纯 SV 语言搭建验证平台

张 静,卜 刚
(南京航空航天大学 电子信息工程学院,江苏 南京 211106)

摘 要:面对日益复杂的芯片系统设计和 IP 的高度集成方式,验证的重要性日益增加。传统的验证主要依赖于直接测试,虽然直接测试平台也可以采用有限的随机方式,但是通常是通过产生随机数的方式来实现的,而不是在每个数据单元简单地写入预先设定的值。直接测试方法适合于小设计,但一个典型 SoC 设计需要上千个测试用例,耗时太长。因此提升验证产量的唯一方法是减少产生测试所消耗时间。基于 SystemVerilog 具有丰富语言能力、能描述复杂验证环境、产生带约束的随机激励、面向对象编程、功能覆盖率统计等诸多优点,因此可以采用 SystemVerilog 语言功能构建一个验证平台。搭建验证环境时,可以应用带约束随机激励产生方法以及覆盖率驱动来提高验证效率,缩短验证周期,平台在 queastasim 上进行了仿真验证,并取得了比较好的结果。

关键词:SystemVerilog;SoC;随机激励;功能覆盖率;验证
中图分类号:TP31 **文献标识码**:A **文章编号**:1673-629X(2020)04-0052-05
doi:10.3969/j.issn.1673-629X.2020.04.010

Building of Verification Platform Using Pure SV Language

ZHANG Jing,BU Gang
(School of Electronic Information Engineering,Nanjing University of Aeronautics and Astronautics,
Nanjing 211106,China)

Abstract:In the face of increasingly complex chip system design and a highly integrated approach to IP, verification is increasingly important. Traditional verification relies mainly on direct testing. Although direct test platforms can also use a limited random approach, they are usually implemented by generating random numbers instead of simply writing pre-set values in each data unit. Direct test methods are suitable for small designs, but a typical SoC design requires thousands of test cases and takes too long. Therefore, the only way to increase verification yield is to reduce the time it takes to generate a test. Based on the advantages of SystemVerilog, such as rich language ability, describing complex verification environment, generating constrained random excitation, object-oriented programming, function coverage statistics and so on, therefore, a verification platform can be built by the SystemVerilog language function. When constructing the verification environment, the constrained random excitation generation method and coverage drive can be applied to improve the verification efficiency and shorten the verification period. The platform is simulated and verified on queastasim, and ideal results are obtained.

Key words:SystemVerilog;SoC;random excitation;functional coverage;verification

0 引 言

芯片的验证[1-7],是提高芯片流片成功的关键。验证工作与设计仿真工作不同,仿真是为了证明设计方案的正确性,而验证是为了证明方案中不存在错误。设计错误很容易造成芯片完全不能工作,而修正错误再重新流片不但需要额外的费用,更会延迟芯片上市时间,这对芯片开发造成了巨大的风险。

现如今,芯片制造工艺更加精细,芯片制造费用的增加,芯片功能变得越来越复杂,验证的重要性也在不断增加。

目前来说,SystemVerilog[3,6-11]已经成为比较主流的验证语言,SystemVerilog 语言具有许多优点,诸如随机约束[12]、功能覆盖率[2,13]、断言技术和利用面向对象思想构建验证平台的一般方法。这些优点能极大提高芯片验证的效率,保障芯片设计的正确率,缩短芯片上市时间。

收稿日期:2019-04-09 **修回日期**:2019-08-12 **网络出版时间**:2019-12-18
基金项目:江苏省自然科学基金(BK2012792)
作者简介:张 静(1995-),女,硕士研究生,通信作者,研究方向为芯片验证;卜 刚,教授,研究方向为集成电路设计。
网络出版地址:http://kns.cnki.net/kcms/detail/61.1450.TP.20191218.1110.020.html

1　SystemVerilog 语言

此处介绍的验证技术,专注于受约束的随机测试,它利用功能覆盖率来衡量进度并指导验证。SystemVerilog 最有意义的优点在于,它允许用户在多个项目中使用连续一贯的语法来构造可靠并且可重复的验证环境。SystemVerilog 提供了一种建模语言,简化了自顶向下的设计,可以在 SystemVerilog 中创建模型,然后在下一层重新定义每个模块,初始的系统级模型也可以被重新用作参考模型,不仅如此,它还引入了直接编程接口(DPI),能更加简单地连接 C、C++或者其他非 Verilog 编程语言。

相比于硬件描述语言(HDL),SystemVerilog 硬件验证语言(hardware verification language,HVL)[5,14] 具有典型的性质:

●受约束的随机激励生成;

●功能覆盖率;

