UVM 入门和进阶实验 4

我们在完成了实验 3 之后，大家已经可以看到，monitor、 reference model 与 checker 之间的通信是通过 TLM 端口或者 TLM FIFO 来完成，相较于之前的 mailbox 句柄连接，更加容易定制，也使得组件的独立性提高。接下来，我们要在实验 4 中完成以下内容:

- 将产生 transaction 并且发送至 driver 的 generator 组件，拆分为 sequence 与 sequencer
- 在拆分的基础上，实现底层的 sequence。完成 sequencer 与 driver 的连接和通信工作。
- 构建顶层的 virtual sequencer。
- 将原有的 mcdf_base_test 拆分为 mcdf_base_virtual_sequence 与 mcdf_base_test，前者发挥产生序列的工作，后者只完成挂载序列的工作。
- 将原有的 mcdf_data_consistence_basic_test 和 mcdf_full_random_test 继续拆分为对应的 virtual sequence 和轻量化的顶层 test。

由于上述要求均是整体服务于我们的最终目标，即将 generator、driver 与 test 的关系最终移植为 sequence、sequencer、driver 和 test 的关系，可谓是一个完整的移植过程，因此我们本实验的目标聚焦为 sequence 的使用。

driver 与 sequencer 的改建

(实验 1.1)移除原有在各个 driver 中的 mailbox 句柄，以及在 do_driver()任务中使用 mailbox 句柄通信的方式，转而用 uvm_driver::seq_item_port 进行通信。同时，请定义对应的 uvm_sequencer。

底层 sequence 的提取

(实验 1.2)请将原来在各个 generator 中发送 transaction 的任务，提取为各个对应的底层 sequence 这里需要特别注意，将 mcdf_base_test 中的 idle_reg()、write_reg()和 read_reg()也可以提取并在 reg_pkg 中定义为对应的 sequence，即 idle_reg_sequence、write_reg_sequence 和 read_reg_sequence。

sequencer 的创建和连接

(实验 1.3)在各个 agent 中声明、创建对应的 sequencer，并且将其与 driver 通过 TLM port 连接起来。

移除 generator 的踪迹

(实验 1.4) generator 这个 SV 实验阶段的历史产物，就要从我们的验证结构中清除了。请在 mcdf_base_test 中移除它们的声明、创建以及和 driver 之间的连接。

移除 uvm base_test 的 transaction 发送方法

(实验 1.5)请将已经被从 uvm_base_test 移植到 reg_pkg 中的方法 idle_reg()、write_reg()和 read_reg()从 uvm_base_test 中移除。由此可以看到，uvm_base_test 只变为了容器的性质，在它内部主要由 mcdf_env 将来添加的 mcdf_config 配置对象以及被用来挂载的顶层 sequence 构成。

添加顶层的 virtual sequencer

(实验 1.6)由于 MCDF 验证环境中存在多个底层的 sequencer 和 sequence，因此这

就需要有顶层的 virtual sequencer 与 virtual sequence 统一调度，可以参考红宝书 13.5.2 节。请实现 MCDF 的顶层 virtual sequencer.

(实验 1.7)在定义了 mcdf_virtual_sequencer 之后，请将其在 mcdf_env 中声明、例化，并且完成其与底层 sequencer 的句柄连接。

重构 mcdf_base_test

(实验 1.8)原有的 mcdf_base_test 除了承担其容器的功能，还在其 run_phase 阶段中实现了 sequence 的分阶段发送功能。我们在添加了顶层的 virtual sequencer 之后，**需要将所有发送序列的顺序和组织等内容均移植到 mcdf_base_virtual_sequence，因此需要将 mcdf_base_test::run_phase()发送序列的功能移植到新定义的 mcdf_base_virtual_sequence 一侧**，而在移植后，mcdf_base_test::run_phase()只需要挂载对应的顶层 virtual sequence 即可。

重构 mcdf_dala_consistence_basic_test

(实验 1.9)在理解了 mcdf_base_test 与 mcdf_base_virtual_sequence 的关系之后，我们可以继续重构 mcdf_data_consistence_basic_test，将其产生和发送 transaction 的任务，均移植到 mcdf_data_consistence_basic_virtual_sequence，而进一步减轻 mcdf_data_consistence_basic_test 的体重。由此大家需要加深认识，即测试的动态场景往往是由 virtual sequence 统一组织的，而 test 层往往之后做 run_phase 前的一些验证环境的配置。

重构 mcdf_full_random_test

(实验 1.10)可以继续参照实验 1.9 的要求，也将 mcdf_full_random_test 的内容重构为 mcdf_full_random_virtual_sequence 和 mcdf_full_random_test。

就是在这样一个完整的实验中，同学们可以掌握底层 sequence、sequencer 和 driver 之间的关系，以及复杂环境中的 virtual sequence、virtual sequencer 和 test 之间的关系。在完成了本次实验之后，我们在下一节将迎来 UVM 入门模块的最后一个实验，即将 UVM 寄存器模块移植到 MCDF 验证环境中，学习寄存器模块的常规使用方法，并且完成初步的测试用例移植。让我们一起期待吧！