文章编号:1671-4598(2018)06-0264-04 DOI:10.16526/j.cnki.11—4762/tp.2018.06.067 中图分类号:TP391.9 文献标识码:A

SV DPI 技术在 FPGA 仿真验证的应用探讨

祝周荣，关俊强，李前进，赵 超，刘芳汝
（上海航天电子技术研究所，上海 201109）

摘要：SystemVerilog 作为近年来逐渐流行的 FPGA 验证语言，包含了丰富的验证特性：DPI、断言技术、功能覆盖率等，为 FPGA 仿真验证提供强大的支持作用；文章提出一种基于 DPI 接口的 FPGA 仿真验证方法，利用 SystemVerilog 的 DPI 接口技术在验证平台中实现对 C 或 C++的调用，通过编写 C 函数来实现复杂激励模型设计，在验证平台中实现 FPGA 和模型输出结果自动比对，从而实现测试的自动化，提供测试的完备性；实验表明：利用该方法搭建的仿真验证平台相对于传统的纯 verilog 验证平台，具有更高的仿真效率和验证的灵活性，也为算法级 FPGA 设计的确认测试提供了新的验证思路。

关键词：FPGA；SV DPI 仿真验证平台

Application of SV DPI Technology in FPGA Simulation Verification

Zhu Zhourong, Guan Junqiang, Li Qianjin, Liu Fangru

Abstract: As gradually popular FPGA validation language in recent years, System Verilog contains rich validation features: DPI, assertion technology, functional coverage, etc. DPI interface technology can help verification engineer to call C or C++ in the verification platform, to implement complex incentive model by writing C function, to compare the output of the FPGA and the model automatically in the verification platform, so as to realize the automation of test, to provide the completeness of the test. Experiments show that: by using the method of simulation, verification platform compared with the traditional pure verilog verification platform, has higher efficiency of the simulation and validation of flexibility, as well as provides a new thought of validation for algorithm level FPGA design validation test.

Keywords: field programmable gate array; systemVerilog direct programming Interface; simulation verification platform

0 引言

目前数字系统设计中应用最为广泛且最为有效的验证方法就是仿真。除了验证单个模块功能的正确性、接口交互以及整个系统功能的正确性，还可以模拟边界测试、故障测试等，而硬件测试上往往是不能进行的。但是随着 FPGA 功能趋于复杂以及实现算法复杂度提高，其输入信号可能是通过复杂模型产生的，比如信号的调制，傅里叶变换等。验证人员受专业限制，无法理解和模拟该类输入信号，只能向设计师要数据源。那么谁来保证数据源的正确性？如果使用该数据源出错，那么问题是出在数据源，还是 FPGA 设计呢？很显然，这样无法保证验证工作的独立性。即使是明白专业的验证人员通过 Verilog 或 VHDL 来实现复杂的输入模型，同样存在致命的缺点。比如调用 Verilog 和 VHDL 的 IP 来实现正弦函数，由于 HDL 并行化的特点，在仿真中将占用大量的内存，极大的影响仿真的速度。此外，如何在一个独立的验证平台中验证 FPGA 算法功能是否正确。比如计算一幅图像的方差，通常方法是额外编写一个 C 程序，然后人工的比对双方的结果，以验证该算法是否正确，没有进行实时比对，仍然存在测试有效性的问题。

本文提出了一种新的验证解决方案，利用仿真工具中的 SV DPI 技术，实现了在 SV 仿真验证平台中调用 C 或 C++编程语言[1]，利用高级语言可以更加方便的实现激励读取、参考模型构建等功能。本文就该类解决方法进行相关的阐述和讨论，这对于提高 FPGA 验证充分性和验证的独立性，以及验证的灵活性都有着重要意义。

