Verification Continuum™

VC Verification IP
PCIe
EP/RC DUT Integration Guide

Version S-2021.06, June 2021

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.

www.synopsys.com

Contents

Preface

About This Document

This guide provides the information on how to integrate an RC/EP DUT in PIPE or Serial mode for VC VIP PCIe.

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:

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

  Enter the information according to your environment and your issue.
- 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.
- 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

Synopsys is committed to creating an inclusive environment where every employee, customer, and partner feels welcomed. We are reviewing and removing exclusionary language from our products and supporting customer-facing collateral. Our effort also includes internal initiatives to remove biased language from our engineering and working environment, including terms that are embedded in our software and IPs. At the same time, we are working to ensure that our web content and software applications are usable to people of varying abilities. You may still find examples of non-inclusive language in our software or documentation as our IPs implement industry-standard specifications that are currently under review to remove exclusionary language.

1 RC/EP DUT Integration

This guide describes the steps to integrate an RC/EP DUT in PIPE or Serial mode to achieve the following objectives:

- Connect the DUT and VIP using macro/interface-based approach.
- Configure the VIP and ensure the link is up by running the sanity test by sending traffic over the link.

This chapter discusses the following topics:

- Integrating the PCIe VIP Testbench Components with a DUT
- Validating the integration

1.1 Integrating the PCIe VIP Testbench Components with a DUT

Perform the following steps to integrate a DUT with VIP:

- Step 1: Install Unified VIP Example
- Step2: Build the “Top” File
- Step 3: Understand the Integration Plan
- Step 4: Start the Integration
- Step 5: Set Up the Environment

Step 1: Install Unified VIP Example

Prerequisites:

Ensure that you have downloaded the PCIe VIP model and have completed the license set up and access to the VCS simulator.

Procedure:

To install the example, perform the steps that follows:

1. Install the example. At the command line, invoke the following:

```
$DESIGNWARE_HOME/bin/dw_vip_setup -path <design dir> -e
pcie_svt/tb_pcie_svt_uvm_unified_vip_sys -svtb
```

Here, `<design_dir>` is the location where the example is installed.

2. Change working directory to the installed example:

   ```
   cd <design dir>/examples/sverilog/pcie_svt/tb_pcie_svt_uvm_unified_vip_sys
   ```

3. The **README** contains a description of the example.
4. Run the VIP back to back example to get familiar with the compilation steps/file/package inclusions and various simulation logs.

   Command:

   ```
   gmake USE_SIMULATOR=vcsvlog base_serdes_test WAVES=fsdb // for generating fsdb
   ```

Table 1-1 USE_SIMULATOR Option Based on Different Flows

Flow	USE_SIMULATOR option
VCSMX three step flow	`vcsmxvlog`
MTI Verilog flow	`mtivlog`
VCS 2 step flow	`vcsvlog`
NC Verilog flow	`ncvlog`
Partition compile flow	`vcsmxpcvlog`
vcsmx PIP(Precompile IP) flow	`vcsmxpipvlog`
VC mixed flow	`ncmvlog`

Keep back to back compile and simulation log ready. These are required to match the customer's compile log and simulation log while integrating the VIP with DUT.

Step2: Build the “Top” File

Include and import the necessary PCIe and UVM packaged in the top file of user in the same way as done in the **top.sv** file of the installed unified example.

In Customer's "Top file" (where all necessary imports of pkgs are done - PCIe, UVM)

```
View of back to back top.sv file:
/*********************************************************/

`timescale 1 ns/1 fs

/** Enable Gen4 */
`define SVT_PCIE_ENABLE_GEN4

/** Defines required for PCIE SVC */
`define EXPERTIO_PCIESVC_GLOBAL_SHADOW_PATHtest_top.global_shadow0
`define SVC_RANDOM_SEED_SCOPEtest_top.global_random_seed

/** Include the PCIE SVT UVM package */
`include "svt_pcie.uvm.pkg"

//=============================================================================
module test_top;

timeunit 1ns; timeprecision 1fs;

/** Import UVM Package */ import uvm_pkg::*;
`include "uvm_macros.svh"

      /** Include PCIE VIP UVM Packages */
      `include "import_pcie_svt_uvm_pkgs.svi"

/** Include Util parms */
`include
`SVC_SOURCE_MAP_SUITE_UTIL_V(pcie_svc,PCIE,latest,svc_util_parms)
`include
`SVC_SOURCE_MAP_SUITE_MODEL_MODULE(pcie_svc,Include,latest,pciesvc_parms)

/** Include the PCIE Device ENV */
`include "pcie_device_unified_vip_env.sv"

/** Signal to generate the reset */ bit reset;
/*********************************************************/
```

The following files are very important for RC/EP DUT integration:

- **Topology file**, and
- **hdl_interconnect_macros.sv**

Topology File

The top-level testbench, that is, "top.sv" file includes "top_test.sv" file. The "top_test.sv" file is generated by prescript by copying from one of the following files based on the test case and the interface used.

List of topology files present in back to back example:

Table 1-2 List of Topology Files

Topology File Name	Topology description
`top.pcie_multi_link_topology.sv`	Multi-link are different types of link, such as PIPE, SerDes, PMA, and others operating simultaneously
`top.pcie_pipe5_topology.sv`	PIPE5 link between 2 VIP instances
`top.pcie_pma_topology.sv`	PMA Link formation with 2 VIP instances
`top.pcie_serdes_topology.sv`	SerDes link between 2 VIP instances
`top.pcie_pie8_eq_topology.sv`	PIE-8 interface topology
`top.pcie_pipe_topology.sv`	PIPE link between 2 VIP instances
`top.pcie_serdes5_topology.sv`	SerDes5 link between 2 VIP instances

You can pick up the reference topology file based on your requirement. Here, the integration has been explained with the **top.pcie_serdes_topology.sv** file for serial topology. The interfaces/wiring connections' definitions of Interconnect Macros (ICM) used in the above files are present in the **hdl_interconnect_macros.sv** file.

hdl_interconnect_macros.sv File:

This file contains definition of macros used to create and connect Unified PCIe VIP module (and interface) instances. The macros are defined for all topologies and connection types. The use of these macros helps in easing the integration process.

