UVM 1.1b 到 1.2 版本的几点变化

• 新功能和更改功能

变量 uvm_sequence_base::starting_phase 已弃用，并由两个新方法 set_starting_phase 和 get_starting_phase 取代，这些方法阻止在 phase 中间修改 starting_phase。此更改与 UVM 1.1 不向后兼容，但变量 starting_phase 虽然已弃用，但尚未从基类库中删除。

新方法 uvm_sequence_base::set_automatic_phase_objection 将在在 sequence 执行前后自动调用 raise_objection 和 drop_objection，从而避免在一种常见情况下手动调用 raise / drop_objection。

消息系统改造提供基于对象的 API 以向报告添加属性，使用新的宏 `uvm_info_begin /`uvm_info_end 等等。现在可以扩展 uvm_report_server 类，并且可以链接扩展报表服务器。

记录系统改造以为类 uvm_recorder 提供基于对象的 API。

现在可以通过显式恢复默认类型来撤消工厂覆盖。

标准工厂可以替换为用户定义的工厂，例如提供改进的调试功能。

可以使用新方法 uvm_objection::set_propagate_mode 关闭 objections 的分层传播（通常是冗余的），以加快执行速度。objections 的传播最终将被弃用。

新方法 uvm_phase::get_objection_count 提供了一种清除所有异议的方法，即 phase.drop_objection（this，“msg”，phase.get_objection_count（this））；

用于 phase 计划内省的新方法 uvm_phase :: get_adjacent_predecessor / successor_nodes。

新的回调类 uvm_phase_cb 允许在 phase 时进行回调。

现在扩展自 uvm_object 的每个类都必须具有构造函数。

现在 uvm_event 类使用（可选）事件有效内容的类型进行参数化。

类型 uvm_bitstream_t 已被策略对象和字段宏中的类型 uvm_integral_t 替换。

新类 uvm_reg_transaction_order_policy 用于指定超宽寄存器访问时的总线事务的顺序。

前缀 UVM_已添加到类型为 uvm_sequence_state_enum 和 uvm_sequencer_arb_mode 的枚举值中。

new version 宏`UVM_MAJOR_REV，`UVM_MINOR_REV 等。

• 下列特性成为官方标准的一部分：

内置运行时 phase（例如 reset_phase）的进入和退出标准已记录在类参考手册中。

类 uvm_sequence_library 现已记录并正式发布。

变量 uvm_sequence :: req 和 rsp 现已记录并正式发布。

uvm_sequencer＃():: get_next_item，try_next_item，item_done，get，peek 和 put 方法现已记录在案并正式发布。

• 不推荐的功能

set / get_config_int，set / get_config_string，set / get_config_object 不建议使用。

不推荐使用 uvm_component :: status，kill 和 do_kill_all。

不推荐使用 stop_request，global_stop_request，set_global_timeout，set_global_stop_timeout 和 stop_timeout。

uvm_component :: stop_phase 和变量 enable_stop_interrupt 已弃用。

不推荐使用变量 uvm_test_done。

不推荐使用宏`uvm_sequence_utils，`uvm_declare_sequence_lib 和`uvm_update_sequence_lib。

不推荐使用配置数据库参数“default_sequence”，“count”，“max_random_count”和“max_random_depth”。

不推荐使用方法 uvm_sequencer_base :: add_sequence，get_seq_kind 和 get_sequence。

有新的迁移脚本可替换某些已修改或已弃用的功能。

理解 UVM-1.2 到 IEEE1800.2 标准的变化，掌握这 3 点就够了

UVM-1.1d 到 UVM-1.2 的变化还是显著的，那已经是很久之前（2014 年）的事了。不过，对于多数 UVM 用户而言，似乎并没有感受到 1.1d 和 1.2 版本间的代码差别。UVM 从 Accellera 发布的 1.2 版到 IEEE 收编后的 1800.2 2017 版，我们也还是沿着大多数 UVM 用户的使用视野来谈。如果你想阅读这些完整的版本对比，可以查看这篇 DVCon 2018 的论文**《IEEE-Compatible UVM Reference Implementation and Verification Components》**。如果你要下载 UVM 1800.2 2017 1.0 版本的代码（11 月刚刚发布），可以通过 Accellera 官网地址：**http://www.accellera.org/downloads/standards/uvm**

