Verification Continuum™

VC Verification IP
MPHY
UVM Getting Started Guide

Version U-2022.12, December 2022

SYNOPSYS®

Contents

Preface

About This Document

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

Web Resources

- Documentation through SolvNetPlus: https://solvnetplus.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:

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 Tool Version in your e-mail so it can be routed correctly.
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:

     https://www.synopsys.com/support/global-support-centers.html

Synopsys Statement on Inclusivity and Diversity

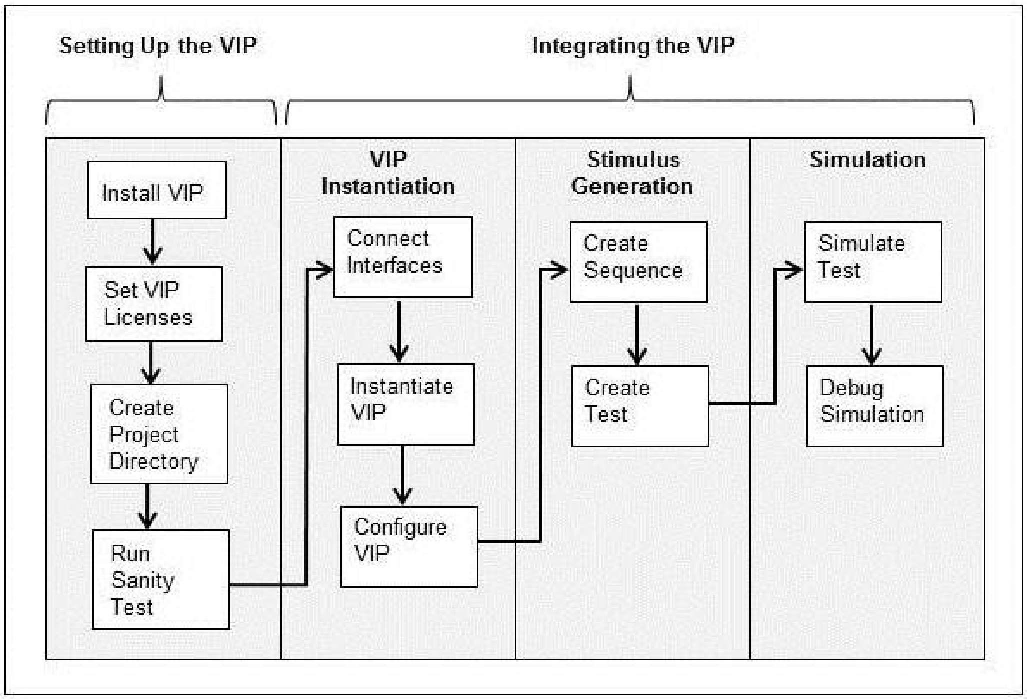
1 Overview of the Getting Started Guide

This Getting Started Guide presents information about integrating the VC VIP for MPHY (referred to as VIP) into testbenches that are compliant with the SystemVerilog Universal Verification Methodology (UVM). This 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 Installation and Setup Guide*. This guide is available on the SolvNetPlus Download Centre (click here -> VC VIP Library -> T-2022.09 -> 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 MPHY protocol and UVM. For more information on the VIP, see the *VC Verification IP MPHY UVM User Guide* on SolvNetPlus (click here) or in the VIP installation at the following location:

```
$DESIGNWARE_HOME/vip/svt/mphy_svt/latest/doc/PDFs/mphy_svt_uvm_user_guide.pdf
```

2 Integrating the VIP Into a User Testbench

The VC VIP for MPHY 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, see the main page of the *VC VIP MPHY Class Reference* (only in HTML format) at the following location:

```
$DESIGNWARE_HOME/vip/svt/mphy_svt/latest/doc/class_ref/mphy_svt_uvm_class_refere
nce/html/
index.html
```