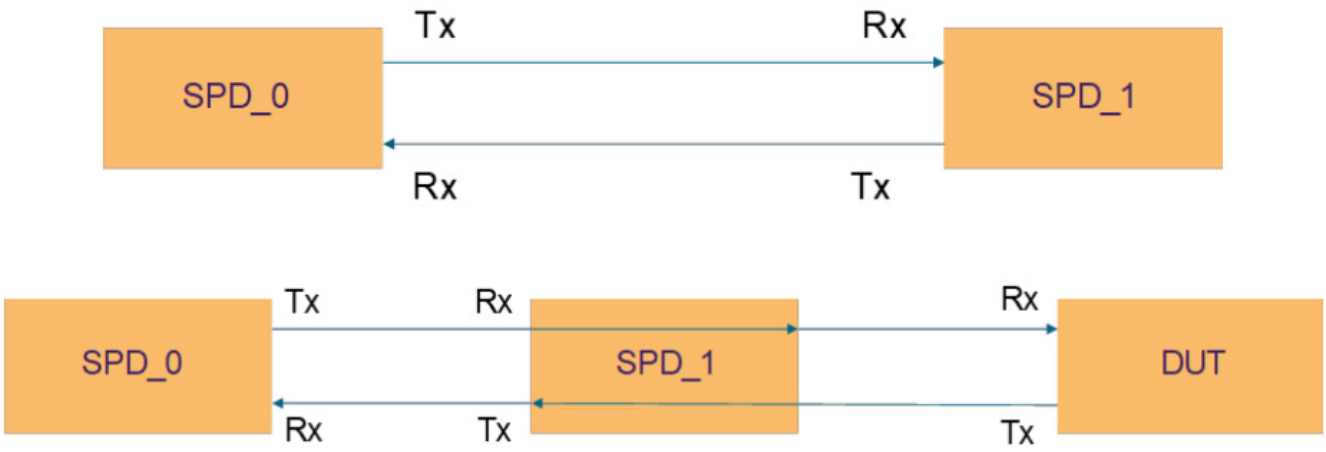
- `include **hdl_interconnect** file (in **"Top file"** of customer right after including PCIe pkgs)
- `include **topology (serdes/pipe, etc.)** file in **"Top file"** of customer

hdl_interconnect_macros.sv and **top.pcie_serdes_topology.sv** file (one of the topology files-serial topology in this case) comes from installed example - copy to your area so that it can compile.

Attention Before connecting the DUT to VIP signals, ensure that you first check and validate that you can compile the VIP in your environment.

Step 3: Understand the Integration Plan

Figure 1-1 Integration Plan



In back to back VIP example, the two VIP instances (SPD 0 and SPD 1) are connected as illustrated in Figure 1-1. To integrate the VIP with the DUT, turn off the functionality of SPD 1 and make it a bundle of wires. Then, connect the Tx and Rx of DUT with SPD 1 VIP instance. Hence, Tx of SPD 0 gets connected to DUT Rx and Tx of DUT gets connected to Rx of VIP instance SPD 0. The data simply passes through the SPD 1 instance of VIP.

VIP Instantiation and DUT Connections in a Topology File

1. Interface based approach
   a. Creating instances of PCIe VIP HDL agent and its associated interface and forming the link using two instances of interface is done directly in the file without the use of helper macros.
2. Macro based approach (**Recommended**):
   a. Macros for connecting the VIP instance are defined in the **hdl_interconnect_macros.sv** file.
   b. User can either use the macros as it is or change the definition in above file. Macros ease the usage by wrapping the entire process in one command.
   c. Example: **`SVT_PCIE_ICM_SER_SER_LINK`** is used for doing the serial connections

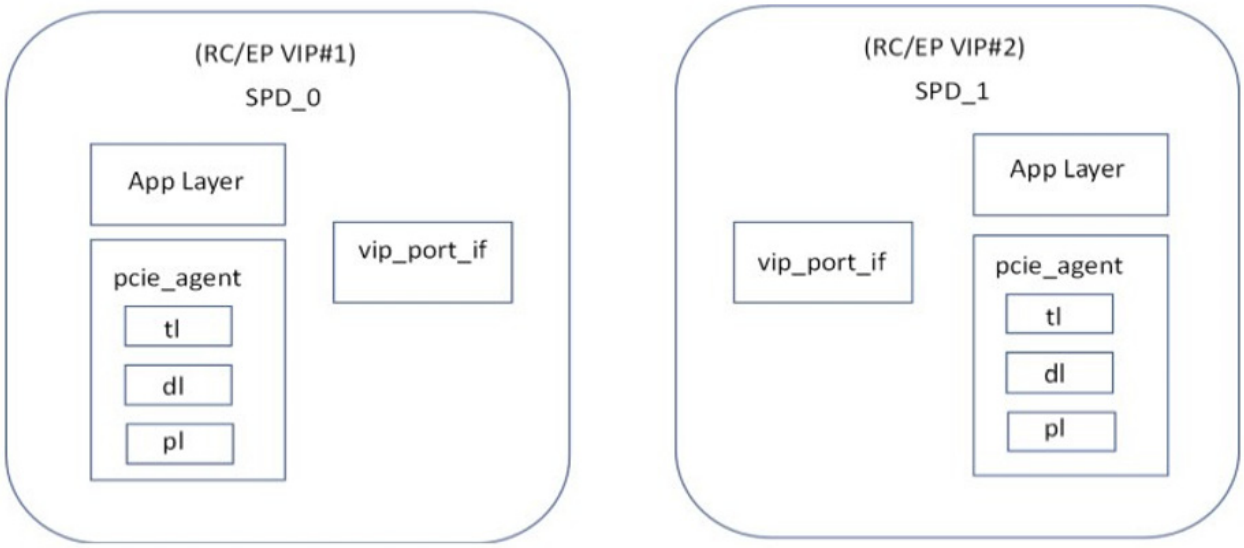
   `SVT_PCIE_ICM_PIPE_PIPE_LINK` is used for the PIPE connections. See description in `hdl_interconnect_macros.sv` file.

- Connections in a topology file
  - Instantiate two instances of dual mode VIP supporting various PCIe signal interconnects.

```
// This is the interface of VIP acting as EP/RC for RC/EP DUT svt_pcie_if
port_if_0(clkreq_n[0], wake_n); svt_pcie_single_port_device_agent_hdl
spd_0 (port_if_0);
// This is second VIP instance, will act as Application VIP
svt_pcie_if port_if_1(clkreq_n[0], wake_n);
svt_pcie_single_port_device_agent_hdl spd_1(port_if_1);
```

Note This is interface based approach.