收稿日期:2018-03-14； **修回日期**:2018-04-10。

作者简介:祝周荣(1977-)女,上海人,硕士,高级工程师,主要从事 FPGA 第三方验证方向的研究。

1 SV DPI 仿真验证方法

1.1 PLI 和 SV DPI

Verilog 是现有航天 FPGA 验证中最常用的语言。Verilog 使用编程语言接口 PLI（Programming Language Interface）来跟 C 语言程序交互。使用 PLI 可以生成延迟计算器，以连接和同步多个仿真器，并增加诸如波形显示等调试工具。但是 PLI 非常繁琐，即使通过 PLI 连接一个简单的 C 程序，都需要编写大量的代码，并需理解很多概念，这些概念包括多个仿真阶段的同步、调用段、实例指针等等。此外，PLI 给仿真带来了额外的负担，因为为了保护 Verilog 数据结构，仿真器必须不断地在 Verilog 和 C 语言域之间复制数据。所以现有的 FPGA 验证过程中，FPGA 验证人员基本不会去使用 PLI。

而新一代的验证语言 SystemVerilog 引入了直接编程接口 DPI[2-3]，它能更加简单地连接 C、C++编程语言。一旦你声明或者使用 import 语言“导入”了一个 C 子程序，你就可以像调用 SystemVerilog 中的子程序一样来调用它，使用起来非常方便。首先，通过高级语言实现复杂模型比使用 hdl 语言要轻松很多，且仿真速度快。比如 C 语言已经提供了很多库函数，直接调用即可，无须重新编写。这样既保证了激励编写的

正确性，又提高了可重用性。同时 C 语言目标代码的执行速度比 hdl 仿真速度要至少提高一个数量级。其次，FPGA 中实现的算法都有可靠的高级语言模型，基于 DPI 技术的 FPGA 仿真验证平台可以直接调用该模型，实现同一份激励输入到 FPGA 设计和高级语言模型，然后比对两个结果，以判断 FPGA 设计实现正确性。通过该方法，验证人员可以不受专业限制，将 FPGA 设计完全看做是一个黑盒，保证验证方的独立性。再次将 C 函数连入验证平台，容易实现验证平台的完整性，为自动测试创造了条件。最后，利用 SystemVerilog 的受限制随机激励生成功能、断言和功能覆盖率功能可以大大提高测试的效率和质量。

1.2　SV DPI 验证技术使用步骤

要使用 SV DPI 验证技术顺利将 SystemVerilog 和 C 联合起来仿真，应按照下面 4 个步骤进行。

1）编写 C 代码实现算法。DPI 包括两个完全独立的层次，分别是 SystemVerilog 和 C，在 C 代码中需要声明包含头文件 svdpi. h，因为在 svdpi. h 中包含了 SystemVerilog DPI 结构和方法的定义[4]。

2）实现 C 与 SystemVerilog 通信。SystemVerilog 平台中通过导入函数和任务来调用 C 代码，DPI 也允许在 C 代码中通过导出函数和任务来调用 SystemVerilog 中的方法[5]。被调用的 SystemVerilog 方法可以是一个保存 C 函数操作结果的简单任务，或者是一个实现部分硬件模型的耗时任务[6]。

SystemVerilog 平台中导入函数和任务的定义如下：

import "DPI" [c_identifier =][pure][context]function type name (args);

import "DPI" [c_identifier =] [context]task type name(args);

SystemVerilog 平台中导出函数和任务的定义如下：

export "DPI" [C_identifier =]function type name;

export "DPI" [C_identifier =]task type name;

3）匹配数据类型映射。由于 SystemVerilog 数据类型和 C 语言数据类型差异比较大，SystemVerilog3.1 语言手册定义了通过 DPI 传递的每个数据类型的匹配模式。具体见表 1[4]。

表 1　SystemVerilog 和 C 语言之间的数据类型映射

SystemVerilog	C(输入)	C(输出)
byte	char	char*
shortint	short int	short int*
int	int	int*
longint	long long int	long int*
shortreal	float	float*
real	double	double*
string	const char*	char**
string[N]	const char**	char**
bit	svBit or unsigned char	svBit* or unsigned char
logic,reg	svLogic or unsigned char	svLogic* or unsigned char*
bit[N:0]	const svBitVecVal*	svBitVecVal*
reg[N:0]	const svLogicVecVal*	svLogicVecVal*
logic[N:0]		
open array[]	const svOpenArrayHandle	svOpenArrayHandle
chandle	const void*	void*

