Verification Continuum™

VC Verification IP
PCIe
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 PCIe.

Web Resources

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

Customer Support

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

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

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

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

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

Synopsys Statement on Inclusivity and Diversity

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

1 General

1.1 General

This section provides a list of frequently asked general questions for the VC VIP for PCIe.

1. How do I access the VIP documentation and examples?

   All PDF documents are available on SolvNetPlus (click here).

   PCIe VIP documentation directory:

   `$DESIGNWARE_HOME/vip/svt/pcie_svt/<version>/doc`

   VC VIP for PCIe UVM User Guide (PDF):

   `$DESIGNWARE_HOME/vip/svt/pcie_svt/<version>/doc/pcie_svt_uvm_user_guide.pdf`

   VC VIP for PCIe UVM Getting Started Guide (PDF):

   `$DESIGNWARE_HOME/vip/svt/pcie_svt/<version>/doc/pcie_svt_uvm_getting_started.pdf`

   Note Provides step-by-step instructions for integrating the VIP into a UVM environment.

   VC VIP for PCIe Class Reference (HTML):

   `$DESIGNWARE_HOME/vip/svt/pcie_svt/<version>/doc/pcie_svt_uvm_class_reference/html/index.html`

   Note Documents all classes provided by the VIP and their attributes.

   VC VIP for PCIe Verification Plans (XML):

   `$DESIGNWARE_HOME/vip/svt/pcie_svt/<version>/doc/VerificationPlans`

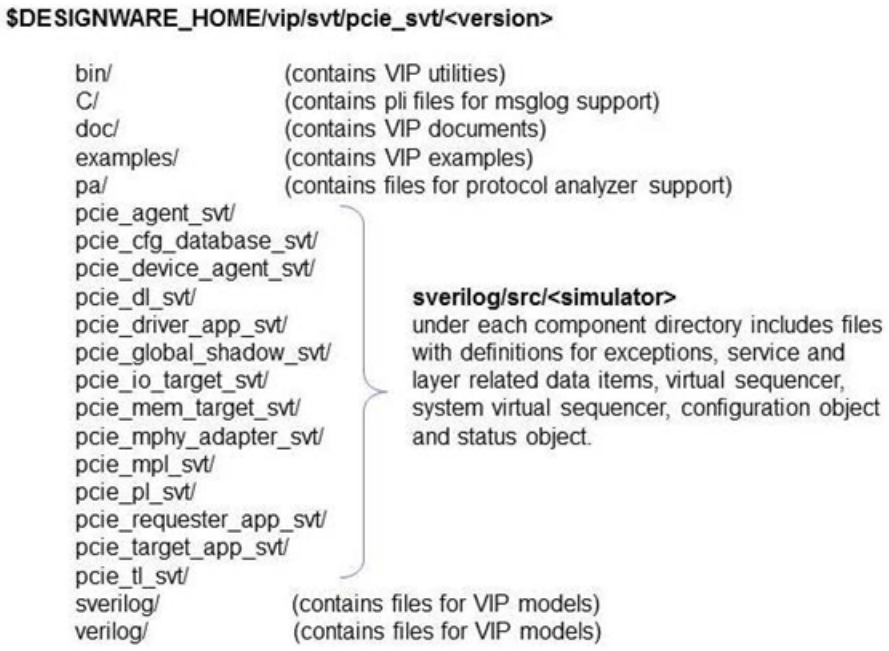
   Directory that contains a list of PCIe VIP example directories:

   `$DESIGNWARE_HOME/vip/svt/pcie_svt/<version>/examples/sverilog`

2. What is the directory structure of the VIP?

   The directory structure of the VIP is as follows:

Figure 1-1 VIP Directory Structure



3. What are the primary classes of the VIP?

   The primary classes of the VIP are listed in the following table.

Table 1-1 VIP Primary Class

Primary Classes	Transaction Layer (TL)	Data Link Layer (DLL)	PL
Configuration class	`svt_pcie_tl_configuration`	`svt_pcie_dl_configuration`	`svt_pcie_pl_configuration`
Service class	`svt_pcie_tl_service`	`svt_pcie_dl_service`	`svt_pcie_pl_service`
Transaction class	`svt_pcie_tlp`	`svt_pcie_dllp`	`svt_pcie_phy_transaction`
Exception class	`svt_pcie_tlp_exception`	`svt_pcie_dllp_exception`	`(symbol exception) svt_pcie_symbol_exception`

- Configuration class

  The configuration class is derived from `svt_configuration` and has layer specific configuration parameters.

For example,

TL -> `remote_max_payload_size` (`Max_Payload_Size` parameter of the DUT)

DL -> `max_acknak_lateny` (Acknak Latency timer of the VIP)

- Service class

  All layer specific services are defined in the service class.

  For example:

  TL -> Service to map TC to VC(Service to enable VC)

- Transaction class

  All layer specific data items and PCIe protocol specific constraints to generate valid Transaction Layer Packets (TLPs) , Data Link Layer Packets (DLLPs) and Ordered Sets are defined in the transaction class.

  For example:

  TL -> Items to generate MRd TLP and valid constraints for the transaction layer

- Exception class

  All layer specific error injection parameters are defined in the exception class.

  For example,

  TL -> `CORRUPT_FMT` (Corrupt the FMT field of transmitted TLPs)

1. How do I specify the configurations of the VIP?

   Configurations are defined for each layer of the PCIe protocol. These configurations and the VIP specific configuration parameters can be customized by extending the configuration classes and specifying the required parameters in the extended classes.

   For example,

   The `remote_max_payload_size` parameter in the TL configurations sets the `max_payload_size` of the DUT.

   The `remote_extended_tag_enable` parameter in the TL configurations enables and disables extended tag on a DUT.

   The `max_payload_size` parameter in the DL configurations sets the `max_payload_size` of the VIP. The `max_payload_size` can be used to calculate the durations of `acknak_latency` and `replay_timer`.

   The `link_width` parameter in the PL configurations sets the maximum negotiated `link_width` between link partners.

   The following are the layer specific configuration classes

Table 1-2 Layer Specific Configuration Classes

Layer	Configuration Classes
TL	`svt_pcie_tl_configuration`
DL	`svt_pcie_dl_configuration`
PL	`svt_pcie_pl_configuration`

2. What is the usage of layer specific service classes?

   Layer specific service classes control the VIP operations. These classes are used to initiate certain modes of operations.

   For example,

   The TL has services to enable particular VC's and map TC's to VC.

   The DL has service to enable DL Link up that allows the VIP to transmit INITFC's and initialize default VC 0.

   The PL has service to initiate the speed change in L0 LTSSM state.

   The following are classes with layer specific service definitions:

Table 1-3 Layer Specific Service Definitions

Layer	Service Definitions
TL	`svt_pcie_tl_service`
DL	`svt_pcie_dl_service`
PL	`svt_pcie_pl_service`

3. What is the usage of status classes?

   Layer specific status classes provides the protocol specific parameter status of each layer.

   For example,

   TL -> `num_tlp_cpld_received` (Continuous status of the number of CplD received by the VIP)

   DL-> `num_bad_tlp_received` (Continuous status of the number of bad TLPs received by the VIP)

   PL -> `num_rx_skp_ordered_sets` (Numbers of SKIP order sets received on the configured lanes)

   The following are the layer specific status classes:

Table 1-4 Layer Specific Status CLasses

Layer	Status Classes
TL	svt_pcie_tl_status
DL	svt_pcie_dl_status
PL	svt_pcie_pl_status

4. What are the types of verbosity for messages issued by the VIP?

   Use the `+uvm_set_verbosity` option in the command line to specify the verbosity of the VIP messages.

   The following is a mapping of the verbosity types between VIP and UVM.

Table 1-5 Verbosity Types

VIP	UVM
`svt_fatal	UVM_FATAL
`svt_error	UVM_ERROR
`svt_warning	UVM_WARNING
`svt_note	UVM_INFO
`svt_trace	UVM_INFO
`svt_verbose	UVM_INFO

For more information, see the SolvNetPlus article at

https://solvnet.synopsys.com/retrieve/2357749.html

5. What are the VIP log files generated in a simulation?

   In the current directory where a test is run, a transaction log (`xact_log`) file and a symbol log (`symb_log`) file are generated. Under the output sub-directory, a transcript log file is generated.

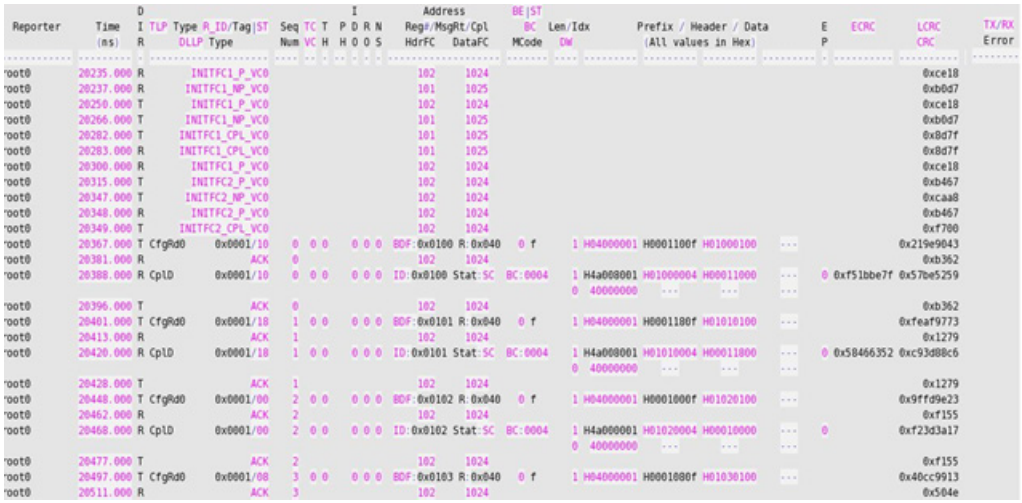
   The transcript log file contains messages generated from the VIP and DUT with the specified verbosity. The transaction log (xact_log) file contains information about transmitted and received TLPs and DLLPs. The symbol log (symb_log) file contains information about the transmitted Ordered Sets and the LTSSM state transitions.

   ```
   <simulation_directory>/output/*transcript
   <simulation_directory>/*xact_log
   <simulation_directory>/*sym_log
   ```

Note The `enable_transaction_logging` attribute in the PCIe configuration class must be set to enable the generation of transaction log file. The `enable_symbol_logging` attributes in the PL configuration class must be set to enable the generation of symbol log file.

