

# 基于 SystemC 参考模型的 UVM 验证平台设计

汪永峰, 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:**随着集成电路设计复杂度的不断提高,作为芯片开发周期中重要一环的芯片验证已经出现了逐渐乏力的趋势,传统的验证主要使用直接测试的方法,验证工程师们需要编写大量定向测试用例来满足验证的需求,这个过程既费时又费力,因此需要寻找新的验证方法来加快验证速度,提高验证效率。基于 SystemC 语言具有的强大的高层次建模能力以及 UVM 验证方法学具有的激励随机化、复用性高、以覆盖率为导向等诸多优势,结合 SystemC 语言和 UVM 验证方法学来搭建验证平台。使用基于 SystemVerilog 语言的 UVM 验证方法学搭建验证环境,并将 SystemC 语言编写的模型作为参考模型接入 UVM 验证平台,对超高频射频识别数字基带处理单元中读写器发送链路进行验证,统计覆盖率。结果表明,代码覆盖率和功能覆盖率均达到 100%,满足了芯片验证要求,相比于传统验证方法有效地缩短了验证时间,提高了验证效率。

**关键词:**芯片验证;UVM;SystemC;UVM Connect;覆盖率

中图分类号:TP33

文献标识码:A

文章编号:1673-629X(2021)07-0075-06

doi:10.3969/j.issn.1673-629X.2021.07.013

## Design of UVM Verification Platform Based on SystemC Reference Model

WANG Yong-feng, BU Gang

(School of Electronic Information Engineering, Nanjing University of Aeronautics and Astronautics,  
Nanjing 211106, China)

**Abstract:** With the increasing complexity of integrated circuit (IC) design, chip verification, as an important part of chip development cycle, has become increasingly weak. Traditional verification mainly uses direct testing methods. Verification engineers need to write a large number of directional test cases to meet the requirements of verification. This process is time-consuming and laborious, so it is necessary to find new verification methods to speed up verification and improve verification efficiency. Based on the powerful high-level modeling capability of SystemC language and the advantages of UVM verification methodology such as incentive randomization, high reusability and coverage orientation, a verification platform is built by combining SystemC language and UVM. The UVM based on SystemVerilog language is used to build the verification environment, and the model coding by SystemC language is connected into the UVM verification platform as the reference model to verify the UHF RFID physical layer forward link baseband circuit. The results show that the code coverage and function coverage reach 100%, which meet the chip verification requirements. Compared to traditional verification methods, it greatly shortens the verification time and improves the verification efficiency.

**Key words:** chip verification; UVM; SystemC; UVM Connect; coverage

## 0 引 言

当下芯片验证技术已经逐渐跟不上芯片设计的步伐,在完整的芯片研发过程中,芯片验证往往要占据整个芯片研发周期的 70% 以上<sup>[1]</sup>。可见,在不降低验证要求的情况下,缩短验证时间可以有效地缩短芯片研发周期,加快芯片上市时间。

该文旨在以超高频 (ultra high frequency, UHF) 射频识别 (radio frequency identification, RFID) 数字基带

处理单元中读写器发送链路为例,设计一款可复用的验证平台。由于 SystemC 语言可以用来搭建事务级、高抽象级的虚拟原型,并且其与 SystemVerilog 语言均支持事务级建模 (transaction level models, TLM) 通信机制,这为 SystemC 和 UVM 之间的通信提供了可能<sup>[2]</sup>。这里选用 SystemC 建模作为参考模型接入 UVM 验证平台,将参考模型的输出和寄存器传输级 (register transformation level, RTL) 设计的输出进行比

收稿日期:2020-09-21

修回日期:2021-01-26

基金项目:江苏省自然科学基金 (BK2012792)

作者简介:汪永峰 (1996-),男,硕士研究生,研究方向为数字集成电路验证;卜 刚,硕导,教授,研究方向为数模混合集成电路设计。

对。SystemC 语言更注重算法级的设计,与更偏向于物理级的 RTL 相比拥有更快的仿真速度。并且将具有可重用、激励随机化等优点的 UVM 与 SystemC 语言结合起来,不仅提高了验证的全面性,还缩短了验证所需要花费的时间。

