Verification Continuum™

VC Verification IP
I2C
UVM Getting Started Guide

Version Q-2020.06, June 2020

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 http://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.

www.synopsys.com

Contents

Preface

About This Document

This Getting Started Guide presents information about integrating the VC VIP for I2C (referred to as VIP) into testbenches that are compliant with the SystemVerilog Universal Verification Methodology (UVM). You are assumed to be familiar with the I2C protocol and UVM.

Web Resources

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

Customer Support

To register a problem, perform any of the following tasks:

1. Go to https://solvnetplus.synopsys.com and open a case.

   Enter the information according to your environment and your issue.

2. Send an e-mail message to support_center@synopsys.com
   - Include the Product name, Sub Product name, and Product version for which you want to register the problem.
3. 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

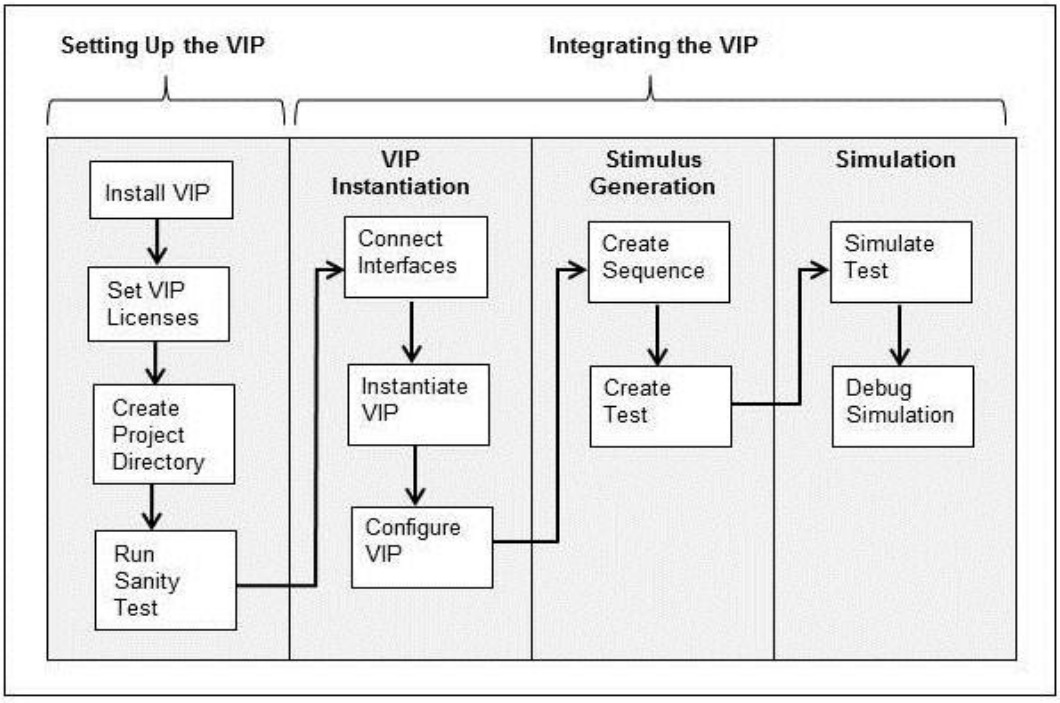
1 Overview of the Getting Started Guide

This Getting Started Guide presents information about integrating the VC VIP for I2C (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-2018.12-> 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 I2C protocol and UVM. For more information on the VIP, refer to the *VC Verification IP I2C UVM User Guide* on SolvNet (click here) or in the VIP installation at the following location:

```
$DESIGNWARE_HOME/vip/svt/i2c_svt/latest/doc/i2c_svt_uvm_user_guide.pdf
```

2 Integrating the VIP into a User Testbench

The VC VIP for I2C 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 I2C Class Reference* (only in HTML format) at the following location:

```
$DESIGNWARE_HOME/vip/svt/i2c_svt/latest/doc/i2c_svt_uvm_class_reference/html/
index.html
```

2.1 VIP Testbench Integration Flow

The I2C system environment (svt_i2c_system_env) is the top-level component provided by the VIP. This environment encapsulates all of the VIP components and implicitly constructs the required number of I2C master and I2C slave agents as specified by its system configuration object. You can instantiate and construct the I2C system environment in the top-level environment of your UVM testbench.