其中对于 bit 和 logic 类型匹配在 svdpi. h 文件中有专门的指定。使用者需根据 svdpi. h 文件中的定义来选择匹配的数据类型。另外需要注意的是：DPI 不会检查数据类型的兼容性，需要使用者自己保证数据匹配的正确性。

4）利用仿真工具编译 C 程序的方法，生成最终的目标码，并和 SystemVerilog 混合运行。

2　工程实例

2.1　仿真模型构建

某 FPGA 产品根据采集到的多音组合调频信号，依据一定的解调算法实现最终的解码[9]。验证人员在 VCS 环境搭建基于 DPI 技术的 FPGA 仿真验证平台如图 1 所示。解出的密码包括 m 个码元，每个码元为 n 位。每个码元都是由 n 个单音（正弦波）中的 4 个叠加而成，而每个单音的频率是不一样的，若 m 个码元解密出的数据与存储密码一致，则解密成功。每个多音组合调频信号指令格式如图 2 所示。

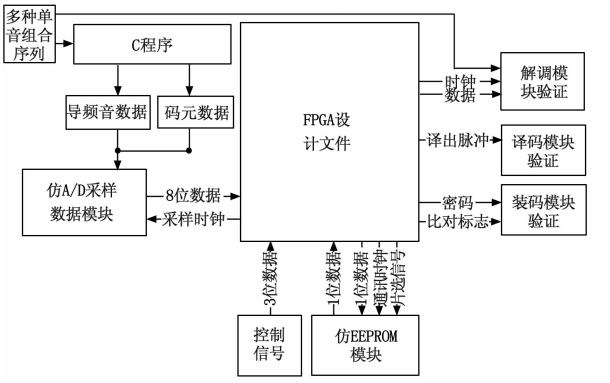

图 1　某 FPGA 产品验证架构

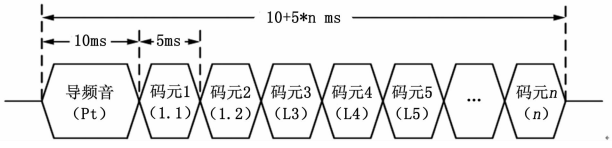

图 2　多音组合调频信号指令格式

该 FPGA 产生 AD 芯片的采样的时钟和读控制信号，AD 芯片根据相应的时序采集外围的模拟信号，AD 芯片将其转换为数字信号输出给 FPGA 设计。基于图 1 的验证架构，验证人员根据 DPI 验证流程按照以下步骤，实现整个 FPGA 验证架构。

1）编写 C 程序模拟完成多个单音数据（正弦波）叠加后的数字化采样。在 SystemVerilog 验证架构中随机挑选 4 个单音进行叠加，生成多种单音组合序列，SystemVerilog 将这些序列指令码元的组合情况通知 C 程序，C 程序根据输入的参数实现 AD 采样数据的模拟。C 程序先根据参数频率计算 15 组码元数据的角速度值和导频音角速度值，利用角速度值和幅值参数生成 16 组标准的 2048 个点精度的二进制偏移正弦波，然后根据输入的单音组合序列，当有组合序列对应位置为 1 时，将对应的正弦波叠加，再根据组合情况，按照时间顺序，前 10 ms 输出导频音正弦波，5 ms 输出第一组叠加的多音组合，下个 5 ms 输出第二组多音组合，依次输出 7 个。

