中国集成电路 CIC
China Integrated Circuit

UVM和 Matlab 的联合仿真方法及应用

张少真,成丹,刘学毅
(航天恒星科技有限公司 集成电路设计中心,北京,100086)

摘要:本文以 LDPC(Low Density Parity Check,低密度奇偶校验)译码模块为例,详细介绍了通过调用 Matlab 引擎方法来实现 UVM(Universal Verification Methodology,通用验证方法学)和 Matlab 联合仿真的应用过程。文章提出的 UVM 和 Matlab 联合仿真方法,使得 Matlab 编程实现的算法模型可以应用到 UVM 验证平台中,缩短验证平台的搭建时间,有效地提高了仿真验证效率。

关键词:UVM 验证;Matlab 引擎;仿真;参考模型

The Method and Application of Simulation Base On UVM and Matlab

ZHANG Shao-zhen, CHENG Dan, LIU Xue-yi
(Space Star Technology CO., LTD., Beijing 100086, China)

Abstract: This paper introduces the verification platform based on UVM (Universal Verification Methodology) and Matlab of LDPC (Low Density Parity Check) module in detail.The simulation method can import matlab into the UVM verification platform to save time and to improve efficiency of setup verification environment.

Key words: UVM verification; Matlab Engine; Simulation; Reference Model

引言

随着集成电路制造工艺技术水平不断进步,芯片设计规模越来越大,验证所花费的时间占据了整个产品设计周期的 70%,而且随着设计复杂度的提高呈指数增加[1]。因此,对芯片验证工作提出了挑战:一是提高验证效率,缩短芯片研发周期,抢占市场;二是验证完备,搭建高质量的验证平台,避免失误。

验证方法学的发展有效地提高了验证效率，并使验证完备成为可能[2]。在 SystemVerilog 验证方法学中，最常用的主要有三种：VMM（Verification Methodology Manual）、OVM（Open Verification Methodology）和 UVM。其中，UVM 验证方法学使用更普遍、功能更强大：它提供了一系列标准类，可以实现验证环境的重用和自动化。在信号处理类芯片的验证过程中，使用硬件语言编写算法的参考模型成为了搭建验证平台的主要工作，但往往需要耗费大量的精力和时间。而使用 Matlab 语言可以快速实现算法，节省大量的时间。所以，如何将 Matlab 嵌入到 UVM 验证平台中成为了关键问题。

文中提出了基于 UVM 和 Matlab 的联合仿真方法，将 Matlab 算法融入到 UVM 验证平台中，来完成芯片的功能验证，并进行实例说明。

1 UVM 验证方法学

UVM 是基于 SystemVerilog 语言的验证方法学，提供由很多封装性好、功能完整的验证组件，每个验证组件遵循一致的架构和具有一套完整的元素组成。如图 1 所示，UVM 验证平台的验证组件一般包括输入组件（In_agent）、输出组件（Out_agent）、参考模型（Reference Model）和记分板（Scoreboard），而环境（env）组件将这些组件之间以及和被测设计对象（DUT，Design Under Test）进行连接，可以实现产生激励、输入激励、监测输入输出、进行结果比对和结果统计等功能。其各组件的功能简述如下：

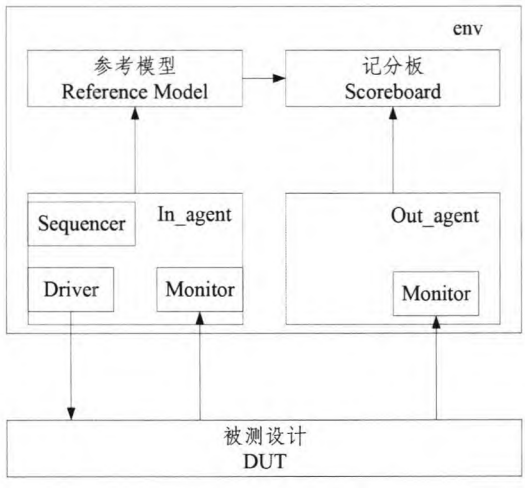

图 1 UVM 验证平台

1. In_agent 封装了 Sequencer（序列发生器）、Driver（驱动器）和 Monitor（监视器）三个基本组件，用于向 DUT 发送数据。其中，Sequencer 为序列发生器，是高性能的激励发生器；Driver 为驱动器，将产生的激励按照时序要求打入 DUT；Monitor 是监视器，用于采样 DUT 的输入信号。

