Verilog HDL语言规范

主 讲：何宾

Email：hebin@mail.buct.edu.cn

2014.06

Verilog HDL数据类型

Verilog HDL数据类型包括：

- 值的集合
- 网络和变量
- 向量
- 隐含声明
- 网络类型
- 寄存器类型
- 整数/实数/时间
- 数组
- 参数
- Verilog名字空间

Verilog HDL数据类型
--值的集合

Verilog HDL有下列四种基本的值：

- 0

逻辑0或“假”状态。

- 1

逻辑1或“真”状态。

- x（X）

未知状态，对大小写不敏感。

- z（Z）

高阻状态，对大小写不敏感。

Verilog HDL数据类型
--值的集合

注意：

- 这四种值的解释都内置于语言中.
  - 一个为z的值总是意味着高阻抗。
  - 一个为0的值通常是指逻辑0。
- 在门的输入或一个表达式中的为‘z’的值通常解释成‘x’。

Verilog HDL数据类型
--网络和变量

在Verilog HDL中，根据赋值和对值的保持方式不同，可将数据类型主要分为两大类：

- 网络型
- 变量型

特别要注意：这两类数据代表了不同的硬件结构！！！

Verilog HDL数据类型
--网络和变量

网络表示器件之间的物理连接，需要门和模块的驱动。

- 网络类型不保存值（除trireg类型以外），其输出始终根据输入的变化而变化。
- 对于没有声明的网络，默认为1位（标量）wire类型。
- Verilog HDL禁止再次声明已经声明过的网络、变量或参数。

Verilog HDL数据类型
--网络和变量

声明网络类型的语法格式如下：

net_type [range] [delay] net_name [,net_name];

其中：

- **net_type**

  表示网络类型数据。

- **range**

  用来表示数据为标量或矢量。若没有声明范围，则表示数据为1位的标量。否则，由该项指定数据的矢量形式。

- **delay**

  指定仿真延迟时间。

- **net_name**

  网络名字。可以一次定义多个网络，多个网络之间用逗号隔开。

Verilog HDL数据类型
--网络和变量

声明网络类型的Verilog HDL描述的例子

```
wand w;                    // 一个标量wand网络类型
tri [15: 0] bus;           // 16位三态总线网络类型
wire [0: 31] w1, w2;       // 两个32位网络类型，MSB为bit0
```

Verilog HDL数据类型
--网络和变量

变量是对数据存储元件的抽象。

- 从当前赋值到下一次赋值之前，变量应当保持当前的值不变。
- 程序中的赋值语句将引起保存在数据元件中值的改变。

注：

- 对于reg，time和integer这些变量类型数据，它们的初始值应当是x。
- 对于real和realtime变量类型数据，默认的初始值是0.0。
- 如果使用变量声明赋值语句，则变量将声明赋值语句所赋的值作为初值，这与initial结构中对变量的赋值等效。
- 在变量数据类型中，只有reg和integer变量型数据类型是可综合的，其它是不可综合的。

Verilog HDL数据类型
--向量

- 在一个网络或寄存器类型声明中，如果没有指定其范围，默认将其当作1比特位宽。也就是通常所说的标量。
- 通过指定范围，声明多位的网络或寄存器型数据，则称为矢量（也叫做向量）。

Verilog HDL数据类型
--向量

向量范围由常量表达式来说明

- msb_constant_expression（最高有效位常量表达式）

代表范围的左侧值

- lsb_constant_expression（最低有效位常量表达式）

代表范围的右侧值。

Verilog HDL数据类型
--向量

向量声明的Verilog HDL描述的例子

```
wand w;                    // wand类型的标量
tri [15:0] busa;           // 一个三态16位总线
trireg (small) storeit;    // 低强度的一个充电保存点
reg a;                     // reg类型的标量
reg[3:0] v;                // 4位reg向量，由v[3], v[2], v[1]和v[0]构成
reg signed [3:0] signed_reg; //一个四位向量，其范围为-8到7
reg [-1:4] b;                  // 一个6位reg类型的向量
wire w1, w2;                   // 声明两个线网络
reg [4:0] x, y, z;             // 声明3个5位的reg类型变量
```

