

引用格式:李晨阳,宋澍申,王涛,等. 一种基于 UVM 的高层次化验证平台设计 [J]. 微电子学与计算机,2019,36(6):79-83.
[LI C Y, SONG S S, WANG T, et al. An high hierarchical verification platform based on UVM [J]. Microelectronics & Computer,2019,36(6):79-83.]

一种基于 UVM 的高层次化验证平台设计

李晨阳¹,宋澍申^{2,3},王 涛¹,黄坤超¹

(1 中国西南电子技术研究所,四川 成都 610036;2 大连理工大学 计算机科学与技术学院,辽宁 大连 116024;
3 密苏里大学 工学院,密苏里 哥伦比亚 65211)

摘 要:在通用 UVM 验证平台的基础上,提出了双层次序列设计、数据库共享、协议模块、事件触发同步化等技术优化验证平台,实现了一种高层次化的 UVM 验证平台,克服了通用 UVM 验证难以适应多重验证环境的问题,具有更高的验证抽象级别,高效、透明和简洁的处理方式,良好的可重用性.通过一个平台,一次脚本化运行,实现功能覆盖率 100%,测试点覆盖率 100%.此方法大大简化了 UVM 测试的复杂性,加快了测试速度,减少了人工测试带来的潜在错误.

关键词:UVM;层次序列;数据库;协议模块;事件触发

中图分类号:TP3

文献标识码:A

文章编号:1000-7180(2019)06-0079-05

An high hierarchical verification platform Based on UVM

LI Chen-yang¹,SONG Shu-shen^{2,3},WANG Tao¹,HUANG Chu-chao¹

(1 Southwest China Institute of Electronic Technology, Chengdu 610036, China;

2 School of Computer Science and Technology, Dalian University of Technology, Dalian 116024, China;

3 College of Engineering, University of Missouri, Columbia 65211, USA)

Abstract: On the basis of general UVM verification platform, put forward a double hierarchical sequence design, database share, protocol module, events triggers to synchronize etc the technique to optimize platform, realize an high hierarchical UVM verification platform, overcame the hard in general UVM in multiple verification environment, having the higher the abstract level in verification, efficiently, transparent handles with pithiness, good reuse. Pass a platform, a script run, realize the function coverage rate 100%, testing dot coverage rate 100%. this method simplified the UVM test complexity, quickly test speed, reduced the latent mistake that artificial test bring.

Key words: UVM; hierarchical sequence; database; protocol module; event trigger

1 引言

随着可编程逻辑器件的广泛应用以及用户对产品质量要求的提高,传统的仿真验证已难以满足日益增长的可编程逻辑器件验证需求,由于其验证抽象层次低、层次结构划分不清晰,使得可重用性不好^[1].近年来,UVM、OVM、VMM 等一类新的验证方法学的有力发展,提升了验证效率.研究表明,越是高层次化的验证则验证效率越高,越是低层次化

的验证则验证准确性越高^[2].新的验证方法学是属于 TLM 级验证,相比于传统的 RTL 级验证,是一种更高层次的验证方法. UVM 主要是对结构相似的 DUT 进行验证^[3],在实际工程中 DUT 对象往往呈现的是多重接口特性,针对多重验证环境,通常在 UVM 中需要重新搭建验证环境,建立连接关系,设计层次化序列,使得验证的复杂度和出错概率都急剧增加^[4].

在分析通用 UVM 验证平台的基础上,重构平

台架构,提出了双层次序列设计、数据库共享、协议模块等技术,实现了一种高层次化、测试自动化、可重用性好、透明和简洁的 UVM 验证平台,能同时适用于多重验证环境,最后举例证实方法的有效性.

2 优化 UVM 验证平台的架构

在通用 UVM 验证平台中,一个 Drive 只能接受一种 sequence 序列格式,各个组件之间信息传递采用规定的 port 端口通信技术实现,一个 agent 中只能构造一个 sequence 的通信端口,端口之间连接关系具有严格要求,这种方式降低了可重用性,增加了复杂性.当存在多重验证环境时,通用验证平台的同步化处理会使测试环境复杂度急剧增加,变成一项非常难以忍受的设计^[5].

