文章编号:1671-4598(2022)07-0304-06 DOI:10.16526/j.cnki.11-4762/tp.2022.07.046 中图分类号:TP3 文献标识码:A

基于 UVM 的远程控制 FPGA 一体化闭环仿真验证平台

于清华，高赛军
（上海航天电子通讯设备研究所，上海 201109）

摘要：远程控制 FPGA 是卫星执行地面指令和转发地面数据的核心部分，且该 FPGA 使用的帧协议较为复杂，帧传输数据协议数据格式变化多样；为了对远程控制 FPGA 进行充分的验证，以提高 FPGA 软件的可靠性，避免存在设计隐患，提出了一种使用目前较先进的通用验证方法学 UVM 建立了一体化闭环仿真验证平台用来测试远程控制 FPGA；仿真结果表面，该验证平台具有带约束收敛的测试向量随机生成和自动检查输出结果正确性功能，能够进行功能覆盖率检测，并有效提高远程控制 FPGA 验证的效率和质量，较好地满足了验证需求。

关键词：UVM；远程控制；FPGA；验证平台；DUT

Design of Remote Control FPGA Closed-loop Simulation Verification Platform Based on UVM

YU Qinghua，GAO Shaijun
（Shanghai Aerospace Electronic Communication Equipment Research Institute，Shanghai 201109，China）

Abstract：Remote control Field Programmable Gata Array（FPGA）is the core part of satellite to execute ground instruction and transmit ground data，and the frame protocol used by FPGA is complex，and the format of frame transmission data protocol varies widely. In order to fully verify the remote control FPGA，improves the reliability of FPGA software and avoids the hidden design problems，an integrated closed-loop simulation verification platform is established to test the remote control FPGA using the advanced Universal Verification Methodology（UVM）. The simulation results show that the verification platform has the functions of test vector random generation with the constraint convergence，which automatically check of the correctness of the output results. It can detect the function coverage and effectively improve the efficiency and quality of remote control FPGA verification，which better meets the verification requirements.

Keywords：universal verification methodology（UVM）；remote control；field programmable gata array（FPGA）；verification platform；device under test（DUT）

0 引言

随着信息量的飞速爆炸，FPGA（Field Programmable Gate Array，现场可编程门阵列）作为提升电子产品速度性能的重要手段，其应用层面越来越广泛面，复杂程度日益提高，FPGA 软件的质量和鲁棒性也越来越受到大家的重视，FPGA 验证尤其是功能仿真变得尤为重要。特别是航空航天产品，基于 FPGA 的应用系统在运行过程中因异常情况发生导致 FPGA 功能错误，使得下行到地面的数据错误或者丢失，甚至导致单次试验任务失败[1]，需要对系统重新上电复位。如果在地面测试或者仿真验证过程中[2]，针对各种正常和异常的测试场景进行全面测试，则会提前发现软件缺陷，从而降低故障的发生概率。

随着 FPGA 设计功能繁杂度增加，其门数和复杂度也日益增加，验证难度越来越大，验证充分性和验证效率的提升是验证的致命瓶颈，也是验证设计的主攻方向[3]。传统定向仿真验证方法由于存在抽象层次低、效率低、验证不充分、验证平台重用性差的缺点，难以满足验证的需求。面向对象的验证方法学的产生，能够很好解决传统验证方法的上述痛点，可解决大规模 FPGA 的验证平台重用性问题，有效的提升验证效率和验证完备性。

基于 SV（System Verilog）的验证方法学从 VMM 发展到 OVM，最后进阶到成为整个电子行业统一验证标准通用验证方法学[4] UVM（Universal Verification Methodology）[5]。该方法学由于采用了较佳的验证框架实现覆盖率驱动的验证，并配有受约束的随机验证方法，可有效实现可重用仿真验证环境，大大缩短了验证时间，提高验证效率。

收稿日期:2022-01-14； **修回日期**:2022-02-18。

作者简介:于清华(1982-)，女，山东烟台人，硕士研究生，主要从事软件测评技术研究、FPGA 验证技术研究、质量体系建设等方向的研究。

引用格式:于清华，高赛军. 基于 UVM 的远程控制 FPGA 一体化闭环仿真验证平台[J]. 计算机测量与控制，2022，30(7)：304-309.