## 1 UVM 的优势

通用验证方法学 (university verification methodology, UVM) 是一个以 SystemVerilog 类库为主体的验证平台开发框架,为验证人员提供了一系列通用验证组件 (university verification component, UVC), 例如 uvm\_driver、uvm\_monitor、uvm\_agent 等<sup>[3]</sup>。总的来说,UVM 之所以能够得到广大验证人员的青睐,是由于 UVM 验证方法学相比于其他的验证方法包含以下几点优势:

(1) UVM 拥有自己的基类库,验证工程师可以利用继承功能复用这些通用验证组件来构建需要的验证环境,为验证平台的搭建提供了便利,缩短了搭建验证平台所需要的时间<sup>[4]</sup>。

(2) UVM 搭建出来的验证平台具有特有的结构,使得整个平台层次清晰,代码可读性大大增加,也省去了工程师自己构建验证结构的时间<sup>[5]</sup>。

(3) UVM 拥用属于自己的运行机制,通过 phase 机制使得仿真阶段也变得层次化,不仅仅指的是不同 phase 之间的层次顺序,也包含了不同组件中相同 phase 的层次化关系<sup>[6]</sup>。

(4) UVM 拥有独特的传输机制。TLM 通信机制增强了 UVM 各个组件之间的独立性,其端口种类繁多,为满足不同通信需求提供了基础。同时由于 SystemC 同样支持 TLM 通信,这为 UVM 与 System C 之间的通信提供了基础<sup>[7]</sup>。

(5) UVM 拥有自己独特的重载和覆盖机制,极大地提高了代码的可重用性。

## 2 待测设计介绍

本次待测设计采用由 Verilog 编写的 UHF\_RFID 数字基带处理单元中的读写器发送链路。该设计从 ISO/IEC 18000-6C 协议标准出发,实现了读写器处理并发送,标签接收并处理的功能。如图 1 所示,阅读器发送侧主要由七个模块构成,分别为时钟模块、计数模块、并串转换模块、CRC 校验码生成模块、异步 FIFO 模块、PIE 编码模块以及同步码添加模块。发送数据时,首先需要通过计数模块计算发送数据位数并判断数据类型,为并串转换及数据编码做准备,接着将并行数据转换成串行数据,传递到 CRC 校验码生成模块,生成数据的循环冗余校验 (cyclic redundancy check, CRC) 码添加在数据尾部。根据数据类型的不同,阅读器发送模块的 CRC 校验分为 CRC5 和 CRC16 两种类型,当发送数据为 Query 命令类型时,进行 CRC5 校验,其他类型则进行 CRC16 校验。之后再将数据送入异步 FIFO 进行缓存,接着对 FIFO 读出的数据进行 PIE 编码,最后为其添加同步码发送给标签<sup>[8]</sup>。

而标签接收侧主要由四个模块构成,分别为同步码检测模块、PIE 解码模块、CRC 校验码检测模块以及串并转换模块。标签接收侧完成的功能则和读写器发送侧功能相反,数据首先通过同步码检测模块,确定数据的位置并提取紧接在同步码之后的正确数据,随后将接收到的数据传递到 PIE 解码模块进行 PIE 解码操作,将解码后的数据按照接收数据命令类型的不同,分别进行 CRC5 和 CRC16 校验,校验成功后将串行数据恢复成并行数据。

