文章编号:1001 - 9081(2009)09 - 2527 - 03

基于 Synopsys VMM 方法的 FPGA 验证技术

吕欣欣,刘淑芬
(北京控制工程研究所,北京 100190)
(lvxinxin1985@ gmail. com)

摘 要:针对可编程器件在数字系统设计领域日益显现的重要性,分析了基于现场可编程门阵列(FPGA)的硬件设计的质量保证方法,指出必须对 FPGA 设计进行充分的验证以提高相应产品的可靠性。从验证方法和方法学角度阐述了验证平台的发展趋势,比较了当前主流的验证方法学,基于 Synopsys VMM 方法提出并实现了一种层次化的通用验证技术,运用该技术搭建的验证平台已在工程实践中得到应用,验证结果表明,在保证平台通用性的同时提高了验证效率。

关键词:现场可编程门阵列;硬件设计;VMM;验证平台;验证方法学
中图分类号: TP302 **文献标志码**:A

FPGA verification technology based on Synopsys VMM

Lü Xin-xin, LIU Shu-fen
(*Beijing Institute of Control Engineering, Beijing* 100190, *China*)

Abstract: Field Programmable Gate Array (FPGA) designs should be thoroughly verified in order to enhance the reliability of corresponding product, with the emerging importance of programmable device in the field of implementation of digital system. The authors analyzed the methods for quality hardware design implemented on FPGA, depicted the trends of verification in terms of method and methodology, as well as conducted a comparison of mainstream verification methodologies. Based on Synopsys Verification Methodology Manual (VMM), the authors proposed and implemented a layered general-purpose verification technique that has been utilized to construct verification platform in application. The experimental results show that this technique can not only maintain the generability of platform but also improve the efficiency of verification.

Key words: Field Programmable Gate Array (FPGA); hardware design; Verification Methodology Manual (VMM); verification platform; verification methodology

0 引言

FPGA 具有集成度高、功耗低、使用灵活和开发周期短等特点,在数字系统设计领域中得到广泛采用。随着设计规模的不断增大和复杂度的上升,验证任务已成为整个设计流程的瓶颈,因此,对 FPGA 设计进行高效和充分的验证是提高 FPGA 设计可靠性的关键。

传统的功能仿真验证采用 HDL 语言对被验证对象(Design Under Test, DUT)编写定向测试用例,人工观察输出波形并判断验证结果。缺点是难以达到预期覆盖率、抽象层次低、通用性差。

本文在分析验证平台发展的基础上,对通用验证技术进行研究,基于 Synopsys VMM 验证方法提出并实现了一种层次化的通用验证平台。

1 验证平台的发展

验证平台,即通常所说的"testbench",用来为硬件设计产生特定的输入测试激励,并观测设计输出的响应是否与预期的结果一致,进而判断设计的功能正确性。验证平台使用硬件描述语言或硬件验证语言进行编写,结合多种验证方法实现,以一套定义完整的验证方法学作为指导原则。本文从验证方法和验证方法学两个角度对验证平台的发展趋势进行研究。

1.1 验证方法的发展

随着验证技术的逐步发展,在验证语言和各种 EDA 工具的支持下,开始利用各种新的功能验证方法来建立验证平台,按照发展顺序大致包括 5 种方法。

1)定向测试用例生成方法。

指验证人员对可能出现的功能问题而编写定向的测试用例,因此需要大量测试用例才能覆盖尽可能多的输出情况。目前该方法只在验证边角情况时采用。

2)带约束的随机验证方法[1]。

指对测试用例进行带有约束的随机化处理。该方法在最初建立验证环境时需要花一定时间编写约束文件,而随后生成测试用例时,只需改变约束,即可提高验证覆盖率及缩短验证时间。

3)基于事务的验证方法[2]。

基于事务的验证是利用总线功能模型和产生事务级激励的测试用例来建立验证平台。总线功能模型根据接口协议要求将事务级动作转换成相关互连线信号上的操作。该方法使抽象层次提高到事务级,更利于验证平台中模块的重用。