本文以航天某卫星型号远程控制 FPGA 为验证对象，采用 UVM 验证方法学搭建一体化闭环仿真验证平台，采用动态数组方法实现了三级激励数据包的嵌套，验证远程控制 FPGA 功能的正确性。

1　UVM 简介

1.1　UVM 验证方法学概述

UVM 有一套 System Verilog 的语法和语义定义的具备面向对象编程的类库[6]，使用者通过使用这些类库创建包括驱动器、监视器和记分板等可重用组件，从而提高验证效率和质量[7]。此外，这些类库还包括各种任务和函数，能够完成驱动器的驱动与被测件（DUT-Device Under Test）通信功能，以及实现监视器的监测 DUT 输入输出接口功能等。最后，UVM 还可通过 phase、factory 等高级机制和寄存器模型等功能，实现了验证平台的高安全性和可重用性[7]。

1.2　UVM 验证平台的组成

图 1 是一个典型的 UVM 验证平台结构示意图，由 1 个验证环境 env 组成，1 个 env 包括 2 个代理器 Agent：In _ Agent 和 Out _ Agent，参考模型（reference model）和记分板（Scoreboard）[8]。In _ Agent 负责驱动和监测总线，Out _ Agent 负责监测 DUT 的响应。在 In _ Agent 中，Sequencer 负责产生[9]随机测试队列传送给驱动器 Driver。

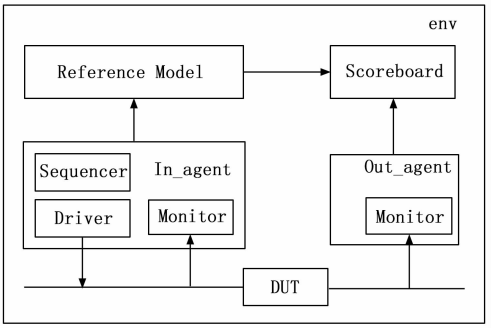

图 1　UVM 验证平台结构示意图

图 1 中，in _ Agent 包含 3 个组件 Sequencer、driver 和[10]Monitor，其主要功能是在序列发生器 Sequencer 调度下产生根据传输协议生成的队列[11]，再由根据约定时序由驱动其 Driver 驱动到 DUT 输入端口上。同时，监视器 Monitor 将产生的队列发送至参考模型 reference model[12]。参考模型通常根据需求模拟被测件的行为，产生期望结果并压到期望堆栈中。out _ Agent 的 Monitor[13]用于监测 DUT 输出，并将实测结果发送至记分板 Scoreboard。Scoreboard 在收到实测结果后，自动提取来自参考模型的数据堆栈中的期望结果和实测数据进行比较，自动化比对结果。

1）序列产生器（Sequencer）：

主要自动产生受约束的[14]随机化激励数据。

2）驱动器（driver）：

主要负责向 Sequencer 请求队列，将队列根据特定的传输协议和接口时序转化为输入信号发送到 DUT 的输入端口。

3）监控器（Monitor）：

负责从驱动器 driver 输出采集数据，传送到参考模型并转换成期望队列发送给记分板（Scoreboard），从而完成功能数据结果的比对和覆盖率信息的收集。

4）代理器（Agent）：

通常用于发送端，可封装序列产生器（Sequencer）、驱动器（Driver）、监视器（Monitor）。

5）参考模型（reference model）：

reference model 用于完成和 DUT 相同的功能。reference model 的输出发送给 Scoreboard，用于和 DUT 的输出相比较。

6）记分板（Scoreboard）：

比较 reference model 和 Monitor 采集到的 DUT 输出信号，并给出比较结果。

7）验证环境（env）：

UVM 验证环境，使用 UVM 机制将代理器（Agent）、参考模型（reference model）和记分板（Scoreboard）配置在一起。

8）测试用例（test case）：

test case 用于例化 env，不同的 test case 用于对 DUT 的不同功能进行验证。

1.3　UVM 的各种机制

UVM 是基于 System Verilog 语言开发的一套开源类库，包括了一系列标准类，如 uvm _ sequence、uvm _ driver、uvm _ monitor、uvm _ sequencer、uvm _ agent、uvm _ env 等，通过对这些基本类进行继承和重载，结合 TLM 标准接口和各种机制，可构造多层次可重用的高效验证平台。

