# Verilog

Verilog HDL模型可以是实际电路的不同级别的抽象。抽象级别可分为五级：

**系统级**(system level): 用**高级语言结构（如case语句）**实现的设计模块外部性能的模型；

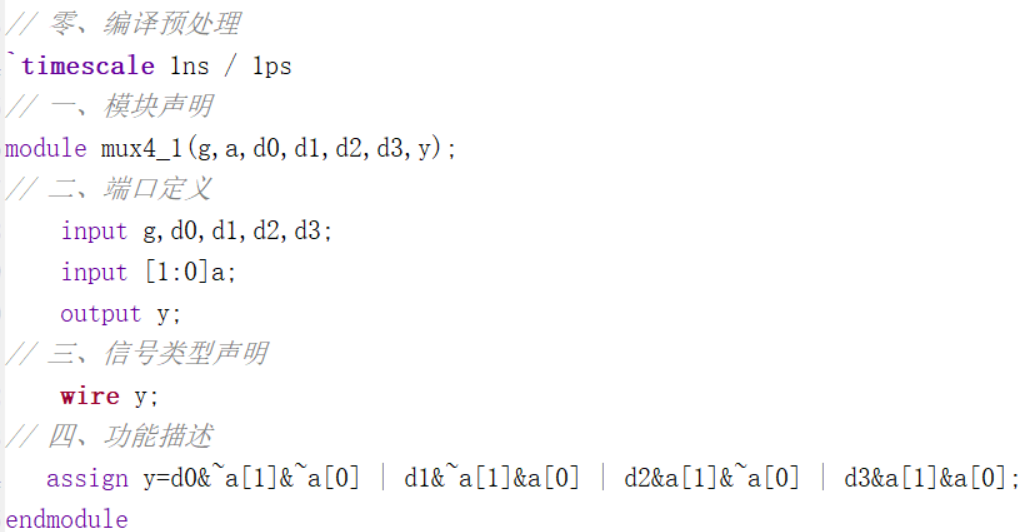
**算法级**(algorithmic level): 用高级语言结构实现的设计算法模型**（写出逻辑表达式）**；

**RTL级**(register transfer level): 描述**数据在寄存器之间流动和如何处理这些数据的模型**；

**门级**(gate level): 描述逻辑门（如**与门、非门、或门、与非门、三态门**等）以及逻辑门之间连接的模型；

**开关级**(switch level): 描述器件中**三极管和储存节点及其之间连接**的模型。

Verilog程序的四部分



一、 Verilog HDL的门级描述

// Verilog HDL的结构级描述（门级/调用已设计好的模块，开关级）

门类型的关键字有26个，常用的有9个：

not，and，nand，or，nor，xor，xnor，buf，

bufif1，bufif0，notif1，notif0（各种三态门）

门类型关键字 <例化的门名称> ( <端口列表>)；

注1：在 端口列表中**输出信号列在最前面**；

注2：门级描述不适于描述复杂的系统！

二、Verilog HDL的行为级描述（包括系统级,算法级,RTL级）

1. 逻辑功能描述，逻辑表达式，常见于连续赋值语句（assign）——算法级

Assign out=(in1 & ~cntrl1 & ~cntrl2)

