基于 UVM 的软硬件协同验证平台设计

李姝萱,卜 刚,韩宇昕
(南京航空航天大学 电子信息工程学院,江苏 南京 211106)

摘 要:随着芯片的规模和复杂度日益增大,软件环境下的功能验证越来越无法满足高效率的芯片生产流程的需求。因此,验证效率的提高变得十分关键。针对 RFID 数字基带系统中标签——阅读器链路的 FM0 和 Miller 编码模块,利用 FPGA 硬件平台的高速性能和面向对象编程的优势,搭建了一种基于覆盖率驱动的 UVM 软硬件协同验证平台。FPGA 端将集成有 RS-232 串口收发模块的可综合待测设计下载到硬件上,PC 端采用 winsock API 编写数据上行和下行通路的 C 程序。验证平台仍保留在仿真软件 QuestaSim 中运行,并通过 DPI 接口调用的方式与硬件平台进行通信。最终,上述方案在 Altera 开发板实现。实验结果表明,该验证平台的功能覆盖率达到 100%,能够有效提高验证效率并且能够为大规模 SoC 的验证所用,同时还具有硬件资源占用率低以及可维护性和可复用性强的优点。

关键词:通用验证方法学;软硬件协同验证;串口;现场可编程门阵列;覆盖率驱动
中图分类号:TP33 **文献标识码**:A **文章编号**:1673-629X(2022)08-0076-06
doi:10.3969/j.issn.1673-629X.2022.08.013

Design of UVM-based Software and Hardware Co-simulation Verification Platform

LI Shu-xuan, BU Gang, HAN Yu-xin
(School of Electronic Information Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing 211106, China)

Abstract: With the increasing size and complexity of chips, functional verification in a software environment is increasingly unable to meet the needs of high-efficiency chip production processes. Therefore, the improvement of verification efficiency becomes critical. Aiming at the FM0 and Miller encoding modules of the tag-reader link in the RFID (Radio Frequency Identification) digital baseband system, a coverage-driven hardware and software co-verification platform based on UVM (Universal Verification Methodology) is implemented by utilizing the high-speed performance of the FPGA hardware and the advantages of OOP (Object Oriented Programming). The design to be verified integrated with RS-232 serial port transceiver module is downloaded to the FPGA hardware, while the PC terminal writes the C programs for the data uplink and downlink channels with winsock API. The verification platform still runs in the simulation software QuestaSim and communicates with the hardware with the DPI interface. Finally, the above-mentioned project implemented with Altera development board. Experimental results show that the functional coverage of the verification platform reached 100%, which can effectively improve the simulation efficiency and can be reusable for the verification of large-scale SoC. It also has the advantages of low hardware resource occupancy, strong maintainability and reusability.

Key words: universal verification methodology (UVM); hardware and software co-verification; serial port; field-programmable gate array (FPGA); coverage-driven

0 引 言

随着大规模集成电路设计和半导体工艺的快速发展,芯片研究周期不断缩短,当前的软件仿真速度日渐难以承受千万门级功能复杂的 SoC 设计。传统的系统设计及验证方法亟待改进,而软硬件协同仿真以及验证的概念由此应运而生。为了提高验证效率,该文利用 FPGA 硬件加速的思想,在 UVM 验证方法学的基础上提出了一种基于 FPGA 的软硬件协同验证方案。

该文旨在将验证平台与待测设计进行明确的软件和硬件划分,将功能和计算复杂的待测设计下载到了 Altera DE2-115 硬件开发板中,UVM 验证平台的各组

收稿日期:2021-09-06 **修回日期**:2022-01-06
基金项目:南京航空航天大学研究生创新基地开放基金(kfjj20200413)
作者简介:李姝萱(1997-),女,硕士研究生,通信作者,研究方向为数字 IC 验证;卜 刚,教授,研究方向为数模混合集成电路设计。

件仍保留在 PC 端 QuestaSim 仿真器中运行。这样不仅可以利用 FPGA 大量的内部资源和高速性能以缩短硬件运行时间,还可以利用 UVM 的面向对象和可重用性高等优点获取更加可靠的验证结果[1]。