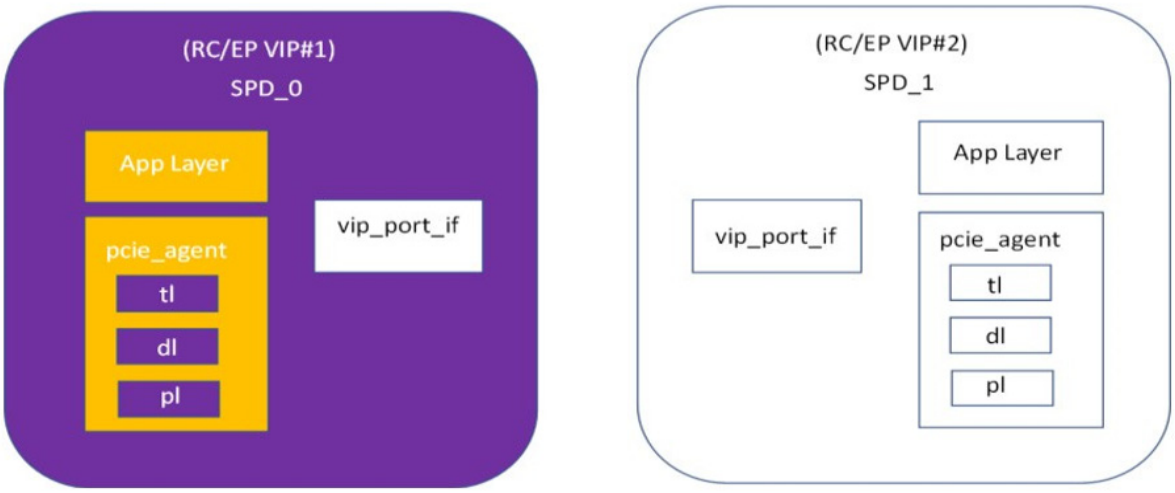
Figure 1-2 VIP and Interface Instantiation



- Compile-time settings of first VIP instance.

```
defparam spd_0.SVT_PCIE_UI_DISPLAY_NAME = " spd_0.";
defparam spd_0.SVT_PCIE_UI_ENABLE_CFG_BLOCK= 0;
defparam spd_0.SVT_PCIE_UI_PCIE_SPEC_VER =
`SVT_PCIE_UI_PCIE_SPEC_VER_4_0;
defparam spd_0.SVT_PCIE_UI_PIPE_SPEC_VER =
`SVT_PCIE_UI_PIPE_SPEC_VER_4_2;
defparam spd_0.SVT_PCIE_UI_NUM_PHYSICAL_LANES= 1;
defparam spd_0.SVT_PCIE_UI_HIERARCHY_NUMBER= 0;
defparam spd_0.SVT_PCIE_UI_MPIPE= 0;
`ifdef SERDES_TB
defparam spd_0.SVT_PCIE_UI_PHY_INTERFACE_TYPE=
`SVT_PCIE_UI_PHY_INTERFACE_TYPE_SERDES;
`else
defparam spd_0.SVT_PCIE_UI_PHY_INTERFACE_TYPE=
`SVT_PCIE_UI_PHY_INTERFACE_TYPE_PIPE;
`endif
defparam spd_0.SVT_PCIE_UI_DEVICE_IS_ROOT= 1; // for EP DUT, VIP is acts
as a root complex.

defparam spd_0.SVT_PCIE_UI_ENABLE_SHADOW_MEMORY_CHECKING= 0;
defparam spd_0.SVT_PCIE_UI_CONNECT_ACTIVE_VIP= 1;
defparam spd_0.SVT_PCIE_UI_PIPE_CLK_FROM_MAC= 0;
defparam spd_0.SVT_PCIE_UI_CONNECT_UPSTREAM_PORT_MONITOR = 0;
defparam spd_0.SVT_PCIE_UI_CONNECT_DOWNSTREAM_PORT_MONITOR = 0;
```

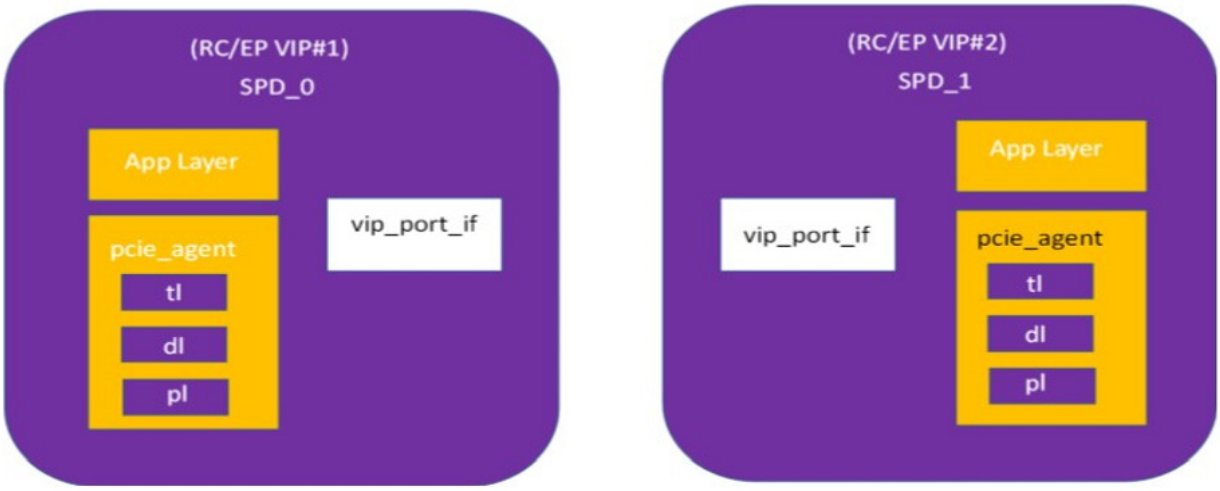
Figure 1-3 VIP #1 Settings



- Compile-time settings of second VIP Instance.

```
defparam spd_1.SVT_PCIE_UI_DISPLAY_NAME = "spd_1.";
defparam spd_1.SVT_PCIE_UI_ENABLE_CFG_BLOCK= 0;
defparam spd_1.SVT_PCIE_UI_PCIE_SPEC_VER =
`SVT_PCIE_UI_PCIE_SPEC_VER_4_0;
defparam spd_1.SVT_PCIE_UI_PIPE_SPEC_VER =
`SVT_PCIE_UI_PIPE_SPEC_VER_4_2;
defparam spd_1.SVT_PCIE_UI_NUM_PHYSICAL_LANES= 1;
defparam spd_1.SVT_PCIE_UI_HIERARCHY_NUMBER= 0;
defparam spd_1.SVT_PCIE_UI_MPIPE= 1;
`ifdef SERDES_TB
defparam spd_1.SVT_PCIE_UI_PHY_INTERFACE_TYPE=
`SVT_PCIE_UI_PHY_INTERFACE_TYPE_SERDES;
`else
defparam spd_1.SVT_PCIE_UI_PHY_INTERFACE_TYPE=
`SVT_PCIE_UI_PHY_INTERFACE_TYPE_PIPE;
`endif

defparam spd_1.SVT_PCIE_UI_DEVICE_IS_ROOT = 0;
defparam spd_1.SVT_PCIE_UI_ENABLE_SHADOW_MEMORY_CHECKING= 0;
defparam spd_1.SVT_PCIE_UI_CONNECT_ACTIVE_VIP= 1;
defparam spd_1.SVT_PCIE_UI_PIPE_CLK_FROM_MAC= 0;
defparam spd_1.SVT_PCIE_UI_CONNECT_UPSTREAM_PORT_MONITOR = 0;
defparam spd_1.SVT_PCIE_UI_CONNECT_DOWNSTREAM_PORT_MONITOR = 0;
```