Verilog HDL数据类型
--隐含声明

如果没有显示声明网络或者变量，则在下面的情况中，默认将其指定为网络类型：

- 在一个端口表达式的声明中，如果没有对端口的数据类型进行显式说明，则默认的端口数据类型就为wire型。并且，默认的wire型矢量的位宽与矢量型端口声明的位宽相同。
- 模块例化的端口列表中，如果先前没有对端口的数据类型进行显式说明，那么默认的端口数据类型为网络型标量。
- 如果一个标识符出现在连续赋值语句的左侧，而该标识符先前未曾被声明，那么该标识符的数据类型就被隐式声明为网络型标量。

Verilog HDL数据类型
--网络类型

下表给出了这些常用的不同网络类型的功能及其可综合性。

类型	功能	Vivado可综合性
wire，tri	标准内部连接线	√
supply1，supply0	电源和地	√
wor，trior	多驱动源线或	×
wand，triand	多驱动源线与	×
trireg	能保存电荷的net	×
tri1，tri0	无驱动时上拉/下拉	√

注：在Vivado集成开发环境的实现工具中，禁止出现多个源驱动同一个网络的情况，遇到这种情况会导致实现过程的失败。

Verilog HDL数据类型
--网络类型

简单的网络类型说明格式为：

net_kind[msb:lsb]net1,net2, . . . , netN;

