Verification Continuum™

VC Verification IP
PCIe Test Suite
FAQ

Version S-2021.06, June 2021

SYNOPSYS®

Copyright Notice and Proprietary Information

Destination Control Statement

All technical data contained in this publication is subject to the export control laws of the United States of America. Disclosure to nationals of other countries contrary to United States law is prohibited. It is the reader's responsibility to determine the applicable regulations and to comply with them.

Disclaimer

SYNOPSYS, INC., AND ITS LICENSORS MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Trademarks

Synopsys and certain Synopsys product names are trademarks of Synopsys, as set forth at https://www.synopsys.com/company/legal/trademarks-brands.html.
All other product or company names may be trademarks of their respective owners.

Free and Open-Source Software Licensing Notices

If applicable, Free and Open-Source Software (FOSS) licensing notices are available in the product installation.

Third-Party Links

Any links to third-party websites included in this document are for your convenience only. Synopsys does not endorse and is not responsible for such websites and their practices, including privacy practices, availability, and content.

www.synopsys.com

Contents

Preface

About This Document

This guide provides you a list of frequently asked questions for VC VIP for PCIe Test Suite.

Web Resources

- Documentation through SolvNetPlus: https://solvnetplus.synopsys.com (Synopsys password required)
- Synopsys Common Licensing (SCL): http://www.synopsys.com/keys

Customer Support

To obtain support for your product, choose one of the following:

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

  Enter the information according to your environment and your issue.
- Send an e-mail message to support_center@synopsys.com
  - Include the Product name, Sub Product name, and Product version for which you want to register the problem.
- Telephone your local support center.
  - North America:

    Call 1-800-245-8005 from 7 AM to 5:30 PM Pacific time, Monday through Friday.
  - All other countries:

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

Synopsys Statement on Inclusivity and Diversity

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

1 General

This section provides a list of frequently asked general questions on VC VIP PCIe Test Suite.

1.1 How do I access the PCIe test suite documentation and examples?

The PCIe test suite documentation is available in the following directory:

$DESIGNWARE_HOME/vip/svt/pcie_test_suite_svt/<version>/doc

- User Guide

  $DESIGNWARE_HOME/vip/svt/pcie_test_suite_svt/<version>/doc/pcie_test_suite_svt_uvm_user_guide.pdf
- Release Notes

  $DESIGNWARE_HOME/vip/svt/pcie_test_suite_svt/<version>/doc/pcie_test_suite_svt_release_notes.pdf
- FAQ

  $DESIGNWARE_HOME/vip/svt/pcie_test_suite_svt/<version>/doc/pcie_test_suite_svt_faq.pdf
- Integration Guide
  - *$DESIGNWARE_HOME/vip/svt/pcie_test_suite_svt/<version>/doc/pcie_test_suite_ep_rc_integration_guide.pdf*
  - *$DESIGNWARE_HOME/vip/svt/pcie_test_suite_svt/<version>/doc/pcie_test_suite_phy_integration_guide.pdf*
- Verification Plan and Test plan can be found under:

  $DESIGNWARE_HOME/vip/svt/pcie_test_suite_svt/<version>/doc/VerficationPlans/
- Examples can be found under:

  $DESIGNWARE_HOME/vip/svt/pcie_test_suite_svt/<version>/Examples/

1.2 Is the *design_dir* part of VIP or the test suite?

The directory structure of *DESIGNWARE_HOME* and *design_dir* is as follows:

	$DESIGNWARE_HOME/vip/	***design_dir***
VIP	*vip/svt/pcie_svt/*	*design_dir/src/sverilog/vcs/svt_pcie_** Note: VIP filenames start with "*svt_pcie_*" and these files are encrypted.

Test Suite	*vip/svt/pcie_test_suite_svt/*	*design_dir/src/sverilog/vcs/svt_pcie_ts* **Note**: Test suite files will have “svt_pcie_ts” in the filename and these files are not encrypted.

You will also find some files at *$DESIGNWARE_HOME/design_dir/src/sverilog/vcs*. These are the duplicate files copied from *dw_home*.