1　待测设计

国际标准 ISO/IEC18000-6 规定,RFID 数字基带系统中标签——阅读器链路需采用 FM0 和 Miller 编码,具体编码方式由阅读器发送给标签的 Query 命令中参数 m[1:0]的值决定。m[1:0]参数的取值范围共有 4 个值,当 m[1:0]为 2'b00 时,采用 FM0 编码方式;当 m[1:0]为 2'b01、2'b10、2'b11 时,则采用 Miller 编码方式[2]。

1.1　FM0 编码

根据 ISO/IEC18000-6 协议规定,FM0 编码原理是基于电平的翻转来实现的。当编码模块检测到有数据输入时,将会在一个编码周期内输出两位码元。在每个编码周期内,数据-0 和数据-1 进行电平翻转的次数不同。

数据-0 的编码在码元周期起始处和码元周期中间分别进行一次电平的翻转;而数据-1 的编码只在码元周期开始时进行一次电平的翻转。图 1 显示了 FM0 的编码规则以及状态转换。在编码结束后,需要依照协议的相关规则给出编码结束符,添加在编码数据流的末端。

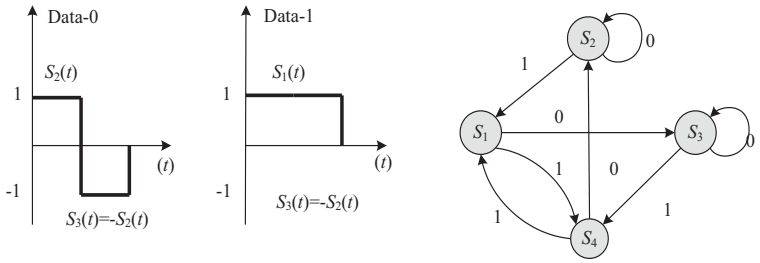

图 1　FM0 编码及状态跳转

1.2　Miller 编码

Miller 编码与 FM0 编码基本类似,同样也是基于电平翻转的原理来实现。其编码规则以及状态转换如图 2 所示。

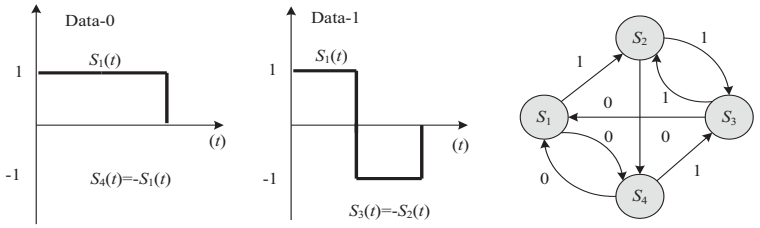

图 2　Miller 编码及状态跳转

相较于 FM0 编码,Miller 编码相对复杂一些。每一位数据经过 Miller 编码后,可能产生 4 位、8 位或 16 位码元,具体编码规则取决于参数 m[1:0]的值。参数 m[1:0]为 2'b01,2'b10,2'b11 所对应的 M 值分别为十进制的 2,4 和 8。M 值不同,每位码元编码后的数据长度也不同。M=2 时,码元长度为 4 位;M=4 时,码元长度位 8;M=8 时,码元长度位 16。由图 3 可见,三种编码方式下的 Miller 编码在两个连续的 0 之间均不发生电平翻转,在 1 和 0 之间以及两个连续的 1 之间则在数据边界处进行一次电平翻转。