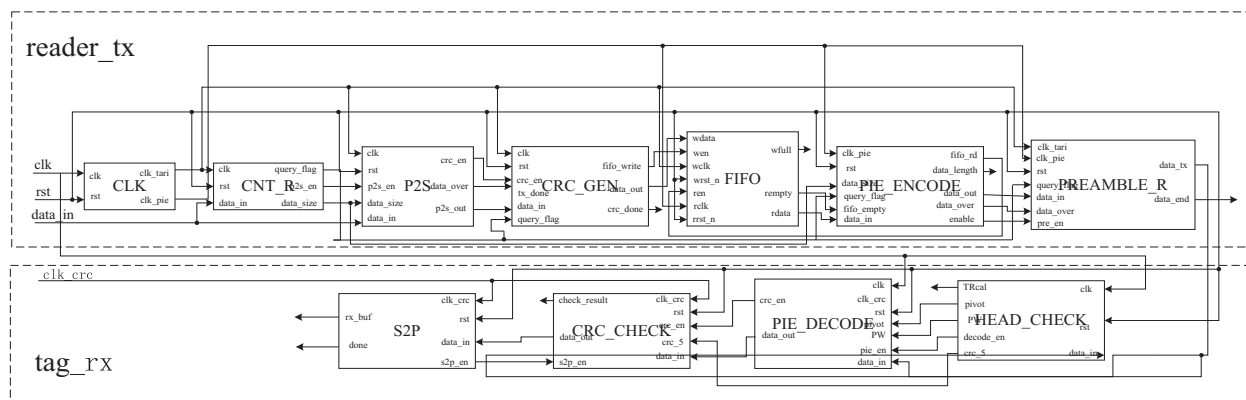


图 1 待测设计电路模型

### 2.1 编码模块设计

协议规定读写器发送链路的编码方式为脉冲宽度编码 (pulse interval encoding, PIE)。通过定义数据-0 和数据-1 编码后不同的码元长度来实现,PIE 编码方

式如图 2 所示,编码后 0 或 1 码元的长度主要取决于 Tari 和 PW 两个参数。读写器对标签发信的基准时间间隔为 Tari,为一个 0 码元持续的时间。这个值可以根据实际情况适当修改,最佳读写器对标签 Tari 值如

表 1 所示。PW 的参数可以在 0 到 Tari 之间选取。从 为 0.5tari 到 tari。  
图上可以看出码元 1 的长度为 Tari+x, x 的取值范围

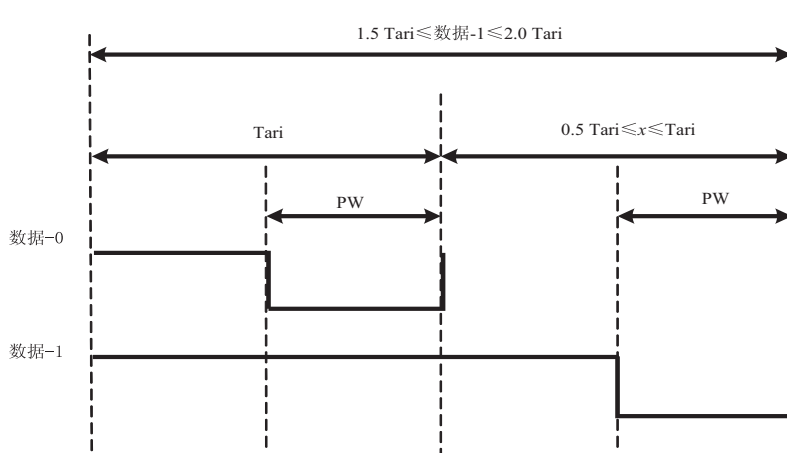
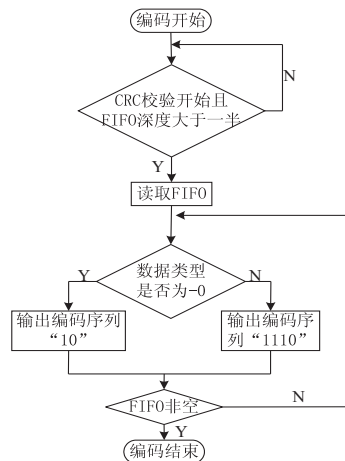


