SYNOPSYS®

VC VERIFICATION IP

PCIE TEST SUITE

CONFIGURABLE UNIFIED TESTBENCH FOR PHY, RC AND EP DUT

USAGE NOTES

VERSION M-2016.12-1

Contents

1.0 **Introduction**

This document provides an overview of key features and usage notes of Unified testbanch. A new testbench area tb_dut_pcie is introduced in 2016.09-1 release. This envirionment can be used to run cases with different configurations and attributes in the same TB area with different command line/topology file/compile file options.

This document contains key features and usage notes for PHY-DUT & EP controller testbench.

2.0 **Key Features**

The following are the key features in the release of test suite.

2.1 Simulator Support: VCS

2.2 Unified Test Environment

A single Unified Test Env ironment “`tb_dut_pcie`” has been created to address the needs of DUT = EP/ RC/ PHY. In addition, it also supports setting topologies with multiple independent pcie links.

Refer *pcie_svt_unified_single_port_device_usage_notes* for more information on PCIe Unified VIP version 2016.09-3.

Note: M-2016.12-1 release supports EP and RC Controller and PHY DUT. It is recommended to run demo cases only in RC Controller.

2.3 Test Features

PHY DUT : For multi-link topology this release supports to run same or different test on each of the links.

3.0 Unified Test Bench Architecture

3.1 Overall Framework

The following figure shows the the general architecture of the Unified test bench.

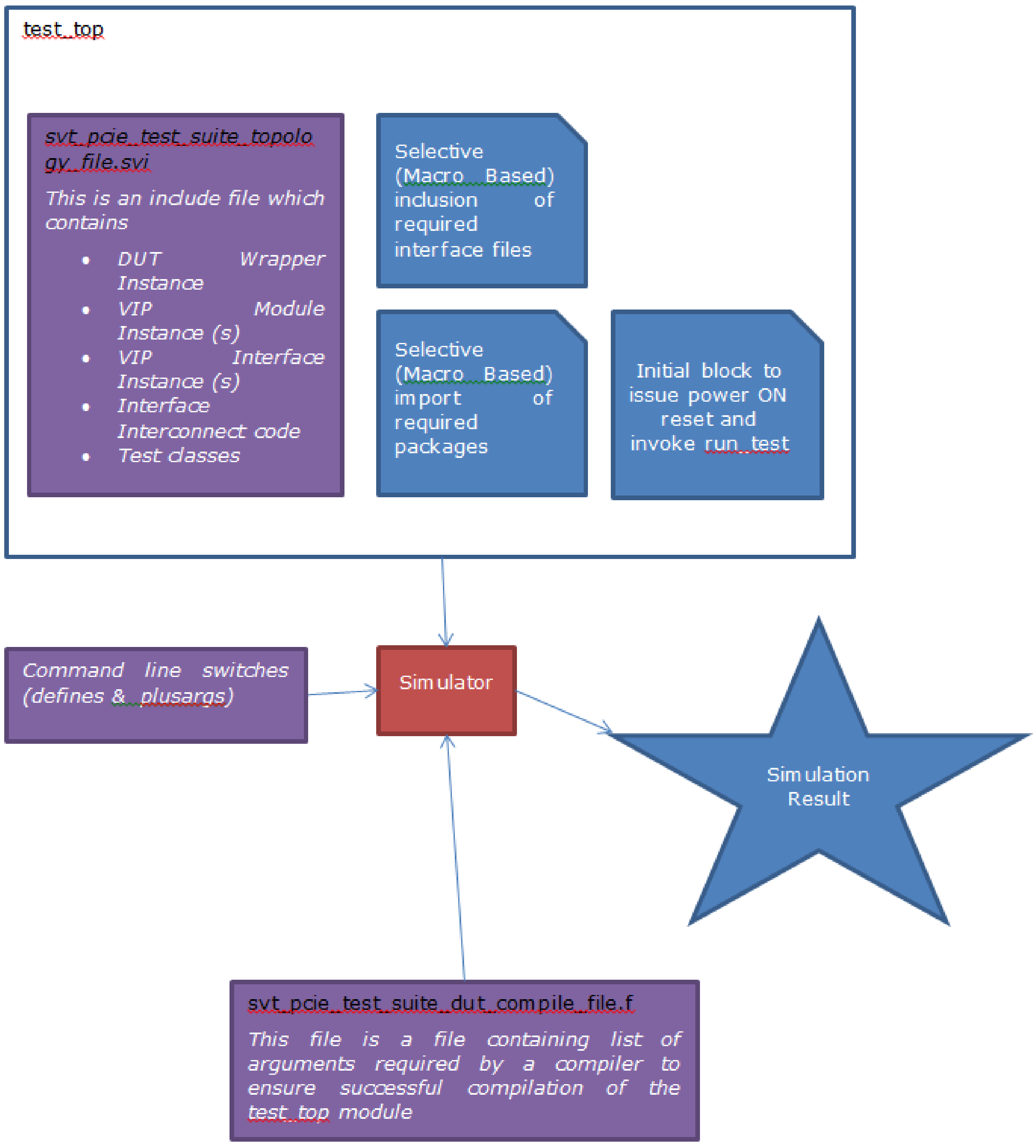

test_top
svt_pcie_test_suite_topology_file.svi
This is an include file which contains
DUT Wrapper Instance
VIP Module Instance (s)
VIP Interface Instance (s)
Interface Interconnect code
Test classes
Selective (Macro Based) inclusion of required interface files
Selective (Macro Based) import of required packages
Initial block to issue power ON reset and invoke run_test
Command line switches (defines & plusargs)
Simulator
Simulation Result
svt_pcie_test_suite_dut_compile_file.f
This file is a file containing list of arguments required by a compiler to ensure successful compilation of the test_top module

Fig 3.1: general architecture of the Unified test bench

The arrangement shown here allows users to model different PCIe Device/System Topologies and leverage a common infrastructure.

The key features of the arrangement are as follows:

- Support multiple independent DUT's / DUT topologies in same directory/infrastructure

- DUT requirements related to non-pcie interfaces are contained in DUT wrapper files. Only the pcie connectivity is exposed to upper modules through `svt_pcie_if` type port.

3.2 Mechanics of SVT_PCIE_TEST_SUITE_DUT_COMPILE_FILE

Create a file containing all the compiler directives required to compile the DUT RTL/DUT Wrapper . This file is then passed as an input to simulator through Makefile switch `SVT_PCIE_TEST_SUITE_DUT_COMPILE_FILE=<dut_specific_compile_switches.f>`

This switch is sticky in nature that is, if for a given run, and if you do not use the switch then the testbench will continue to use the last value passed through this switch, if you have never used the switch since creation of the example then the value (that is, the reset/default) value of the switch is `compile_dut_snps_pcie_ep_vip_serdes.f`

See the **compile_dut_snps_pcie_ep_vip_serdes.f** for example file content .