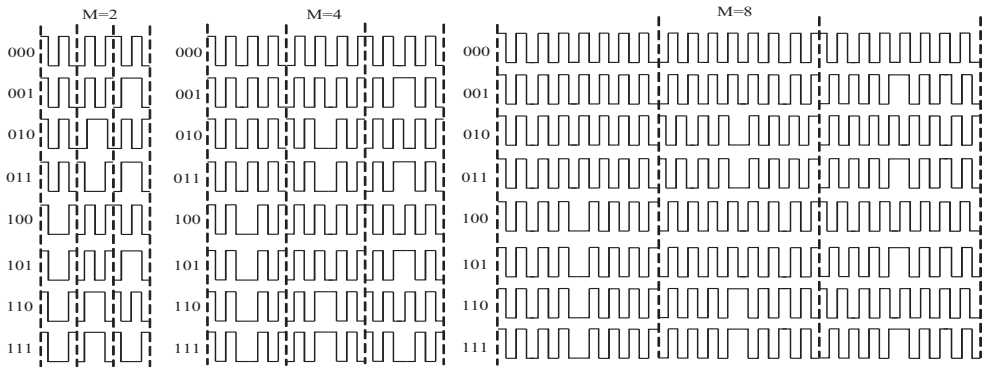

图 3　Miller 编码序列

与 FM0 编码相同,所有待编码数据进行 Miller 编码结束后也将按照协议相关规则在数据后面添加相应

的结束符标识。

2 软硬件协同验证平台

FPGA 作为硬件原型,其运行速度为真实的周期时间,而仿真器仅仅作为 PC 端一个软件仿真应用程序,其运行硬件模型的每一个周期都将需要耗费几百个真实的时钟周期,因此,对于同一个待测设计的验证,在做好软件和硬件模块划分以及数据交换链路的前提下,利用 FPGA 进行软硬件协同验证相较于仿真器下的软件验证具有较为明显的加速效果。

该文提出的基于 UVM 验证方法学的软硬件协同验证平台的基本原理是将待测设计综合、布局布线并使用 FPGA 开发软件 Quartus Ⅱ通过 JTAG 配置到 Altera DE2-115 硬件开发板上,UVM 验证平台仍保留在 PC 端仿真器 QuestaSim 软件中进行,FPGA 目标测试硬件平台利用其 RS-232 串口与 PC 主控机之间进行收发通信,PC 端使用 winsock API 进行串口编程,两者之间连接一根 9-pin 公转母电缆即可实现数据上下行通路,各自完成激励的下传和结果的上传,数据传输速率为 115 200 bps[3]。该平台架构如图 4 所示。

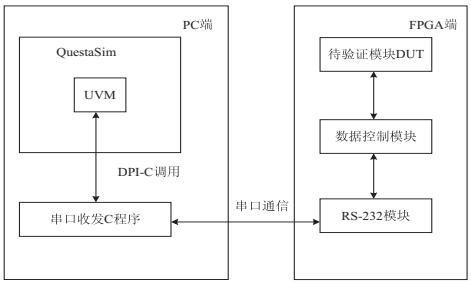

图 4 软硬件协同验证平台

2.1 UVM 验证方法学

UVM 是一个以 SystemVerilog 类库为主体搭建验证平台的验证方法学,最早由 Accellera 标准组织推出,并得到三大厂商(Cadence、Synopsys 和 Mentor)的一致支持,于 2014 年推出了最新版本 UVM 1.2。其之所以能够成为当今数字 IC 验证最为广泛使用的主流验证方法学,主要是因为其具备以下几点优势:

2.1.1 UVM 的优势

(1)UVM 验证平台兼具封装、继承的特点,又具备 SystemVerilog 编程语言的面向对象等特点,其验证组件具有很高的可重用性、可移植性以及可继承性,使得验证人员无需耗费大量精力在搭建验证平台上,可以将大部分精力专注于待测设计功能点的研究以及测试用例的编写。

(2)验证人员利用其可重用组件构建具有标准化层次结构和接口的功能验证环境[4],平台可维护性强。

(3)主流 EDA 工具支持,方便验证人员学习和调试。

(4)与传统的 Testbench 相比,UVM 验证平台实现了受约束的随机激励的产生,自动化的结果比对以及代码和功能覆盖率的收集[5]。

(5)开发了 factory、config、phase、override 等机制,可根据工程特性灵活地搭建验证结构,实现了平台的自动化和层次化运行,一定程度上减少了验证人员的调试成本。

