VC Verification IP
UART
UVM Getting Started Guide

Version O-2018.09, September 2018

SYNOPSYS®

Copyright Notice and Proprietary Information

Destination Control Statement

All technical data contained in this publication is subject to the export control laws of the United States of America. Disclosure to nationals of other countries contrary to United States law is prohibited. It is the reader's responsibility to determine the applicable regulations and to comply with them.

Disclaimer

SYNOPSYS, INC., AND ITS LICENSORS MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Trademarks

Synopsys and certain Synopsys product names are trademarks of Synopsys, as set forth at https://www.synopsys.com/company/legal/trademarks-brands.html
All other product or company names may be trademarks of their respective owners.

Free and Open-Source Software Licensing Notices

If applicable, Free and Open-Source Software (FOSS) licensing notices are available in the product installation.

Third-Party Links

Any links to third-party websites included in this document are for your convenience only. Synopsys does not endorse and is not responsible for such websites and their practices, including privacy practices, availability, and content.

Synopsys, Inc.
690 E. Middlefield Road
Mountain View, CA 94043
www.synopsys.com

Contents

Preface

About This Document

This Getting Started Guide presents information about integrating the VC VIP for UART (referred to as VIP) into testbenches that are compliant with the SystemVerilog Universal Verification Methodology (UVM). You are assumed to be familiar with the UART protocol and UVM.

Web Resources

- Documentation through SolvNet: https://solvnet.synopsys.com (Synopsys password required)
- Synopsys Common Licensing (SCL): http://www.synopsys.com/keys

Customer Support

To obtain support for your product, choose one of the following:

- Enter a call through SolvNet.
  - Go to http://solvnet.synopsys.com/EnterACall
    and provide the requested information, including:
    - Product: **Verification IP**
    - Sub Product: **UART SVT**
    - Tool Version: **O-2018.09**
    - Fill in the remaining fields according to your environment and your issue
    - For simulation issues, provide a UVM_FULL verbosity log file of the VIP instance and a VPD or FSDB dump file of the VIP interface.
- Send an e-mail message to support_center@synopsys.com.
  - Include the Product name, Sub Product name, and Tool Version (as noted above) in your e-mail so it can be routed correctly.
- Telephone your local support center.
  - North America:

    Call 1-800-245-8005 from 7 AM to 5:30 PM Pacific time, Monday through Friday.
  - All other countries:

    http://www.synopsys.com/Support/GlobalSupportCenters

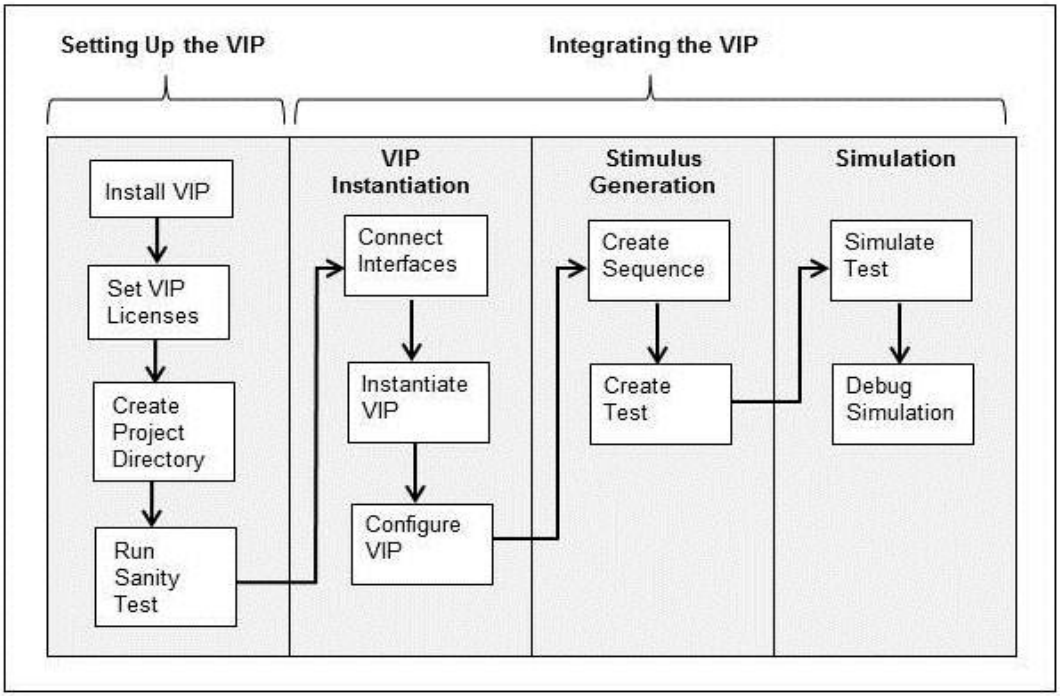
1 Overview of the Getting Started Guide

This Getting Started Guide presents information about integrating the VC VIP for UART (referred to as VIP) into testbenches that are compliant with the SystemVerilog Universal Verification Methodology (UVM). Figure 1-1 is the VIP integration and test work flow presented in this document. The steps for setting up the VIP are documented in the *VC Verification IP UVM Installation and Setup Guide*. This guide is available on the SolvNet Download Center (click here -> VC VIP Library -> O-2018.09 -> Installation Guide) and in the VIP installation at the following location:

```
$DESIGNWARE_HOME/vip/svt/common/latest/doc/uvm_install.pdf
```

The VIP setup should be completed before executing the steps in this document.

Figure 1-1 VIP Integration and Test Work Flow



You are assumed to be familiar with the UART protocol and UVM. For more information on the VIP, see the *VC Verification IP UART UVM User Guide* on SolvNet (click here) and in the VIP installation at the following location:

```
$DESIGNWARE_HOME/vip/svt/uart_svt/latest/doc/uart_svt_uvm_user_guide.pdf
```

2 Integrating the VIP into a User Testbench

The VC VIP for UART provides a suite of advanced SystemVerilog verification components and data objects that are compliant to UVM. Integrating these components and objects into any UVM compliant testbench is straightforward. For a complete list of VIP components and data objects, refer to the main page of the *VC VIP UART Class Reference* (only in HTML format) at the following location:

```
$DESIGNWARE_HOME/vip/svt/uart_svt/latest/doc/uart_svt_uvm_class_reference/html/
index.html
```

2.1 VIP Testbench Integration Flow

The UART agent (svt_uart_agent) is the top-level component provided by the VIP for modeling Data Terminal Equipment (DTE) and Data Communication Equipment (DCE). This generic agent encapsulates the following components:

- Sequencer
- Monitor
- Driver

You can instantiate and construct the UART agent in the top-level environment of your UVM testbench.

Figure 2-1 Top-level Architecture of an UART VIP Testbench

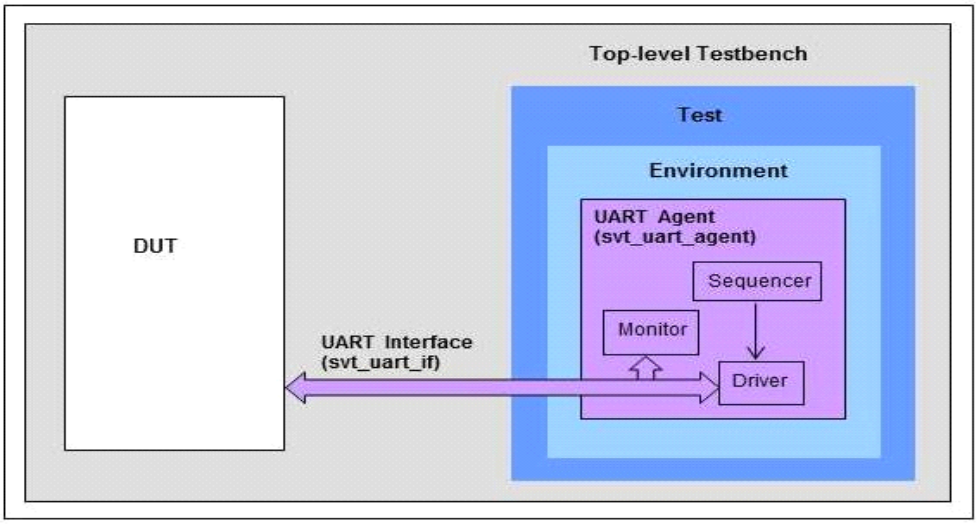


Figure 2-1 is a top-level architecture of a simple VC VIP for UART testbench. The steps for integrating the VIP into a UVM testbench are described in the following sections:

- "Connecting the VIP to the DUT"
- "Instantiating and Configuring the VIP"
- "Creating a Test Sequence"
- "Creating a Test"

The code snippets presented in this chapter are generic and can be applied to any UVM compliant testbench. For more information on the code usage, refer to the following example:

```
$DESIGNWARE_HOME/vip/svt/uart_svt/latest/examples/sverilog/
tb_uart_svt_uvm_basic_sys
```

2.1.1 Connecting the VIP to the DUT

The following are the steps to establish a connection between the VIP to the DUT in your top-level testbench:

- Include the standard UVM and VIP files and packages.

```
`include "uvm_pkg.sv"
`include "svt_uart.uvm.pkg"

import uvm_pkg::*;
import svt_uvm_pkg::*;
import svt_uart_uvm_pkg::*;
```