3.3 Mechanics of SVT_PCIE_TEST_SUITE_TOPOLOGY_FILE

Create a file containing the instance of DUT wrapper module and VIP's along with the interconnect code and list of applicable tests. This file is parsed by simulator at compile time. You can point to any topology file using Makefile switch `SVT_PCIE_TEST_SUITE_TOPOLOGY_FILE=<dut_specific_topology.svi>.`

This switch is sticky in nature that is, if for a given run, and if you do not use the switch then the testbench will continue to use the last value passed through this switch, if you have never used the switch since creation of the example the the value (that is, the reset/default) value of the switch is `topology_dut_snps_pcie_ep_vip.svi`

See the **topology_dut_snps_pcie_ep_vip.svi** for example file content.

3.4 Mechanics of SVT_PCIE_TEST_SUITE_PLUSARG_FILE(for PHY-DUT only)

If you have to supply the run time switches to simulator they can leverage the following gmake switch `SVT_PCIE_TEST_SUITE_PLUSARG_FILE=<user_defined_file_a>`. If specified the Makefile simply invokes the simulator with “-f <user specified file>”.

See the **tests/dual_link_x2_x4.scr** for more example file content.

3.5 Command line switches

Please refer the below command line switches:

Some of the command line switches listed below translate into Verilog defines and some into Verilog plusargs .

The defines that are result of the command line switches stored are stored as part of file svt_pcie_test_suite_cmd_line_defines.svi (created at time of launching simulation) and compiled via inclusion in top.sv.

Alternatively users can choose to place these switches as part of the file passed via switch `SVT_PCIE_TEST_SUITE_DUT_COMPILE_FILE and avoid inclusion of` svt_pcie_test_suite_cmd_line_defines.svi .

Command Line Option	**Actual Verilog Define**	**Actual Verilog Plusarg**	**prescript**	**Comments**
SVT_PCIE_SERDES_TB	SERDES_TB	X	Supported	
SVT_PCIE_AXI_WIDTH	X	svt_pcie_axi_width	Supported	Applicable for RC and EP IIP modes only
SVT_PCIE_DISABLE_EQUALIZATION_SETUP_IN_APP_BFM	X	svt_pcie_disable_equalization_setup_in_app_bfm	Supported	Applicable for RC DUT only
SVT_PCIE_DISABLE_SRIOV	X	svt_pcie_disable_sriov	Supported	
SVT_PCIE_ENABLE_BACKDOOR_CFG_UPDATES	X	svt_pcie_enable_backdoor_cfg_updates	Supported	
SVT_PCIE_ENABLE_COVERAGE	X	svt_pcie_enable_coverage	Supported	
SVT_PCIE_ENABLE_ENUMERATION	X	svt_pcie_enable_enumeration	Supported	

SVT_PCIE_ENABLE_EXTERNAL_APP_BFM	SVT_PCIE_TEST_SUITE_DUT_DRIVER_WITH_EXTERNAL_APP_BFM	X	Supported	
SVT_PCIE_ENABLE_LANE_SKEW	X	svt_pcie_enable_lane_to_lane_skew	Supported	
SVT_PCIE_ENABLE_PA		svt_pcie_enable_pa	Supported	
SVT_PCIE_ENABLE_POLARITY_INVERSION	X	svt_pcie_enable_polarity_inversion	Supported	Applicable Only for topologies with serial bus
SVT_PCIE_ENABLE_TRANSACTION_LOGGING	X	svt_pcie_enable_transaction_logging	Supported	Applicable Only for topologies with serial bus
SVT_PCIE_ENABLE_TRANSACTION_LOGGING	X	svt_pcie_enable_transaction_logging	Supported	
SVT_PCIE_LINK_SPEED	X	svt_pcie_link_speed	Supported	
SVT_PCIE_MAX_LINK_WIDTH	SVT_PCIE_TEST_SUITE_MAX_LINK_WIDTH	X	Supported	
SVT_PCIE_MAX_PPM	X	svt_pcie_max_ppm	Supported	Applicable for serial interface only .
SVT_PCIE_MIN_PPM	X	svt_pcie_min_ppm	Supported	Applicable for serial interface only .

SVT_PCIE_PIPE_WIDTH	X	svt_pcie_pipe_width	Supported	
SVT_PCIE_SSC_MODE	X	svt_pcie_ssc_mode	Supported	Applicable for serial interface only .
SVT_PCIE_TARGET_LINK_WIDTH	X	svt_pcie_target_link_width	Supported	
SVT_PCIE_TEST_SUITE_PIPE_SPEC_VER_4_2_PCLK_INPUT	SVT_PCIE_TEST_SUITE_PIPE_SPEC_VER_4_2_PCLK_INPUT	X	Supported	If VIP instances are created via helper macros
SVT_PCIE_TEST_SUITE_PIPE_SPEC_VER_4_3	SVT_PCIE_TEST_SUITE_PIPE_SPEC_VER_4_3	X	Supported	If VIP instances are created via helper macros
SVT_PCIE_TEST_SUITE_PLUSARG_FILE	X	X		
SVT_PCIE_TEST_SUITE_TOPOLOGY_FILE	X	X	Supported	
SVT_PCIE_TEST_TIMEOUT	X	UVM_TIMEOUT	Supported	

Table : 3.5 : command line and switches

4.0 PHY DUT

4.1 Summary of Changes

4.1.1 Changes in release L-2016.9-03

Added Gen4 speed support. You can use SVT_PCIE_LINK_SPEED=GEN4, for more details, see Known Limitations and Issues.

4.1.2 Changes in release L-2016.9-02

- Compile file for PHY as DUT in tests directory 'compile_dut_snps_pcie_phy_vip.f' is removed to avoid any confusion. There are other 2 pre-existing compile files which describes a proper configuration namely

  compile_dut_snps_pcie_phy_vip_ep_on_pipe.f
  compile_dut_snps_pcie_phy_vip_rc_on_pipe.f

- A new file added in tb_dut_pcie directory 'cust_pre_tb_top.svi' which is included in `top.sv`. Using this file Customer can add `define or `include directives that may be conditional in nature (based on defines in listed compile file)

  Testcases related to RC are enabled in 'pcie_rc_dut_testlist.svi'. Affected tests:
  ts.tl_no_link_retry_with_ep_bit_set-rc.sv

  ts.tl_random_driver_exceptions-rc.sv

  ts.tl_random_target_exceptions-rc.sv

- compile_dut_snps_pcie_bifurcated_phy_vip.f is updated with :

  // Phy DUT testsuite relies on the presence of Macro
  SVT_PCIE_TEST_SUITE_EP_IS_DUT

  > +define+SVT_PCIE_TEST_SUITE_EP_IS_DUT

4.1.3 Changes in release L-2016.9-01

- Name changes to all the testcases with respect to PHY, EP, RC TB Areas. With suffix -phy, -ep, -rc respectively.
- Equivalent name changes in env/pcie_phy_dut_testlist.svi.
- Added rest of the testcases including Gen4.
  - Number of new Gen4 cases : 69
- In the current release, it is recommended to run EP Controller testcases only (with suffix -ep). Other cases are part of the release but not advised to run.

