Verification Continuum™

VC Verification IP JEDEC UFS UVM Getting Started Guide

Version Q-2020.06, June 2020

SYNOPSYS®

Contents

Preface

About This Document

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

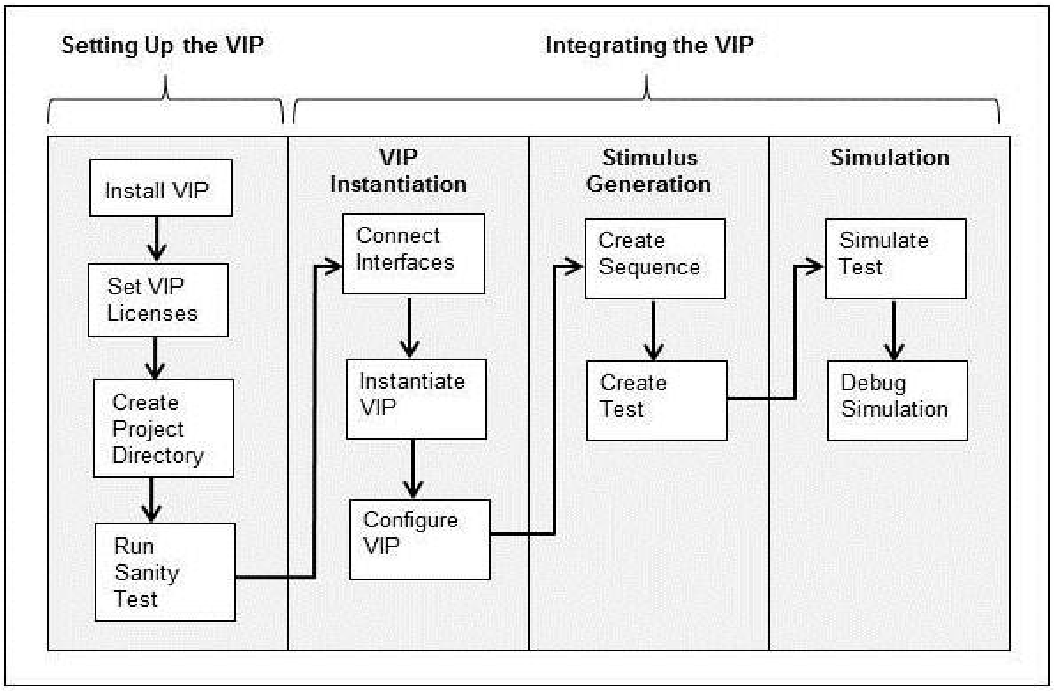
1
Overview of the Getting Started Guide

This Getting Started Guide presents information about integrating the VC VIP for JEDEC UFS (referred to as VIP) into testbenches that are compliant with the SystemVerilog Universal Verification Methodology (UVM). The figure VIP Integration and Test Work Flow 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 SolvNet Download page 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 JEDEC UFS protocol and UVM. For more information on the VIP, see the *VC Verification IP JEDEC UFS UVM User Guide* on SolvNet (click here) or in the VIP installation at the following location:

$DESIGNWARE_HOME/vip/svt/jedec_ufs_svt/latest/doc/jedec_ufs_svt_uvm_user_guide.pdf

2 Integrating the VIP into a User Testbench

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

$DESIGNWARE_HOME/vip/svt/jedec_ufs_svt/latest/doc/jedec_ufs_svt_uvm_class_reference/html/index.html

2.1 VIP Testbench Integration Flow

The JEDEC UFS agent (`svt_jedec_ufs_host_agent/svt_jedec_ufs_device_agent`) is the top-level component provided by the VIP for modeling UFS Host and Device agents. This generic agent encapsulates the following components:

- ❖ Sequencer
- ❖ Monitor
- ❖ Driver

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

Figure 2-1 Top-level Architecture of a JEDEC UFS VIP Testbench

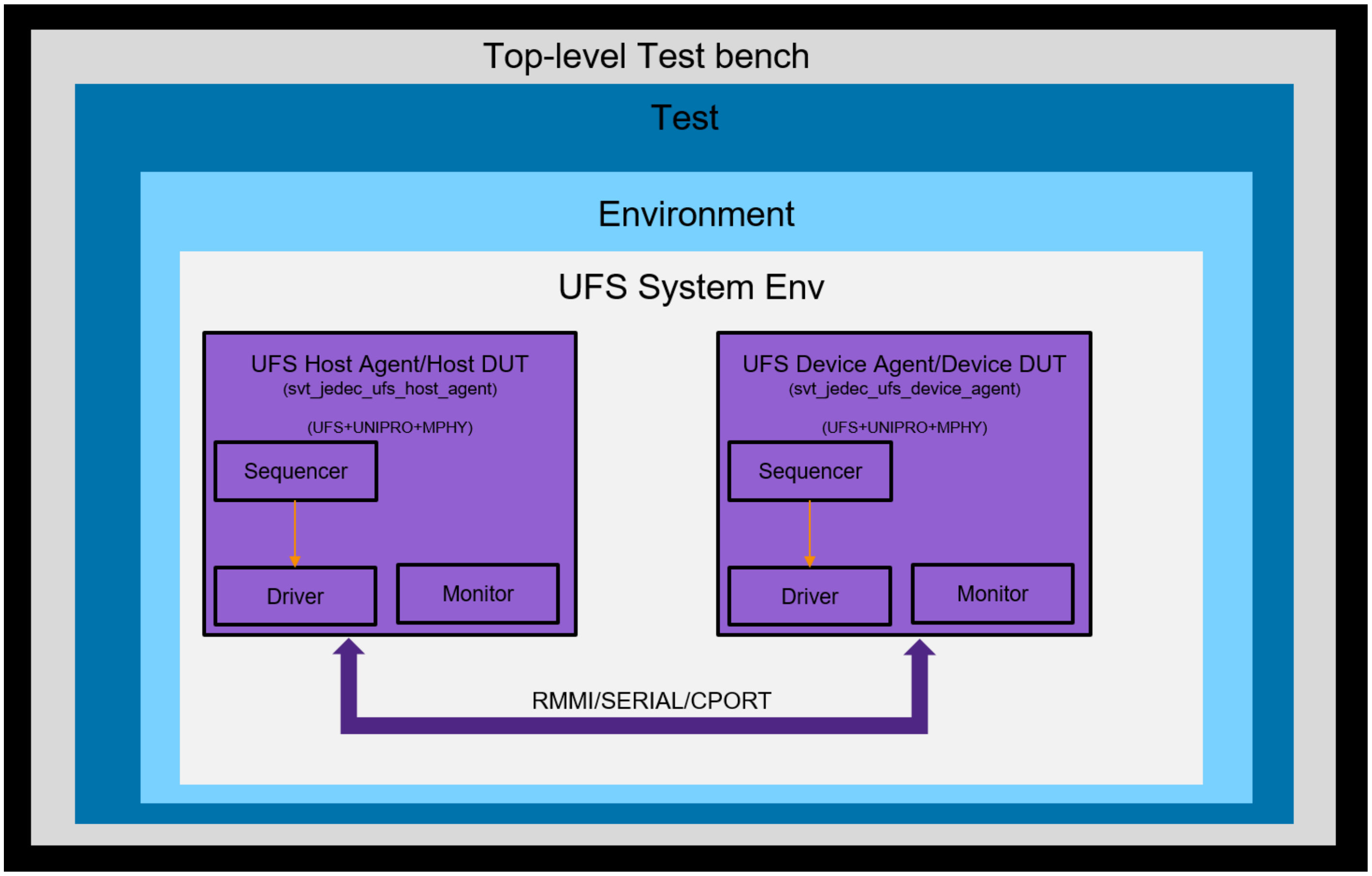


Figure 2-1 is a top-level architecture of a simple VC VIP for JEDEC UFS testbench. The steps for integrating the VIP into a UVM testbench are described in the following sections:

- "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, see the following example:

```
$DESIGNWARE_HOME/vip/svt/jedec_ufs_svt/latest/examples/sverilog/
tb_jedec_ufs_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_jedec_ufs.uvm.pkg"
 import uvm_pkg::*;

`include "uvm_macros.svh"
import svt_uvm_pkg::*;
import svt_mem_uvm_pkg::*;
import svt_mphy_component_uvm_pkg::*;
import svt_mipi_unipro_uvm_pkg::*;
import svt_jedec_ufs_uvm_pkg::*;
```

2.1.1.1 Topology: DUT as UFS Device

If Host VIP, then follow the following steps:

❖ In testbench top, import UVM package, macros and UFS UVM package. Instantiate Serial interface.

```
svt_mphy_serial_rx_if
mphy_serial_rx_if_devA[`MPHY_RX_LANES]();
svt_mphy_serial_tx_if
mphy_serial_tx_if_devA[`MPHY_TX_LANES]();
```

❖ Declare global slck and sdata signals in top file.

```
/**
   * Instantiate interconnect. This makes the connections between the
interfaces of the
   * TX-DUT Controller I/F and TX-DUT PHY I/F. It has a module to connect
control signals.
   */
  genvar i;
  generate
    for(i=0; i<`MPHY_TX_LANES; i++) begin : mphy_serial
      mphy_svt_hdl_interconnect_sv_wrapper
interconnect_wrapper_serial_tx(mphy_serial_rx_if_devA[i], DUT_tx_if[i]);
      mphy_svt_hdl_interconnect_sv_wrapper
interconnect_wrapper_serial_rx(DUT_rx_if[i], mphy_serial_tx_if_devA[i]);
    end
  endgenerate
```

❖ Connect top-level UFS system interface to Virtual interface of UFS system environment.

```
genvar k;
  generate
  for (k=0; k<`MPHY_TX_LANES; k=k+1) begin
    initial
    begin
      /** Provide the Master and Slave Tx and Rx Serial Interface to the
UFS_SYS_ENV. This step
        * establishes the connection between the MPHY UFS_SYS_ENV and the HDL
Interconnect,
        * wrapper, through the MPHY interface
```

```
           */
                uvm_config_db#(virtual svt_mphy_serial_tx_if)::set(uvm_root::get(),
          "uvm_test_top.ufs_sys_env",$sformatf("mphy_serial_tx_if_devA[%0d]",k),
          mphy_serial_tx_if_devA[k]);
                uvm_config_db#(virtual svt_mphy_serial_rx_if)::set(uvm_root::get(),
          "uvm_test_top.ufs_sys_env",$sformatf("mphy_serial_rx_if_devA[%0d]",k),
          mphy_serial_rx_if_devA[k]);
              end
            end
            endgenerate
```

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

2.1.2 Instantiating and Configuring the VIP

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

- Create a customized JEDEC UFS system configuration class by instantiating host and device agent configuration (`svt_jedec_ufs_agent_configuration`) classes and specifying the required configuration parameters.

