     在Verilog中除了在Testbench（仿真测试激励）中使用for循环语句外，在Testbench中for语句在生成激励信号等方面使用较普遍，**但在RTL级编码中却很少使用for循环语句**。主要原因就是for循环会被综合器展开为所有变量情况的执行语句，每个变量独立占用寄存器资源，每条执行语句并不能有效地复用硬件逻辑资源，造成巨大的资源浪费。简单的说就是：for语句循环几次，就是将相同的电路复制几次，因此循环次数越多，占用面积越大，综合就越慢。

**在RTL硬件描述中，遇到类似的算法，推荐的方法是先搞清楚设计的时序要求，做一个reg型计数器。在每个时钟沿累加，并在每个时钟沿判断计数器情况，做相应的处理，能复用的处理模块尽量复用，即使所有的操作不能复用，也采用case语句展开处理。**

对于下面的for循环语句：

1 for(i=0;i<16;i++)

2 DoSomething();

可以采用如下代码实现：

reg [3:0] counter;

always @(posedge clk)

if(syn\_rst)

counter<=4'b0;

else

counter<=counter+1;

always @(posedge clk)

begin

case(counter)

4'b0000:

4'b0001:

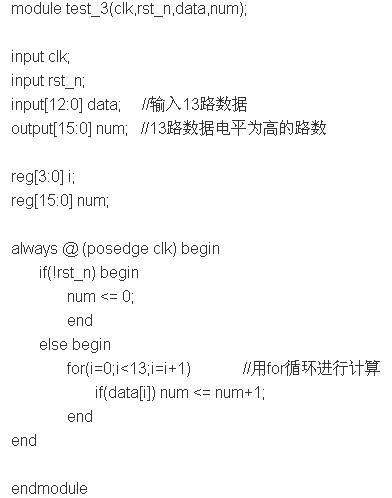
......

default:

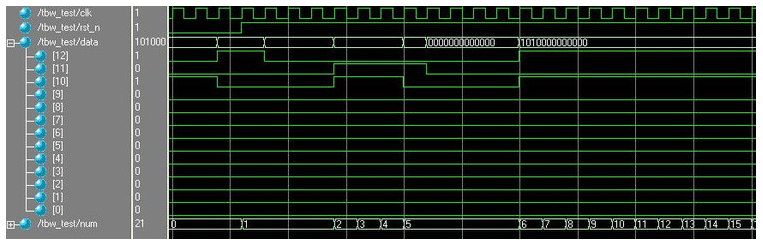
endcase

end

     另外，有几个语法的细节需要注意一下。for(i=0;i<16;i=i+1)中的i既可以是reg型的变量也可以是integer类型的变量，但是当i是reg型的变量时，需要注意因为判断语句i<16的缘故，i应定义为reg[4:0] i而不是reg[3:0] i 。由于verilog中没有自增运算符，文中提到的for语句不能写成for(i=0;i<16; i++)的形式。  
 下面简单的列举几个用for实现的程序代码：  
 示例一：



 仿真结果如下：



仿真后的结果，由于采用了非阻塞赋值语句，所以每次在always借宿后才把值付给左边的寄存器。

不过在使用了阻塞赋值语句后，得到了目的，但是由于for语句的综合效率不高，且在时序逻辑中一般采用非阻塞赋值，因此最好不能这样写

示例二：for用在纯组合逻辑中

举例：4位左移器（将低4位输入的数移到高4位）

[复制代码](javascript:void(0);)

1 //Leftshift for 4 bits

2 module For\_Leftshift(

3 input wire [3:0]inp,

4 input wire L\_EN,

5 output reg [7:0]result

6 );

7

8 integer i;

9 always@(inp or L\_EN)

10 begin

11 result[7:4] = 0;

12 result[3:0] = inp;

13 if(L\_EN == 1)

14 begin

15 for(i=4;i<=7;i=i+1)

16 begin

17 result[i] = result[i-4];

18 end

19 result[3:0] = 0;

20 end

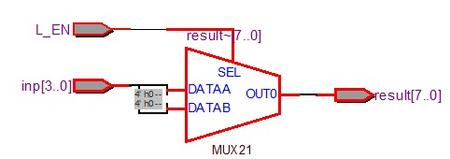
21 end

22

23 endmodule

[复制代码](javascript:void(0);)

综合结果（RTL视图，实际是一个4位选择器）

[](http://download.bbs.ednchina.com/images/attachments/201403/original/5000179125_TIME_1395720514102.jpg)

示例三：for不仅可以用在组合逻辑中，而且还可以用在时序逻辑中，用于在1个周期类完成整个for循环。

举例：在一个周期类完成对输入总线中高电平位的计数，则利用for循环实现加法器

[复制代码](javascript:void(0);)

1 module For\_Counter(

2 input wire clk,

3 input wire rst\_n,

4 input wire [12:0] data,

5 output wire [3:0] numout

6 );

7 integer i;

8 reg[3:0] num;

9

10 always @(posedge clk)

11 begin

12 if(!rst\_n)

13 num = 0;

14 else

15 begin

16 for(i=0;i<13;i=i+1)

17 if(data[i]) num = num + 1;

18 end

19 end

20

21 assign numout = num;

22

23 endmodule

[复制代码](javascript:void(0);)

综合结果（RTL视图，加法器+触发器）

[2.jpg](http://download.bbs.ednchina.com/images/attachments/201403/original/5000179125_TIME_1395720542725.jpg)

综上，可以看出for循环是可以综合的，而且效率很高。但所消耗的逻辑资源较大。在对速度（时钟周期数）要求不是很高的情况下，可以多用几个时钟周期完成任务，而没有必要用for循环来做。