4)覆盖率驱动的验证方法[2]。

验证需要有覆盖率的度量来衡量验证进度。一是为了识别验证漏洞,以指导下一步验证的方向;二是指示验证的完全性,逐步提高覆盖标准,帮助确定是否已足够稳定进而结束。

收稿日期:2009 - 03 - 19;**修回日期**:2009 - 05 - 15。

作者简介:吕欣欣(1985 -),女,山东济南人,硕士研究生,主要研究方向:星载计算机; 刘淑芬(1961 -),女,吉林长春人,研究员,主要研究方向:星载计算机。

5)基于断言的验证方法[1]。

在验证过程中,仅靠人的智慧较难判断跨多个时钟周期等操作的时序功能是否符合规范,为此提出基于断言的验证。该方法包括可观察性可控性高,以及可重用性好等优点。

1.2 验证方法学的发展

验证方法学是以已有验证方法为基础,在相应工具和验证语言的支持下,提供模块级或系统级的验证解决方案。

目前的验证方法学主要包括 Synopsys 公司提出的 RVM(Reference Verification Methodology)[3]、ARM 和 Synopsys 公司联合推出的 VMM(Verification Methodology Manual)[1]、Mentor 公司提出的 AVM(Advanced Verification Methodology)[4]、Cadence 公司提出的 UVM(Unified Verification Methodology)[5]以及 Cadence 公司和 Mentor 公司联合推出的 OVM(Open Verification Methodology)[6]等验证方法学。

RVM 作为 VMM 的前身,正在逐步被 VMM 所取代;AVM 和 UVM 最高抽象层次为事务级,尚未提高到场景级;UVM 对 SOC 设计的意义重大,但需要在系统级设计时建立功能虚拟原型,不适合已设计完成的模块级验证;于 2008 年正式发布的 OVM 将逐步取代 AVM 和 UVM,但尚未广泛应用。VMM 是目前实际应用最为广泛的验证方法学,用户可以选择使用基类库或开发自己的基类并利用面向对象特性进行扩展。对五种验证方法学进行比较,结果如表 1 所示。

表 1 五种验证方法学比较

验证方法	方法学侧重点	EDA 验证工具环境	主要支持的验证语言	最高的抽象层次	基类库是否开源
RVM	层次化验证	Discovery 验证平台	OpenVera	场景级	否
VMM	RTL 级模块验证和系统级验证	Discovery 验证平台	SystemVerilog	场景级	是
AVM	强调软件架构和软件技术	Questa 验证平台	SystemC/SystemVerilog	事务级	是
UVM	强调系统级设计和验证	Incisive 验证平台	E	事务级	否
OVM	强调开源和不同仿真器之间的透明性	支持不同工具	SystemVerilog	事务级	是

2 基于 VMM 的 FPGA 通用验证平台

面向实际应用领域,基于 VMM 验证方法学的思想,采用面向对象的验证语言 SystemVerilog,从平台结构、测试用例和提取通用基类三方面入手,对 FPGA 设计建立通用验证平台,如图 1 所示。

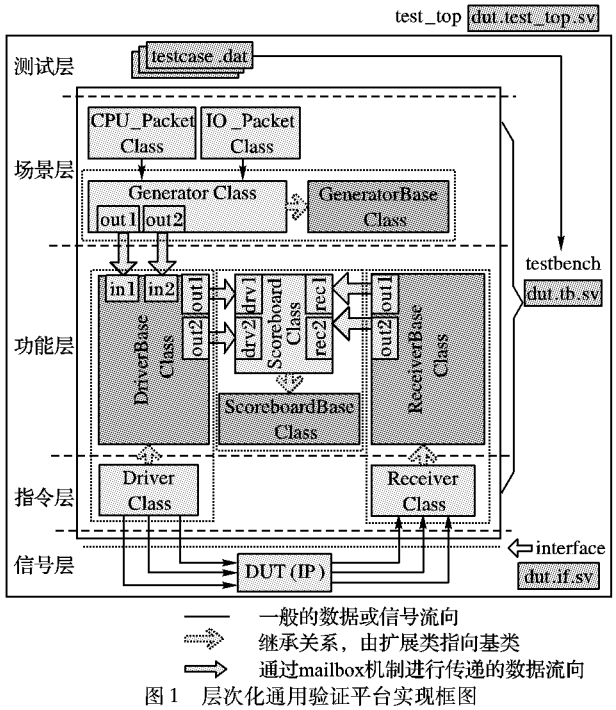

图 1 层次化通用验证平台实现框图

2.1 平台结构

验证平台要在不同的 DUT 之间通用,首先验证平台结构要被有效地模块化,使得改变平台各模块变得容易;其次验证平台必须有明确的层次化定义,清晰地定义层与层之间的接口,保持各层之间有一定的独立性。

