Verilog wire 和reg使用注意事项

1. 从仿真的角度来说，HDL语言面对的是编译器（如Modelsim等），相当于软件思路。  
   这时：  
   （1）wire对应于连续赋值，如assign（连续赋值语句），在assign左侧赋值。不能在always语句中赋值。  
   （2）reg对应于过程赋值，如always，initial(过程赋值语句)，在always左侧赋值。

（3）都能用于assign与always @模块表达式的右侧。

寄存器型数据保持最后一次的赋值，而线型数据需要持续的驱动。wire使用在连续赋值语句中，而reg使用在过程赋值语句（initial ，always）中。wire若无驱动器连接，其值为z（高阻或者悬空），reg默认初始值为不定值 x 。

x代表一个未被预置初始状态的变量或者是由于由两个或多个驱动装置试图将之设定为不同的值而引起的冲突型线型变量。z代表高阻状态或浮空量。

**用关键词inout声明一个双向端口, inout端口不能声明为寄存器类型，只能是wire类型。**

wire型数据常用来表示以assign关键字指定的组合逻辑信号。模块的输入输出端口类型都默认为wire型。默认初始值是z。

reg型表示的寄存器类型。always模块内被赋值的信号，必须定义为reg型，代表触发器。默认初始值是x。

2、从综合的角度来说，HDL语言面对的是综合器（如DC等），要从电路的角度来考虑。  
这时：  
wire型的变量综合出来一般是一根导线；  
reg变量在always块中有两种情况：  
(1)、always后的敏感表中是（a or b or c）形式的，也就是不带时钟边沿的，综合出来还是组合逻辑  
(2)、always后的敏感表中是（posedge clk）形式的，也就是带边沿的，综合出来一般是时序逻辑，会包含触发器（Flip－Flop）

3、在设计中，输入信号一般来说你是不知道上一级是寄存器输出还是组合逻辑输出，那么对于本级来说就是一根导线，也就是wire型。而输出信号则由你自己来决定是寄存器输出还是组合逻辑输出，wire型、reg型都可以。但一般的，整个设计的外部输出（即最顶层模块的输出），要求是寄存器输出，较稳定、扇出能力也较好。

4、有几种情况变量需要定义成wire。  
（1）assign 语句  
例如：  
reg a,b;  
wire and\_result;  
...  
assign and\_result =a&&b;  
如果把wire定义成reg。综合器会报错。

（2）元件例化时候的输出必须用wire  
例如：  
wire dout;  
  
ram u\_ram  
(  
...  
.out(dout)  
...  
);

大体上来说，wire和reg类似于C/C++的变量，但若此变量放在begin…end内，该变量就需使用reg，在begin…end之外，则使用wire。

使用wire时，需搭配assign；reg不必。  
  
input，ouput，inout预设值都是wire。

若wire和reg用错地方，compiler都会提醒，所以不必太担心。

**下面所列是常出的错误及相应的错误信息(error message)**

•用过程语句给一个wire类型的或忘记声明类型的信号赋值。

           信息：illegal …… assignment.

•将实例的输出连接到声明为register类型的信号上。

           信息：<name> has illegal output port specification.

•将模块的输入信号声明为register类型。

           信息：incompatible declaration, <signal name> ……