1）factory 机制：

factory 机制是 UVM 非常重要的一个机制，采用 UVM 中的宏注册实现 factory 机制。当某个类经过注册并且实例化后，其 main _ phase 会自动调用。使用 factory 机制可以在不改变原代码的基础上，调用 UVM 内部封装的大量功能，实现验证平台组件的重用，提高整个验证平台的重用性。

2）phase 机制：

UVM 通过 phase 机制实现了验证流程的细分，在不同的 phase 中实现相应的任务，或者在不同的 phase 之间进行跳转，可以更加容易地控制验证的进程，使得验证平台中各组件按照各自的需求自动阶段性执行。

按照是否需要消耗仿真时间，phase 可以分成两大类：function phase 和 task phase，function phase 包括 build _ phase、connect _ phase 等，不会消耗仿真时间；task phase 包括 run _ phase 和 main _ phase 等，通过任务实现，需要消耗仿真时间。整个验证平台按照 phase 的执行顺序，执行完一个状态，自动跳转到下一个状态。build _ phase 完成 UVM 中各个类成员的实例化工作，connect _ phase 将 UVM 中各个功

能组件实例化对象进行连接，run _ phase 则运行整个验证平台，按照事务生成器顺序驱动整个平台运行。

3）sequence 机制：

sequence 机制用来产生激励，包括 sequence 和 sequencer。sequence 不属于验证平台组件，但对于验证平台来说处于特殊位置，sequencer 只有在 sequence 出现的情况下才有价值，sequence 通过 sequencer 才能将它产生的 sequence 传送到驱动器。

sequence 指定了 DUT 需要的 transaction 类型数据，每一个 sequence 类都有一个 body 函数，启动一个 sequence 时，body 函数会自动执行。实际使用时经常使用以下 4 个宏`uvm _ do、`uvm _ do _ with、`uvm _ do _ pri、`uvm _ do _ pri _ with 来创建 transaction 的实例，同时对 transaction 进行约束和随机，并把这些数据通过端口发送到 sequencer。当 sequencer 同时检测到 driver 和 sequence 的请求时，将 sequence 产生的 transaction 发送给 driver。实际程序中通过 default _ sequence 方式启动 sequence。

通常采用 virtual sequence 实现 sequence 之间的同步。使用 virtual sequencer[15]来使用 virtual sequence，virtual sequencer 里面包含了指向其他 sequencer 的指针。

4）objection 机制：

objection 机制通过 raise _ objection 和 drop _ objection 对验证平台组件内部的 phase 进程进行控制，用于开启和关闭验证平台。

5）TLM 机制：

TLM（transaction level model）即事务级建模，是 System C 的一种通信标准。transaction 是把具有某一特定功能的一组信息封装在一起形成的一个类。UVM 中内置了各种 port，用于实现各组件之间 transaction 级别的通信。

6）config _ db 机制：

通过 config _ db 机制，可以修改验证平台的参数，对接口和寄存器进行配置。UVM 通过 set 和 get 函数实现这一过程。config _ db 机制可以在不同的层次对同一个参数进行配置。

7）field _ automation 机制：

field _ automation 机制是使用 UVM 中定义好的宏对 transaction 数据成员进行注册，transaction 可以直接调用 UVM 中特有的内部函数，如比较、复制、打印等。

1.4 验证平台的工作流程

在设计好验证平台之后，当定义好一个 test case 后，通过在命令行中添加＋UVM _ TESTNAME 指令要启动执行的 test _ case。仿真器首先进入顶层模块 top，当执行到 run _ test 后，启动 UVM 验证平台。验证平台首先根据＋UVM _ TESTNAME 后的字符创建一个类的实例，然后自动执行该实例的 build _ phase，创建 env，当 build _ phase 执行完毕后，接下来自动执行 env 中的 build _ phase，创建 env 中的验证组件。如此循环，自上而下执行所有 component 中的 build _ phase，建立 UVM 树。执行完 build _ phase 后，执行 connect _ phase，之后再执行 main _ phase 等 phase。当所有的 phase 执行完毕后，结束仿真。