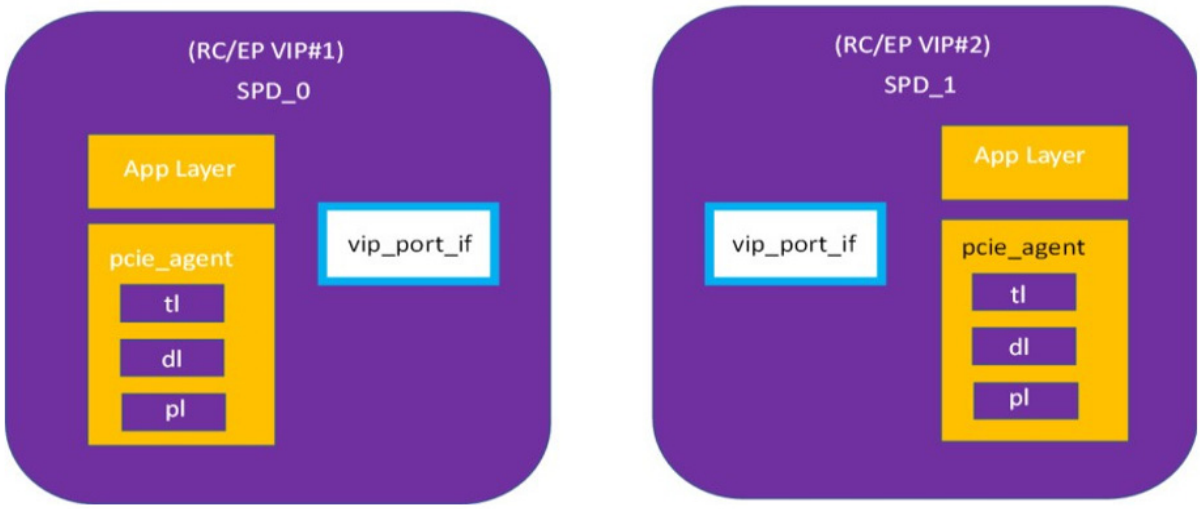
Figure 1-4 RC and EP VIP Settings



- Map the parameter values to interface variables by invoking `update_if_variables(...)` for each VIP instance.

```
initial begin spd_0.update_if_variables(4'h0, //port_id 4'h0, //link_id
"uvm_test_top" // UVM class instance to which the Active VIP

Interface will be targeted targeted. Not used for now

,"" // UVM class instance to which the Passive VIP Interface will be
,1 // Bit to enable whether link_id value should be used in

construction of string used to store the ACTIVE VIP virtual interface
handle in uvm_config_db
,0 // Bit to enable whether link_id value should be used in construction
of string used to store the Passive VIP virtual interface handle in
uvm_config_db
);

spd_1.update_if_variables(4'h1, //port_id
4'h0, //link_id
"uvm_test_top" // UVM class instance to which the Active VIP

Interface will be targeted targeted. Not used for now

,"" // UVM class instance to which the Passive VIP Interface will be
,1 // Bit to enable whether link_id value should be used in construction
of string used to store the ACTIVE VIP virtual interface handle in
uvm_config_db
,0 // Bit to enable whether link_id value should be used in construction
of string used to store the Passive VIP virtual interface handle in
uvm_config_db
);
end
```

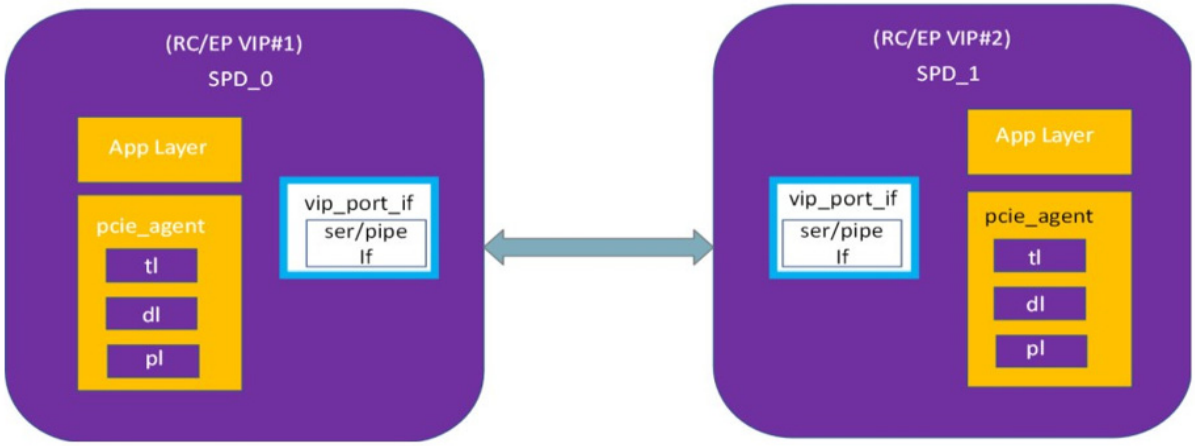
Figure 1-5 VIP Settings Mapped to Interface Variables



- Cross-connect signals of one interface instance to another interface instance using macro.

```
//connect PCIe Serial interfaces of VIP instances to form PCIe Serial
link.

//Macro assumes the arguments as link_number, a serial vip instance ,
another serial vip instance
`SVT_PCIE_ICM_SER_SER_LINK(0, spd_0,spd_1)
//OR
//cross connect PCIe SERIAL interfaces of VIP instances to form PCIe PIPE
link

// Macro assumes the arguments as link_number, vip instance representing
phy side i.e. MPIPE=0 , vip instance representing Controller (MPIPE=1)
`SVT_PCIE_ICM_PIPE_PIPE_LINK(0, spd_0, spd_1)
```