2）在 SystemVerilog 中调用 C 程序，实现数据的相互通

```
extern "C" { //2*3.14/w = T/t = 1/ft
void MaYuan(int ZhiLing[7],double DaoPin_FUZHI,double PinLv)
{
    int i,j;
    int ZhiLing_i,ZhiLing_j;
    int sum;
    float  sin_sum;
    float w_ =  2*3.14; //
    float w[15];//单音角速度
    float w0;//导频音角速度
    //对15组多音组合调频正弦函数角速度进行赋值
    for(i = 0;i<15;i++) {w[i] = w_ * ( 6.4 + i * 0.4 ) /204.8;  }
    w0 = w_ * PinLv * 5.12 /204.8;
    i = 0;  j = 0;
    char fileName[20], buffer[20];
    FILE* writeData;
    //生成导频音
    cout << "已生成导频音，生成文件为DaoPin.txt\n";
    memcpy(fileName, "DaoPin.txt", sizeof("DaoPin.txt"));//生成文件名
    writeData = fopen(fileName, "w");
    while(i<2048)
    {
        sum = 128+DaoPin_FUZHI/1.25*128*sin(w0*i);//导频音正弦值
        sprintf((char*)buffer, "%02X ", sum);
        fwrite(buffer, 1, 3, writeData);
        if((i+1)%16==0) fwrite("\n", 1, 1, writeData);i++;
    }
    fclose(writeData);
    i=0;//生成码元
    memcpy(fileName, "frame00.txt", sizeof("frame00.txt"));
    int zhi[7];
    while(i<7){zhi[i] = ZhiLing[i]; i++;}
    i = 0;
    int zhilingl[7][15];
    for(i=0;i<7;i++)
    {
        for(j=0;j<15;j++){zhilingl[i][j] = zhi[i]%2;
        zhi[i] = zhi[i]/2;}
        j = 0;
    }
    i=0;j=0;
    for(ZhiLing_i=0;ZhiLing_i<7;ZhiLing_i++)
    {
        cout << "please input the four number of mayuan";
        fileName[5] = ZhiLing_i/10+'0';
        fileName[6] = ZhiLing_i%10+'0';
```

图 3 C 程序模拟 AD 输入数据生成代码

信。在 SystemVerilog 中，通过 import “DPI－C” 声明定义 C 函数的原型[7]，接着在 SystemVerilog 的任务或者进程中就可以调用 C 函数 MaYuan。为了提高仿真的效率，应尽量减少 SystemVerilog 和 C 函数的通讯，所以在该平台中，将 C 函数计算出的 7 个码元对应的数字化采样数据一次性写入到文件中，SystemVerilog 平台通过 Read _ file 过程使用读取文件的方式来实现 AD 数据的注入，而不是通过 C 函数返回的方式，这样可以适当提高仿真的效率。

```
import "DPI" function  MaYuan(input int ZhiLing[0:6],real DaoPin_FUZHI,real PinLv);
task set_ZhiLing_MaYuan;
input int ZhiLing[0:6];
input real DaoPin_FUZHI;
input real PinLv;
begin
    MaYuan(ZhiLing,DaoPin_FUZHI,PinLv);
    Read_file;
    ad_flag = 1;
    @(negedge ad_flag);
end
endtask
task funcl_7test;
begin
    $fdisplay(Sim_Log_File,"********FUNCl_2测试，导频音提取功能测试********");
    $fdisplay(Sim_Log_File,"     设置导频幅值为0.5V");
    DaoPin_FUZHI = 0.06;
    set_ZhiLing_MaYuan(ZhiLing,DaoPin_FUZHI,PinLv);
end
endtask
```

图 4 在 SV 中调用 C 程序

3）注意匹配 SystemVerilog 的数据类型和 C 语言的数据类型的兼容性映射。在该应用实例中 SystemVerilog 传递给 C 程序的接口包括指令组合（ZhiLing［0：6］）、幅值（DaoPin _ FUZHI）、频率（PinLv）。SystemVerilog 的数据类型和 C 语言的数据类型的兼容性如表 2 所示。

表 2 SystemVerilog 的数据类型和 C 语言的数据类型映射

信号名	SystemVerilog	C(输入)
ZhiLing	int［0：6］	int［7］
DaoPin_FUZHI	real	double
PinLv	real	double

4）完成解调出的密码与输入的指令码元自动比对功能。图 1 的解调模块将根据时钟收集解调出的码元，同时将其按照特定的格式转换为指令，与传递给 C 程序的指令组合（ZhiLing［0：6］）进行比较，完成自动比对功能，以确认 FPGA 设计的解调功能是否正确。