●更高层次的结构,尤其是面向对象编程;

●多线程及线程的通信;

●支持 HDL 数据类型,例如 Verilog 的四状态数值;

●集成了事件仿真器,便于对设计施加控制。

还有其他很多有用的性质,但上述特性已经能够支撑创建高度抽象的测试平台。

采用随机测试[14] 的原因如图 1 所示。

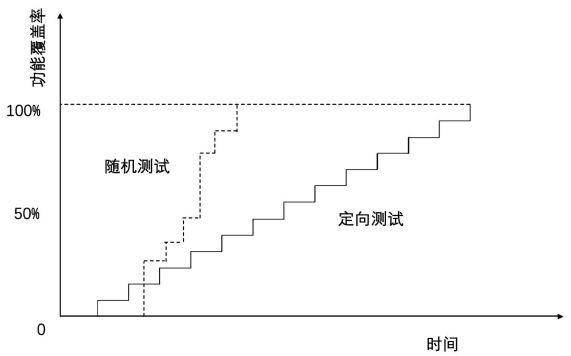

图 1　随机测试与定向测试的时间比较

图 1 是传统验证的定向测试方法和高级验证的随机测试方法的比较[15],对这两种方法所用的验证周期以及功能覆盖率进行对比。采用定向测试方法时,使用的时间与功能覆盖率呈线性关系。随着增加定向测试例的编写数量,功能覆盖率稳步上升。采用随机测试方法时,由于需要准备随机验证环境,开始时会有一段覆盖率没有增加的时间段。当随机验证环境准备完毕,运行随机测试例,可以随机产生大量激励,功能覆盖率会有大幅增加。当覆盖率增加不明显时,分析未覆盖情况,加紧约束或添加定向测试例,对未覆盖的情况进行覆盖,直至功能覆盖率达到 100%。由图可以看出,虽然采用随机测试的方法会有一段覆盖率为零的时间段,但从整个验证周期来看,随机测试方法会比定向测试方法更具优势,尤其是随着集成电路规模和复杂度的增加,这种基于覆盖率驱动的受约束随机激励方法会更符合验证的需要。

2　DUT 介绍

图 2 中的 DUT 是由两个子模块构成的具有两级流水的系统模块[16],第一级为一个预处理器模块(EXECUTE PREPROCESSOR),第二级为一个算术逻辑运算单元(ALU)模块:

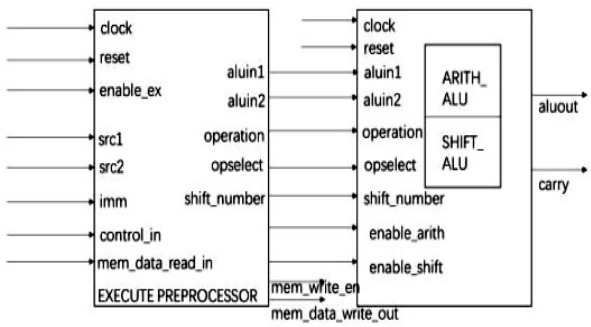

图 2　DUT 结构

●预处理器:对外部输入的数据和控制信号进行处理,并产生对应的 ALU 可接受的数据和控制信号。

●算数逻辑运算单元:根据输入的 opselect,operation 和 shift_number 信号执行相关算数逻辑运算。

这两个子模块都可以具有各自独立的时钟和复位信号,文中这两个子模块的时钟和复位信号是一致的。

基本功能描述:

●复位:复位时,所有的时序逻辑输出都为 0,这些信号包括:预处理输出信号中的 aluin1,aluin2,operation_out,opselect_out,shift_number enable_arith,enable_shift 以及 ALU 输出的信号 aluout,carry。

●预处理器功能描述:所有时序逻辑的输出只有在 enable_ex = = 1 的条件下才会发生变化。

●ALU:所有指令信号只在以下两个条件成立之一才有效:enable_arith = = 1 or enable_shift = 1。这些信号均在时钟的上升沿处进入 ALU 子模块,ALU 可执行移位操作和算数逻辑运算,以及数据传输。

3　验证架构

SystemVerilog 平台如图 3 所示。

组件搭建描述[15]:

3.1　Packet 类

将数据信息封装进入 Packet 类,利用随机化和相关约束产生随机数据,创建两个 packet 对象,一个包在

DUT 输入端输入,另一个包和 DUT 输出的数据相参照。

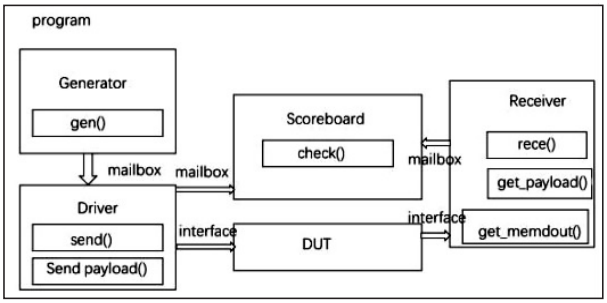

图 3 SystemVerilog 平台

随机化的部分代码:

```
rand reg[`REGISTER_WIDTH-1:0] src1;
rand reg[`REGISTER_WIDTH-1:0] src2;
rand reg[`REGISTER_WIDTH-1:0] imm;
……
```

相关约束的部分代码:

```
constraint Limit {
src1 inside{[0:65534]}
src2 inside{[0:65534]};
imm inside{[0:65534]};
mem_data inside{[0:65534]};
opselect_gen inside {[0:1],[4:5]};
```

3.2 Generate 类

封装了 gen()任务和一个 start()方法,其中 start()方法的循环由系统全局变量 number_packets 控制。如果 number_packets≤0,则这个循环是无限的。如果 number_packets>0,则这个循环在循环这么多次后会停止。在调用 gen()之后,会创建一个随机化 packet 对象(pkt2send)的拷贝,并且通过 out_box 邮箱发送到 Driver。

3.3 Interface

连接验证平台和 DUT 的端口。但是在类里面不可直接定义 interface,须使用 virtual 来定义。

3.4 Driver 类

其中 in_box 将会用来从 Generator 向 Driver 接收 Packet 对象。out_box 将用来从 Driver 向 Scoreboard 发送 Packet 对象。in_box 和 out_box 都是 pkt_mbox 的类型。在 Driver 中 start()方法是在一个无限的循环中执行的。在这个循环的每一次执行中,都会从 in_box 中获取一个 Packet 对象。这个数据包对象的内容将会通过 send()发送经过 DUT。一旦这个 Packet 对象的发送过程完成后,这个 Packet 对象会被发送到 scoreboard。

Generator 到 Driver 的 mailbox 的代码:

```
typedef mailbox#(Packet) in_box_type;
in_box_type in_box=new;
```

Driver 到 Scoreboard 的 mailbox 的代码:

```
typedef mailbox#(Packet) out_box_type;
out_box_type out_box=new;
```

3.5 Receiver 类

调用 recv()来从 DUT 中重新获取 Packet 对象,将从 DUT 中获取的 Packet 对象复制一份到 out_box 中去,在 receiver 中有 start()方法,它将会执行一个非阻塞的无限循环。每次循环中,都会从 DUT 中重新构建一个 Packet 对象。一旦重新获取后,这个 Packet 对象将会通过发出邮箱(out mailbox)发送到 scoreboard。

Receiver 连接到 Scoreboard 的 mailbox 的代码:

```
typedef mailbox#(Packet) rx_box_type;
rx_box_type rx_out_box;
```

3.6 Scoreboard 类

封装 check()任务,实现 Scoreboard,Driver 和 Receiver 之间的通信。Driver 在发送数据包对象到 DUT 中时,也会将它发送到 driver_mbox 邮箱。一个 Receiver 在从 DUT 接收到数据时,也会发送 Packet 对象到 receiver_mbox 邮箱中去。

利用 driver_mbox. num()和 receiver_mbox. num()查看是否有包传入,如果存在利用 get()函数将从 Driver 传入的包放入 pkt2send,receiver 传入的包放入 pkt2cmp,之后利用 check()函数作比较。在 Scoreboard 这个类中还定义了 coveragegroup 来收集覆盖率,定义 DUT 的各个功能点比如 ALU 的算术和移位功能,采用交叉覆盖率,用 sample()采样,最后用 get_coverage()收集覆盖率。

Scoreboard 用来连接 Driver 的 mailbox 的代码:

```
typedef mailbox#(Packet) out_box_type;
out_box_type driver_mbox;
```

Scoreboard 用来连接 Receiver 的 mailbox 的代码:

```
typedef mailbox #(OutputPacket) rx_box_type;
rx_box_typereceiver_mbox;
```

覆盖率功能点定义部分代码:

```
covergroup Arith_Cov_Ver2;
coverpoint pkt_sent. imm;
src1_cov:coverpoint pkt_sent. src1 ;
……
opselect_cov2:coverpoint pkt_sent. opselect_gen {
bins shift={0};
bins arith={1};
bins mem={[4:5]};
}
……
cross src1_cov, src2_cov;
endgroup
```

covergroup 的例化与采样:

```
Arith_Cov_Ver1=new();
```

……

Arith_Cov_Ver1. sample( );