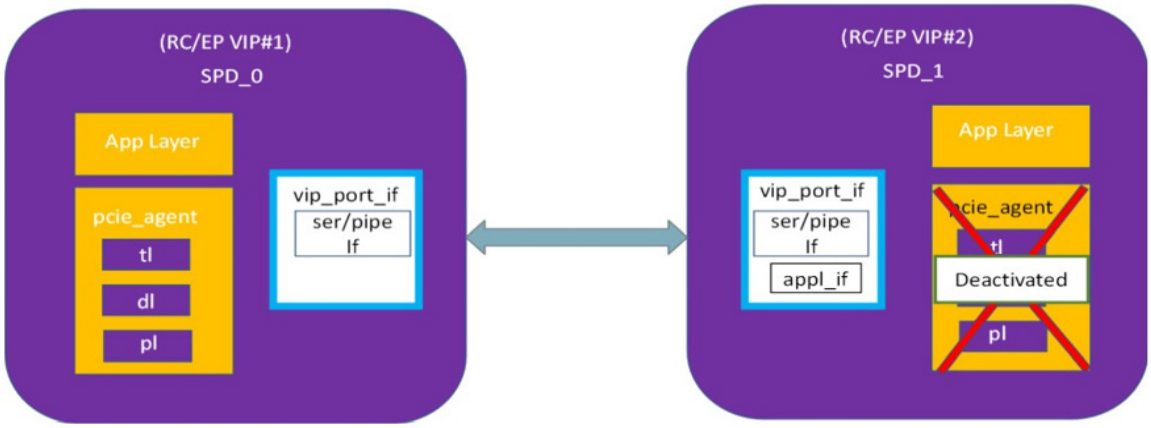
Figure 1-6 RC and EP VIP Interface Cross Connected



- Deactivate the TL, DL and PL stack of EP/RC VIP#2 (VIP acting as a DUT) instance by setting the interface type to `SVT_PCIE_UI_PHY_INTERFACE_TYPE_APP.

```
// Application VIP connection and configuration and DUT connection
defparam spd_1.SVT_PCIE_UI_PHY_INTERFACE_TYPE =
`SVT_PCIE_UI_PHY_INTERFACE_TYPE_APP;
```

Figure 1-7 VIP#2 Converted to Application VIP



  - This setting will disable the TL, DL and PL stack of the VIP#2 that is acting as a DUT (`spd_1`).
  - With this change, the VIP#2 now becomes an Application VIP which is connected to the DUT through application interface `app_if`. The Application interface contains clock and reset only.
- Supply clock and reset to Application VIP.

```
// Supply clock and reset to the VIP whose application layer is active but
TL,DL and PL stack is INACTIVE

always @(*)spd_1.vip_port_if.app_if.reset =
`CUSTOMER_DUT_INSTANCE.app_or_tl_reset; // input to VIP instance to reset
the app layer which is driving the DUT TL,DL,PL stack

always @(*) spd_1.vip_port_if.app_if.appl_clk =
`CUSTOMER_DUT_INSTANCE.free_running_or_gated_clock; //input to VIP
instance for app layer which is driving the DUT TL,DL,PL stack.
```

  - The RC/EP VIP#1 interface (`ser/pipe_if`) is connected to VIP#2 Interface (`ser/pipe_if`) which in turn gets connected to DUT as shown in the code below. This makes the VIP#2 interface connection just a pass through (see Figure 1-7).

Note The above connection is equivalent to EP/RC DUT directly connected to RC/EP VIP.

  - The above Macro based approach ensures proper connection between RC and EP interface which are defined in macros provided in *hdl_interconnect.macros.sv* file

- Additionally, you can run any test in demo (VIP-VIP) mode without modifying the connection.

Note

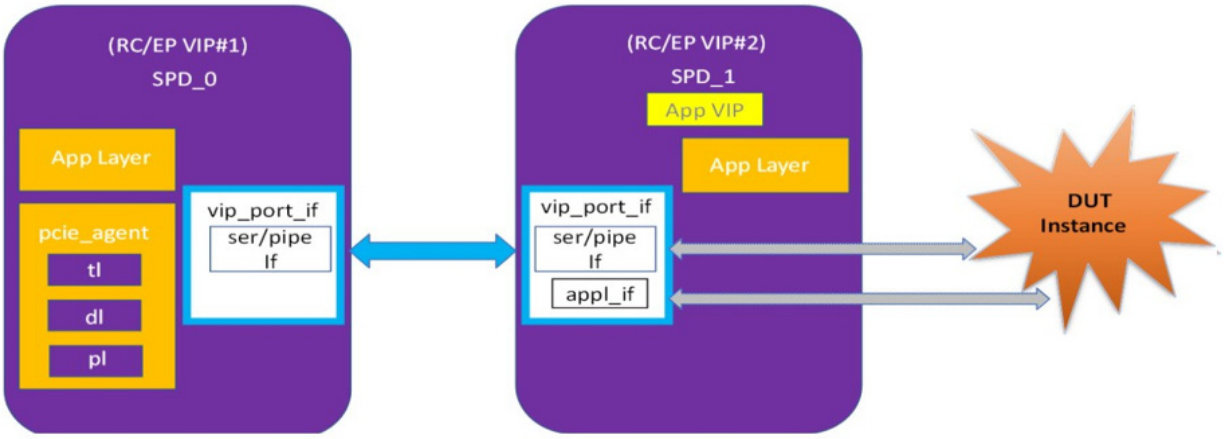
The Tx signals of VIP#2 interface (`spd_1.vip_port_if`) must be connected to Tx signals of the DUT. Similarly, Rx of VIP#2 interface (`spd_1.vip_port_if`) to Rx signals of the DUT. The Tx-Rx cross-connection is already done in the macro (`SVT_PCIE_ICM_SER_SER_LINK/SVT_PCIE_ICM_PIPE_PIPE_LINK`) used to connect the two interfaces.