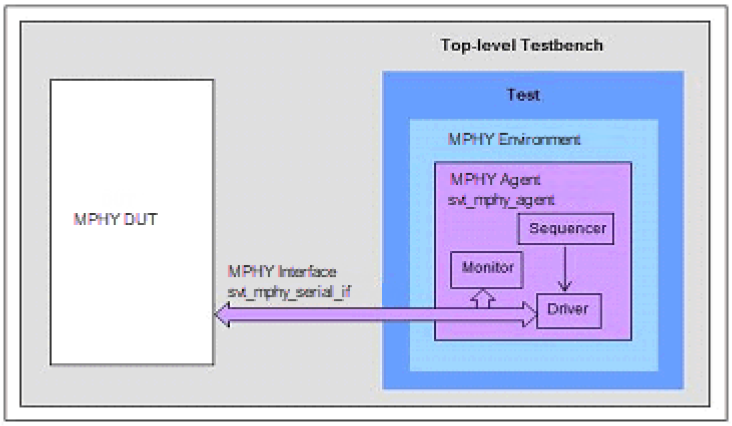
2.1 VIP Testbench Integration Flow

The MPHY agent (svt_mphy_agent) is the top-level component provided by the VIP for modeling MPHY hosts, devices and hubs. This generic agent encapsulates the following components:

- ❖ Sequencer
- ❖ Monitor
- ❖ Driver

A MPHY environment is a user-defined UVM environment that encapsulates all of the VIP components, and implicitly constructs the required number of MPHY host and MPHY device agents as specified by its system configuration object. You can instantiate and construct the MPHY environment in the top-level environment of your UVM testbench.

Figure 2-1 Top-level Architecture of a MPHY VIP Testbench



This is a top-level architecture of a simple VC VIP for MPHY testbench. The testbench includes an instance of a MPHY agent connecting through the MPHY serial interface to an instance of a device that represents the DUT. 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"

These steps are documented for a VIP configured as a Master MPHY verifying a PHY(DUT) connected through a serial interface. Similar steps can be followed for other serial interfaces. 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, see the following example:

```
$DESIGNWARE_HOME/vip/svt/mphy_svt/latest/examples/sverilog/
tb_mphy_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 "svt_mphy_defines.svi"
`include "svt_mphy.uvm.pkg"     //VIP package
`include "svt_mphy_if.uvm.svi"  //top-level mphy interface

import uvm_pkg::*;
`include "uvm_macros.svh"
import svt_uvm_pkg::*;
import svt_mphy_uvm_pkg::*;
```

Note You must compile all VIP files with the timescale 10ps/1fs. You can use the timescale option at compile-time or add the `` `timescale 10ps/1fs `` statement before including the VIP files

- Instantiate the top-level MPHY interface (svt_mphy_if) and connect the serial signals to the DUT.

```
svt_mphy_serial_tx_if      slave_mtx_serial_if ();
svt_mphy_serial_rx_if      slave_mrx_serial_if ();

assign slave_mrx_serial_if[j].rx_dp = DUT_if[j].tx_dp;
assign slave_mrx_serial_if[j].rx_dn = DUT_if[j].tx_dn;

assign DUT_if[j].rx_dp = slave_mtx_serial_if[j].tx_dp;
assign DUT_if[j].rx_dn = slave_mtx_serial_if[j].tx_dn;
```

- Connect the top-level MPHY interface to the virtual interface of the MPHY environment.

```
        intial begin
        uvm_config_db#(virtual svt_mphy_serial_if)::set(uvm_root::get(),
        "uvm_test_top.env", "mtx_serial_if", mphy_slave_mtx_serial_if);
        uvm_config_db#(virtual
        svt_mphy_serial_if)::set(uvm_root::get(),
        "uvm_test_top.env", "mrx_mphy_if",
        mphy_slave_mrx_serial_if);
        end
```

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

2.1.2 Instantiating and Configuring the VIP

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

- ✦ Instantiate the MPHY environment in the build phase of your testbench environment.

```
        mphy_lm_env env;
        env = mphy_lm_env env::type_id::create("env", this);
```