Transaction log file with TLPs generated by the VIP is as follows.

Figure 1-2 Transaction Log File With TLPs Generated



For more information, see the SolvNetPlus article at

https://solvnet.synopsys.com/retrieve/2058441.html

6. How do I specify the timeout value of the VIP?

   There are two methods to specify the timeout value of the VIP.

   a. Add a simple plusarg in the following testbench:

      ```
      CMDLINE_VCSPLUSARGS='+PCIE_TO_MS=<integer or real value>'
      ```

      The specified value is the new top-level UVM timeout in ms.

   b. Call the `set_timeout` method in the build phase of a test.

      ```
      uvm_top.set_timeout (<time value including units>);
      ```

Note The default timeout value is 3ms based on a timescale of 1ns.

7. What is the difference between the TLP transaction class and the driver application transaction class?

   A majority of TL transactions can be generated by the driver application transaction class (`svt_pcie_driver_app_transaction`). These transactions are sufficient for verifying DL specific scenarios. However, the `svt_pcie_driver_app_transaction` class does not provide control for the attributes and `requester_id` of TL packets.

   The TLP transaction class (`svt_pcie_tlp`) provides control for fields in the TL packet header such as the `requester_id`, `attr_id_order`, `attr_relaxed_ordering` and tags. If control of the TL packet header is required, the `svt_pcie_tlp` class should be used to generate TL packets. Otherwise, the `svt_pcie_driver_app_transaction` class can be used.

   For more information, see the following SolvNetPlus articles:

   https://solvnet.synopsys.com/retrieve/1623967.html

   https://solvnet.synopsys.com/retrieve/1623926.html

   https://solvnet.synopsys.com/retrieve/1598307.html

https://solvnet.synopsys.com/retrieve/2327932.html

[Back to top] [Go to General] [Go to TL] [Go to DL] [Go to PL]

8. How do I set specific behaviors as defined in the PCIe specification?

   Each PCIe layer has its own configuration class. The configuration members in each of these classes can be used to set specific behaviors as defined in the PCIe specification. You can set the value of individual configuration attributes by extending a configuration class and specifying the required configuration values in the extended class.

   For example:

   ```
   // Configuration object used by the Root Complex PCIE Device objects.
    rand svt_pcie_device_configuration root_cfg;
   // Configuration object used by the Endpoint PCIE Device objects.
   rand svt_pcie_device_configuration endpoint_cfg;
   // Define whether devices are root or endpoint.
   root_cfg.device_is_root == 1'b1;
   endpoint_cfg.device_is_root == 1'b0;
   ```

   [Back to top] [Go to General] [Go to TL] [Go to DLL] [Go to PL

9. What is a potential cause of the message `Simulation error: tx_scrambler0: 8G is not supported` ?

   The message is issued when 8G support is not enabled. You can enable 8G support by defining the `SVT_PCIE_ENABLE_GEN3` parameter in the top-level test file or on the command line.

   Top-level test file:

   ```
   // Enable Gen3
   `define   SVT_PCIE_ENABLE_GEN3
   ```

   Command line:

   ```
   +define+SVT_PCIE_ENABLE_GEN3=1
   ```

10. How do I demote error messages from `UVM_ERROR` to `UVM_INFO` when errors are injected by the VIP?

    Use the error catcher class (`svt_err_catcher`) to change the severity of error messages by comparing the error message string.

    For example:

    ```
    svt_err_catcher err_catcher;
    virtual function void build_phase(uvm_phase phase);
    `uvm_info("build_phase", "Enter...", UVM_HIGH)
    super.build_phase(phase);

    //Create report catcher object
    err_catcher=new();

    //Add error message string to error catcher.
    ```

```
err_catcher.add_message_text_to_demote("/is not enabled./");

//To affect all reporters, use null for the object.
uvm_report_cb::add(null,err_catcher);
`uvm_info("build_phase", "Exit...", UVM_HIGH)
endfunction: build_phase
```

For more information, see the SolvNetPlus article at
https://solvnet.synopsys.com/retrieve/040138.html

2 Transaction Layer

This section provides a list of frequently asked Transaction Layer (TL) questions for the VC VIP for PCIe.

1. How to enable transaction ordering support?

   To enable transaction ordering support, set the following configuration attribute from `svt_pcie_dut_capabilities` class.

   Example code:

   ```
   cfg.ep_cfg.pcie_cfg.dut_capabilities.enable_transaction_ordering_support  = 1;
   ```

2.1 Transaction Types for Different Address Spaces

1. How do I select the 32 or 64-bit memory address for requests generated by the driver application transaction class?

   Memory address is generated randomly according to the constraints specified in the VIP. The memory addressing is determined by the address attributes in the driver application transaction class. If the upper 32 bits of the address is set to 0 (address [63:32] == 0), then 32-bit memory addressing is used in FMT. Otherwise, 64-bit memory addressing is used.

   You can use constraints in the sequence to select specific addressing:

   ```
   // Transmit MWR by VIP
   `uvm_do_on_with (mem_wr_sequence, vip_seqr.driver_transaction_seqr[0],{
   mem_wr_sequence.length inside {[1:31]};
   mem_wr_sequence.traffic_class == rc_tc_map_seq.tc_num;
   mem_wr_sequence.address_translation == 0;
   mem_wr_sequence.ep == 0;
   mem_wr_sequence.address > 32'hffff;   // For 64-bit addressing only
   // For 32-bit addressing, specify the following:
   //   mem_wr_sequence.address < 32'hffff;
   ```

2. How do I generate message request with data (MsgD)?

   Specify constraints for the following attributes in the TLP class (`svt_pcie_tlp`) to generate message request with data such as Vendor-Defined messages and `Set_Slot_Power_Limit` messages.

Table 2-1 Attribute Description

Attribute	Description
fmt	Format field
tlp_type	TLP type as defined in PCIe specification
message_code	Message code as defined in PCIe specification
length	Specifies the number of DWORDs to be read or written

For example,

Vendor-Defined message:

```
`uvm_create_on(random_tlp,dut_seqr.pcie_virt_seqr.tlp_seqr)

random_tlp.tlp_type= svt_pcie_tlp::MSG_REQ_BY_ID;
random_tlp.fmt                  = svt_pcie_tlp::NO_DATA_4_DWORD;
random_tlp.message_code    = svt_pcie_tlp::VENDOR_DEFINED_0;
random_tlp.td                          = 0;
random_tlp.traffic_class        = 0;
random_tlp.ep                          = 0;
random_tlp.at                          = svt_pcie_tlp::UNTRANSLATED;
random_tlp.length                    = 0;
random_tlp.requester_id        = 16'b0000;
random_tlp.bus_number       = 8'h00;
random_tlp.device_number  = 5'h00;
random_tlp.function_number  = 3'b000;

`uvm_send(random_tlp)

SSPL Msg:

`uvm_create_on(random_tlp,dut_seqr.pcie_virt_seqr.tlp_seqr)

// Local - Terminate at Receiver
random_tlp.tlp_type = svt_pcie_tlp:: MSG_REQ_TERM_RCVR;
random_tlp.fmt= svt_pcie_tlp:: WITH_DATA_3_DWORD;
random_tlp.message_code= svt_pcie_tlp:: SET_SLOT_POWER_LIMIT;
random_tlp.td= 0;
random_tlp.traffic_class= 0;
random_tlp.ep= 0;
random_tlp.at                   = svt_pcie_tlp::UNTRANSLATED;
```

```
random_tlp.length= 0;
random_tlp.application_id = msg_map_appl_id_seq.appl_id;
random_tlp.requester_id= 16'b0000;
random_tlp.bus_number= 8'h00;
random_tlp.device_number= 5'h00;
random_tlp.function_number= 3'b000;

`uvm_send(random_tlp)
```

For more information, see the SolvNetPlus articles at,

https://solvnet.synopsys.com/retrieve/2096908.html

https://solvnet.synopsys.com/retrieve/2335176.html

3. How do I generate memory write and memory read transactions?

   Extend the `svt_pcie_driver_app_transaction_base_sequence` class to specify memory write and memory read transactions.

   For example:

```
// Set up the write transaction
`uvm_create(write_tran)
write_tran.cfg = cfg;
write_tran.transaction_type =
svt_pcie_driver_app_transaction::MEM_WR;
write_tran.address = 32'h0000_4000 | ('h100 < < i);
write_tran.length = 2 + i;
write_tran.traffic_class = 0;
write_tran.address_translation = 0;
write_tran.first_dw_be = 4'b1111;
write_tran.last_dw_be = 4'b1111;
write_tran.ep = 0;
write_tran.block = 1;
write_tran.payload = new[write_tran.length];
foreach (write_tran.payload[i]) begin
write_tran.payload[i] = 32'h1234_0000 + i;
end
// Send the write transaction
`uvm_send(write_tran)
```

4. How do I generate memory write followed by memory read back-to-back transactions to the same memory location?

Generate memory write followed by memory read back-to-back transactions to the same memory location by setting the block attribute of the `svt_pcie_driver_app_transaction_base_sequence` class to 0.

For example:

class `pcie_driver_transaction_directed_seq` extends `svt_pcie_driver_app_transaction_base_sequence;`

```
`uvm_do_on_with(mem_wr_request_seq,p_sequencer.endpoint_virt_seqr.driver_transaction_seqr[0],
{
mem_wr_request_seq.traffic_class  == vc_enable_seq_rc.vc_num;
mem_wr_request_seq.length inside {[1:16]};
mem_wr_request_seq.block == 0;
mem_wr_request_seq.ep ==0;
});

`uvm_do_on_with(mem_rd_request_seq,p_sequencer.endpoint_virt_seqr.driver_transaction_seqr[0],
{
mem_rd_request_seq.address == mem_wr_request_seq.address;
mem_rd_request_seq.first_dw_be == mem_wr_request_seq.first_dw_be;
mem_rd_request_seq.last_dw_be == mem_wr_request_seq.last_dw_be;
mem_rd_request_seq.length == mem_wr_request_seq.length;
mem_rd_request_seq.traffic_class == mem_wr_request.traffic_class;
mem_rd_request_seq.block == 0;
});
```

For more information, see the SolvNetPlus article at

https://solvnet.synopsys.com/retrieve/2060809.html

5. What is an efficient way to generate a large number of memory transactions?

   Use either the driver application or the request initiator (requester) application to generate memory transactions.

