Verification Continuum™

VC Verification IP
CXL Subsystem Test Suite
UVM User 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 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 Manual

This manual contains installation, setup, and usage material for SystemVerilog UVM users of the VC VIP CXL Subsystem, and is for design or verification engineers who want to verify CXL Subsystem operation using a UVM testbench written in SystemVerilog.

Web Resources

- Documentation through SolvNet: 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:

     http://www.synopsys.com/Support/GlobalSupportCenters

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 Introduction

1.1 Introduction

This guide describes the use model and high-level architecture of the CXL Testsuite for a Compute Express Link (CXL) based system.

These are the specification references:

- Compute Express Link Specification: 20190701_ComputeExpressLink_Specification_r1p1.pdf Revision 1.1 (June 2019)
- Compute Express Link Specification: CXL Specification_rev2p0_ver1p0_2020Oct26_clean_Approved.pdf Revision 2.0 (Oct 2020)
- PCIe Base specification Gen5 v1.0: NCB-PCI_Express_Base_5.0r1.0-2019-05-22.pdf
  - Key interest: Gen5 and Alternate Protocol Negotiation (APN) feature.
- PIPE Specification: phy-interface-pci-express-sata-usb30-architectures-3.1_v5_1_1.pdf
  - Key interest: SerDes Arch features

1.2 Key Features of CXL Test Suite

- CXL Host VIP with Device as DUT connected in full stack topology with one of the following signaling interfaces:
  - Serial interface
  - PIPE 5.1 (SerDes Arch) interface
- Framework for integration of Synopsys DWC CXL Controller DUT
  - Protocol signaling Interface: Serial, PIPE (SerDes Arch)
  - Application signaling Interface: CXL.io Native, CXL.Cache/Mem Native
- Alternate Protocol Negotiation (APN) with link-up in Flexbus mode
  - Up to x16 lanes supported @ Gen5 (normal mode). Degraded speed mode up to Gen3 and Degraded width up to x2
- CXL Test Application Layer over Device DUT:
  - Algorithm Supported: Algorithm 1a, 1b, 2
  - Protocol Supported: CXL.io, CXL.Cache, CXL.Mem
- CXL Device Enumeration, ability to run TS tests with /without enumeration

- CXL.io Resources (BAR's, CXL DVSEC) identification and programming
- Checks required for CXL Device like RCiEP device, CXL DVSEC Presence
- CXL Upstream port RCRB and MEMBAR0 areas are programmed and hooks added in

- Topology and Compile files-based framework for setup and test execution
- Ability to execute applicable PCIe Test Suite tests in CXL Subsystem Test Suite
  - CXL Mode (Flexbus framing)
  - PCIe Mode (PCIe Framing)
- Operational Modes
  - Demo mode: VIP-VIP topology
  - DUT mode: VIP - DUT RTL topology
- Methodology Support
  - UVM 1.1d and 1.2
- Simulator support: VCS.

  Refer to the latest version of release notes for exact simulator version support.
- PHY integration with behavioral model

  Refer to the *VC Verification IP PCIe Test Suite PHY Integration Guide* for PHY integration with few differences as mentioned below.

  $DESIGNWARE_HOME/vip/svt/pcie_test_suite_svt/latest/doc/pcie_test_suite_phy_integration_guide.pdf

  a. Reference Topology file: `topology_snps_dut_pcie_cxl_phy_iip.svi`

  b. Following are the modes of operations supported by four different compile files. Note that the reference compile files can be found under *$DESIGNWARE_HOME/vip/svt/cxl_test_suite_svt/latest/examples/sverilog/tb_dut_cxl/dut/iip_5.90a*
     - Device on PIPE, pclk output mode (PHY DUT is Device): `compile_dut_snps_pcie_cxl_phy_iip_pipe51_lpc_serdes_arch.f`
     - Host on PIPE, pclk output mde (PHY DUT is Host): `compile_dut_snps_pcie_cxl_phy_iip_host_on_pipe51_lpc_serdes_arch.f`
     - Device on PIPE, pclk input mode (PHY DUT is Device): `compile_dut_snps_pcie_cxl_phy_iip_pipe51_lpc_serdes_arch_pclk_inp.f`
     - Host on PIPE, pclk input mde (PHY DUT is Host): `compile_dut_snps_pcie_cxl_phy_iip_host_on_pipe51_lpc_serdes_arch_pclk_inp.f`

  c. Command to run the example test:

```
gmake cxl_io_random_mem_wr_rd USE_SIMULATOR=vcsvlog
SVT_CXL_SUBSYSTEM_COMPILE_FILE=compile_dut_snps_pcie_cxl_phy_iip_pipe51_lpc_se
rdes_arch.f
SVT_CXL_SUBSYSTEM_TOPOLOGY_FILE=topology_snps_dut_pcie_cxl_phy_iip.svi
IIP_VERSION=5.90a CC_CONFIG=gphy_gen5_esm_4sx16_pipe5_lpc DUT_TYPE=IIP
SVT_PCIE_MAX_LINK_WIDTH=16 SVT_PCIE_PIPE_WIDTH=2
```

- E32 PHY usage notes

  a. Framework for integration of Synopsys actual PHY DUT
     - Protocol Signaling Interface: PIPE5.1 (SerDes Arch)

b. PHY Features and Configuration Settings

- Supported topology: VIP (Serial) – PHY DUT - VIP(MAC) topology, Host on PIPE and Device on PIPE
- All Link width combination support: X1, X2, X4, X8, X16
- Speed Upgrade/Downgrade support
- Validated width/rate: PIPE 5.1, 20 bits for Gen1/2, and 40 bits for Gen3/4/5 in SerDes Arch mode
- Configuration settings required on MAC/Serial VIP:

```
pcie_cfg.pl_cfg.pclk_toggling_after_resetn_timeout=10000
pcie_cfg.pl_cfg.pipe_max_pclk_mode=2
pcie_cfg.pl_cfg.rx_message_bus_write_buffer_depth=8
pcie_cfg.pl_cfg.recovery_speed_electrical_idle_time_ns=100000
pcie_cfg.pl_cfg.detect_quiet_timeout_ns=110000
pcie_cfg.dl_cfg.aspm_timeout_cnt_limit=60000
pcie_cfg.dl_cfg.pm_timeout_cnt_limit=60000
pcie_cfg.pl_cfg.downstream_lanes_recovery_eq_phase1_timeout_ns=150000
pcie_cfg.pl_cfg.upstream_lanes_recovery_eq_phase0_timeout_ns=150000
pcie_cfg.pl_cfg.upstream_lanes_recovery_eq_phase1_timeout_ns=150000
cfg.max_mbi_response_timer_ns=22000
cfg.max_mbi_read_completion_timer_ns=22000
pcie_cfg.pl_cfg.phystatus_timeout_ns=150000
cfg.total_timeout_timer=1250000
```

Contact Synopsys support for Test Suite solution with Host, PHY as CXL DUT and HVP usage.

2 Products and VIP Dependencies

2.1 Overview

The CXL VIP Test Suite products can be seen as individual suites within the VIP product line. However, just as some VIP product suites depend on other VIP product suites, the CXL VIP Test Suites depend on other VIP product suites such as `cxl_svt`, `cxl_subsystem_svt`, `pcie_svt`, `pcie_test_suite_svt`.

2.2 CXL Test Suite Products

- **cxl_testsuite products**: The CXL VIP Test Suite product gets installed as `cxl_test_suite_svt`.

2.2.1 VIP Dependencies

The following VIP products are utilized within the CXL Test Suite testbenches, and therefore are installed in the `DESIGNWARE_HOME/vip/svt installation` tree of the user along with `cxl_test_suite_svt`:

- cxl_subsystem_svt
- cxl_svt
- pcie_svt
- pcie_test_suite_svt.

3 Installation and Setup

3.1 Installing the Product

You need to follow these steps for installation of the CXL Testsuite:

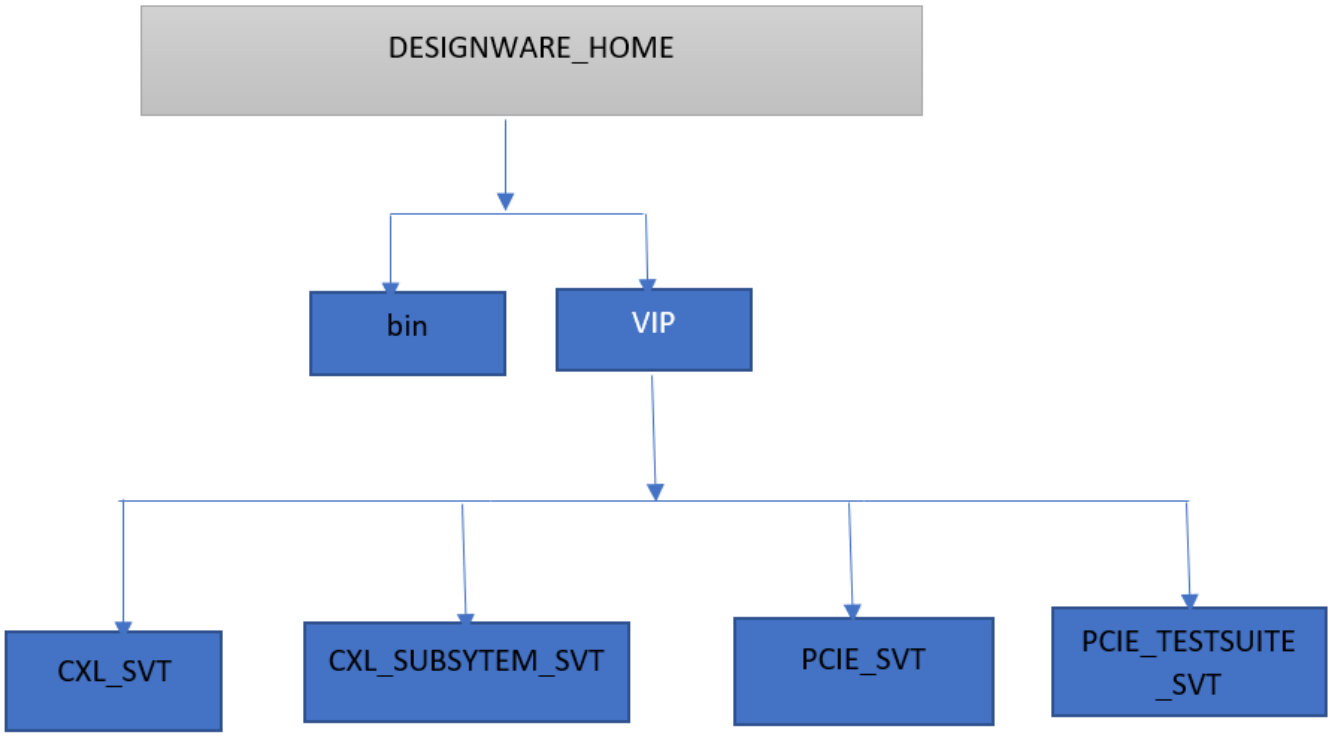
- Download the .run file (vip_cxl_test_suite_svt_<version>.run) for the CXL Testsuite.
- Set `DESIGNWARE_HOME` to the absolute path where Synopsys CXL Testsuite needs to be installed.
  - `setenv DESIGNWARE_HOME <absolute_path_to_designware_home>`
- Execute the .run file by invoking its filename.
  - For example, `vip_cxl_test_suite_svt_<>. run --dir $DESIGNWARE_HOME`.