- ✦ Instantiate the MPHY agent (svt_mphy_model_agent) in the build phase of your testbench environment and bind the environment virtual interface to the agent virtual interface.

```
        svt_mphy_tx_agent tx_agent;
        svt_mphy_rx_agent rx_agent;
        svt_mphy_serial_tx_if mtx_mphy_if;
        svt_mphy_serial_rx_if mrx_mphy_if;

        tx_agent = svt_mphy_tx_agent::type_id::create("tx_agent",this);
        rx_agent = svt_mphy_rx_agent::type_id::create("rx_agent",this);
        void'(uvm_config_db#(svt_mphy_mtx_if)::get(null,get_full_name(),"mtx_m
        phy_if",mtx_mphy_if);
        void'(uvm_config_db#(svt_mphy_mrx_if)::get(null,get_full_name(),"mrx_m
        phy_if",mrx_mphy_if);
```

- ✦ Create a basic configuration class by extending the uvm_object class. This configuration class specifies the MPHY protocol layer configuration for the MPHY host and device.

  For example,

```
        class mphy_shared_cfg extends uvm_object;
          `uvm_object_utils(mphy_shared_cfg)

           svt_mphy_agent_configuration tx_cfg;
           svt_mphy_configuration model_cfg;
           svt_mphy_agent_configuration rx_cfg;

           function new(string name = "mphy_shared_cfg");
             super.new(name);
```

```
            tx_cfg = svt_mphy_agent_configuration::type_id::create("tx_cfg");
            model_cfg = svt_mphy_configuration::type_id::create("model_cfg");
            rx_cfg = svt_mphy_agent_configuration::type_id::create("rx_cfg");

            rand_success = tx_cfg.randomize() with {
            tx_cfg.dir == svt_mphy_configuration::M_TX;
            tx_cfg.interface_type == svt_mphy_configuration::SERIAL;
                                    };
                 if (!rand_success)
                  `uvm_fatal("new","unable to randomize tx_cfg");

                 rand_success = 0;
                 rand_success = rx_cfg.randomize() with {
                                rx_cfg.dir == svt_mphy_configuration::M_RX;
                                rx_cfg.interface_type ==
            svt_mphy_configuration::SERIAL;
            };
                 if (!rand_success)
                  `uvm_fatal("new","unable to randomize rx_cfg");

                 this.tx_cfg.is_active = 1;
                 this.tx_cfg.enable_chk = 1;
                 this.tx_cfg.mphy_module_type = svt_mphy_types::TYPE_I;
                 this.tx_cfg.mphy_ls_signaling_type = svt_mphy_types::PWM;
                 this.tx_cfg.mphy_ls_pwm_type = svt_mphy_types::FIXED_RATIO;
                 this.tx_cfg.interface_type = svt_mphy_configuration::SERIAL;
                 this.tx_cfg.component_type = svt_mphy_configuration::PHY;
                 this.tx_cfg.operating_freq = svt_mphy_configuration::FREQ_26;
                 this.tx_cfg.init_mode = svt_mphy_types::LS_MODE;
                 this.tx_cfg.init_pwmgear = svt_mphy_types::PWM_G1;
                 this.tx_cfg.init_fsm_state = svt_mphy_types::SLEEP;
```

```
             this.tx_cfg.mphy_protocol_version =
            svt_mphy_configuration::MPHY_VERSION_30;
            `uvm_info("new","Initialising the component configuration for RX ...",
            UVM_MEDIUM);
                 this.rx_cfg.is_active = 1;
                 this.rx_cfg.enable_chk = 1;
            this.rx_cfg.mphy_module_type = svt_mphy_types::TYPE_I;

            this.rx_cfg.mphy_ls_signaling_type = svt_mphy_types::PWM;
                 this.rx_cfg.mphy_ls_pwm_type = svt_mphy_types::FIXED_RATIO;
                 this.rx_cfg.interface_type = svt_mphy_configuration::SERIAL;
                 this.rx_cfg.component_type = svt_mphy_configuration::PHY;
                 this.rx_cfg.operating_freq = svt_mphy_configuration::FREQ_26;
                 this.rx_cfg.mphy_protocol_version =
            svt_mphy_configuration::MPHY_VERSION_30;

               endfunction // new

            endclass
```

Note The MPHY host configuration must include the configuration of a MPHY device that is to communicate with the host.