(6)UVM 组件之间的通信采用事务级建模(Transaction Level Modeling,TLM)的方式,不同于 DUT 中各模块之间的比特级数据传输方式,UVM 验证组件之间的数据传输是以数据包为单位的。TLM 通信使得数据在验证组件的端口之间流通更为方便。

2.1.2 UVM 的结构和组件

图 5 为标准的 UVM 验证平台结构[6]。

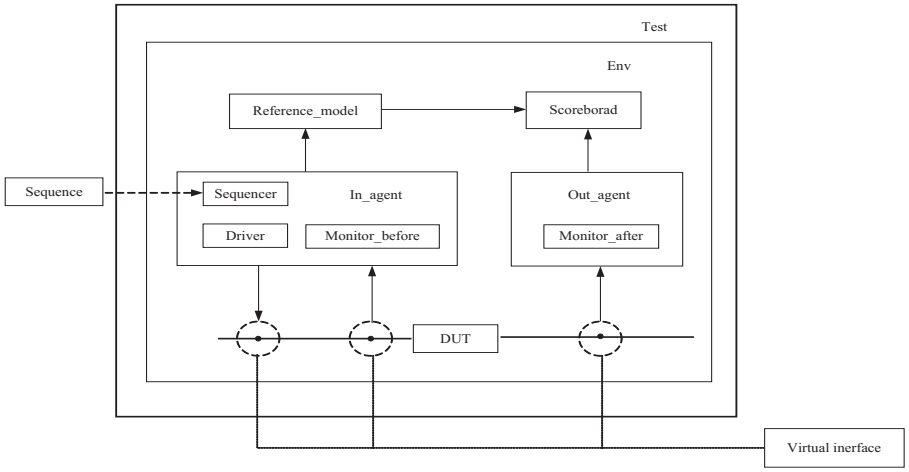

图 5 UVM 验证平台结构

UVM 验证平台是由 uvm_componet 和 uvm_object 组成的。uvm_componet 继承于 uvm_object 的同时,拥有 uvm_object 没有的一些特性。下面介绍几个重要组件的功能特性。

(1)test:派生于 uvm_test,是整个验证平台的入口,内部例化了 env,是 case 的父类。由于 DUT 的待

测功能各有不同,所以需要编写不同的 case 进行验证。case 与 sequence 一一对应,在每个 case 中利用 UVM 的 config 机制将其对应的 sequence 设置为 sequencer 的 default_sequence。

(2) transaction:继承于 uvm_sequence_item,在 UVM 验证平台中,各组件之间的通信是以数据包 transaction 为单位的[7]。定义两个 transaction 类型,分别为 transaction_in、transaction_out。前者为输入端数据包,后者为输出端数据包。driver 首先需要把数据从 transaction_in 中提出来,然后把其转换为 DUT 能够接收的 pin 级别数据类型,驱动到 DUT 的各个管脚。类似地,transaction_out 在 monitor 内部的转化机制也是如此,监测到 DUT 的输出管脚后将信息整合为数据包再送至 scoreboard 中与参考模型进行对比。

(3) sequence:继承于 uvm_sequence,通过随机化数据的方式产生数据包,并由 sequencer 传递给 driver,driver 则将数据包中定义的数据信息通过 virtual interface 配置给另一端的 DUT 管脚。验证过程中,sequence 利用 repeat( ) 和 uvm_do 等宏来产生大量不同的激励施加给 DUT,从而使功能覆盖率达到 100%[7]。

(4) sequencer:从 driver 的职能中分离出来的组件,继承于 uvm_sequencer,将 sequence 通过 seq_item_port 端口发送给 driver。

(5) monitor:继承于 uvm_monitor,分别创建 monitor_before 和 monitor_after 两个类,各用来监测 driver 施加给 DUT 的激励以及 DUT 的输出结果,另外还负责收集功能覆盖率,验证 FM0 和 Miler 编码的三种方式是否都得到验证以及待编码数据是否覆盖所有可能值[8]。其中,monitor_before 中的覆盖率组定义如下:

```
covergroup fm0_miller_in_cg;
data_cov:coverpoint tr. data[7:4]{
bins value={[0:7]};
}
m_cov: coverpoint tr. data[1:0]{
bins m0={2'b00};
bins m2={2'b01};
bins m4={2'b10};
bins m8={2'b11};
  }
endgroup
```