2. Out_agent 中只有 Monitor 一个基本组件，用于监测 / 采集 DUT 的输出，输出的数据用于比对结果和覆盖率统计。

3. Reference Model 为参考模型。参考模型是模拟 DUT 的功能，使用同样的输入数据，产生数据，用于与 DUT 的输出进行比对。通常，参考模型由验证人员使用硬件语言（如 SystemVerilog）来实现，耗费大量的时间。由于 C、Matlab 等高级语言更容易进行参考模型的开发，因此可以将这些参考模型引入到 UVM 验证平台中，缩短参考模型的开发周期。

4. Scoreboard 为记分板，用于对比参考模型和 DUT 的输出是否一致，判断测试结果是否符合要求。

5. env 是环境组件，环境是验证组件的顶层调用组件。环境可以包含一个或多个的输入、输出组件以及其他组件。环境具有可配置性，允许对它的拓扑结构和行为进行自定义的设定，使得环境组件可以复用。

2 UVM 与 Matlab 的联合仿真方法

Matlab 是一款功能强、使用灵活的数值计算软件。它不但提供了矩阵处理、数值计算、图画显示等功能，还带有功能丰富的数学函数库。应用 Matlab 软件，用户可以很方便的实现许多复杂数学算法，而这些算法在其它开发环境中却需要大量代码才能实现，有时甚至是难以实现的[3]。而且，Matlab 还提供了应用程序接口，可以实现 Matlab 与外部环境的交

互，其中 Matlab 与 C/C++ 的接口通信应用最为广泛。

UVM 验证平台大多采用 SystemVerilog 语言来实现，SystemVerilog 语言是一种工业标准硬件设计和验证语言，能够把 RTL 设计、测试平台、断言和覆盖率全面综合在一起。但是在开发 UVM 验证平台中的参考模型时，对于需要进行大量数据运算的模型，比如编码或译码，采用 SystemVerilog 语言实现无异于重新实现了一次 RTL（Register Transfer Level，寄存器传输级）设计，从而浪费了大量时间。不过，由于 SystemVerilog 语言中提供了 DPI（Direct Programming Interface，直接编程接口）接口，也可以实现与高级语言 C/C++ 的通信。

所以，以 C/C++ 语言为桥梁，将 Matlab 强大的数值处理功能结合 UVM 验证方法学，即可实现 UVM 与 Matlab 的联合仿真。经过分析，UVM 和 Matlab 的联合仿真，有两种实现方法：一是 Matlab 转换为 C 代码，二是 C/C++ 调用 Matlab 引擎。下文将简单介绍这两种方法的实现过程。

2.1 Matlab 转换

Matlab 转换为 C 代码，是指使用 Matlab 软件中的 Matlab Coder 工具，将 Matlab 函数直接转换为 C 代码，可以直接用于 C 环境中。

使用 Matlab Coder 工具产生 C/C++ 代码共需要三个步骤：（1）准备用于产生 C 代码的 Matlab 算法；（2）检查 Matlab 代码的兼容性，有些 Matlab 语句不能转换为 C；（3）使用 codegen 命令或者 GUI 界面生成最终使用的源文件.c 和.h 及.a 静态库文件。通过 Matlab 转换为 C 的方法，实现 UVM 和 Matlab 的联合仿真流程如图 2 所示。

其方法是，首先使用 C 函数来调用 Matlab Coder 生成的源代码，再通过 SystemVerilog 的 DPI 接口将 C 函数引入到 UVM 验证平台中，实现 UVM 和 Matlab 的联合仿真，节省了开发参考模型的时间。但是这种方法仍存在局限性，要求 Matlab 工程中所有代码都具有良好的兼容性，所生成的 C/C++ 源代码的可读性不好。