其中：

- net_kind

  是上述网络类型的一种。

- msb和lsb

  用于定义网络范围的常量表达式。其范围定义是可选的。如果没有定义范围，默认的网络类型为1位。

Verilog HDL数据类型
--网络类型

默认网络型数据的初始化值为Z。带有驱动的网络型数据应当为它们的驱动输出指定默认值。

注：

trireg网络型数据是一个例外。它的默认初始值为x，而且在声明语句中应当为其指定电荷量强度。

网络类型
--wire和tri网络类型

用于连接单元的连线是最常见的网络类型。连线（wire）网络与三态（tri）网络语法和语义一致。

- 三态网络可以用于描述多个驱动源驱动同一根线的网络类型，并且没有其他特殊的意义。
- 如果多个驱动源驱动一个连线（或三态网络）由下表确定网络的有效值。

网络类型
--wire和tri网络类型

网络数据类型wire和tri的真值表

wire/tri	0	1	x	z
0	0	x	x	0
1	x	1	x	1
x	x	x	x	x
z	0	1	x	z

网络类型
--wire和tri网络类型

由关键词wire定义常用的网络类型。wire型网络的语法格式如下：

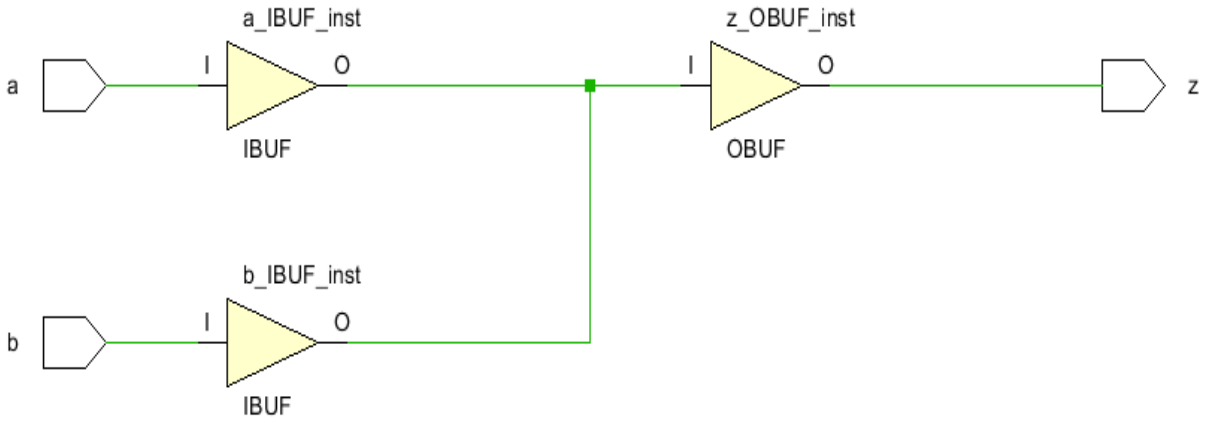
wire [n-1:0] <name1>,<name2>,…<namen>；

其中：

name1,…,namen表示wire型网络的名字。

网络类型
--wire和tri网络类型

```
module top(
    input a,
    input b,
    output z
     );
assign z=a;
assign z=b;
endmodule
```



网络类型
--wire和tri网络类型

```
module test;
reg a,b;
wire z;
top uut(.a(a),.b(b),.z(z));
initial
begin
  a=1'b0;
  b=1'bz;
  #100;
  a=1'b1;
  b=1'bz;
  #100;
  a=1'b0;
   b=1'b1;
   #100;
   a=1'bz;
   b=1'bz;
    #100;
    a=1'bx;
    b=1'bz;
    #100;
end
endmodule
```

网络类型
--wor和trior网络类型

线或（wor）和三态线或（trior）用于为连线型逻辑结构建模。当有多个驱动源驱动wor和trior型数据时，将产生线或结构。

- 如果驱动源中任一个为“1”，那么网络型数据的值也为“1”。
- 线或与三态线或在语法和功能上一致。

网络类型
--wor和trior网络类型

如果多个驱动源驱动这类网络，下表决定了网络的有效值。

wor/trior	0	1	x	z
0	0	1	x	0
1	1	1	1	1
x	x	1	x	x
z	0	1	x	z

网络类型
--wor和trior网络类型

```
module test;
 wire a=1'bz;
 wire b=1'bz;
 wor z;
 assign z=a;
 assign z=b;
endmodule
```

网络类型
--wand和triand网络类型

线与（wand）网络和三态线与（triand）网络用于为连线型逻辑结构建模。如果某个驱动源为“0”，那么网络输出为“0”。

- 当有多个驱动源驱动wand和triand型网络时，将产生线与结构。
- 线与网络和三态线与网络在语法和功能上一致。

网络类型
--wand和triand网络类型

如果这类网络存在多个驱动源，由下表决定网络的有效值。

Wand/triand	0	1	x	z
0	0	0	0	0
1	0	1	x	1
x	0	x	x	x
z	0	1	x	z

网络类型
--wand和triand网络类型

```
module test1;
wire a=1'bz;
wire b=1'bz;
wand z;
 assign z=a;
 assign z=b;
endmodule
```

网络类型
--tri0和tri1网络类型

这类网络类型可用于包含上拉或下拉电阻网络的建模。

- tri0/tri1网络的特征是：
  - 无驱动源驱动该网络时，它的值为0（tri1的值为1）。
  - 网络值的驱动强度都为pull。
- tri0相当于这样一个网络：由一个强度为pull的0值连续驱动该网络。
- tri1相当于这样一个网络：由一个强度为pull的1值连续驱动该网络。