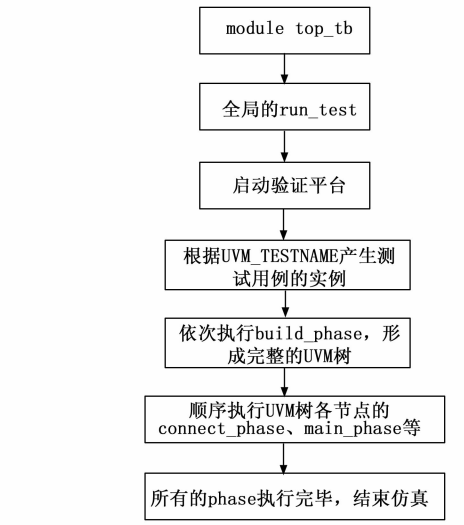

图 2 UVM 验证平台工作流程图

2 远程控制 FPGA 概述

远程控制产品是卫星的重要组成部分，主要完成卫星对地面上传数据的处理、直接指令执行和数据转发。远程控制 FPGA 是远程控制产品的核心，其主要功能框图如图 3 所示。首先对输入的远程控制上行数据进行 BCH 解码、解扰、CRC 校验，CRC 校验正确后将远程控制上行数据中包含的传送帧写入 SRAM 缓存，然后再从 SRAM 读出传送帧进行传送帧解析，当传送帧中数据域为重要指令时，将传送帧包含的内部指令数据指令码和 PROM 预先存储的指令码表进行比对，比对正确后译码输出相应的 48 条指令控制信号，传送给外部的 BM2711 器件进行指令驱动输出；当传送帧中数据域为远程数据包时，根据 VCID 值的不同，选择将传送帧中包含的远程数据包通过异步串行接口或者两线制通信接口进行转发。

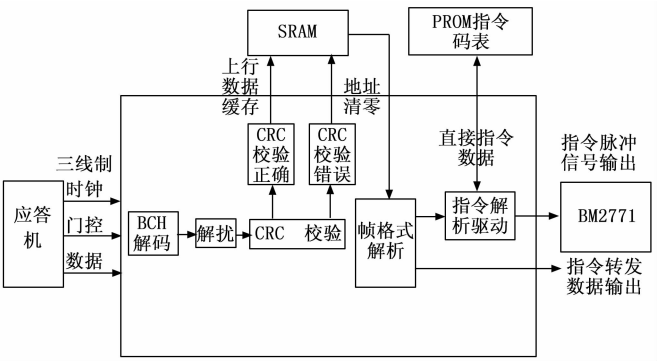

图 3 远程控制 FPGA 功能框图

输入远程控制上行数据帧格式示意图如图 4 所示，按照执行终端和处理途径分，可以分为遥控包和内部指令。

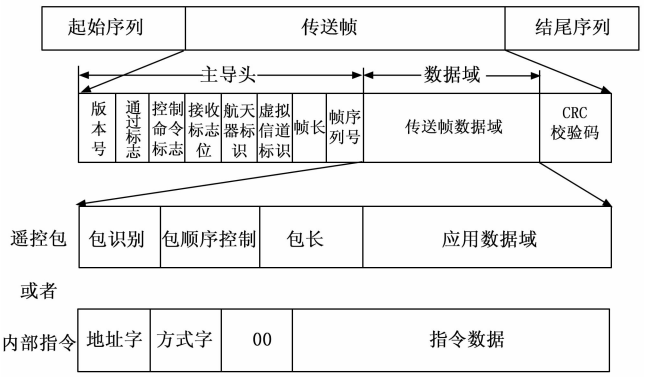

图 4　远程控制 FPGA 输入远程控制上行数据帧格式图

远程控制 FPGA 是航天产品中的一种重要工作软件，安全关键等级高，需要测试大量的测试用例，采用传统的定向验证方法耗时耗力，无法进行随机指令组合，无法确保不同数据帧长均测试全面，故很[16]难保证测试的充分性。采用 UVM 高级验证方法学搭建验证平台，能随机产生符合帧协议的测试队列，并自动比对测试结果，提高测试效率和[17]测试质量，有助于保证航天型号试验任务的顺利完成。

3　仿真验证平台的设计与实现

3.1　验证策略

为了保证远程控制 FPGA 验证的完备性和高效性，在搭建验证平台前，需要制定验证策略和计划，确定平台应该具有的基本要求和功能。

1）提取验证功能点：

根据远程控制 FPGA 需求规格说明，需要测试远程控制 48 条指令执行功能和 4 种远程控制数据（直接指令、间接指令、直接注数和软件数据）转发功能。

2）设计验证平台架构：

根据远程控制 FPGA 需求规格说明，DUT 需要接受 2 种格式的输入远程控制上行数据，对应 2 种数据协议，将验证环境 ykzd _ env 划分为 2 个 Agent：远程控制指令执行代理器 Agent 和远程控制数据转发代理器 Agent。