The VIP is unpacked, and all the files and directories are installed under the path specified by the `DESIGNWARE_HOME` environment variable. The `.run` file can be executed from any directory.

Note The important step is to set the DESIGNWARE_HOME environment variable before executing the .run file.

Once the .run file is extracted, you can see the following directories inside the `DESIGNWARE_HOME`.

Figure 3-1 Structure of a DESIGNWARE_HOME Tree



3.2 Licensing

The VC VIP Test Suite for CXL Subsystem requires feature key for extracting its release run file.

Table 3-1 License Requirement

License	Supported Tests
VIP-LIBRARY2019-SVT	Both CXL 1.1 and 2.0 Tests
SUB-CXL20-TESTSUITE-SVT	Both CXL 1.1 and 2.0 Tests
SUB-CXL-TESTSUITE-SVT	Both CXL 1.1 and 2.0 Tests

For the execution of test cases, the CXL Subsystem VIP licenses are checked out. Refer to the CXL Subsystem UVM user guide for details of license keys and their usage.

CXL Subsystem VIP User Guide:

`$DESIGNWARE_HOME/vip/svt/cxl_subsystem_svt/latest/doc/cxl_subsystem_svt_uvm_user_guide.pdf`

Contact Synopsys support for installation and further information on CXL Subsystem Test Suite license.

3.3 Setting up Test Suite and Testing for Sanity of Installation

Once the Test Suite is extracted, perform the following steps to setup the test suite:

- Create a new directory called 'design_dir' to install the example inside this folder.

  ```
  % mkdir design_dir
  ```

  Note: `<design_dir>` is the project directory where the test suite is going to be installed. You can provide any name of your choice.

- Setup the CXL Subsystem Test suite in the directory named <design_dir>.

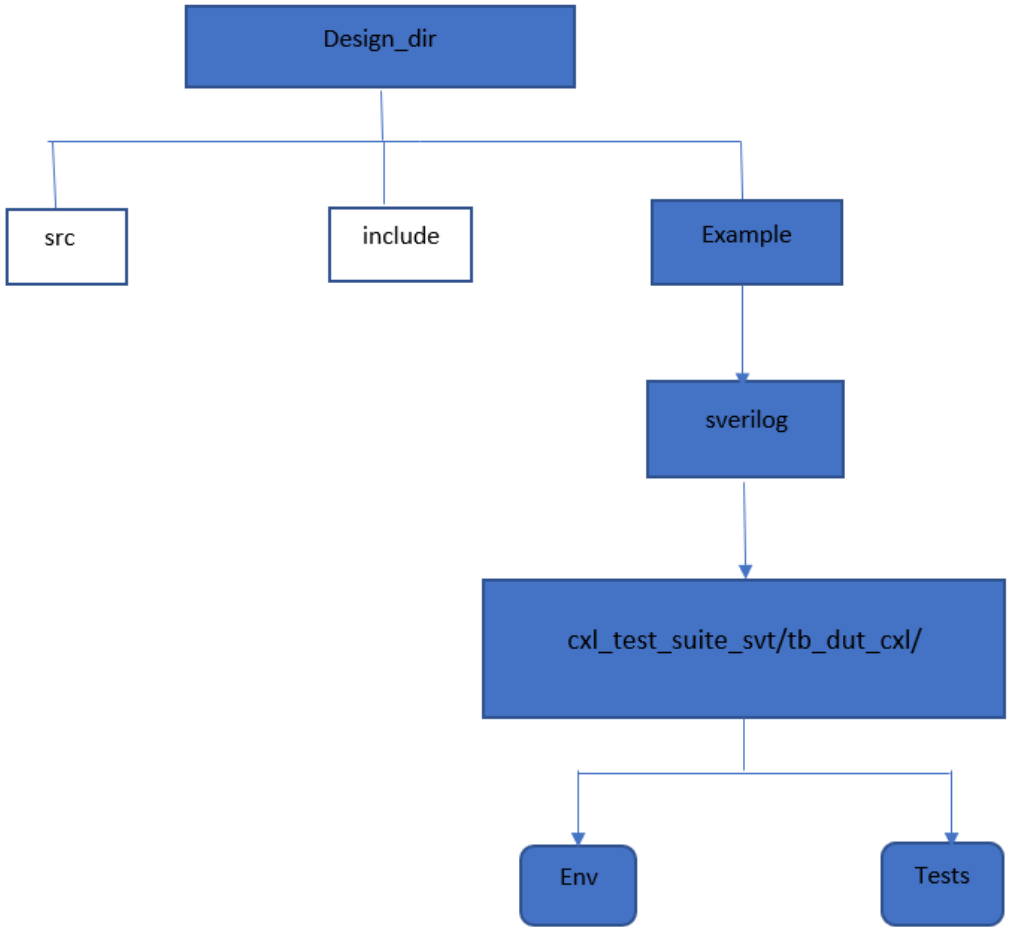
  ```
  % $DESIGNWARE_HOME/bin/dw_vip_setup -p ./design-dir -e cxl_test_suite_svt/tb_dut_cxl -svtb
  ```

  To extract pcie testsuite, you need to execute this command in the same <design_directory>.

  ```
  $DESIGNWARE_HOME/bin/dw_vip_setup -path ./design_dir -e pcie_test_suite_svt/tb_dut_pcie_gen5_file -svtb
  ```

  The above command sets up all the required files inside 'design_dir'. The `dw_vip_setup` utility creates three directories under `design_dir`, which contains all the necessary model files.

Figure 3-2 Directory Structure



The general usage of the structure is as follows:

- tb_dut_cxl

  Top level of testbench. Includes top level Verilog module definition, Makefile, and scripts supporting the execution of the testbench.
- env

  All the files that define elements of the structural UVM verification environment are found in this directory.
- tests

  The test case files(files with names of the form ts.*.sv are test case files) are found in this directory. These test case files define test classes that extend the uvm_test class, set up the configuration, create the test environment, and assign the default sequences to be run for the test case.

3.3.1 Running the Test

After installation is completed, you can run the sanity test.You can use these steps to run the test cases:

- Using the Makefile.

  In the testbench directory, the provided Makefile file is the entry point for running test cases using make (intended to be run with gmake).

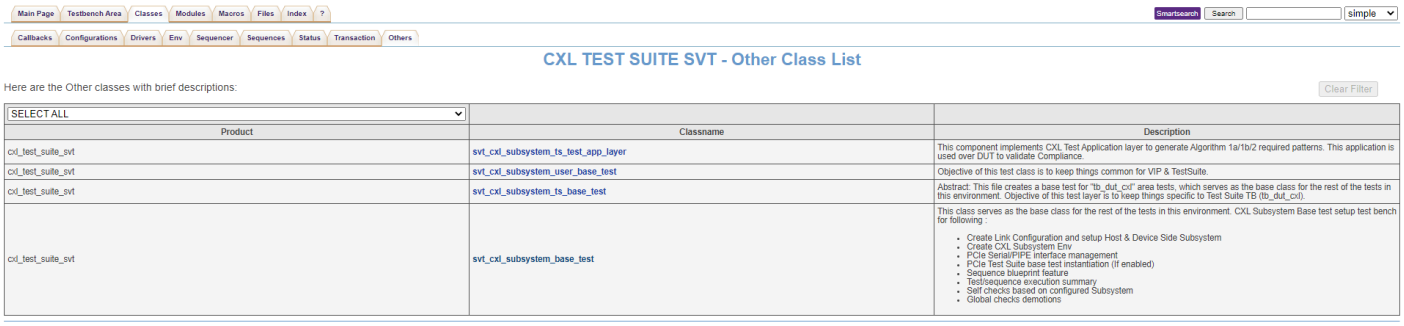
  For example, to run the test in Back-to-Back mode using PCIE serial interface, you must do the following:

```
gmake cxl_io_random_mem_wr_rd USE_SIMULATOR=vcspcvlog
SVT_CXL_SUBSYSTEM_TOPOLOGY_FILE=topology_snps_vip_cxl_b2b.svi
SVT_CXL_SUBSYSTEM_COMPILE_FILE=compile_snps_vip_pcie_serial.f
```

4 Test Suite Test Bench Architecture

This chapter acts as reference and it provides a high-level architecture overview of CXL Test Suite test bench environment and the features of different components.

Refer to CXL test suite class reference document tab 'Classes' for details of the classes and attributes. In this snapshot, the 'Others' tab provides the details of test classes and test application layer.



Main Page | Testbench Area | Classes | Modules | Macros | Files | Index | ?

Callbacks | Configurations | Drivers | Env | Sequencer | Sequences | Status | Transaction | Others

SmartSearch | Search | simple

CXL TEST SUITE SVT - Other Class List

Here are the Other classes with brief descriptions:

Clear Filter

SELECT ALL		
Product	Classname	Description
cxl_test_suite_svt	svt_cxl_subsystem_ts_test_app_layer	This component implements CXL Test Application layer to generate Algorithm 1a/1b/2 required patterns. This application is used over DUT to validate Compliance.
cxl_test_suite_svt	svt_cxl_subsystem_user_base_test	Objective of this test class is to keep things common for VIP & TestSuite.
cxl_test_suite_svt	svt_cxl_subsystem_ts_base_test	Abstract: This file creates a base test for "tb_dut_cxl" area tests, which serves as the base class for the rest of the tests in this environment. Objective of this test layer is to keep things specific to Test Suite TB (tb_dut_cxl).
cxl_test_suite_svt	svt_cxl_subsystem_base_test	This class serves as the base class for the rest of the tests in this environment. CXL Subsystem Base test setup test bench for following : • Create Link Configuration and setup Host & Device Side Subsystem • Create CXL Subsystem Env • PCIe Serial/PIPE interface management • PCIe Test Suite base test instantiation (If enabled) • Sequence blueprint feature • Test/sequence execution summary • Self checks based on configured Subsystem • Global checks demotions

4.1 Block Diagram of VIP-VIP

This block diagram presents the VIP back to back connection in CXL test suite environment.