……

覆盖率的收集:

coverage_value1 = Arith_Cov_Ver1. get_coverage( );

……

3.7　Test 类

声明并例化各组件,正确连接邮箱,调用每个类的 start( )方法,设置 reset( )任务。

部分代码:

组件的声明:

Driver drv;

……

组件的例化:

drv = new( );

……

组件的启动:

drv. start( );

……

3.8　Top 模块

声明接口,实现 DUT 和平台互连,设置时钟信号。

部分代码:

平台互连:

Top dut(. clock(top_if. clock), . enable_ex(top_if. enable_ex),……);

设置时钟信号:

#(simulation_cycle/2)

Clock = ~Clock;

4　仿真结果

编写完 SystemVerilog 代码后开始进行平台验证,主要采用 questamsim 软件进行仿真验证,使用命令启动平台、生成打印信息报告以及收集覆盖率报告。

根据输出的波形图和生成的验证报告来比较 DUT 功能的正确与否,并查看输出的功能覆盖率报告来检查各功能点的实现状况。

```
ALUIN1: DUT = 00004440   & Golden Model = 00004440

ALUIN2: DUT = 00003a1e   & Golden Model = 00003a1e

ENABLE_ARITH: DUT = 0    & Golden Model = 0

ENABLE_SHIFT: DUT = 0    & Golden Model = 0

OPERATION: DUT = 7    & Golden Model = 7

OPSELECT: DUT = 1    & Golden Model = 1

SHIFT_NUMBER: DUT = 00    & Golden Model = 00

:RAGE] Coverage Result for cover 1 At present =          58
:RAGE] Coverage Result for cover 2 At present =          57
:RAGE] Coverage Result for cover 3 At present =          51
:RAGE] Coverage Result for cover 4 At present =          37
:RAGE] Coverage Result for cover 5 At present =          38
:RAGE] Coverage Result for cover 6 At present =          38
```

图 4　信息打印

由图 4 可以发现 DUT 的输出和模型的输出是一致的,DUT 的设计满足功能实现,图 4 还打印出各个功能点的覆盖率状况以便于分析如何添加测试用例来使功能覆盖率达到 100%。

波形图中定位处由随机输入的信号来看实现的是算数逻辑,即将 aluin2 的低十六位赋值给 aluout 的高十六位,并将 aluout 的低十六位置零。由图中 aluin2 为 32'h00003a1e,aluout 为 32'h00000000 可得运算是正确的,且各使能信号也正确无误。

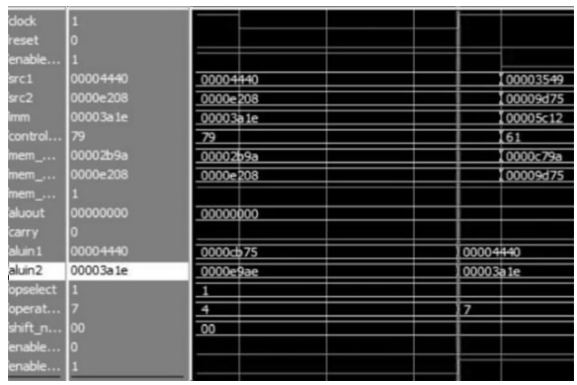

图 5　DUT 部分波形(1)

图 5 检测了 DUT 的算数逻辑,只是 DUT 其中一个功能点。图 6 列出了 DUT 基于随机激励测试的所有功能点实现的波形图,查看整体波形,检查每个功能点实现的具体情况,可以发现 DUT 的设计是没问题的。

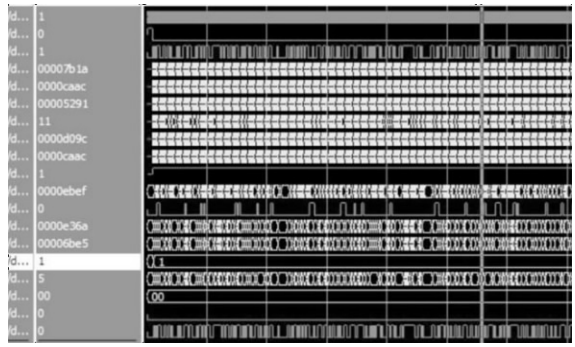

图 6　DUT 部分波形(2)

图 7 描述了此 DUT 的功能覆盖率,可见覆盖率还未达到 100%,尤其是算数和移位运算单元比较低,因此可以修改约束条件增加测试用例来使功能覆盖率提高,从而保证验证的完备性。