对 FPGA 设计所建立的通用验证平台包括五层结构和主控制文件。

1)测试层。

测试层主要包括文本测试用例文件(testcase.dat)。该文件中的测试命令需要借助场景层的静态数据对象为载体,经由功能层和指令层的动态驱动,最终传送到 DUT 的端口信号。该层抽象级别最高。

2)场景层。

场景是指 DUT 端口上一组相关信号的组合,这组信号组合起来表达一个确定含义,比如 CPU 对某个外设的写操作即是一个场景,需要相关信号的有效组合。这些场景由该层中的产生器(Generator 和 GeneratorBase)产生的数据结构(CPU_Packet 和 IO_Packet)来表示,其中数据结构和 DUT 的端口信号相对应,是负责把测试场景运送到 DUT 端口的载体。

场景层的特点是所有场景都在仿真 0 时刻产生,即是静态的,且不与 DUT 端口信号直接相连。

3)功能层。

功能层和指令层负责对场景层产生的静态场景进行动态驱动,仿真时间向前推进,将数据结构中的信号值最终传送到 DUT 端口上。

功能层包括不与 DUT 实际端口相连接的驱动器(DriverBase)、接收器(ReceiverBase)和自比对模块(Scoreboard 和 ScoreboardBase)。在这一层中,主要完成数据自动比对功能,并且与 DUT 类型无关。功能层不与 DUT 引脚信号直接相连,在改变 DUT 时不需要进行修改,因此通用性强。

4)指令层。

指令层主要包括与 DUT 实际端口相连接的驱动器(Driver)和接收器(Receiver)部分。在这一层中,驱动器和接收器调用功能层中各基类接口进行相应的扩展,动态地向 DUT 直接驱动测试激励并从 DUT 输出端进行动态接收,由于该层与 DUT 实际端口相连,因此抽象级别较低。

5)信号层。

信号层主要包括 DUT 本身,以及和 DUT 相关的一些其他功能模块。

6)主控制文件。

主控制文件(dut.test_top、dut.tb、dut.if)的作用是控制和协调验证平台中各层模块的工作,使各模块在仿真时间内按

照一定顺序依次启动,有效地完成整个验证过程。用户在使用该通用验证平台对 DUT 进行验证时,只需要对主控制文件进行相应的配置,即可在不同 DUT 之间实现通用。

2.2　测试用例

将测试用例文件用单独的格式存放,并与验证平台分开,这样可以增强验证平台的可重用性和易维护性等[7]。

CPU 的外设接口通常包含地址线、数据线和控制线三类,对这一类接口相关的 DUT 进行操作一般包含复位、读/写操作等。写操作时,可以写入定值(如设置控制寄存器中的控制字),也可以为了验证写入随机值(如模拟 CPU 要发送给外设的一个随机值)。因此,需要将测试用例中测试激励的编写格式进行统一。用户只需根据命令格式要求并结合 DUT 的功能来编写文本文件,就可完成测试用例的开发。

测试用例由主控制文件使用 SystemVerilog 中系统函数 MYMreadmemh 将测试用例文件中每行内容分别读入 32 位宽的二维数组 read_test_file 中。

二维数组的基本格式及含义如图 2 所示。

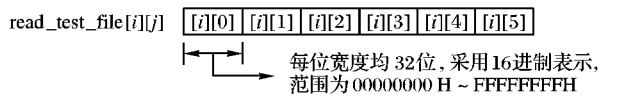

图 2　read_test_file 基本格式及含义

每行测试激励包含下列 6 项信息:

1) read_test_file[i][0] 表示片选信号,"0" 有效,"1" 无效;

2) read_test_file[i][1] 表示复位信号,"0" 有效,"1" 无效;

3) read_test_file[i][2] 表示最多支持 32 位有效地址信号,实际使用时根据 DUT 需要的地址信号宽度赋值,范围为 00000000H ~ FFFFFFFFH,"X" 表示地址无效;