Figure 2-1 Top-level Architecture of an I2C VIP Testbench

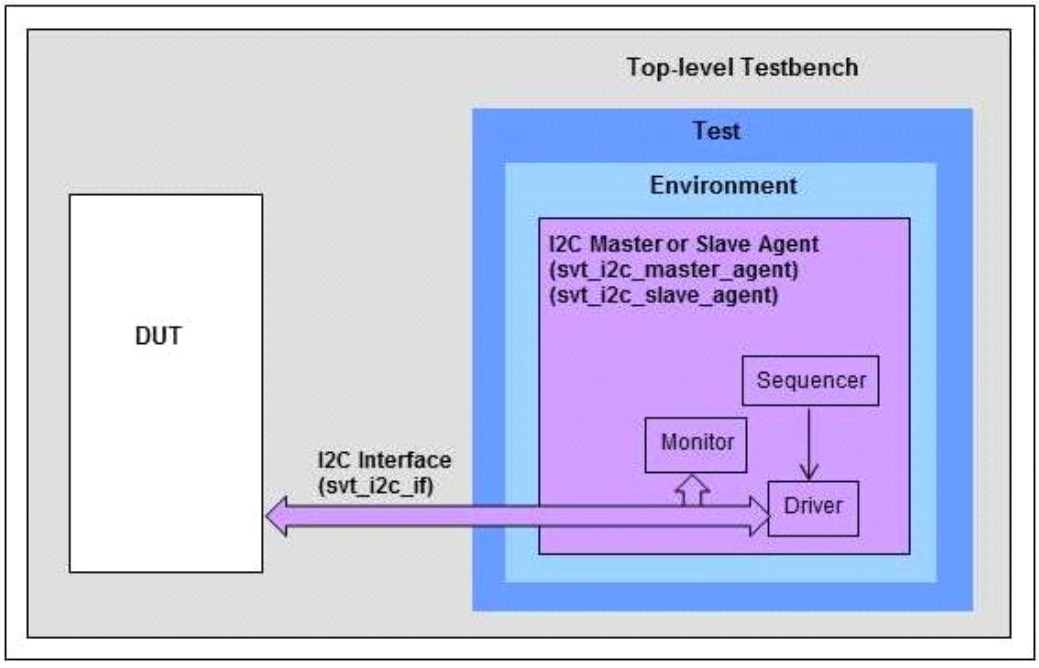


Figure 2-1 is a top-level architecture of a simple VC VIP for I2C testbench. The steps for integrating the VIP into a UVM testbench are described in the following sections:

- "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/i2c_svt/latest/examples/sverilog/
tb_i2c_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_i2c.uvm.pkg"
`include "svt_i2c_if.svi" //top-level I2C interface

import uvm_pkg::*;
import svt_uvm_pkg::*;
import svt_i2c_uvm_pkg::*;
import svt_i2c_enum_pkg::*;
```

- Instantiate and connect the top-level I2C interfaces (svt_i2c_if) to a system clock.

```
svt_i2c_if i2c_if(<System Clock>);
```

- Instantiate I2C master and slave BFM wrappers to act as master and slave.

```
//Intantiate master wrapper
svt_i2c_master_wrapper #(.I2C_AGENT_ID(0)) Master0 (i2c_if);
svt_i2c_master_wrapper #(.I2C_AGENT_ID(1)) Master1 (i2c_if);

//Intantiate slave wrapper
svt_i2c_slave_wrapper #(.I2C_AGENT_ID(0)) Slave0 (i2c_if);
svt_i2c_slave_wrapper #(.I2C_AGENT_ID(1)) Slave1 (i2c_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. If only a single master or slave is used, it is mandatory to instantiate at least one master and one slave wrapper, and specify the unused components as PASSIVE. If multiple components such as multiple masters or multiple slaves are created, then a unique agent ID must be specified in each wrapper module instantiation.

- Connect the top-level I2C interface to the DUT and the I2C system environment.

```
dut dut_inst(i2c_if);

uvm_config_db#(virtual
svt_i2c_if)::set(uvm_root::get(),"uvm_test_top.env","vif",
i2c_if);
```

The `uvm_config_db` command connects the top-level I2C interface to the virtual interface of the I2C system environment. 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 I2C agent interface to the desired reset signal.

```
assign i2c_if.RST = <User-select Reset Signal>;
```

Note
The reset pin (RST) of the i2c agent interface should be driven by toggled reset to allow proper VIP internal initialization. The RST pin is active HIGH. It should be toggled from 0 -> 1 -> 0 at the starting of simulation before transactions are generated.

2.1.2 Instantiating and Configuring the VIP

The following are steps to instantiate and configure the I2C system environment in your testbench environment.

- Instantiate the I2C system environment (`svt_i2c_system_env`) in the build phase of your testbench environment.

```
svt_i2c_system_env i2c_system_env;

i2c_system_env =
     svt_i2c_system_env::type_id::create("i2c_system_env",
this);
```

- Create a customized I2C system configuration class by extending the I2C system configuration class (`svt_i2c_system_configuration`) and specifies the required configuration parameters.