为满足多重验证环境需要,对通用 UVM 验证平台进行重新优化,优化后的 UVM 验证平台架构如下图 1 所示.

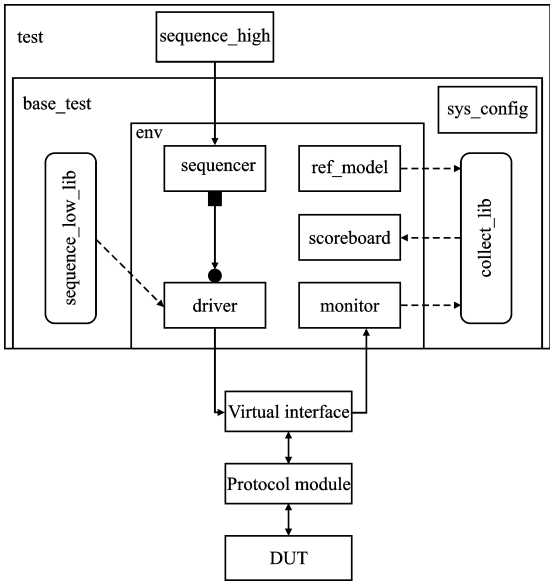


图 1 优化后的 UVM 验证平台

优化后的 UVM 验证平台与通用验证平台层次关系一致,BASE_TEST 层在通用平台中作为辅助层,但在优化后成为一种关键层,包含了对数据库控制和管理. ENV 层包含 sequencer、Drive、Monitor、Ref_model、Scoreboard 共 5 个组成部分,不再做进一步层次划分,取消 agent 封装,每个组件之间信息交换采用数据库共享方式传递,连接关系简单,同时适用多个 sequence 处理,消除同步化带来的问题.

TOP 层作为测试平台的最顶层,优化后的 TOP 层在设计时,由于在接口设计时利用了协议模型搭建,因此不再需要指定具体信号,采用隐含模式

即可,顶层设计简单化,受到 DUT 影响减小. TEST 层主要控制发送 sequence_high(高层次序列)给 sequencer,使用 starting_phase 来控制验证平台的开启和关闭.

BASE_TEST 层是优化后的验证环境中一个关键层,主要作用是建立数据库和 ENV 层的各个组件之间连接关系,建立数据传输通道,其基本树形结构如下图 2 所示.

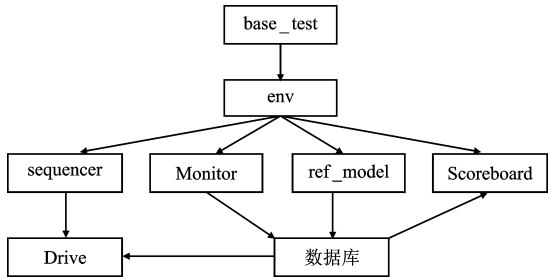


图 2 BASE_TEST 层树形架构

BASE_TEST 属于 env 的上一个层次,需要先对 env 例化,然后再例化其他模块,使用 uvm_config_db::set 函数将需要共享的数据库传递给指定的模块,最后在使用的文件中,使用 uvm_config_db::get 函数将指定的模块导出数据库.

数据库主要包含双层次序列(包含高层次序列和底层次序列库)、数据收集库、系统配置 3 个部分,如下表 1 所示.

表 1 数据库

名称	数据库	备注
高层序列	Sequence_high	一个高层次的测试序列
底层序列库	Sequence_low_lib	大量的底层次的测试序列
数据收集库	Collect_lib	监控到的数据、覆盖率收集的信息

ENV 层中取消了 agent 封装,取消了采用端口传递数据,简化设计,在 ENV 层中直接统一封装 Drive、Monitor、ref_model、Scoreboard 等组件,每个组件例化后,只保留 sequencer 到 Drive 之间端口传递数据,如上图 1 所示.