本文设计的远程控制 FPGA 验证平台如图 5 所示。ykzd _ env 与 DUT 之间采用虚拟接口 ykzd _ if 进行数据通信。采用自底向上创建代理中的各组件。

ykzd _ env 主要由远程控制数据包代理器主 ykzd _ master _ agent 和远程控制指令代理器 ykzd _ zl _ agent 组成。如图 5 所示，利用 UVM 机制，2 个代理器分别封装了相应的序列生成器 sequencer、驱动器 driver 和监控器 monitor。验证平台各组件之间的通信以字节为单位，ykzd _ if 和 DUT 之间的引脚通信以 bit 为单位。

远程控制数据包代理器的 driver 用于随机不定时驱动远程控制数据帧发送给 DUT，同时利用 ykzd _ master _ monitor 监控驱动队列数据根据队列数据的 VCID 将传输数据帧数据域提取出来，分别送各自的参考模型 rs422 _ ref 和 lvds _ ref 生成期望动态数组堆栈传输给相应记分板 rs422 _ scoreboard 和 lvds _ scoreboard；远程控制指令代理器 ykzd _ zl _ agent 用于随机不定时驱动远程控制指令帧发送给 DUT，同时利用 ykzd _ zl _ monitor 监控驱动队列数据将传输指令帧数据域提取出来，传输给参考模型 cmd _ ref 根据指令匹配生成期望动态数组堆栈传输给相应记分板 cmd _ scoreboard。3 个接收 monitor 分别监控 DUT 连接的接口模型产生的实测数据并分别送记分板进行比对，实现整个仿真验证平台的一体化闭环验证。

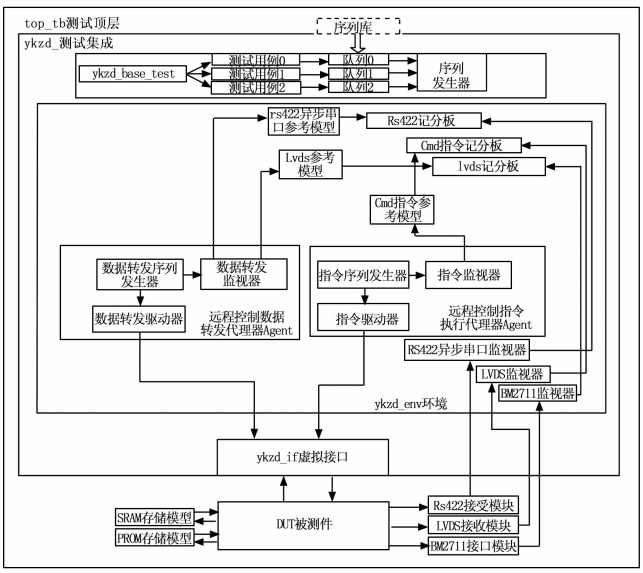

图 5　远程控制 FPGA 的 UVM 验证平台

3）测试激励产生方案：

受约束的随机激励可以根据软件功能需求设置输入数据的约束范围，实现正常情况测试和故障情况测试，本验证平台设置远程控制数据包占空比为直接指令：间接指令：直接注数：软件数据＝10：5：2：1，验证平台的 ykzd _ master _ sequencer 序列发生器会自动调用数据包 transasction 按照队列方式产生远程控制数据包。远程控制指令包数据值约束按照 PROM 与设置的 48 路指令数据和异常数据指令进行随机分配，验证平台的 ykzd _ zl _ sequencer 序列发生器会自动调用指令包 transasction 按照队列方式产生远程控制指令。

4）结果检查：

通过仿 BM2711 模块用 1 ms 时钟频率分别对 48 路通道输出的指令脉冲信号进行计数，并将通道号和计数器值组成结构作为实测结果传输给 cmd _ scoreboard，cmd _ scoreboard 会自动查询期望动态数组，验证指令通道是否匹配，根据计数值检查是否满足 80 ms±10 ms 指令脉冲宽度要求，并给出日志文件，从而验证远程控制指令执行功能的正确性。