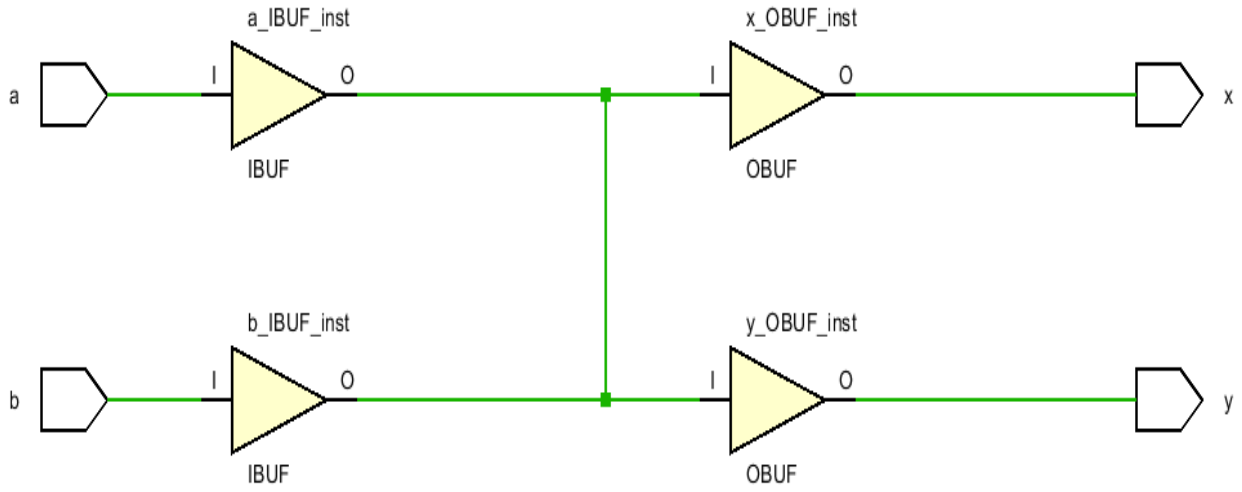
网络类型
--tri0和tri1网络类型

下表给出在多个驱动源情况下tri0或tri1网的有效值。

tri0/tri1	0	1	x	z
0	0	x	x	0
1	x	1	x	1
x	x	x	x	x
z	0	1	x	0/1

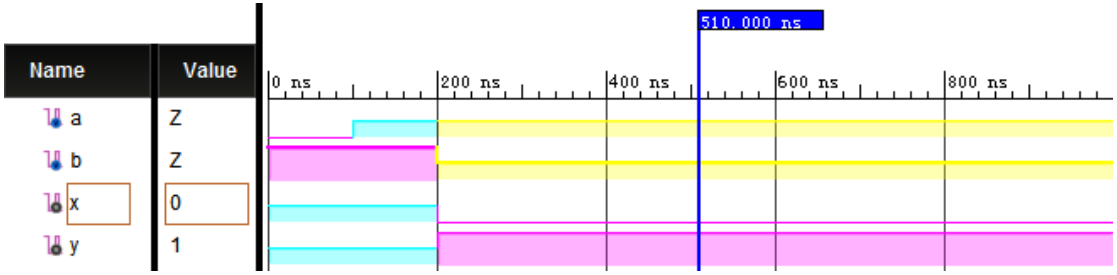
网络类型
--tri0和tri1网络类型

```
module top(
   input a,
   input b,
   output tri0 x,
   output tri1 y
    );
assign x=a;
assign x=b;
assign y=a;
assign y=b;
endmodule
```



网络类型
--tri0和tri1网络类型

```
module test1;
reg a,b;
tri0 x;
tri1 y;
top1 uut(.a(a),.b(b),.x(x),.y(y));
initial
begin
  a=1'b0;
  b=1'b1;
  #100;
  a=1'bx;
  b=1'b1;
  #100;
  a=1'bz;
  b=1'bz;
  #100;
end
endmodule
```



网络类型
--supply0和supply1网络类型

Supply0网络类型用于对“地”建模，即低电平（逻辑“0”）；supply1网络类型用于对“电源”建模，即高电平（逻辑“1”）。

supply0和supply1网络类型描述

```
supply0 Gnd,ClkGnd;
supply1 [2:0] Vcc;
```

网络类型
--未说明的网络

在Verilog HDL中，可能未声明某种网络类型。在这种情况下，默认为一位的网络类型。通过`default_nettype编译器指令改变隐含生命网络类型的方式。使用方法如下：

```
`default_nettype net_kind
```

