Verification Continuum™

VC Verification IP
MIPI I3C
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 MIPI I3C (referred to as VIP) into testbenches that are compliant with the SystemVerilog Universal Verification Methodology (UVM). You are assumed to be familiar with the MIPI I3C 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

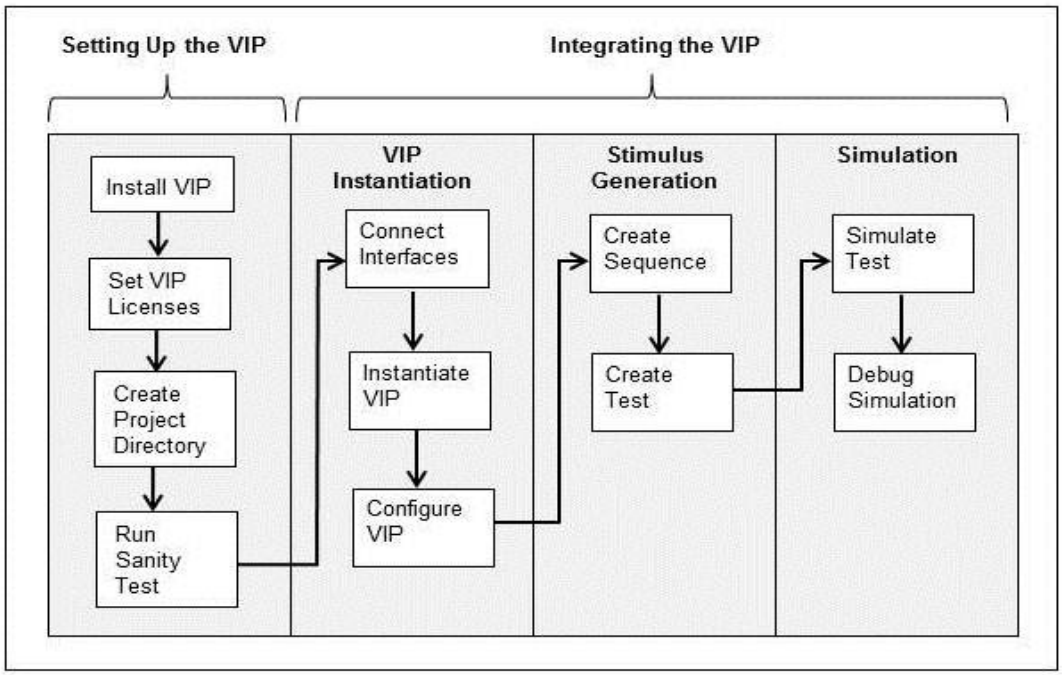
1 Overview of the Getting Started Guide

This Getting Started Guide presents information about integrating the VC VIP for MIPI I3C (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 -> Q-2020.03-> 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 MIPI I3C protocol and UVM. For more information on the VIP, refer to the *VC Verification IP MIPI I3C UVM User Guide* on SolvNet (click here) or in the VIP installation at the following location:

```
$DESIGNWARE_HOME/vip/svt/mipi_i3c_svt/latest/doc/mipi_i3c_svt_uvm_user_guide.pdf
```

2 Integrating the VIP into a User Testbench

The VC VIP for MIPI I3C 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 MIPI I3C Class Reference* (only in HTML format) at the following location:

```
$DESIGNWARE_HOME/vip/svt/mipi_i3c_svt/latest/doc/mipi_i3c_svt_uvm_class_reference/html/
index.html
```

2.1 VIP Testbench Integration Flow

Figure 2-1 Top-level Architecture of an MIPI I3C VIP Testbench

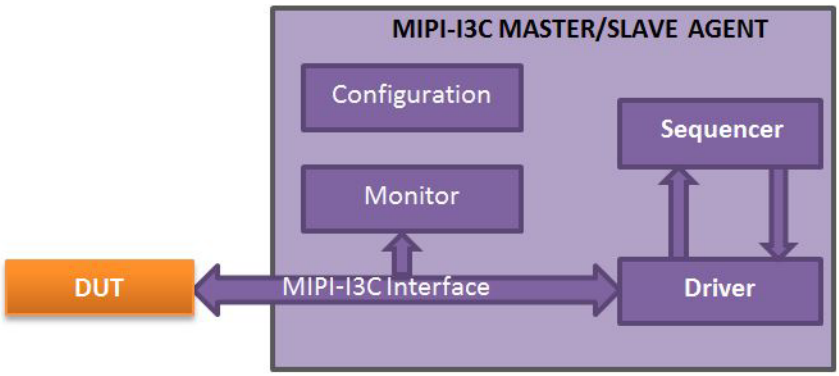


Figure 2-1 is a top-level architecture of a simple VC VIP for MIPI-I3C testbench. The steps for integrating the VIP into a UVM testbench are described in the following sections:

- "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/i3c_svt/latest/examples/sverilog/tb_i3c_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 the I3C SVT UVM package */
`include "svt_mipi_i3c.uvm.pkg"
/** Import required packages */
import uvm_pkg::*;
import svt_uvm_pkg::*;
import svt_mipi_i3c_uvm_pkg::*;
```

Instantiate and connect the top-level I3C interfaces (svt_mipi_i3c_if) to a system clock.