The L-2016.09-1 release provides configurable PHY DUT test bench environment, which supports verification of single port (single PCIE link) and multi-port (multiple PCIE links) topologies for PHY DUT.

A single Unified Test Env ironment "`tb_dut_pcie`" has been created to address the needs of DUT = EP/ RC/ PHY. In addition, it also supports setting topologies with multiple independent pcie links.

To address the need of run time configurable signal connections between DUT and PCIe VIP the Unified Test Environment makes use of a new sub-block N-Furcation mux. With this feature added the user can compile a single max-link topology (that is, a topology supporting maximum number of PCIe links) but alter the signal connections at runtime (i.e. without recompile) and in effect choose a runtime topology with number of operational PCIe links less than or equal to maximum links.

The above mentioned features are implemented using unified SVT PCIE VIP model (that is, "svt_pcie_single_port_device"). Each VIP instance can support unique compile time options as we do not need to use `defines to configure them.

Refer *pcie_svt_unified_single_port_device_usage_notes* for more information on PCIe Unified VIP.

4.2 Known Limitations and Issues

1. Re-use of existing tests (tests developed as part of single link DUT test suite) in multi link env relies on.
   - Absence of test code doing hierarchical access to VIP internal nodes.
   - Test suite configuration flag "enable_integration_in_user_environment" set to 0.
   - Demotion of error messages due to duplicate call back registrations.
2. The following test cases are applicable:

Test cases	**Applicable Speed**	**Remarks**
Common test cases `tl_*/pl_*/dl_*`	Gen1/Gen2/Gen3*	Final link-up will be based on their corresponding speed only.
Gen1 test cases `(gen1_pl)`	Gen1/Gen2/Gen3*	Final link-up will be on Gen1 speed only.
Gen2 test cases `(gen1_pl/gen2_dl)`	Gen2/Gen3*	Final link-up will be on Gen2 speed only.
Gen3 test cases `(gen3_pl)`	Gen3*	Final link-up will be on Gen3 speed only.
Gen4 test cases `(gen4_pl)`	Gen4**	Final link-up will be on Gen4 speed only.

* Test cases common/Gen1/Gen2/Gen3 as of now not applicable for Gen4 speed.

**All Gen4 test cases are applicable only for Gen4 speed.

3. The following five tests are failing in Unified TB area due to VIP PHY model limitation, These tests

are applicable after 4.0 PIPE specification (New in 4.2 or 4.3):
The tests are successfully validated in EP and RC DUT area.
- gen3_pl_recovery_equalization_legal_preset_request_dynamic_phy-phy

- gen3_pl_recovery_equalization_link_eval_direction_change-phy
- gen3_pl_recovery_equalization_link_eval_fom_change-phy
- gen3_pl_recovery_equalization_phase2_coefficient_eqeval_abort_error-phy
- gen3_pl_skip_err_check_rxstatus_error-phy

4. The following tests are failing in PHY DUT TB area with SVT_PCIE_TEST_SUITE_RC_ON_PIPE switch due to VIP PHY model limitation (VIP PHY model does not support signals meant for equalization evaluation procedure):
    - gen4_pl_recovery_equalization_coefficient_setting_error
    - gen4_pl_recovery_equalization_preset_setting_error

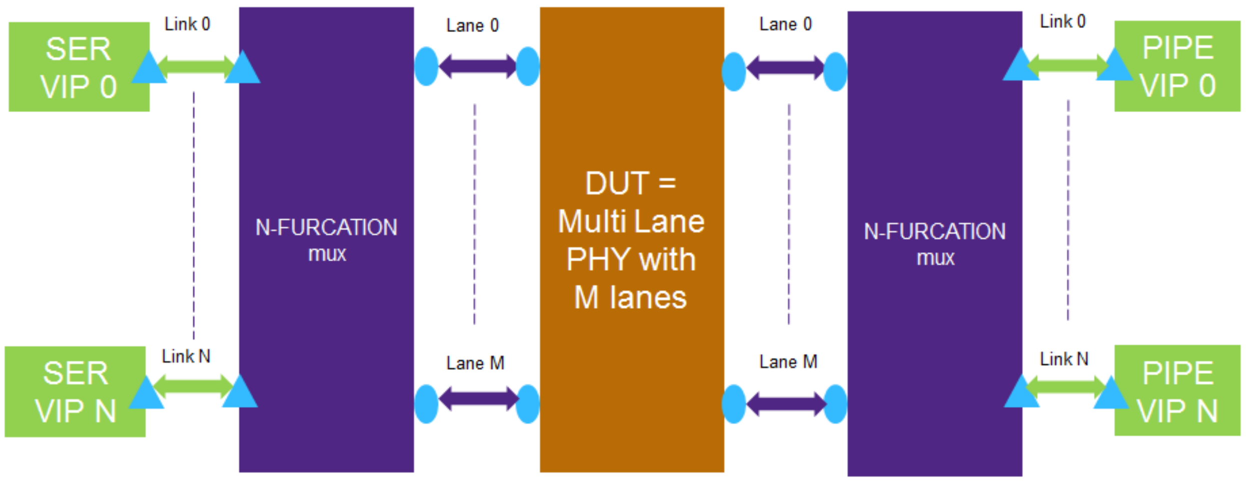
4.3 Simulator Command Usage Summary

```
gmake USE_SIMULATOR=<vcsvlog> \
<testname: class to be instantiated using UVM_TESTNAME> \
SVT_PCIE_TEST_SUITE_DUT_COMPILE_FILE=<dut_specific_compile_switches.f
> \
SVT_PCIE_TEST_SUITE_TOPOLOGY_FILE=<dut_specific_topology.svi>
SVT_PCIE_TEST_SUITE_PLUSARG_FILE=<tests/dual_link_x2_x4.scr> \
```

Note: As part of legacy test suite a lot of command line switches were added to Makefile to alter the default test behavior. These makefile switches resulted in addition of `+defines` or `plusargs` which were global in nature (that is they affect all link operations and all VIP instances). Unified Test bench approach does not support these independent Makefile switches and recommends use of above listed files to achieve the same result.