- Instantiate and connect the top-level UART interface (svt_uart_if) to a system clock.

```
svt_uart_if uart_dte_if(SystemClock);
svt_uart_if uart_dce_if(SystemClock);
```

- Instantiate a UART BFM wrapper that serves as a DTE or DCE

```
svt_uart_bfm_wrapper #(.UV_DEVICE_TYPE(`UV_DTE)) Bfm0 (uart_dte_if);
svt_uart_bfm_wrapper #(.UV_DEVICE_TYPE(`UV_DCE)) Bfm1 (uart_dce_if);
```

Note

Specifying the above BFM wrapper instantiations in the top-level testbench file are mandatory. These statements must be specified to ensure proper VIP operations.

- Connect the top-level UART interface to the DUT and VIP.

  DTE:

```
dut dut_inst(uart_dte_if);

uvm_config_db#(virtual svt_uart_if)::set(uvm_root::get(),
     "uvm_test_top.env", "dte_vif", uart_dte_if);
```

DCE:

```
dut dut_inst(uart_dce_if);

uvm_config_db#(virtual svt_uart_if)::set(uvm_root::get(),
     "uvm_test_top.env", "dce_vif", uart_dce_if);
```

The `uvm_config_db` command connects the top-level UART interface to the virtual interface of the UART agent. The "`uvm_test_top`" represents the top-level module in the UVM environment. The "`env`" is an instance of your testbench environment.

- Connect the reset pin of the UART agent interface to the desired reset signal.

```
assign uart_dte_if.reset = <User-select Reset Signal>;
assign uart_dce_if.reset = <User-select Reset Signal>;
```

Note
The reset pin of the UART agent interface should be driven by toggled reset to allow proper VIP internal initialization.

2.1.2 Instantiating and Configuring the VIP

The following are steps to instantiate and configure the UART agent (svt_uart_agent) in your testbench environment.

- Instantiate the UART agent in the build phase of your testbench environment.

```
svt_uart_agent dte_agent;
svt_uart_agent dce_agent;

dte_agent = svt_uart_agent::type_id::create("dte_agent",this);
dce_agent = svt_uart_agent::type_id::create("dce_agent",this);
```

- Create a customized UART agent configuration class by extending the UART agent configuration class (svt_uart_agent_configuration) and specifies the required configuration parameters.

  For example:

```
class cust_svt_uart_agent_configuration extends
     svt_uart_agent_configuration;

 //Class constructor
  function new(string name="cust_svt_uart_agent_configuration");
    super.new(name);

  //User constraint to override the value of baud_divisor
    constraint cust_reasonable_baud_divisor { baud_divisor == 1;}
  endfunction
endclass
```

For more information on the configuration class, refer to the *svt_uart_configuration* and *svt_uart_agent_configuration Class References* at the following locations:

```
$DESIGNWARE_HOME/vip/svt/uart_svt/latest/doc/
uart_svt_uvm_class_reference/html/class_svt_uart_configuration.html

$DESIGNWARE_HOME/vip/svt/uart_svt/latest/doc/
uart_svt_uvm_class_reference/html/class_svt_uart_agent_configuration
.html
```

- Construct the customized UART agent configuration and pass the configuration to the UART agent (instance of svt_uart_agent) in the build phase of your testbench environment.

```
dte_cfg = cust_svt_uart_agent_configuration::type_id::create("dte_cfg");

dce_cfg = cust_svt_uart_agent_configuration::type_id::create("dce_cfg");

uvm_config_db#(svt_uart_agent_configuration)::set(this, "dte_agent",
"cfg", dte_cfg);

uvm_config_db#(svt_uart_agent_configuration)::set(this, "dce_agent",
"cfg", dce_cfg);
```

The "`cust_svt_uart_agent_configuration`" is the customized UART agent configuration as defined in the previous step. The "dte_cfg" and "dce_cfg" are instances of this configuration.

2.1.3 Creating a Test Sequence

The VIP provides a base sequence class for the UART DTE agent (svt_uart_dte_base_sequence) and the UART DCE agent (svt_uart_dce_base_sequence). You can extend these base sequences to create test sequences for the UART DTE and DCE agents.