In CXL test suite, the signaling connection is either PIPE or Serial. The testbench environment includes two instances of CXL Subsystem VIP (acting as Host and Device), configuration, status classes, and sequencer components.

Support for all topologies (DUT) in single testbench with different interface types. The test setup and execution is controlled by configuration and topology files. Currently, the PIPE and Serial topology are supported.

Note Refer the CXL Subsystem VIP user guide and VIP-DUT Integration documents for more details on topology and compile files.

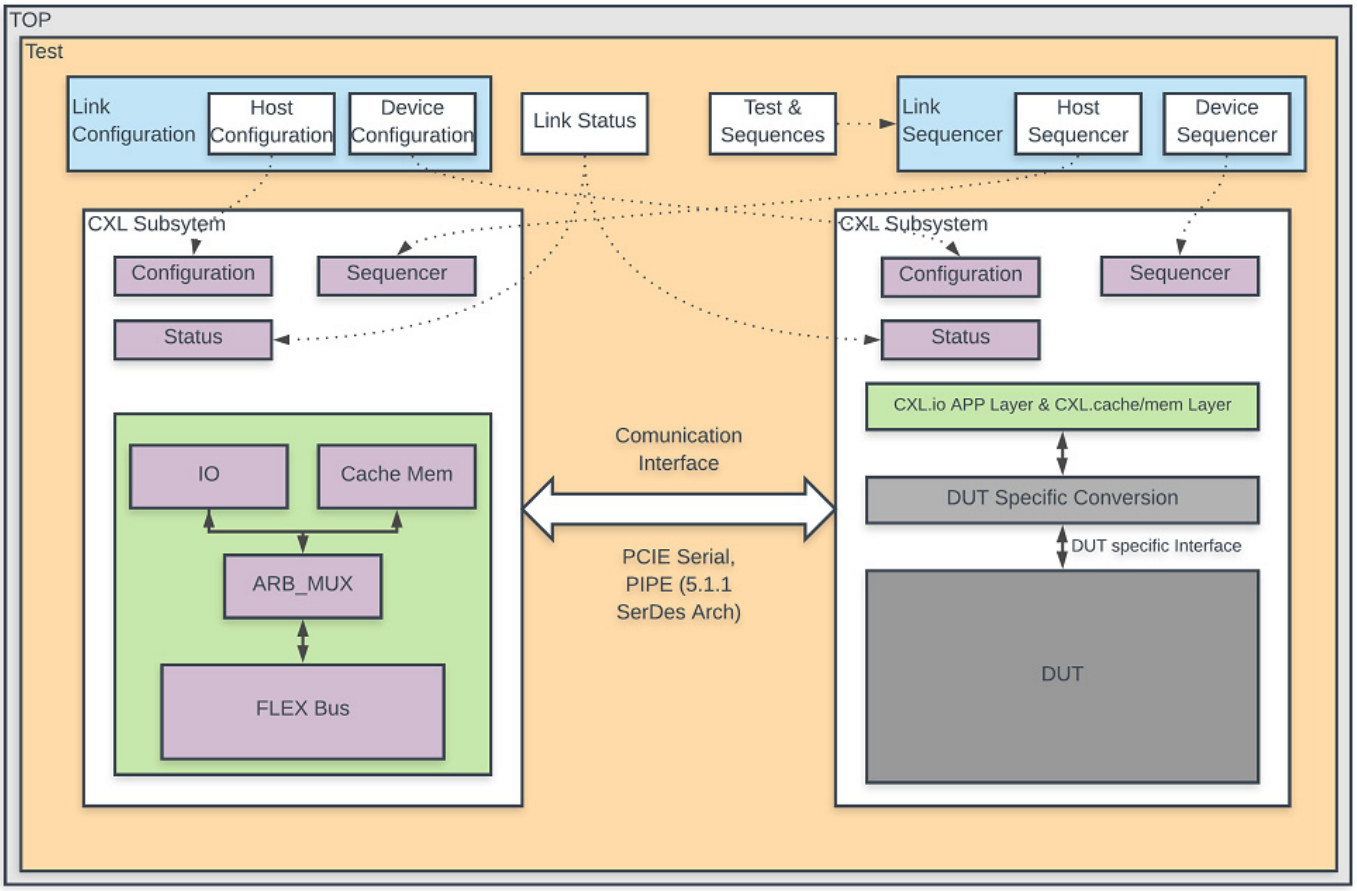
CXL Subsystem VIP User Guide:

```
$DESIGNWARE_HOME/vip/svt/cxl_subsystem_svt/latest/doc/cxl_subsystem_svt_uvm_user_guide.pdf
```

CXL VIP-DUT Integration Guide:

```
$DESIGNWARE_HOME/vip/svt/cxl_subsystem_svt/latest/doc/cxl_subsystem_svt_dut_integration_guide.pdf
```

Figure 4-1 Block Diagram for VIP-VIP



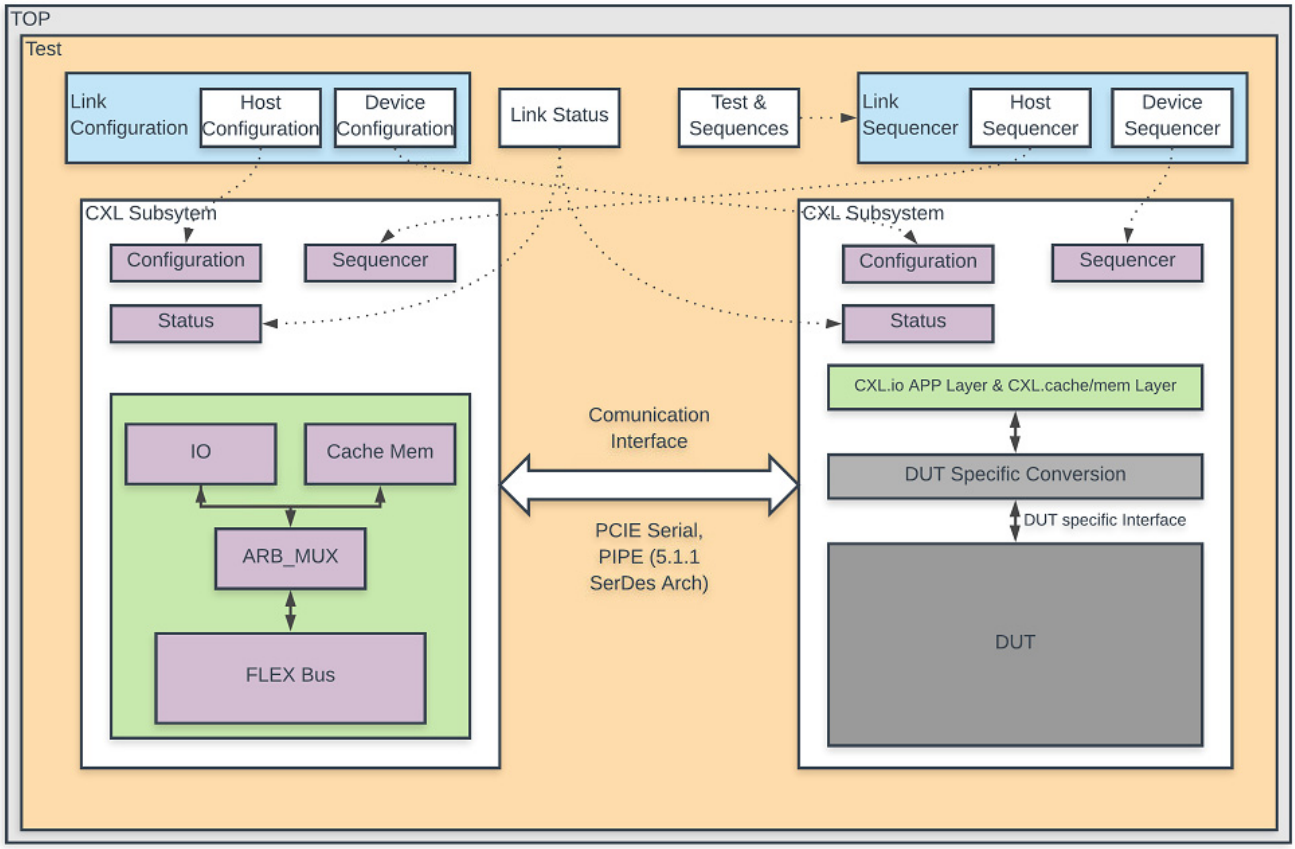
4.2 Block Diagram for VIP_RTL

CXL test suite environment is designed such that one instance of VIP (CXL Device) is replaced by CXL RTL component as depicted in the following figure.

The architecture also supports the ability to connect VIP on top of the DUT acting as application layer but requires the BFM model to convert the VIP TLM objects to DUT Signaling and its reverse.

Note Contact Synopsys Support if you need to use the VIP acting as application layer on top of the DUT.

