The Platform Discuss Based VMM Verification Methodology

ZHENG Jian-hong[1], LI Zhen-zhong[2]

(College of Communications and Information Engineering, Chongqing University of Posts and Telecommunications, Chongqing 400065,China)

Email: 1. zhengjianhongi@cqcyit.com, 2. lzz_3816@yahoo.com.cn

Abstract: VMM is a verification methodology, it based System Verilog language, This paper describes the structure of VMM random verification platform, and introduces the characteristics of VMM verification methodology,in the end，Through constuct the IIS platform, to describe of the key links according to actual example,Give the IIS functional verification codes coverage result based on VMM verification platform and the Digital-analog simulation waveform.

Keywords: VMM; random; coverage; IIS

基于 VMM 方法学随机验证平台研究

郑建宏 [1]，李振中 [2]

(重庆邮电大学 通信与信息工程学院 重庆 400065 中国)

Email: 1. zhengjianhongi@cqcyit.com, 2. lzz_3816@yahoo.com.cn

【摘要】VMM 是一种基于 SV 语言的验证方法学，本文简述了 VMM 随机验证平台的架构，并详细介绍了 VMM 验证方法学的特性，最后通过 IIS 平台的搭建，对重点环节进行实例描述，给出了基于 VMM 验证平台下 IIS 功能验证的代码覆盖结果，以及仿真数模波形。

【关键词】: VMM ;随机;覆盖;IIS

1 引言

随着数字芯片的高速发展，芯片设计的复杂度也在提升，根据 Colllett 公司的调查，造成芯片流片失败而重新制造的原因大多是因为其功能验证不足而导致，所以伴随芯片规模的扩大，功能验证面临巨大的挑战，而在整个芯片开发过程中，功能验证占用了较多的工作量，并且影响了开发周期，如何减少验证工作量，提高验证效率已成为验证中考虑的问题。

VMM 验证方法学是一种新型的验证方法学学，VMM 中涉及的技术最初是用于 Open Vera 语言的，2005 年才被扩展用于 System Verilog，针对 System Verilog 这个硬件验证语言（HVL），它与硬件描述语言（HDL）相比具有：受约束的随机激励生成、功能覆盖率、面向对象的编程等典型性质，所以 VMM 验证方法学中通过采用面向对象的程序设计（object oriented programming, OOP）来确保平台代码量的最小化和代码重用的最大化，通过随机激励生成来遍历所有功能点和功能覆盖来确保功能验证的完备性。VMM 为我们提供了三个基类，分别为 vmm_data, vmm_env, vmm_xactor，我们在平台中可以继承这三个基类来完成一定的功能，图 1 为 VMM 方法学随机验证平台构架。

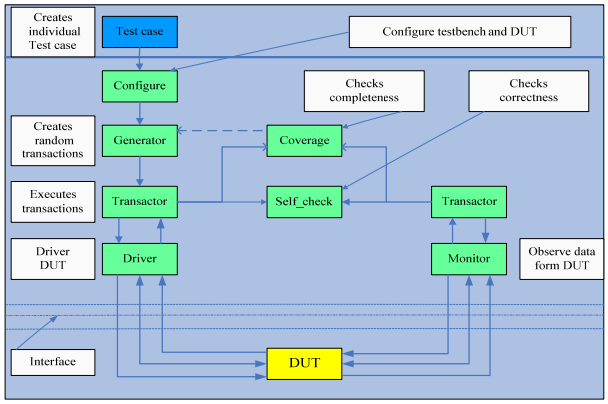

Figure 1.The testbench Environment/Architecture

图 1 VMM 方法学随机验证平台构架

978-1-935068-33-4 © 2010 SciRes.

1.1 平台构架基本介绍

测试例：VMM 方法学是 Top_Down 的执行方法，它强调一个平台多个测试例，它通过前期直接测试例，中期大量随机测试例，后期边角测试例来确保功能验证的完备性，在 testcase 中可以对任何类以及变量进行约束，以达到不同功能的验证。

配置器：VMM 是一种基于随机的验证方法学，为了让 DUT 工作在不同的功能下，平台满足 DUT 不同功能的验证，我们应该让传输的事务以及对平台的配置和 DUT 的配置是一些随机数，所以在配置器中我们对这些随机数进行随机类型的定义，并且对随机条件进行约束。