(6) reference_model:由 uvm_blocking_get_port 端口从 monitor_before 中获取数据,并由 uvm_analysis_port 端口将参考模型结果传递到 scoreboard 中进行比对[9]。本设计采用 C 语言编写参考模型,SystemVerilog 通过 DPI-C 接口调用该模型。对于 FM0 编码,首先依次读入待编码数据,设置一个变量 data_tmp,初始值为 1。若对数据-0 编码,则依次输出码元 ~data_tmp,data_tmp,data_tmp 值保持不变;若对数据-1 编码,则依次输出码元 ~data_tmp 和 ~data_tmp,data_tmp 值进行一次翻转。以这样的规则依次对待编码数据的每一位进行编码,对于 Miller 编码,data_tmp 初值为 0,结合 m 值,不断更新 data_tmp 的值来输出编码码流。

(7) scoreboard:继承于 uvm_scoreboard,在 main_phase 中使用 fork-join 块开启两个进程,其中一个用于从 reference_model 中获取数据包,另外一个用于从 monitor_after 中获得数据包,调用 compare 函数将两者进行对比,若一致则对比成功,DUT 功能正确,若出现对比失败,则代表 DUT 和参考模型输出不一致,存在漏洞。

(8) env:继承于 uvm_env,其中分别例化了 reference_model、scoreboard 和 agent 组件,并在 connect_phase 中使用 fifo 实现以上组件之间的互连。连接部分代码如下:

```
agt. i_ap. connect( agt_mdl_fifo. analysis_export) ;
model. port. connect( agt_mdl_fifo. blocking_get_ export) ;
agt. o_ap. connect( agt_scb_fifo. analysis_export) ;
scb. from_dut. connect( agt_scb_fifo. blocking_get_export) ;
model. ap. connect( mdl_scb_fifo. analysis_export) ;
scb. from_rfm. connect( mdl_scb_fifo. blocking_get_export) 。
```

2.2　数据通路

软硬件协同验证平台中,QuestaSim 仿真器中运行的 UVM 平台要与硬件中运行的待测设计进行数据交互,该文采用的方法就是利用仿真器提供的 DPI (Direct Programming Interface) 接口来实现[10]。通过 DPI,SystemVerilog 能够间接地连接 C、C++等编程语言。使用 import 语句声明并导入一个 C 程序,并将 DPI 接口 C 语言端的参数在 SystemVerilog 中对应一致,例如 SV 的 string 类型,C 使用 char * 类型。这样,C 程序就可以作为 SystemVerilog 的子程序一样被调用[11]。

winsock 是一个基于 socket 模型的 API,在 Windows 操作系统类中使用。本设计使用 winsock API 编写带有返回值的 C 函数,该函数在对串口进行一系列必要的参数配置之后,调用系统函数 WriteFile( ) 和 ReadFile( ) 来对串口进行写入和读取数据[12],其主要功能有:

(1) 获取仿真器中 UVM 验证平台所产生的随机测试激励,在对其进行数据分析之后,将其发送至串口端,并由串口下传至 FPGA 硬件平台。

(2) 每一次 PC 端向 FPGA 端发送激励,待 DUT

运行完成一次之后,串口获取 FPGA 端的回传结果,进行数据分析、打包等处理后发送至仿真器,验证平台将其和参考模型进行结果对比。

(3)实现 PC 端的时序控制,确保验证流程的有序进行[13]。

仿真器端与外部 C 函数连接部分核心代码实现如下:

```
import "DPI-C" function longintserial_port(input int data_in,
longint data_str);
……
task main_phase(uvm_phase phase);
……
data_pin=send_receive(tr5.data,data_str1);
tr_out.data_out=data_pin[63:0]。
```