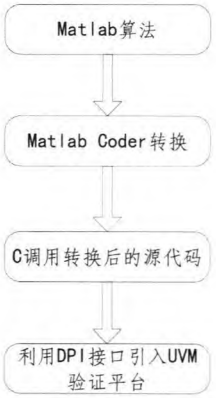

图 2 UVM 和 Matlab 联合仿真流程 1

2.2 Matlab 引擎

Matlab 引擎函数库是 Matlab 自带的一系列程序集。它允许用户对其中的 Matlab 函数进行调用，将 Matlab 作为一个计算引擎在后台运行，以完成复杂的运算。C/C++ 调用 Matlab 引擎的基本过程简述如下：

C/C++调用 Matlab 引擎的机制。它向 Matlab 引擎传递命令和数据信息，并从 Matlab 引擎接收数据信息，而 Matlab 在后台进行计算和数据处理，从而缩短了开发时间。

C/C++调用 Matlab 引擎的操作过程。它先定义 Matlab 引擎，engOpen（）函数打开引擎，将 mxArray 数组类型的输入变量放入引擎中，使用 engEvalString（）函数向 Matlab 中传递执行命令，Matlab 在后台运行计算，再将计算结果输出，释放内存、关闭引擎。通过 C/C++ 调用 Matlab 引擎的方法，实现 UVM和 Matlab 的联合仿真流程如图 3 所示。

从中看出，Matlab 引擎不仅可以调用 Matlab 中的 C/C++ 函数，还可以调用工具箱中的函数，应用程序整体性能较好[4]。调用 Matlab 引擎的方法相比 Matlab 转换为 C 的方法，有很多优点，省去了 Matlab Coder 转换的过程；Matlab 运行过程在后台执行，只需链接一部分引擎函数，容易掌握和实现；节省大量的系统资源，整体性能好等等。因此，在实际芯片项目开发过程中，调用 Matlab 引擎的方法更为方便，