数据生成器：在传统的编程，例如 verilog、C 是将数据和函数分开的，而在面向对象编程中，例如 Vera、SV3.1 是将数据根使用它们的程序紧密地结合一起，也就是我们所谓的事务和数据包。在 VMM 中，数据生成是以工厂模式进行工作，在事务中定义一定的数据格式以及功能函数，数据生成器就是按照事务中的模式一一生产。在事务中也可以通过宏来定义通道，此通道也只能传输此类型的事务，做到一一匹配。

驱动器（监测器）：在驱动器中，它接受来自数据生成器中的事务，并将其按照一定的协议对 DUT 进行驱动，或者接受 DUT 传输出来的数据，并将这些数据包传入对比器进行比较。

比较器：驱动器和检测器将数据一一传入比较器在比较器中进行比较，在比较器中数据通过对比函数（完成特定功能）来检查数据在 DUT 的传输过程中是否正确的完成了 DUT 的某种特定功能来检查 DUT 功能的正确性。在信号时序的检查上，我们不能通过函数去完成，而是通过 SVA（System Verilog Assertions）去严格精确的检查时序功能。

覆盖：在 VMM 方法学验证中，我们定义一些 DUT 的功能点，当所有的功能点在不同的随机配置值下一一运行过，我们可以说此 DUT 功能验证基本完成，然而此功能点是否运行过就由覆盖来判断。

接口：与 RVM 验证方法学相比，VMM 采用了兼容 DUT 语言的 SV 语言，无须再去定义一些虚拟端口与 DUT 去绑定，而是直接用 SV 语言直接生成与 DUT 相连的接口，在我们的平台中直接可以对此接口操作而达到对 DUT 的相应操作，大大简化了平台的复杂性。

2 VMM 特性

2.1 面向对象编程

对结构化编程语言，例如 Verilog 和 C 语言来讲，他们的数据结构和使用这些数据的代码之间存在很大的沟壑，而面向对象编程（OOP）使用户能够创建复杂的数据类型，并且将他们跟使用这些数据类型的程序紧密地结合在一起，用户可以在更加抽象的层次建立测试平台和系统级模型。

OOP 真正强大的地方在于它能够继承现有类，它不仅大幅度提升了平台的灵活度，同时又降低了平台的复杂度，在进行不同功能的验证时候，我们仅需改变扩展类便可对不同的功能进行验证，这必须提及其蓝图（Blue Print）模式以及虚方法。

在数据生成器中，如果将事务的对象构建（new）并立即使用，那么生成器生成的数据仅受默认事务类的约束，但是在 DUT 不同功能面前，它可能需要多种类型的事务，这样，我们的数据生成器就无法满足其要求，蓝图模式便解决了这个难题，它使得我们在想构建一个事务发生器的时候，不需要知道怎样建立各种类型的事务，只需要能够根据给定的事务建立一个类似的新的事务。我们先构建一个事务对象的蓝图，然后修改它的约束，甚至使用一个扩展对象替换它，然后在我们随机化这个蓝图的时候，它就会具有我们想赋予的随机值，接着复制这个对象，并将拷贝的值发送给下游的事务处理器。值得注意的是蓝图对象在一个地方构建但是在另一个地方使用，我们对蓝图的改变必须在环境的完成和运行阶段去完成。

OOP 中多个子程序使用一个共同的名字的现象叫做“多态”，它解决了如何在物理内存很小的情况下让处理器能够对一个很大的地址空间寻址。在 VMM 中提供的基类或者说我们自己创建的基类，我们在事先是无法预估一些方法去实现所有的功能，所有我们将这些方法设置成虚拟方法，采用修饰符 virtual 来区分，这样，我们通过继承以及多态性去完善各个基类的方法功能，在使用时候，我们将基类句柄指向扩展类对象，即可调用被完善的基类方法。

2.2 受约束的随机激励生成

随着涉及变得越来越大，要生产一个完整的激励集来测试涉及的功能也变得越来越困难了。可以编写一个定向测试集来检查某些功能项，但当一个项目的功能项目成倍增加时，编写足够的定向测试集就不可能了，解决的办法是采用受约束的随机测试法（CRT）自动产生测试集，定向测试集能找到你认为可能存在的 Bug，CRT 方法通过随机激励，可以找到你都无法

确定的 Bug。

在随机化激励的时候，我们仅仅对数据进行随机这种方法对找到 Bug 的能力是有限的，而具有挑战性的 Bug 大都在控制路径里，因此对 DUT 里面所有的关键点都采用随机化技术。一般对 DUT 的配置，环境的配置以及数据、延时和错误异常都要采用随机。