软硬件协同验证平台中,FPGA 硬件平台主要完成的工作有串口收发模块、数据控制模块以及顶层模块的设计。

其中串口模块包括波特率产生、发送模块和接收模块。本设计采用的波特率为 115 200 bps,即串口所能达到的最大传输速率。接收模块将来自串口的数据流通过数据控制模块解析出 DUT 管脚对应的各项参数并将其配置给 DUT。若干周期后,DUT 输出 FM0 或 Miller 编码码流以及完成标识,发送模块一旦监测到完成标识从 0 变成 1,立即将编码结果上传至 PC 端。该过程通过循环算法实现了串口的多字节发送。顶层模块例化了串口收发模块、数据控制模块以及 FM0 和 Miller 编码模块,并协调各个模块之间的工作有序进行。

整个待测设计的工程在 Quartus Ⅱ中完成综合、布局布线、时序约束以及编译并且通过 JTAG 下载到 DE2-115 开发板中,至此,完成了软硬件协同验证平台硬件部分的设计。

3 验证结果

3.1 验证过程

当验证流程开始启动时,首先由 PC 端仿真器开始仿真,sequence 组件中产生的测试激励通过 seq_item_port 端口传递给 driver,由 driver 通过 uvm_analysis_port 端口发送给 monitor_before 和 monitor_after[14]。monitor_before 将激励传递给参考模型 C_model,在 monitor_after 中开始对 FPGA 端的 DUT 单元进行访问,仿真器通过 winsock 将激励下传给 FPGA,经过数据控制模块的解析后配置给 DUT 管脚。经过若干周期,DUT 编码结束,FPGA 端将输出结果进行分析整合并回传到仿真器中,仿真器继续下一阶段的仿真。此时,由于 FPGA 开发板的高速性能,DUT 的回传结果比参考模型 C_model 的结果先到达于 scoreboard,所以将 DUT 的结果 get_actual 数据包压入队列 actual_queue 中,待 C_model 将编码结果 get_expect 数据包传给 scoreboard 后,将队列中的 get_actual 弹出,调用 compare 函数将两者进行对比,输出对比结果[15-17]。至此,一次完整的软硬件协同验证完成。

3.2 验证结果

图 6 为仿真软件在一个验证周期内的打印信息,如图所示,待编码数据 data 为 4,m 值为 2,对应的 M 值为 4,则依照图 2 和图 3 的规则进行 Miller 编码,理论输出值的二进制码流应该为 1010_0101_0101_0101_1010_1010_1010_0101,对应的十六进制值为 0xa555aaa5。图中 data_str 为 monitor_after 监测到的 DUT 的编码结果,data_out 为 C_model 的编码结果,经 scoreboard 对比,二者的输出一致,scoreboard 输出 Compare SUCCESSFULLY。底部分别打印 DUT 和 C_model 编码数据和编码长度信息。

由于该文采用了软硬件协同验证的方法,DUT 在 FPGA 上运行,所以 PC 端仿真软件无法收集代码覆盖率。在搭建验证平台之前,首先要针对待测设计制定一套完整的验证计划,要求在可控的时间范围内完成高质量的验证工作。其中包括确定端口、提取功能点、编写受约束的随机激励、收集覆盖率以及模拟故障等。

```
# UVM_INFO miller_driver.sv(32) @ 1290: uvm_test_top.fm0_miller_env.agt.drv [driver send] data=4
# UVM_INFO miller_driver.sv(33) @ 1290: uvm_test_top.fm0_miller_env.agt.drv [driver send] m=2
# read actual length:8
# data from dut=0x00000000a555aaa5
# UVM_INFO monitor_before.sv(51) @ 1290: uvm_test_top.fm0_miller_env.agt.mon_bef [mon_bef send] tr.data=0x4
# UVM_INFO monitor_before.sv(52) @ 1290: uvm_test_top.fm0_miller_env.agt.mon_bef [mon_bef send] tr.m=0x2
# miller_Result: 1 0 1 0 0 1 0 1 0 1 0 1 0 1 0 1 1 0 1 0 1 0 1 0 1 0 0 1 0 1
# miller_len:32
# m:2
#
# data_out:0xa555aaa5
# UVM_INFO scoreboard.sv(41) @ 1290: uvm_test_top.fm0_miller_env.scb [my_scoreboard] Compare SUCCESSFULLY
# the expect pkt is
# ----------------------------------------
# Name        Type             Size  Value
# ----------------------------------------
# tr2         transaction_out  -     @1026
#   data_out  integral         64    'ha555aaa5
# ----------------------------------------
# the actual pkt is
# ----------------------------------------
# Name        Type             Size  Value
# ----------------------------------------
# tr_out      transaction_out  -     @971
#   data_out  integral         64    'ha555aaa5
# ----------------------------------------
```