   While the driver application supports the generation of all transaction types (configuration, I/O, memory), the requester app (`svt_pcie_requeser_app`) is designed for generating only memory transactions.

   The requester app generates memory transactions with random memory addresses and lengths. Use the `max_seg_address` and `min_seg_address` attributes in the requester service class (`svt_pcie_requester_app_sevice`) to modify the address range. The requester app provides an easy way to create background traffic with a large number of memory transactions.

   For example:

```
class requester_pipe_sequence extends
svt_pcie_device_system_virtual_base_sequence;
…
task requester_pipe_sequence:: body();
```

```
begin
…
// Add memory ranges to Requester application
`uvm_do_on_with( req_mem_range_seq,
p_sequencer.endpoint_virt_seqr.requester_seqr,
                                { req_mem_range_seq.service_type ==
svt_pcie_requester_app_service::ADD_MEM_RANGE;
req_mem_range_seq.sequence_length == 1;
req_mem_range_seq.min_seg_address == 32'h0001_0000;
req_mem_range_seq.max_seg_address == 32'h000F_FFFF; });

`uvm_do_on_with( req_mem_range_seq,p_sequencer.endpoint_virt_seqr.requester_seqr,
                                { req_mem_range_seq.service_type ==
svt_pcie_requester_app_service::ADD_MEM_RANGE;
req_mem_range_seq.sequence_length == 1;
req_mem_range_seq.min_seg_address == 32'hC000_0000;
req_mem_range_seq.max_seg_address == 32'hFFFF_FFFF; });
…
end
endtask
endclass
```

6. How can I configure the VIP to drop a Completion packet after the DUT transmits a non-posted request to the VIP?

   Use the drop feature in the target application callback class (`svt_pcie_target_app_callback`) to drop TLPs. TLPs can be a Requested Packet or a Completion Packet.

   For example:

```
//Callback class that drops the CplD generated automatically by the VIP
class svt_pcie_tl_pre_tx_tlp_put_copy_first_cpld_of_alternate_series_callback
extends
svt_pcie_target_app_callback;
//Factory registration
`uvm_object_utils(svt_pcie_tl_pre_tx_tlp_put_copy_first_cpld_of_alternate_series_
callback)
//Handle to the CPLD generated automatically by the VIP
svt_pcie_tlp cpld;

//Keep track of CPLDs generated automatically by the VIP
local bit cpld_series_started = 0;

//All memory reads are expected twice. Inject errors in the first series via
sequence and
```

```
//transmit the second series without errors
local bit error_injected = 0;

//CONSTRUCTOR: Create a new callback instance
function new(string name =
"svt_pcie_tl_pre_tx_tlp_put_copy_first_cpld_of_alternate_series_callback");
super.new(name);
endfunction

/*
 * Callback issued by the component after scheduling a TLP transaction for
transmission on
 * the link, just prior to framing.
 * Testbench copies the first CPLD of every alternate series into a shared TLP
handle with
 * the sequence as every memory read request is expected twice.
 *
 * target_app => A reference to the component object issuing this callback.
 * transaction => A reference to the svt_pcie_tlp descriptor object of interest.
 * drop => A reference, when set will cause a transaction to be dropped prior to
transmission.
*/

virtual function void pre_tx_tlp_put(svt_pcie_target_app target_app, svt_pcie_tlp
transaction,
ref bit drop);
if (transaction.tlp_type == svt_pcie_tlp::CPL) begin
if (error_injected == 0) begin
drop = 1;
if (cpld_series_started == 0) begin
cpld.copy(transaction);
end
end

if ((transaction.byte_count + transaction.lower_address[1:0]) >
transaction.length*4) begin
cpld_series_started = 1;
end else begin
cpld_series_started = 0;
error_injected = ~error_injected;
end
end
```

```
endfunction: pre_tx_tlp_put
endclass: svt_pcie_tl_pre_tx_tlp_put_copy_first_cpld_of_alternate_series_callback
```

For more information, see the SolvNetPlus article at

https://solvnet.synopsys.com/retrieve/2327562.html

7. How do I validate the completion of an Unsupported Request (UR) with error injection?

   Use the `EXPECT_UR` enum value of the error_kind attribute in the driver application transaction exception class (`svt_pcie_driver_app_transaction_exception`) to confirm that an Unsupported Request (UR) is received by the VIP for a non-posted request.

   For example:

```
`uvm_do_on_with(random_tlp_exception_seq,vip_seqr.driver_transaction_seqr[0],
{
random_tlp_exception_seq.transaction_type inside
{svt_pcie_driver_app_transaction::ATOMIC_OP_SWAP,
svt_pcie_driver_app_transaction::ATOMIC_OP_FETCH_ADD,
svt_pcie_driver_app_transaction::ATOMIC_OP_CAS};

if (random_tlp_exception_seq.transaction_type ==
svt_pcie_driver_app_transaction::ATOMIC_OP_FETCH_ADD ||
random_tlp_exception_seq.transaction_type ==
svt_pcie_driver_app_transaction::ATOMIC_OP_SWAP )
{
random_tlp_exception_seq.length inside {1, 2};
}
else if (random_tlp_exception_seq.transaction_type ==
svt_pcie_driver_app_transaction::ATOMIC_OP_CAS)
{
random_tlp_exception_seq.length inside {2, 4, 8};
}

random_tlp_exception_seq.address == mem_wr_request_seq.address;
random_tlp_exception_seq.payload[0] != temp_data;
random_tlp_exception_seq.traffic_class == tc_map_seq_ep.tc_num;
random_tlp_exception_seq.address_translation  == 0;
random_tlp_exception_seq.ep  == 1;
random_tlp_exception_seq.block == 1'b1;
random_tlp_exception_seq.transaction_error_kind ==
svt_pcie_driver_app_transaction_exception::EXPECT_UR;
random_tlp_exception_seq.num_exceptions == 1;
});
```

8. What is blocking and non-blocking read in PCIe SVT?

'Blocking' read prevents other transactions from being queued. Whereas in case of non-blocking read, a process does not wait for the completion. As a result the transaction is queued. The driver application layer uses the block attribute to control when the driver queues the next transaction.

When it is set to 1, the driver will wait until the transaction is completed before the next transaction is queued.

When set to 0, the driver does not wait for transaction to complete and drives the sub-sequent transaction.

The function of the block bit is to not return control to the user until the completion is received in the case of a read.

```
* block = 1, block until completion is received.
* block = 0, non-block.
For cfg_rd request:

`uvm_do_on_with(cfg_rd_rq,p_sequencer.root_virt_seqr.driver_transaction_seqr[0],
{
            bdf == 16'h0100;
            block == 1'b1; // block until completion is received.
            first_dw_be == 4'hf;
            register_number inside {[0:50]};
});
```

For `Mem_rd` request:

```
`uvm_do_on_with(mem_rd_request_seq,vip_seqr.driver_transaction_seqr[0],
{
address == mem_wr_request_seq.address;
traffic_class == mem_wr_request_seq.traffic_class;
length == mem_wr_request_seq.length;
block == 1'b0; // it will not wait for completion
first_dw_be == mem_wr_request_seq.first_dw_be;
last_dw_be == mem_wr_request_seq.last_dw_be;
});
```

For more information, see the SolvNetPlus article at

https://solvnet.synopsys.com/retrieve/041043.html

9. How can I transmit Type0/Type1 configuration packet from VIP?

   Use `svt_pcie_driver_app_cfg_request_sequence` sequence to transmit Type0/Type1 Cfg TLP from VIP. `Cfg_type` field of the sequence class represents `Type0_Cfg_Req` for value 0, `Type1_Cfg_Req` for value 1.

   For example,

   ```
   svt_pcie_driver_app_cfg_request_sequence  cfg_seq; // Create handle of the
   sequence
   ```

```
------------------------------------------------------
// CFG_WR

`uvm_do_on_with(cfg_seq,p_sequencer.root_virt_seqr.driver_transaction_seqr[0],
{
        transaction_type == svt_pcie_driver_app_transaction::CFG_WR;
        cfg_type == 0;   //represents cfg type0 packet
        bdf == {8'h01, 5'b00000, 3'b0};
        first_dw_be == 4'b1111;
        ep == 0;
        block == 0;
});
```

10. How can I display the entire TLP payload in transaction log file?

    The default setting in VIP is to display 4 DW of header and data payload. There is an attribute in the `svt_pce_dl_disp_pattern` class for printing all the payload data in the log file.

    `max_payload_print_dwords`

    Description: Object Maximum number of payload lines to print

    For example,

    /** Set the payload display limit */

    ```
    svt_pcie_dl_disp_pattern::default_max_payload_print_dwords = 1024;
    ```

11. How can I configure the VIP to not expect a completion for a read request?

    Use the following error injection type so that the VIP expects no completion:

    ```
    svt_pcie_tlp_exception::error_kind_enum : EXPECT_TIMEOUT
    ```

    The error does the following: Informs the VIP that the transaction will eventually timeout waiting for a completion.

Note The transaction is not modified by the VIP in any way to make it violate PCIe specifications. Thus the transaction/packet remains a legal PCIe TLP, but based on some DUT's internal state like credit unavailability, the completion might not reach the VIP.

    The usage is shown in the following code snippet:

    ```
    svt_pcie_driver_app_mem_request_exception_sequence mem_request_seq;
    `uvm_do_on_with(mem_request_seq,p_sequencer.root_virt_seqr.driver_transaction_seq
    r[0],{ mem_request_seq.transaction_type ==
    svt_pcie_driver_app_transaction::MEM_WR;
    mem_request_seq.transaction_error_kind ==
    svt_pcie_driver_app_transaction_exception::EXPECT_TIMEOUT;
    ```

12. How do I control the number of completions sent in response to a read request, in the VC VIP PCIe?

    The number of completions sent mainly depends on the following factors:

- The Read Completion Boundary (RCB). This can be either 64 bytes or 128 bytes.
- The start address in the read request.

- The length of the read request.

  If the length of the request crosses the RCB, the VIP will split the completion into multiple Cpl packets. If the length is within the RCB boundary, only one completion will be sent.

  For example,

- If the RCB is 64 bytes, and the start address of a read request is 0x00004100.
- Read request has a length of 16 dwords (64 bytes), only one completion is sent since, from this start address, the entire completion is within the RCB boundary.
- If the read request has a length of 17 dwords (68 bytes), the RCB boundary is crossed. Hence two completions are sent: one with 64 bytes of data; and the other with the remaining 4 bytes.