Drive 组件主要负责每个接口信号驱动,包括设置必要的信息发送和接收;Driver 根据接收到高层次序列命令,从数据库中提取相应的底层次序列和系统配置参数,进行序列初始化、序列解析、序列保存,随后通过虚拟接口将底层次序列送给 DUT.同时,Driver 必须调用系统配置参数来控制 DUT 的时序,因为 DUT 是一种时序电路,驱动时会有时序要求.因此,必须要有系统配置中的时序参数来控

制接口信号正确的时序关系,如时钟频率、建立时间、保持时间、发包间隔等.最后,利用事件触发产生驱动完成标志.

Monitor 组件取消模式区分和端口传递.一方面收集 DUT 的各种输出数据,另一方面收集覆盖率数据,并将收集好的两类数据传递给 Collect_lib(数据收集库),完成不同数据类型的收集和传递.由于监控到的数据时都是在不同时刻下完成,采集时与 Driver 一致,利用事件触发确保数据传递的准确性,在收集数据完成后,产生不同的同步标志信息.

Scoreboard 组件取消端口传递,通过读取标志信号判决,从数据收集库 Collect_lib 提取要比较的数据,比较完成后输出结果.与传统 UVM 的 Scoreboard 相比,连接关系只与数据库关联,隔离 DUT 程度高,一个 Scoreboard 可以同时满足多组序列自动对比.

Interface 层在通用 UVM 基础上,在 virtual Interface 与 DUT 之间增加协议模块 protocol module,进一步隔离 DUT 与 TB,使 TB 可以单独复用,把设计平台和测试平台彻底分隔开来,保证其不受设计改动的影响,实现事务交易处理器的更大可重用性.

3 关键性技术

3.1 双层次序列

双层次序列包含高层序列和底层序列库,将多序列分解为高层次序列和低层次序列两类,高层次序列抽象级别较高,低层次序列是直接测试序列,抽象级别较低,两者之间顺序执行,则改进后 UVM 序列器如下图 3 所示.

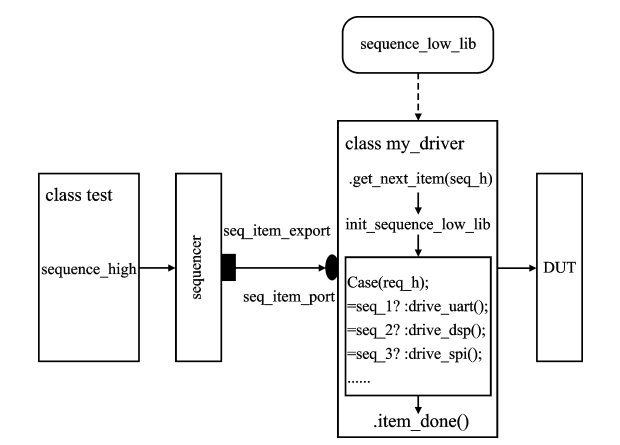


图 3 改进后 UVM 序列器设计

当测试开始时,Driver 通过 get_next_item()向 Sequencer 请求数据;Sequencer 从 Sequence 中得到

高层次序列 sequence_high,并将 sequence_high 以指针的形式传递给 Driver;Driver 得到后,初始化低层次序列库 sequence_low_lib,解析并判决 sequence_high,根据判决结果将低层次序列 sequence_low 发送给 DUT;待 Driver 处理完序列后,通过 item_done()通知 Sequencer 数据已发送.紧接着 Driver 依据 sequence_high 继续发送下一个 sequence_low,依次不断循环,直至 sequence_high 要求的所有 sequence_low 都发送完毕,这时 TEST 才确认并结束测试程序.

(1)高层次序列设计

首先将得到的测试计划按照需求规格提炼出验证特性,将验证特性进行归类处理后,对应的测试用例编写成测试功能点,采用用例标识表示,如下表 2 所示.

表 2 测试用例设计

软件测试计划		
用例特性	用例名称	用例标识
功能特性	时钟处理功能	SRS_GN_01
	复位处理功能	SRS_GN_02
	UV 控制功能	SRS_GN_03
	HF 控制功能	SRS_GN_04
接口特性	时钟接口	SRS_JK_05
	RS422 接口	SRS_JK_06
	DSP6455 接口	SRS_JK_07
性能特性	RS422 处理速率	SRS_XN_08
	数据回传时间	SRS_XN_09
.....		