5）利用仿真工具 VCS 编译[8] C 程序的方法，从而生成最终的目标码。该 FPGA 仿真工具采用了 linux 环境下的 VCS 仿真工具，VCS 调用 linux 下的 gcc 编译器，可以实现在对 FPGA 代码编译的同时，也完成对 C 程序的编译以及目标码的生成，如图 5 所示。

```
comp:
    vcs -full64  -debug_all -lca -sverilog  ./ad.cpp  +vcs+initmem+0 +vcs+initreg+0 $(MODULE_NAME) -o $(OUTPUT)
```

图 5 仿真工具 VCS 编译 C 程序

2.2 仿真结果与分析

对于复杂算法类 FPGA 设计，怎样自动生成测试用例数据、怎样进行测试用例数据注入和进行结果分析是测试验证工程师面临的一大难题。传统的仿真验证方法只能通过 matlab 生成数据源，利用时钟导入的方式进行仿真验证，只能进行正常的功能测试，更无法进行结果比对。利用 3.1 节描述的仿真模型，验证人员通过 ZhiLing、DaoPin _ FUZHI 和 PinLv 参数的配置以及 MaYuan 函数的灵活调用，实现了表 3 中的 11 个测试用例，并发现当预令码元不是由四个单音组成时，解码结果不正常的问题。而使用传统的仿真验证或者硬件测试方法要实现表 3 中的所有测试用例非常麻烦，耗时巨大，因此之前一直没有发现问题。

由此可知，通过上述方法搭建的基于 DPI 技术的 FPGA 仿真验证平台，具备如下几个优点：

1）由高级语言 C 编写的激励输入，相比 Verilog 中的 IP 调用[10]，无疑极大的提高了仿真效率。

2）验证人员通过 C 代码编写的激励向量，无须向设计师索取激励输入数据，保证了验证工作的独立性。

3）依据随机产生的指令组合，然后根据 C 代码产生的 AD 数据作为 FPGA 输入，再对 FPGA 解码出的指令组合与随机产生的指令组合进行自动比对，提高了测试的自动化和测试的完备性。

3 结语

复杂算法 FPGA 仿真验证的有效性一直是 FPGA 验证的热点问题，本文提出的利用 SV DPI 仿真技术实现在 SV 仿真验证平台中调用 C 或 C＋＋的验证方案。经工程应用论证，利用 SV DPI 技术在 SV 仿真验证平台中调用 C 或 C＋＋编程语言，可以更加方便的实现激励读取、参考模型构建、测试结果自动比对等功能。基于 SystemVerilog 的 FPGA 验证平台，可以通过 DPI 实现 SystemVerilog 平台与高级语言（C、C＋＋