图 2 PIE 编码

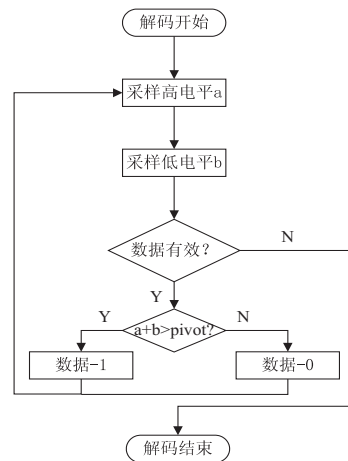
表 1 最佳读写器对标签 Tari 值

Tari 值/ $\mu\text{s}$	Tari-值公差
6.25	+/-1%
12.50	+/-1%
25.00	+/-1%

为方便编码,在下面的设计和验证中,取 x 的值为 Tari。因此,在 PIE 编码中,0 码元被编码为“10”序列,而 1 码元被编码为“1110”序列。



(1) PIE 编码流程图



(2) PIE 解码流程图

图 3 PIE 编解码算法流程

## 2.2 解码模块设计

根据协议可知,在解码时,最重要的参数是 RTcal,其值为一个数据-0 与一个数据-1 的长度之和。由于数据-0 和数据-1 均由高电平和低电平两个部分构成,当接收到的数据为有效数据时,采用频率高于码元速率的时钟分别对接收到数据的高低电平进行采样,得到数据 a 和数据 b,并取 RTcal 值的一半为常量 pivot,与 a 和 b 之和进行对比,若 a 与 b 的和小于 pivot,则接收到的数据为数据-0,反之接收到的数据为数据-1<sup>[9]</sup>。PIE 解码流程如图 3(2)所示。

在 PIE 编码模块开始工作之前,首先要判断 CRC 校验模块是否已经开始工作并将校验后的数据写入 FIFO 进行数据缓冲,根据 FIFO 中有效数据的状态决定是否进行编码操作,若 FIFO 已经写入超过一半数据,则 PIE 编码模块开始工作,从异步 FIFO 中读取数据并进行编码直到 FIFO 为空,表示已对全部数据进行编码。PIE 编码算法流程如图 3(1)所示。

根据协议标准,在解码时,若接收到比  $4 * RTcal$  长的符号为不良数据,因此在解码时, a 与 b 还需满足下列两个条件:

- (1)  $a+b < 4 * RTcal$ , 即  $a+b < 8 * pivot$
- (2)  $0.625 * Tari < b < 0.525 * Tari$

## 3 验证平台设计

### 3.1 实现基础

SystemC 和 UVM 验证平台的搭建,其实现基础是 UVM Connect(UVMC)通信包,这个包集成了已有的

SystemC 和 UVM 的 TLM 通信标准,使得 UVM 和 SystemC 之间 TLM 模型可以方便地实现通信并且不需要修改原本已有的代码,降低了 SC 和 UVM 的 TLM 模型复用的难度。原有的 TLM 模型不再需要继承其他的基类,而是通过外部代理的模式解决了这一问题,同时 UVM 一侧的 transaction 类并不要求在 factory 中注册,减少了对 factory 机制的依赖。SV 和 SC 两侧的 transaction 类并不要求完全一致,数据在 SV 和 SC 之间可以由各自的 converter 函数完成数据转换,这样的方式进一步提高了原有 TLM 模型的复用性。

UVMC 库不仅提供了 SystemC 和 SystemVerilog 模型与组件之间的 TLM1.0 和 TLM2.0 连接方法,它还将 UVM 命令 API 方法输出到 SystemC 一侧,用于从 SystemC(C 或 C++)控制和访问 UVM 仿真<sup>[10]</sup>,这些 API 主要有以下几个方面:

- (1) 等待 UVM 到一个特定的仿真阶段。
- (2) 挂起或放下 objection 来控制 UVM 测试进程。
- (3) 通过 UVM config\_db 设置或得到配置的对象。
- (4) 通过 config\_db 覆盖类或实例的类型。
- (5) 打印 UVM 环境组件的拓扑结构。

对于 SV 和 SC,在代码中分别利用 UVMC 库提供的类似的函数来完成相应的 TLM port 的注册,在本设计中的注册代码如下:

SV TLM2:

```
uvmc_tlm#( my_transaction )::connect( mdl. out,
“reader2tag” );
```