2. case语句描述（**只需知道输入与输出间的真值表！**）——系统级

3. 条件运算符描述（A?B:C），常见于连续赋值语句（assign）,**数据流风格的描述（寄存器传输级）**——RTL级

A?B:C**比调用门原语，采用逻辑表达式或 case语句描述代码更简单！但也更抽象！且耗用器件资源更多！**

四、**测试台 (*test bench*)**

（使用行为级描述）

1、没有端口（test bench为最顶层模块，不会被任何模块实例化）

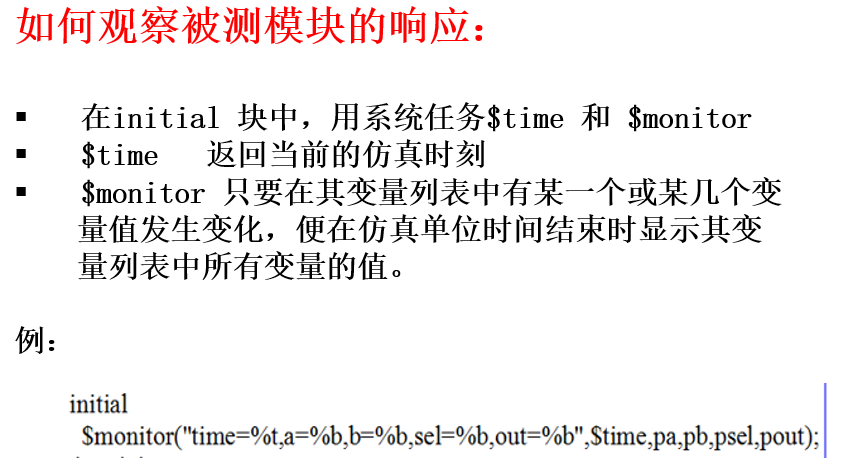
2、信号类型声明：输入reg，输出wire

3、逻辑功能描述：提供测试激励及验证机制

被测试模块的实例化

产生测试信号（initial begin…end，always/forever/repeat)常用于产生时钟信号）

观察被测模块的响应



系统任务

*$monitor*

*$display(“…”)*，字符串同上

$random

**一般用法为：**

**1、$random为系统任务，返回一个32位的带符号的随机数；**

**2、*$random % b*，其中b>0，它给出了一个范围在-b+1~b-1之间的随机数。**

**3、*{$random}%15*通过位拼接操作，产生一个0~14之间的随机数**

五、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！

**·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，……；**

**·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）**

·**‵define 标志符（即宏名）字符串（即宏内容）**

对应的宏定义参数使用：**`<宏参数名>**

* ‵define语句可以写在模块定义的**外面**或**里面**。宏名的有效范围为定义命令之后到源文件结束。
* 预处理时只是将程序中的宏名替换为字符串，不管含义是否正确。只有在编译宏展开后的源程序时才报错。
* 宏定义不是Verilog HDL语句，不必在行末加分号！如果加了分号，会连分号一起置换！

·`include ”文件名”

**- 文件包含语句——**一个源文件可将另一个源文件的全部内容包含进来。（相对路径）

**- 一个‵include语句只能指定一个被包含的文件；若要包含n个文件，需用n个‵include语句。**

**- 可将多个‵include语句写在一行；在该行中，只可出现空格和注释行。**

**‵include “aaa.v” ‵include “bbb.v” //合法！**

**（考过的）**

**·`timescale <时间单位>/<时间精度>**

**时间尺度语句——用于定义跟在该命令后模块的时间单位和时间精度。**

**·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；**

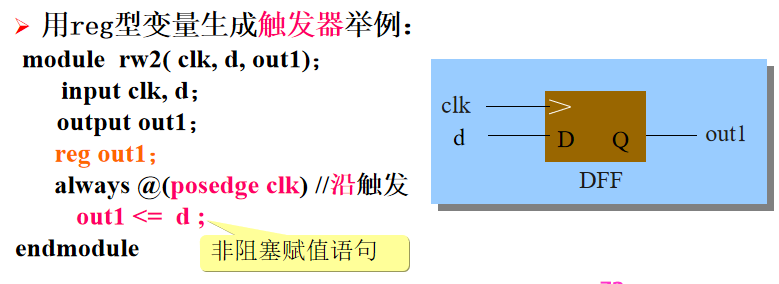
**·时间精度——用来声明该模块的仿真时间和延迟时间的精确程度。**

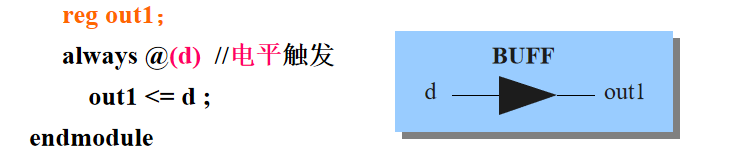
**·在同一程序设计里，可以包含采用不同时间单位的模块。此时用最小的时间精度值决定仿真的时间单位。**

**·时间精度至少要和时间单位一样精确，时间精度值不能大于时间单位值！**

* **在‵timescale语句中，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；**
* **单位为秒（s）、毫秒（ms）、微秒（us）、纳秒（ns）、皮秒（ps）、毫皮秒（fs）。**