4) read_test_file[i][3] 表示 CPU 的读/写信号,"0" 表示 CPU 写有效,"1" 表示 CPU 读有效,"X" 表示读写均无效;

5) read_test_file[i][4] 表示要写入的定值数据,如对控制寄存器写入控制字,或者定义空闲状态的周期个数等,范围为 00000000H ~ FFFFFFFFH,"X" 表示不写入定值,而是写入由验证平台产生的随机化数据;

6) read_test_file[i][5] 表示 DUT 类型,最多支持 2^{32} 个 DUT。如"0" 表示 DUT 为 UART,"1" 表示 DUT 为 GPIO。

测试激励命令主要包括 5 种:复位命令、CPU 写入定值命令、CPU 写入随机值命令、CPU 读命令和空闲状态命令。如 "0 1 00000004 0 X 1",表示 CPU 向 GPIO 的发送数据寄存器中写入要输出的并行数据,并行数据在平台中随机产生。对以上五种命令进行组合使用即可实现对不同 DUT 不同功能测试用例的编写。

2.3　通用基类

验证平台中有相当一部分工作与处理器输入信号相关,而针对同一处理器的输入信号的时序是已经确定的。因此,采用面向对象思想,将这一部分与处理器输入信号相关的固定操作和数据结构提取到通用的基类中,以避免用户使用平台时对其进行错误修改。

在该平台的场景层和功能层一共提取出 4 个基类,分别是 GeneratorBase、DriverBase、ReceiverBase 和 ScoreboardBase。这些基类给扩展类仅提供一个使用的黑盒接口,在更换 DUT 时,利用基类提供的接口函数,能够迅速形成新的扩展类,缩短验证时间,提高验证效率。

3　通用验证平台的应用

在本文提出的通用验证平台上,选取两个已经过验证的较成熟的商业 IP 核,Gaisler Research 公司[8] 的 GPIO 和 UART 作为验证对象进行验证,从而对该平台的通用性和有效性进行验证。验证使用的仿真器为 VCS-MX2008.12,对这两个不同的 DUT 分别进行验证,均记录了正确的数据比对结果。

为便于比较,本文选择 Gaisler Research 公司提供的验证程序作为传统验证方法的代表进行参考,如表 2 所示。

表 2　传统功能仿真验证和通用验证平台比较

验证平台	验证对象	代码覆盖率/%	条件覆盖率/%	分支覆盖率/%	功能覆盖率/%	仿真周期数
传统功能仿真验证	GPIO	68.42	49.41	50.82	不可收集	约 29 600
	UART	83.62	50.63	64.52	不可收集	约 50 758
通用验证平台	GPIO	100.00	100.00	99.21	100	约 13 200
	UART	98.98	93.67	96.87	100	约 27 750

分别使用传统功能仿真验证方法和通用验证平台进行覆盖率和仿真周期数的统计,结果表明,通用验证平台具有一定的通用性,比传统功能仿真验证提高大约 50% 的时间效率,有效地提高验证覆盖率。

4　通用验证平台的优势

该层次化通用验证平台与针对特定 DUT 的传统功能仿真验证相比,具有以下优势:

1) 采用文本测试用例文件的形式且统一测试激励编写的命令格式,实现测试用例文件的编写标准化,从而在短时间内完成更多测试用例的开发,提高验证覆盖率;

2) 采用自比对模块,仿真时期可不再存储波形文件,节省仿真资源,提高仿真器运行速度,并且不需要人工观察输出波形来参与结果的判读,有效避免了人工判读再次引入其他失误的风险;

3) 专门用于收集功能覆盖率的覆盖率组,实现了覆盖率驱动的验证;

4) 通用基类按照处理器时序提取,可形成通用有效的 VIP(Verification IP),VIP 的使用[9] 则可以有效地缩短验证时间,提高验证效率。

5　结语

本文基于 VMM 验证方法学实现了一个层次化的通用验证平台,该平台采用面向对象思想解决了传统功能仿真验证中存在的通用性差效率低等问题,利用该平台对两个 DUT 进行验证所得的统计数据表明,该方法建立起的验证平台具备一定的通用性,可有效地提高验证覆盖率和验证效率。

参考文献:

[1] BERGERON J, NIGHTINGALE A. Verification methodology manual for SystemVerilog[M]. New York: Springer-Verlag, 2006.