Figure 4-2 Block diagram for VIP-RTL



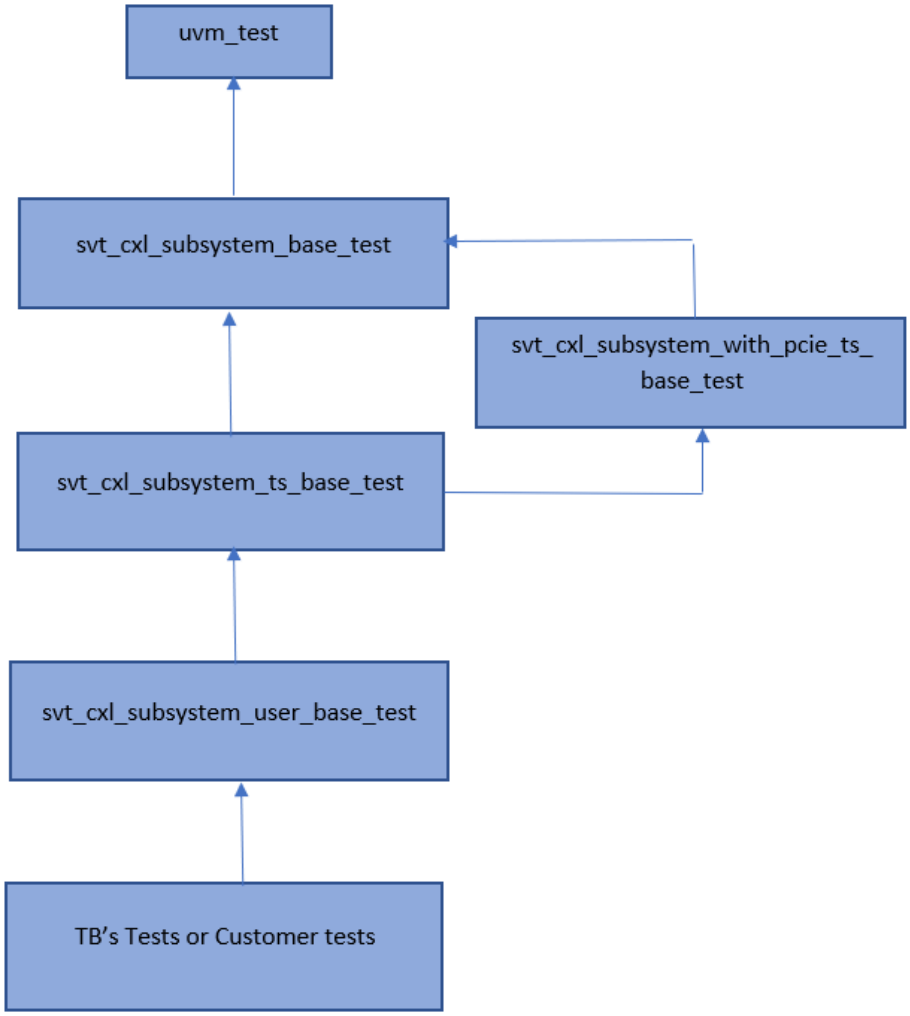
4.3 Test and Environment Class Framework

This section describes how the environment classes are partitioned into base classes and extended classes.

4.3.1 Base Test Classes

This flow chart gives the overall hierarchy of base test classes.

Figure 4-3 Test Layer Structure



Test Flow and Usage

svt_cxl_subsytem_base_test:

This is the base test. It instantiates all the required objects like agents, env, configuration, and status required for using the CXL Subsystem VIP. This also contains the basic configurations required to configure the Subsystem VIP.

Here CXL subsystem environment class is created which test suite is using.

```
`SVT_CXL_SUBSYSTEM_TEST_ENV env;
```

The above macro represents the `svt_cxl_subsystem_user_basic_env` class, which is specific to test suite.

It has an attribute called `disable_global_timeout` to control the global timeout for the test cases.

This attribute has set to disable mode default. You can enable it in the extended test. Along with that global timeout value needs to be configure based on the requirement. Also, you can re-define the macro in testbench `SVT_CXL_SUBSYSTEM_TEST_TIMEOUT` to override the default value.

svt_cxl_subsytem_ts_base_test:

This is a base test for `tb_dut_cxl` area tests, which serves as the base class for the rest of the tests in this environment.

The objective of this test layer is to keep things specific to Test Suite TB (tb_dut_cxl).

In normal CXL mode, this test extends from `svt_cxl_subsystem_base_test`.

With PCIe mode, this test extends from `svt_cxl_subsystem_with_pcie_ts_base_test` to get PCIe features.

This code represents picture about `svt_cxl_subsystem_ts_base_test`.

```
/*
`ifndef SVT_CXL_SUBSYSTEM_EXCLUDE_PCIE_TS_DATA
`SVT_CXL_SUBSYSTEM_TEST_CONSTRUCTOR(svt_cxl_subsystem_ts_base_test,svt_cxl_subsystem_wit
h_pcie_ts_base_test)
`else
`SVT_CXL_SUBSYSTEM_TEST_CONSTRUCTOR(svt_cxl_subsystem_ts_base_test,svt_cxl_subsystem_bas
e_test)
`endif
*/
```

`SVT_CXL_SUBSYSTEM_EXCLUDE_PCIE_TS_DATA` macro definition helps you to run with PCIe mode or CXL mode, which you can pass as a command line argument.

It has the instantiation of application layer configuration class and application layer status class.

svt_cxl_subsystem_ts_base_test

----------> host_app_cfg {type svt_cxl_subsystem_ts_test_app_layer_configuration}

|

----------> device_app_cfg {type svt_cxl_subsystem_ts_test_app_layer_configuration}

|

----------> host_app_status {type svt_cxl_subsystem_ts_test_app_layer_status}

|

----------> device_app_status {type svt_cxl_subsystem_ts_test_app_layer_status}

Setting of application layer configuration class and status class handle helps you to get it in environment class.

Passing test application layer configuration and status objects to test suite environment.

```
/*
svt_config_object_db#(svt_cxl_subsystem_ts_test_app_layer_configuration)::set(this,
"env", "host_app_cfg", host_app_cfg);
*/
```

Example Usage:

```
host_app_cfg.is_test_app_layer_over_host=1;
```

This setting decides whether APP Layer would present over Host or Device.

svt_cxl_subsytem_user_base_test:

The `svt_cxl_subsystem_user_base_test` can be used by you to create own test to execute specific scenario.

This test class objective is to keep things common for VIP and Test Suite and offers you to add your own features by extending from this class.

```
/*
The svt_cxl_subsystem_user_base_test is extended from svt_cxl_subsystem_ts_base_test
`SVT_CXL_SUBSYSTEM_TEST_CONSTRUCTOR(svt_cxl_subsystem_user_base_test,svt_cxl_subsystem_ts_base_test)
*/
```

This useful base test class methods can be defined inside derived tests.

`create_cfg()`: Overall all VIP configurations are performed in this method of base test, you can override user test configuration also through this method.

- `create_cfg()` method is also called from main_phase of CXL Subsystem base test to resolve CXL.io configure override from PCIe TS Base test (if enabled). Also CXL.io component is reconfigured for final configuration in main_phase.
- Configuration other than CXL.io component can be wrapped with `create_cfg_called` flag to avoid reprocessing. This can also lead to side effect if randomization of configuration is performed with different values generated as compared to build_phase.

`create_sequence_blueprints()`: Method to change/randomize any test sequence attributes outside the test. If test is derived from any other test and wants to shape the sequence attributes, then the sequence blueprint can be created.

`cxl_test_main_phase ()`: CXL TS specific main_phase. Independent of methodology.
Test must ensure that the link is up (or at operational state) before it exits the main phase.

These PCIe related base tests are added to testsuite:

- **svt_cxl_subsytem_with_pcie_ts_base_test**:

  This test to include PCIE TS test for verification test bench and CXL TS test bench. When we link the PCIe test case from `tb_dut_pcie_gen5_file`, this base test would execute with specific PCIE feature.

- **pcie_unified_vip_example_base_test**:

  pcie unified example base test.

- **svt_cxl_pcie_base_test**:

  Extension of PCIe TS base Test so for instance override when we are executing PCIe TestSuite tests through CXL with CXL Framing.

Creating your own test

- You need to extend your test case from `svt_cxl_subsytem_user_base_test`.
- By using below Macro user can define their test.

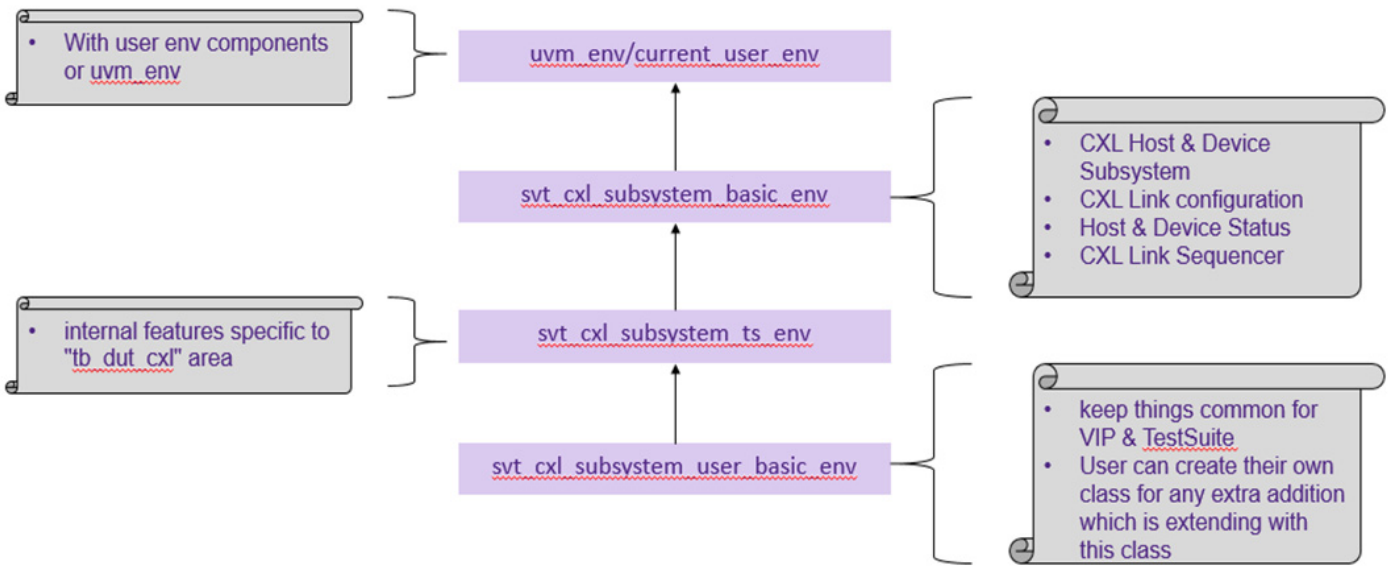
```
SVT_CXL_SUBSYSTEM_TEST_CONSTRUCTOR(user_test, base_test_class)

    Class user_test extends base_test_class
        `uvm_component_utils (user_test)
         function new( string name = "user_test",uvm_component parent = null);
     super.new(name,parent)
endfunction
```

4.3.2 Environment Class Framework

Overview

This diagram gives an overall description of environment inside the CXL testsuite.

