1. wire和reg型变量可以同类型赋值。过程块内，wire型可赋给reg型，反之不行；块外reg型可赋给wire型，反之不行。但always的敏感信号、case检测信号无要求。
2. 输入和双向端口不能声明为寄存器型，如果信号的数据类型没有定义，则综合器将其默认为wire型。
3. “always”块语句与assign语句是并发执行的，assign语句一定要放在“always”块语句之外。
4. 抽象级别有五种：系统级、算法级、RTL级、门级、开关级，系统级、算法级和RTL级是属于行为级的，门级是属于结构级的。
5. Verilog HDL提供了四种方式描述具体的逻辑行为：行为级描述方式、数据流（寄存器传输级/RTL）描述方式、结构级描述方式、开关级描述方式。
6. 行为级描述方式到底和数据流描述方式的本质区别在什么地方？  
   （1）行为级描述中，包含一些设计元素，在FPGA内无法找到相应  
   的逻辑单元来实现这些设计元素。而数据流描述中，只包含可以在FPGA内实现的设计元素。  
   （2）行为级描述，一般只用于对设计进行仿真，也就是生成对设  
   计的测试向量，通过特定的仿真软件来测试设计有无设计缺陷。但是，不能转换成FPGA的具体物理实现。而数据流描述，用于对设计进行综合，最后下载到FPGA器件进行具体的物理实现。
7. 门级元件中，在端口列表中**输出**信号列在最前面。
8. Case语句：系统级，assign语句：RTL级。
9. 各assign语句是并行执行的，即各语句的执行与语句的编写顺序无关。
10. $开头的任务为系统任务，例如$time返回当前的仿真时刻。
11. 采用的描述级别越高，设计越容易，程序代码越简单，但耗用器件资源更多。对特定综合器，可能无法将某些抽象级别高的描述转化为电路。
12. 标识符可由字母、数字、下划线和**$**符号构成，但第一个字符  
    必须是**字母或下划线**，不能是数字或$符号，在Verilog HDL中变量名是区分大小写的。
13. （1）**参数定义**（用一个标识符来代表一个常量）的格式：  
    parameter 参数名1=表达式，参数名2=表达式，……；

注：每个赋值语句的右边必须为常数表达式，即只能包含数字或先  
前定义过的符号常量。  
（2）**宏定义**（用一个简单的宏名来代替一个复杂的表达式）的格式：  
’define 标志符（即宏名）字符串（即宏内容）

1. 时钟沿触发的always块必须是单时钟的，且任何异步控制输入（通常是复位或置位信号）必须在控制事件列表中列出。  
   例： always @(posedge clk or negedge set or negedge reset)
2. 在无时钟的always块中，若有的输出变量被赋了某个信号变量值，而该信号变量并未在该always块的电平敏感控制事件中列出，则会在综合中生成不想要的**锁存器**。
3. 在时钟沿触发的always块中，如果用非阻塞赋值语句对reg型变量赋值；或者当reg型变量经过多次循环其值仍保持不变，则会在综合中生成**触发器**。例：

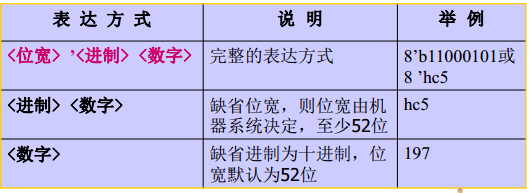
（1）用reg型变量生成触发器：

reg out；  
always @(posedge clk) //沿触发  
out <= d ; //D触发器

（2）用reg型变量生成组合逻辑：

reg out；  
always @(d) //电平触发  
out <= d ;

1. 整常数表达方式：



这里位宽指对应二进制数的宽度。

1. X（不定值）与Z（高阻值）  
   如8’b1001xxxx = 8 ’h9x，8’b1010zzzz = 8 ’haz。  
   当用二进制表示时，已标明位宽的数若用x或z表示某些位，则只有在**最左边**的x或z具有扩展性。

如8’bzx = 8’bzzzz\_zzzx，8’b1x = 8’b0000\_001x  
“？”是z的另一种表示符号，建议在case语句中使用？表示高阻态z。

1. 负数：在位宽前加一个减号。

如：-8’d5 //5的补数， = 8’b11111011

1. 减号不能放在位宽与进制之间，也不能放在进制与数字之间。
2. 为提高可读性，在较长的数字之间可用下划线\_隔开，但不可以用在<进制>和<数字>之间。
3. 数据类型有19种，常用的有3种：  
   （1）网络型（nets type）：输出始终随输入的变化而变化的变量，如wire型。  
   （2）寄存器型（register type）：对应具有状态保持作用的电路元件（如触发器、寄存器等）,常用来表示过程块语句（如initial，always，task，function）内的指定信号，如reg、integer、time、real型。  
   （3）数组（memory type）
4. wire型变量  
   （1）以assign语句赋值，可用做任何方程式的输入，或“assign”语句和实例元件的输出。