通过 rs422 _ monitor 监控仿 RS422 _ rec 模块产生的数据，形成实测动态数组发送给 rs422 _ scoreboard 进行和期望动态数组进行比对，并打印日志文件，从而验证远程控

制数据转发功能中通过异步串行接口转发远程控制数据功能的正确性；通过 lvds _ monitor 监控仿 LVDS _ rec 模块产生的数据，形成实测动态数组发送给 lvds _ scoreboard 进行和期望动态数组进行比对，并打印日志文件，从而验证远程控制数据转发功能中通过两线制通信接口转发远程控制数据功能的正确性。

5）覆盖率分析：

仿真验证覆盖率包括代码覆盖率和功能覆盖率。代码覆盖率统计的是仿真过程中程序代码语句、分支和状态机的执行情况，但是不能从功能点的角度考核设计功能的测试覆盖情况。功能覆盖率是针对功能点的通过比率进行覆盖率统计。本验证平台不仅验证代码覆盖率，还可以通过收集 VCID、数据包长度确定远程控制数据转发功能覆盖率 100%，通过收集通道指令号确定 48 路指令和异常指令覆盖和击中情况，从而确定远程控制指令执行功能覆盖率 100%。

3.2 验证平台的设计流程

1）设计 ykzd _ if：

设计虚拟接口模块 ykzd _ if 三线制输入接口 KP _ CLK1、LOCK _ A、KP _ DATA1，用于实现验证平台与 DUT 之间的数据通信。

2）设计队列：

由于要接受 2 种格式的远程控制上行数据，为了避免相互干扰，方便设计，本文设计了 2 种队列：ykzd _ 队列 _ zl 和 ykzd _ 队列。ykzd _ 队列 _ zl 为远程控制指令执行 Agent 中的事务级数据包，ykzd _ 队列为远程控制数据转发 Agent 中的事务级数据包。

根据功能需求，对 2 种队列的数据成员变量采用 constraint 进行约束设置。采用 field 机制对 2 种队列进行宏注册，设计 BCH 编码函数、加扰函数和 CRC 校验函数。

3）创建 Sequencer：

按照 UVM 机制创建的 sequencer 命名为 ykzd _ zl _ master _ sequencer 和 ykzd _ master _ sequencer。

4）创建驱动器 driver：

继承自 uvm _ driver，将 ykzd _ transaction _ zl 转换为信号级，并按照三线制接口时序驱动到 DUT 输入接口上。创建的 driver 命名为 ykzd _ zl _ master _ driver 和 ykzd _ master _ driver。通过虚拟接口 ykzd _ if 连接 DUT

5）创建监测器 monitor：

发送端 monitor 用于通过虚拟接口 ykzd _ if 收集 DUT 端口数据，并把端口数据转换为事务级数据。创建的发送端 monitor 命名为 ykzd _ zl _ master _ monitor（收集指令号）和 ykzd _ master _ monitor（收集数据帧长度和 VCID）。创建的接收端 monitor 命名为 RS422 _ monitor（根据串口协议接收转发串口数据）、LVDS _ monitor（在时钟上升沿接收数据，找到帧头并打包成数组）、BW2711 _ monitor（对各通道脉冲宽度用 1 ms 时钟进行计数）。

6）创建代理器 agent：

为了区分调用不同的 transaction，不同的 agent 代表不同的协议。创建的 agent 命名为 ykzd _ zl _ master _ agent 和 ykzd _ master _ agent。

7）创建参考模型 reference model：

M _ rs422 _ ref 参考模型对 ykzd _ master _ monitor 监控产生的需要转发的直接指令和间接指令进行参考模型设计用产生给期望结果；M _ lvds _ ref 参考模型对 ykzd _ master _ monitor 监控产生的需要转发的直接注数和软件数据进行参考模型设计用产生期望结果；Zl _ cmd _ ref 参考模型对 ykzd _ zl _ monitor 监控产生的重要指令进行参考模型设计得到期望的通道号序列。

8）创建记分板 scoreboard：

RS422 _ scoreboard 记分板对期望产生的直接指令和间接指令结果与实测结果进行一致性比对，正确打印 correct，错误打印 error、期望正确数据和实测错误数据；LVDS _ scoreboard 记分板期望产生的直接注数和软件数据与实测结果进行一致性比对，同理会自动打印正确和错误结果；cmd _ scoreboard 记分板对期望产生指令序列号和实测指令序列号进行比对，匹配一致则打印正确，否则给出丢失的指令号，同时当脉冲指令宽度不满足 80 ms±10 ms 时报错。

