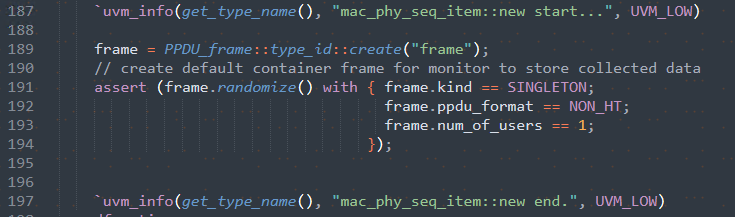
## 谨慎在post\_randomize中对其他对象进行create和randomize

曹改建

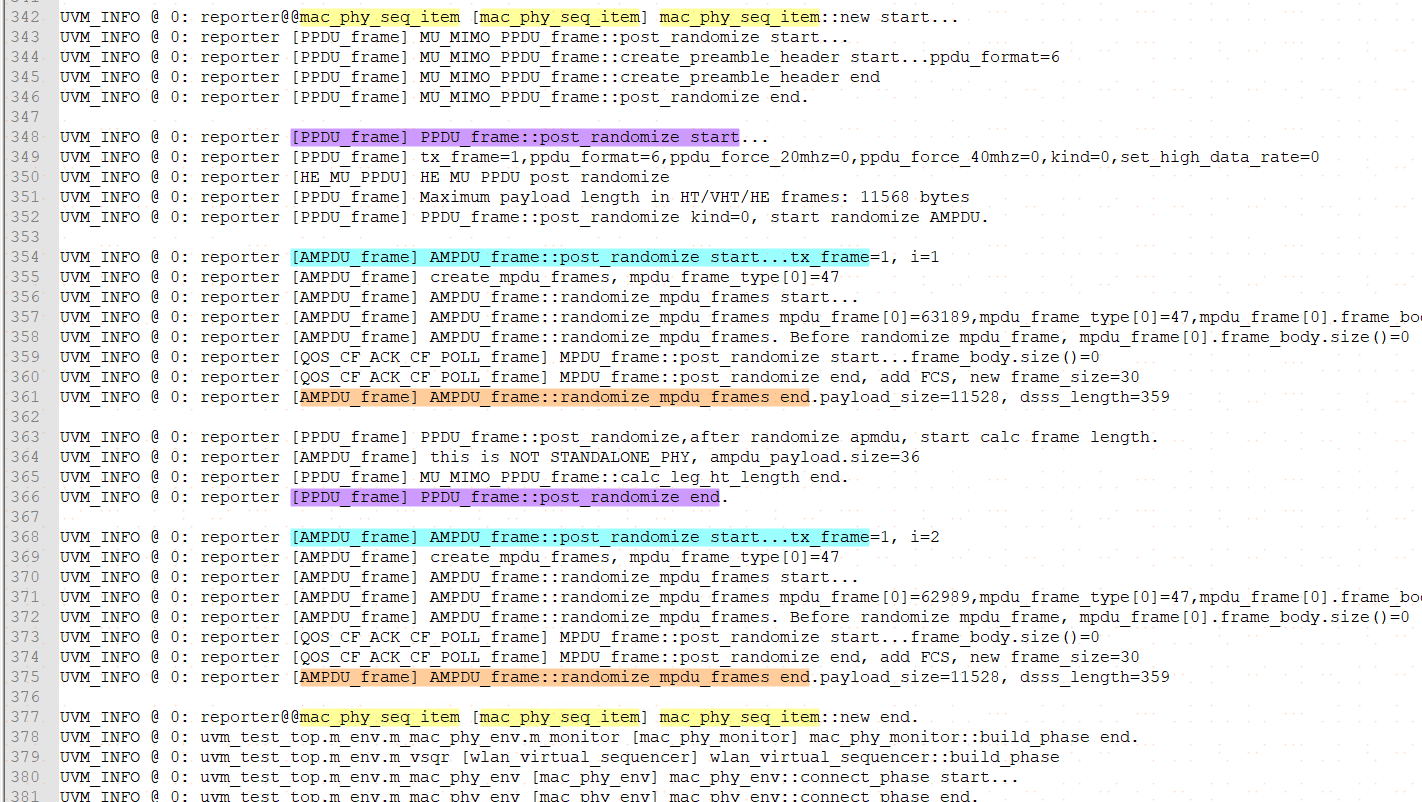
2022.10

#### 问题产生背景

*在执行现有验证脚本时（比如test\_mac\_tx\_data\_frame，其他case也有），会发现在用例case*进行run phase之前，会先进行一次PPDU frame的创建和随机化。该过程在mac\_phy\_seq\_item中进行，如下图所示。