对于随机，我们首先要有一个带有随机变量的简单类，在需要随机的时候，我们可以调用很多随机函数对这些随机变量进行随机，可以约束随机变量在一个范围中随机，也可以对随机变量所随机到的随机值进行权重的设置，可以按照具体情况对随机进行关闭以及打开。在调用 randomize()函数的时候，VMM 会自动调用 pre_randomize 和 post_randomize 函数，对随机随机变量进行随机前处理和随机后处理。总之，对我们需要某种特定格式的随机数据，我们都可以通过约束的方法以及技巧得到。

2.3 覆盖

随着各种设计变得越来越复杂，采用受约束的随机测试方法(CRT)是对他们进行全面验证的唯一有效途径，但是测试平台在所有随机情况下游走，无法确定已达到最终目标，覆盖率衡量设计特性已经被测试程序测试过的一个有效指标。图 2 验证过程覆盖率增长图。

首先在创建平台时，可以覆盖到 DUT 极少数功能点，随后开始大量的随机验证，在较短时间内 DUT 覆盖率急速上升，接下来如果覆盖率增速放慢，那就需要添加额外的约束来产生更加“有意义”的数据来再次进行约束性随机验证，最后，对覆盖率接近 100%的时候，分析覆盖率，通过直接测试去保证一些边角以及异常情况的验证，使我们的验证达到目标。

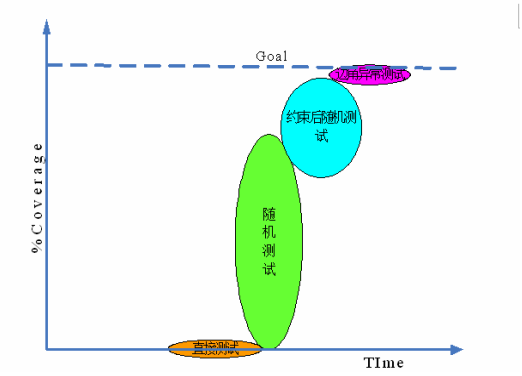

Figure 2.phases of random stimules based verification

图 2 随机验证各阶段

对于覆盖，我们一般进行代码覆盖和功能覆盖。在代码覆盖中，我们一般会去衡量行覆盖率、列覆盖率、翻转覆盖率以及有限状态机覆盖，我们不用添加任何额外的 HDL 代码，工具会通过分析源代码来帮我们完成代码覆盖统计，并创建相应的数据库。在功能覆盖中，我们的目的就是确保设计在实际环境中的行为正确，在验证前我们规定 DUT 必须去完成某种功能（即出现某种配置），当验证结束之后，我们通过功能覆盖率去了解具体验证了 DUT 的哪写功能。

3 基于 VMM 方法学的 IIS 验证

3.1 IIS 概述

IIS（inter-IC sound）总线是一种专门用于数字音频的总线，它有四根引脚，两根数据引脚（DOUT 和 DIN）一根位率时钟引脚（BCLK）以及一根左右通道选择引脚（LRCLK）。

IIS 工作在 SLAVE 模式下，它的 BCLK 以及 LRCLKL 由外部音频设备提供，工作在 MASTER 模式下，它的 BCLK 以及 LRCLKL 是由自己产生的，它可以进行数据的发送，也可以进行数据的接收，在每一个 LRCLK 电平下伴随着每一个 BCLK 的跳变沿，IIS 收发一个 16-32bit 的有效音频数据。

3.2 IIS 随机平台介绍

Test:针对 IIS 不同的功能，我们构造不同的的测试例，在每一个测试例下，对配置器进行继承，对随机的变量进行约束，使 IIS 工作在特定的某些功能下，简单代码如下：

```
Program automatic test(i2s_io io);
`include” vmm,sv”
`include” configuration.sv”
`include” Driver.sv”
class newcfg extend dut_cfg;
constraint new_valid{
                                type_tinside{STD};
                                }
endclass
initial begin
    cfgt=new();        //新的配置类进行例化
    env=new();          //环境初始化
    env.gen_cfg();   //产生新配置的变量
    env.run();          //环境开始运行
end
endprogram
```

在测试例中，env.run() 中包括了基类的九个函数，他们控制了整个环境的运行流程，从环境中各个类的例化、配置到开始、结束。

978-1-935068-33-4 © 2010 SciRes.