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

`$DESIGNWARE_HOME/vip/svt/mphy_svt/latest/doc/class_ref/mphy_svt_uvm_class_reference/html/index.html`

✦ Construct the customized MPHY configuration and pass the configuration to the MPHY environment (instance of `svt_mphy_top_env`) in the build phase of your testbench environment.

```
mphy_shared_cfg cfg;
cfg = mphy_shared_cfg::type_id::create("cfg", this);
```

Note The `mphy_shared_cfg` is the customized MPHY configuration as defined in the previous step. The `‘cfg’` is an instance of this configuration.

✦ Assign configuration to the agent object in the build phase of your testbench environment.

```
mphy_shared_cfg cfg;

void'(uvm_config_db#(mphy_shared_cfg)::get(get_parent(), get_name(),
"cfg", cfg));

if (cfg == null) begin

end
else
begin
  `uvm_info("build_phase", "Using externally provided cfg.",
            UVM_LOW)
end

if (this.cfg.is_valid(0)) begin
  `uvm_info("build_phase", $sformatf("cfg passed is_valid()
check.Contents:\n%0s", this.cfg.sprint()), UVM_LOW)
end
else
begin
  `uvm_fatal("build_phase", "cfg failed is_valid() check. Unable to
continue.")
end
  uvm_config_db#(svt_mphy_agent_configuration)::set(this, "tx_agent",
"tx_cfg",  this.cfg.tx_cfg);
```

2.1.3 Creating a Test Sequence

The VIP provides a base sequence class for the MPHY(`svt_mphy_transaction_base_sequences`). You can extend these base sequences to create test sequences for MPHY tx agent.

For more information on the MPHY base sequences, and the VIP sequence collection, see the Sequence Page of the *VC VIP MPHY Class Reference* at the following location:

$DESIGNWARE_HOME/vip/svt/mphy_svt/latest/doc/class_ref/mphy_svt_uvm_class_reference/htmlsequncepages.html

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

$DESIGNWARE_HOME/vip/svt/mphy_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 MPHY host and device sequences.

```
class ls_hs_data_transaction extends mphy_base_test
// build_phase
uvm_config_db#(uvm_object_wrapper)::set(this, "env.sequencer.main_phase",
"default_sequence",
svt_mphy_ls_hs_data_transaction_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 MPHY:

```
+define+SVT_UVM_TECHNOLOGY
+define+UVM_PACKER_MAX_BYTES=24000
+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.

Table 2-1 Macro

Macro	Description
`SVT_UVM_TECHNOLOGY`	Specifies SystemVerilog based VIPs that are compliant with UVM
`UVM_PACKER_MAX_BYTES`	Sets to 24000 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

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=ls_hs_data_transaction
```

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 MPHY

VC VIP UVM Installation and Setup Guide:

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

VC VIP MPHY UVM User Guide:

`$DESIGNWARE_HOME/vip/svt/mphy_svt/latest/doc/PDFs/mphy_svt_uvm_user_guide.pdf`

VC VIP MPHY Getting Started Guide:

`$DESIGNWARE_HOME/vip/svt/mphy_svt/latest/doc/PDFs/mphy_svt_uvm_getting_started.pdf`

VC VIP MPHY QuickStart:

```
$DESIGNWARE_HOME/vip/svt/mphy_svt/latest/doc/class_ref/mphy_svt_uvm_class_reference/html
/index_basic.html
```

VC VIP MPHY Class Reference:

```
$DESIGNWARE_HOME/vip/svt/mphy_svt/latest/doc/class_ref/mphy_svt_uvm_class_reference/html
/index.html
```

VC VIP MPHY Verification Plans:

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

A.3 Example Home Directory

Directory that contains a list of VIP example directories:

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

View simulation options for each example:

```
gmake help
```