一、 条件、循环、块语句（续）

（一） always 语句

1、always #<halfperiod> clk = ~clk;

2、简单的 2 分频

```
reg [7:0]   counter;
reg   tick;
always @(posedge areg)
  begin
    tick = ~tick;
    counter = counter + 1;
  end
```

3、always 的时间控制可以是沿触发也可以是电平触发的，可以单个信号也可以多个信号，中间需要用关键字 or 连接

4、沿触发的 always 块常常描述时序行为，如有限状态机，而电平触发的 always 块常常用来描述组合逻辑的行为。

5、关键词“or”也可以使用“,”来代替。

6、Verilog 语言用关键字 wait 来表示等待电平敏感的条件为真。

```
always
    wait (count_enable)     #20 count = count + 1;
```

（二） task 和 function

1、语句的不同点

任务和函数有些不同，主要的不同有以下四点：

1) 函数只能与主模块共用同一个仿真时间单位，而任务可以定义自己的仿真时间单位。
2) 函数不能启动任务，而任务能启动其它任务和函数。
3) 函数至少要有一个输入变量，而任务可以没有或有多个任何类型的变量。
4) 函数返回一个值，而任务则不返回值。

2、常量函数

常量函数实际上是一个带有某些限制的常规 Verilog 函数。这种函数能够用来引用复杂的值，因而可用来代替常量。在例 15 中我们声明了一个常量函数，它可以用来计算模块中地址总线的宽度。

[例 15] 常量函数

```
module ram ( ... ... ...) ;
parameter   RAM_DEPTH = 256 ;
```

```
input   [ clog2( RAM_DEPTH) - 1 : 0 ]   addr_bus ;   //
```

（三） 其它

在 Verilog HDL 语言中每个系统函数和任务前面都用一个标识符$来加以确认。

二、 时钟分频的另一种方法

按照指数分频的简单方法

```
reg [3:0] count;
wire clk2, clk4, clk8, clk16;
always @(posedge clk)
        begin
              count <= count+1;
        end
assign clk2 = count[0];
assign clk4 = count[1];
assign clk8 = count[2];
```

三、 调试用系统任务和常用编译预处理语句

（一） 常用系统任务语句

$monitor、$time、$readmemb、$readmemh、$stop、$random

$Random 生成有符号的随机数

（二） 编译预处理语句

define、include

（三） 时间尺度

timescale

时间精度参量是用来声明该模块的仿真时间的精确程度的；

四、 通过练习进一步熟悉 ISE 的操作

采用 1s、2s、4s 三种不同频率控制 led 灯

五、 Verilog HDL 模型的不同抽象级别

9.1 门级结构描述

9.2 Verilog HDL 的行为描述建模