13. How to initiate AtomicOp transactions?

    AtomicOp commands are commands that can do a series of things atomically within the target device rather than requiring a series of separate uninterruptable commands on the interface.

    There are three types of AtomicOp commands:

    - FetchAdd – Fetch and Add
    - Swap – Unconditional Swap
    - CAS – Compare and Swap

    Use the following sequences within the `svt_pcie_driver_app_transaction_sequence_collection.sv` to send AtomicOp transactions.

```
svt_pcie_driver_app_transaction_atomicop_fetchadd_sequence
svt_pcie_driver_app_transaction_atomicop_cas_sequence
svt_pcie_driver_app_transaction_atomicop_swap_sequence
```

14. How to generate Malformed TLPs?

    The `svt_pcie_tlp_exception` class provides multiple ways to generate Malformed TLPs.

    To corrupt the format field of a TLP, the `CORRUPT_FMT` exception can be used, similarly to corrupt the type field of a TLP, the `CORRUPT_TLP_TYPE` exception can be used.

    For example, to corrupt FMT field use the following code snippet:

```
class svt_pcie_corrupt_fmt_callback extends svt_pcie_tl_callback;
    // Facroty registration
  `uvm_object_utils(svt_pcie_corrupt_fmt_callback)
  // Exception list instance
  svt_pcie_tlp_exception my_tlp_exc = new("my_tlp_exc");
  svt_pcie_tlp_exception_list my_tlp_exc_list = new("my_tlp_exc_list");

  function new(string name = "svt_pcie_corrupt_fmt_callback");
   super.new();
    my_tlp_exc.error_kind = svt_pcie_tlp_exception::CORRUPT_FMT; // Instruct to corrupt
FMT
    my_tlp_exc.corrupted_data = 7;
    my_tlp_exc_list.add_exception(my_tlp_exc); //Add exception to the list
  endfunction
endclass
```

    To generate `Malformed_TLPs`, use the following exceptions:

    - `CORRUPT_AT`
    - `CORRUPT_TLP_TYPE`

- CORRUPT_FMT
- CORRUPT_PH
- CORRUPT_TH
- CORRUPT_ATTR
- CORRUPT_LENGTH
- CORRUPT_FIRST_DW_BE
- CORRUPT_LAST_DW_BE

15. How to generate MSG TLP's?

    Generate message TLP's, using driver application sequence
    svt_pcie_driver_app_transaction_msg_sequence

    For example,

```
svt_pcie_driver_app_transaction_msg_sequence msg_sequence;
if ((message_code == svt_pcie_driver_app_transaction::VENDOR_DEFINED_0) ||
     (message_code == svt_pcie_driver_app_transaction::VENDOR_DEFINED_1))
begin
     `uvm_do_on_with(msg_sequence,
                       {transaction_type == svt_pcie_driver_app_transaction::MSG;
         message_code == local::message_code;
                        length == local::length;
                        routing_type == local::routing_type;
                        traffic_class == local::traffic_class; })
end
```

16. How to transmit nullified TLP packet from VIP?

    To generate nullified TLP packet, use the error_kind, AUTO_TX_NULLIFIED_TLP attribute of svt_pcie_tlp_exception class.

    The following example explains how to use the svt_pcie_tlp_exception inside the DL callback to transmit a nullified TLP.

    For example,

```
// Create a callback class
class tx_nullified_tlp_cb extends svt_pcie_dl_callback;
  //Exception list instance
  svt_pcie_tlp_exception my_tlp_exc = new("my_tlp_exc");
svt_pcie_tlp_exception_list my_tlp_exc_list = new("my_tlp_exc_list");

 // Factory registration.
`uvm_object_utils(tx_nullified_tlp_cb)
function new(string name = " tx_nullified_tlp_cb ");
  super.new();
  endfunction
// This callback is issued by the component after scheduling a TLP transaction
for transmission on the link, just prior to framing.
 virtual function void pre_tlp_framed_out_put(svt_pcie_tl tl,
svt_pcie_tlp_transaction transaction, ref bit drop);
   my_tlp_exc.error_kind = svt_pcie_tlp_exception:: AUTO_TX_NULLIFIED_TLP;
```

```
    //Add exception to the list
   my_tlp_exc_list.add_exception(my_tlp_exc);
   $cast(transaction.exception_list, my_tlp_exc_list.`SVT_DATA_COPY());
 endfunction
endclass
```

Note The nullification of packets can be verified by searching for EDB symbol at the end of the TLP packet indicating a nullified packet.

The following information is from the `symbol.log` file generated by a 1-lane `pcie_svt` VIP:

Table 2-2 TLP Packet

TLP Packet	TX	RX
89569: root0	01	STP
89573: root0	80	00
89577: root0	8e	01
89581: root0	END	40
89585: root0	00	00
89589: root0	SDP	00
89593: root0	c3	02
89597: root0	19	00
89601: root0	84	01
89605: root0	00	00
89609: root0	3a	ff
89613: root0	cf	10
89617: root0	END	00
89621: root0	00	01
89625: root0	SDP	00
89629: root0	e4	ed
89633: root0	19	cb
89637: root0	44	ff
89641: root0	01	ff
89645: root0	01	ed
89649: root0	a1	cb
89653: root0	END	ff

Table 2-2 TLP Packet

TLP Packet	TX	RX
89657: root0	SDP	fe
89661: root0	c7	70
89665: root0	19	79
89669: root0	84	a6
89673: root0	00	c6
89677: root0	cc	EDB

The following information is from the transaction log file generated by `pcie_svt` VIP:

```
89573.000 89681.000 R MWr32      0x0001/00     1  0 0    0 0 0
0x10000100   f f      2 H40000002 H000100ff H10000100    ---    0
0x7079a6c6 NullTLP
```

17. How to check whether all theTLPs are sent from/received by `pcie_svt` VIP?

    We have `tl_status` variable inside `svt_pcie_tl_status` class which can be used to check whether all the TLPs have been received/sent by VIP.

    For example,

- `num_tlps_sent`

  Validates that all the TLPs are transmitted by `pci_svt` VIP

  //Store the number of tlps sent by VIP

```
no_of_tlps_sent = root_status.pcie_status.tl_status.num_tlps_sent;
```

  //Initiate TLPs transmission from VIP

```
for (i=0;i<sequence_length;i++)
begin
 `uvm_do_on_with(mem_request_seq,p_sequencer.root_virt_seqr.driver_transaction_seqr[0],{
mem_request_seq.transaction_type ==   svt_pcie_driver_app_transaction::MEM_WR;
mem_request_seq.transaction_error_kind ==
svt_pcie_driver_app_transaction_exception::NO_ERROR;     mem_request_seq.num_exceptions
== 0;
mem_request_seq.address inside {[32'h4000_0000 : 32'h8000_0000]};
mem_request_seq.length inside {[16:32] };
mem_request_seq.address_translation == 0;
mem_request_seq.ep == 0;
mem_request_seq.block == 0; });
end
//wait until all the TLPs are sent by VIP
wait(root_status.pcie_status.tl_status.num_tlps_sent == no_of_tlps_sent+sequence_length)
```

- `num_tlps_received`

  Validates that all the TLPs are received by `pci_svt` VIP

  //local variable to store the number of tlps received by `pcie_svt` VIP

```
no_of_tlps_received == root_status.pcie_status.tl_status.num_tlps_received;
```

Note TL status variable is available for every type of TLP (Memory,Configuration,IO,Msg) to track the number of TLP received/sent by VIP.

18. How to check the numbers of posted/Non-posted/completion packet received/sent by `pcie_svt` VIP?

    We have `tl_status` variable for each type of posted/non-posted/completion TLP to count the number of packets received/sent by VIP.

- Posted packet:

  The following example validates the number of posted TLP type: Mwr32 received/sent by VIP

  For example,

```
//Variable to store the number of mem_wr packet received
count_num_mwr32_rcvd = root_status.pcie_status.tl_status.
num_tlp_mem_wr32_received;
//Variable to store the number mem_wr packet sent
count_num_mwr32_sent=
root_status.pcie_status.tl_status.num_tlp_mem_wr32_sent;
```

- Non-posted packet

  The following example validates the number of non-posted TLP type: Mrd received and `io_rd` sent by `pcie_svt` VIP using the following status variable.

```
num_tlp_mem_rd32_received
num_tlp_io_rd_sent
```

  For example,

  //Local variable to store the number of `mem_rd` packet received

```
count_num_mrd32_rcvd =
root_status.pcie_status.tl_status.num_tlp_mem_rd32_received;
```

  //Local variable to store the number of `io_rd` packet sent

```
count_num_io_rd_sent = root_status.pcie_status.tl_status. num_tlp_io_rd_sent;
```

- Completions

  The number of completions received by the VIP can be tracked, with the `num_tlp_cpld_received` tl status variable.

  For example,

```
count_num_cpl_rcvd = root_status.pcie_status.tl_status.num_tlp_cpl_received;
count_num_cpld_rcvd = root_status.pcie_status.tl_status.num_tlp_cpld_received;
```

  The number of completions received by the VIP can be tracked, with the `num_tlp_cpl_sent/ num_tlp_cpld_sent tl_status` variable.

  For example,

```
count_num_cpl_rcvd = root_status.pcie_status.tl_status. num_tlp_cpl_sent;
count_num_cpld_rcvd = root_status.pcie_status.tl_status. num_tlp_cpld_sent;
```

Note We can count for any type of `posted/non_posted` TLP which are received/sent by VIP.

For more information on the `tl_status` variable, see the from the following path

`$DESIGNWARE_HOME/vip/svt/pcie_svt/latest/doc/pcie_svt_uvm_class_reference/html/class_svt_pcie_tl_status.html`

19. How do I limit BDF in TLPs?

    In order to limit the bus number, device numbers and function, you must modify the `svt_pcie_driver_app_transaction` sequence as shown in the following example code:

```
`uvm_do_on_with(cfg_seq
       ,p_sequencer.root_virt_seqr.driver_transaction_seqr[0],{
        transaction_type == svt_pcie_driver_app_transaction::CFG_WR;
        cfg_type == 0;

        bdf == {8'h01, 5'b00000, 3'b0};

        first_dw_be == 4'b1111;
        ep == 0;
        payload == 32'hfacecafe;
        block == 1;
      });
```

20. How can a sequence wait for a Msg TLP?

    This can be done using the `post_tlp_framed_in_get` callback in a test case.