图 6 打印信息

为了确保验证工作的完备性和可靠性,针对 DUT 的关键特性编写了大量受约束的随机激励,此时的功能覆盖率已经达到 70% ~80% 左右。为了进一步提高功能覆盖率,确保验证的完整性和可靠性,再针对事先制定的验证计划中的少数没有被随机测试所覆盖到的边界条件和逻辑施加对应的定向激励。

功能覆盖率情况如图 7 所示。本设计定义了两个功能覆盖率组,分别为发送数据功能覆盖率组 fm0_miller_in_cg 以及接收数据功能覆盖率组 fm0_miller_cg,每个功能覆盖率组里定义了若干仓。由图可见,待编码数据的所有可能值和 FM0 编码以及 Miller 编码的 3 种编码方式均被覆盖到,最终实现功能覆盖率达到 100%。

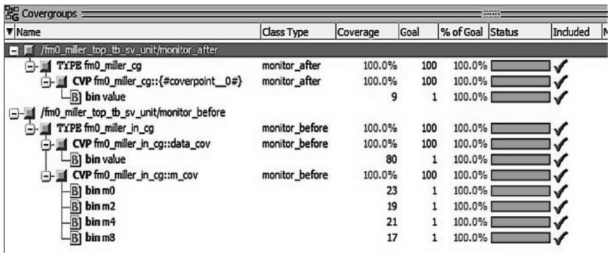

图 7　功能覆盖率

纯软件环境下的验证和基于 FPGA 的软硬件协同验证所得到的 UVM Report Summary 分别如图 8 和图 9 所示。

```
# --- UVM Report Summary ---
#
# ** Report counts by severity
# UVM_INFO :  722
# UVM_WARNING :    0
# UVM_ERROR :    1
# UVM_FATAL :    0
# ** Report counts by id
# [Questa UVM]     2
# [RNTST]     1
# [monitor_after]   160
# [monitor_after get from dut]   160
# [monitor_before]   160
# [monitor_before send]   160
# [my_scoreboard]    80
# ** Note: $finish    : D:/questasim/win64/../verilog_src/uvm-1.1d/src/base/uvm_root.svh(430)
#    Time: 16410 ns  Iteration: 105  Instance: /fm0_miller_top_tb
```

图 8　纯软件验证结果

```
# --- UVM Report Summary ---
#
# ** Report counts by severity
# UVM_INFO :  645
# UVM_WARNING :    0
# UVM_ERROR :    0
# UVM_FATAL :    0
# ** Report counts by id
# [Questa UVM]     2
# [RNTST]     1
# [driver send]   160
# [monitor_after]     1
# [monitor_after get from fpga]    80
# [monitor_before]     1
# [monitor_before get from driver]   160
# [monitor_before send]   160
# [my_scoreboard]    80
# ** Note: $finish    : D:/questasim/win64/../verilog_src/uvm-1.1d/src/base/uvm_root.svh(430)
#    Time: 11590 ns  Iteration: 105  Instance: /fm0_miller_top_tb
```

图 9　软硬件协同验证结果

对比两者可以发现,在运行相同测试用例的情况下,纯软件仿真所耗费的时间为 16 410 ns,而软硬件协同验证平台所耗费的时间为 11 590 ns,为纯软仿耗时的 71.81%。