Figure 4-4 Environment Flow



Usage of Environment Class

svt_cxl_subsytem_user_basic_env:

This environment class is extended from the `svt_cxl_subsystem_ts_env`, which offers you to extend your own environment to update or change your environment according to the requirement.

```
*/
class svt_cxl_subsystem_user_basic_env extends svt_cxl_subsystem_ts_env;
  `svt_xvm_component_utils(svt_cxl_subsystem_user_basic_env)
  // ----------------------------------------------------------------------------
 */
```

svt_cxl_subsystem_ts_env:

- It implements internal features specific to `tb_dut_cxl` area.
- The `svt_cxl_subsystem_ts_env` is specific to test suite, which is extended from the environment `svt_cxl_subsystem_basic_env`.

svt_cxl_subsystem_ts_env

-------> host_app {type svt_cxl_subsystem_ts_test_app_layer}

--------> device_app {type svt_cxl_subsystem_ts_test_app_layer}

--------> host_app_service_seqr {type svt_cxl_subsystem_ts_test_app_layer_service_sequence}

---------> device_app_service_seqr {type svt_cxl_subsystem_ts_test_app_layer_service_sequencer }

-------> host_app_cfg {type svt_cxl_subsystem_ts_test_app_layer_configuration }

--------> device_app_cfg {type svt_cxl_subsystem_ts_test_app_layer_configuration}

--------> host_app_status {type svt_cxl_subsystem_ts_test_app_layer_status}

-------->device_app_status {type svt_cxl_subsystem_ts_test_app_layer_status}

- This environment getting the application layer configuration and status handle from the test.
- When you use your own derived test and environment, this class creates and sets the configuration or status for the host and device.
- The configuration setting of host/device_app_cfg, host/device_app_status will pass to the Test Application Layer.

```
*/
                if
(svt_config_object_db#(svt_cxl_subsystem_ts_test_app_layer_configuration)::get(this,"",
                 "host_app_cfg", host_app_cfg)) begin
                    `svt_note(method_name, "Using test case provided host_app_cfg");
                end
               else begin
                `svt_note(method_name, "External host_app_cfg not provided, creating
default host_app_cfg.");
            host_app_cfg =
svt_cxl_subsystem_ts_test_app_layer_configuration::type_id::create("host_app_cfg");
         end
       ----------------------------------------------------------------
    svt_config_object_db#(svt_cxl_subsystem_ts_test_app_layer_configuration)::set(this,
      "host_app_service_seqr", "cfg", host_app_cfg);
```

svt_cxl_subsystem_basic_env:

- This is a top level environment class to encapsulate Host and Device both side components.

svt_cxl_subsystem_basic_env

------------------> host_env {type svt_cxl_subsystem_env}

|

------------------> device_env {type svt_cxl_subsystem_env}

|

------------------> cfg {type svt_cxl_subsystem_link_configuration}

|

------------------> link_status {type svt_cxl_subsystem_link_status}

|

------------------> cxl_link_seqr {type svt_cxl_subsystem_link_virtual_sequencer}

4.3.3 Configuration Data Objects

This section describes about configuration class details. All the configuration data objects used in test suite is same for VIP.

For a complete list and its details, refer the CXL Subsystem Class reference and user guide:

```
$DESIGNWARE_HOME/vip/svt/cxl_subsystem_svt/latest/doc/cxl_subsystem_svt_uvm_user_guide.pdf
```

svt_cxl_subsystem_ts_test_app_layer_configuration - This class is specific to used inside the test suite and it implements the CXL Test DVSEC Capabilities and Control registers for CXL Test Application Layer component.

This class is set from test or env level and the update of configurations pass to the test application layer. Refer the CXL Test suite class reference fore more details about configuration variable.

CXL Test Suite class Reference:

```
$DESIGNWARE_HOME/vip/svt/cxl_test_suite_svt/latest/doc/cxl_test_suite_svt_uvm_reference/html/index
```

This example show how to set the different value of the configuration filed from test/env:

```
svt_cxl_subsystem_ts_test_app_layer_configuration device_app_cfg, host_app_cfg;
device_app_cfg.protocol = 3'b001;     //CXL.io only
```

You need to configure the type of the device (type1/2/3) in build phase based on the DUT settings (in user base test). For example,

```
cache_mem_sys_cfg.host_cfg[link_num].device_type =
svt_cxl_cache_mem_configuration::TYPE3_DEVICE;

cache_mem_sys_cfg.device_cfg[link_num].device_type =
svt_cxl_cache_mem_configuration::TYPE3_DEVICE;
```

4.3.4 Status

This section describes about all status class. All the status data objects used in test suite is same for VIP. The explanation of status classes is present in the CXL Subsystem User Guide.

```
$DESIGNWARE_HOME/vip/svt/cxl_subsystem_svt/latest/doc/cxl_subsystem_svt_uvm_user_guide.pdf
```

`svt_cxl_subsystem_ts_test_app_layer_status` - This class captures the status of CXL Subsystem Control and Status Registers (CSR) and is specific to the test suite.

```
*/
svt_cxl_subsystem_ts_test_app_layer_status host_app_status;
We can use the attribute of application layer as below format :
host_app_status.test_completed = 1'b0;
```

5 Test Suite Tests

In CXL test suite, the tests are in grouped into these two categories:

- Compliance tests
  - CXL specification section "14.0 CXL Compliance Testing" is the test specification for this group of tests.
  - Naming convention: `cxl_compliance` in test name is used in the test name for this group of tests. For example, ts.cxl_compliance_pl_eds_token.sv.
  - Rest of the name indicate the protocol area and specific test name is derived from the sub-section of test specification.
- Synopsys defined tests
  - CXL specification is the basis for this group of tests and specification is defined by Synopsys. Typically, they cover the test attributes not part of the 'CXL Compliance Testing' of the specification. These tests can help with sanity testing of installation, integration testing and extended areas of protocol.
  - Naming convention: It follows layer and test area in test name. For example, ts.cxl_tl_type1_cache_wr_rd.sv indicates a TL layer test related to Type1 device Cache Read/Write transfers.

Additionally, CXL test suite product is designed to support COMBO designs supporting PCIE and CXL modes of operation. The PCIE test suite tests can be executed CXL `tb_dut_cxl` test bench environment. PCIE tests can be run with legacy framing (when APN is unsuccessful) or in Flex bus mode (when APN is successful).

Refer to section 'Test Execution' on how to run the tests with `gmake command`.

5.1 Test Lists

The details list of tests delivered as part of the product are available in the test list files installed as part of environment directory of examples.

CXL Test list files for Synopsys defined tests:

- cxl_cache_mem_testlist.svi
- cxl_io_testlist.svi

CXL Test list file for Compliance tests

- cxl_compliance_testlist.svi

Note In the test list files ignore the literals "COMBO_*" as they are used for product installation and testing.

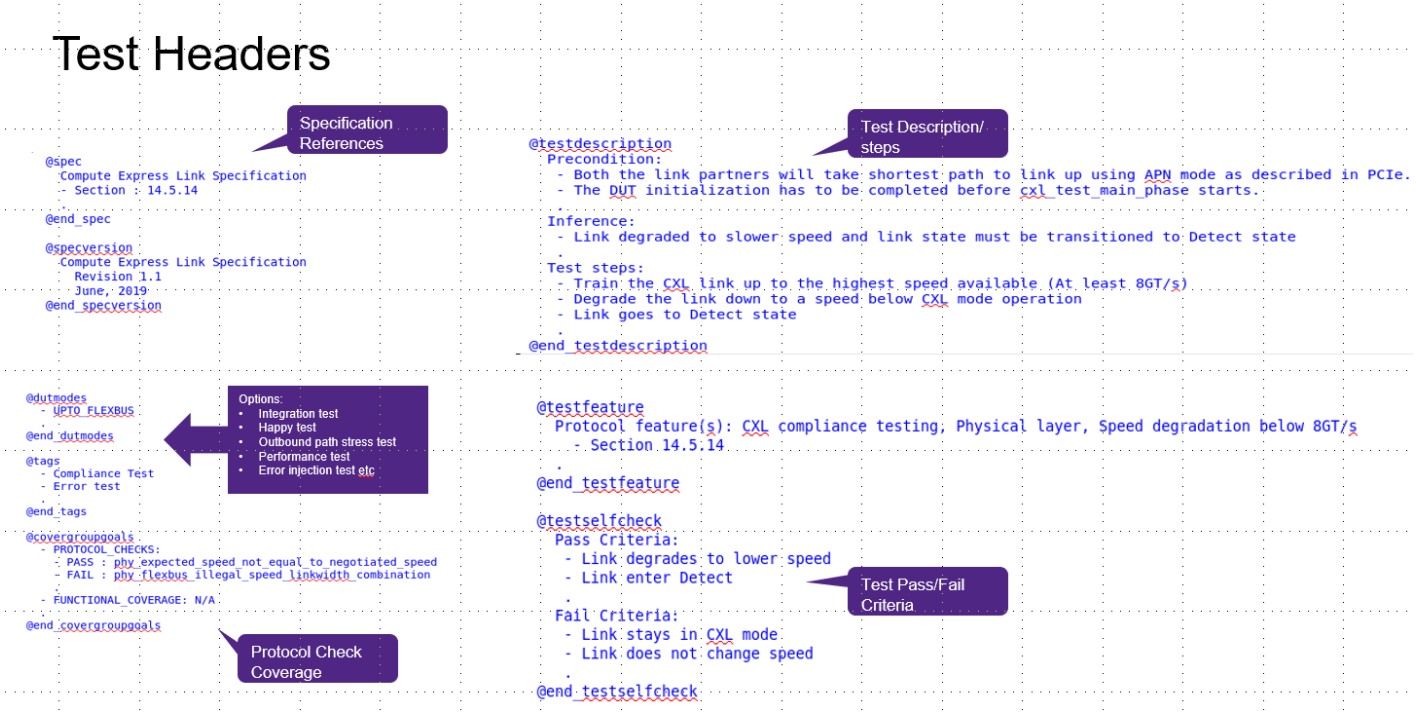
PCIE Test list:

Only subset of PCIE test suite tests that are applicable in CXL mode and is also list is based on the DUT capabilities.

Contact Synopsys if you want to select and execute PCIE tests in CXL test suite for your COMBO design verification.