    Any TLP can be accessed using the following callback:

```
================
// Global System Verilog event
event received_inta;

// VIP callback class
class dl_tlp_callback extends svt_pcie_dl_callback;

  // Factory registration
  `uvm_object_utils(dl_tlp_callback)
...
  // Received TL packet
  virtual function void post_tlp_framed_in_get(svt_pcie_dl dl,
                                               svt_pcie_dl_tlp
transaction,
                                               ref bit drop);
    // Check if a MSG INTA is received.
    if(transaction.tlp.message_code == ASSERT_INTA) begin

     // Trigger the testbench event. The test can use the event to wait.
     ->received_inta;
    end
    else begin
      // Do nothing
    end
  endfunction
endclass
```

```
// UVM test class
class my_test extends uvm_test;
...
  // Callback instance
  dl_tlp_callback dl_tlp_cb;

  function void end_of_elaboration_phase(uvm_phase phase);
    int error_kind;

...
   // Register callback
   dl_tlp_cb = dl_tlp_callback::type_id::create("dl_tlp_cb");
   uvm_callbacks#(svt_pcie_dl
,svt_pcie_dl_callback)::add(env.endpoint.port.dl,dl_tlp_cb);

...
    `uvm_info(get_full_name(), "Exiting...", UVM_HIGH);
  endfunction : end_of_elaboration_phase

  task run_phase(uvm_phase phase);
  ...
  // Wait for inta where ever you like in the test
  wait(received_inta);
  endtask
endclass
```

21. How do I resolve the duplicate TLP error?

    Duplicate TLP error occurs when the same TLPs get retried when the `replay_timer` is expired.

    You can use `Max_payload_size`, `rate`, and `linkwidth` to calculate the `replay_timer` value.

    If `MAX_PAYLOAD_SIZE` of both the device does not match, then this may lead to the above error.

    You can configure the `MAX_PAYLOAD_SIZE` of VIP using `max_payload_size` variable of DL configuration (`svt_pcie_dl_configuartion`).

    ```
    root_cfg.pcie_cfg.dl_cfg.max_payload_size = 1024;
    ```

22. How do I initiate memory lock transaction?

    You can use `MEM_RD_LK` as `transaction_type` for `svt_pcie_driver_app_transaction` to issue memory lock transaction from PCIe SVT.

    ```
    //Example for MEM_RD_LK transaction by constraining an exception sequence.
     `uvm_do_on_with(rc_mem_rdlk_sequence,p_sequencer.root_virt_seqr.driver_transaction_s
    eqr[0],{
          traffic_class  == 0;
          transaction_type ==svt_pcie_driver_app_transaction::MEM_RD_LK;
          length == rc_mem_wr_request_seq.length;
          address== rc_mem_wr_request_seq.address;
          transaction_error_kind == svt_pcie_driver_app_transaction_exception::EXPECT_UR;
          ep  == 0;
          block  == 1'b1;
          num_exceptions == 1;
        });
    ```

23. How to restrict a valid `target_memory` range?

    Use the `ADD_MEM_RANGE` `service_type` with the attributes set to 1.

    ```
      svt_pcie_requester_app_service_mem_range_sequence  mem_range_seq;
    ```

```
      `uvm_do_on_with(mem_range_seq,p_sequencer.endpoint_virt_seqr.mem_target_seqr,
                  { mem_range_seq.service_type
==svt_pcie_mem_target_service::ADD_MEM_RANGE;
                      mem_range_seq.min_addr == 64'h0003_0000;
                      mem_range_seq.max_addr == 64'h0005_0000;
                      mem_range_seq.attributes == 1;             });
```

24. How to ensure the VIP sends completions in order and does not interleave completions?

    Set the `force_non_interleaved_completions` configuration attribute of `svt_pcie_target_app_configuration` class to send non-interleave completions.

    ```
    force_non_interleaved_completions = 1'b0;
    ```

    This attribute indicates whether the Target application will interleave completions from multiple requests or not. Following are the allowed values:

- 0 – This is the default value and the Target application may interleave completions from multiple requests.
- 1 – Force the Target application to return all completions for a given request before starting to return completions for any subsequent request.

25. How to resolve the `"MemRd accessed uninitialized memory data at addr"` error from the PCIe SVT VIP?

    Generally, this error can be resolved by performing writes to the memory locations before the read transactions are initiated to the same location. This scenario can be achieved by performing a MemWr transaction followed by a MemRd transaction with the same start address and length using the `svt_pcie_driver_app_mem_request_sequence`.

    For more information, see the SolvNetPlus article 000017622, "

    PCIe SVT VIP: Memory Read Transactions Causing an Uninitialized Memory Error".

    If the VIP needs to be configured to flag no errors when you issue read without writing to its target memory, then the following configurations can be used to disable the check:

    ```
    rc_cfg.target_cfg[0].flag_uninitialized_mem_read=1'b0;
    ep_cfg.target_cfg[0].flag_uninitialized_mem_read=1'b0;
    ```

26. How to get the payload received from DUT inside the VIP agent?

    You need to block the memory read request in order to get the payload from completion packet as a result of memory read request.

    Add the following code block after firing memory read to access the CplD payload:

    ```
    for(int i=0; i<mem_rd_seq.req.payload.size(); i++)
    $display("payload[%0d] = %0h", i, mem_rd_seq.req.payload[i]);
    ```

    Example Code:

    ```
    `uvm_do_on_with(mem_request_seq,p_sequencer.root_virt_seqr.driver_transaction_seqr[0],

    {mem_request_seq.transaction_type==svt_pcie_driver_app_transaction::MEM_WR;
    mem_request_seq.address inside \{[32'h4000_0000 : 32'h8000_0000]}
    ;
        mem_request_seq.length == 1;
        mem_request_seq.traffic_class == 0;
        mem_request_seq.address_translation == 0;
        mem_request_seq.ep == 0;
    ```

```
    mem_request_seq.write_payload[0]==32'h1234_5678;
    mem_request_seq.block == 0;
    mem_request_seq.th== 0;
    mem_request_seq.first_dw_be ==4'b1111;
    mem_request_seq.last_dw_be == 4'h0000;});
    local_addr = mem_request_seq.address;
    local_len = mem_request_seq.length;
    local_first_dw_be = mem_request_seq.first_dw_be;
    local_last_dw_be = mem_request_seq.last_dw_be;
    local_ep = mem_request_seq.ep;

`uvm_do_on_with(mem_request_seq,p_sequencer.root_virt_seqr.driver_transaction_seqr[0],

{mem_request_seq.transaction_type==svt_pcie_driver_app_transaction::MEM_RD;
    mem_request_seq.address == local_addr;
    mem_request_seq.length == local_len;
    mem_request_seq.first_dw_be == local_first_dw_be;
    mem_request_seq.last_dw_be == local_last_dw_be;
    mem_request_seq.traffic_class == 0;
    mem_request_seq.address_translation == 0;
    mem_request_seq.ep == local_ep;
    mem_request_seq.block == 0; }
    );

for(int i=0; i<mem_request_seq.req.payload.size(); i++)
$display("payload[%0d] = %0h", i, mem_request_seq.req.payload[i]);
```

2.2 Packet Field Related Queries

1. How do I control the maximum payload size?

   Use the `remote_max_payload_size` attribute in the TL configuration class (`svt_pcie_tl_configuration`) to control the maximum payload size.

   For example,

   ```
   // Extend the create_cfg() method to customize the configuration in a test case
   virtual function void create_cfg();
   super.create_cfg();
   cfg.rc_cfg.pcie_cfg.tl_cfg.remote_max_payload_size = 512;
   endfunction: create_cfg
   ```

   For example,

   ```
   // Reconfigure the max_payload_size value in a sequence
   vip_cfg.pcie_cfg.tl_cfg.remote_max_payload_size = 256;


   `uvm_do_on_with
   (refresh_cfg_seq,p_sequencer.root_virt_seqr.device_agent_service_seqr,
   {
   service_type == svt_pcie_device_agent_service::REFRESH_CFG;
   })
   ```

For more information, see the SolvNetPlus article at

https://solvnet.synopsys.com/retrieve/2359341.html

2. How do I control the extended tag field?

   Use the `remote_extended_tag_filed_enabled` attributes in the TL configuration class (`svt_pcie_tl_configuration`) to control the extended tag field.

   For example:

```
// Extend the create_cfg() method to customize the configuration in a test case
virtual function void create_cfg();
super.create_cfg();
cfg.rc_cfg.pcie_cfg.tl_cfg.remote_extended_tag_field_enabled = 1'b1;
endfunction: create_cfg
```

3. How do I inject error in the TLP header?

   Use the error_kind attribute in the TLP exception class (`svt_pcie_tlp_exception`) to inject error in the TLP header. The following is a list of error types. For more information on the error types, see the HTML class reference.

Table 2-3 Error Types

Error	Type
NO_ERROR	AUTO_TX_NULLIFIED_TLP
RETAIN_ECRC	AUTO_TX_NULLIFIED_TLP_GOOD_LCRC
RECALC_ECRC	AUTO_TX_NULLIFIED_TLP_CORRUPT_LCRC
FORCE_ECRC	TX_CORRUPT_DISPARITY
CORRUPT_ECRC	AUTO_TX_CODE_VIOLATION
CORRUPT_FMT	TX_MISSING_START
CORRUPT_TLP_TYPE	AUTO_TX_MISSING_END
CORRUPT_MSG_CODE	TX_CORRUPT_8G_HEADER_CRC
CORRUPT_CPL_STATUS	TX_CORRUPT_8G_HEADER_PARITY
CORRUPT_AT	AUTO_TX_RETAIN_LCRC
CORRUPT_PH	AUTO_TX_FORCE_LCRC
CORRUPT_TH	AUTO_TX_CORRUPT_LCRC
CORRUPT_ATTR	AUTO_RX_WITHHOLD_ACK_NAK
CORRUPT_LENGTH	AUTO_RX_NAK_GOOD_TLP
CORRUPT_FIRST_DW_BE	AUTO_RX_REPLAY_COUNT_FAILURE
CORRUPT_LAST_DW_BE	EXPECT_UR

Table 2-3 Error Types

Error	Type
MALFORMED_TLP	EXPECT_CRS
AUTO_TX_ILLEGAL_SEQ_NUM	EXPECT_CA
AUTO_TX_DUPLICATE_SEQ_NUM	EXPECT_TIMEOUT

For example,

Using sequence:

```
`uvm_do_on_with(corrupted_mem_wr_sequence, vip_seqr.pcie_virt_seqr.tlp_seqr,
{
corrupted_mem_wr_sequence.num_exceptions == 1;
corrupted_mem_wr_sequence.transaction_error_kind ==
svt_pcie_tlp_exception::CORRUPT_FIRST_DW_BE;
// Specify additional constraints per user requirements
})
```

Using callback:

```
function new(string name =
"svt_pcie_tl_post_seq_item_get_fbe_0_corrupt_fbe_callback");
super.new();
// Specify to corrupt first_dw_be
my_tlp_exc.error_kind = svt_pcie_tlp_exception::CORRUPT_FIRST_DW_BE;
my_tlp_exc.corrupted_data = 0;

// Add exception to the list
my_tlp_exc_list.add_exception(my_tlp_exc);
endfunction
```

For error types that are not provided in the TLP exception class, you can use constraints to inject error. For more information, see the

For more information, see the SolvNetPlus articles at

https://solvnet.synopsys.com/retrieve/2315907.html

https://solvnet.synopsys.com/retrieve/2369278.html

4. How do I corrupt the VIP completion status?

   Use the `svt_pcie_target_app_callback` class to corrupt completion issued by the VIP. This class includes a feature to drop the TLP and a handle to the Cpl TLP for modifying the completion fields.