1.3 How to run user sequence in PCIe test suite?

1. Replace the sequences as suggested.

```
/**
* This is user developed sequence that will triggered before PL Link Up.
* To Do : User need to replace sequence
* svt_pcie_device_system_virtual_user_controllable_sequence with their
* developed user sequence, if they want to hook this sequence with TS sequences.
*/
svt_pcie_device_system_virtual_user_controllable_sequence pre_ts_user_seq;
/**
* This is user developed sequence that will be triggered after PL Link Up but before DL
Link Up.
* To Do : User need to replace sequence
svt_pcie_device_system_virtual_user_controllable_sequence
* with their developed user sequence, if they want to hook this sequence with TS
sequences.
*/
svt_pcie_device_system_virtual_user_controllable_sequence post_pl_link_up_user_seq;
/**
* This is user developed sequence that will be triggered after DL Link Up but before
Enumeration gets started.
* To Do : User need to replace sequence
svt_pcie_device_system_virtual_user_controllable_sequence
* with their developed user sequence, if they want to hook this sequence with TS
sequences.
*/
svt_pcie_device_system_virtual_user_controllable_sequence post_dl_link_up_user_seq;
/**
* This is user developed sequence that will be triggered once Enumeration completed but
before the execution of
* test intention.
* To Do : User need to replace sequence
svt_pcie_device_system_virtual_user_controllable_sequence
* with their developed user sequence, if they want to hook this sequence with TS
sequences.
*/
svt_pcie_device_system_virtual_user_controllable_sequence post_enumeration_user_seq;
/**
* This is user developed sequence that will be triggered after test intention gets
completed.
* To Do : User need to replace sequence
svt_pcie_device_system_virtual_user_controllable_sequence
* with their developed user sequence, if they want to hook this sequence with TS
sequences.
*/
svt_pcie_device_system_virtual_user_controllable_sequence post_ts_user_seq;
/**
```

```
* This sequence is TS virtual base sequence and it will cast with pre_ts_user_seq
* and then shared it with TS virtual base sequence.
*/
svt_pcie_ts_device_system_virtual_base_sequence base_pre_ts_user_seq;
/**
* This sequence is TS virtual base sequence and it will cast with
post_pl_link_up_user_seq
* and then shared it with TS virtual base sequence.
*/
svt_pcie_ts_device_system_virtual_base_sequence base_post_pl_link_up_user_seq;
/**
* This sequence is TS virtual base sequence and it will cast with
post_dl_link_up_user_seq
* and then shared it with TS virtual base sequence.
*/
svt_pcie_ts_device_system_virtual_base_sequence base_post_dl_link_up_user_seq;
/**
* This sequence is TS virtual base sequence and it will cast with
post_enumeration_user_seq
* and then shared it with TS virtual base sequence.
*/
svt_pcie_ts_device_system_virtual_base_sequence base_post_enumeration_user_seq;
/**
* This sequence is TS virtual base sequence and it will cast with post_ts_user_seq
* and then shared it with TS virtual base sequence.
*/
svt_pcie_ts_device_system_virtual_base_sequence base_post_ts_user_seq;
```

2. Enable the test suite configuration variable based on your requirement. For example, if you want to run user sequence once the DL link is up, then you must set the following test suite configuration variables to 0:

   - `enable_user_sequence`
   - `enable_post_dl_link_up_user_seq`

   File: *pcie_ep_dut_cust_base_test.sv*

```
function void build_phase(uvm_phase phase);
super.build_phase(phase);
. . . . .
. . . . .
. . . . .
// If user wants to hook their sequences with TS sequences then test suite
// configuration variable enable_user_sequence need to be set.
if (cfg.enable_user_sequence) begin
// ============================================================================
// Requirement specific Configuration settings if prime configuration variable
enable_user_sequence is enabled.
// ============================================================================
cfg.enable_pre_ts_user_seq = 1'b1;
cfg.enable_post_pl_link_up_user_seq = 1'b1;
cfg.enable_post_dl_link_up_user_seq = 1'b1;
cfg.enable_post_enumeration_user_seq = 1'b1;
cfg.enable_post_ts_user_seq = 1'b1;
set_user_base_sequence();
end
. . . . .
. . . . .
. . . . .
```

```
endfunction
```

3. The `set_user_base_sequence` method is invoked during `build_phase` when the `enable_user_sequence` is set. It is used to create user-specific sequences, then cast it to virtual base sequence, and then share the information with test suite virtual base sequence.

   Usage:

   - If you want to run customized verification code on a single test case. For example, test *tl_sample_test_to_invoke_user_sequence* (it is a demo case created to validate user sequences hook up).
   - Attach the user sequences, `pre_ts_user_seq`, `post_pl_link_up_user_seq`, `post_dl_link_up_user_seq`, `post_enumeration_user_seq`, `post_ts_user_seq` using `uvm_config_db` set method from `cust_base_test` during `set_user_base_sequence` method. For details, see the following example.
   - If you want to run customized verification code on all test suite cases (Limitations describes the cases in which this hook will not work), then you need to use method to set every test sequences during `uvm_config_db` call in place of `<inst_name>`.
   - All user sequences are an extension of `svt_pcie_ts_device_system_virtual_base_sequence`.