4.4 Multi-Lane PHY as DUT in Unified Test Environment(for PHY-DUT only)

This section describes the general architecture of the PHY DUT test bench. The following figure provides the top level diagram of usage model for PHY DUT verification.



▲ svt_pcie_if ● svt_pcie_x1_if

svt_pcie_if: Interface capable of handling any type of interconnect (PMA,Serdes,PIPE), Used to form a *multiple lane link. Connects VIP signals and N-Furcation Mux*

svt_pcie_x1_if : Interface capable of handling any type of interconnect (PMA,SERDES,PIPE), Used to form a *single lane link. Connects DUT signals and N-Furcation Mux*

The RC ,EP, and N-Furcation Mux VIP module instances are part of the top module scope in the environment.

The maximum number of physical lanes used by each VIP instance is configurable through compile time parameter. The actual link width of PCIe link is configurable using run time settings.

Based on PHY design requirements it is possible to use the N-Furcation Mux and share PHY DUT lanes across multiple VIP instances.

For example, With 16 lanes at PIPE / Serdes interface the following topologies can be supported.

- 1 link with 16 lanes (x1,x1..)
- 2 links with 8 lanes (x8,x8)
- 4 links with 4 lanes (x4, x4, x4, x4)
- 8 links with 2 lanes (x2, x2...)

More details on usage of N-Furcation mux are listed in section 7.0

4.5 Re-use of Existing Single Link Tests in Multi-Link Topologies

The UVM Methodology permits use of a single top level test class instance whose type is determined using run time switch `UVM_TESTNAME`.

Considering the fact that each existing single link test is self-sufficient to operate independently on a single link (collection of environment instance ,config object instance and some error demotion logic), the requirement to re-use the existing test in multi link topology could be best addressed by creating separate/multiple instances of these test classes.

To meet the technical requirement stated above without violating the constarints laid down by `UVM_METHODOLOGY` a new class `pcie_unified_base_test` has been introduced and test component arrangement is as per the following figure.

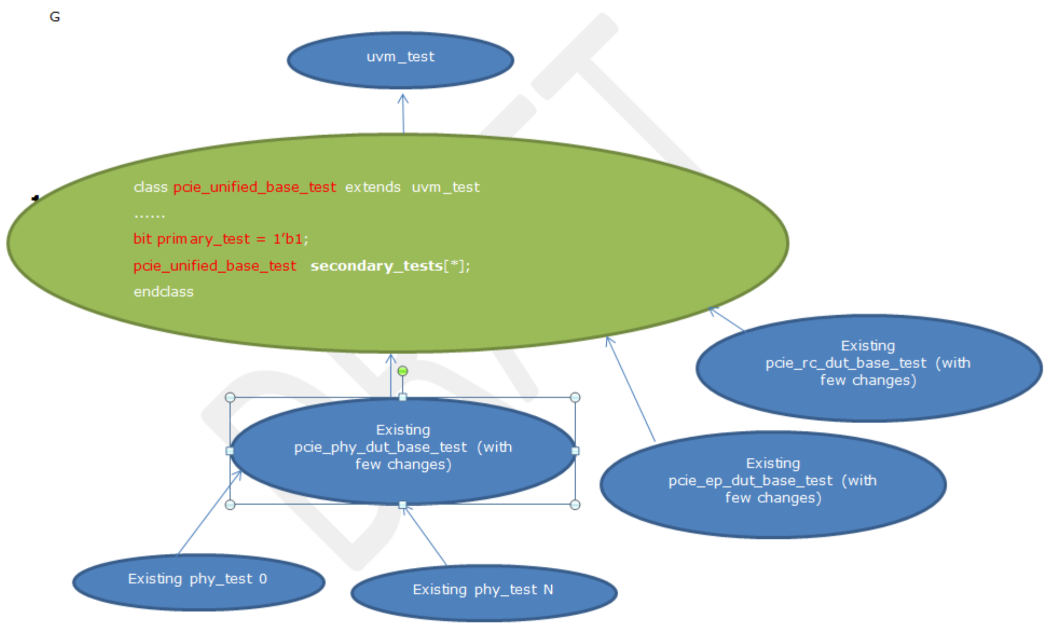

G
uvm_test
class pcie_unified_base_test extends uvm_test
......
bit primary_test = 1'b1;
pcie_unified_base_test secondary_tests[*];
endclass
Existing pcie_rc_dut_base_test (with few changes)
Existing pcie_phy_dut_base_test (with few changes)
Existing pcie_ep_dut_base_test (with few changes)
Existing phy_test 0
Existing phy_test N

Fig 4.3 : tb_dut_pcie class heirarchy

When `run_test` is invoked by initial block in top.sv, the first (primary) instance of uvm_test component is created (type is determined by UVM_TESTNAME), during it's build phase execution this test instance does the following things

1. Executes factory override (`set_type_override_by_name`) for `pcie_unified_base_test` with type requested by UVM_TESTNAME.
2. Retrieves hex plusarg `svt_pcie_disable_link_creation`.(See 4.6)
3. Retrieves compile time topology from uvm_config_db (that is,. `highest_link_num`, handles to s`vt_pcie_unified interfaces`).
4. If required (based on information from 2 and 3) it invokes create calls for one or more instances of pcie_unified_test and populates the array secondary_tests[*] with handles of these newly created child (secondary) test instances.
5. Flips the value of `primary_tes`t flag in any of the newly created secondary test instances (this prevents recursive creation of test instances)
6. Executes the legacy flow of setting up the configuration object and the environment instance.

Note: By default, any code that is part of test class is going to be independently executed for each of the test instances, if the code needs to be executed only once then it should be guarded with if (primary_test) begin ..... end.

4.6 Details of plusarg svt_pcie_disable_link_creation

Need for "svt_pcie_disable_link_creation". The compile time topology may consist of N links but this can be changed at run time. This plusarg informs the `build_phase of uvm_test_top` which links from compile time configuration are actually active.

The default setting of `uvm_test. disable_link_creation` variable is 32'hFFFF_FFFE that means use create only one test instance and assign it to link 0 from compile time topology.

- For operating with 2 links, this pluarg value must me 32'hFFFF_FFFC.
- For operating with 3 links, this pluarg value must me 32'hFFFF_FFF8.Not supported by example topology but supported by the base test.
- For operating with 32 links, this pluarg value must me 32'h0. Not supported by example topology but supported by the base test.