- Connect the EP/RC DUT to RC/EP VIP through `spd_1` instance of VIP.

```
`ifdef SERDES_TB
// per lane Serial interconnect code to be replicated as per DUT
requirement in terms of number of lanes. Code placed below is only
applicable for lane 0
always @(*) `CUSTOMER_DUT_INSTANCE.rx_datap_0 =
spd_1.vip_port_if.ser_if.rx_datap_0 ; always @(*)
`CUSTOMER_DUT_INSTANCE.rx_datan_0 = spd_1.vip_port_if.ser_if.rx_datan_0 ;
always @(*)spd_1.vip_port_if ser_if.tx_datap_0 =
`CUSTOMER_DUT_INSTANCE.tx_datap_0 ;

always @(*)spd_1.vip_port_if ser_if.tx_datan_0 =
`CUSTOMER_DUT_INSTANCE.tx_datan_0 ;
`else
// PIPE Interconnect code irrespective of number of lanes
always @(*)`CUSTOMER_DUT_INSTANCE.pclk =spd_1.vip_port_if pipe_if pclk;
always @(*)`CUSTOMER_DUT_INSTANCE.max_pclk =
spd_1 vip_port_if pipe_if max_pclk;
always @(*)`CUSTOMER_DUT_INSTANCE.reset = common_pwr_on_reset;
always @(*)spd_1.vip_port_if.pipe_if.rate= `CUSTOMER_DUT_INSTANCE rate;
always @(*)spd_1.vip_port_if.pipe_if.pclk_rate=
`CUSTOMER_DUT_INSTANCE.pclk_rate;
. . . . . .
. . . . . .
// per lane PIPE interconnect code to be replicated as per DUT requirement
in terms of number of lanes always @(*)`CUSTOMER_DUT_INSTANCE.rx_data_0 =
spd_1.vip_port_if.pipe_if.rx_data_0 ;
always @(*)`CUSTOMER_DUT_INSTANCE.rx_data_k_0 =
spd_1 vip_port_if.pipe_if.rx_data_k_0 ; always
@(*)`CUSTOMER_DUT_INSTANCE.rx_status_0 =
spd_1.vip_port_if.pipe_if.rx_status_0 ; always
@(*)`CUSTOMER_DUT_INSTANCE.rx_valid_0 =
spd_1.vip_port_if.pipe_if.rx_valid_0 ;
. . . . . .
. . . . . .
```

```
always @(*) spd_1.vip_port_if.pipe_if.rx_polarity_0=
`CUSTOMER_DUT_INSTANCE.rx_polarity_0 ;
always @(*) spd_1.vip_port_if.pipe_if.tx_data_0=
`CUSTOMER_DUT_INSTANCE.tx_data_0 ;
always @(*) spd_1.vip_port_if.pipe_if.tx_data_k_0=
`CUSTOMER_DUT_INSTANCE.tx_data_k_0 ;
always @(*) spd_1.vip_port_if.pipe_if.tx_ei_code_0=
`CUSTOMER_DUT_INSTANCE.tx_ei_code_0 ;
always @(*) spd_1.vip_port_if.pipe_if.tx_compliance_0=
`CUSTOMER_DUT_INSTANCE.tx_compliance_0 ;
always @(*) spd_1.vip_port_if.pipe_if.tx_elec_idle_0=
`CUSTOMER_DUT_INSTANCE.tx_elec_idle_0 ;
always @(*) spd_1.vip_port_if.pipe_if.tx_data_valid_0=
`CUSTOMER_DUT_INSTANCE.tx_data_valid_0 ;
. . . . . .
. . . . . .
```

Figure 1-8 VIP-DUT Connection



Topology Reference File:

```
`ifndef GUARD_PCIE_SERDES_TOPOLOGY
`define GUARD_PCIE_SERDES_TOPOLOGY
/**
*  Abstract:
*  Serdes Link formation between 2 VIP instances using helper macros
*  provided in hdl_interconnect_macros.sv
*/

/** Define the elements of the test registry*/
`include "ts.base_serdes_test.sv"
`include "ts.address_translation_test.sv"

`SVT_PCIE_ICM_CREATE_PORT_INST(0,0)
`SVT_PCIE_ICM_CREATE_PORT_INST(0,1)
`SVT_PCIE_ICM_CREATE_LINK(0,spd_0,spd_1) // Create link 0
`SVT_PCIE_ICM_SER_SER_LINK(0,spd_0,spd_1) //cross connect relevant
(serdes) interface of link created in above step
`SVT_PCIE_ICM_DO_CONDITIONAL_INTERCONNECT(0,spd_0,1,spd_1) // Certain
signal inter-connections are dependent on VIP parameter settings. This
macro handles those connections and simplified the topology file
//comment this macro

defparam spd_0.SVT_PCIE_UI_PCIE_SPEC_VER = `SVT_PCIE_UI_PCIE_SPEC_VER_3_0;
defparam spd_0.SVT_PCIE_UI_PIPE_SPEC_VER = `SVT_PCIE_UI_PIPE_SPEC_VER_4_3;
defparam spd_1.SVT_PCIE_UI_PCIE_SPEC_VER = `SVT_PCIE_UI_PCIE_SPEC_VER_3_0;
defparam spd_1.SVT_PCIE_UI_PIPE_SPEC_VER =
`SVT_PCIE_UI_PIPE_SPEC_VER_4_3;

defparam spd_0.SVT_PCIE_UI_PHY_INTERFACE_TYPE =
`SVT_PCIE_UI_PHY_INTERFACE_TYPE_SERDES;
defparam spd_0.SVT_PCIE_UI_DEVICE_IS_ROOT = 1;
defparam spd_0.SVT_PCIE_UI_NUM_PHYSICAL_LANES= 8;


defparam spd_1.SVT_PCIE_UI_PHY_INTERFACE_TYPE =
`SVT_PCIE_UI_PHY_INTERFACE_TYPE_SERDES;// =
`SVT_PCIE_UI_PHY_INTERFACE_TYPE_APP;
defparam spd_1.SVT_PCIE_UI_DEVICE_IS_ROOT = 0;
defparam spd_1.SVT_PCIE_UI_NUM_PHYSICAL_LANES= 8;

`endif //`ifndef GUARD_PCIE_SERDES_TOPOLOGY
```

```
/*******************Dummy Connection***********************************/
// Supply clock and reset to the VIP whose application layer is active but
TL,DL and PL stack is INACTIVE
/*
always @(*) spd_1.vip_port_if.app_if.reset = 0;//common_pwr_on_reset;
always @(*) spd_1.vip_port_if.app_if.appl_clk =
0;//u_pcie_device.core_clk;

reg [7:0] rxp;// = 4'b0000; reg [7:0] rxn;

always @(*) rxp[7] = spd_1.vip_port_if.ser_if.rx_datap_7; always @(*)
rxn[7] = spd_1.vip_port_if.ser_if.rx_datan_7; always @(*) rxp[6] =
spd_1.vip_port_if.ser_if.rx_datap_6; always @(*) rxn[6] =
spd_1.vip_port_if.ser_if.rx_datan_6; always @(*) rxp[5] =
spd_1.vip_port_if.ser_if.rx_datap_5; always @(*) rxn[5] =
spd_1.vip_port_if.ser_if.rx_datan_5; always @(*) rxp[4] =
spd_1.vip_port_if.ser_if.rx_datap_4; always @(*) rxn[4] =
spd_1.vip_port_if.ser_if.rx_datan_4; always @(*) rxp[3] =
spd_1.vip_port_if.ser_if.rx_datap_3; always @(*) rxn[3] =
spd_1.vip_port_if.ser_if.rx_datan_3; always @(*) rxp[2] =
spd_1.vip_port_if.ser_if.rx_datap_2; always @(*) rxn[2] =
spd_1.vip_port_if.ser_if.rx_datan_2; always @(*) rxp[1] =
spd_1.vip_port_if.ser_if.rx_datap_1; always @(*) rxn[1] =
spd_1.vip_port_if.ser_if.rx_datan_1; always @(*) rxp[0] =
spd_1.vip_port_if.ser_if.rx_datap_0; always @(*)   rxn[0]  =
spd_1.vip_port_if.ser_if.rx_datan_0;

always @(*) spd_1.vip_port_if.ser_if.tx_datap_3 = DUT.txp[3] ; always @(*)
spd_1.vip_port_if.ser_if.tx_datap_2 = DUT.txp[2] ; always @(*)
spd_1.vip_port_if.ser_if.tx_datap_1 = DUT.txp[1] ; always @(*)
spd_1.vip_port_if.ser_if.tx_datap_0        = DUT.txp[0] ;

always @(*) spd_1.vip_port_if.ser_if.tx_datan_3 = DUT.txn[3] ; always @(*)
spd_1.vip_port_if.ser_if.tx_datan_2 = DUT.txn[2] ; always @(*)
spd_1.vip_port_if.ser_if.tx_datan_1        = DUT.txn[1] ;
always @(*) spd_1.vip_port_if.ser_if.tx_datan_0 = DUT.txn[0] ;

always @(*) spd_1.vip_port_if.ser_if.tx_datap_7 = DUT.txp[7] ; always @(*)
spd_1.vip_port_if.ser_if.tx_datap_6 = DUT.txp[6] ; always @(*)
spd_1.vip_port_if.ser_if.tx_datap_5 = DUT.txp[5] ; always @(*)
spd_1.vip_port_if.ser_if.tx_datap_4        = DUT.txp[4] ;

always @(*) spd_1.vip_port_if.ser_if.tx_datan_7 = DUT.txn[7] ; always @(*)
spd_1.vip_port_if.ser_if.tx_datan_6        = DUT.txn[6] ;

always @(*)spd_1.vip_port_if.ser_if.tx_datan_5= DUT.txn[5] ; always @(*)
spd_1.vip_port_if.ser_if.tx_datan_4= DUT.txn[4] ;
```

```
USER_DUT DUT(
  .refclk_p                    (refclk_p),
 .refclk_n                    (refclk_n),
  .txp                         (txp),
  .txn                         (txn),
  .rxp                         (rxp[7:0]),
  .rxn                         (rxn[7:0]),

      )
/*********************Dummy
Connection***********************************/
```