     File: *pcie_ep_dut_cust_base_test.sv*

```
/**
* This method will be invoked if this 'enable_user_sequence' test suite
configuration variable is set.
* It will first create user specific sequences then cast it to virtual base
seqeunce and finally
* shared information with TS virtual base sequence.
*/
function void set_user_base_sequence();
begin
// Shared configuration setting information with TS virtual base sequence.
uvm_config_db#(bit)::set(this,{cfg.path_to_pcie_test_suite_env,
".test_env_seqr.svt_pcie_ts_device_system_virtual_tl_sample_test_to_invoke_use
r_sequence
*"},"enable_user_sequence",cfg.enable_user_sequence);
uvm_config_db#(bit)::set(this,{cfg.path_to_pcie_test_suite_env,
".test_env_seqr.svt_pcie_ts_device_system_virtual_tl_sample_test_to_invoke_use
r_sequence
*"},"enable_pre_ts_user_seq",cfg.enable_pre_ts_user_seq);
uvm_config_db#(bit)::set(this,{cfg.path_to_pcie_test_suite_env,
".test_env_seqr.svt_pcie_ts_device_system_virtual_tl_sample_test_to_invoke_use
r_sequence
*"},"enable_post_pl_link_up_user_seq",cfg.enable_post_pl_link_up_user_seq);
uvm_config_db#(bit)::set(this,{cfg.path_to_pcie_test_suite_env,
".test_env_seqr.svt_pcie_ts_device_system_virtual_tl_sample_test_to_invoke_use
r_sequence
*"},"enable_post_dl_link_up_user_seq",cfg.enable_post_dl_link_up_user_seq);
uvm_config_db#(bit)::set(this,{cfg.path_to_pcie_test_suite_env,
".test_env_seqr.svt_pcie_ts_device_system_virtual_tl_sample_test_to_invoke_use
r_sequence
*"},"enable_post_enumeration_user_seq",cfg.enable_post_enumeration_user_seq);
uvm_config_db#(bit)::set(this,{cfg.path_to_pcie_test_suite_env,
```

```
".test_env_seqr.svt_pcie_ts_device_system_virtual_tl_sample_test_to_invoke_use
r_sequence
*"},"enable_post_ts_user_seq",cfg.enable_post_ts_user_seq);
// Shared test default sequence name with TS virtual base sequence
uvm_config_db#(string)::set(this,{cfg.path_to_pcie_test_suite_env,
".test_env_seqr.svt_pcie_ts_device_system_virtual_tl_sample_test_to_invoke_use
r_sequence
"},"default_test_seq_name","svt_pcie_ts_device_system_virtual_tl_sample_test_t
o_invoke_u
ser_sequence");
// create method for user sequences.
pre_ts_user_seq =
svt_pcie_device_system_virtual_user_controllable_sequence::type_id::create("pr
e_ts_user_
seq");
post_pl_link_up_user_seq =
svt_pcie_device_system_virtual_user_controllable_sequence::type_id::create("po
st_pl_link
_up_user_seq");
post_dl_link_up_user_seq =
svt_pcie_device_system_virtual_user_controllable_sequence::type_id::create("po
st_dl_link
_up_user_seq");
post_enumeration_user_seq =
svt_pcie_device_system_virtual_user_controllable_sequence::type_id::create("po
st_enumera
tion_user_seq");
post_ts_user_seq =
svt_pcie_device_system_virtual_user_controllable_sequence::type_id::create("po
st_ts_user
_seq");
// Calling $cast method.
void'($cast(base_pre_ts_user_seq,pre_ts_user_seq));
void'($cast(base_post_pl_link_up_user_seq,post_pl_link_up_user_seq));
void'($cast(base_post_dl_link_up_user_seq,post_dl_link_up_user_seq));
void'($cast(base_post_enumeration_user_seq,post_enumeration_user_seq));
void'($cast(base_post_ts_user_seq,post_ts_user_seq));
// Shared information with TS virtual base sequence.
uvm_config_db#(svt_pcie_ts_device_system_virtual_base_sequence)::set(this,{cfg
.path_to_p
cie_test_suite_env,
".test_env_seqr.svt_pcie_ts_device_system_virtual_tl_sample_test_to_invoke_use
r_sequence
*"},"pre_ts_user_seq",base_pre_ts_user_seq);
uvm_config_db#(svt_pcie_ts_device_system_virtual_base_sequence)::set(this,{cfg
.path_to_p
cie_test_suite_env,
".test_env_seqr.svt_pcie_ts_device_system_virtual_tl_sample_test_to_invoke_use
r_sequence
*"},"post_pl_link_up_user_seq",base_post_pl_link_up_user_seq);
uvm_config_db#(svt_pcie_ts_device_system_virtual_base_sequence)::set(this,{cfg
.path_to_p
cie_test_suite_env,
".test_env_seqr.svt_pcie_ts_device_system_virtual_tl_sample_test_to_invoke_use
r_sequence
*"},"post_dl_link_up_user_seq",base_post_dl_link_up_user_seq);
uvm_config_db#(svt_pcie_ts_device_system_virtual_base_sequence)::set(this,{cfg
.path_to_p
cie_test_suite_env,
```

```
".test_env_seqr.svt_pcie_ts_device_system_virtual_tl_sample_test_to_invoke_use
r_sequence
*"},"post_enumeration_user_seq",base_post_enumeration_user_seq);
uvm_config_db#(svt_pcie_ts_device_system_virtual_base_sequence)::set(this,{cfg
.path_to_p
cie_test_suite_env,
".test_env_seqr.svt_pcie_ts_device_system_virtual_tl_sample_test_to_invoke_use
r_sequence
*"},"post_ts_user_seq",base_post_ts_user_seq);
end
endfunction:set_user_base_sequence
```

Limitations

- Hook up does not work for test suite cases that are neither using PL link up sequence (`svt_pcie_device_system_virtual_enable_link_up_sequence`) nor activate link sequence (`svt_pcie_device_system_virtual_activate_link_sequence`).