  For example:

```
class cust_svt_i2c_system_configuration extends
     svt_i2c_system_configuration;

   function new(string name="cust_svt_i2c_system_configuration");
       super.new(name);

   //Create a single I2C master agent and a single slave agent
   this.num_masters = 1;
   this.num_slaves = 1;

   //Create port configurations
   this.create_sub_cfgs(1,1);

     endfunction
   endclass
```

For more information on the configuration class, refer to the *svt_i2c_system_configuration* and *svt_i2c_port_configuration Class References* at the following locations:

```
$DESIGNWARE_HOME/vip/svt/i2c_svt/latest/doc/
i2c_svt_uvm_class_reference/html/
class_svt_i2c_system_configuration.html

$DESIGNWARE_HOME/vip/svt/i2c_svt/latest/doc/
i2c_svt_uvm_class_reference/html/
class_svt_i2c_port_configuration.html
```

- Construct the customized I2C system configuration and pass the configuration to the I2C system environment (instance of `svt_i2c_system_env`) in the build phase of your testbench environment.

```
cfg =
cust_svt_i2c_system_configuration::type_id::create("cfg");

uvm_config_db#(svt_i2c_system_configuration)::set(this,
     "i2c_system_env", "cfg", cfg);
```

The `cust_svt_i2c_system_configuration` is the customized I2C configuration as defined in the previous step. The "cfg" is an instance of this configuration.

2.1.3 Creating a Test Sequence

The VIP provides a base sequence class for the I2C master agent (`svt_i2c_master_transaction_base_sequence`) and the I2C slave agent (`svt_i2c_slave_transaction_base_sequence`). You can extend these base sequences to create test sequences for the I2C master and slave agents.

For more information on the I2C transmitter and receiver base sequences, and the VIP sequence collection, refer to the Sequence Page of the *VC VIP I2C Class Reference* at the following location:

```
$DESIGNWARE_HOME/vip/svt/i2c_svt/latest/doc/
i2c_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/i2c_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 I2C transmitter and receiver sequences.

```
class i2c_base_test extends uvm_test;

  //Build phase
   env = i2c_basic_env::type_id::create("env", this);

uvm_config_db#(uvm_object_wrapper)::set(this,
    "env.i2c_system_env.master*.sequencer.main_phase",
    "default_sequence",i2c_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
+incdir+project_directory_path/src/verilog/simulator
+incdir+project_directory_path/include/verilog
```

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
+incdir+/home/project1/testbench/vip/src/verilog/simulator
+incdir+/home/project1/testbench/vip/include/verilog
```

2.2.2 VIP Compile-time Options

The following are the required compile-time options for compiling a testbench with the VC VIP for I2C:

```
+define+SVT_UVM_TECHNOLOGY
+define+UVM_PACKER_MAX_BYTES=4096
+define+UVM_DISABLE_AUTO_ITEM_RECORDING
+define+SYNOPSYS_SV
```

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 1500000 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

The following compile-time option is required if and only if you have created a user-defined file to override the VIP default values of the system constants such as `scl_high_time_ss` and `scl_low_time_ss`.

```
+define+SVT_I2C_INCLUDE_USER_DEFINES
```

For more information, refer to Section 6.4, "Timing Parameters" of the *VC Verification IP I2C UVM User Guide*.

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 I2C

VC Verification IP UVM Installation and Setup Guide:

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

VC VIP I2C UVM User Guide:

```
$DESIGNWARE_HOME/vip/svt/i2c_svt/latest/doc/i2c_svt_uvm_user_guide.pdf
```

VC VIP I2C Getting Started Guide:

```
$DESIGNWARE_HOME/vip/svt/i2c_svt/latest/doc/i2c_svt_uvm_getting_started.pdf
```

VC VIP I2C QuickStart:

```
$DESIGNWARE_HOME/vip/svt/i2c_svt/latest/examples/sverilog/
tb_i2c_svt_uvm_basic_sys/doc/tb_i2c_svt_uvm_basic_sys/index_basic.html
```

VC VIP I2C Class Reference:

```
$DESIGNWARE_HOME/vip/svt/i2c_svt/latest/doc/i2c_svt_uvm_class_reference/html/
index.html
```

VC VIP I2C Verification Plans:

```
$DESIGNWARE_HOME/vip/svt/i2c_svt/latest/doc/VerificationPlans
```

A.3 Example Home Directory

Directory that contains a list of VIP example directories:

```
$DESIGNWARE_HOME/vip/svt/i2c_svt/latest/examples/sverilog
```

View simulation options for each example:

```
gmake help
```