Note

Here, you need to edit the topology file as per your requirement. See reference topology file. The connections are made as explained in the above diagrams.

Setting **`SVT_PCIE_UI_PHY_INTERFACE_TYPE** to **`SVT_PCIE_UI_PHY_INTERFACE_TYPE_APP** disables the functionality of **SPD 1** VIP instance.

The definition of the macros present in topology file can be found in **hdl_interconnect_macros.sv** file.

Note

Here, the example is presented for Serial topology. Similar process can be followed for other topologies.

Step 4: Start the Integration

Open back to back compile log to check the **incdir**.

incdir needs to be handpicked and added in the user's script.

```
Command: vcs -l ./logs/compile.log -Mdir=./output/csrc +incdir+/<DESIGNWARE_HOME_PATH>/design_dir/src/sverilog/vcs \
+incdir+/<DESIGNWARE_HOME_PATH>/design_dir/include/sverilog \
+incdir+/<DESIGNWARE_HOME_PATH>/design_dir/src/verilog/vcs \
+incdir+/<DESIGNWARE_HOME_PATH>/design_dir/include/verilog \
-ntb_opts uvm -full64 -sverilog +define+UVM_DISABLE_AUTO_ITEM_RECORDING +define+UVM_PACKER_MAX_BYTES=8192 \
+define+SVT_PCIE_ENABLE_GEN5 +define+SVT_PCIE_ENABLE_PIPE5 +define+SVT_PCIE_ENABLE_SERDES_ARCH \
-debug_acc -timescale=1ns/1fs +libext+.v+.sv -y /<DESIGNWARE_HOME_PATH>/design_dir/src/verilog/vcs \
-y /<DESIGNWARE_HOME_PATH>/design_dir/src/sverilog/vcs \
+define+UVM_VERDI_NO_COMPWAVE -debug_acc -P pli.tab msglog.o +define+SVT_FSDB_ENABLE \
+define+WAVES_FSDB +define+WAVES="fsdb" +plusarg_save -debug_access+pp+dmptf+thread \
-debug_region=cell+encrypt -notice -P /global/apps/verdi_2019.06-SP2/share/PLI/VCS/LINUX64/novas.tab \
/global/apps/verdi_2019.06-SP2/share/PLI/VCS/LINUX64/pli.a +define+SVT_UVM_TECHNOLOGY \
+define+SYNOPSYS_SV +incdir+/<DESIGNWARE_HOME_PATH>/design_dir/examples/sverilog/pcie_svt/tb_pcie_svt_uvm_unified_vip_sys/. \
+incdir+/<DESIGNWARE_HOME_PATH>/design_dir/examples/sverilog/pcie_svt/tb_pcie_svt_uvm_unified_vip_sys/../../env \
+incdir+/<DESIGNWARE_HOME_PATH>/design_dir/examples/sverilog/pcie_svt/tb_pcie_svt_uvm_unified_vip_sys/../env \
+incdir+/<DESIGNWARE_HOME_PATH>/design_dir/examples/sverilog/pcie_svt/tb_pcie_svt_uvm_unified_vip_sys/env \
+incdir+/<DESIGNWARE_HOME_PATH>/design_dir/examples/sverilog/pcie_svt/tb_pcie_svt_uvm_unified_vip_sys/dut \
+incdir+/<DESIGNWARE_HOME_PATH>/design_dir/examples/sverilog/pcie_svt/tb_pcie_svt_uvm_unified_vip_sys/hdl_interconnect \
+incdir+/<DESIGNWARE_HOME_PATH>/design_dir/examples/sverilog/pcie_svt/tb_pcie_svt_uvm_unified_vip_sys/lib \
+incdir+/<DESIGNWARE_HOME_PATH>/design_dir/examples/sverilog/pcie_svt/tb_pcie_svt_uvm_unified_vip_sys/tests \
-o ./output/simvcssvlog -f top_files -f hdl_files
                        Chronologic VCS (TM)
       Version P-2019.06-SP2_Full64 -- Tue Jan 21 07:02:23 2020
               Copyright (c) 1991-2019 by Synopsys Inc.
                        ALL RIGHTS RESERVED
```

Note **design_dir** has two major folders - **src** and **include**

These files are mandatory (include in same sequence):

```
+incdir+/<Designaware_Home path>/design_dir/src/sverilog/vcs \
+incdir+/<Designaware_Home path>/design_dir/include/sverilog \
+incdir+/<Designaware_Home path>/design_dir/src/verilog/vcs \
+incdir+/<Designaware_Home path>/design_dir/include/verilog \
```

Then with **-y option** (include this also in same sequence):

```
-y /<Designaware_Home path>/design_dir/src/verilog/vcs \
-y /<Designaware_Home path>/design_dir/src/sverilog/vcs
```

The list of defines that needs to be added is based on the requirement. For more information, see *PCIe SVT UVM User Guide*.