1.4 How do I transaction/symbol logging in the test suite?

Use the following command to enable transaction logging and symbol logging:

```
SVT_PCIE_ENABLE_TRANSACTION_LOGGING=1
```

1.5 How to control the test cases timeout value?

The test execution is controlled by a global timeout attribute and is terminated when the timer expires. The default value of test timeout is 1200000ns. For a specific test run, the default value can be overridden by the command-line argument:

```
SVT_PCIE_TEST_TIMEOUT= timeout-value-in-ns.
```

1.6 How to override an already existing test suite sequence class with an user-defined sequence?

Perform the following steps to override an existing test suite sequence class with an user-defined sequence:

1. Create a file (which you have to maintain it locally—that is, *cust_override_seq_file.sv*) and make all the extended class declaration in this file.

```
class cust_svt_pcie_ts_device_system_virtual_tl_transaction_ordering_rule_a2a_sequence
extends svt_pcie_ts_device_system_virtual_tl_transaction_ordering_rule_a2a_sequence
…
…
endclass
```

2. Include this file (*cust_override_seq_file.sv*) in the *cust_pcie_test_suite_pkg_files.svi* file and ensure to save a local copy of this file.

```
/**
 * Abstract:
   Hook to include customer classes as part of package snps_pcie_test_suite
 */
  `ifndef GAURD_CUST_PCIE_TEST_SUITE_PKG_FILES_SVI
  `define GAURD_CUST_PCIE_TEST_SUITE_PKG_FILES_SVI

  `include "cust_override_seq_file.sv"

   `endif
```

3. Add run simulation after adding the following plusarg:

```
'+uvm_set_type_override=
svt_pcie_ts_device_system_virtual_tl_transaction_ordering_rule_a2a_sequence,cust_svt_pci
e_ts_device_system_virtual_tl_transaction_ordering_rule_a2a_sequence
```

1.7 What is the difference between the following directories *design_dir/src/sverilog/vcs* *dw_home/vip/svt/pcie_svt/<version>/sverilog/src/vcs*

The files inside both the directories are same except the extension name (*.sv* and *.svp*). As per use model of the PCIe test suite, VIP files and example files will reside in different directories.

- Install PCIe VIP test suite product using the "*.run*" file.
- After installing the test suite product, you will find *$DESIGNWARE_HOME/vip/* directories created parallel to "*.run*" file.
- Install one of the examples (for example, `tb_dut_ep_gen3_serdes`).
- After installing the example, *design_dir* directory will get created which will copy the required files for `tb_dut_ep_gen3_serdes` example into *design_dir*.
- All the required scripts for running the test cases are available inside the *design_dir* path. Hence, same files are present in *design_dir* (installed example) and *DESIGNWARE_HOME* (installed VIP).

1.8 How to control each port (link) of the Unified VIP configuration differently?

Refer to class member `pcie_unified_bas_test::all_links_cfg`

The following class serves as an anchor for all instances of `` `SVT_PCIE_TEST_SUITE_CONFIGURATION_TYPE ``. This anchor instance allows you to use a single text file to hold config properties for all the links. The primary test instance populates the elements of `all_links_cfg.cfg[]` during the execution of `extract_module_topology`. All test instances pick the handle to their respective configurations when they invoke the method `create_cfg` (via legacy handle configuration). Hence, the anchor class is defined as static.

```
static svt_pcie_test_suite_unified_configuration  all_links_cfg;
```

Example Code

```
all_links_cfg.cfg[0]   represents test suite configuration for link 0
all_links_cfg.cfg[1]   represents test suite configuration for link 1
```

If you have a handle configuration and want to know which link it corresponds to, then `cfg.ep_cfg.unified_if.link_id` or `cfg.rc_cfg.unified_if.link_id` can give the details about the port it corresponds to. For more details and example code, see "Multi-Link Use Model" in the *VC Verification IP PCIe Test Suite UVM User Guide*.

1.9 How do I resolve the following shadow_memory_checking warning?

```
UVM_WARNING :: [check_for_hdl_parameter_consistency] PARAMETER VALUE
test_top.spd_0.SVT_PCIE_UI_ENABLE_SHADOW_MEMORY_CHECKING does not align with
requester_cfg.enable_shadow_memory_checking or
driver_cfg[0].enable_shadow_memory_checking
```

The default value of the parameter in Unified env is 0. But it has been reconfigured from the test case. Therefore, you need to set the default value to 1 from your topology file and run the test case.

```
defparam  SVT_PCIE_TEST_SUITE_ENABLE_SHADOW_MEMORY_CHECKING_P``port_num`` = 1;
```

2 Configuration

2.1 How to configure test suite test cases in 16-bit PIPE mode?

Set the SVT_PCIE_PIPE_WIDTH configuration to 1 for 16-bit as a command-line option.

To enable different PIPE width configuration:

Value	Bit
0	8
1	16
2	32

2.2 How to configure VIP test suite as x1 lane?

Set the following command-line option to control the target link width:

```
SVT_PCIE_TARGET_LINK_WIDTH
```

It can be passed with values 1, 2, 4, 8, 12, 16 and 32.

The link width for RC and EP will be configured according to the passed value. But the passing value should be equal to or less than the value of SVT_PCIE_MAX_LINK_WIDTH.

For more details, “Changing the Link Width” in the *VC Verification IP PCIe Test Suite UVM User Guide*.