4.7 VIP Dependancy and Test Cases

The L-2016.06-T0715 version of test suite deliverable is qualified with L-2016.06-2 release of VIP. The supported simulator version is VCS 2016.06.

4.8 Test Bench Area

This release includes only unified test bench area. The TB area name is *tb_dut_pcie.*

4.9 Test Cases

The following tests were included as part of the previous release, these tests are used to exercise the N-Furcation mux functionality.

- `ts.demo_gen3_pl_all_supported_speed_change_test-phy.sv`

- `ts.demo_pl_gen1_linkup_test-phy.sv`
- `ts.demo_pl_gen1_to_gen2_speed_change_test-phy.sv`
- `ts.demo_pl_gen1_to_gen3_speed_change_test-phy.sv`
- `ts.demo_tl_gen1_linkup_followed_by_tlps-phy.sv`
- `ts.demo_tl_gen2_speed_change_followed_by_tlps-phy.sv`
- `ts.demo_tl_gen3_speed_change_followed_by_tlps-phy.sv`
- `ts.tl_directed_traffic-phy.sv`

In addition to the above tests this release has 234 phy test files (tb_dut_pcie/tests/ts.*-phy.sv). tb_dut_pcie/env/pcie_phy_dut_testlist.svi has the complete list of tests available in this release.

All these tests have been originally developed using the Synopsys PCIe single link Phy VIP model as DUT as part of the legacy test suite. These tests have now been ported to the unified test suite area.

Most of these tests are also exercised with Synopsys PCIe Bifurcated Phy VIP model as DUT operating in single link mode. For multi link Phy DUT few tests have been excluded due to backwawrd compatibility issues with legacy test suite.These exclusions will be addressed in future release. For more details about the excluded tests refer tb_dut_pcie/compile_dut_snps_pcie_bifurcated_phy_vip.f with lines +define+SVT_PCIE_TEST_SUITE_GUARD_<TESTNAME>

4.10 PHY DUT Integration Steps

Step 1: Create a DUT wrapper file similar to `env/svt_pcie_bifurcated_phy_model.sv`.

To simplify connection in topology file we ensure the DUT wrapper port(s) are of type `svt_pcie_x1_if`. Any other interconnect requirement can be handled within the DUT wrapper file .

Step 2: Create custom compile time topology files with user defined name to meet the needs for PHY DUT verification.

Reference: `topology_dut_snps_pcie_bifurcated_phy_vip.svi`. This file shows 4 instances of VIP forming 2 links using a 8 lane Phy as DUT. The model `svt_pcie_bifurcated_phy_model.sv` is a scaled down version of real Phy DUT. It can only support 1 link or 2 links operation and the second link must use lane 4.

Step 3: Create custom defined compile file with user defined name and include your DUT wrapper specific compile directives example `defines,+incdir and so on.

Reference: **c**`ompile_dut_snps_pcie_bifurcated_phy_vip.f`

Step 4: DUT initialization sequence.

As the test sequences are started during the `main_phase` (sub phase of run_phase) any DUT specific initialization sequence must be placed in sub phases of run_phase preceeding the main_phase.

Step 5: Overriding compile time topology using Run time switches Plusargs.

You can distribute the Phy Lanes across different VIP instances by modifying their targeted link width settings. The hook to do this is by loading plusargs through file supplied as input using `SVT_PCIE_TEST_SUITE_PLUSARG_FILE` option.

Reference: `tests/dual_link_x2_x4.scr`

// Following plusarg does width over ride for link 0

```
+cfg[0]=ep_cfg.pcie_cfg.pl_cfg.link_width:2,rc_cfg.pcie_cfg.pl_cfg.li
nk_width:2
```

// Following plusarg does width over ride for link 1

```
+cfg[1]=ep_cfg.pcie_cfg.pl_cfg.link_width:4,rc_cfg.pcie_cfg.pl_cfg.li
nk_width:4
```

// Following plusarg setting does instance override of test type, thus facilitating running different tests on different links

```
+uvm_set_inst_override=pcie_unified_base_test,demo_pl_gen1_linkup_tes
t,uvm_test_top.secondary_tests[1]
```

4.11 Running Test Cases

- **Compile and Run (Default setting)**

  gmake tl_directed_traffic USE_SIMULATOR=vcsvlog

  Results in traffic on single link:

  Rate Gen 3 and Width = 8 (Max Width Supported by the DUT model)

- **To run same test sequence on 2 links** (run without recompiling)

  gmake tl_directed_traffic USE_SIMULATOR=vcsvlog \
  SVT_PCIE_TEST_SUITE_PLUSARG_FILE=tests/dual_link_x4_x4.scr NOBUILD=1

  Results in traffic on 2 links:

  Link 0 : Rate Gen 3 and Width= 4

  Link 1 : Rate = Gen 3 and Width = 4

- **To run different test sequence on 2 links** (run without recompiling)

  gmake tl_directed_traffic USE_SIMULATOR=vcsvlog \
  SVT_PCIE_TEST_SUITE_PLUSARG_FILE=tests/dual_link_x2_x4.scr NOBUILD=1

  Results in traffic on 2 links

  Link 0 : Rate Gen 3 and Width= 2

  Link 1 : Rate = Gen 1 and Width = 4

Note : All dual_link_x2*.scr files demonstrate running a different test sequences on link 1 and link 0 because these files include extra plusarg `+uvm_set_inst_override=pcie_unified_base_test`.

The Sample Message content from Link 0 related uvm_components:

UVM_INFO ./vip/src/sverilog/vcs/pciesvc_ltssm.sv(8614) @ 88601549.10 ps: uvm_test_top.env.rc_env.rc_agent.port0.pl0 [report_message] LTSSM: Link training completed. The link is READY! Speed is 8Gb/s. Link width is 1. Link number is 0. Scrambling is ENABLED.

The Sample Message content from Link 1 related uvm_components:

UVM_INFO ./vip/src/sverilog/vcs/pciesvc_ltssm.sv(8614) @ 82211111.50 ps: uvm_test_top.secondary_tests[1].env.rc_env.rc_agent.port0.pl0 [report_message] LTSSM: Link training completed. The link is READY! Speed is 2_5Gb/s. Link width is 4. Link number is 0. Scrambling is ENABLED.

4.12 svt_pcie_nfurcation_mux Usage Details

4.13 Assumptions

- Connections to Physical Lane 0 of each VIP instance (active at Run time) will be used to connect shared signals of VIP.
- If a certain DUT lane is not used due to run time topology settings then the signals corresponding to it will be left in unkown state (that is, x).