Test_top: 环境的最顶层，它进行接口与 IIS 端口的连接，例化测试例以及接口类，并且模拟产生 IIS 工作时需要的工作时钟，我们可以根据脚本由 IIS 代码直接生成 top 文件

Interface:连接 IIS 端口，在 VMM 方法学中，我们仅需要对接口信号进行驱动，就可以完成对 IIS 的驱动，同样也可以根据脚本根据 IIS 代码完全匹配的生成接口文件

Configure: 随机 IIS 的工作模式，寄存器相应 bit 位，使 IIS 能工作在各个功能下，简单代码如下：

```
class dut_cfg
typede     enum{LEF,STD,PCM0,PCM1} mode_type;
rand       mode_type      type_t;
rand       bit            tx_en;
rand       bit            rx_en;
 Constraint     calid{tx_en        inside{1};
                      rx_en        inside{1};
                     }
endclass
```

Gen: 用工厂模式生产我们预先在 Packet 中定义的事务进行传输，事务的定义简单代码如下：

```
class Packet extends vmm_data;
    rand bit[31:0]   data;
    virtual function vmm_data allocate()
endfunction
virtual function vmm_data copy()
endfunction
endclass
`vmm_channel(Packet)
`vmm_atomic_gen(Packet, “Pa”)
```

Broadcast:将 Gen 产生的事务分别传输给驱动器，让驱动器对 IIS 进行驱动。简单代码如下：

```
vmm_broadcast     bcast;
bcast = new(“Broadcast”, “bcast”,gen.out_chan,,2);
bcast.new_output(drv_chan,0)
bcast.new_output(recv_chan,0)
```

Driver(Recerve):驱动器，得到产生器发来的事务，他们将分别驱动 IIS，并将数据接收，其中，Driver 模拟 ARM 操作，需要对 IIS 寄存器进行配置，对中断进行处理以及相应的处理，通过将 bcast 得到的数据并行写入发送寄存器，IIS 则按照 IIS 接口时序发送出去，而 Recerve 则模拟一个音频设备，按照 IIS 接口时序对数据进行串行接收和发送，直至数据发送完毕，则停止工作。简单代码如下：

```
Class Driver extends vmm_xactor;
   function   new()   endfunction
   function
   cfg_dut();
   endfunction
   function
         send_data();
   endfunction
   function
         recv_data();
   endfunction
   function
         check_interrupt().
   endfunction
   function
         deal_interrupt();
   endfunction
endclass
```

Scoreboard: 将 Driver 和 Recerve 发送、接收的数据进行收集，通过编写比较函数，将收发数据进行比较，来检查 IIS 传输过程中是否正确。

Coverage: 在验证前，我们定义 IIS 必须进行某些功能的运行，比如工作模式、传输数据有效位宽、收发数据极性、时钟分频等等，当经过大量的随机运行之后，我们可以通过查看覆盖率来获取随机的过程中，IIS 运行了哪些功能，没有运行哪些配置，简单代码如下：

```
class coverage
    bit[1:0]   mode_ctrl;
    bit[4:0]   data_len;
    ……….
    covergrop i2s_mode_cov;
    coverpoint mode_ctrl{
            bins mode_ctrl_0 = {0};
            bins mode_ctrl_1 = {2};
                        }
    coverpoint data_len{
            bins data_lin_0 = {1};
            bins data_lin_1 = {3};
            bins data_lin_2 = {8};
            bins data_lin_3 = {15};
                        }
endclass
```

3.3IIS 验证波形以及覆盖结果

图 3 为经过大量随机之后 IIS 各个块的代码覆盖结果图，图中表明了行覆盖、列覆盖以及状态机跳转覆盖，通过覆盖率的统计，我们可以对随机范围再次进行约束，或者对边角进行直接测试来提高覆盖率，以达到 IIS 各个功能完全检查。

图 4 为在进行验证的时候，IIS 四线状态，其中 din 为外部根据 IIS 时序将数据送入 IIS，dout 为 ARM 将数据写入 IIS 发送寄存器后，根据 IIS 时序输出的数字信号，LRCLK 以及 BCLK 是 IIS 发出的左右选通时钟以及位率时钟，从图中可以看到，配置中的随机完全可以做到数据收发极性、数据有效位宽、时钟分频、工作模式等各种情况交织出现，从而灵活的达到 IIS 功能的检查。