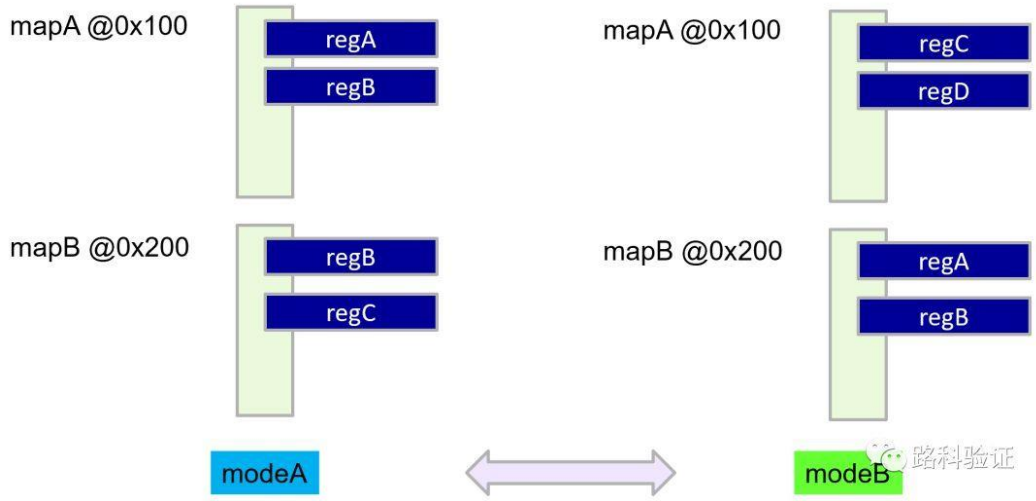
小乌龟地址：(http://10.166.15.61/svn/XIAN_Chip/XIAN_IC_development/03-研发三处/09_常用手册/UVM-18002-2017-11)

在展开阐述之前，请读者首先检查目前使用的 UVM 版本。一般而言，EDA 仿真器已经自己预编译或者预装了 UVM 库，只是你需要检查目前使用的是 UVM-1.1d 还是 UVM-1.2。如果接下来 EDA 仿真器要将 UVM 1800.2 版本代码安装进去，那作为 UVM 用户，你至少需要知道下面**三个变化**。

1. UVM_REG 动态地址索引

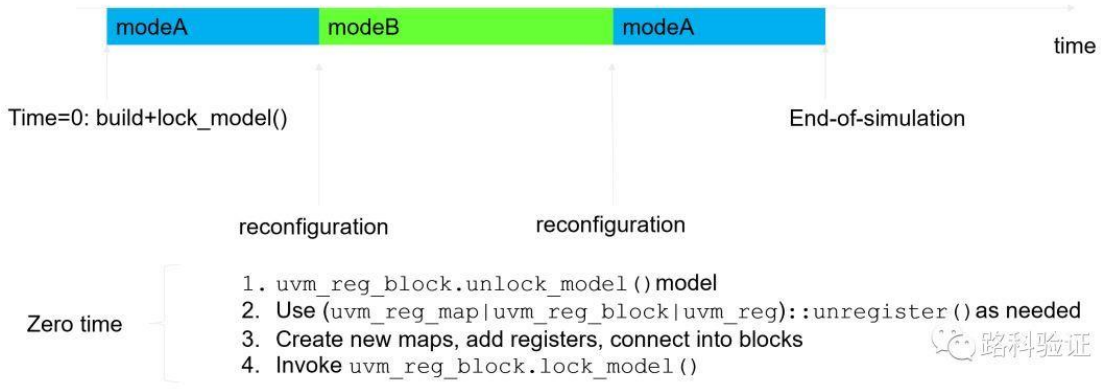
UVM-1.2 时，uvm_reg_block::lock_model()是必备的一步，它用来检查和缓存地址映射，但是一旦 lock 后，无法再对 map 做二次修改。