   For example:

```
// Drop CPLD generated by VIP
```

```
class svt_pcie_tl_pre_tx_cpl_compl_status_callback extends
svt_pcie_target_app_callback;

// Factory registration
`uvm_object_utils(svt_pcie_tl_pre_tx_cpl_compl_status_callback)

// Keep track of CPLDs generated by VIP
local bit cpld_series_started = 0;

static int i=0;

// CONSTRUCTOR: Create a new callback instance
function new(string name = "svt_pcie_tl_pre_tx_cpl_compl_status_callback");
super.new(name);
endfunction

// Issue callback after scheduling a TLP transaction for transmission on the link,
// just prior to framing

// Testbench copies first CPLD of every series into a shared tlp handle with the
sequence.
//     @param target_appA reference to the component object issuing this callback.
//     @param transactionA reference to the svt_pcie_tlp descriptor object of
interest.
//     @param dropA reference, when set will cause the transaction to be
//dropped prior to transmission.
virtual function void pre_tx_tlp_put (svt_pcie_target_app target_app,
                                                    svt_pcie_tlp transaction,
ref bit drop);
if(transaction.tlp_type == svt_pcie_tlp::CPL) begin
if(i==0) begin
if(cpld_series_started == 0) begin
transaction.completion_status = svt_pcie_tlp::UNSUPPORTED_REQ;
transaction.fmt = svt_pcie_tlp::NO_DATA_3_DWORD;
transaction.length = 0;
transaction.td = 0;
transaction.payload.delete();
end

if((transaction.byte_count + transaction.lower_address[1:0]) >

transaction.length*4) begin
```

```
cpld_series_started = 1;
end else begin
cpld_series_started = 0;
end

i=1;
end
else begin
drop=1;
end
end
endfunction: pre_tx_tlp_put
endclass: svt_pcie_tl_pre_tx_cpl_compl_status_callback
```

For more information, see the SolvNetPlus articles at

https://solvnet.synopsys.com/retrieve/2061608.html

https://solvnet.synopsys.com/retrieve/2339492.html

5. How do I inject error in TLP?

   Use sequence, callback and factory to inject error in TLP.

   For example,

   - Sequence:

```
A sequence can be specified with an exception that includes a defined error kind.
The specified error type is injected in the TLP.
`uvm_do_on_with (corrupted_mem_wr_sequence, vip_seqr.pcie_virt_seqr.tlp_seqr,
{
corrupted_mem_wr_sequence.num_exceptions == 1;
corrupted_mem_wr_sequence.transaction_error_kind ==
svt_pcie_tlp_exception::CORRUPT_FIRST_DW_BE;

// Specify additional constraints per user requirements
})
```

   - Callback:

```
function new(string name =
"svt_pcie_tl_post_seq_item_get_fbe_0_corrupt_fbe_callback");
super.new();

// Corrupt first_dw_be
my_tlp_exc.error_kind = svt_pcie_tlp_exception::CORRUPT_FIRST_DW_BE;
my_tlp_exc.corrupted_data = 0;

// Add exception to the list
my_tlp_exc_list.add_exception(my_tlp_exc);
endfunction
```

- Factory:

  Error can be injected using factory in the test case during the build phase. All TLPs will be transmitted with the injected error.

```
// Add exception to DL so that VIP always issues NAK for received TLPs
xact_exc.NO_ERROR_wt = 0;
xact_exc.set_constraint_weights(0);
xact_exc.AUTO_RX_NAK_GOOD_TLP_wt = 1;
xact_list = new("xact_list", xact_exc);
xact_list.EXCEPTION_LIST_EMPTY_wt = 0;

// Pass the exception list to DL
uvm_config_db#(svt_pcie_dl_tlp_exception_list)::set(this,
"env.endpoint.port0.dl0", "tlp_xact_exception_list", dl_tlp_exc_list);
```

  For more information, see the following SolvNetPlus articles:

  https://solvnet.synopsys.com/retrieve/2298977.html

  https://solvnet.synopsys.com/retrieve/2315907.html

  https://solvnet.synopsys.com/retrieve/1598307.html

  https://solvnet.synopsys.com/retrieve/2363332.html

6. How do I find the number of malformed TLP received by the VIP?

   The TL status class (`svt_pcie_tl_status`) includes the `num_malformed_tlps_received` attribute. Use this attribute in sequence to track the number of malformed TLP received.

   For example:

```
begin
@vip_status.pcie_status.tl_status.num_malformed_tlps_received;
`svt_debug("body","Desired number of malformed TLPs received");
end
```

7. How do I enable analysis ports to observe endpoint TLPs?

   Analysis ports are typically used for scoreboarding. Use the `sent_tlp_interface_mode` and `received_tlp_interface_mode` attributes in the `svt_pcie_dl_configuration` class to enable analysis ports for sent and received TLPs. Port connections are specified in the connect phase of the test.

   For example,

```
root_cfg.pcie_cfg.dl_cfg.sent_tlp_interface_mode = 1;
root_cfg.pcie_cfg.dl_cfg.received_tlp_interface_mode = 1;
```

   For more information, see the SolvNetPlus article at

   https://solvnet.synopsys.com/retrieve/1497240.html

8. How to set EP bit in completion?

   The PCIe VIP enables callbacks during the transmission of TLP. The EP bit can be set with the callbacks after the completion packet is generated by the VIP.
   For more information, see the following SolvNetPlus article at

https://solvnet.synopsys.com/retrieve/2363332.html

9. How to create completion TLPs with status other than Successful Completion?

   TLP completion packets (CPL/CPLD) with a status other than SC (Successful Completion) are called spurious completion TLPs. Spurious completion TLPs can be created by changing the status field of the completion packet using a callback.

   For more information, see the following SolvNetPlus article at

   https://solvnet.synopsys.com/retrieve/2317888.html

10. How to vary completion response latencies?

    There is a min/max range for latency for various address space types. These are available in `svt_pcie_target_app_configuration`:

    ```
    min/max_{io|cfg|mem}_cpl_latency
    ```

    The VIP's strategy is to randomize responses within the window you specify. Simply set the latency in the configuration and the model will automatically vary the completion latency between the transactions.

    Note The `completion_delay_in_ns` attribute is different from the above attributes. For more information on the attributes, see the VC VIP PCI Express UVM User Guide.

    For more information, see the following SolvNetPlus article at

    https://solvnet.synopsys.com/retrieve/1878299.html

11. How to avoid completion timeout errors?

    To avoid completion timeout errors, use the following error injection type:

    ```
    svt_pcie_tlp_exception::error_kind_enum : EXPECT_TIMEOUT
    ```

    For more information, see the following SolvNetPlus article at

    https://solvnet.synopsys.com/retrieve/2351152.html

12. How do I create out of order completion?

    To create out of order completions, use the following configuration variables from `svt_pcie_target_app_configuration` . Using these values, the `pcie_svt` randomize the delays. The entire range between min and max represents out of order completions:

    ```
    rand int unsigned  min_cfg_cpl_latency_ns = 0;
    rand int unsigned  max_cfg_cpl_latency_ns = 0;
    rand int unsigned  min_io_cpl_latency_ns = 0;
    rand int unsigned  max_io_cpl_latency_ns = 0;
    rand int unsigned  min_mem_cpl_latency_ns = 0;
    rand int unsigned  max_mem_cpl_latency_ns = 0;
    rand int unsigned  min_msg_cpl_latency_ns = 0;
    rand int unsigned  max_msg_cpl_latency_ns = 0;
    ```

    For example,

    ```
    endpoint_cfg.target_cfg[0].min_mem_cpl_latency_ns  =    0;
    endpoint_cfg.target_cfg[0].max_mem_cpl_latency_ns  =  1000;
    ```

13. How to corrupt a completion packet transmitted from `pcie_svt` VIP?

The `target_application` callback (`svt_pcie_target_app_callback`) tracks the completion packet transmitted from VIP. We can use the exception (`svt_pcie_tlp_exception`) inside callback class to corrupt the completion packet.

The following example describes how to corrupt some of the fields (`completion_status`, `ep`, `fmt`, `tlp_type`) of the completion packet. Similarly we can corrupt other fields of the completion packet.