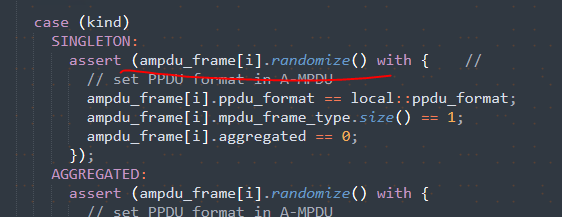


该过程会依次对PPDU\_frame、AMPDU\_frame进行随机化。仔细分析会发现，AMPDU\_frame的post\_randomize被调用了2次，如下图中的行354~361、行368~375，而AMPDU\_frame的pre\_randomize仅调用了1次（下图未显示）。



#### 原因分析

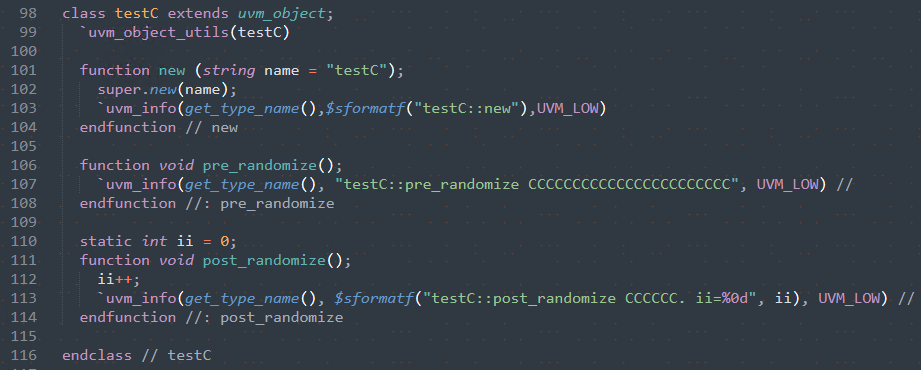
分析代码逻辑，原作者在PPDU的post\_randomize函数对AMPDU\_frame进行了随机化，如下截图所示，而正是该操作导致了重复调用AMPDU\_frame的post\_randomize操作。