9）创建验证环境 env：

根据 UVM 机制 env 是验证平台组件的顶层，包含了验证平台的所有验证组件。本验证平台创建的 env 命名为 ykzd _ env，包含了代理器（ykzd _ zl _ master _ agent、ykzd _ master _ agent）、参考模型（M _ rs422 _ ref、M _ lvds _ ref、Zl _ cmd _ ref）、监视器（RS422 _ monitor、LVDS _ monitor、BW2711 _ monitor）、记分板（RS422 _ scoreboard、LVDS _ scoreboard、cmd _ scoreboard）。

10）创建虚拟激励产生器 virtual sequencer：

创建了 virtual sequencer，命名为 my _ vsqr0，my _ vsqr0 包含了指向 ykzd _ zl _ master _ sequencer 和 ykzd _ master _ sequencer 的指针。在基本类 ykzd _ base _ case 中，实例化 my _ vsqr0，将相应的 sequencer 赋值给 my _ vsqr0 中各 sequencer 的指针。

11）仿 SRAM 模型：

模拟 SRAM 根据输入的 sram _ addr [18：0]、sram _ data [7：0]、sram _ cs、sram _ we 和 sram _ oe 按照 SRAM 芯片手册的时序要求，将数据 sram _ data [7：0] 存储在存储区中或者将 sram _ data [7：0] 反馈输出给 DUT，同时根据 DUT 设计 SRAM 控制时序进行时序参数自动检测。

12）仿 PROM 模型：

模拟 PROM 存储指令码表，根据输入的 sram _ addr [11：0]、rom _ ce、rom _ oe 按照 PROM 芯片手册的时序要求，输出 sram _ data [7：0] 给 DUT，同时根据 DUT 设计 PROM 控制时序进行时序参数自动检测。

13）创建测试用例 test case：

测试用例命名为 ykzd _ case，继承于基本类 ykzd _ base _ case，而 ykzd _ base _ case 继承于 uvm _ test，包含了虚拟激励产生器 my _ vsqr0。

14）顶层设计：

顶层为 top _ tb，定义了虚拟接口 ykzd _ if，连接 DUT 和验证平台；例化了 DUT 和 SRAM、PROM、RS422 _ rev、LVDS _ rev、BW2711 模型，提供系统输入时钟和上电复位信号。

3.3　验证结果

该平台设计了多个 sequence，组成 sequence 的集合 sequence _ lib，设计了多个测试用例 ykzd _ case，每个测试用例 ykzd _ case 对应 1 个 sequence，分别单独测试远程控制指令执行功能、远程控制数据转发功能以及帧头错误、帧尾错误、CRC 校验错误、BCH 编码错误等异常情况。

将多个 sequence 用 virtual sequence 进行统一管理，将 virtual sequence 作为某个测试用例 ykzd _ case 的 default _ sequence，对远程控制指令执行功能、远程控制数据转发功能的正常情况、异常情况进行组合测试。通过 RS422 _ rev 模型和 LVDS _ rev 模型接收数据，并通过监控模块和参考模型输出在对应记分板中进行对比，检查远程控制数据转发功能的正确性；通过 BM2711 模型接收指令，除了通过观察波形检查远程控制指令执行情况，更通过 BM2711 模型检测脉冲宽度，通过监控模型和参考模型通道号序列在记分板中进行对比，检查远程控制指令执行功能的正确性和脉冲宽度的正确性，实现闭环的一体化仿真验证平台。

文中设计的仿真验证平台可通用于 Modelsim、Rivera、VCS 和 NC-sim 等主流支持 UVM 的仿真工具，图 6 是用 VCS 仿真器进行仿真的 DUT 接口仿真波形结果图。从仿真波形图看出，验证平台产生的激励数据如远程控制转发数据、远程控制指令数据通过 DUT 的三线制输入接口 KP _ CLK1、LOCK _ A、KP _ DATA 1 先后输入给 DUT，在输出端口上可看到相应的输出波形如两线制转发的 LVDS 远程控制数据和 48 路指令控制信号输出，说明功能正常执行。