带有下列编译器指令：

```
`default_nettype wand
```

任何未被说明的网默认为一位线与（wand）网。

Verilog HDL数据类型
--寄存器类型

通过过程分配语句给寄存器类型变量分配值（赋值）。

在每个分配的过程中间，寄存器保持上次分配的值。

- 它用于对硬件寄存器进行建模，包括对边沿敏感（比如：触发器）和电平敏感（比如：置位/复位和锁存器）的存储元件。

注：

一个寄存器变量不一定代表一个硬件存储元件，这是因为它也能用于表示一个组合逻辑。

Verilog HDL数据类型
--寄存器类型

寄存器型变量与网络类型的区别主要在于：

- 寄存器型变量保持最后一次的赋值。只能在initial或always内部对寄存器型变量进行赋值操作。
- 网络型数据需要有连续的驱动源。

Verilog HDL数据类型
--寄存器类型

寄存器型变量声明的格式如下：

reg_type [range] reg_name[, reg_name];

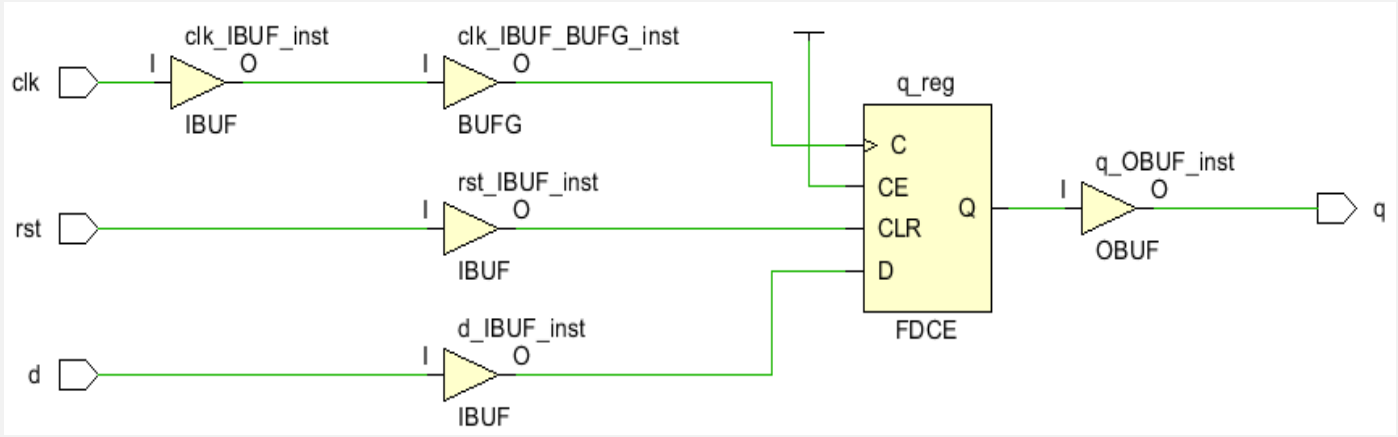
其中：

- reg_type为寄存器类型。
- range为矢量范围，[MSB：LSB]格式，只对reg类型有效。
- reg_name为reg型变量的名字，一次可定义多个reg型变量，使用逗号分开。

Verilog HDL数据类型
--寄存器类型

寄存器型变量的Verilog HDL描述例子

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



Verilog HDL数据类型
--整数、实数、时间和实时时间

整型变量声明

- 整型变量常用于对循环控制变量的声明。在算术运算中被视为二进制补码形式的有符号数。
- 整型变量和32位的寄存器型数据在实际意义上相同，只是寄存器型数据被当做无符号数来处理。