**综合：将用HDL语言或图形方式描述的电路设计转换为实际门级电路（如触发器、逻辑门等），得到一个网表文件，用于进行适配（在实际器件中进行布局和布线）。**



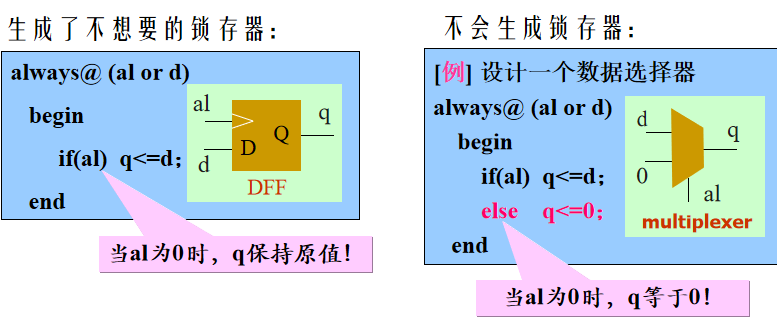


|  |  |
| --- | --- |
| **赋值语句** | 连续赋值语句 |
| 过程赋值语句 |
| **块语句** | begin\_end语句 |
| Fork…join语句 |
| **条件语句** | if\_else语句 |
| case语句 |
| **循环语句** | forever语句 |
| repeat语句 |
| while语句 |
| for语句 |
| **结构说明语句** | **initial**语句 |
| always语句 |
| task语句 |
| function语句 |