SC TLM2:

```
uvmc_connect( cons. in, “reader2tag” );
```

reader2tag 是用来注册该端口的字符串,在使用这些函数时,UVMC 会在 SV 和 SC 两侧都注册端口的句柄和名字(reader2tag),而在后期连接阶段,只要注册端口的名字匹配,UVMC 就会将这两个端口连接起来,而并不关心它们是什么语言。

### 3.2 验证组件设计

验证平台的结构框架如图 4 所示,该验证平台主要包含的组件及其对应的功能如下所示:

(1) interface:接口声明,主要包含读写器需要发送的数据和命令接口,用于实现待测设计( design under test, DUT)和 testbench 之间的通信<sup>[11]</sup>。

(2) transaction:产生组件之间通信最基本的数据包,本设计中包含读写器需要发送的命令和数据,由于需要预留 8 bit 用于储存 data\_size,因此在 transaction 内需要限制发送数据位宽不能超过 120 bit,因此在 transaction 添加如下约束:

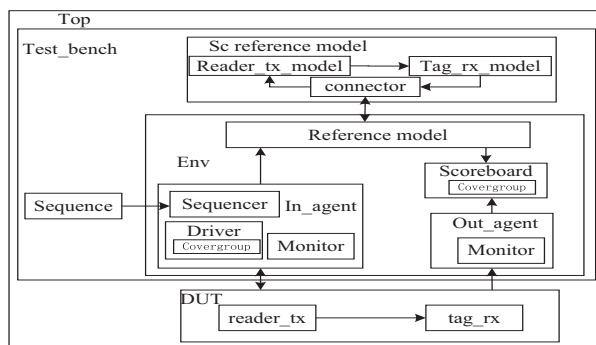


图 4 验证平台结构

```
constraint data_cons {
data_in[ 127:120 ] == 8'h0;
}
```

(3) sequence:用于产生 transaction,可以控制 transaction 的数量,以及对 transaction 进行约束从而产生符合预期的激励等<sup>[12]</sup>。

(4) sequencer:用于将 sequence 产生的数据包传递给 driver。

(5) driver:负责驱动 transaction,将激励分别通过 virtual interface 和 seq\_item\_port 送到 DUT 和 reference model,本设计在 driver 中定义了一个覆盖组,用于功能覆盖率的统计,主要包含对随机输入数据可能存在情况的覆盖,由于传输数据位宽较大,用随机的方法全部遍历不太方便,因此对数据范围进行了划分,设置各个范围的最低覆盖次数为 1,并且对一些特殊情况、边界情况进行了覆盖,确保 DUT 在所有情况下都能完成正常的功能<sup>[13]</sup>。覆盖组定义代码如下:

```
covergroup trans_cg;
data_cov: coverpoint req. data_in {
bins all1 = { 18'h3ffff };
bins special1 = { 18'h15555 };
bins special2 = { 18'h2aaaa };
bins data_0 = { [0:18'h03fff] };
bins data_1 = { [ 18'h04000:18'h07fff] };
bins data_2 = { [ 18'h08000:18'h0bfff] };
.....
}
endgroup
```

(6) monitor:本设计中包含两个 monitor,一个从输入虚接口获取数据包,用于监测 driver 发送给 DUT 的数据包,确保读写器接收到的数据包正确无误,另一个从输出虚接口获取数据包,用于监测标签最终输出的数据包,并将其通过 uvm\_analysis\_port 传递给 scoreboard 进行数据比对。monitor 关键代码如下:

```
task my_monitor;:main_phase(uvm_phase phase);
while(1) begin
tr=new("tr");
collect_one_pkt(tr);
```



```

        ap. write( tr );
    end
endtask
task my_monitor;:collect_one_pkt( my_transaction tr );
    while(1) begin
        @( posedge vif. clk );
        if( vif. valid) break;
    end
    ...
    while( vif. valid) begin
        tr. data_in=vif. data_in;
        @( posedge vif. clk );
    end
    ...
endtask