```
svt_mipi_i3c_if i3c_if (SystemClock);

Toggle reset pin of interface from 0 ->1 -> 0.
initial begin
assign i3c_if.RST = 0;
#5;
assign i3c_if.RST = 1;
#5;
assign i3c_if.RST = 0;
end
```

Instantiate MIPI-I3C master and slave BFM wrappers to act as master and slave

```
/** Intantiate Master Wrappers */
svt_mipi_i3c_master_wrapper #(.AGENT_ID(0)) Master0 (i3c_if);
svt_mipi_i3c_master_wrapper #(.AGENT_ID(1)) Master1 (i3c_if);

/** Intantiate Slave Wrappers */
svt_mipi_i3c_slave_wrapper #(.AGENT_ID(0)) Slave0 (i3c_if);
svt_mipi_i3c_slave_wrapper #(.AGENT_ID(1)) Slave1 (i3c_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 MIPI-I3C interface to the DUT and the I3C system environment. Specify one of the following two DUT instantiations:

```
dut  dut_inst(i3c_if);


/** Set the Master Interface to factory */
uvm_config_db#(virtual svt_mipi_i3c_if)::set(uvm_root::get(),
"uvm_test_top.env", "vif", i3c_if);
```

The `uvm_config_db` command connects the top-level I3C interface to the virtual interface of the MIPI-I3C system environment. The `uvm_test_top` represents the top-level module in the UVM environment. The *env* is an instance of your testbench environment.

Note
The reset pin (RST) of the i3c 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 MIPI-I3C system environment in your testbench environment.

- Instantiate the I3C system environment (`svt_mipi_i3c_system_env`) in the build phase of your testbench environment.

```
      svt_mipi_i3c_system_env  i3c_system_env;
       i3c_system_env = svt_mipi_i3c_system_env::type_id::create("i3c_system_env",
this);
```

- Create a customized I3C system configuration class by extending the I3C system configuration class (svt_mipi_i3c_system_configuration) and specifies the required configuration parameters.

  For example:

```
class cust_svt_mipi_i3c_system_configuration extends
svt_mipi_i3c_system_configuration;
function new(string name="cust_svt_mipi_i3c_system_configuration");
super.new(name);
/** Set mode */
foreach(this.master_cfg[i])
this.master_cfg[i].is_active = 1;
foreach(this.slave_cfg[i])
this.slave_cfg[i].is_active = 1;
endfunction
endclass
```

For more information on the configuration class, see the svt_mipi_i3c_agent_configuration Class References at the following locations:

```
$DESIGNWARE_HOME/vip/svt/i3c_svt/latest/doc/i3c_svt_uvm_class_reference/html/class_svt_mipi_i3c_agent_configuration.html
```

2.1.3 Creating a Test Sequence

The VIP provides a base sequence class for the I3C master agent (`svt_mipi_i3c_master_transaction_base_sequence`) and the I3C slave agent (`svt_mipi_i3c_slave_transaction_base_sequence`). You can extend these base sequences to create test sequences for the I3C master and slave agents.

For more information on the I3C master and slave base sequences, and the VIP sequence collection, refer to the Sequence Page of the VC VIP I3C Class Reference at the following location:

```
$DESIGNWARE_HOME/vip/svt/mipi_i3c_svt/latest/doc/mipi_i3c_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/mipi_i3c_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 can construct the testbench environment and set the respective I3C master and slave sequences.

```
class i3c_base_test extends uvm_test;
//Build phase
/** Instantiate the configuration for Master*/
  cust_svt_mipi_i3c_system_configuration cfg;

    env = i3c_basic_env::type_id::create("env", this);
    /** Create the configuration object for Master agent */
    cfg = cust_svt_mipi_i3c_system_configuration::type_id::create("cfg");
/** Configure Master and Slave configurations */
    if(!cfg.randomize() with {
                                num_masters            == 2;
                                num_slaves             == 2;
                                master_cfg[0].mst_type ==
svt_mipi_i3c_configuration::MAIN_MST;
                                master_cfg[1].mst_type ==
svt_mipi_i3c_configuration::SEC_MST;
                                slave_cfg[0].slv_type  ==
svt_mipi_i3c_configuration::I2C_LEGACY_SLV;
```

```
cslave_cfg[1].slv_type  == svt_mipi_i3c_configuration::I3C_SLV;
}
      ) begin
      `svt_error("build_phase", "Randomization failure.");
    end

    /**** Configure Master and Slave configurations *****/

  // configuring master[0] / main master
    cfg.master_cfg[0].is_active = 1;

    // configuring master[1] / Sec. Master
    cfg.master_cfg[1].is_active = 1;
    cfg.master_cfg[1].device_static_address   = 'h31;
    cfg.master_cfg[1].device_dynamic_address  = 'h34;