Table 1-3 Defines to be added

S.No.	Defines	Value/Requirement
1	UVM_DISABLE_AUTO_ITEM_RECORDING	Optional **Note**: • If you are using UVM version 1.1b, 1.1c, or 1.1d, then you must define the UVM_DISABLE_AUTO_ITEM_RECORDING macro. Since PCIe is a pipelined protocol (that is, the previous transaction does not necessarily complete before another transaction is started), the PCIe VIP handles triggering of the begin and end events and transaction recording. The PCIe VIP does not use the UVM automatic transaction begin and end event triggering feature. If UVM_DISABLE_AUTO_ITEM_RECORDING is not defined, the VIP issues a fatal error. • If you are using UVM version 1.2 or newer, you do not need to set the macro. • When the UVM_DISABLE_AUTO_ITEM_RECORDING macro is set, recording is disabled for all VIPs in the design.
2	UVM_PACKER_MAX_BYTES	8192 **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 1-3 Defines to be added

S.No.	Defines	Value/Requirement
3	SVT_PCIE_ENABLE_GEN5	Remove if not required
4	SVT_PCIE_ENABLE_PIPE5	Remove if not required
5	SVT_PCIE_ENABLE_SERDES_ARCH	Remove if not required
6	SVT_PCIE_ENABLE_GEN4	For Enabling Gen 4

Compile after performing the inclusions.

Attention If you find any "macros not defined" errors or any other error, check the DESIGNWARE_HOME path settings and the +incdir settings in the compile script or commands.
If the errors does not get resolved, then check the generated compile log. Ensure that everything included is in order. Match it with the back to back VIP example compile log.

Attention Each file should be compiled in the correct order. Reference can always be taken from back to back compile log. If the error is still not resolved, compare pkgs and includes in top.sv (back to back) and user top file.
Also check, if the same pkg/file is not included/imported twice.

Note VIP needs around 200ns to configure its internal registers and buffers. As shown in the top.sv file, add **reset** in user top file.

Step 5: Set Up the Environment

If all the previous steps have been performed successfully and the compilation has passed, then you can continue and set up the environment:

Refer **base test and environment in back to back example** and create your own.

Most of the things can directly be copied from the example. You can leave out the part where configurations are done for SPD 1 VIP instance (endpoint) in the pcie_device_unified_vip_env.sv file.

Attention If you create agent for second VIP in the environment, you will see a fatal message similar to the message illustrated:

```
UVM_FATAL <$USER_PATH>/vip/svt/pcie_svt/Q-2020.03/pcie_agent_svt/sverilog/src/vcs/svt_pcie_agent.svp(1195) @ 0.00000 ns: uvm_test_top.env[0].endpoint.port0 [build_tl] SVT PCIE Agent uvm_test_top.env[0].endpoint.port0 was not passed a valid HDL model instance scope "test_top.spd_1.m_app.port0" - please check that svt_pcie_device_agent::cfg.model_instance_scope matches instantiation of PCIE VIP module
```

Since you are using only one VIP instance to connect to DUT, it is not required to create agent for the other VIP.

Attention If you face any error while performing UVM get/set for the interface (**pcie_if**), match the hierarchy using run-time switch, **+UVM_CONFIG_DB_TRACE.** Check `update_if_variables` task in `hdl_interconnect_macros` file.

Example error signature:

UVM_INFO *<$USER_PATH>/my_base_test.sv(80) @ 0.00000 ns: uvm_test_top [build_phase] vif_0[0] virtual interface handle(s) is/are null. Hence skipping the remaining build_phase*

UVM_FATAL *<$USER_PATH>/my_base_test.sv(167) @ 0.00000 ns: uvm_test_top [build_phase] Number of valid links is 0 should be at least 1*

Attention Ensure that the UVM configuration DB, get (base test) and set (in topology file – `update_if_variables`) for interface are in the same library scope.

1.2 Validating the integration

After successfully completing the integration without any errors or warnings, you should be able to see symbols exchanged between VIP and DUT.

In your default sequence of the sanity test, wait for the VIP/DUT to reach L0 state.

Note Take the reference from the sequence already available in the example in the mentioned path: *<$DESIGNWARE_HOME_PATH>/<design_dir>/examples/sverilog/pcie_svt/tb_pcie_svt_uvm_unified_vip_sys/env/**pcie_device_random_traffic_sequence.sv***

- Run your sanity test.
- When the simulation is over, check if the symbol log and transaction logs are generated for the VIP.

  After successfully following the above steps without any errors or warnings, you should be able to see symbols exchanged between VIP and DUT.

- Open the symbol log. You can see the symbols transmitted and received by the VIP.
- LTSSM states should progress towards L0.

After the Link up is achieved, you can start writing your sequences to send traffic over the link.

Attention You first need to create an **enumeration** sequence to configure BARs of the DUT (EP). And, then you can send the Memory Write and Read TLPs (take reference from the sequence pointed above).

Attention If you try to send Memory writes and read without completing the enumeration part, you might face below error (for **EP DUT**):

UVM_ERROR <$USER_PATH>/src/sverilog/vcs/pciesvc_driver.sv(4976) @ 212201.48300 ns: uvm_test_top.my_env.env[0].root.driver0 [register_fail:ACTIVE_DRIVER_APP:COMPLETION:appl_driver_missing_good_status] ProcessReceivedPacket**: Received completion for MRd32 with unexpected status of 1 (Unsupported Req), expecting status GOOD**

After sending the traffic, check if the status of completion is successful.

Ensure that you do not get any UVM_ERROR, UVM_WARNING or UVM_FATAL in the **report catcher summary in the simulation log**.

Note

If you see the following error:

```
UVM_ERROR <$USER_PATH>/src/sverilog/vcs/pciesvc_driver.sv(4976) @
212201.48300 ns: uvm_test_top.my_env.env[0].root.driver0
[register_fail:ACTIVE_DRIVER_APP:COMPLETION:appl_driver_low_byte_count]
ProcessReceivedPacket: byte count field 113 is less than expected value of
116. Discarding completion.
```

then, check if TH bit becomes 1. This indicates that the Byte Enables have been replaced with Steering tag, and the DUT responds with the ST (steering tag) as the byte enables. This causes the VIP model to issue an error with byte counts.

Set TH bit 0 by adding **mem_request_seq.th == 0** in your memory request sequence. If the TH bit is set to 1, then it is required to control the Steering. However, if that is not the requirement, you can simply constrain this TH bit to zero in your random transaction.

If you do not see any other errors/warnings, your integration is complete.