# 如何在SystemVerilog中手动设置随机化seed

　　以产生受约束的随机激励是sv验证语言中最主要的feature，这里有一个常常会被验证工程师忽视的问题，就是[随机化](https://www.baidu.com/s?wd=%E9%9A%8F%E6%9C%BA%E5%8C%96&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1d9m1mdPHPBPjbLPHDznH040ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3En1TknHcknWbLrHTknWcvPW6Y" \t "https://zhidao.baidu.com/question/_blank)种子（seed）。  
我们知道，用verilog里面的$random或者sv里面的$urandom产生的都只是[伪随机数](https://www.baidu.com/s?wd=%E4%BC%AA%E9%9A%8F%E6%9C%BA%E6%95%B0&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1d9m1mdPHPBPjbLPHDznH040ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3En1TknHcknWbLrHTknWcvPW6Y" \t "https://zhidao.baidu.com/question/_blank)，也就是说，如果不改变seed，每次仿真产生的随机数都一样。  
sv的受约束的[随机化](https://www.baidu.com/s?wd=%E9%9A%8F%E6%9C%BA%E5%8C%96&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1d9m1mdPHPBPjbLPHDznH040ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3En1TknHcknWbLrHTknWcvPW6Y" \t "https://zhidao.baidu.com/question/_blank)方法与上述情况其实也有点相同。sv中，每个对象维持自身的内部RNG，排他地用于randomize()方法，这使得对象的[随机化](https://www.baidu.com/s?wd=%E9%9A%8F%E6%9C%BA%E5%8C%96&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1d9m1mdPHPBPjbLPHDznH040ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3En1TknHcknWbLrHTknWcvPW6Y" \t "https://zhidao.baidu.com/question/_blank)保持各自独立。当生成对象时，创建它的线程的RNG的下一个值被用于设置成它的RNG的随机化种子。此时对象的new函数()默认的seed为1，如果不改变seed的值，则每次run仿真时，仍旧会产生相同的激励数据。  
因此，我们需要手动设置new()函数中的随机化seed，使得每次run仿真时可以得到真正意义上的随机激励。  
手动设置对象RNG的随机化seed的方法是：使用srandom()将种子传给随机的变量seed，这能确保在任意类[成员变量](https://www.baidu.com/s?wd=%E6%88%90%E5%91%98%E5%8F%98%E9%87%8F&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1d9m1mdPHPBPjbLPHDznH040ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3En1TknHcknWbLrHTknWcvPW6Y" \t "https://zhidao.baidu.com/question/_blank)被randomize之前，为对象的RNG设置新的随机化seed。举例如下：  
  
1 class Packet;  
2 rand bit[15:0] header;  
3 ...  
4 function new (int seed);  
5 this.srandom(seed);  
6 ...  
7 endfunction  
8 endclass  
  
　　这样，我们便从外部对RNG设置新的随机化seed了：  
　　1 Packet p = new(200); //create p with seed 200.  
2 p.srandom(300); //re-seed p with seed 300.  
　　实际代码中，我们可以将seed[宏定义](https://www.baidu.com/s?wd=%E5%AE%8F%E5%AE%9A%E4%B9%89&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1d9m1mdPHPBPjbLPHDznH040ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3En1TknHcknWbLrHTknWcvPW6Y" \t "https://zhidao.baidu.com/question/_blank)为不同的值，还可以使用系统时间作为seed，我们先定义变量seed：  
  
1 module test ;  
2 integer seed ;  
3 initial begin  
4 if(!$value$plusargs("seed=%d",seed))  
5 seed = 10 ;  
6 ...  
7 end  
8 endmodule  
  
　　使用仿真命令即可将系统时间作为seed：  
　　vcs -R test.v +plusargs\_save +seed=`date +%N  
  
　　另外一种经常用到的方法是只需在仿真命令中加入+ntb\_random\_seed\_automatic，代码中不需要出现变量seed，只需要有随机约束:  
  
　　1 `timescale 1ns/1ns  
 2 program test ;  
 3 integer i ;  
 4 class rc ;  
 5 rand int a ;  
 6 constraint con {a >0;}  
 7 endclass  
 8   
 9 initial begin  
10 rc ua = new();  
11 for(i=0;i<10;i++) begin  
12 ua.randomize();  
13 $display("%d",ua.a);  
14 end  
15 $display("%d",$urandom);  
16 end  
17 endprogram  
  
　　仿真命令如下：  
　　vcs -sverilog -R test.sv +plusarg\_save +ntb\_random\_seed\_automatic