapshot 'string_test_behav' loaded.
INFO: [USF-XSim-97] XSim simulation ran for 1000ns
launch_simulation: Time (s): cpu = 00:00:01 ; elapsed = 00:00:11 . Memory (MB): peak = 2098.535 ; gain = 0.000
```


Verilog HDL语言要素

--字符串

特殊字符

在某些字符之前可以加上一个引导性的字符（转义字符），
这些的字符只能用于字符串中。下表列出这些特殊字符表示和意义。

特殊字符表示	意义
\n	换行符
\t	Tab键
\\	符号\
\”	符号”
\ddd	3位八进制数表示的ASCII 值 ($0 \leq d < 7$)

Verilog HDL语言要素

--属性

随着工具的扩展，除了仿真器使用Verilog HDL作为其输入源外，还包含另外一个机制，即在Verilog HDL源文件中指定对象、描述和描述组的属性。

- 这些属性可以用于各种工具中，包括：仿真器、控制工具的操作行为。这些属性称为attribute。

Verilog HDL语言要素

--属性

指定属性的格式

`(* attribute_name = constant_expression *)`

或者

`(* attribute_name *)`

Verilog HDL语言要素

--属性

将属性添加到case描述的Verilog HDL描述例子。

```
(* full_case, parallel_case *)
```

```
case (foo)
```

```
    <rest_of_case_statement>
```

```
(* full_case=1 *)
```

```
(* parallel_case=1 *) // 多个属性
```

Verilog HDL语言要素

--属性

```
case (foo)
```

```
<rest_of_case_statement>
```

或者

```
(* full_case, // 没有分配值
```

```
parallel_case=1 *)
```

```
case (foo)
```

```
<rest_of_case_statement>
```

Verilog HDL语言要素

--属性

添加full_case属性, 但是没有parallel_case属性的

Verilog HDL描述

```
(* full_case *) // parallel_case not specified  
case (foo)  
    <rest_of_case_statement>
```

或者

```
(* full_case=1, parallel_case = 0 *)  
case (foo)  
    <rest_of_case_statement>
```

Verilog HDL语言要素

--属性

将属性添加到模块定义的Verilog HDL描述例子

```
(* optimize_power *)
```

```
module mod1 (<port_list>);
```

或者

```
(* optimize_power=1 *)
```

```
module mod1 (<port_list>);
```

Verilog HDL语言要素

--属性

将属性添加到模块例化的Verilog HDL描述例子

```
(* optimize_power=0 *)
```

```
mod1 synth1 (<port_list>);
```

将属性添加到操作符的Verilog HDL描述例子

```
a = b + (* mode = "cla" *) c;
```

//将属性模式 的值设置为字符串cla。

Verilog HDL语言要素

--属性

将属性添加到reg声明的Verilog HDL描述例子

```
(* fsm_state *) reg [7:0] state1;
```

```
(* fsm_state=1 *) reg [3:0] state2, state3;
```

```
reg [3:0] reg1; // 这个reg没有fsm_state设置
```

```
(* fsm_state=0 *) reg [3:0] reg2; // 这个也没有
```

Verilog HDL语言要素

--属性

将属性添加到一个Verilog函数调用的Verilog HDL描述例子

```
a = add (* mode = "cla" *) (b, c);
```

将属性添加到一个有条件操作符的Verilog HDL描述例子

```
a = b ? (* no_glitch *) c : d;
```

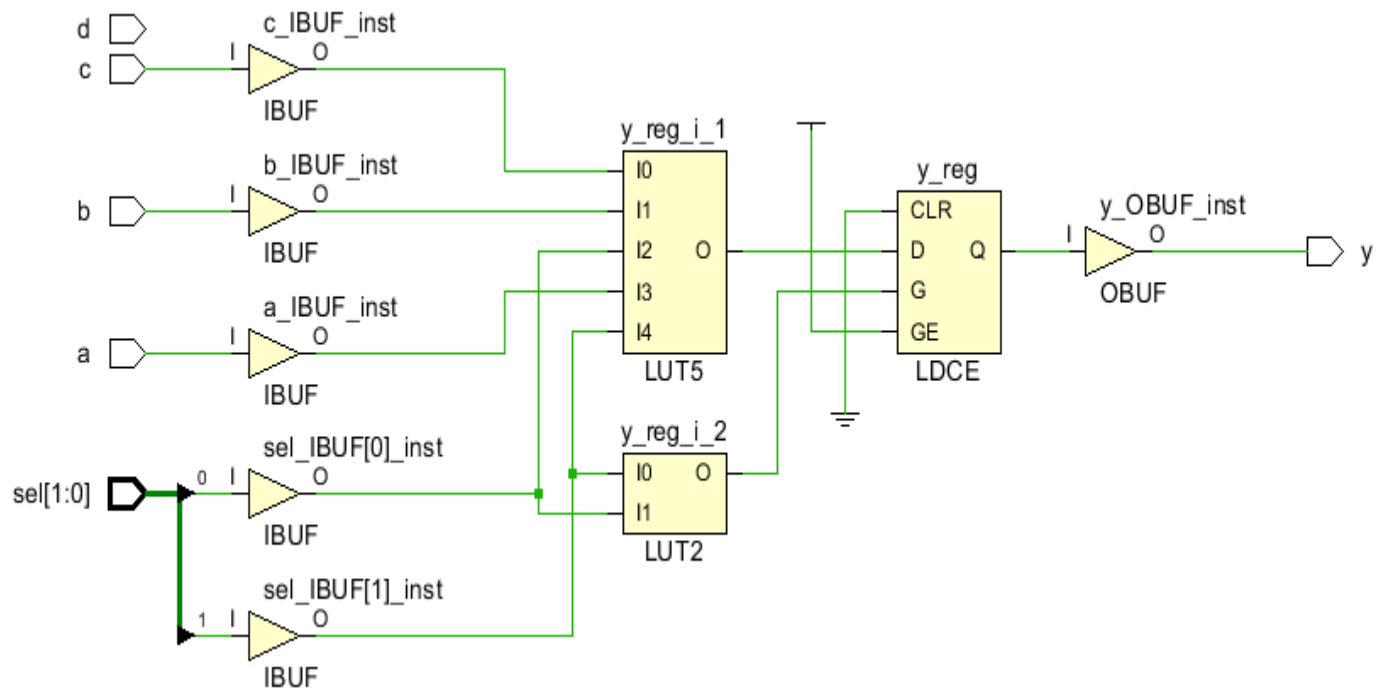
Verilog HDL语言要素

--属性

一个未充分描述case的例子，没有添加full_case属性

```
module mul(  
    input [1:0] sel,  
    input a,b,c,d,  
    output reg y  
);  
always @(*)  
begin  
    // (* full_case *)  
    case (sel)  
        2'b00 : y=a;  
        2'b01 : y=b;  
        2'b10 : y=c;  
    endcase  
end  
endmodule
```

Vivado综合的结果，产生了锁存器



该设计保存在本书配套资源的\eda_verilog\attribute_1目录下

Verilog HDL语言要素

--属性

一个未充分描述case的例子，添加full_case属性

```
module mul(  
    input [1:0] sel, input a,b,c,d,  
    output reg y  
);  
always @(*)  
begin
```

```
    (* full_case *)
```

```
    case (sel)
```

```
        2'b00 : y=a;
```

```
        2'b01 : y=b;
```

```
        2'b10 : y=c;
```

```
    endcase
```

```
end
```

```
endmodule
```

Vivado综合的结果，未产生锁存器