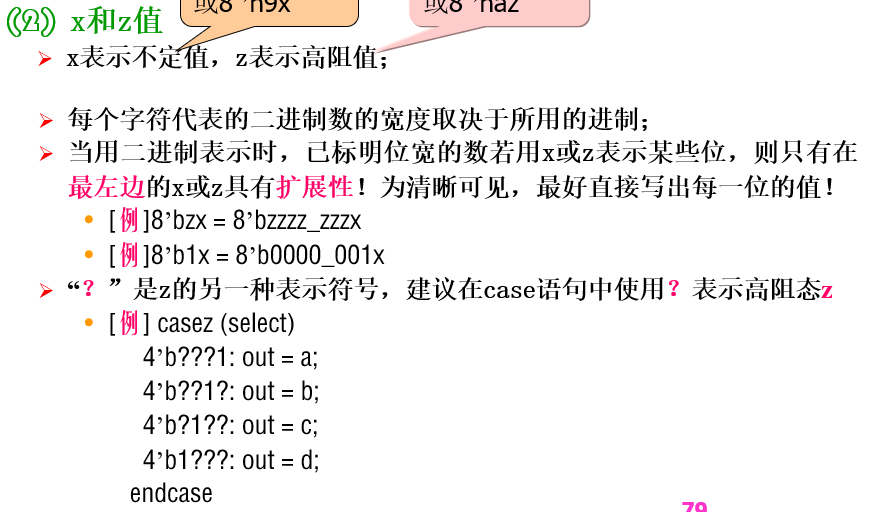
在case语句中，分支表达式每一位的值都是确定的（或者为0，或者为1）；

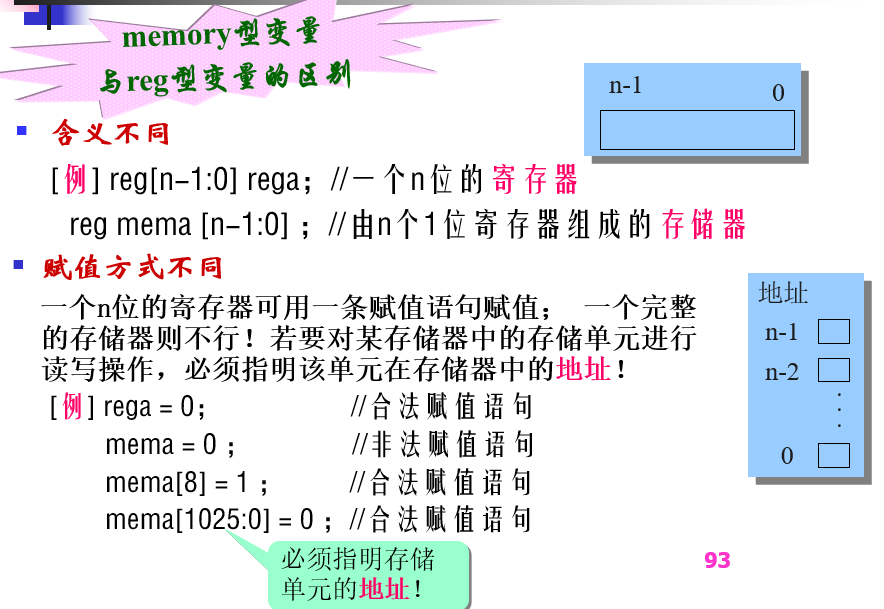
在casez语句中，若分支表达式某些位的值为高阻值z，则不考虑对这些位的比较；

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。









负数：在位宽前面加一个负号 <符号><位宽><’进制数><数值>

**-8’d5 //5的补数，= 8‘b11111011**

**8 ’ d-5 //非法格式**

模块实例化的参数传递：

*<元件名> <实例名><输入输出信号>;*

*defparam <实例名><.><参数名>;*

或

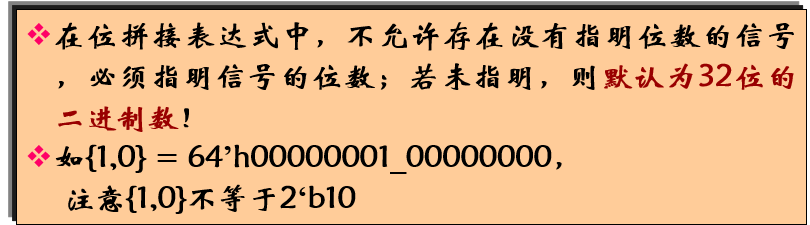
*<元件名>#(参数列表：位置映射/名称映射) <实例名><输入输出信号>;*

移位运算的注意事项：<< >>（移动n位，不足补0）

[**例**] 4’b1001>>5 = 4’b0001；4’b1001>>4 = 4’b0000（**右移位数不变，但右移的数据会丢失！**）

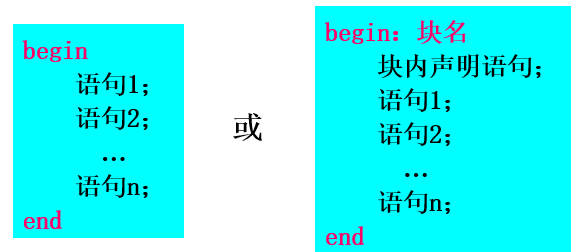
4’b1001<<1 = 5’b10010；4’b1001<<2 = 6’b100100；1<<6 = 52’b1000000（**左移会扩充位数！**）

位拼接运算符的注意事项：



*begin…end*块语句注意事项

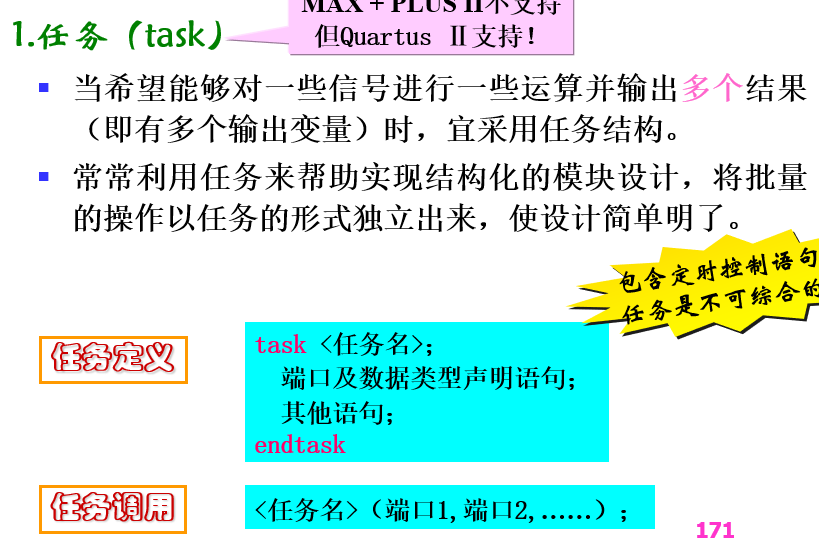
只有*begin:块名 …end*才能块内声明变量（局部变量）



always块注意事项：

**1、当always块有多个敏感信号时，一定要采用if - else if语句，而不能采用并列的if语句！否则易造成一个寄存器有多个时钟驱动，将出现编译错误**

Task块注意事项



**注0：对一些信号进行一些运算并输出多个结果（即有多个输出变量）**

**注1：任务的定义与调用必须在一个module模块内！**

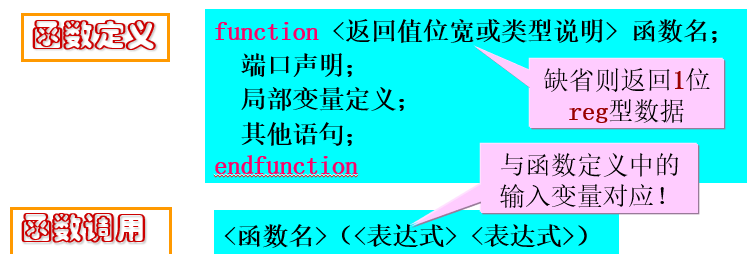
**注2：任务被调用时，需列出端口名列表，且必须与任务定义中的I/O变量一一对应**！