5.2 Test Documentation

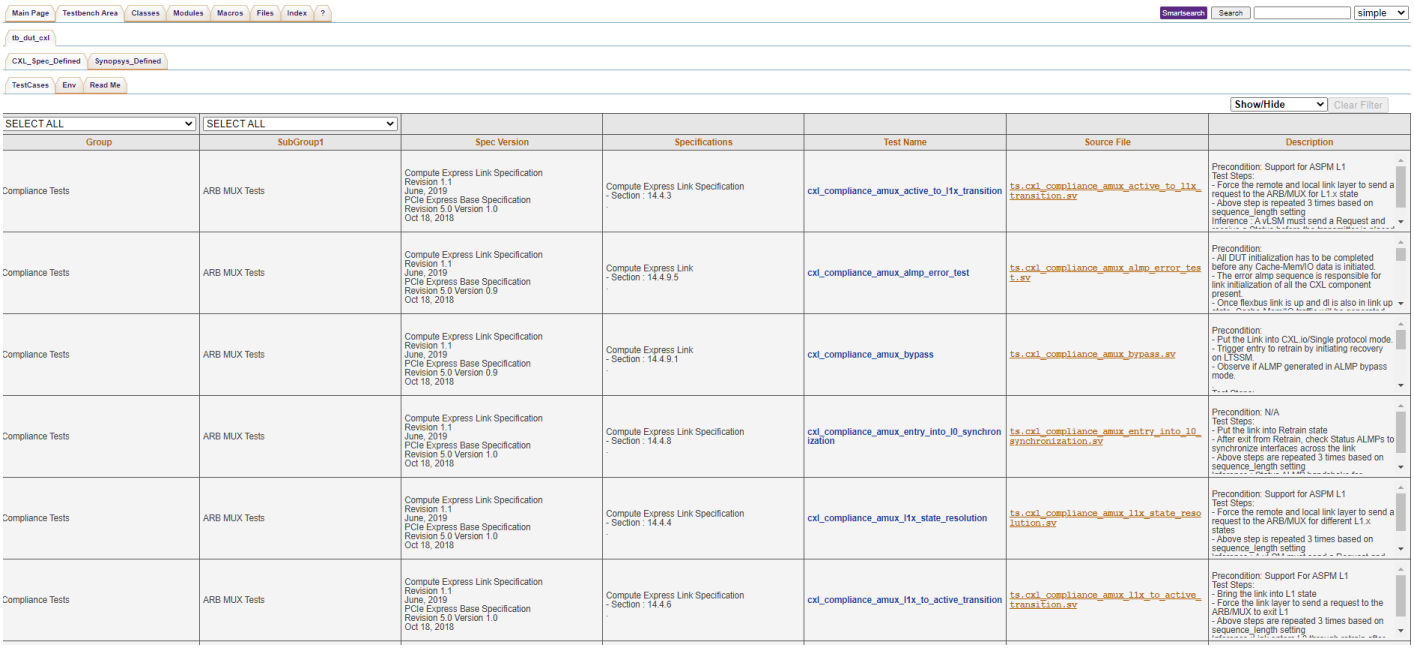
All the tests implement a standard header format as depicted in the snapshot below providing comprehensive information with specification and the test description.



Refer the CXL Subsystem Test Suite Class reference for test documentation of all tests available in the product as displayed in the figure below. The class reference provides Filter feature Show/Hide (circled below) and can be used to add / remove the columns to be displayed for the test documentation.

Test Suite Class Reference path :

```
$DESIGNWARE_HOME/vip/svt/cxl_test_suit_svt/latest/doc/pcie_test_suite_svt_uvm_reference
/html/index
```



Group	SubGroup1	Spec Version	Specifications	Test Name	Source File	Description
Compliance Tests	ARB MUX Tests	Compute Express Link Specification Revision 1.1 June, 2019 PCIe Express Base Specification Revision 5.0 Version 1.0 Oct 18, 2018	Compute Express Link Specification - Section : 14.4.3	cxl_compliance_amux_active_to_l1x_transition	ts.cxl_compliance_amux_active_to_l1x_transition.sv	Precondition: Support for ASPM L1 Test Steps: - Force the remote and local link layer to send a request to the ARB/MUX for L1.x state - Above step is repeated 3 times based on sequence_length setting Inference: A vLSM must send a Request and
Compliance Tests	ARB MUX Tests	Compute Express Link Specification Revision 1.1 June, 2019 PCIe Express Base Specification Revision 5.0 Version 0.9 Oct 18, 2018	Compute Express Link - Section : 14.4.9.5	cxl_compliance_amux_almp_error_test	ts.cxl_compliance_amux_almp_error_test.sv	Precondition: - All DUT initialization has to be completed before any Cache-Mem/IO data is initiated. - The error almp sequence is responsible for link initialization of all the CXL component present. - Once flexbus link is up and dl is also in link up
Compliance Tests	ARB MUX Tests	Compute Express Link Specification Revision 1.1 June, 2019 PCIe Express Base Specification Revision 5.0 Version 0.9 Oct 18, 2018	Compute Express Link - Section : 14.4.9.1	cxl_compliance_amux_bypass	ts.cxl_compliance_amux_bypass.sv	Precondition: - Put the Link into CXL.io/Single protocol mode. - Trigger entry to retrain by initiating recovery on LTSSM. - Observe if ALMP generated in ALMP bypass mode.
Compliance Tests	ARB MUX Tests	Compute Express Link Specification Revision 1.1 June, 2019 PCIe Express Base Specification Revision 5.0 Version 1.0 Oct 18, 2018	Compute Express Link Specification - Section : 14.4.8	cxl_compliance_amux_entry_into_l0_synchronization	ts.cxl_compliance_amux_entry_into_l0_synchronization.sv	Precondition: N/A Test Steps: - Put the link into Retrain state - After exit from Retrain, check Status ALMPs to synchronize interfaces across the link - Above steps are repeated 3 times based on sequence_length setting
Compliance Tests	ARB MUX Tests	Compute Express Link Specification Revision 1.1 June, 2019 PCIe Express Base Specification Revision 5.0 Version 1.0 Oct 18, 2018	Compute Express Link Specification - Section : 14.4.4	cxl_compliance_amux_l1x_state_resolution	ts.cxl_compliance_amux_l1x_state_resolution.sv	Precondition: Support for ASPM L1 Test Steps: - Force the remote and local link layer to send a request to the ARB/MUX for different L1.x states - Above step is repeated 3 times based on sequence_length setting
Compliance Tests	ARB MUX Tests	Compute Express Link Specification Revision 1.1 June, 2019 PCIe Express Base Specification Revision 5.0 Version 1.0 Oct 18, 2018	Compute Express Link Specification - Section : 14.4.6	cxl_compliance_amux_l1x_to_active_transition	ts.cxl_compliance_amux_l1x_to_active_transition.sv	Precondition: Support For ASPM L1 Test Steps: - Bring the link into L1 state - Force the link layer to send a request to the ARB/MUX to exit L1 - Above steps are repeated 3 times based on sequence_length setting

5.2.1 Test Group

CXL 1.1 tests that are untouched for CXL 2.0 compliance tests are not updated to capture the specification reference. For example, specification in the test as illustrated in the figure below is only having reference to 1.1 spec.

Figure 5-1 Tests having reference to 1.1 spec

Group	SubGroup1	Spec Version	Specifications	Test Name	Source File	Sequences	Description	Test Configurations	Test Features	Self Check
SPEC-1.1&SPEC-2.0	Physical Layer Tests	Compute Express Link Specification Revision 1.1 June, 2019 PCIe Express Base Specification Revision 5.0 Version 0.9 Oct 18, 2018	Compute Express Link Specification - Section : 14.5.3(CXL 1.1)	cxl_compliance_pl_eds_token	ts.cxl_compliance_pl_eds_token.sv	-	Precondition: - The DUT initialization has to be completed before any Cache-Mem/IO data is initiated. - The EDS token sequence is responsible	- TS_EXCLUSION_REASON: TS_EX_LINK_WIDTH: N/A TS_EX_SPEED: N/A TS_EX_OPTIONAL_FEA	Protocol feature(s): CXL Compliance Testing, Physical layer, EDS token - Section 14.5.3	Pass criteria: - EDS token is the last flit in the data block - EDS token does not cross the data block boundary - Next block after EDS

Some of the tests are captured only with CXL 1.1 Spec reference whereas some tests are captured with CXL 2.0 reference also. See the following figure with sections of both mentioned.

Figure 5-2 Tests having reference to both 1.1 and 2.0 spec

SPEC-1.1&SPEC-2.0	Configuration Register Tests	Compute Express Link Specification Revision 1.1 June, 2019 Revision 2.0 October, 2020 PCIe Express Base Specification Revision 5.0 Version 1.0	Compute Express Link Specification - Section : 14.6.1(CXL 1.1) - Section : 14.8.1(CXL 2.0)	cxl_compliance_crt_device_presence	ts.cxl_compliance_crt_device_presence.sv	-	Precondition: - All DUT initialization will be completed before any CfgRd/CfgWr is issued. - Once flexbus link is up and dl is also in dl up state, CXL DVSEC is	- TS_EXCLUSION_REASON: TS_EX_LINK_WIDTH: N/A TS_EX_SPEED: N/A TS_EX_OPTIONAL_FEA	Protocol feature(s): CXL Compliance Testing, Configuration Register Tests, Device Presence	Pass criteria: - CXL 1.1 : One RCiEP device found. - CXL 2.0 : One PCIe device with CXL PCI Express DVSEC Capability found.

Followed Approach:

If the test is updated in CXL 2.0 specification, then capture both the sections. If not updated, then continue with CXL 1.1 specification section.

5.3 Test Execution

Refer the Installation and Setup chapter section for running the examples on steps to execute the tests using the `gmake` command for running the CXL tests or PCIE tests in CXL environment.

The details are also available as part of the 'README' file available as part of `tb_dut_cxl` directory in the installation.