（2）定义m个位宽为n的变量：

wire[n-1:0] 数据名1,数据名2, ……,数据名m;//最高位是n-1

wire[0:n-1] 数据名1,数据名2, ……,数据名m;//最高位是0

1. register型变量
2. 必须通过过程赋值语句赋值，不能通过assign语句赋值。在过程块内被赋值的每个信号必须定义成register型。
3. reg型变量既可生成触发器，也可生成组合逻辑；wire型变量只能生成组合逻辑。
4. Memory型变量（数组）

（1）由若干个相同宽度的reg型向量构成的数组。

（2）定义由m个位宽为n的储存单元构成的数组：

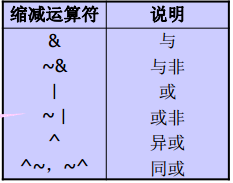
reg[n-1:0] 存储器名 [m-1:0];

1. 等于运算符(= =)和全等运算符(= = =)  
    使用等于运算符时，两个操作数必须逐位相等,结果才为1；若  
   某些位为x或z，则结果为x。  
    使用全等运算符时，若两个操作数的相应位完全一致（如同  
   是1，或同是0，或同是x，或同是z）,则结果为1；否则为0。

如：a=00x1,b=00x1,则：(a==b)=x,(a===b)=1。

1. 缩减运算符

对单个操作数进行递推运算,即先将操作数的最低位与第二位进行与、或、非运算，再将运算结果与第三位进行相同的运算，依次类推，直至最高位。运算结果缩减为1位二进制数。



例如：&a;

1. 条件语句
2. 分类：if-else语句和case语句  
    它们都是顺序语句，应放在“always”块内。
3. If-else语句

若表达式的值为0、或z，则判定的结果为“假”；若为1，则结果为“真”。

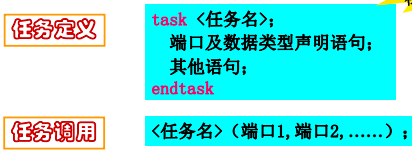
1. case语句

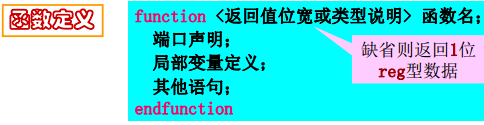
在case语句中，分支表达式每一位的值都是确定的（为0或1）；  
在casez语句中，若分支表达式某些位的值为高阻值z，则不考虑对这些位的比较；  
在casex语句中，若分支表达式某些位的值为z或不定值x，则不考虑对这些位的比较。  
在分支表达式中，可用“？”来标识x或z。

1. 循环语句
2. 分类：4类

for语句、while语句——同C语言  
repeat语句——连续执行一条语句n次，例：repeat(3);  
forever语句——无限连续地执行语句，可用disable语句中断，一般用在initial语句块中

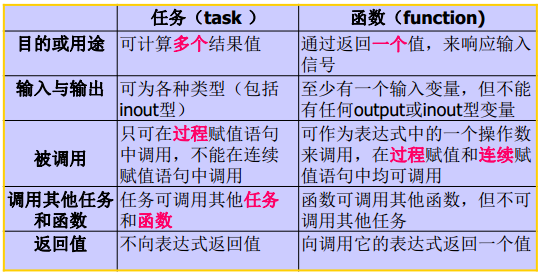
1. 任务与函数的区别



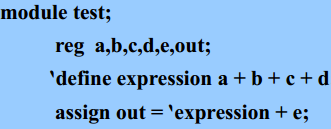


IMG_256

任务的定义与调用必须在一个module模块内，函数不一定在一个模块内调用。



1. ‵define语句（宏定义语句）  
   （1）定义：用一个指定的标志符（即宏名）来代表一个字符串（即宏内容）
2. 格式（行末无分号）：‵define 标志符（即宏名）字符串（即宏内容）



1. ‵include语句（文件包含语句）  
   （1）定义：一个源文件可将另一个源文件的全部内容包含进来。

（2）格式：‵include “文件名”

1. ‵timescale语句（时间尺度语句）  
   （1）定义：用于定义跟在该命令后模块的时间单位和时间精度。  
   （2）格式：‵timescale <时间单位> / <时间精度>  
   时间单位——用于定义模块中仿真时间和延迟时间的基准单位；  
   时间精度——用来声明该模块的仿真时间和延迟时间的精确程度。  
   （3）注意：

<1> 在同一程序设计里，可以包含采用不同时间单位的模块，此时用最小的时间精度值决定仿真的时间单位。

<2> 时间精度值不能大于时间单位值，用来说明时间单位和时间精度  
参量值的数字必须是整数，其有效数字为1、10、100。

1. 设计技巧

（1）一个变量不能在多个always块中被赋值

（2）在always块语句中，当敏感信号为两个以上的时钟边沿触发信号时，不要使用多个if语句

（3）当输出信号为总线信号时，一定要在I/O说明中指明其位宽