2.3 How to configure VIP test suite for PCIe Gen1 or Gen2?

To configure the test suite for Gen1 or Gen2 data rate, set the following configuration:

```
SVT_PCIE_LINK_SPEED=GEN2
SVT_PCIE_LINK_SPEED=GEN1
```

All the data rates will be controlled using the following options:

- GEN1
- GEN2
- GEN3
- GEN4

2.4 How to control SRIS/SRNS mode in the test suite?

- SRIS

  Use the following command to enable SRIS:

```
gmake <test_name> SVT_PCIE_SSC_MODE=SSC_ON_PPM_ON SVT_PCIE_MAX_PPM=<+/-max osc ppm
value> SVT_PCIE_MIN_PPM=< +/- min osc ppm value >
```

The command-line option `SVT_PCIE_SSC_MODE=SSC_ON_PPM_ON` will introduce additional 5000PPM in clock as SSC.

- SRNS

  Use the following command to enable SRNS (Tx, Rx Refclk rates differ, allowing up to 600 PPM):

```
gmake <test_name> SVT_PCIE_SSC_MODE=SSC_OFF_PPM_ON SVT_PCIE_MAX_PPM=<+/- max osc ppm
value> SVT_PCIE_MIN_PPM=< +/- min osc ppm value >
```

  The `SSC_OFF_PPM_ON` parameter enables SRNS mode—that is, there will not be any spread spectrum clocking.

2.5 How to enable fast simulation mode in the test suite?

To facilitate faster simulation of tests, the Test Suite provides a mechanism to set up scaled down values of specification timers and the number of ordered sets communicated during the link initialization phase. This assumes that the DUT also supports the fast simulation mode and the timers can be configured accordingly. Also, the values for VIP configuration attributes must be same as the DUT values.

To configure all LTSSM timers based on the scale factor in the DUT, set the value of the `SVT_PCIE_TEST_SUITE_FAST_SIM_1MS` macro in ns (default value is 1024). All the timer values are calculated internally (for example, `` `define SVT_PCIE_TEST_SUITE_FAST_SIM_12MS 12 *`SVT_PCIE_TEST_SUITE_FAST_SIM_1MS) ``. Based on the macros defined for different time-outs, the LTSSM timer attributes in the `svt_pcie_test_suite_configuration` class of VIP are set up.

2.6 How to disconnect the VIP after the default sequence has been completed?

In multi-link scenarios, when test intention of one link is complete and the test is still waiting for the other link to complete, the simulation may hang. To avoid this, you can set the test suite configuration, `disconnect_active_vip_after_default_sequence` to 1 so that the VIP gets disconnected after the default sequence has been completed.

```
int disconnect_active_vip_after_default_sequence = 0;
```

You can set this configuration parameter in the test build phase for all VIP instances.

```
cfg.disconnect_active_vip_after_default_sequence =1
```

3 Scoreboard

3.1 How to handle the test cases that hang?

When using the test suite and scoreboard is enabled, sometimes simulation results in hang condition.

The reason for the test case hang can be undropped objections. Use the following options to check undropped objections:

- If you are using VCS simulator, enable `debug_all` compilation option and then view the undropped objected in `UVM DEBUG` icon.
- Enable `+UVM_OBJECTION_TRACE` at runtime.

3.2 How scoreboarding is done in test suite?

PCIe Test Suite Scoreboard has a uniform scoreboarding architecture for [VIP - VIP/VIP - DUT] modes, which compares:

- Inbound TLP (received by DUT) with the transmitted TLP VIP.
- Outbound TLP (transmitted by DUT) with the received TLP VIP.

The Test Suite environment provides analysis ports at an external application BFM and DL layer. The VIP provides the write and response data.

- Address based transactions are stored into associative arrays with addresses as index.
- ID based transactions are stored into associative arrays with Requester ID as index.
- Run phase, `uvm_post_shutdown_phase` waits for associative arrays to be empty.

3.3 How to change the severity of miscompare messages coming from scoreboard?

The severity of MISCMP messages coming from UVM `compare()` method by default is set to `UVM_INFO`. To change the severity to `UVM_WARNING`, set the `enable_sb_miscmp_severity_warning` test suite configuration attribute to 1.

```
cfg.enable_sb_miscmp_severity_warning=1;
```

4 LTSSM Timeout

4.1 How do I resolve the following Loopback entry/exit error?

```
UVM_ERROR /src/sverilog/vcs/pciesvc_ltssm.sv(1482) @ 271490 ns:
uvm_test_top.env.rc_env.rc_agent.port0.pl0[register_fail:ACTIVE_PL:LOOPBACK_ENTRY:phy_lo
opback_entry_no_compliance_timeout]LTSSMStateMachine: Timeout in state LOOPBACK_ENTRY as
no TS1 with loopback enable bit set was detected. VIP will now transition to
Loopback.Exit.
```

As per the specification: If the data rate was changed, the Master must take into account the amount of time the Slave can be in electrical idle and transmit a sufficient number of TS1 Ordered Sets for the Slave to acquire Symbol lock or Block alignment before proceeding to `Loopback.Active`.