表 3　调频和多音组合解调功能测试表

序号	测试用例	测试内容	要求	结果	结论
1	正常幅值输入	输入信号为 12 位 AD，进行归一化，对输入的调频信号进行 0.25、0.5、0.75 和 1.0 的测试	能正确解调出码元	正确解调出码元	满足要求
2	最小幅值输入	AD 输入的调频信号有效位为 4bit(0.0052V)和 2bit(0.0013V)；符号位＋1bit 数据位；只有符号位	能正确解调出码元	正确解调出码元	满足要求
3	调频信号信噪比	生成幅值 0.237 的噪声信号，输入信号幅值 0.75，使得信噪比为 10dB 测试	信噪比不低于 10dB 时可以正常解调	正确解调出码元	满足要求
		和输入信噪比大于 10dB 随机噪声测试	可以正确解调信噪比大于 10dB 的调频信号	正确解调信噪比大于 10dB 的调频信号	满足要求
4	码元帧调频信号发送方式	两帧调频信号中间无间隔发送	可以正确解调连续发送调频信号	正确解调出码元	满足要求
		两帧调频信号中间间隔时间随机不同	可以正确解调突发发送调频信号	正确解调出码元	满足要求
5	调制信号频率错误	导频音调制信号频率设置与码元一样	不能解调出码元	无码元输出	满足要求
		码元调制信号设置间隔不正常	解调出的码元不正确	解调的码元与设置值不一致	满足要求
6	码元单音个数测试	调频信号发送超过 4 个单音的码元	正确解调出码元	实测解调出低 4 个单音码元	出具问题单
		调频信号发送少于 4 个单音的码元	正确解调出码元	实测多解出 1 个单音码元	
7	导频音使能信号宽度测量	输入一帧码元，测量导频音提取成功后使能译码有效时间	＞＝35ms	35.156ms	满足
8	载波频率偏移验证	载波频率 5 个频偏点测试	载波频率范围内均正常解码	5 个频偏点均正常解码	满足要求
9	调制信号频率偏移测试	调制频率 5 个测试点	调制频率均能正常解码	5 个测试点均正常解码	满足要求
10	15 位码元的 4 单音组合测试	把 15 位选 4 单音的所有组合按顺序分成五段，每段取一帧预令和一帧动令测试	把 15 位选 4 单音的所有组合都能正常解码	把 15 位选 4 单音的所有组合分成五段，每段取一帧预令和一帧动令都能正常解码	满足要求
11	载波频偏和调制频偏综合测试	载波频偏 1 加调制频率 1	正确解出输入的码元	正确解出输入的码元	满足要求
		载波频偏 2 加调制频率 2	正确解出输入的码元	正确解出输入的码元	满足要求

等）的通信，让 Systemverilog 的强大验证能力能够在事物处理级模型的验证工作中充分发挥[11]，相对于传统的纯 verilog 验证平台，大大提高了仿真效率和验证的灵活性，同时也为算法级 FPGA 设计的确认测试提供了新的验证思路。

参考文献：

[1] 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 (11)：117－119.

[2] 闫　沫，张　媛. 基于 SystemVerilog 语言的设计验证技术 [J]. 现代电子技术，200806：8－11.

[3] 耿　介，于治楼，毕研山. 一种 UVM 验证环境中复用 C 程序测试向量的方法：CN104899138A [P]. 2015－09－09.

[4] Synopsys VCS MX/VCS MXi User Guide [S]. Version E－2011.03－SP1 August 2011.

[5] 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.

[6] 王　纪，冯志华. SOC 多语言协同验证平台技术研究 [J]. 电子设计工程，2015，23 (20)：130－133.

[7] Synopsys. Reference Verification Methodology User Guide [S]. September，2004.

[8] Synopsys. Design Ware AHB Verification IP Databook [S]. March 24，2003.

[9] 王俊蕊，李艳斌. 短波多音并行信号全数字解调算法设计 [J]. 无线电工程，2016 (1)：76－79.

[10] 鲍晓利，冯永新. 宽带扫频式干扰的仿真与 FPGA 实现 [J]. 沈阳理工大学学报，2008，27 (3)：61－64.

[11] Schutten R. 基于 ESL 并采用 SystemC 和 SystemVerilog 的设计流程 [J]. 电子设计技术，2006，13 (4)：142－142.

(上接第 263 页)

[9] 左　力. 航空装备维修保障模式研究 [J]. 电子设计工程，2017，25 (3)：1－4.

[10] 袁　辉，王　夷，郑震山. 航空装备 MTBF 动态评估与预测模型 [J]. 机械设计与制造，2015 (4)：31－34.

[11] 杨　召，肖明清，胡　斌. 测试信息不完备条件下航空电子装备故障诊断 [J]. 计算机应用，2013，33 (S1)：46－47.

[12] 罗晓军，文　莹，方甲永. 复杂航空电子装备故障诊断规则提取 [J]. 计算机测量与控制，2012，20 (2)：294－296.

[13] 程进军，夏智勋，胡雷刚. 基于遗传神经网络的航空装备故障预测 [J]. 空军工程大学学报 (自然科学版)，2011，12 (1)：15－19.

[14] 郭双民. 基于决策树的航空装备维修级别 [J]. 四川兵工学报，2008 (4)：60－61.