Verilog HDL数据类型
--整数、实数、时间和实时时间

整数变量声明的Verilog描述的例子

```
integer i,j;
integer [31:0] D;
```

注：

- 虽然interger有位宽度的声明，但是integer型变量不能作为位向量访问。D[6]和D[16:0]的声明都是非法的。
- 在综合时，integer型变量的初始值是x。

Verilog HDL数据类型
--整数、实数、时间和实时时间

实数型变量声明

- 在机器码表示法中，实数型数据是浮点型数值。该变量类型可用于对延迟时间的计算。
- 实数型变量是不可综合的。对于实数来说：
  - 不是所有的Verilog HDL操作符都能用于实数值。
  - 实数变量不使用范围声明。
  - 实数变量默认的初始值为0。

Verilog HDL数据类型
--整数、实数、时间和实时时间

时间型变量声明

- 时间型变量与整型变量类似，只是它是64位的无符号数。
- 时间型变量主要用于对仿真时间的存储与计算处理，常与系统函数$time一起使用。

实时时间变量声明

- 实时时间声明和实数声明进行同样的处理，能互换使用。

Verilog HDL数据类型
-- 数组

向量可以将已声明过类型的元素组合成多维的数据对象。

- 声明向量时，应当在所声明数据标识符的后面指定元素的地址范围。每个维度代表一个地址范围。
- 数组可以是一维向量（一个地址范围）也可以是多维向量（多重地址范围）。
- 向量的索引表达式应当是常量表达式，该常量表达式的值应当是整数。

Verilog HDL数据类型
--数组

- 通过一条分配（赋值）语句为一个数组中的每个元素赋值，但是不能为整个向量或向量的一部分赋值。
- 要给一个向量元素赋值，需要为该向量每个维度指定索引。向量索引可以是一个表达式，这就为向量元素的选择提供了一种机制，即根据电路中其他网络或变量的值来引用向量元素。

Verilog HDL数据类型
--数组

如果一个向量的元素类型为寄存器型，那么这样的一维向量也称为存储器。

- 存储器只用于对下面建模：
  - ROM（只读存储器）
  - RAM（随机存取存储器）
  - 寄存器组

Verilog HDL数据类型
--数组

- 向量中的每一个寄存器称为元素或字，并且通过一个索引实现寻址。
- 可以通过一条单独的赋值语句给一个n位的寄存器赋值，但是不能通过这样的一条语句对整个存储器赋值。
- 为了对存储器的某个字赋值，需要为该字指定数组索引。该索引可以是一个表达式。该表达式中含有其他的变量或网络数据，通过对该表达式的运算，得到一个结果值，从而定位存储器中的某个字。

Verilog HDL数据类型
--数组

数组声明Verilog HDL描述例子

```
reg [7:0] mema[0:255];      // 声明一个数组mema为256×8比特
                              寄存器，其索引0~255
reg arrayb[7:0][0:255];     // 声明一个二维数组，其数据宽度为1
                              位寄存器
wire w_array[7:0][5:0];     // 声明线类型网络数组
integer inta[1:64];         // 64个整数值的数组
time chng_hist[1:1000];     // 有1000个时间值的数组
integer t_index;
```

Verilog HDL数据类型
--数组

分配数组元素的Verilog HDL描述例子

```
mema = 0;                    // 非法的描述，尝试给整个数组写0
arrayb[1] = 0;               // 非法的描述，尝试写元素 [1][0]..[1][255]
arrayb[1][12:31] = 0;        // 非法的描述，尝试写元素[1][12]..[1][31]
mema[1] = 0;                 // 给mema的第二个元素分配0
arrayb[1][0] = 0;            // 给索引[1][0]指向的元素分配0
inta[4] = 33559;             // 给数组中的某个元素分配整数值33559
chng_hist[t_index] = $time; // 给当前索引指向的元素分配时间
```

Verilog HDL数据类型
--数组

不同存储器的Verilog HDL描述例子

```
reg [1:n] rega;          // 一个n位的1个深度的寄存器（存储器）

reg mema [1:n];          // 一个1位的n个深度的寄存器（存储器）
```