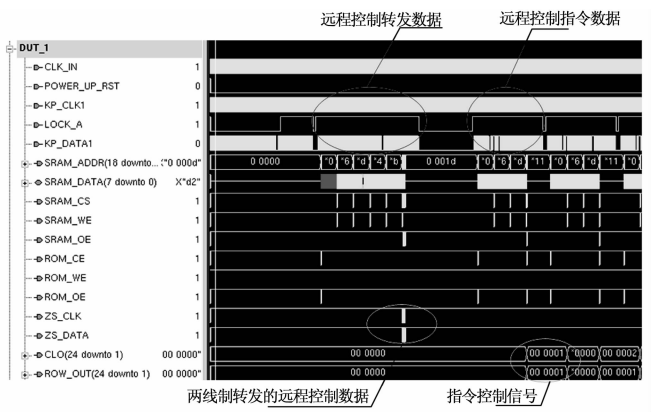

图 6　远程控制 FPGA 组合测试仿真波形图

4　结束语

UVM 验证方法学吸收了当前许多[15]验证方法学的优点，本文搭建了基于 UVM 的远程控制 FPGA 验证平台，实现了验证平台的各个组件。该验证平台结构层次分明、便于扩展和管理，具有受约束的随机验证和输出结果自动检查功能，有效的提高了验证的效率，确保了验证功能的正确性和鲁棒性。该验证平台具有验证效率高、完备性好、自动化程度高、通用性强、可扩展性强的优点，可以被其他验证项目所重用。

参考文献：

[1] 周　珊，杨雅雯，王金波. 航天高可靠 FPGA 测试技术研究[J]. 计算机技术与发展，2017，27（3）：1－2.

[2] 吕欣欣，刘淑芬. FPGA 通用验证平台建立方法研究[J]. 微电子学与计算机 2010，27（5）：46－49.

[3] 徐金甫，李森森. 采用 UVM 方法学实现验证的可重用与自动化[J]. 微电子学与计算机，2014，31（11）：14－17.

[4] 张　静，卜　刚，等. 基于 C _ Model 的 UVM 验证平台设计与实现[J]. 电子技术应用，2019，45（10）：100－104.

[5] 习建博. 一种基于 UVM 的 FPGA 通用可配置 UART 协议的验证方法[J]. 电子技术与软件工程，2017，13：159.

[6] 倪　伟，袁　琳，王笑天，等. 基于 UVM 的 I2S 验证 IP 设计[J]. 合肥工业大学学报（自然科学报），2018，41（1）：49－54.

[7] 习建博，朱　鹏，崔留争，等. 基于 UVM 方法的 FPGA 验证技术[J]. 电子科学技术，2016，3（3）：204－207.

[8] 刘　明，鲁建壮，等. 基于 UVM 构建 L2 模块级自动验证平台[C]//第十九届计算机工程与工艺年会暨第五届微处理器技术论坛论文集，487－492.

[9] 曾清乐，宋文强，李敬磊，等. 基于 UVM 的 FPGA 测试技术的研究[J]. 电脑与电信，2016，5：65－67.

[10] 克里斯·斯皮尔著. System Verilog 验证测试平台编写指南[M]. 张　春，等译. 北京：科学出版社，2009.

[11] 张　强. UVM 实战[M]. 北京：机械工业出版社，2014.

[12] 孟　宾，欧国东. 基于 UVM 验证方法学的 MCU 验证[C]//第十七届计算机工程与工艺年会暨第三届微处理器技术论坛论文集，286－288.

[13] 田　劲，王小力. 基于 UVM 验证方法学的 AES 模块级验证[J]. 微电子学与计算机，2012，29（8）：86－90.

[14] 卢　康. 基于 UVM 的 PCIE3.0PHY 测试芯片的验证[D]. 西安：西安电子科技大学，2020.

[15] 谢　峥，王　腾，雍姗姗，等. 一种基于 UVM 面向 RISC CPU 的可重用功能验证平台[J]. 北京大学学报：自然科学版，2014，50（2）：221－227.

[16] 祝周荣，关俊强，李前进，等. SV DPI 技术在 FPGA 仿真验证的应用探讨[J]. 计算机测量与控制，2018，26（6）：264－267.

[17] 何铭俊，陆文高，曾　鸿，等. 一种卫星任务解译闭环仿真验证系统的设计与实现[J]. 计算机测量与控制，2019，27（1）：271－274.