5.3.1 Test Execution with Debug

Configuration:

By default, the test suite is setup to dump the configuration classes at connect/run phase for debug. The details are logged into the following files:

- configuration_attributes_connect_phase_info.txt
- configuration_attributes_run_phase_info.txt

Sequence execution flow for compliance tests:

- Uniform debug prints at CXL Subsystem Test Suite sequence level.
  - Before and after prints for each `wait` statement.
  - Specific ID based debug prints "body : <ID>". Example: 'body : `SNPS_CXL_TS_COMPLIANCE`'

Example:

```
UVM_INFO ./env/seq_and_cb/svt_cxl_subsystem_ts_io_sequence_collection/
svt_cxl_subsystem_ts_io_wr_rd_sequence.sv(158) @ 27307192.10 ps: reporter
[body : SNPS_CXL_TS_COMPLIANCE] dut executing CXL IO Mem Read/Write
Sequence[38].....
```

Individual Log Files

- For usage of available CXL Subsystem level layer specific symbol/xact logs for debug, refer the CXL Subsystem UVM user guide for detailed information on log formats and usage.
- Different layer transaction/symbol logging can be enabled using `SVT_CXL_SUBSYSTEM_ENABLE_LOGGING`=1 command line option and the following logs gets generated during the simulation run:
  - CXL.IO TL/DL (.xact_log)
  - CXL.Cache.Mem TL (.xact_log)
  - ARB-MUX (.xact_log)
  - Flex Bus (.sym_log)

Message Dump Control

This is the UVM based approach to control the messages from CXL Product and this disables all UVM_INFO prints from Synopsys CXL Product:

```
+uvm_set_verbosity=*,report_message,UVM_NONE,time,0
+uvm_set_verbosity=uvm_test_top.pcie_tests_0.env.global_shadow*,_ALL_,UVM_NONE,time,0
+uvm_set_verbosity=uvm_test_top.env.host_env.*,_ALL_,UVM_NONE,time,0
+uvm_set_verbosity=uvm_test_top.env.device_env.*,_ALL_,UVM_NONE,time,0
+uvm_set_verbosity=uvm_test_top.pcie_tests_0.env.rc_env*,_ALL_,UVM_NONE,time,0
+uvm_set_verbosity=uvm_test_top.pcie_tests_0.env.ep_env*,_ALL_,UVM_NONE,time,0
```

This feature has been enabled as default in `tb_dut_cxl`. You can use the Makefile switch `ENABLE_CXL_SUBSYSTEM_MESSAGES` to enable the internal message dumping.

5.4 Test Controls

5.4.1 Simulating Extended Protocol Scenarios

Tests implementation is based on the CXL 'Compliance Testing' chapter and by default runs the test steps indicated in the test scenario. The 'Pass / Fail' criteria is based on the requirement outlined in the test specification.

The product is architected with control knobs to quickly scale for simulating extended scenarios error conditions and for stress testing for CXL features.

The following illustration provides the details on how you can leverage control knobs to create additional simulating protocol scenarios for DUT flex bus layer verification.

Speed/link-width Control for degraded mode of operation

- Test features and extension - an Illustration
  - Flex bus layer tests: (Specification defined - 5 tests) - Normal mode: x16, 32GT/s: 5 simulation scenarios
  - 14.5.1 : Protocol ID
  - 14.5.2 : Null Flit
  - 14.5.3 : EDS Token
  - 14.5.8 : Link Speed Advertisement
  - 14.5.9 : Idle Transition to L0
- Controls
  - Degraded speed mode (16GT/s, 8GT/s): 2
    - Degraded speed mode using svt_pcie_pl_configuration::set_link_speed_values API.
  - Degraded link width: (x1, x2, x4, x8): 4
    - Degraded link width mode using svt_pcie_pl_configuration::set_link_width_values API.
- Total simulation scenarios: 40 (5 tests x2 speed modes x 4 link widths)

5.4.2 Traffic Generation Control

The framework also provides control knobs for traffic generation whether it is from Link partner VIP or VIP layer on top of DUT or both.

These are the user controls to enable application layer additional traffic from VIP/DUT:

- svt_cxl_subsystem_link_configuration::initiate_traffic_direction[1:0]:
- 00 - No Application layer traffic initiated.

- 01 - Traffic from VIP Application layer only.
- 10 - Traffic from DUT Application layer only.
- 11 - Traffic from both VIP and DUT Application layer

5.4.3 Traffic Length Control

The amount of traffic generated by the test is controlled through `svt_cxl_subsystem_ts_base_sequence` class property `sequence_length`. This is a global control and using the UVM config_db to configure the amount of traffic generation for all tests.

5.4.4 Blueprint Based Sequence to Control Traffic

Test suite test control framework also provides ability to use the user-supplied sequence to control properties defined in the individual test sequences. You need to override `create_sequence_blueprints()` in base test to control properties defined in the sequences. Both rand/ non-rand attributes can be controlled. Refer to TS sequence documentation in TS class reference for details of sequence attributes.

In case same sequence is used in multiple tests, then blueprint override sequence is effective for all tests.

This is an example to control maximum/minimum sequence length attribute.

```
function void create_sequence_blueprints();
svt_cxl_subsystem_ts_io_wr_rd_sequence io_mem_wr_rd_seq_blueprint;
super.create_sequence_blueprints();
io_mem_wr_rd_seq_blueprint =
svt_cxl_subsystem_ts_io_wr_rd_sequence::type_id::create("io_mem_wr_rd_seq_blueprint");
`SVT_CXL_SUBSYSTEM_TS_STORE_SEQ_BLUEPRINT(io_mem_wr_rd_seq_blueprint);
io_mem_wr_rd_seq_blueprint.min_seq_length = 1;
io_mem_wr_rd_seq_blueprint.max_seq_length = 50;
endfunction // create_sequence_blueprints
```

5.4.5 Configuration Control Based on Text File

CXL test suite leverages simple <name > = < value > pair for setting the configuration attributes defined in a text file. The text file can be provided as input during the simulation run of the test.

Use model:

- The configurations are applied during run time. So, no need to recompile to apply the .txt files.

  Multiple .txt files can be provided for a single run with ',' separator

  ```
  gmake <test_name> SVT_CXL_SUBSYSTEM_TEXT_FILES=<test1>.txt, <text2>.txt
  ```

```
// Configuration class name=value pair settings to be loaded

// The target link speed can only be set by the Host port
// So, this text file will only work for when Device is IIP and Host is VIP

//`define SVT_PCIE_SPEED_2_5G     32'h0000_0002 = 2
//`define SVT_PCIE_SPEED_5_0G     32'h0000_0004 = 4
//`define SVT_PCIE_SPEED_8_0G     32'h0000_0008 = 8
//`define SVT_PCIE_SPEED_16_0G    32'h0000_0010 = 16
//`define SVT_PCIE_SPEED_32_0G    32'h0000_0020 = 32
//`define SVT_PCIE_SPEED_64_0G    32'h0000_0001 = 1
// Target Link Speed variable will be only set by Downstream Port
host_cfg.cxl_io_cfg[0].pcie_cfg.pl_cfg.target_speed=32'h0000_0010

//`define SVT_PCIE_SPEED_2_5G     32'h0000_0002 = 2
//`define SVT_PCIE_SPEED_5_0G     32'h0000_0004 = 4
//`define SVT_PCIE_SPEED_8_0G     32'h0000_0008 = 8
//`define SVT_PCIE_SPEED_16_0G    32'h0000_0010 = 16
//`define SVT_PCIE_SPEED_32_0G    32'h0000_0020 = 32
//`define SVT_PCIE_SPEED_64_0G    32'h0000_0001 = 1
host_cfg.cxl_io_cfg[0].pcie_cfg.pl_cfg.expected_speed=32'h0000_0010
device_cfg.cxl_io_cfg[0].pcie_cfg.pl_cfg.expected_speed=32'h0000_0010
```

5.4.6 Plusargs

PCIe Test Suite based PLUSARGS is available to reuse in CXL Subsystem Test Suite for additional scenario generation and control.

If you want to use Plusargs or TXT file-based configuration control features, contact Synopsys support.

5.4.7 Enumeration Control

CXL Test suite uses the Enumeration feature and API available in CXL Subsystem VIP in the tests. Refer the 'Enumeration' section in the CXL Subsystem VIP user guide for key features and use models.

CXL Subsystem VIP User Guide:

`$DESIGNWARE_HOME/vip/svt/cxl_subsystem_svt/latest/doc/cxl_subsystem_svt_uvm_user_guide.pdf`

5.4.8 Back-Annotation of Test Pass/Fail Results

The Synopsys Verification Planner tool provides a mechanism for back-annotating the test pass/fail results to a test plan from VDB. Perform the following procedure to back-annotate the test results:

1. Modify the `sim_build_options` to define the `SVT_CXL_SUBSYSTEM_TS_ENABLE_TEST_PASS_FAIL_COVERAGE`.

   For example, add the command in `sim_build_options`.

   ```
   +define+SVT_CXL_SUBSYSTEM_TS_ENABLE_TEST_PASS_FAIL_COVERAGE
   ```

2. Modify the `sim_build_options` to include a VCS include directory for the `CoverMeter.vh` file.

   For example, add the command in `sim_build_options`.

   ```
   +incdir+${VCS_HOME}/include/
   ```

3. Modify the `sim_run_options` with the following options (only when functional coverage/check rule coverage is not enabled).

   ```
   -cm_test $scenario -cm_name $scenario -cm_stats all -cm_dir mysimv.vdb
   ```

4. Use the hvp annotate command to generate the back-annotated sub-plan file:

   ```
   hvp annotate -plan top_plan_name.xml -dir "<vdb_name>.vdb"
   ```

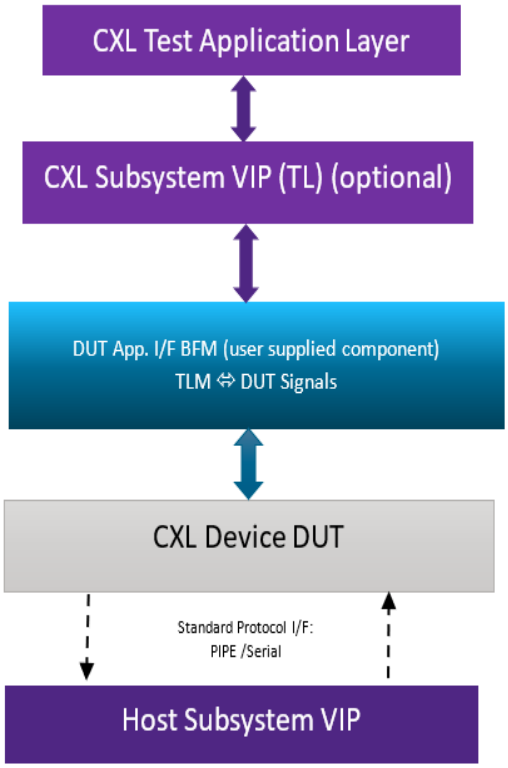
5.5 Test Application Layer and Use Model

CXL test suite provides a 'Test Application Layer' component and is part of test suite product only. This component implements the algorithms outlined as part of the CXL ‘Compliance Testing’ and provides attributes to setup the algorithms for test execution.

When the test is run in VIP-VIP B2B mode this component is enabled by default for test execution.

When the DUT is CXL device this component can be optionally used on top of the DUT to setup the TB. This feature facilitates execution of compliance tests before the Design RTL for algorithms defined for compliance testing becomes available.