Verilog HDL数据类型
--参数

Verilog HDL中的参数既不属于变量类型也不属于网络类型范畴。

- 参数不是变量，而是常量。
- Verilog HDL中，提供了两种类型的参数：
  - 模块参数
  - 指定参数
- 这些参数可以指定范围。默认地，为parameter和specparams保持必要的宽度，用于保存常数的值。当指定范围时，按照指定的范围确定。

Verilog HDL数据类型
--参数

模块参数

模块参数定义的格式：

parameter par_name1=expression1,……,par_namen=expression;

其中：

- par_name1,....par_namen为参数的名字。
- expression1,....,expression为表达式。
- 可一次定义多个参数，用逗号隔开。
- 参数的定义是局部的，只在当前模块中有效。

Verilog HDL数据类型
--参数

一个模块参数可以指定类型和范围。根据下面的规则指定模块参数类型和范围：

- 对于没有指定类型和范围的参数，将根据分配给参数的最终值来确定其类型和范围。
- 一个指定范围，但没有指定类型的参数，将是参数声明的范围，并且是无符号的。符号和范围将不受到后面覆盖值的影响。

Verilog HDL数据类型
--参数

- 一个指定类型，但没有指定范围的参数，将是参数指定的类型。对于一个有符号的参数，默认为分配给参数最后值的范围。
- 一个指定有符号类型和范围的参数，将是有符号的，并且是参数指定的范围。其符号和范围将不受到后面覆盖值的影响。
- 一个没有指定范围，但是有指定符号类型或者没有指定类型的参数，有一个隐含的范围，其lsb为0，msb等于或者小于为分配给参数最后的值。
- 在编译时，可以改变参数值。当改变参数值时，可以使用参数定义语句或在模块初始化语句中定义的参数值。

Verilog HDL数据类型
--参数

参数的Verilog HDL描述的例子

```
parameter msb = 7;                  // 定义msb为常数值7

parameter e = 25, f = 9;            // 定义两个常数

parameter r = 5.7;                  // 定义r为实数参数

parameter byte_size = 8, byte_mask = byte_size - 1;
```

Verilog HDL数据类型
--参数

```
parameter average_delay = (r + f) / 2;

parameter signed [3:0] mux_selector = 0;

parameter real r1 = 3.5e17;

parameter p1 = 13'h7e;

parameter [31:0] dec_const = 1'b1;        // 值转换到32位

parameter newconst = 3‘h4;                // 暗示其范围[2:0]

parameter newconst = 4;                   // 暗示其范围[31:0]
```

Verilog HDL数据类型
--参数

本地参数

- 除了不能直接被defparam描述修改，以及不能通过模块例化参数分配以外，本地参数（localparam）和参数是一致的。
- 本地参数可以分配包含参数的常数表达式，这些参数可以通过defparam描述或者模块例化参数值分配进行修改。

Verilog HDL数据类型
--参数

指定参数

- 关键字specparam声明了一个特殊类型的参数，这个参数专用于提供时序和延迟值，但是可以出现在任何没有分配给一个参数的表达式内。
- 它不是一个声明范围描述的一部分。
- 在指定的块内以及在主模块内，允许指定参数。

Verilog HDL数据类型
--参数

- 当在一个指定块的外部声明了一个指定的参数时，在引用之前必须声明。
- 分配给指定参数的值可以是任何常数表达式。
- 不像模块参数那样，不能在语言内修改一个指定的参数。但是，可以通过SDF注解修改。
- 指定参数和模块参数是不能互相交换的。
- 模块参数不能分配一个包含指定参数的常数表达式。
- 下表给出了specparams和parameters的不同之处。

