（1）所有综合工具都支持的结构：always，assign，begin，end，case，wire，tri，aupply0，supply1，reg，integer，default，for，function，and，nand，or，nor，xor，xnor，buf，not，bufif0，bufif1，notif0，notif1，if，inout，input，instantitation，module，negedge，posedge，operators，output，parameter。

（2）所有综合工具都不支持的结构：time，defparam，$finish，fork，join，initial，delays，UDP，wait。

（3）有些工具支持有些工具不支持的结构：casex，casez，wand，triand，wor，trior，real，disable，forever，arrays，memories，repeat，task，while。

建立可综合模型的原则

要保证 Verilog HDL 赋值语句的可综合性，在建模时应注意以下要点：

（1）不使用 initial。

（2）不使用#10。

（3）不使用循环次数不确定的循环语句，如 forever、while 等。

（4）不使用用户自定义原语（UDP 元件）。

（5）尽量使用同步方式设计电路。

（6）除非是关键路径的设计，一般不采用调用门级元件来描述设计的方法，建议采用行为语句来完成设计。

（7）用 always 过程块描述组合逻辑，应在敏感信号列表中列出所有的输入信号。

（8）所有的内部寄存器都应该能够被复位，在使用 FPGA 实现设计时，应尽量使用器件的全局复位端作为系统总的复位。

（9）对时序逻辑描述和建模，应尽量使用非阻塞赋值方式。对组合逻辑描述和建模，既可以用阻塞赋值，也可以用非阻塞赋值。但在同一个过程块中，最好不要同时用阻塞赋值和非阻塞赋值。

（10）不能在一个以上的 always 过程块中对同一个变量赋值。而对同一个赋值对象不能既使用阻塞式赋值，又使用非阻塞式赋值。

（11）如果不打算把变量推导成锁存器，那么必须在 if 语句或 case 语句的所有条件分支中都对变量明确地赋值。

（12）避免混合使用上升沿和下降沿触发的触发器。

（13）同一个变量的赋值不能受多个时钟控制，也不能受两种不同的时钟条件（或者不同的时钟沿）控制。

（14）避免在 case 语句的分支项中使用 x 值或 z 值。

不可综合 verilog 语句 2009-04-14 19:33

1、initial

只能在 test bench 中使用，不能综合。（我用 ISE9.1 综合时，有的简单的 initial 也可以综合，不知道为什么）

2、events

event 在同步 test bench 时更有用，不能综合。

3、real

不支持 real 数据类型的综合。

4、time

不支持 time 数据类型的综合。

5、force 和 release

不支持 force 和 release 的综合。

6、assign 和 deassign

不支持对 reg 数据类型的 assign 或 deassign 进行综合，支持对 wire 数据类型的 assign 或 deassign 进行综合。

7、fork join

不可综合，可以使用非块语句达到同样的效果。

8、primitives

支持门级原语的综合，不支持非门级原语的综合。

9、table

不支持 UDP 和 table 的综合。

10、敏感列表里同时带有 posedge 和 negedge

如：always @(posedge clk or negedge clk) begin...end

这个 always 块不可综合。

11、同一个 reg 变量被多个 always 块驱动

12、延时

以#开头的延时不可综合成硬件电路延时，综合工具会忽略所有延时代码，但不会报错。

如：a=#10 b;

这里的#10 是用于仿真时的延时，在综合的时候综合工具会忽略它。也就是说，在综合的时候上式等同于 a=b;

13、与 X、Z 的比较

可能会有人喜欢在条件表达式中把数据和 X(或 Z)进行比较，殊不知这是不可综合的，综合工具同样会忽略。所以要确保信号只有两个状态：0 或 1。

如：

```
1 module synthesis_compare_xz (a,b);
2 output a;
3 input b;
4 reg a;
5
6 always @ (b)
7 begin
8   if ((b == 1'bz) || (b == 1'bx)) begin
9     a = 1;
10   end else begin
11     a = 0;
12   end
13 end
14
15 endmodule
```