通过以上流程,就可以得到验证的功能点,采用列表方式或其他方式形成一组高层次序列.由于高层次序列是依据测试计划而来,因此,高层次序列与测试计划之间就具有追踪关系,能够保证测试的完整性和充分性.

在程序中,高层次序列采用枚举方式实现,枚举集合中的不同名称代表不同的测试用例标识.采用 class 形式将高层次序列封装起来,形成一组序列,向下层提供一个统一的任务接口.最终,一个 sequence_high 完成所有的测试用例设计,通过 Test 层完成所有的测试用例执行,如下图 4 所示.

(2)低层次序列库设计

一个软件测试计划需要从不同方面、不同输入数据类型来进行验证,每一个输入数据的组合就可以看作一个测试用例,将多个相似的测试用例合并,产生一个新的 sequence 即低层次序列.由于 UVM 是一个连续运行的过程,序列会一直在不同的层次中存在,sequence 本身又是一种 object.基于这一特

```
typedef enum(
    SRS_GN_01,SRS_GN_02,SRS_GN_03,SRS_GN_04,
    SRS_JK_05,SRS_JK_06,SRS_JK_07,
    SRS_XN_08,SRS_XN_09,SRS_AQ_10
) cmd_op_e;
class my_transaction extends uvm_sequence_item;
    rand cmd_op_e cmd_op;
    .....
```

图 4 高层次序列设计示例

性,创建一个由多个低层次序列组成的低层次序列库 sequence_low_lib,序列库可以在不同层次之间实现数据共享。

sequence_low_lib 包含了大量的低层次序列,使用 uvm_config_db 函数获取,根据高层次序列特性创建不同数据格式的低层次序列库.当 Driver 从低层次序列库提取序列时,这时提取到的序列是固定的序列,在不加处理直接使用时,那么每次仿真会得到相同的随机结果,因此,在提取后需要对内容进行随机化处理,包含所有激励以及配置的 class,通过运行`uvm\_do\_with`函数完成.实现序列的随机化。

双层次序列有效解决了测试用例与测试数据之间的匹配,高层次序列完成了测试用例产生,低层次序列完成了测试数据产生。

3.2 接口设计

在通用 UVM 接口,TB 与 DUT 之间通过增加虚拟接口 Virtual interfacer 实现隔离。

优化后的 UVM 接口中,采用协议模块 Protocol Module 方法,即在虚拟接口 Virtual interfacer 与 DUT 之间增加 Protocol Module 方法,进一步隔离 DUT 和 TB 之间带来的影响^[6].协议模块是一种封装 DUT,可以完成对多个 DUT 独立封装,保护 DUT、TB 以及 Top 层.当 DUT 发送变化时,仅对协议模块做出改变,而不会影响到 TB 中去;当 TB 发生变化时,不会影响到 Top,如下图 5 所示。

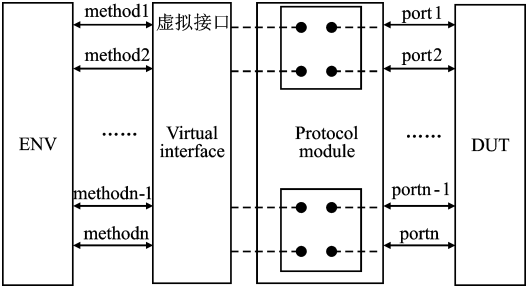


图 5 改进后的 UVM 接口层

使用协议模块来隔离不同类型接口,实行接口

分类包装,接口信号命名的变化、信号删除、信号增加等不会受到影响,提高可重用性.规范 Virtual interface 接口设计,设计接口时钟域分类管理,兼容多时钟域接口的设计需求,形成层次清晰,可以满足不同接口需要。