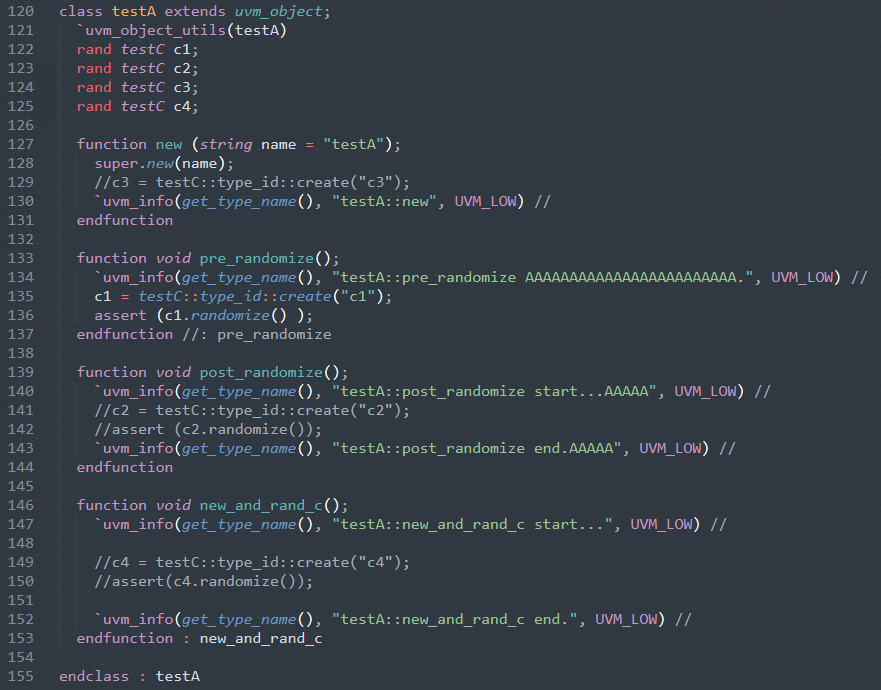
#### 简单示例说明

下面通过一段简单的代码模拟该过程，并提供几种不同的处理思路。

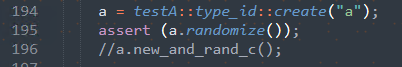
1. 定义一个testC类（模拟AMPDU）：



1. 定义一个testA类（模拟PPDU），内部定义4个testC变量c1~c4，分别用于不同场景的模拟：

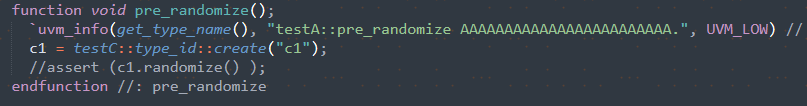


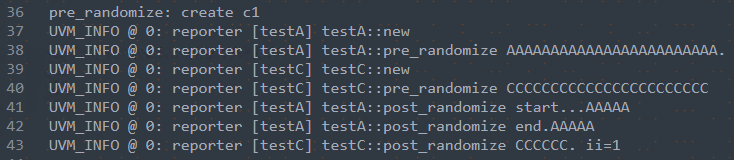
（3）在外部创建一个testA的实例a，并进行随机化：



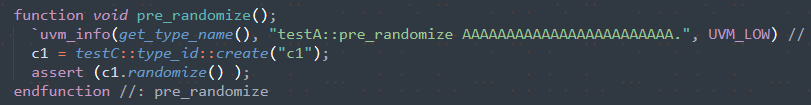
不同的场景，结果不同：

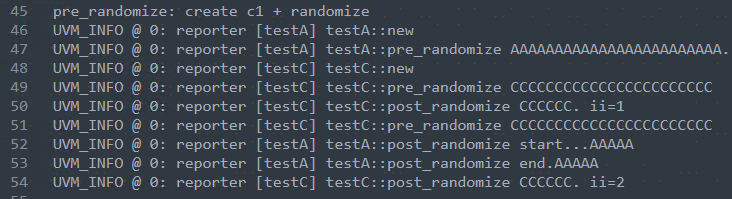
1. 放开c1的代码，仅在pre\_randomize中进行c1的create操作，不进行randomize，结果如下。此时执行顺序是：A.pre -> C.pre -> A.post -> C.post，randomize各函数只执行一次。虽然没有显式调用c1.randomize，但C的随机化函数仍被调用。



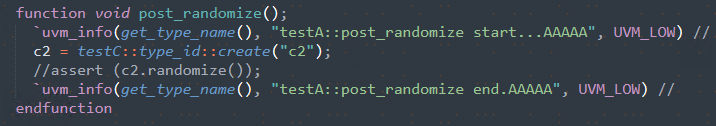


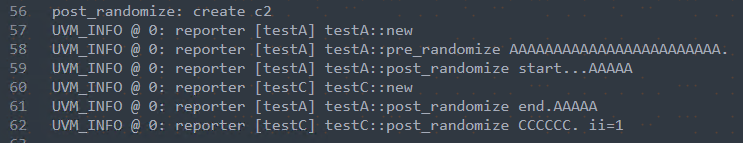
1. 放开c1的代码，在pre\_randomize中进行c1的create操作和randomize，结果如下。此时执行顺序是：A.pre -> C.pre -> C.post -> C.pre -> A.post -> C.post，与（1）对比，多了一次C的pre/post调用。



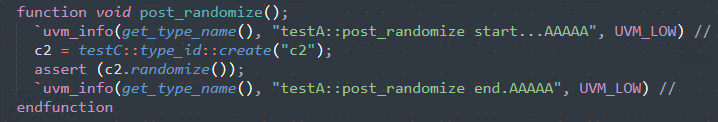


1. 注释掉c1代码，单独放开c2的代码，仅在post\_randomize中进行c2的create操作，不进行randomize，结果如下。此时执行顺序是：A.pre -> A.post -> C.post，C.pre函数不会调用。