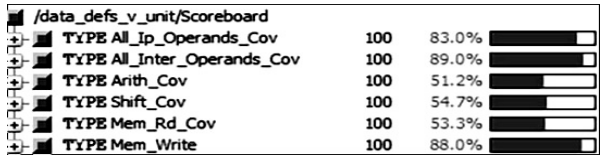

图 7　功能覆盖率(1)

由图 8 可知,通过增加测试用例以及修改相关约束条件,可以发现覆盖率有了明显的提高,所有的覆盖率都达到了 100%,说明该测试点已经得到了充分的

验证。由此可见,增加定向测试激励配合随机激励可以大幅提高功能覆盖率,由图 8 可见所有的测试点的覆盖率都达到了 100% 。因此可得,基于 SV 语言的优点,可以减少验证时间并提高验证效率。

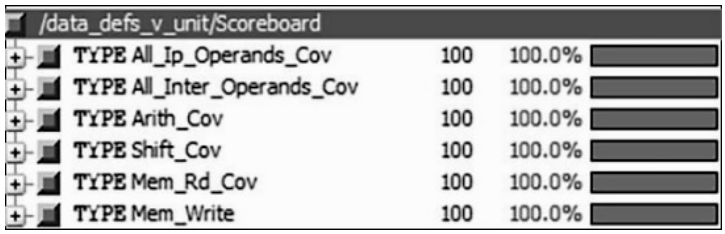

图 8 功能覆盖率(2)

5 结束语

定向测试每次只能测试一个特性,而无法模拟设备在实际应用中所面对的复杂的激励和配置,为了得到稳健的设计,使用受约束的随机激励加上功能覆盖率,才能在最大限度内创建出最广泛的激励。文中演示了如何使用 SV 语言面向对象编程的方法建立一个由覆盖率驱动并且受约束的随机分层测试平台。基于 SV 的验证,充分利用了 SV 语言中提供的随机约束和断言机制,改变了传统的直接测试的方法,基于覆盖率驱动的验证极大提高了验证的效率和完备性,有效缩短了验证周期。

参考文献:

[1] LAM W K. Hardware design verification:simulation and formal method-based approaches (prentice hall modern semiconductor design series)[M]. [s. l.]:Prentice Hall PTR, 2005.

[2] BERGERON J. Writing testbenches:functional verification of HDL models[M]. [s. l.]: Springer Science & Business Media,2012.

[3] 闫 沫,张 媛. 基于 System Verilog 语言的设计验证技术[J]. 现代电子技术,2008,31(6):8-11.

[4] 吴英攀,于立新,薛 可,等. 基于层次化验证平台的存储器控制器功能验证[J]. 微电子学与计算机,2009,26(2):25-28.

[5] 克里斯·斯皮尔. SystemVerilog 验证[M]. 北京:科学出版社,2009.

[6] BERGERON J, CERNY E, HUNTER A, et al. Verification methodology manual for SystemVerilog[M]. [s. l.]: Springer Science & Business Media,2006.

[7] SPEAR C. SystemVerilog for verification:a guide to learning the testbench language features[M]. [s. l.]: Springer Science & Business Media,2008.

[8] BERGERON J, CERNY E, HUNTER A. SystemVerilog 验证方法学[M]. 北京:北京航空航天大学出版社,2007.

[9] 程 刚,蔡 敏. 基于 SystemVerilog 的 SoC 功能验证方法研究[J]. 科学技术与工程,2009,9(22):6814-6818.

[10] 胥 林,丁婷婷. 应用 SystemVerilog 搭建 USB 验证平台[J]. 黑龙江科技信息,2008(17):72.

[11] SUTHERLAND S, DAVIDMANN S, FLAKE P. SystemVerilog for design second edition:a guide to using SystemVerilog for hardware design and modeling[M]. [s. l.]: Springer Science & Business Media,2006.

[12] 杨 鑫,徐伟俊,陈先勇,等. SystemVerilog 中的随机化激励[J]. 中国集成电路,2007,16(10):37-41.

[13] 徐伟俊,杨 鑫,陈先勇,等. 针对功能覆盖率的验证过程[J]. 中国集成电路,2007,16(7):58-62.

[14] 刘 萌,冯海洲,李 康,等. 基于 SystemVerilog 的网络处理器验证平台设计[J]. 电子器件,2011,34(3):320-333.

[15] 克里斯·斯皮尔. SystemVerilog 验证:测试平台编写指南[M]. 张 春,麦宋平,赵益新,译. 北京:科学出版社,2009.

[16] 阎 石. 数字电子技术基础[M]. 北京:高等教育出版社,2005.