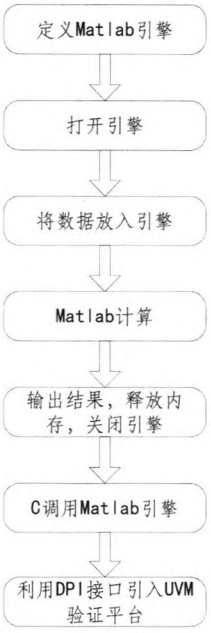

图 3 UVM 和 Matlab 联合仿真流程 2

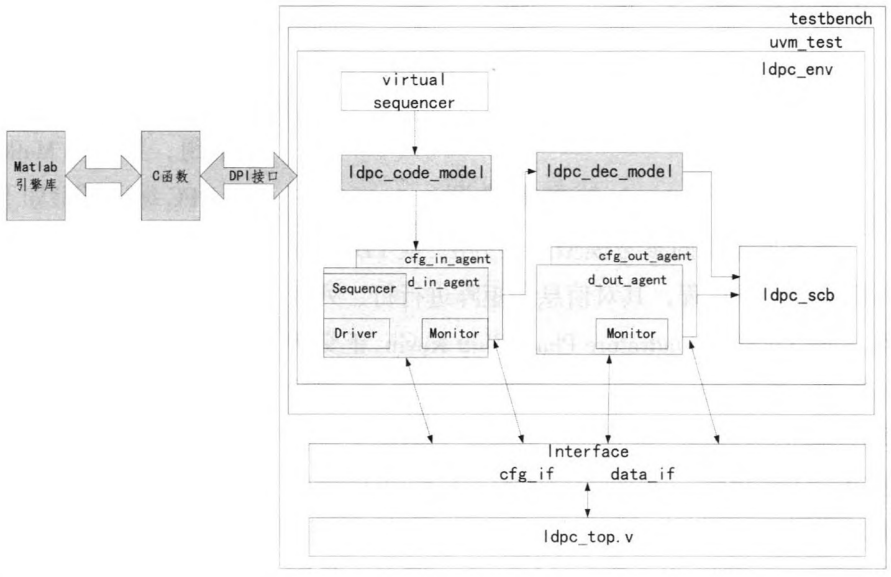

图 4 UVM 和 Matlab 的联合仿真平台

下面详细介绍该种方法在调制解调芯片中的应用过程。

3 基于 UVM 和 Matlab 的联合仿真实例

以调制解调芯片中的 LDPC 译码模块为例，搭建基于 UVM 和 Matlab 的联合仿真平台，开展 LDPC 译码模块的验证工作。

3.1 基于 UVM 的验证平台

LDPC 译码模块的验证平台，如图 4 所示，其中的 ldpc_code_model 编码模型和 ldpc_dec_model 译码模型都是使用 Matlab 函数实现的，以 C 语言为桥梁，采用上文提到的调用 Matlab 引擎的方法，通过 DPI 接口将 Matlab 程序嵌入到 UVM 验证平台中，实现 UVM 和 Matlab 的联合仿真。

如图 4 所示，LDPC 验证平台中共有两路独立的验证组件，分别为配置（config）通路和数据（data）通路；其中，配置通路用于配置 LDPC 的码率和类型，数据通路用于将配置后的 LDPC 码进行验证并比对输出结果。

该联合仿真平台的特点：（1）每路验证组件中都包括各自的验证组件，有 packet、sequencer、driver、monitor、input_agent、interface 等；（2）两路共用的验证组件有 virtual sequencer、model、scb、env 等。（3）所有的验证组件都继承于 uvm 的类，比如 ldpc_env 组件，将验证平台中的所有组件进行例化，并连接所有的组件。

```
class ldpc_env extends uvm_env;                    //ldpc_env 派生自 uvm_env 类
    data_input_agent    d_in_agent;     // 实例化 data_input_agent 组件
    data_output_agent   d_out_agent;    // 实例化 data_output_agent 组件
    config_input_agent  cfg_in_agent;   // 实例化 config_input_agent 组件
    config_output_agent cfg_out_agent;  // 实例化 config_output_agent 组件
    ldpc_model          ldpc_dec_model;// 实例化 ldpc_model 组件
    ldpc_scoreboard     ldpc_scb;       // 实例化 ldpc_scoreboard 组件
```

```
……                                    // 将实例化的所有组件进行连接
….
endclass
```

3.2 基于 Matlab 的参考模型

基于 Matlab 的参考模型应用方法：就 LDPC 编码模型的计算过程，其对信息码矩阵进行行、列变换，经过 QPSK（Quadrature Phase Shift Keyin，正交相移键控）调制，再对调制后的码进行噪声处理，最后对信息码进行量化和归一化；Matlab 引擎库中存在相应的功能函数，比如矩阵函数 reshape 可以进行矩阵变换，噪声函数 awgn 可以实现加噪过程，归一化函数 floor 可以进行量化和归一化。所以，使用 Matlab 来实现编码模型比 SystemVerilog 实现更加方便。使用 Matlab 语言实现 LDPC 编码模型 ldpc_code.m 的程序如下：

```
%%Matlab function
 %% function [code_out_m] = ldpc_code (code_in_m, len, snr)
qpsk_table = [0.7071+0.7071i 0.7071-0.7071i -0.7071+0.7071i -0.7071-0.7071i];
rshp_bits = reshape(code_in_m,2,[]);      %%调用 reshape 函数进行行列变换
msg_tx = qpsk_table([2 1]*rshp_bits+1);   %% qpsk 调制过程
rz = awgn (msg_tx,snr,'measured')         %% 调用 awgn 函数进行加噪处理
……
code_out_m = floor(code_out_m/max*2^5); %% 调用 floor 函数，对结果进行量化
%%end Matlab function
```

在调用该参考模型函数时，根据输入的随机信息码 code_in_m，信息码长度 len 及噪声系数 snr，就可以计算产生不同码率的 LDPC 码。同理，LDPC 的译码模型也可以使用 Matlab 引擎库中的函数来完成。