For example,

```
Corrupt COMPLETION_STATUS
// Target application callback:
class corrupt_cpl_target_app_callback extends svt_pcie_target_app_callback;
`uvm_object_utils(corrupt_cpl_target_app_callback)
  //Create instance of exception list
  svt_pcie_tlp_exception_list my_tlp_exc_list = new("my_tlp_exc_list");
  svt_pcie_tlp_transaction_exception my_tlp_exc = new("my_tlp_exc");
    // This function is associated with svt_pcie_target_app_callback, will be
called for each TLP packet prior to transmission.
  virtual function void pre_tx_tlp_put(svt_pcie_target_app target_app,
svt_pcie_tlp transaction, ref bit drop);
    if(transaction.tlp_type == svt_pcie_tlp::CPL) begin
             //Set the error_kind attribute to CORRUPT_CPL_STATUS to corrupt the
status field of CPL
            my_tlp_exc.error_kind = svt_pcie_tlp_transaction_exception::
CORRUPT_CPL_STATUS;
      //set the corrupted_data attribute to replace the corrupted field value of
the CPL by this value.
      my_tlp_exc.corrupted_data = 12;
      // Attach the exception to the exception list
      my_tlp_exc_list.add_exception(my_tlp_exc);

            // Attach the exception list to the completion transaction
      $cast(transaction.exception_list, my_tlp_exc_list.`SVT_DATA_COPY());
    end
  endfunction
endclass
Corrupt EP
    my_tlp_exc.error_kind = svt_pcie_tlp_transaction_exception::CORRUPT_EP;
        // Specify the corrupted data value. In this case EP=1.
    my_tlp_exc.corrupted_data[0] = 1;

Corrupt FMT :
    my_tlp_exc.error_kind = svt_pcie_tlp_transaction_exception:: CORRUPT_FMT;
    my_tlp_exc.corrupted_data = 7;
Corrupt TLP TYPE;
```

```
    my_tlp_exc.error_kind = svt_pcie_tlp_transaction_exception::
CORRUPT_TLP_TYPE;
    my_tlp_exc.corrupted_data = 5;
Corrupt LENGTH:
 my_tlp_exc.error_kind = svt_pcie_tlp_transaction_exception:: CORRUPT_LENGTH ;
 my_tlp_exc.corrupted_data = 123;
The below information is from simulation log file generated by pcie_svt VIP shows
the type of corruption being done.
UVM_INFO
../../../../pcie_svt/pcie_target_app_svt/src/svt_pcie_target_app_proxy.sv(1387) @
16829482.90 ps: uvm_test_top.env.endpoint.target_appl0
[process_tlp_tx_exception_list_to_get_error_kind] TX TLP Exception[0] with
error_kind=CORRUPT_CPL_STATUS
UVM_INFO
../../../../pcie_svt/pcie_target_app_svt/src/svt_pcie_target_app_proxy.sv(1387) @
16813482.90 ps: uvm_test_top.env.endpoint.target_appl0
[process_tlp_tx_exception_list_to_get_error_kind] TX TLP Exception[0] with
error_kind=CORRUPT_EP
UVM_INFO
../../../../pcie_svt/pcie_target_app_svt/src/svt_pcie_target_app_proxy.sv(1387) @
16813482.90 ps: uvm_test_top.env.endpoint.target_appl0
[process_tlp_tx_exception_list_to_get_error_kind] TX TLP Exception[0] with
error_kind= CORRUPT_FMT
UVM_INFO
../../../../pcie_svt/pcie_target_app_svt/src/svt_pcie_target_app_proxy.sv(1387) @
16829482.90 ps: uvm_test_top.env.endpoint.target_appl0
[process_tlp_tx_exception_list_to_get_error_kind] TX TLP Exception[0] with
error_kind= CORRUPT_TLP_TYPE
UVM_INFO
../../../../pcie_svt/pcie_target_app_svt/src/svt_pcie_target_app_proxy.sv(1387) @
16861482.90 ps: uvm_test_top.env.endpoint.target_appl0
[process_tlp_tx_exception_list_to_get_error_kind] TX TLP Exception[0] with
error_kind=CORRUPT_LENGTH
UVM_INFO
../../../../pcie_svc/PCIE/Verilog/Application_PCIE/pciesvc_target_appl.sv(6399) @
16861482.90 ps: uvm_test_top.env.endpoint.target_appl0 [report_message]
TxCplTLPCallback: CB - Callback directed us to cause EI=TX_TLP_EI_CORRUPT_LENGTH
(49)
```

14. How to transmit Error poisoned TLP from VIP?

    While creating the driver transaction sequence of the TLP, the `ep` attribute of `svt_pcie_driver_app_transaction` must set to a value of 1.

    For example,

```
// Transmit MWR by VIP
`uvm_do_on_with (mem_wr_sequence, vip_seqr.driver_transaction_seqr[0],{
mem_wr_sequence.length inside {[1:31]};
mem_wr_sequence.traffic_class == 1;
mem_wr_sequence.address_translation == 0;
mem_wr_sequence.ep == 1;
mem_wr_sequence.address ==  32'hABCD_ABCD;
```

15. Does the nullified/malformed Tx TLPs consume credits?

    Credit consumption for nullified/malformed TLP depends on the following factors:

- If the transaction does not have an error injected, then the TL would have consumed credits as appropriate.
- But if the TLP is malformed or nullified via callback (using exceptions), then the credit consumption by the VIP cannot be changed. You must use driver exception to inject nullified TLP error if you want VIP to consume credits against TX nullified TLPs.

16. How to apply different requester IDs to TLPs?

    PCIE_SVT supports changing requester ID by using the `request_id` variable in the `svt_pcie_driver_app_configuration` class:

    ```
    rand bit [15:0]   attribute svt_pcie_driver_app_configuration::requester_id =
    32'h0000_0001
    ```

    Example:

    ```
    cust_cfg.root_cfg.driver_cfg[0].requester_id = 16'hc001;
    ```

    This `request_id` will be applied to all TLPs that VIP will transmit. The `request_id` attribute can be overridden by setting the `enable_tlp_field_user_control_vector` variable with bit 3 set which is in class `svt_pcie_driver_app_configuration` and then setting the `request_id` field in the `svt_pcie_tlp` class.

    Example:

    ```
    cust_cfg.root_cfg.driver_cfg[0].enable_tlp_field_user_control_vector = 16'h0004;
    ```

17. How to set up the VIP to accept error response?

    Use the `svt_pcie_driver_app_transaction_exception` class to set up the expected response using this error_kind.

    Here is the HTML snippet for setting up the VIP to expect CA:

    ```
    typedef enum  svt_pcie_driver_app_transaction_exception::error_kind_enum
    Selects the type of error that will be injected. The error_kind attribute is the top-
    selector of a hierarchical set of descriptors.
    EXPECT_CA( `SVT_PCIE_DRIVER_XACT_EXPECT_CA )
    ```

    Informs the VIP that the request will be terminated by DUT with `COMPLETER_ABORT` completion status. Note that the transaction is not modified by VIP in any which way to make it violate PCIe specifications and thus the transaction/packet remains a legal PCIe TLP, but based on DUT's IO/Memory/Configuration space programming the particular transaction could be terminated as CA by DUT.

18. How to configure the `completer_id` of VIP?

    Completer ID is a concatenation of bus, device and function number.

    PCIe VIP has a target application configuration attribute `completer_id`, and its default value is 32'h0100. This is the default completer ID used until configuration cycles are received to change it.

    ```
    svt_pcie_target_app_configuration::completer_id = SVT_PCIE_COMPLETER_ID
    ```

    Default Completer ID used by the Target application in the generated completions until Configuration Write requests are received on the link to program the completer ID.

19. How to disable interleaving completions packet generated from VIP for multiple request?

You can configure VIP target application class not to interleave the completion packets for multiple request using the following configuration parameter.

```
svt_pcie_target_app_configuration::force_non_interleaved_completions
```

Example:

For more details, see the following HTML class reference documentation:

https://solvnet.synopsys.com/dow_retrieve/latest/verification_compiler/doc/pcie_svt_uvm_class_reference/html/class_svt_pcie_target_app_configuration.html

20. How to configure PCIe SVT VIP to send tags in order?

    Tag selection is random in the VIP. Set the following configuration parameter to generate the ordered tags:

    ```
    svt_pcie_device_configuration::use_ordered_tags
    ```

    Example code for configuration:

    ```
    cfg.rc_cfg.driver_cfg[0].use_ordered_tags = 1;
    ```

21. How to change the max payload size of VIP(RC)?

    The `max_payload_size` parameter in the DL configurations sets the `max_payload_size` of the VIP in bytes. You can configure the `MAX_PAYLOAD_SIZE` of VIP using the `max_payload_size` DL configuration variable (`svt_pcie_dl_configuartion`).

    Example code:

    ```
    <vip_cfg>.pcie_cfg.dl_cfg.max_payload_size = 512;
    ```

Note

- `<vip_cfg>` can be declared by the user.
- `<vip_cfg>` is an object of `svt_pcie_device_configuration` class.

Below VAR from waveform can be used to confirm the configuration settings:

For more information, see the SolvNetPlus article 000020404, "VC VIP: Maximum Allowable Payload Size in PCIe".

2.3 TC and VC

1. How do I enable VC and TC/VC mapping?

   Use service sequences in the TL service class (`svt_pcie_tl_service`) to enable VC and TC/VC Mapping. The `SET_VC_ENABLE` service enables the supported VC in the VIP. The VIP issues an error if more than 7 VCs are enabled. The `SET_TRAFFIC_CLASS_MAP` service maps Traffic Class (TC) with the enabled VC.

   For example,

   ```
   // This sequence enables VC and maps the VC with corresponding TC.
   `uvm_do_on_with(vc_enable_seq_rc,vip_seqr.pcie_virt_seqr.tl_seqr,
   {
   ```

```
vc_enable_seq_rc.vc_num inside {[first_vc:extended_vc_count]};  // Selects 1 VC
out of
// max supported VC
vc_enable_seq_rc.vc_enable == 1'b1;   // Enable the selected VC
});

// TC-VC Mapping
`uvm_do_on_with(tc_map_seq_rc,vip_seqr.pcie_virt_seqr.tl_seqr,
{
tc_map_seq_rc.vc_num == vc_enable_seq_rc.vc_num;
tc_map_seq_rc.tc_num inside {[1:7]};   // Select 1 random TC out of 7 TCs
tc_map_seq_rc.tc_enable == 1'b1;   // Enable TC-VC Mapping
});
```

For more information, see the SolvNetPlus article at

https://solvnet.synopsys.com/retrieve/1822575.html

2. How do I inject error in the TC?

   TLPs generated by the VIP are constrained by valid constrains that are compliant with the PCIe specification. There is no exception available to inject error in TC. You can inject error by disabling the constraints for TC. With constraints disabled, error scenarios such as configuration transactions with non-zero TC can be generated.

   For example,

   `uvm_create_on (transaction_req, vip_seqr.pcie_virt_seqr.tlp_seqr)

```
// Constraint mode off for the resonable_reserved_fields
transaction_req. reasonable_traffic_class.constraint_mode(0);   // Disable
constraint mode

`uvm_rand_send_with(transaction_req,
{
transaction_req.traffic_class  !=  0;

// Specify additional constraints per user requirements
});
```

3. What is a potential reason for the VIP to issue error messages for disabled VCs?

   Error messages are issued when TLPs are transmitted to a TC that is mapped to a disabled VC. The transmitted TLPs are handled as malformed and are dropped before transmitting to the link.