[2] BERGERON J. Writing testbenches using SystemVerilog[M]. New York: Springer-Verlag, 2006.

(下转第 2533 页)

即对于质量因素集 $U'_i = \{u_{i_1}, u_{i_2}, \cdots, u_{i_{n_i}}\}, i = 1, 2, \cdots, s$,其中 U'_i 是一级质量因素,u_{ij} 是二级质量因素。若质量因素 U'_i 属于 I 或 II 象限,将根据质量因素 u_{ij} 位于不同象限分为四种情况:a) u_{ij} 属于 I 象限,则此类质量因素应该采用稳定增强策略;b) u_{ij} 属于 II 象限,则此类质量因素应该采用保持现状策略;c) u_{ij} 属于 III 象限,则此类质量因素应该采用改进策略;d) u_{ij} 属于 IV 象限,则此类质量因素应该采用优先改进策略;若质量因素 U'_i 属于 III 或 IV 象限,又将根据质量因素 u_{ij} 位于不同象限分为两种情况:a) u_{ij} 属于 I 或 II 象限,则此类质量因素应该采用改进策略;b) u_{ij} 属于 III 或 IV 象限,则此类质量因素应该采用优先改进策略。

表 1　一二级质量因素综合评价分析表

一级质量因素		二级质量因素 $u_{ij}(i = 1, 2, \cdots, s;\ j = 1, 2, \cdots, n_i)$			
		I	II	III	IV
U'_i ($i = 1, 2, \cdots, s$)	I	①	②	③	④
	II	①	②	③	④
	III	③	③	④	④
	IV	③	③	④	④

采用的策略:①稳定增强;②保持现状;③改进;④优先改进。

3　实例

下面以已开发的实验室开放管理平台来简要说明基于最小置信度和评价分析的软件质量模糊综合评价改进方案的具体应用。限于篇幅原因,以下仅给出计算结果。

1)等级评价。

按照软件质量模糊综合评价体系,将 U 分为 $U_1, U_2, U_3, U_4, U_5, U_6$ 其中 $U_1 = \{u_1, u_2, u_3, u_4, u_5\}$, $U_2 = \{u_6, u_7, u_8, u_9\}$, $U_3 = \{u_{10}, u_{11}, u_{12}, u_{13}, u_{14}\}$, $U_4 = \{u_{15}, u_{16}, u_{17}, u_{18}, u_{19}\}$, $U_5 = \{u_{20}, u_{21}, u_{22}, u_{23}, u_{24}\}$, $U_6 = \{u_{25}, u_{26}, u_{27}\}$。

由式(2) ~ (6)可得一级评价向量 $\boldsymbol{B} = \boldsymbol{A} \circ \boldsymbol{R} = (0.23, 0.26, 0.16, 0.21, 0.14)$,最后根据最小置信度准则(7),可得 $k_0 = 3$,因此该实验室开放管理平台的质量评价等级为 v_3,即较好。

2) 评价分析。

设量分权重评测尺度为 5 级:非常重要(0.9),很重要(0.7),重要(0.6),一般(0.3),不重要(0.1),即 $\boldsymbol{E} = (e_1, e_2, e_3, e_4, e_5)$。根据评价集的划分,满意度测评尺度也分为 5 级,分别是:优秀(0.9),良好(0.7),较好(0.6),一般(0.3),差(0.1)。

由式(8) ~ (14)可得:第二级量分的质量因素权重向量为 $\boldsymbol{W}_1 = (0.56, 0.61, 0.63, 0.49, 0.41)$, $\boldsymbol{W}_2 = (0.63, 0.44, 0.69, 0.55)$, $\boldsymbol{W}_3 = (0.66, 0.62, 0.61, 0.42, 0.63)$, $\boldsymbol{W}_4 = (0.61, 0.63, 0.66, 0.40, 0.51)$, $\boldsymbol{W}_5 = (0.51, 0.54, 0.60, 0.67, 0.41)$, $\boldsymbol{W}_6 = (0.69, 0.47, 0.58)$;