综上，我们采用 Matlab 引擎库中的函数来编写完成 ldpc_code.m 程序和 ldpc_dec.m 程序，实现 LDPC 编码和译码的过程，并封装成 function。编写调用 Matlab 引擎的 C 函数，声明变量及引擎指针、数组，并调用 Matlab 引擎，进行运算后输出结果。LDPC 编码模型如下：

```
#include <stdlib.h>   // 调用基本函数库
#include <stdio.h>    // 调用基本接口库
#include <math.h>     // 调用数学库
#include "engine.h"   // 调用引擎库，加载 engine 函数库
void ldpc_code (double *din ,double len,double *dout) //LDPC 编码模型函数
{
  Engine *ep;                          // 声明引擎指针
  double *code_in = NULL;              // 声明变量
  ……
  mxArray *code_in_mx = NULL;          // 声明 mx 数组
  ……
  //input，将输入变量转换到声明的 mx 数组中，便于输入到引擎
  code_in_mx = mxCreateDoubleMatrix (1,len, mxREAL);
  code_in = mxGetPr(code_in_mx);
  ……
  //put to engin，将输入变量数组的值输入到引擎中
  ep=engOpen("matlab -nosplash");
  engPutVariable (ep,"code_in_m", code_in_mx);
  ……
  // 调用自定义的 Matlab 函数参考模型
engEvalString (ep," path ('/home/model/ldpc_code', path);");
……
//output，将引擎计算后的结果输出
```

```
code_out_mx = engGetVariable( ep," code_out_m" )
//close engine,关闭引擎,释放内存
mxDestoryArray( code_in_mx );
……
    engClose( ep );
}
```

同理,LDPC 的译码模型如下:

```
#include <stdlib.h>                    // 调用基本函数库
……
void ldpc_dec  ( double *din ,double len,double *dout ) //LDPC 译码模型函数
{
  Engine *ep;                          // 声明引擎指针
  ……
  engClose( ep );                      // 关闭引擎

}
```

3.3 UVM 与 Matlab 的联合仿真

基于 UVM 的验证平台和 Matlab 的参考模型,采用 C 语言将编写完成的 LDPC 编码模型 ldpc_code 封装为 ldpc_code_top 函数,而译码模型封装为 ldpc_dec_top 函数,通过 DPI 接口将 C 语言封装的函数引入到 UVM 验证平台中。

(1)LDPC 编码模块。比如,封装 ldpc_code_top 函数的过程如下:

```
#include <svdpi.h>     // 调用 svdpi 函数库
void ldpc_code_top ( svOpenArrayHandle data_in, double len, svOpenArrayHandle data_out )
// 使用 svOpenArrayHandle 句柄,来传递 SystemVerilog 与 C 之间的数组
{
  double * din;                        // 声明指针
  double * dout;                       // 声明指针
  din = ( double * ) svGetArrayPtr( data_in ); // 传递数据
  dout = ( double * ) svGetArrayPtr ( data_out );
// 传递数据
  ldpc_code ( din,len,dout );          // 调用使用 Matlab 引擎实现的参考模型
}
```

基此,将 ldpc_code_top 函数作为 LDPC 编码模型,引入到 UVM 验证平台的 ldpc_data_packet 组件中。另外,在 LDPC 译码模块的仿真验证平台中,只需要配置 LDPC 码的类型和码率,就可以随机产生相应码率的 LDPC 码,如表 1 所示。从中看出,如欲生成码率为 2/5 的 Normal 类型的 LDPC 码,则需要配置类型序号为 1,码率序号为 3,即可得到长度为 64800bit 的 LDPC 码。

表 1 LDPC 码类型及长度

类型序号	类型	LDPC 码率	LDPC 码长(bit)
1	Normal	1/4, 1/3, 2/5, 1/2, 3/5, 2/3, 3/4, 4/5, 5/6, 8/9, 9/10	64800
2	Short	1/4, 1/3, 2/5, 1/2, 3/5, 2/3, 3/4, 4/5, 5/6, 8/9	16200
3	Normal (s2x)	2/9	64800
4	Short (s2x)	1/5, 4/15, 1/3	16200
5	Medium (s2x)	1/5, 11/45, 1/3	32400
6	Short (SF2) (s2x)	1/5, 11/45	16200