SCORE	LINE	COND	FSM	
83.40	92.67	82.53	75.00	dut
SCORE	LINE	COND	FSM	
80.00	97.50	62.50		i2s_clkgen
78.42	90.16	66.67		i2s_mem
SCORE	LINE	COND	FSM	
78.33	90.00	66.67		i2s_fifo_rx
SCORE	LINE	COND	FSM	
83.33	100.00	66.67		i2s_fifomem
81.25	81.25			i2s_rptr_empty
100.00	100.00			i2s_sync_r2w
100.00	100.00			i2s_sync_w2r
80.00	80.00			i2s_wptr_full
SCORE	LINE	COND	FSM	
78.49	90.32	66.67		i2s_fifo_tx
SCORE	LINE	COND	FSM	
83.33	100.00	66.67		i2s_fifomem
100.00	100.00			i2s_sync_r2w
100.00	100.00			i2s_sync_w2r
86.67	86.67			i2s_txrptr_empty
77.78	77.78			i2s_txwptr_full
SCORE	LINE	COND	FSM	
85.06	84.42	85.71		i2s_reginf
84.61	95.49	83.33	75.00	i2s_txrx
SCORE	LINE	COND	FSM	
84.02	94.97	82.09	75.00	i2s_main_ctrl
95.62	97.92	93.33		i2s_receiver
83.82	94.92	72.73		i2s_transmitter

Figure 3.the result of code coverage
图 3：随机之后代码覆盖统计图

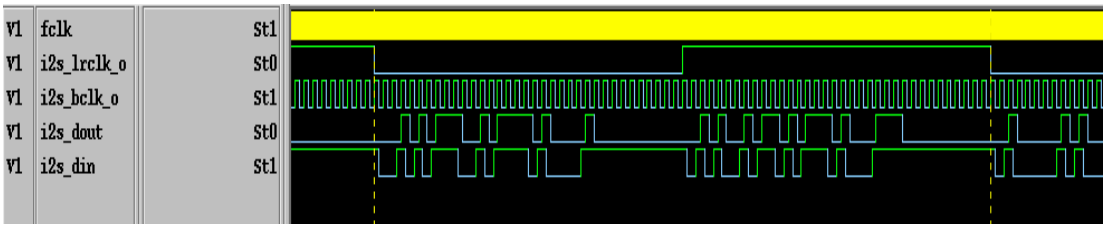

Figure 4.the input and output of digital signal
图 4 IIS 数字信号输入输出

Figure5.the input and output of digital signal

图 5 经过 IIS 模数、数模转换

图 5 为 IIS 外接数模模式转换器（CODEC）进行的实际情况模拟仿真，绿色曲线表示 CODEC 输入的模拟信号，当 CODEC 进行模数转换之后，数字信号将发送至 IIS 接受，IIS 做 LOOP 循环之后将数据再次发送出来送达 CODEC 的数字信号输入处，则红色曲线则为 CODEC 的模拟信号输出，从图中可以看到 IIS 在实际情况下工作正常。

4 总结

978-1-935068-33-4 © 2010 SciRes.

基于 System verilog 语言的 VMM 随机验证方法，依据它的继承性以及随机性明显提高验证效率，它采用覆盖来驱动激励的随机产生完全做到了功能验证的完备性，VMM 随机验证平台具有很强的维护性、灵活性，以及可重用性，完全改善了传统的直接测试模式。

致谢:

在整个项目和论文编写中，我的导师郑建宏老师以及重邮信科给了很大的帮助和支持，感谢郑建宏老师和重邮信科 ASIC 部门的同事。

References (参考文献)

[1] Chris, Speer was, Zhang, Mai Song Ping Translation System Verilog verification using VMM Methodology Science Press, 2006
克里斯、斯皮尔著，张春、麦宋平译 System Verilog verification using VMM Methodology 科学出版社 2006.
[2] Synopsys System Verilog VMM Lab Guide version 40-I-052-SSG-005 2006
[3] Alon Glusk. Coverage_oriented verification of banias.
[4] System verilog 3. la language reference manual. Acceuera's Extensions to verilog. Acceuera, Napa, California, 2004
[5] A. Benso, A. Bosio, S. Di carb etc. A Functional verification based Fault Injection Environment [C]. 22nd IEEE International Symposium on Defect and Fault to lerance in VL SI Systems ,2007, 114_122.