- CXL Subsystem VIP TL only and application layer gets connected to the TLM ports of the VIP
  - It is assumed that the BFM to convert CXL Subsystem VIP TLM object to DUT compatible signaling is available and supplied by the user.



Note If you want to use Test application layer and VIP component on top of DUT, contact Synopsys support.

A Integrating Synopsys DWC Controller in Test Suite

These steps are applicable when the DUT is Synopsys-DWC Controller with default configuration. These are recommended in case of an evaluation of CXL Test Suite with DUT (default configuration).

A.1 Pre-Checklist:

You must ensure the following points before integration:

1. Install and extract the 'DUT DWC CXL Controller' successfully.

   Contact Synopsys IP team for support with controller installation.

2. VIP Subsystem configuration setting should be aligning with DWC Controller

   Setting:

   - Link Width setting
   - Speed setting
   - Ltssm Timeouts

For Example:

If DUT supports 8 Lanes for proper link up with x8, then the VIP must have this configuration:

```
cust_cfg.host_cfg.cxl_io_cfg[0].pcie_cfg.pl_cfg.set_link_width_values(8);
```

A.2 Synopsys DWC Controller (with default configuration) Integration Steps

1. Install and extract the test suite properly.

   Refer to Installation and Setup chapter section for detailed explanation of test suite installation and running the sanity test.

2. Go to the directory where IIP is installed using Cc and set this path as 'IIP_PATH'

   ```
   % cd <to_the_path_where_IIP_is_installed>
   % setenv IIP_PATH `pwd`
   ```

3. Need to create symbolic links for the following files:

   ```
   % cd $IIP_PATH/src
   ```

```
% ln -f -s DWC_pcie_ctl_cc_constants.svh DWC_pcie_cc_constants.v
```

Note Currently, the following symbolic links should also be created to avoid compile time issues.

```
% cd $IIP_PATH/src/include/
% ln -f -s cxpl_defs.svh cxpl_defs.vh

% cat  port_cfg.svh > port_cfg.vh
% cat cap_port_cfg.svh >> port_cfg.vh
```

4. Create a symbolic link for IIP src, sim, example directories inside the testbench area:

```
% cd $DESIGNWARE_HOME/design_dir/examples/sverilog/cxl_test_suite_svt/tb_dut_cxl
% ln -s $IIP_PATH/src src
% ln -s $IIP_PATH/sim sim
% ln -s $IIP_PATH/examples examples

% cd $DESIGNWARE_HOME/design_dir/examples/sverilog/cxl_test_suite_svt
% ln -s $IIP_PATH/sim/testbench testbench
```

5. Get the IIP file_list inside the tb_dut_cxl area:

```
% cd $DESIGNWARE_HOME/design_dir/examples/sverilog/cxl_test_suite_svt/tb_dut_cxl/
% grep -v "\-undef" $IIP_PATH/src/DWC_pcie_ctl.lst > iip.f
% grep -v "\-undef" $IIP_PATH/src/Phy/generic/compile.f >> iip.f
```

6. Set the following environment variables:

```
% setenv PUE_HOME
$DESIGNWARE_HOME/design_dir/examples/sverilog/pcie_test_suite_svt/tb_dut_pcie_gen5_file/
env/dwc_pcie_dut_bfm/pue
% setenv UVM_EXT_HOME $PUE_HOME/uvm_ext
```

7. Valid combination of Topology/compile files for DUT as Synopsys IIP with serial interface.

   Topology file is:

   ```
   SVT_CXL_SUBSYSTEM_TOPOLOGY_FILE=topology_snps_cxl_pcie_iip.svi
   ```

   Compile file is:

   ```
   SVT_CXL_SUBYSTEM_COMPILE_FILE= compile_snps_iip_cxl_pcie_serial.f
   ```

Note To get the clean compile, ensure that you have loaded Design Compiler in your environment before compiling the files. Do module load of the Design Compiler tool that would set the environment variable - SYNOPSYS.

```
% echo $SYNOPSYS
```

The compile file - `compile_snps_iip_cxl_pcie_serial.f` checks `$SYNOPSYS`, which must be setup by Design Compiler.

```
+incdir+./sim
    +incdir+./src
    +incdir+./src/include
    -y $(IIP_PATH)/src/include

+libext+.v+.sv+ -y $SYNOPSYS/dw/sim_ver
```

- An example set of compile files are delivered as part of product and can be found under *tb_dut_cxl/dut/iip_<version>/* directory. These compile files are DWIP version dependent and needs to be picked based on installed DWIP version.

For example,
`SVT_CXL_SUBYSTEM_COMPILE_FILE=dut/iip_5.95a/compile_snps_iip_cxl_pcie_serial.f`

8. Run the example using the following command line options:

   Before running the example, if there is a requirement to add some configurations with respect to VIP setting then you can do the following in the function- `create_cfg()` of the test

   Example: To make the VIP to move from Gen1 to Gen5. This configuration helps you to bypass the equalization:

```
/*
virtual function void create_cfg();

super.create_cfg();

cust_cfg.host_cfg.cxl_io_cfg[0].pcie_cfg.pl_cfg.set_link_eq_attribute_values(svt_pcie_pl
_configuration::LINK_EQ_MODE_EQ_BYPASS_TO_HIGHEST_RATE);
   cust_cfg.host_cfg.cxl_io_cfg[0].pcie_cfg.pl_cfg.set_link_speed_values(32'h0000_003E,
`SVT_PCIE_SPEED_32_0G, `SVT_PCIE_SPEED_32_0G);
   cust_cfg.host_cfg.cxl_io_cfg[0].pcie_cfg.pl_cfg.highest_enabled_equalization_phase =
3;

endfunction

*/
```

   Command to run the example:

```
gmake cxl_io_random_mem_wr_rd
SVT_CXL_SUBSYSTEM_COMPILE_FILE=compile_snps_iip_cxl_pcie_serial.f
SVT_CXL_SUBSYSTEM_TOPOLOGY_FILE=topology_snps_cxl_pcie_iip.svi USE_SIMULATOR=vcsvlog
VERBOSE=1 WAVES=fsdb
```

B
Running PCIe TS Tests inside tb_dut_cxl

For running any PCIe TS test inside `tb_dut_cxl`, follow the steps mentioned in this chapter. To extract the `tb_dut_pcie_gen5_file`, you must follow the command mentioned in the section, "Setting up Test Suite and Testing for Sanity of Installation".

B.1 Pre-Checklist

You must ensure the following points before running a PCIe TS test case inside `tb_dut_cxl`:

1. TL and DL Layer PCIe TS tests initiate traffic from DUT side. So, the tests either require Synopsys APP BFM over DUT or need a wrapper to execute.
2. DUT LTSSM State and Link-up status attributes are updated.
3. Ordering Tests require Ordering APP (DUT side also if DUT natively do not support Ordering).

   Refer the *PCIe TestSuite User Guide* and *Integration Guide* for more details.

B.2 Test Execution Steps

1. Include the PCIe test name inside the test list (`cxl_io_test_list.svi` or user defined test list).
2. Edit the `Makefile` to update `$scenario_list` and other things for the imported test.
3. Remove the macro `SVT_CXL_SUBSYSTEM_EXCLUDE_PCIE_TS_DATA` which exclude PCIe TS dependency from CXL TS from `sim_build_option` file.

Many PCIe Test Suite tests are not applicable to CXL Specification, and hence you must exclude those. Contact Synopsys support for CXL refined PCIe Test Suite test list.

Some generic exclusions are Gen1/2/3/4 PL tests, GEN6 Feature tests, DL Flit tests, and so on. Reference Command line to execute PCIe TestSuite Tests in CXL Framing:

```
gmake svt_cxl_subsystem_base_test SVT_CXL_SUBSYSTEM_COMPILE_FILE=
compile_snps_vip_pcie_serial.f SVT_CXL_SUBSYSTEM_TOPOLOGY_FILE=
topology_snps_vip_cxl_b2b.svi USE_SIMULATOR=vcsvlog
CMDLINE_VCSPLUSARGS="+uvm_set_inst_override=svt_cxl_pcie_base_test,<PCIE TS Test
name>,uvm_test_top.pcie_tests_0,+EN_CXL_ENUMERATION_FOR_PCIE_TS_TEST_IN_CXL=1"
```

Note

These are some of the key information:

- Base Test used in this is CXL Subsystem base test, and hence you must pass user-created base test. This mainly configures the `CXL.io/PCIe` TS for CXL/User defined configurations.
- PCIe TS test is overridden through PLUSARGS, and this step mainly replaces internally created PCIe TS base test with user executing tests so that applicable configurations and sequence of that test becomes applicable.

Note You can contact Synopsys support to get the list of PCIe test suite tests that can be used in CXL Test Suite.

B.3 Enabling CXL Enumeration via PCIe TS Tests

- While executing PCIe TS tests, CXL Memory mapped registers enumeration support is added in CXL TS. PCIe TS configurations have been added in the `"svt_cxl_subsystem_with_pcie_ts_base_test"` for this support.
  - Virtual method `"configure_pcie_ts_for_cxl_enumeration"` is added with the following configuration. This method can be overridden in the extended class.
    - Config DB for `"enable_post_enumeration_user_seq"` is added to setup the Post enumeration user sequence for PCIe TS.
    - The `"enable_user_sequece"` config DB is enabled.
- The `"initiate_enumeration_for_pcie_ts_tests"` is added in CXL Base test, which gets triggered during `"run_phase"`.
  - Wait for PCIe Enumeration to be completed and de-assert the status attribute for further processing.
  - Trigger CXL Memory mapped registers enumeration based on spec version negotiated.
  - Again assert `"enumeration_done"` of PCIe TS status class.
- Sequence `"svt_pcie_ts_device_system_virtual_tl_post_enumeration_seq"` is added in CXL Env sequence collection. Overall responsibility of this sequence is to delay the PCIe TS test execution after enumeration so that CXL memory mapped registers can be enumerated first.
  - This sequence waits for `"enumeration_done"` to be de-asserted first and then wait for assertion.
  - Alternate implementation is available in the sequence to use `"delay"` based approach.
    - You can set the config-db for `"enumeration_status_or_delay_based_framework"` and delay via `"hold_pcie_test_exec_delay_in_ns"` to enable this feature.

Plusargs is added to enable CXL Enumeration for PCIe TS test:

```
CMDLINE_VCSPLUSARGS=+EN_CXL_ENUMERATION_FOR_PCIE_TS_TEST_IN_CXL=1
```

Note This plusargs is applicable only for PCIe TS tests running in CXL Mode.