VIP can wait until the Slave can be in electrical idle and transmit sufficient OS to acquire symbol lock. Increase `loopback_master_no_enter_compliance_tx_ts1_timeout_ns` to a large value so that DUT can transmit TS1 OS with Loopback enable bit set.

```
rand int unsigned  attribute
svt_pcie_pl_configuration::loopback_master_no_enter_compliance_tx_ts1_timeout_ns =
SVT_PCIE_LOOPBACK_MASTER_NO_ENTER_COMPLIANCE_TX_TS1_TIMEOUT_NS_DEFAULT
```

Specifies the time in ns before the loopback master enters detect if no TS1s are received with the enter compliance bit clear. If randomized, the variable will resolve to a value within the range specified by the constraint.

4.2 gen3_pl_recovery_rcvrcfg_to_detect* test cases fail to move from recovery_rcvrcfg to Detect state.

According to test case requirement, when LTSSM in Gen3 reaches `recovery_rcvrcfg`, it has to move to Detect state after 48ms timeout. But this will happen only when `ts_idle_to_rlock_transitioned` reaches a value of FFh.

Currently, LTSSM in Gen3 reaches `recovery_rcvrcfg` to `recovery_idle`, as `idle_to_rlock_transitioned` variable is less than FFh and the current data rate is 8.0 GT/s or higher.

The default value of `ts_idle_to_rlock_transitioned` is set to '0' as SNPS IIP considers all 255 state transition iterations to happen. But for DUTs considering less state transitions, to set the variable accordingly from the base test.

For example, if DUT in simulation mode fix the number of iterations to 7 instead of 255, set the `set_ts_idle_to_rlock_transitioned_variable` to 248 from base test as shown below:

```
uvm_config_db#(int)::set(this, {cfg.path_to_pcie_test_suite_env,".test_env_seqr"},
"set_ts_idle_to_rlock_transitioned_variable", 248);
```

4.3 How to increase the LTSSM timeout values?

You may encounter LTSSM timeout errors while running the test cases. This is due to the DUT taking longer at that LTSSM state than the threshold timeout value.

You can increase the timeout values of respective LTSSM states by applying the following configurations.

Example:

```
<vip_cfg>.pcie_cfg.pl_cfg.detect_quiet_timeout_ns=<value>
```

Other available VIP's LTSSM timeout values can be configured in the same way. Following is the list of LTSSM timeout values that can be configured.

Table 4-1 LTSSM Timeout Values

LTSSM State	Default Timeout Values
disabled_timeout_ns	20000
detect_quiet_timeout_ns	12000
detect_active_timeout_ns	24000
polling_active_timeout_ns	240000
polling_configuration_timeout_ns	480000
configuration_linkwidth_start_timeout_ns	240000
configuration_linkwidth_accept_timeout_ns	240000
configuration_lanenum_wait_timeout_ns	240000
min_configuration_lanenum_accept_timeout_ns	20000
max_configuration_lanenum_accept_timeout_ns	20000
configuration_idle_timeout_ns	20000
configuration_complete_timeout_ns	20000
recovery_idle_timeout_ns	20000
recovery_rcvrcfg_timeout_ns	240000
recovery_rcvrlock_timeout_ns	240000
downstream_lanes_recovery_eq_phase1_timeout_ns	24000
downstream_lanes_recovery_eq_phase2_timeout_ns	32000
downstream_lanes_recovery_eq_phase3_timeout_ns	24000
upstream_lanes_recovery_eq_phase0_timeout_ns	12000
upstream_lanes_recovery_eq_phase1_timeout_ns	12000
upstream_lanes_recovery_eq_phase2_timeout_ns	24000
upstream_lanes_recovery_eq_phase3_timeout_ns	32000

Table 4-1 LTSSM Timeout Values

LTSSM State	Default Timeout Values
`hot_reset_timeout_ns`	2000

5 Usage Issues

5.1 Transaction Layer

5.1.1 Why does the test tl_application_error_reporting fail?

The `tl_application_error_reporting` test is specific for DUT as SNPS IIP only.

Following are the list of SNPS IIP specific cases:

- EP DUT
  - *ts.tl_application_error_reporting.sv*
- RC DUT
  - *ts.tl_application_error_reporting.sv*
  - *ts.tl_root_error_message_controls_with_system_notification.sv*
  - *ts.tl_root_native_pme_software_interrupt_enable.sv*
  - *ts.tl_root_tx_obff_msg.sv*

For more details, see "Application Error Reporting Interface Test (Only for SNPS IIP)" in the *VC Verification IP PCIe Test Suite UVM User Guide*.

5.1.2 How do I resolve the following error?

```
UVM_ERROR /
sverilog/src/vcs/svt_pcie_ts_device_system_virtual_sequence_collection.svp(4257)
@ 285650 ns: uvm_test_top.env.test_env_seqr@@sv
t_pcie_ts_device_system_virtual_tl_aer_unexpected_cpl_error_sequence
[check_aer_status_bits] Value found in Uncorrectable Error Status Register at bdf
= 0x100 different from expected. Observed = 0x11000, Expected = 0x10000
```

As per standard PCIe 4.0 ver 1.0 specification, when multiple errors are detected in a single TLP, only one error should be logged and reported as per precedence. Therefore, the status register should not log more than one error.