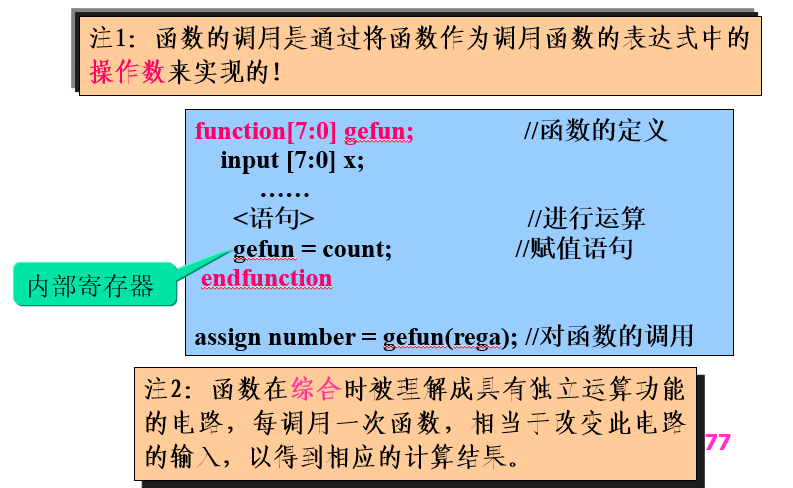
**注3：一个任务可以调用其他任务和函数。**

（不能有always块）

Function块语句的注意事项

**通过返回一个用于某表达式的值，来响应输入信号。**





注1：函数的定义不能包含任何时间控制语句——用延迟#、事件控制@或等待wait标识的语句。

注2：函数不能启动（即调用）任务

注3：定义函数时至少要有一个输入参量！且不能有任何输出或输入/输出双向变量。

注4：在函数的定义中必须有一条赋值语句，给函数中的一个内部寄存器赋以函数的结果值，该内部寄存器与函数同名

（不能有always块）



Always块里可以有task\function，不能有initial块，但是task、function里面不能有always块

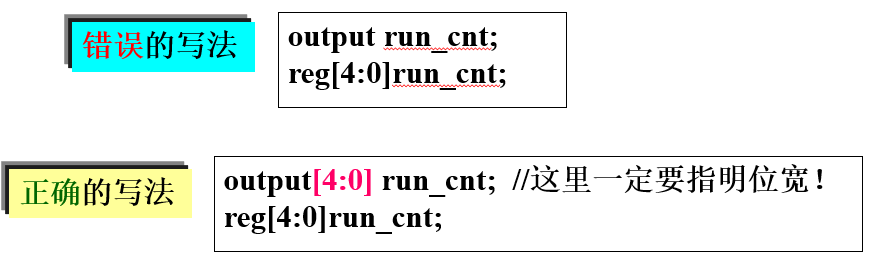
写Verilog程序的注意事项

**1．一个变量不能在多个always块中被赋值！**

2、always模块不能有反馈语句，即把定义为输出（output）的信号当作输入信号来使用（如赋值语句的右值，if的条件判断的表达式）

3、**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，应注意不要使用多个if语句！以免因逻辑关系描述不清晰而导致编译错误。 只能if…else if…else每个逻辑保证相互独立**

4、**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！否则在生成逻辑符号时，输出信号被误认为是单个信号，而没有标明位宽，就不会当成总线信号。**



**5．当要用到计数器时，一定要根据计数最大值事先计算好所需的位宽！若位宽不够，则计数器不能计到你设定的最大值，当该计数器用作分频时，则输出时钟始终为0，所设计电路将不能按预定功能正常工作！**

6、**．注意区分阻塞赋值和非阻塞赋值的区别。**

* **在一个源程序中，要么都采用阻塞赋值语句，要么都采用非阻塞赋值语句，最好不要混合使用，否则可能逻辑关系出错！**
* **为易于综合，建议均采用非阻塞赋值语句！（<=有对应的硬件表示，=没有）**