```

(7)agent:将 driver 和 monitor 以及 sequencer 封装在一起,可以通过配置 is\_active 和 is\_passive 参数来选择是否使用 driver 和 sequencer<sup>[14]</sup>,并在 connect\_phase 中建立 driver 和 sequencer 之间的连接。

(8)scoreboard:分别接收 out\_agent 和 reference\_model 传递来的数据作为期望结果和实际结果,并在 scoreboard 中将期望结果和实际结果进行自动比对,若两者一致,则数据传输正确,打印“Compare SUCCESSFULLY”表示数据对比成功,若两者不一致,则数据传输错误,打印“Compare FAILED”表示数据对比失败,并分别打印出期望的结果和实际的结果以作对比。和 driver 类似,在 scoreboard 中也定义了一个覆盖组,用于对接收到的数据统计功能覆盖率。

(9)reference\_model:该类在本设计中自身不对数据进行任何操作,仅用于和 SystemC 语言编写的真正参考模型进行数据通信以及控制与 SystemC 参考模型间的事务数量。UVM 侧实现通信的关键代码如下:

```

my_transaction pkt=new();
delay. set_abstime(3,1e-4);
port. get(pkt);
pkt. print();
out. b_transport( pkt,delay);
pkt. print();
ap. write( pkt);

```

(10)env:封装 in\_agent、out\_agent、scoreboard、reference\_model 并将它们在 connect\_phase 中进行连接,实现组件之间的数据通信。

(11)testbench:例化 env,启动 sequence。

(12)top:连接 testbench 和 DUT,运用 config 机制配置顶层接口连接,启动 testcase。

(13)DUT:由 Verilog 语言编写的 UHF\_RFID 数字基带处理单元读写器发送链路设计,根据 testbench 送来的激励产生相应的输出结果,并送往 scoreboard

做比对<sup>[15]</sup>。

(14)SC Ref Model:由 SystemC 编写的 UHF\_RFID 数字基带处理单元读写器发送链路模型,由 UVM Connect 包将其接入 testbench,根据 testbench 送来的激励产生标准的输出结果,并送往 scoreboard 做比对。SC 侧实现数据通信的关键代码如下:

```

virtual void b_transport( T& t, sc_core::sc_time& delay ) {
    cout << sc_time_stamp() << " SC consumer executing pack-
et;"

    << endl << " " << t << endl;
    data_in=t. data_in;
    cnt_en=1;
    wait(40,SC_NS);
    cnt_en=0;
    wait(s2p_done. posedge_event());
    t. data_in=data_out;
    cout << sc_time_stamp() << " SC consumer packet execu-
ted;"

    << endl << " " << t << endl;
    delay=SC_ZERO_TIME;
}

```

## 4 仿真结果

本设计使用 synopsys 公司的 VCS 软件进行最后的仿真实验,查看波形和计分板的打印输出来判断 DUT 的功能实现情况,并根据代码覆盖率和功能覆盖率来判断验证的完整性。

图 5 为发送 50 个随机数据产生的 DUT 顶层波形图。在图中随机选取一时间点分析数据,如图所示,选取时间为 82418316451ns,在此时间点读写器发送数据 data\_in 为 128'h008d\_7334\_626e\_67ad\_60db\_4cb6\_d1d3\_536f,标签接收到数据寄存在 rx\_buf 中,可见接收到的数据为 128'h8d73\_3462\_6e67\_ad60\_db4c\_b6d1\_d35d\_6f78,接收数据末八位 8'h78 指的是发送数据长度,换算成 10 进制为 120,其余位为发送数据,经对比数据及数据长度与实际发送数据一致,DUT 功能正确。



图5 仿真波形

任意截取一个数据包的仿真对比结果报告如图 6 所示。图中 expect\_pkt 是由 SC 参考模型输出的数据,其数值为 128'he005\_30e2\_8907\_2fa5\_7f0a\_0e90\_f420\_4078,actual\_pkt 是由输出数据 monitor 监测 DUT 标签接收模块输出最终传递到 scoreboard 的数据,其数值