Reference

- *6.2.3.2.3 Error Pollution (as per spec rev 4.0 ver 1.0)*

  For errors detected in the Transaction layer and Uncorrectable Internal Errors, it is permitted and recommended that no more than one error be reported for a single received TLP.
- *6.2.4.2 Multiple Error Handling (Advanced Error Reporting Capability)(as per spec rev 4.0 ver 1.0)*

If multiple header recording is supported and enabled, and the First Error Pointer is valid, it is recommended that software not write a 1b to any status bit other than the one indicated by the First Error Pointer.

5.1.3 How to set the min_range/max_range for the address?

TLP from EP VIP to RC DUT should ideally be 64-bit and not 48-bit because traffic is generated as per the PCIe specification. Here, DUT should translate to 48-bit internally.

Set the type override mechanism to program RC configuration sequence. Populate test suite TL status for both EP VIP and RC application side of the testbench as shown below:

Class to override: `svt_pcie_ts_cfg_database_service_program_rc_cfg_space_sequence`

- For Transactions from EP VIP to RC DUT: Constraint or make upper 16-bit of `rc_mem_range_min` and `rc_mem_range_max` accordingly.

```
`uvm_debug("svt_pcie_ts_cfg_database_service_program_rc_cfg_space_sequence",
$sformatf("Programmed RC BAR in finction number %0d with 0x%0x address. Found
aperture=0x%0x.",function_number, rc_bar_addr, rc_mem_bar_aperture))
test_suite_tl_status.rc_mem_range_min [num_rc_bar_mem_ranges]= rc_bar_addr;
test_suite_tl_status.rc_mem_range_max[num_rc_bar_mem_ranges++]=(rc_bar_addr+rc_mem_bar_a
pperture);
end
```

- For Transactions from RC DUT to EP VIP

  You need to modify/constraint `mem_range_min/ mem_range_max`.

  The above solution provided to set the `min_range/max_range` for the address if DUT supports BAR. If DUT does not support BAR, the `min_range/max_range` setting for the address is as follows:

```
test_suite_tl_status.rc_mem_range_min[0] = 64`h0;
test_suite_tl_status.rc_mem_range_mix[0] = 64`hFFFF_FFFF;
test_suite_tl_status.rc_mem_range_min[0] = 64`h0000_0000;
test_suite_tl_status.rc_mem_range_min[0] = 64`hFFFF_FFFF_FFFF;

test_suite_tl_status.rc_mem_range_max[num_rc_bar_mem_ranges++] = (rc_bar_addr +
rc_mem_bar_aperture);
if(num_rc_bar_mem_ranges == 0) begin
//test_suite_tl_status.rc_mem_range_min.delete();
test_suite_tl_status.rc_mem_range_max.delete();
svt_pcie_ts_address_generator::ep_mem_req_target_rc_bar_wt = 0;
`svt_xvm_debug("svt_pcie_ts_cfg_database_service_program_rc_cfg_space_sequence", "Found
no BAR supported")
```

Note Set the values as per the requirement. Set the address range outside the loop so that even if BAR is not present, the test suite finds the address range.

5.2 Link Layer

5.2.1 Why does the test case dl_tlp_replay_roll_over_sequences_number fail with the following error when DUT transmits packet correctly?

```
UVM_ERROR @ 517422
ns:@svt_pcie_ts_device_system_dl_tlp_replay_roll_over_sequences_number_sequence
[svt_pcie_ts_device_system_dl_tlp_replay_roll_over_sequences_number_sequence]
VIP did not receive all MWr Req which are sent from DUT within expected timeframe 210000
ns, Number of MWr Received is 00000882d and Expected is 00001146d
```

This error may occur due to the expiry of timer before the sequence execution is completed. Setting a larger value of `total_timeout_timer` will resolve the issue.

Set a larger the value of `total_timeout_timer` from your cust base test as shown below:

```
uvm_config_db#(int)::set(this, {cfg.path_to_pcie_test_suite_env,".test_env_seqr"},
"total_timeout_timer", 1150000); //this is the default value , you need to assign larger
value to resolve the error.
```

5.2.2 Are there any test cases for credit counter rollover scenario testing of DUT?

Following are the test cases for sequence number rollover and replay number rollover.

- *ts.dl_tlp_seq_number_rollover-ep.sv*
- *ts.dl_aer_replay_num_rollover_error-ep.sv*
- *ts.dl_tlp_replay_roll_over_sequences_number-ep.sv*

Test case for the said scenario of credit rollover is not available. However, you use any memory test case to send packets till credit rollover happens.

The test case needs to send enough TLPs so that DUT's 12-bit credit counter will rollover. The `tl_tx_mem_mapped` test case can be modified to achieve the above scenario or any other memory test case. Ensure to send packets till rollover happens.

5.3 Physical Layer

5.3.1 How to enable Gen4 EIEOS pattern?

To enable Gen4 EIEOS, define the macro `SVT_PCIE_TEST_SUITE_ENABLE_EIEOS_16G_0000_FFFF` with a value of 1 in compile option. This will invoke VIP to transmit and expect for new 16G EIEOS pattern. By default, it is set to 0.

