# Always模块

包含一个或一个以上的声明语句(如:过程赋值语句、任务调用、条件语句和循环语句等），在仿真运行的全过程中，在定时控制下被反复执行。

1.使用规则：

在always块中被赋值的只能是register型变量（如reg，integer，real，time）。

每个always块在仿真一开始便开始执行，当执行完块中最后一个语句，继续从always块的开头执行。

注1：如果always块中包含一个以上的语句，则这些语句必须放在begin\_end或fork\_join块中！

注2：always语句必须与一定的时序控制结合在一起才有用！如果没有时序控制，则易形成仿真死锁！

[例]生成一个0延迟的无限循环跳变过程——形成仿真死锁！

always areg = ~areg;

2.always块语句模板

always @ (<敏感信号表达式>)

begin

// 过程赋值语句

// if语句

// case语句

// while，repeat，for循环

// task，function调用

End

-在敏感信号表达式中应列出影响块内取值的所有信号!

-敏感信号可以为单个信号，也可为多个信号，中间需用关键字or连接！

-敏感信号不要为x或z，否则会阻挡进程！

always的时间控制可以为沿触发（常用于描述时序逻辑），也可为电平触发（常用于描述组合逻辑）。

关键字posedge表示上升沿；negedge表示下降沿。

注意：

1. 当always块有多个敏感信号时，一定要采用if - else if语句，而不能采用并列的多个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。
2. 通常采用异步清零！只有在时钟周期很小或清零信号为电平信号时（容易捕捉到清零信号）采用同步清零。

# 语句的顺序执行与并行执行

一、语句的顺序执行

在 “always”模块内，逻辑按书写的顺序执行。

注意阻塞赋值语句当本语句结束时即完成赋值操作！

二、语句的并行执行

“always”模块、“assign”语句、实例元件都是同时（即并行）执行的！

注意：

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，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。
3. 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！