For example,

```
class ufs_system_configuration extends uvm_object;

  /**
   * Configuration object for host
   */
  rand svt_jedec_ufs_agent_configuration  host_cfg;

 `uvm_object_utils_begin(ufs_system_configuration)
    `uvm_field_object   (host_cfg,      UVM_ALL_ON|UVM_REFERENCE)
  `uvm_object_utils_end

//------------------------------------------------------------------------
-------------------------------
  /**
   * Constructor
   */
//------------------------------------------------------------------------
-------------------------------
  function new(string name = "ufs_system_configuration");
    super.new(name);
    host_cfg =
svt_jedec_ufs_agent_configuration::type_id::create("host_cfg");
    host_cfg.protocol_transport_cfg = new();
    host_cfg.protocol_transport_cfg.is_host = 1'b1;
    host_cfg.pa_avail_tx_data_lanes = `MPHY_TX_LANES;
    host_cfg.pa_avail_rx_data_lanes = `MPHY_RX_LANES;
      host_cfg.initial_transport_status[0] = new();
          host_cfg.initial_network_status = new();
          host_cfg.initial_data_link_status = new();
          host_cfg.initial_phy_adapter_status = new();
          host_cfg.cport_cfg[0] = new();
      host_cfg.protocol_transport_cfg.device_cfg = new();

          host_cfg.cport_cfg[0].cport_id = 0;
           host_cfg.initial_network_status.n_device_id = 0;
          host_cfg.initial_transport_status[0].t_peer_cport_id = 0;
         host_cfg.initial_transport_status[0].t_peer_cport_id_init
      = 0;
          host_cfg.initial_transport_status[0].t_peer_device_id =
      1;

      host_cfg.initial_transport_status[0].t_peer_device_id_init =
      1;
          host_cfg.initial_transport_status[1].t_peer_device_id =
      1;

      host_cfg.initial_transport_status[1].t_peer_device_id_init =
      1;
 host_cfg.initial_transport_status[2].t_peer_device_id = 1;

host_cfg.initial_transport_status[2].t_peer_device_id_init =
1;

host_cfg.initial_phy_adapter_status.pa_connected_tx_data_lane
= 2;

host_cfg.initial_phy_adapter_status.pa_connected_rx_data_lane
= 2;

host_cfg.initial_phy_adapter_status.pa_connected_tx_data_lane
_reset = 2;

host_cfg.initial_phy_adapter_status.pa_connected_rx_data_lane
_reset = 2;

host_cfg.initial_phy_adapter_status.pa_active_tx_data_lanes =
1;

host_cfg.initial_phy_adapter_status.pa_active_rx_data_lanes =
1;

host_cfg.initial_phy_adapter_status.pa_active_tx_data_lanes_r
eset = 1;

host_cfg.initial_phy_adapter_status.pa_active_rx_data_lanes_r
eset = 1;
    host_cfg.protocol_transport_cfg.enable_ufs_version =
svt_jedec_ufs_protocol_transport_configuration::JEDEC_UFS_VER
SION_2_1;
host_cfg.protocol_transport_cfg.b_max_number_lu = 1;
 // Copy device lun information to host
    host_cfg.initial_protocol_transport_status = new();
    for (int i = 0; i <
this.host_cfg.protocol_transport_cfg.get_max_num_of_luns();
i++)
    begin

host_cfg.initial_protocol_transport_status.lun_status.push_ba
ck(null);
host_cfg.initial_protocol_transport_status.lun_status[i] =
device_cfg.initial_protocol_transport_status.lun_status[i];

host_cfg.protocol_transport_cfg.lun_cfg.push_back(null);
      host_cfg.protocol_transport_cfg.lun_cfg[i] =
device_cfg.protocol_transport_cfg.lun_cfg[i];
    end
```

```
//Host configuration
    host_cfg.protocol_transport_cfg.is_host = 1'b1;
    host_cfg.enable_protocol_transport_cov = 2'b01;
    host_cfg.enable_protocol_transport_xml_gen = 1'b1;
    host_cfg.enable_protocol_transport_reporting = 32'h1;
    host_cfg.enable_protocol_transport_tracing = 32'h1;
    host_cfg.enable_protocol_transport_exceptions = 1'b1;
    host_cfg.initial_dme_state =
svt_mipi_unipro_types::DME_LINK_UP_STATE;
    host_cfg.enable_transport_reporting = 1;
    host_cfg.enable_transport_tracing = 1;
    host_cfg.enable_transport_cov = 1;
    host_cfg.enable_transport_xml_gen = 1;
    host_cfg.enable_network_reporting = 1;
    host_cfg.enable_network_tracing = 1;
    host_cfg.enable_network_cov = 1;
    host_cfg.enable_network_xml_gen = 1;
    host_cfg.enable_data_link_reporting = 1;
    host_cfg.enable_data_link_tracing = 1;
    host_cfg.enable_data_link_xml_gen = 1;
    host_cfg.enable_data_link_cov = 1;
    host_cfg.enable_phy_adapter_reporting = 1;
    host_cfg.enable_phy_adapter_tracing = 1;
    host_cfg.enable_phy_adapter_cov = 1;
    host_cfg.enable_phy_adapter_xml_gen = 1;
    host_cfg.enable_dme_cov = 1;
    host_cfg.enable_dme_xml_gen = 1;
    host_cfg.enable_dme_reporting = 1;
    host_cfg.enable_dme_tracing = 1;
    host_cfg.enable_phy_cov = 2'b11;
    host_cfg.enable_phy_xml_gen = 1;
    host_cfg.enable_phy_reporting = 1;
    host_cfg.enable_phy_tracing = 1;
```