3.3 脚本化流程

一个工程中的源文件不计其数,为了便于管理,按其类型、功能、模块应分别放在若干个目录中,采用脚本设计测试过程和复杂操作,实现工程完全自动化操作,提高软件测试效率^[6].脚本根据被测件以及测试文件不同,将 Lib、DUT、TEST、Log、Run 过程分类设计,顺序执行测试工作流程,脚本设计架构如下图 6 所示。

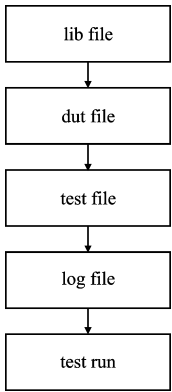


图 6 脚本设计架构

利用脚本执行验证平台,自动呈现测试结果,避免大量人工重复劳动,提高软件测试效率,脚本化的验证平台工作流程如下图 7 所示。

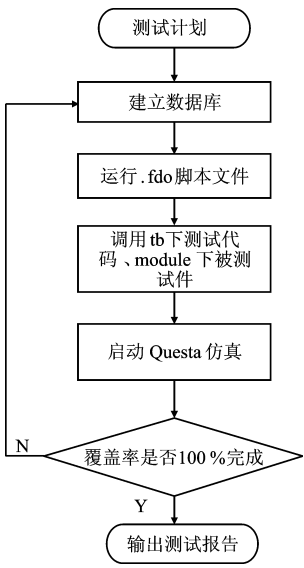


图 7 测试平台工作流程

依据测试计划,建立好数据库,然后运行自动化脚本文件,获取被测试模块信息和测试用例信息.获取信息后,调用当前的测试代码、被测件等内容,开始编译源代码、链接代码,启动仿真工具.在仿真过程中会输出覆盖率报告文件,检查是否 100%覆盖,若无则修改数据库,重新执行直至 100%满足.仿真结束后,测试平台显示所有测试结果和覆盖率信息,完成整个测试过程.

4 验证结果分析

改进后的平台运用于某工程的 FPGA 软件测试中,该工程分别包含 DSP、UART、SPI 等不同接口类型,共梳理出测试用例 20 个(功能测试点),每个测试用例需要覆盖的测试点共计 120 个,平台一次运行,完成了测试计划的所有需求,功能覆盖率 100%,测试点覆盖率 100%.

5 结束语

通过采用多种技术优化验证架构,实现了一种高层次化的 UVM 验证平台,不仅解决了数据传递和同步化问题,使得序列设计变得简单、快速,而且可以满足多重验证环境需要.随着自动化测试工具在软件测试过程替代的不断前移,数据库设计也需要向规范化和标准化发展,需要不断的补充和完善.

参考文献:

[1] 杨坤,徐金甫,李伟. 基于 SystemVerilog-UVM 的 Mickey 2.0 RTL 级验证[J]. 计算机工程与设计, 2016, 37(10):2644-2647.

[2] Gong L K,OLIVER D. Functional Verication of Dynamically Recon gurable FPGA-based Systems [D]. Springer, 2015: 63-64.

[3] 王国军,景为平. 基于 UVM 的验证平台设计研究[J]. 微电子学与计算机, 2016, 33(10):164-168.

[4] SHARON R, KATHLEEN A M. A Practical Guide to Adopting the Universal Verification Methodology (UVM) [M]. USA: Cadence Design Systems, Inc, 2010.

[5] 熊涛,蒋见花. 基于 UVM 验证方法学的纵向可重用研究[J]. 微电子学与计算机, 2016, 33(4):64-68.

[6] 谈笑,王小力. 一种基于 UVM 的模块级可重用随机化验证平台构建方法[J]. 微电子学与计算机, 2015, 32(3):67-72.

作者简介:

李晨阳 男,(1974-),硕士,工程师.研究方向为 FPGA 软件测试以及数字信号处理. E-mai:kkwei_222@163. com.

宋澍申 男,(1996-),研究方向为计算机软件及算法.

王 涛 男,(1972-),高级工程师.研究方向为航空电子信息.

黄坤超 男,(1971-),硕士,研究员.研究方向为计量测试技术.