问题在于目前的 SoC 子系统有动态映射，或者重新映射的功能，因此如果在仿真中 UVM 寄存器模型只有静态地址映射，那么就无法很好地支持这一点。

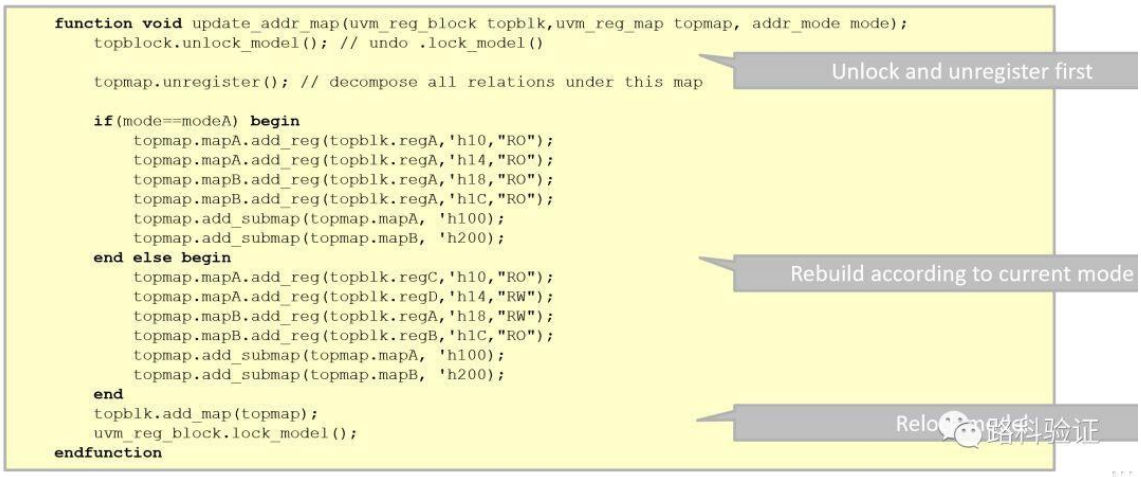


UVM-1800.2 中，我们可以调用 uvm_reg_block::unlock_model 先来“解锁”，再 uvm_reg_block::unregister()来卸载寄存器，完成这些动作以后，只需要重复添加寄存器的动作，再 lock 寄存器块就好了。

例如，想在仿真过程中，先建立 modelA 的地址映射关系，然后在仿真过程中重新建立地址映射。在 UVM-1800.2 中，我们可以在仿真过程中（不一定是 build 阶段），unlock_model()和 unregister()，接下来再为 uvm_reg_block 建立新的 uvm_reg_map 并将其添加和 lock_model()。



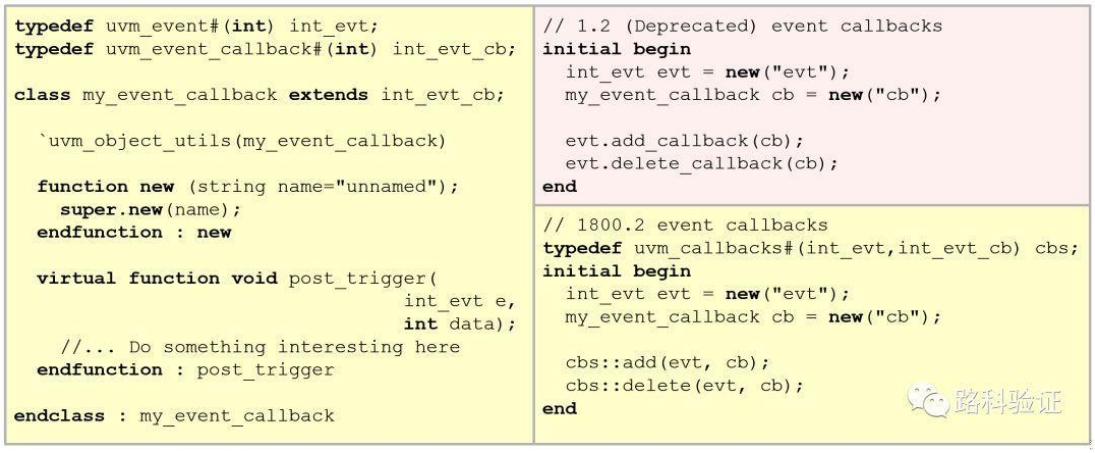
这种灵活的重新配置，可以从上面的步骤实现为下面的方法update_addr_map()，用户可以在仿真中任何一个阶段来更新地址映射。



2. uvm_event_callback 并入 uvm_callback

UVM-1.2 时，uvm_event_callback 虽然也是 callback 类，但和 uvm_callback 是分割的，所以在将它添加到 uvm_event 时，其添加方法与添加其它 uvm_callback 对象的方式是不同的。

UVM-1800.2 中，这一切得到了统一，uvm_event_callback 也变为了 uvm_callback 类，也由此，添加 uvm_event_callback 对象的方法就变为了添加 uvm_callback 对象的方法。



3. sequence 的宏得到简化

UVM-1.2 时，`uvm_do()和其众多变种让 UVM 新人无所适从，对于测试用例的阅读也不那么友好。

UVM-1800.2 中，只需要`uvm_do()就足够了，因为用户可以为其添加其它参数（如果不使用缺省参数的话）。