Verilog HDL数据类型
--参数

指定参数	模块参数
使用关键字specparam	使用关键字parameter
在一个模块内或者指定的块内声明	在一个指定的块外声明
只能在一个模块内或者指定的块内使用	不能在指定的块内使用
可以被分配指定参数和模块参数	不能分配指定参数
使用SDF注解覆盖值	使用defparam或者例化声明参数值传递来覆盖值

Verilog HDL数据类型
--参数

声明specparam的Verilog HDL描述

```
module test;
reg clk;
specify
  specparam low=10,high=10;
endspecify
initial clk=0;
always
 begin
   #low clk=1;
   #high clk=0;
 end
endmodule
```

Verilog HDL数据类型
--参数

声明parameter和specparam的Verilog HDL描述

```
module RAM16GEN (output [7:0] DOUT, input [7:0] DIN,
                                 input [5:0] ADR,input WE, CE);
specparam dhold = 1.0;
specparam ddly = 1.0;
parameter width = 1;
parameter regsize = dhold + 1.0;     //非法 – 不能将指定参数分配
                                       分配给模块参数
endmodule
```

Verilog HDL数据类型
--Verilog HDL名字空间

在Verilog HDL中有几类名字空间，其中:

- 两类为全局名字空间。
- 其余为局部名字空间。

Verilog HDL数据类型
--Verilog HDL名字空间

全局名字空间

- 定义名字空间包括所有module（模块）、marcomodule（宏模块）、primitive（基本原语）的定义。
- 一旦某个名字用于定义一个模块、宏模块或基本原语，那么该名字将不能再用于声明其他模块、宏模块或基本原语，也就是这个名字在定义名字空间具有唯一性。
- 文本宏名字空间也是全局的。由于文本宏名由重音符号（`）引导，因此它与别的名字空间有明显的区别。文本宏名的定义逐行出现在设计单元源程序中，它可以被重复定义，也就是同一宏名后面的定义将覆盖其先前的定义。

Verilog HDL数据类型
--Verilog HDL名字空间

局部名字空间

- 局部名字空间包括：
  - block（块）
  - module（模块）
  - generate block（生成块）
  - port（端口）
  - specify block（延时说明块）
  - attribute（属性）
- 一旦在这几个名字空间中的任意一个空间内定义了某个名字。就不能在该空间中重复定义这个名字（即具有唯一性）。

Verilog HDL数据类型
--Verilog HDL名字空间

语句块名字空间

包括：语句块名、函数名、任务名、参数名、事件名和变量类型声明。

注：

其中变量类型声明包括：**reg、integer、time、real和realtime**声明。

Verilog HDL数据类型
--Verilog HDL名字空间

模块名字空间

包括：函数名、任务名、例化名（模块调用名）、参数名、事件名和网络类型声明与变量类型声明。

注：

其中网络类型声明包括：**wire、wor、wand、tri、trior、triand、tri0、tri1、trireg、supply0和supply1。**

Verilog HDL数据类型
--Verilog HDL名字空间

生成块名字

包括：函数、任务、命名的块、模块例化、生成块、本地参数、命名事件、genvars、网络类型的声明和变量类型的声明。

端口名字空间

用于连接两个不同名字空间中的数据对象。连接可以是单向的或双向的。端口名字空间是模块名字空间与语句块名字空间的交集。

Verilog HDL数据类型
--Verilog HDL名字空间

从本质上说，端口名字空间规定了不同空间中两个名字的连接类型。端口的类型声明包括input、output、inout。

只需要在模块名字空间中声明一个与端口名同名的变量或wire型数据，就可以在模块名字空间中再次引用端口名字空间中所定义的端口名。

Verilog HDL数据类型
--Verilog HDL名字空间

指定块

用来说明模块内的时序信息。sprcparams用来声明延迟常数，很象模块内的一个普通的参数，但是不能被覆盖。

指定块以specify开头，以endspecify结束。

属性

是由符号（* *）所包含的语言结构。属性名只能在属性名字空间中被定义和使用，不能在属性名字空间中被定义其他任何名字。