4.14 Configurable Parameters

Parameter Name	**Default Value**	**Description**
`SVT_PCIE_NMUX_MAX_NUM_VIP_IFS`	2	Indicates the number of VIP instances that share the DUT interface Valid values 1 to 31
`SVT_PCIE_NMUX_MAX_NUM_X1_LANES`	4	Indicates the maximum number of lanes that the shared DUT supports.
`SVT_PCIE_NMUX_UVM_CFG_DB_RELATIVE_SCOPE`	uvm_test_top	String indicating which uvm_component(s) can access the API class instance through `uvm_config_db calls`.
`SVT_PCIE_NMUX_UVM_CFG_DB_VAR_NAME`	numx_api	String indicating what value should be used by uvm_component(s) to

		access the API through `uvm_config_db calls`.

4.15 Signal interface

`vip_ifs[SVT_PCIE_NMUX_MAX_NUM_VIP_IFS]`	An array of interface of type **svt_pcie_if** to connect to VIP instances
`vip_x1_ifs[SVT_PCIE_NMUX_MAX_NUM_X1_LANES]`	An array of interface of type **svt_pcie_x1_if** to connect to DUT lanes
`reset_to_unused_vip`	Output signal with width = `SVT_PCIE_NMUX_MAX_NUM_VIP_IFS` . Each bit corresponds to a VIP instance. The logic to generate reset to the VIP instance should use this. It ensures if any of the VIP instances are unused due to run time configuration then we should hold it in reset state.

4.16 Interfacing the Nfurcation Mux Module to class world

Each `svt_pcie_nfurcation_mux` instance is connected to uvm_test instance through the following class pair:

- **virtual class** `svt_pcie_nfurcation_mux_api`
- **concrete class** `svt_pcie_nfurcation_mux_api_impl` **extends** `svt_pcie_nfurcation_mux_api` **impelmented within module** svt_pcie_nfurcation_mux.

An initial begin end block within `svt_pcie_nfurcation_mux` creates an instance of `svt_pcie_nfurcation_mux_api_impl` and deposits its handle in `uvm_config_db`.

pcie_unified_base_test instance (only with primary_test == 1) retrieves this handle during connect_phase and using the methods provided by the above class pair configures the mux with required run time configuration

The method used from the above class pair:

- function int `set_curr_link_width` (int vip_number, int link_width , int max_link_width);
- function int `apply_link_widths();`
- function bit `get_on_off_status_for_vip_if` (int vip_number);

Expected Use Model using the above listed methods:

- Feed run time topology to mux instance using set_curr_link_width calls

```
set_curr_link_width(rc_0_if_index, rc_0_run_time_width,
rc_0_compile_time_width);

set_curr_link_width(ep_0_if_index, ep_0_run_time_width,
ep_0_compile_time_width);
```

- Take action of information fed by preceding set_curr_link_width calls

```
apply_link_widths();
```

- Retrieve status of individual VIP interfaces to determine what should be done with uvm_components corresponding to these VIP during remaining phases (on each VIP instance basis)

```
rc_0_off = get_on_off_status_for_vip_if(rc_0_if_index);

ep_0_off = get_on_off_status_for_vip_if(ep_0_if_index);
```

5.0 **EP and RC Controller**

5.1 Summary of Changes in M-2016.12-1

Added support for Gen4 RC.

5.2 Summary of Changes in M-2016.12

Star 9001093030: Remove XMR from the test cases of PHY DUT.

Updated the following cases to handle XMR issues for multi-link runs(bifurcated_phy):

- `ts.gen2_pl_8b_10b_err_with_skp_removed-phy.sv`
- `ts.gen2_pl_tx_skp_length_1-phy.sv`
- `ts.pl_fifo_overflow_in_all_speeds_error-phy.sv`
- `ts.pl_fifo_underflow_in_all_speeds_error-phy.sv`

The following tests will fail, if run in multi-link(bifurcated_phy). Because the signals accessed are not part of the configuration yet.

- `ts.gen2_pl_recovery_low_swing_with_no_deemphasis.sv`
- `ts.gen3_pl_recovery_equalization_phase2_coefficient_eqeval_abort_error.sv`
- `ts.gen3_pl_skip_err_check_rxstatus_error.sv`
- `ts.pl_recovery_lock_txmargin_check_l0.sv`

5.3 Known Limitations and Issues

Following are the limitations and known issues:

1. The following test cases are applicable:

Test cases	**Applicable Speed**	**Remarks**
Common test cases `tl_*/pl_*/dl_*`	Gen1/Gen2/Gen3*	Final link-up will be based on their corresponding speed only.
Gen1 test cases `(gen1_pl)`	Gen1/Gen2/Gen3*	Final link-up will be on Gen1 speed only.
Gen2 test cases `(gen1_pl/gen2_dl)`	Gen2/Gen3*	Final link-up will be on Gen2 speed only.
Gen3 test cases `(gen3_pl)`	Gen3*	Final link-up will be on Gen3 speed only.
Gen4 test cases `(gen4_pl)`	Gen4**	Final link-up will be on Gen4 speed only.

* Test cases common/Gen1/Gen2/Gen3 as of now not applicable for Gen4 speed.

**All Gen4 test cases are applicable only for Gen4 speed.

2. You can use RC controller cases only for demo cases. Although other RC controller cases could be run but may result is unpredictable results because these cases are not thoroughly validated yet.

3. The present set-up of unified testbench is VIP-VIP set-up and do not claim IIP support.

4. EP and RC controller verification environment has not been tested for multilink operation.

   Tese case "gen3_pl_skip_err_check_rxstatus_error-phy" is expected to fail with signature "`Cannot create a component of type gen3_pl_skip_err_check_rxstatus_error' because it is not registered with the factory`". This case is having XMR issue and it is excluded. A STAR 9001093030 is filed to track the issue.

5.4 VIP Dependancy and Test Cases

The M-2016.12-1 version of test suite deliverable is qualified with M-2016.12-1 release of VIP. The supported simulator version is VCS 2016.06-SP1.