    // configuring slave[0] / Legacy I2C Slave
    cfg.slave_cfg[0].is_active  = 1;
cfg.slave_cfg[0].device_static_address =
`SVT_MIPI_I3C_DEFAULT_STATIC_ADDRESS;
   cfg.slave_cfg[0].legacy_i2c_slave_type = `SVT_MIPI_I3C_I2C_GENERIC
;

    // configuring slave[1] / I3C Slave
    cfg.slave_cfg[1].is_active = 1;
    cfg.slave_cfg[1].device_static_address        = 'h00;
    cfg.slave_cfg[1].device_dynamic_address       = 'h35;

    // Version 0.7r1, Section 5.1.4.2 : Notice that there is no possibility for two or
  // moredevices to have the same Characteristic Register on the same I3C
  // bus instantiation; at least the BCR shall be different. However, due
  // to various error conditions, there is a very slim possibility for
  // such coincidence to happen.
// Note that the above stated error condition is not supported by the
// VIP, and hence the BCR and DCR for each device need to be
// configured with unique values.
```

```
    cfg.master_cfg[1].reg_cfg.bus_char_reg    = 8'b1101_0010;
       cfg.master_cfg[1].reg_cfg.device_char_reg = 8'b1101_0011;
       cfg.slave_cfg[0].reg_cfg.legacy_i2c_device_char_reg = 8'b1101_0001;
       cfg.slave_cfg[1].reg_cfg.bus_char_reg    = 8'b1101_0100;
       cfg.slave_cfg[1].reg_cfg.device_char_reg = 8'b1101_0101;
       cfg.slave_cfg[0].device_static_address  =
`SVT_MIPI_I3C_DEFAULT_STATIC_ADDRESS;  // Set Slave Address
      cfg.slave_cfg[0].legacy_i2c_slave_type  = `SVT_MIPI_I3C_I2C_GENERIC
;  // Set Slave as Generic


    // TBD: Remove this: Set timing configurations
       cfg.master_cfg[0].timing_cfg.i2c_scl_high_time_fs = 1300;
       cfg.master_cfg[0].timing_cfg.i2c_scl_low_time_fs  = 600;
       cfg.master_cfg[0].timing_cfg.i2c_scl_rise_time_fs = 300;
       cfg.master_cfg[0].timing_cfg.i2c_scl_fall_time_fs = 300;
       cfg.master_cfg[1].timing_cfg.i2c_scl_high_time_fs = 1300;
       cfg.master_cfg[1].timing_cfg.i2c_scl_low_time_fs  = 600;
       cfg.master_cfg[1].timing_cfg.i2c_scl_rise_time_fs = 300;
       cfg.master_cfg[1].timing_cfg.i2c_scl_fall_time_fs = 300;
       cfg.slave_cfg[0].timing_cfg.i2c_scl_high_time_fs  = 1300;
       cfg.slave_cfg[0].timing_cfg.i2c_scl_low_time_fs   = 600;
       cfg.slave_cfg[0].timing_cfg.i2c_scl_rise_time_fs  = 300;
       cfg.slave_cfg[0].timing_cfg.i2c_scl_fall_time_fs  = 300;
       cfg.slave_cfg[1].timing_cfg.i2c_scl_high_time_fs  = 1300;
       cfg.slave_cfg[1].timing_cfg.i2c_scl_low_time_fs   = 600;
       cfg.slave_cfg[1].timing_cfg.i2c_scl_rise_time_fs  = 300;
       cfg.slave_cfg[1].timing_cfg.i2c_scl_fall_time_fs  = 300;
```

```
    uvm_config_db#(uvm_object_wrapper)::set(this,"env.i3c_system_env.sequence
    r.main_phase",
    "default_sequence",i3c_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 I3C:

- `+define+SVT_UVM_TECHNOLOGY`
- `+define+UVM_PACKER_MAX_BYTES=1500000`
- `+define+UVM_DISABLE_AUTO_ITEM_RECORDING`
- `+define+SYNOPSYS_SV`

For more information, see *Timing Parameters* section in the VC Verification IP I3C UVM User Guide.

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.

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 MIPI I3C

- VC Verification IP UVM Installation and Setup Guide:

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

- VC VIP I3C UVM User Guide:

  `$DESIGNWARE_HOME/vip/svt/mipi_i3c_svt/latest/doc/i3c_svt_uvm_user_guide.pdf`

- VC VIP I3C Getting Started Guide:

  `$DESIGNWARE_HOME/vip/svt/mipi_i3c_svt/latest/doc/i3c_svt_uvm_getting_started.pdf`

- VC VIP I3C Class Reference:

  `$DESIGNWARE_HOME/vip/svt/mipi_i3c_svt/latest/doc/i3c_svt_uvm_class_reference/html/index.html`

- VC VIP I3C Verification Plans:

  `$DESIGNWARE_HOME/vip/svt/mipi_i3c_svt/latest/doc/VerificationPlans`

A.3 Example Home Directory

Directory that contains a list of VIP example directories:

```
$DESIGNWARE_HOME/vip/svt/mipi_i3c_svt/latest/examples/sverilog
```

View simulation options for each example:

```
gmake help
```