经进一步验证发现,加速效果随着测试用例数量的增加及设计的复杂度而提高,即验证过程中运行的测试用例数量越多,待测设计越复杂,软硬件协同验证平台的验证效率越高。面对大规模 Soc 的验证工作,由于待测设计功能十分复杂,所需要的测试用例可能达到成百上千,再者,待测设计运行一次需要耗费大量时钟周期,若采用该软硬件协同验证平台将会大大缩短验证所需时间,提高验证效率。

4　结束语

利用硬件加速思想,结合 UVM 验证方法学,搭建了一个基于 FPGA 的软硬件协同验证平台。用 Verilog 语言编写串口收发模块、数据控制模块,连同待测设计 FM0 和 Miller 编码模块下载到 FPGA 硬件开发板中。采用串口通信作为物理通道,PC 端则使用 winsock API 编写串口驱动函数,仿真器通过 DPI 调用该接口函数,将 UVM 平台产生的受约束的随机激励下传至 DUT,经过 DUT 内部运行后输出编码结果,回传到仿真器中。仿真器将 DUT 结果与 C 语言编写的参考模型进行比对,判断 DUT 是否能正确地完成编码功能。该平台实现了软硬件整体协同运行,提高了验证效率,功能覆盖率达到了 100%,经验证,该平台能够为大型、复杂的 SoC 验证所复用。

参考文献:

[1] 魏文强,杜慧敏. 一种改进的软硬件协同验证平台的设计与实现[J]. 电子科技,2014,27(7):109-112.

[2] AHUJA S, POTTI P. An introduction to RFID technology [J]. Communications and Network,2010,2(3):183-186.

[3] 何　诚,陈小平,廖恬瑜,等. Co-Simulation 模式软硬件协同仿真体系结构及实现[J]. 电子科技大学学报,2007,36(S2):1114-1116.

[4] 张　强. UVM 实战[M]. 北京:机械工业出版社,2014:4-17.

[5] 吴迪飞. 基于 UVM 的 FPGA 数字下变频模块级验证方法的实现[J]. 电脑知识与技术,2020,16(10):269-272.

[6] 岳义杰. 基于 UVM 的 PCIe 总线接口数据传送顺序的验证方法[D]. 西安:西安电子科技大学,2019.

[7] 王国军,景为平. 基于覆盖率驱动的高频 RFID 芯片验证平台设计[J]. 电子技术应用,2016,42(4):28-30.

[8] 徐　飞,秦水介. 基于 UVM 的基带射频接口电路的验证[J]. 电子技术应用,2018,44(3):11-14.

[9] TASIRAN S, KEUTZER K. Coverage metrics for functional validation of hardware designs[J]. IEEE Design & Test of Computers,2001,18(4):3-7.

[10] 虞致国,魏敬和. 基于 SystemVerilog DPI 的 ARM SoC 虚拟调试验证平台的设计[J]. 微电子学与计算机,2009,26(11):117-119.

[11] 李　璐,周春良,冯　曦,等. 基于 DPI-C 接口的可扩展 SOC 验证平台[J]. 电子设计工程,2018,26(4):136-140.

[12] 陈　艇,陈少琴,肖尔丹. 基于 WinSock 的通信程序的设计[J]. 电脑编程技巧与维护,2013(12):55-57.

[13] 徐金娜,付方发,王进祥. 基于 FPGA 的软硬件协同仿真平台的数据通路设计[J]. 微电子学与计算机,2014(3):107-110.

[14] 游必辉,孙向明,肖　乐. 基于 UVM 的 MIC4 像素传感器读出系统验证[J]. 电子设计工程,2019,27(21):68-71.

[15] 苏　渊. 基于 UVM 的 AES 模块验证平台设计[D]. 西安:西安电子科技大学,2018.

[16] 赵经天. 基于 UVM 的 SAS 控制器模块验证[D]. 杭州:杭州电子科技大学,2019.

[17] XIE F, YANG G, SONG X. Component-based hardware/software co-verification for building trustworthy embedded systems[J]. Journal of Systems and Software,2007,80(5):643-654.