(2)LDPC 译码模块。在 LDPC 译码模块的 UVM 与 Matlab 联合仿真平台中,ldpc_data_packet 组件的实现代码如下:

```
import "DPI-C" function void ldpc_code_top ( input real ldpc_in [], input real bit_len, output real ldpc_out[] );// 使用 DPI 接口引入参考模型函数
class ldpc_data_packet extends uvm_sequence_item;// 派生自 uvm_sequence_item 类
  rand bit ldpc_info[];                // 定义随机的信息码数组
  rand int code_type;                  // 定义 LDPC 码类型,范围为 1 至 6
  rand int rate_type;                  // 定义 LDPC 码率类型,范围为 1 至 11
  rand bit ldpc_data[];                // 定义 ldpc_data 数组,存放 LDPC 码
  ……
```

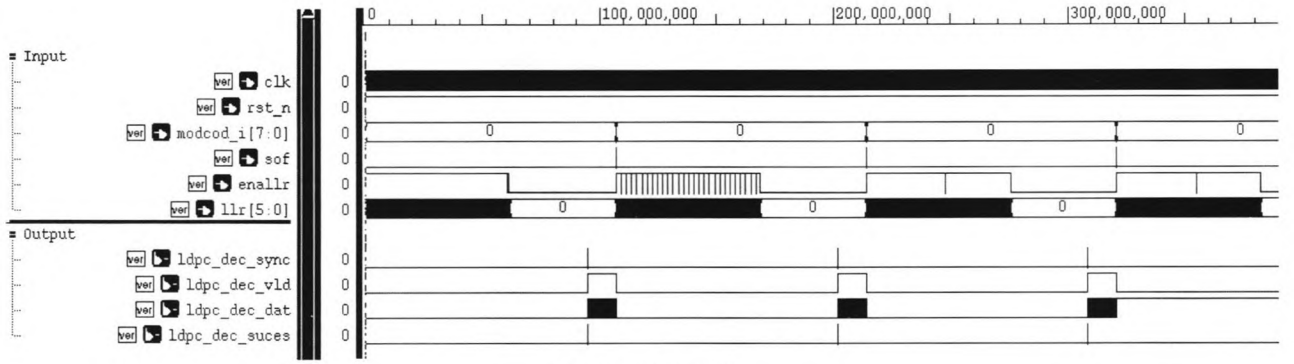

图 5 仿真运行结果

```
    function void post_randomize( );          // 对随机信息码进行编码
        // 计算 LDPC 码长
        // 调用 LDPC 编码模型
        ldpc_code_top( ldpc_info,len,ldpc_data );
        // 将结果输出到数组中
    endfunction
endclass
```

同理,将 LDPC 译码模型引入到 UVM 验证平台的 ldpc_dec_model 中,如下:

```
import "DPI-C" function void ldpc_dec_top ( input real ldpc_data[],…… );
// 使用 DPI 接口引入参考模型函数
class ldpc_dec_model extends uvm_component; // 派生自 uvm_component 类
    ……
    task ldpc_dec_model::main_phase ( uvm_phase phase ); // 译码处理过程
        ……
            // 调用 LDPC 译码参考模型函数
        ldpc_dec_top( );
        ……
    endtask
endclass
```

3.4 仿真运行结果

在仿真运行过程中,需要将 Matlab 库的路径加入到编译参数中,比如,-rpath= /usr/nfsbin/MATLAB/R2012b/bin/glnxa64,进行编译、仿真。该测试用例中,其仿真运行过程是:(1)先配置 LDPC 码的类型序号即 modcod_i 信号,ldpc_data_packet 按照相应类型和码率产生 LDPC 码;(2)LDPC 码通过 ldpc_dirver 组件,生成相应时序的 llr 信号、使能 enallr 信号和 sof 信号,同时打入到 ldpc_dec_model 参考模型中;(3) 通过 ldpc_monitor 采集输出信号 ldpc_dec_dat 得到译码之后的输出结果;(4)ldpc_dec_model 和 ldpc_monitor 将结果输入到 ldcp_scb 中,完成比对过程,比对成功后可以看到 ldpc_dec_suces 的脉冲信号。图 5 显示了仿真运行的结果,UVM 和 Matlab 的联合仿真平台能够正确的产生相应码率的 LDPC 码,并将这些激励输入的 DUT 和 ReferenceModel 中,最终通过 Scoreboard 组件,给出仿真运行的比对结果。

