MCDF 实验 3

实验 3 的部分我们将主要就随机约束和环境结构做实践。在这一个试验中，同学们将升级实验 2 部分中对 generator 和 initiator 之间的数据生成和数据传输的处理，同时我们也将完善何时结束测试，将其主动权交于 generator 而不再是 test 组件。在组件结构实践部分中，同学们将在原有的 initiator、generator、 agent 和 test 组件的基础上再认识 monitor 和 checker，并且使其构成一个有机的整体，最终可以通过在线比较数据的方式完成对 MCDT 的测试。

TB1.随机约束

为了使同学们更早习惯各个验证文件独立放置的原则，我们已经先将 chnl_pkg1.sv 文件和 tb1.sv 文件独立开来，所以 tb1 需要编译两个文件即 chnl_ pkg1.sv 和 tb1.sv。在这个试验中我们会进一步了解随机约束在类中定义方式、如何随机化对象、随机种子的使用方法、对象的产生等等。接下来,请同学们按照实验要求开始练习吧。

要求 1.1.

我们继承了大部分实验 2 的代码，包括 chnl_basic_test 类,而对于这个类所需要生成的数据包我们提出了新的约束要求。需要注意的是，与实验 2 不同的是，这次数据类 chnl_trans 的定义发生了很大的变化，它不再只局限于包含一个数据，而是多个数据，同时跟数据之间时间间隔的控制变量也在该类中声明为随机成员变量，那么请按照代码中具体的约束来定义 chnl_basic_test 类，注意该代码的修改需要在 chnl_trans 类中实现，因为目前的代码结构只有 chnl_trans 类的更新是较为合适的办法。

要求 1.2.

我们需要将原本在 chnl_root_test 类中用来结束仿真的$finish()变迁到 generator 中，那么请将它放置到合适的地方，然后由 generator 来结束仿真吧。

要求 1.3。

同学们尝试着多次重新启动仿真，可以使用"restart" 命令来重启，再对比连续两次生成的随机数据，看看它们之间是否相同呢?然后再在仿真器命令行处使用命令

"vsim -novopt -solvefaildebug -sv_seed 0 work.tb1"

来加载仿真。这里我们多传递了两个必须的仿真参数，-solvefaildebug.是为了调试随机变量的，而-SV seed NUM 则是为了传递随机的种子。那么使用这个命令再看，是否与之前没有使用-sV_ seed 0 的命令产生了相同的数据呢?最后，请改为使用

"vsim -novopt -solvefaildebug -sv seed random work.tb1"

命令再来比较前后两次的数据，是否相同呢?那么，你对-sV seed random 的仿真选项的认识是什么?

要求 1.4

在仿真的最后，同学们可以发现最后一个打印出来的 chnl_trans 对象的 obj. id 值是 1200，那么这代表什么含义?为什么会有 1200 个 chnl_obj 对象产生呢?整个仿真过程中，在同一时刻，最多的时候一同有几个 chnl _trans 对象在仿真器内存中存在呢?这么做对内存的利用是否合理?你是否还有更好的办法使得在同一时间 chnl _trans 对象的数量比代码中用到的更少呢?

TB2.更加灵活的测试控制

如果要实现不同的 test 类，例如 chnl_basic_test、chnl_burst_test 和 chnl_fifo_full_test，那么我们对于不同的 test 需要对 cinl_generator 的随机变量做出不同的控制，继而进一步控制其内部随机的 chnl_trans 对象。也就是说，随机化也是可以分层次的，例如在 test 层可以随机化 generator 层，而依靠 generator 被随机化的成员变量，再来利用它们进一步随机化 generator 中的 chnl_trans 对象，由此来达到顶层到底层的随机化灵活控制。那么从这个角度出发，我们就需要将 generator 从 agent 单元中搬迁出来，并且搁置在 test 层中来方便 test 层的随机控制，因此我们在 chnl_pkg2.sv 和 tb2.sv 中主要带领大家认识如何更好的组织验证结构，从而实现更加方便的测试控制。

要求 2.1

由于我们将 generator 搬迁到 test 层次中，所以在要求 2.1 中需要将 gen 和 agent 中组件的 mailbox 连接起来，方便 gen 与 agent 中 init 的数据通信。

要求 2.2

在领略了如何在 test 中的 do_ config 对 gen[0]进行随机化控制后，你需要对 gen[1]也按照代码中的具体要求进行随机控制。

要求 2.3

请按照代码中的具体要求对 gen[2]进行随机控制。

要求 2.4.

请按照代码中的具体要求，在 chnl burst test::do_ config()任务中对三个 generator 进行随机控制。

要求 2.5

请按照代码中的具体要求，在 chnl fifo_ full test:do_ config()任务中对三个

generator 进行随机控制。

要求 2.6

在 tb2.sv 中，我们对于测试的选择将由仿真时的参数传递来完成。这意味着，以后的递归测试，即创建脚本命令，由仿真器读入，分别传递不同的随机种子和测试名称即可完成对应的随机测试，而这种方式即是回归测试的雏形。所以请同学们按照之前的仿真命令，在命令窗口中添加额外的命令" +TESTNAME=testname" ，这里的+ TESTNAME=表示的仿真命令项，在由内部解析之后，testname 会被捕捉并且识别，例如你可以传递命令 "+TESTNAME=chnl_burst_test" 来在仿真时运行测试 chnl_burst_test。请同学充分理解要求 2.6 的代码部分，懂得如何捕捉命令，如何解析命令，最后如何选择正确的测试来运行。

TB3 测试平台的结构

最后一个实验部分即指导同学们认识验证环境的其它组件，monitor 和 checker。并且通过合理的方式来构成最终用来测试 MCDT 的验证环境，在这个环境中同学需要再回顾 generator、initiator、monitor 和 checker 各自的作用。在顶层环境中，我们将 checker 置于 test 层中，而不是 agent 中，需要同学们思考这么做的好处在什么地方。同时需要在认识 generator 和 Initiator 有数据通信的同时，可以掌握 monitor 与 checker 之间的数据通信，还有 checker 如何针对 MCDT 利用内部的数据缓存进行数据比较。

要求 3.1

在 chnl_monitor 类和 mcdt_monitor 类各自的 mon_trans()方法中需要采集正确的数据，将它们写入 mailbox 缓存，同时将捕捉的数据也打印出来,便于我们的调试。

要求 3.2

在 chnl_agent 中，参考如何例化的 initiator 对象，也对 chnl_monitor 对象开始例化、传递虚接口和使其运行。

要求 3.3

在 chnl_checker 的任务 do_compare()中， 同学需要从 checker 自己的数据缓存 mailbox 中分别取得一个输出端的采集数据和一个输入端的采集数据，继而将它们的内容进行比较,需要注意的是，输出端的缓存只有一个，而输入端的缓存有三个，同学需要考虑好从哪个输入端获取数据与输出端缓存的数据进行比对。

要求 3.4

在顶层环境 chnl_root_test 中。同学们先要对 mcdt_monitor 和 chnl_checker 进行例化。传递虚接口，并且将 chnl_monitor. mcdt_monitor 的邮箱句柄分别指向 chnl_checker 中的邮箱实例。

实验 3 的整体要求较之前的两个实验都要困难一些，因为它首次引入和随机约束和完整的验证结构。在充分理解实验 3 的要求基础上，我们接下来的实验 4 将会迎来更大的 MCDF 子系统验证环境，到时候我们可以更好地梳理子系统的验证环境，结合课程所讲的验证结构逐步完成 MCDF 的验证计划。