For more information on the UART DTE and DCE base sequences, and the VIP sequence collection, refer to the Sequence Page of the *VC VIP UART Class Reference* at the following location:

```
$DESIGNWARE_HOME/vip/svt/uart_svt/latest/doc/
uart_svt_uvm_class_reference/html/sequencepages.html
```

In addition, a list of random and directed sequences are available in the VIP examples. For more information on the example sequences, refer to the example directories at the following location:

```
$DESIGNWARE_HOME/vip/svt/uart_svt/latest/examples/sverilog
```

2.1.4 Creating a Test

You can create a VIP test by extending the uvm_test class. In the build phase of the extended class, you construct the testbench environment and set the respective UART DTE and DCE sequences.

```
class uart_base_test extends uvm_test;

 //build_phase
  env = uart_basic_env::type_id::create("env", this);

  uvm_config_db#(uvm_object_wrapper)::set(this,
       "env.sequencer.main_phase", "default_sequence",
       uart_default_virtual_sequence::type_id::get());
```

2.2 Compiling and Simulating a Test with the VIP

The steps for compiling and simulating a test with the VIP are described in the following sections:

- "Directory Paths for VIP Compilation"
- "VIP Compile-Time Options"
- "VIP Runtime Option"

2.2.1 Directory Paths for VIP Compilation

You need to specify the following directory paths in the compilation commands for the compiler to load the VIP files.

```
+incdir+project_directory_path/include/sverilog
+incdir+project_directory_path/src/sverilog/simulator
```

Where, *project_directory_path* is your project directory and *simulator* is vcs, ncv or mti.

For example:

```
+incdir+/home/project1/testbench/vip/include/sverilog
+incdir+/home/project1/testbench/vip/src/sverilog/vcs
```

2.2.2 VIP Compile-Time Options

The following are the required compile-time options for compiling a testbench with the VC VIP for UART:

```
+define+SVT_UVM_TECHNOLOGY
+define+UVM_PACKER_MAX_BYTES=8000
+define+UVM_DISABLE_AUTO_ITEM_RECORDING
+define+SYNOPSYS_SV
+define+SVT_UART
```

Note

UVM_PACKER_MAX_BYTES define needs to be set to maximum value as required by each VIP title in your testbench. For example, if VIP title 1 needs UVM_PACKER_MAX_BYTES to be set to 8192, and VIP title 2 needs UVM_PACKER_MAX_BYTES to be set to 500000, you need to set UVM_PACKER_MAX_BYTES to 500000.

Macro	Description
SVT_UVM_TECHNOLOGY	Specifies SystemVerilog based VIPs that are compliant with UVM
UVM_PACKER_MAX_BYTES	Sets to 8000 or greater
UVM_DISABLE_AUTO_ITEM_RECORDING	Disables the UVM automatic transaction begin and end event triggering and recording
SYNOPSYS_SV	Specifies SystemVerilog based VIPs that are compliant with UVM
SVT_UART	Specifies SystemVerilog based UART VIP implementation that is compliant with UVM

2.2.3 VIP Runtime Option

No VIP specific runtime option is required to run simulations with the VIP. Only relevant UVM runtime options are required.

For example:

```
+UVM_TESTNAME=directed_test
```

A Summary of Commands, Documents, and Examples

A.1 Commands in This Document

Display VIP models and examples under the VIP installation directory specified by $DESIGNWARE_HOME:

```
% $DESIGNWARE_HOME/bin/dw_vip_setup -info home
```

Add VIP models to the project directory:

```
% $DESIGNWARE_HOME/bin/dw_vip_setup -path project_directory -add
VIP_model -svlog
```

Add VIP examples to the directory where the command is executed:

```
% $DESIGNWARE_HOME/bin/dw_vip_setup -e VIP_example -svlog
```

A.2 Primary Documentation for VC VIP UART

VC Verification IP UVM Installation and Setup Guide:

```
$DESIGNWARE_HOME/vip/svt/common/latest/doc/uvm_install.pdf
```

VC VIP UART UVM User Guide:

```
$DESIGNWARE_HOME/vip/svt/uart_svt/latest/doc/uart_svt_uvm_user_guide.pdf
```

VC VIP UART Getting Started Guide:

```
$DESIGNWARE_HOME/vip/svt/uart_svt/latest/doc/uart_svt_uvm_getting_started.pdf
```

VC VIP UART QuickStart:

```
$DESIGNWARE_HOME/vip/svt/uart_svt/latest/examples/sverilog/
tb_uart_svt_uvm_basic_sys/doc/tb_uart_svt_uvm_basic_sys/index_basic.html
```

VC VIP UART Class Reference:

```
$DESIGNWARE_HOME/vip/svt/uart_svt/latest/doc/uart_svt_uvm_class_reference/html/
index.html
```

VC VIP UART Verification Plans:

```
$DESIGNWARE_HOME/vip/svt/uart_svt/latest/doc/VerificationPlans
```

A.3 Example Home Directory

Directory that contains a list of VIP example directories:

```
$DESIGNWARE_HOME/vip/svt/uart_svt/latest/examples/sverilog
```

View simulation options for each example:

```
gmake help
```