4 结论

文中以 LDPC 译码模块验证平台为例,通过调用 Matlab 引擎的方法开发参考模型,实现了 UVM 和 Matlab 的联合仿真。从仿真结果看,UVM 和 Matlab 的联合仿真方法能够满足芯片研发过程中功能验证的要求,并且参考模型的实现过程方便、简单。从 LDPC 译码模块参考模型的实现代码量来看,使用 Matlab 引擎方法实现参考模型,代码量仅仅为 192 行;而使用 SysemVerilog 语言开发 LDPC 编码和译码参考模型,代码量至少为 1000 行。从而可见,UVM 和 Matlab 联合仿真的方法,既能够满足基于

UVM验证平台的功能验证需求，又降低了参考模型的开发难度，缩短了开发参考模型的时间，提高了芯片验证效率。CIC

参考文献

[1] 山蕊，蒋林，李涛，基于 system verilog 的可重用验证平台，电子技术应用，2013，39(5)：128-131.
[2] 徐金甫，李森森，采用 UVM 方法学实现验证的可重用与自动化，微电子学与计算机，2014，31(11)：14-17.
[3] 李迎，王帮峰，孙亚飞等，MATLAB 与 VC++ 接口通信，计算机测量与控制，2004.12(6)：587-590.
[4] 姜浩智，廖宁华，VC++ 与 MATLAB 混合编程的实现方法，中国高新技术企业，2010，9，13-15.

作者简介

张少真，硕士，芯片验证工程师，主要研究方向为集成电路验证。
成丹，硕士，芯片验证工程师，主要研究方向为集成电路验证。
刘学毅，硕士，芯片设计工程师，主要研究方向为集成电路设计。

应特格与福建博纯材料新工厂开幕，推动中国电子特殊气体制造国产化进程

全球半导体气体材料最大供应商美国应特格有限公司（Entegris）与国内半导体材料高新技术企业福建博纯材料有限公司共同宣布，双方迎来合作的第一个里程碑。日前，应特格的特殊气体已在福建博纯公司位于泉州永春的工厂顺利投产。双方在泉州永春的新化学工厂举办了盛大的开幕式，Entegris 首席运营官 Todd Edlund 先生、企业市场副总裁杨文革先生、博纯材料创始人、董事会主席兼首席执行官陈国富先生及 Entegris 的客户、中国半导体行业协会信息交流部主任任振川，重要地方政府相关领导及媒体等 100 余人参加了此次活动。

通过此次合作，Entegris 将继续加强中国业务。Entegris 是首家本地化生产特殊气体产品的国际材料公司，旨在为中国日益增长的半导体市场提供支持，截至开幕日，从该工厂制造的首批合格气体产品正在运往当地客户途中。

Entegris 首席运营官 Todd Edlund 在盛大开幕活动上表示："此次合作是我们持续计划的一部分，通过本地化生产和技术支持更好地服务于中国客户，满足他们对于制造和工艺开发的需求"。"我们与博纯材料合作，为我们的特殊化学品在中国生产建造出了世界一流的工厂。"

Entegris 企业市场副总裁杨文革表示："外资的引入在一定程度上推进了国内材料和设备的国产化进程。对于此次合作，应特格表示希望帮助推动中国电子特殊气体制造的国产化，未来，Entergris 所有的技术都会转移至泉州的新工厂，这在一定程度上推进了国内材料和设备的国产化进程。"

为此次活动揭幕的博纯材料创始人、董事会主席兼首席执行官陈国富表示："我们与 Entegris 团队密切合作，完成了此次扩建并开始制造运营。我可以自豪地说，所有期望和期限都已得到满足，商业化顺利实现。"

总体上，博纯材料泉州工厂占地 55000 m^2（13.6 亩），包括两个 2.3 级危险品库房以及 20 多个电子气体包装、纯化和合成工艺中心。该工厂可在国内制造世界上最先进的半导体离子注入材料，同时创造了当地就业机会。该工厂位于中国福建省泉州市永春县大荣村。（来自 Entegris）