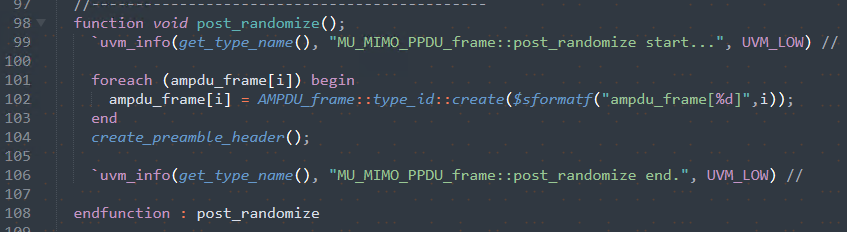


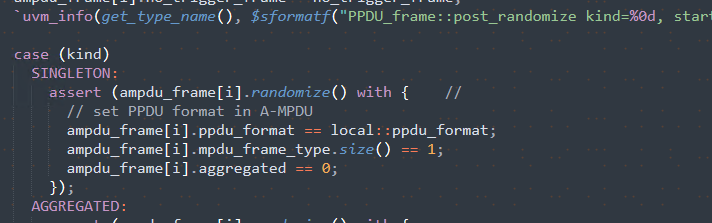
1. 单独放开c2的代码，在post\_randomize中进行c2的create + randomize操作，结果如下。此时执行顺序是：A.pre -> A.post\_start -> C.pre -> C.post -> A.post\_end -> C.post，C.post被重复调用。



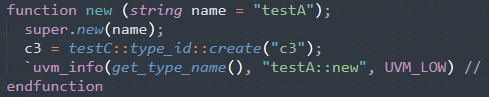


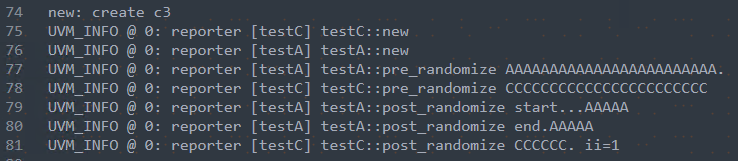
目前CEVA代码即使用了该方式：在PPDU的父类MU\_MIMO\_PPDU的post函数中进行了AMPDU.create，在PPDU自己类的post函数中进行了AMPDU.randomize。



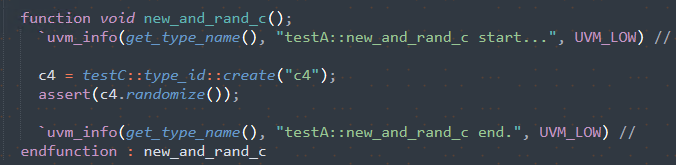


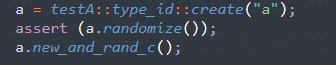
1. 单独放开c3的代码，在new中进行c3的create 操作，结果如下。此时执行顺序是：A.pre -> C.pre -> A.post -> C.post。

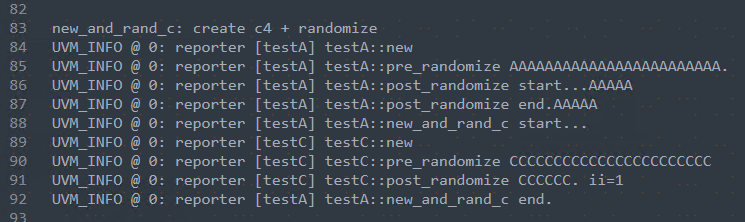




1. 单独放开c4的代码，在单独的函数中进行c4的create和randomize 操作，结果如下。此时执行顺序是：A.pre -> A.post -> C.pre -> C.post。







注意：

1. 上述现象在非UVM框架下也会出现；
2. 将create替换为new，现象不变。

#### 可能存在的问题和后续处理

重复调用post\_randomize，是否会产生问题与实际代码的处理逻辑有关。如果在该函数内部进行了计数器的更新操作，或者其他的修改操作，可能会导致逻辑错误。

具体到本文讨论的问题，由于函数代码较长，内部处理相互交叉，逻辑较复杂，未再进一步分析。建议暂时先不修改原有代码，后续新增的代码需要小心处理。

#### 讨论过程

本文问题历时2天分析和定位，先后与步荣、菱阳、天龙等多位同事讨论交流，期间碰撞出了较多的火花，也加深了对SV的理解。