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

```
$DESIGNWARE_HOME/vip/svt/jedec_ufs_svt/latest/doc/
jedec_ufs_svt_uvm_class_reference/html/
class_svt_jedec_ufs_agent_configuration.html
```

❖ Construct the customized JEDEC UFS system configuration and pass the configuration to the JEDEC UFS system environment (instance of `svt_jedec_ufs_system_env`) in the build phase of your testbench environment.

```
sys_cfg =
ufs_system_configuration::type_id::create("sys_cfg");

uvm_config_db#(svt_jedec_ufs_agent_configuration)::set(this,
"host", "cfg", sys_cfg.host_cfg);
```

The `ufs_system_configuration` is the customized JEDEC UFS system configuration as defined in the previous step. The `cfg` is an instance of this configuration.

2.1.3 Creating a Test Sequence

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

$DESIGNWARE_HOME/vip/svt/jedec_ufs_svt/latest/doc/jedec_ufs_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, see the example directories at the following location:

$DESIGNWARE_HOME/vip/svt/jedec_ufs_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 JEDEC UFS sequences.

```
class ufs_system_base_test extends uvm_test;
 ufs_system_env ufs_sys_env;
 uvm_config_db#(ufs_system_configuration)::set(this,
"ufs_sys_env", "sys_cfg", cust_sys_cfg);
  ufs_sys_env =
ufs_system_env::type_id::create("ufs_sys_env",this);
uvm_config_db#(uvm_object_wrapper)::set(this,
"ufs_sys_env.sys_virt_sequencer.main_phase",
"default_sequence",backdoor_mem_wr_rd_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"

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_design_dir_path/include/sverilog
+incdir+project_design_dir_path/src/sverilog/simulator
```

Where, *`project_design_dir_path`* is your project design directory and *`simulator`* is vcs, ncv or mti.

For example:

```
+incdir+/home/project/design_dir/vip/include/sverilog
+incdir+/home/project/design_dir/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 JEDEC UFS:

```
+define+SVT_UVM_TECHNOLOGY
+define+UVM_DISABLE_AUTO_ITEM_RECORDING
+define+SYNOPSYS_SV
```

Macro	Description
SVT_UVM_TECHNOLOGY	Specifies SystemVerilog based VIPs that are compliant with UVM
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=base_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 JEDEC UFS

VC VIP UVM Installation and Setup Guide:

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

VC VIP JEDEC UFS Getting Started Guide:

$DESIGNWARE_HOME/vip/svt/jedec_ufs_svt/latest/doc/jedec_ufs_svt_uvm_getting_started.pdf

VC VIP JEDEC UFS Class Reference:

$DESIGNWARE_HOME/vip/svt/jedec_ufs_svt/latest/doc/jedec_ufs_svt_uvm_class_reference/html/index.html

A.3 Example Home Directory

Directory that contains a list of VIP example directories:

$DESIGNWARE_HOME/vip/svt/jedec_ufs_svt/latest/examples/sverilog

View simulation options for each example,

```
        gmake help
```