5.5 Test Bench Area

This release includes only unified test bench area. The TB area name is *tb_dut_pcie.*

Environment Class Structure for EP DUT Testing

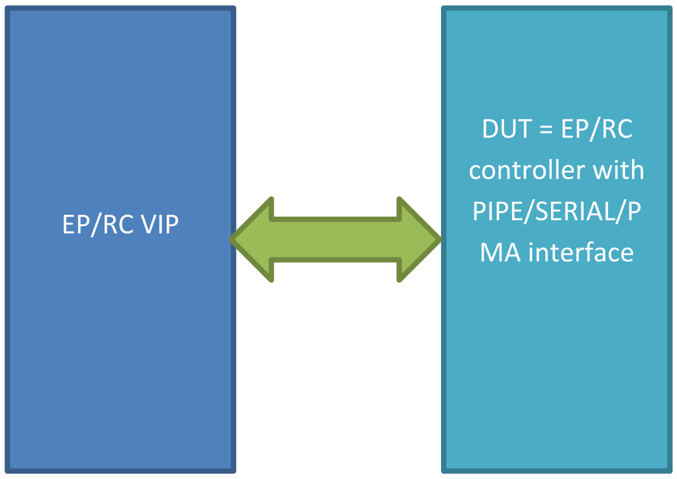


Figure 1 : Introduction of New Pcie_Unified_Base_Test as a Common Layer From Which EP/RC is Extended

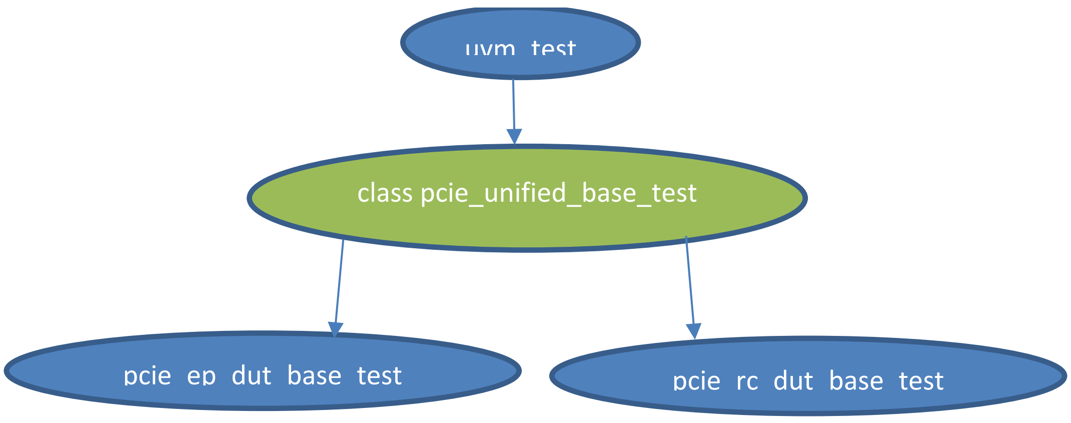
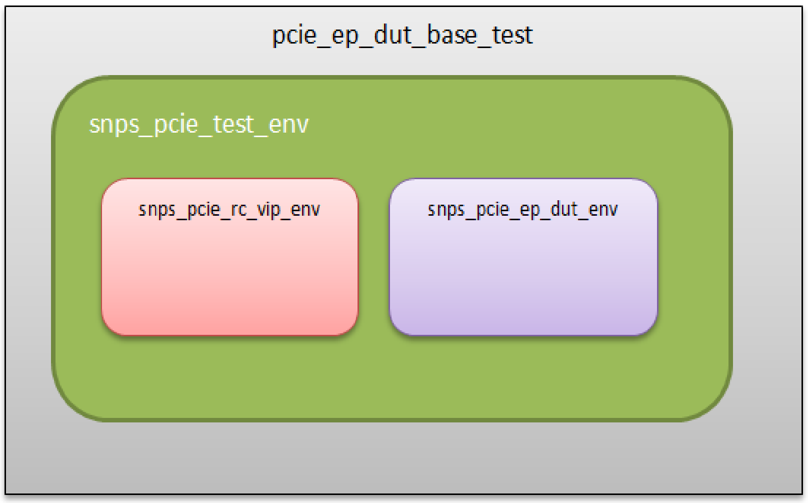


Figure 2: Further Class Structure



5.6 *EP and RC DUT Integration Steps*

Step 1: Create custom compile time topology files with user defined name to meet the needs for EP controller verification.

Reference:

- EP: *topology_dut_snps_pcie_ep_vip.svi*
- RC*: topology_dut_snps_pcie_rc_vip.svi*

Step 2: Create custom defined compile file with user defined name and include your DUT wrapper specific compile directives example `defines,+incdir and so on.

Reference:

- EP: *compile_dut_snps_pcie_ep_vip_serdes.f*

- RC: *compile_dut_snps_pcie_rc_vip_serdes.f*

Step 3: DUT initialization sequence.

As the test sequences are started during the `main_phase` (sub phase of run_phase) any DUT specific initialization sequence must be placed in sub phases of run_phase preceeding the main_phase.

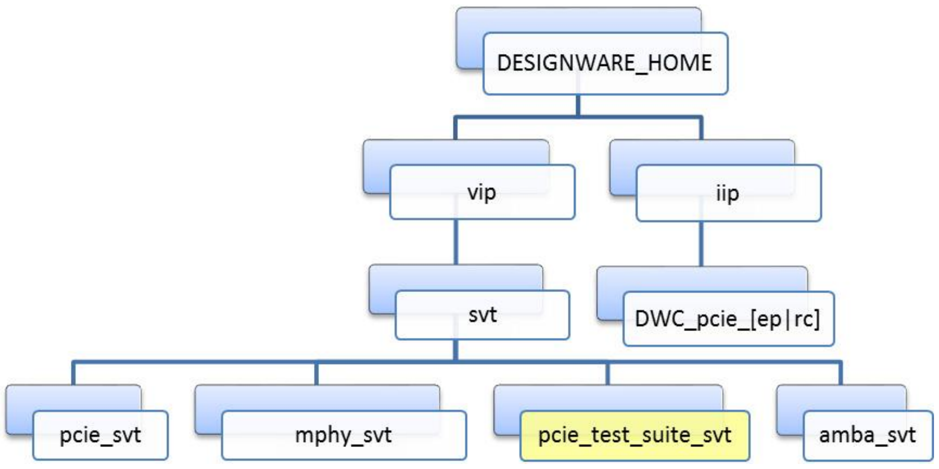
Refer to *env/snps_pcie_test_env.sv* fo rany DUT specific initialization phase.

Provision given in env/pcie_ep_dut_cust_base_test.sv & env/pcie_rc_dut_cust_base_test.sv for DUT implementation-specific customization in the base test. Every test extends from macro `SVT_PCIE_TEST_SUITE_EP_DUT_TEST which points to pcie_ep_dut_cust_base_test for EP controller. macro `SVT_PCIE_TEST_SUITE_RC_DUT_TEST which points to pcie_rc_dut_cust_base_test for RC controller

5.7 Installation and Setup

When unpacked from their self-installer (.run file) packages, the PCIe VIP Test Suite products are installed into the user-specified DESIGNWARE_HOME directory, under the vip sub-tree. Since they are related/dependent products the pcie_svt, mphy_svt, and amba_svt VIP products are also included in the self-installer for the Test Suite product (along with other supporting items such as common and svt). However, the IIP DWC_pcie Core is not included in the PCIe VIP Test Suite product delivery: Users of that Core acquire it using a separate license and delivery.
The basic structure of a user's DESIGNWARE_HOME tree that includes the PCIe IIP Core (purchased and installed separately), and the PCIe VIP Test Suite product is shown in Figure .