第二级质量因素满意度向量为 $\boldsymbol{D}_1 = (0.58, 0.63, 0.53, 0.61, 0.49)$, $\boldsymbol{D}_2 = (0.62, 0.53, 0.58, 0.65)$, $\boldsymbol{D}_3 = (0.65, 0.63, 0.61, 0.51, 0.60)$, $\boldsymbol{D}_4 = (0.60, 0.61, 0.61, 0.48, 0.61)$, $\boldsymbol{D}_5 = (0.41, 0.53, 0.64, 0.59, 0.51)$, $\boldsymbol{D}_6 = (0.58, 0.46, 0.61)$;

第一级量分的质量因素权重向量为 $\boldsymbol{W} = (0.51, 0.63, 0.62, 0.55, 0.59, 0.70)$;

第一级质量因素满意度向量为 $\boldsymbol{D} = (0.56, 0.58, 0.61, 0.59, 0.54, 0.55)$。

根据上述一二级质量因素综合评价分析表,经过综合评价可得出:需要保持现状的质量因素是:u_4(安全性)、u_7(成熟性)、u_{13}(吸引性)、u_{24}(移植依从性);需要改进的质量因素是:u_5(功能依从性)、u_{18}(易测试性)、u_{26}(资源利用性);需要优先改进的质量因素是:u_{20}(适应性);此外剩下的质量因素都需要稳定增强。

4　结语

本文对现行的软件质量模糊综合评价方法进行改进,实例结果表明,改进后的基于最小置信度和评价分析的软件质量模糊综合评价方法可以兼顾所有质量因素的权重大小,使软件质量评价结果更加合适,以及通过对评价结果的分析,更容易使软件质量的好坏与价值得以判断和表现出来。

参考文献:

[1] ISO/IEC 9126 软件工程规范[S]. 北京: 中国标准出版社, 2001.

[2] 朱三元. 软件质量及其评价技术[M]. 北京: 清华大学出版社, 1990.

[3] BEVAN N. Measuring usability as quality of use [J]. Journal of Software Quality, 1995, 4(2): 115 - 150.

[4] WONG B, JEFFERY R. A framework for software quality evaluation [C]// PROFES 2002: Proceedings of the 4th International Conference on Product Focused Software Process Improvement. Berlin: Springer-Verlag, 2002: 103 - 108.

[5] 杨扬. 计算机软件质量模糊综合评价方法[J]. 小型微型计算机系统, 2000, 21(3): 313 - 315.

[6] 李良宝, 韩喜双. 软件质量的多级模糊综合评价[J]. 哈尔滨工业大学学报, 2003, 35(7): 812 - 819.

[7] 周津慧, 王宗, 杨宗奎, 等. 基于模糊评价方法的软件质量评价研究[J]. 系统工程与电子技术, 2004, 26(7): 987 - 991.

[8] 陆鑫, 廖建明. 基于模糊集理论的软件质量评估研究[J]. 电子科技大学学报, 2007, 36(3): 652 - 655.

[9] 方海光. 我国教育软件价值评测研究[D]. 成都: 中国科学院成都计算机应用研究所, 2006.

(上接第 2529 页)

[3] Synopsys, Inc. Reference verification methodology user guide version 8.6 [EB/OL]. [2009 - 02 - 20]. http://www.synopsys.com/.

[4] GLASSER M. Advanced verification methodology cookbook version 2.0[EB/OL]. (2006 - 07 - 24)[2009 - 02 - 10]. http://www.mentor.com/.

[5] Cadence Design Systems, Inc. The unified verification methodology white paper [EB/OL]. (2005 - 02 - 01)[2009 - 01 - 05]. http://www.cadence.com/.

[6] COLGAN J. Open verification methodology relieves inefficiencies [EB/OL]. (2007 - 09 - 07)[2009 - 01 - 10]. http://electronic-design.com/Articles.

[7] TSAI T. Techniques for selective reuse of verification components in hierarchical verification of large designs [C]// SNUG: Synopsys Users Group. San Jose: [s.n.], 2008: 97 - 106.

[8] GAISLER J, CATOVIC E, ISOMÄKI M. GRLIB IP Core user's manual version 1.0.19 [S]. Sweden: Gaisler Research, 2008.

[9] STÖHR B, SIMMONS M, GEISHAUSER J. FlexBench: Reuse of verification IP to increase productivity [C]// Proceedings of the 2002 Design, Automation and Test in Europe Conference and Exhibition. Washington, DC: IEEE Computer Society, 2002: 35 - 41.