5.3.2 How do I increase the value of ppm to 5600 from SRIS mode?

To enable SRIS along with PPM value set, use the following command:

```
SVT_PCIE_SSC_MODE=SSC_ON_PPM_ON
SVT_PCIE_MAX_PPM=<+/- max osc ppm
value> SVT_PCIE_MIN_PPM=< +/- min osc ppm value >
```

For more details, see “SRIS Usage” in the *VC Verification IP PCIe Test Suite UVM User Guide*.

5.3.3 What is the allowed Jitter/ expected range of allowed `bit_clk` for Gen1? Why is the bit_clk of RC out of specification?

You can modify the following code in *pcie_phy_dut_base_test.sv* file to get the `bit_clk` in-between the expected range as per the specification (Table 4-18: Transmitter Specifications of PCIe 3.1 spec.), the UI for 2.5 GT/s should be within 399.88 ps (min)/400.12 (maximum) range.

Table 5-1 Required Modification

Existing Code	Modified Code
*** Attribute to pick random ppm value from the min and max value provided **/* `rand int choose_ppm_within_min_max = 600;` *// Initializing default value with 600* *// Default value is set to 600PPM* `int max_xo_ppm = 600;` `int min_xo_ppm = 600;`	*** Attribute to pick random ppm value from the min and max value provided **/* `rand int choose_ppm_within_min_max = 100;` *// Initializing default value with 100* *// Default value is set to 100PPM* `int max_xo_ppm = 100;` `int min_xo_ppm = 100;`

5.3.4 How to avoid framing_error related failures in configuration.idle state when optional checks are supported in DUT?

As per specification, when `framing_error` is injected in `configuration.idle` state, the following transitions are supported:

- `configuration.idle->L0->recovery transition`
- `configuration.idle->recovery transition`

These two transitions can be differentiated using the following configuration attribute:

```
pcie_cfg.dut_capabilities.enable_framing_error_in_config_idle_and_recovery_idle_check == 1
```

5.3.5 How do I resolve the following error?

```
/root.port0.pl[register_fail:ACTIVE_PL_PIPE:PIPE_chk:pipe_txdeemph_3_5db_check]
Description: MAC must ensure that TxDeemph is selecting -3.5db whenever Rate is
selecting 2.5 GT/s, Reference: PIPE v4.3: 7.8 Selectable De-emphasis - PCI
Express Mode - Active VIP detected that MAC has driven TxDeemph signal for lane 0
to 0. This is not a legal value as per PIPE Specifications.
```

The `pipe_txdeemph_3_5db_check` check verifies that MAC must ensure that `TxDeemph` is selecting -3.5db whenever Rate is 2.5 GT/s.

Reference

- PIPE specification
  - 7.8 Selectable De-emphasis - PCI Express Mode

    The MAC must ensure that TxDeemph is selecting -3.5db whenever Rate is selecting 2.5 GT/s.

A PL configuration variable has been added for disabling this check when `tx_deemphasis` changes prior to a speed change from 2.5G.

```
cfg.rc_cfg.pcie_cfg.pl_cfg.enable_pipe_txdeemph_3_5db_check_during_tx_elec_idle = 1'b0;
```

When set to a 1, VIP will check that tx deemphasis is always set to -3.5db at 2.5GT/s, even when `tx_elec_idle` is asserted.

When set to a 0, the VIP will only check `tx_deemphasis` when `tx_elec_idle` is 0 and valid data is being transmitted. Devices which change `tx_deemphasis` prior to a speed change from 2.5G should leave this set to 0.

6 Error Injection

6.1 How to disable a specific error injection settings which are not supported by DUT for AER (ECRC) while running the test tl_application_error_reporting?

Set the following configuration attributes in case particular error type is not supported by the Application.

Configure the `ts_app_err_bus` variable with following values from cust base test.

Table 6-1 Error Types with Corresponding Values

Error Types Not Supported by Application	Corresponding Value for Variable ts_app_err_bus
Malformed TLP Status	13'h1ffe
Receiver Overflow	13'h1ffd
Unexpected Completion	13'h1ffb
Completion Abort	13'h1ff7
Completion Timeout	13'h1fef
Unsupported Request	13'h1fdf
ECRC Check Failed	13'h1fbf
Poisoned TLP	13'h1f7b
AtomicOp Egress Blocked	13'h1efb

The following example shows how to disable ECRC error:

```
cfg.ts_app_err_bus = 13'h1fbf   //ECRC check failed
```

7 Coverage

7.1 How to enable the coverage using the test suite command line options?

To enable the coverage, pass the `SVT_PCIE_ENABLE_COVERAGE` command line option:

For example, `SVT_PCIE_ENABLE_COVERAGE = 5'b00000`

This is a five-bit variable and each layer corresponds to a unique layer.

- [4] bit when set to 1'b1, enables coverage for LTSSM checks.
- [3] bit when set to 1'b1, enables coverage in TL Layer.
- [2] bit when set to 1'b1, enables coverage in DL Layer.
- [1] bit when set to 1'b1, enables coverage in PL Layer.
- [0] bit when set to 1'b1, enables coverage for PIPE.