Structure of a DESIGNWARE_HOME tree that includes the PCIe IIP Core



Synopsys tools such as coreConsultant and dw_vip_setup are used to set up a workspace that may include a configured core and an example testbench as a starting point.

5.8 Example Testbench Setup

The VIP product setup script is **$DESIGNWARE_HOME/bin/dw_vip_setup**. It is used to set up the example testbench in a specified directory. For example:

```
$DESIGNWARE_HOME/bin/dw_vip_setup -p design-dir -svlog
pcie_test_suite_svt/tb_dut_pcie
```

The directory structure produced by **dw_vip_setup** for setting up a tb_dut_pcie (Unified) Test Suite testbench, starting at the *design-directory*, is shown below

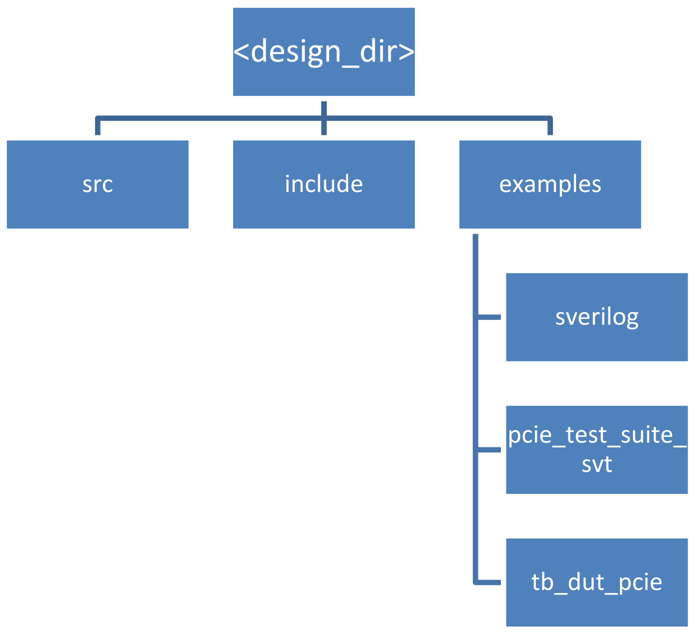


Fig 5.3.1 Directory Structure Produced by dw_vip_setup

5.9 Testbench directory structure

The initial directory structure within the testbench will be as below

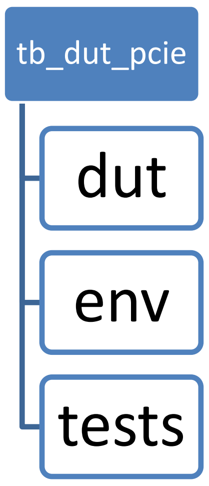


Fig 5.5 : Test bench directory structure.

The general usage of the structure is as follows:
• tb_dut_pcie : Top level of testbench. Includes top level Verilog module definition, Makefile, and scripts supporting the execution of the testbench.

Testcase Naming convention:

Unified testbanch runs the same testcases in different configuration. E.g endpoint as DUT, root as DUT or PHY as DUT. As the same testcases could be used for all different configurations, so a trailing - suffix used to differentiate the testcases. The class name inside the testcase does not have this suffix. It is used to get pass the compilation. Below is the conventions used.

1. PHY : ts.<testcase name>-phy.sv
2. EP (ep as dut) : ts.<testcase name>-ep.sv
3. RC (rc as dut) : ts.<testcase name>-rc.sv

5.10 Simulator Command Usage Summary

```
gmake USE_SIMULATOR=<vcsvlog> \
<testname: class to be instantiated using UVM_TESTNAME> \
SVT_PCIE_TEST_SUITE_DUT_COMPILE_FILE=<dut_specific_compile_switches.f
> \
SVT_PCIE_TEST_SUITE_TOPOLOGY_FILE=<dut_specific_topology.svi>
```

Example :

```
EP:

gmake SVT_PCIE_LINK_SPEED=GEN4 demo_dl_link_up_test-ep
SVT_PCIE_TEST_SUITE_TOPOLOGY_FILE=topology_dut_snps_pcie_ep_vip.svi
SVT_PCIE_TEST_SUITE_DUT_COMPILE_FILE=compile_dut_snps_pcie_ep_vip_ser
des.f

RC:

gmake SVT_PCIE_LINK_SPEED=GEN4 demo_dl_link_up_test-rc
SVT_PCIE_TEST_SUITE_TOPOLOGY_FILE=topology_dut_snps_pcie_rc_vip.svi
SVT_PCIE_TEST_SUITE_DUT_COMPILE_FILE=compile_dut_snps_pcie_rc_vip_ser
des.f
```

5.11 Running Test Cases

- **Compile and Run (Default setting)**

```
gmake SVT_PCIE_TEST_SUITE_TOPOLOGY_FILE=topology_dut_snps_pcie_ep_vip.svi
SVT_PCIE_TEST_SUITE_DUT_COMPILE_FILE=compile_dut_snps_pcie_ep_vip_serdes.
f  USE_SIMULATOR=vcsvlog VERBOSITY=debug
SVT_PCIE_ENABLE_TRANSACTION_LOGGING=1 demo_dl_link_up_test-ep
```

Table 1: Complie and Topology File Combination for PIPE and SERDES

Configuration	Complile File	Topology File
EP PIPE	compile_dut_snps_pcie_ep_vip.f	topology_dut_snps_pcie_ep_vip.svi
RC PIPE	compile_dut_snps_pcie_rc_vip.f	topology_dut_snps_pcie_rc_vip.svi
EP SERDES	compile_dut_snps_pcie_ep_vip_serdes.f	topology_dut_snps_pcie_ep_vip.svi
RC SERDES	compile_dut_snps_pcie_rc_vip_serdes.f	topology_dut_snps_pcie_rc_vip.svi

Results in traffic on single link:

Rate Gen 3 and Width = 32 (Max Width Supported by the DUT model)

For the command line options used dere please refer to section 3.5

The above run command generates the following log files:

- demo_dl_link_up_test_[0]_ep.sym_log

- demo_dl_link_up_test_[0]_ep.xact_log
- demo_dl_link_up_test_[0]_rc.sym_log
- demo_dl_link_up_test_[0]_rc.xact_log

the test_[0] convention in the symbol and xact log signifies which link this log belongs to. [0] signifies that it belongs to link 0.