4. How to enable/disable VC's?

   The `svt_pcie_tl_service_set_vc_en_sequence` in TL service class is used to enable/disable VC's as per requirement.

Enable/disable VC's 1 to 7 during the simulation using this sequence.

Note VC0 is enabled by default.

All VC's cannot be disabled simultaneously as the protocol restricts disabling all the VCs. VC0 needs to be active; as VCs serve as gateways to send out TLPs to the Application layer or the Software layer of the system, so the path cannot be closed by disabling all the VCs.

For example,

```
svt_pcie_tl_service_set_vc_en_sequence  set_vc_en_sequence;
  `uvm_do_on_with( set_vc_en_sequence,
                virt_seqr.pcie_virt_seqr.tl_seqr,
                {vc_enable == enable; vc_num    == vc;  }   )
```

5. How do I generate tlps from VIP with non0 TC?

   While creating the driver transaction sequence of the TLP, the `traffic_class` attribute of `svt_pcie_driver_app_transaction` must be set to a non-zero value.

   For example,

```
// Transmit MWR by VIP

`uvm_do_on_with (mem_wr_sequence, vip_seqr.driver_transaction_seqr[0],{
mem_wr_sequence.length inside {[1:31]};
mem_wr_sequence.traffic_class == 1;   // Other than 0
mem_wr_sequence.address_translation == 0;
mem_wr_sequence.ep == 0;
mem_wr_sequence.address ==  32'hABCD_ABCD;
```

Note You must ensure that the appropriate virtual channel has also been mapped to the traffic class. By default, only Traffic Class number 0(TC0) is mapped to Virtual Channel number 0(VC0).

   For TC and VC mapping, see section 2.3.

6. How do I reduce the frequency with which the following message is displayed?
   `DisplayVcQueueContents: VC0 Queue is empty`

   You can use the following configuration parameter to reduce the frequency of this message:

   `svt_pcie_tl_configuration::display_vc_queues_period_ns`

   The default value is 50000.

7. How does the PCIe VIP get configured to follow the reordering rules from the specification?

   The PCIe VIP follows the transaction ordering rules from the specification. By default, it follows the PCI Strongly Ordered Model, first in first out method. You can enable the VIP's Relaxed Ordering rule by setting the `enable_reorder_vc_queues` attribute in the `svt_pcie_tl_configuration` class. The attribute specifies if a TLP from within the queue can be pushed for transmission while there are insufficient credits to transmit the TLP at the head of the VC queue.

   Allowed values:

- 0 – Reordering not allowed during credit starvation for TLP at head of VC queue.
- 1 – Reordering allowed during credit starvation for TLP at head of VC queue.

   Example:

```
cust_cfg.endpoint_cfg.pcie_cfg.tl_cfg.enable_reorder_vc_queues = 1;
```

2.4 ECRC and Digest Field

1. How do I inject error in the ECRC?

   Use the TL callback class (`svt_pcie_tl_callback`) to inject error in the End-to-end Cyclic Redundancy Check (ECRC).

   For example,

```
//Extend the TL callback class to corrupt the ECRC
class svt_pcie_tl_post_seq_item_get_corrupt_ecrc_callback extends
svt_pcie_tl_callback;

//Factory registration
`uvm_object_utils(svt_pcie_tl_post_seq_item_get_corrupt_ecrc_callback)

//Exception list instance
svt_pcie_tlp_exception my_tlp_exc = new("my_tlp_exc");
svt_pcie_tlp_exception_list my_tlp_exc_list = new("my_tlp_exc_list");

//Single error indicator
static int error_count = 0;
static int first_error = $urandom_range(0,10);
static int second_error = $urandom_range(11,20);
static int loop_variable = 0;

function new(string name =
"svt_pcie_tl_post_seq_item_get_corrupt_ecrc_callback");
super.new();

//Instruct to corrupt ECRC
my_tlp_exc.error_kind = svt_pcie_tlp_exception::CORRUPT_ECRC;
my_tlp_exc.corrupted_data = 40;

//Add exception to the list
my_tlp_exc_list.add_exception(my_tlp_exc);
endfunction

//The post_seq_item_get callback method is executed by the TL transactor
//immediately after a sequence item is popped from the sequence_item_port,
//but before any action is taken on it.
//The testbench extension attaches an exception to the transaction and passes
//it back to the transactor
```

```
virtual function void post_seq_item_get(svt_pcie_tl tl, svt_pcie_tlp tlp, ref bit
drop);
`svt_note("post_seq_item_get",$sformatf("error_count: %0d.", error_count));

//Test case decision to determine how many errors to introduce.
if(error_count < 2 &&  (loop_variable == first_error || loop_variable ==
second_error)) begin      //Attach exception list to incoming transaction
`svt_note("post_seq_item_get",$sformatf(" Attaching TLP Force Corrupt ECRC
exception to list.\n."));
$cast(tlp.exception_list, my_tlp_exc_list.`SVT_DATA_COPY());
error_count++;
end

loop_variable++;
endfunction
endclass:svt_pcie_tl_post_seq_item_get_corrupt_ecrc_callback
```

2. How do I generate TLP packets that contain a digest field?

   Use the `percentage_use_tlp_digest` attribute in the driver application configuration class (`svt_pcie_driver_app_configuration`) to generate TLP packets that contain a digest field.

   For example,

   ```
   cfg.driver_app[0].percentage_use_tlp_digest = 50;
   ```

   For more information, see the SolvNetPlus article at

   https://solvnet.synopsys.com/retrieve/2298977.html

   https://solvnet.synopsys.com/retrieve/2339416.html

3. How do I demote ECRC error messages in the TLPs generated by an Endpoint after an ECRC error is injected?

   Use the `svt_err_catcher` class to demote ECRC error messages in the TLPs generated by an Endpoint after an ECRC error is injected.

   For example,

   ```
   virtual function void build_phase(uvm_phase phase);
   `uvm_info("build_phase", "Enter...", UVM_HIGH)
   super.build_phase(phase);

   //Create report catcher object
   err_catcher=new();

   //Add error message string to error catcher
   //Check is active only in VIP-VIP environment
   ```

```
err_catcher.add_message_text_to_demote("/is not enabled./");

//To affect all reporters, use null for the object.
uvm_report_cb::add(null,err_catcher);

`uvm_info("build_phase", "Exit...", UVM_HIGH)
endfunction: build_phase
```

For more information, see the SolvNetPlus article at

https://solvnet.synopsys.com/retrieve/040138.html

4. How to generate TLP without ECRC when TLP is generated by TL layer instead of `driver_app`?

   When TLP is generated by `driver_app`, the `percentage_use_tlp_digest` configuration variable controls the generation of ECRC. But in some cases when TLP is generated by `TL_layer`, td==0 is set inside the sequence to avoid the generation of ECRC.

   For example,

```
svt_pcie_tlp_mem_request_sequence mem_req;
`uvm_do_on_with(mem_req,{tlp_type == local::tlp_type;
                         fmt == local::fmt;
                         traffic_class == local::traffic_class;
                         td == 0;
                         address == local::address; })
```

2.5 TL Flow Control

1. How do I set initial credits in the InitFCs sent by the VIP?

   Use the following attributes in the TL configuration class (`svt_pcie_tl_configuration`) to set the credits in an InitFC for an enabled VC.

Table 2-4 Attribute Description

Attribute	Description
`init_p_hdr_tx_credits[8]`	Initial P header type credits sent to link partner for each VC.
`init_p_data_tx_credits[8]`	Initial P data type credits sent to link partner for each VC.
`init_np_hdr_tx_credits[8]`	Initial NP header type credits sent to link partner for each VC.
`init_np_data_tx_credits[8]`	Initial NP data type credits sent to link partner for each VC.
`init_cpl_hdr_tx_credits[8]`	Initial CPL header type credits sent to link partner for each VC.
`init_cpl_data_tx_credits[8]`	Initial CPL data type credits sent to link partner for each VC.

   For example,

```
// Extend the create_cfg() method to customize the configuration
virtual function void create_cfg();
```

```
`uvm_info("create_cfg", "Enter...", UVM_HIGH)

super.create_cfg();

// configure initial credits for infinite credit advertisement.
for(int i=0; i<8; i++) begin
cfg.rc_cfg.pcie_cfg.tl_cfg.init_p_hdr_tx_credits[i] = 0;
cfg.rc_cfg.pcie_cfg.tl_cfg.init_p_data_tx_credits[i] = 0;
cfg.rc_cfg.pcie_cfg.tl_cfg.init_np_hdr_tx_credits[i] = 0;
cfg.rc_cfg.pcie_cfg.tl_cfg.init_np_data_tx_credits[i] = 0;
cfg.rc_cfg.pcie_cfg.tl_cfg.init_cpl_hdr_tx_credits[i] = 0;
cfg.rc_cfg.pcie_cfg.tl_cfg.init_cpl_data_tx_credits[i] = 0;
end

`uvm_info("create_cfg", "Exit...", UVM_HIGH)
endfunction: create_cfg
```

For more information, see the SolvNetPlus articles at

https://solvnet.synopsys.com/retrieve/2113874.html

https://solvnet.synopsys.com/retrieve/1878756.html

2. How do I delay UpdateFC DLLP transmission from the VIP?

   Use the following attributes in the TL configuration class (`svt_pcie_tl_configuration`) to configure the UpdateFC DLLP delays in the VIP.

Note <n> is the VC number, ranges from 0 to 7.

Table 2-5 Min Max Attributes

Max Attribute	Min Attribute
max_vc<n>_cpl_updatefc_delay	min_vc<n>_cpl_updatefc_delay
max_vc<n>_np_updatefc_delay	min_vc<n>_np_updatefc_delay
max_vc<n>_p_updatefc_delay	min_vc<n>_p_updatefc_delay

For example:

```
// Extend the create_cfg() method to customize the configuration
virtual function void create_cfg();
super.create_cfg();
```

```
// Min delay from rx of TLP to tx of corresponding credit. This can be used to
model delay
// associated with freeing rx TLP buffer space.
cfg.rc_cfg.pcie_cfg.tl_cfg. min_vc0_p_updatefc_delay = 1;

// Max delay from rx of TLP to tx of corresponding credit. This can be used to
model delay
// associated with freeing rx TLP buffer space.
cfg.rc_cfg.pcie_cfg.tl_cfg. max_vc0_p_updatefc_delay = 2;

endfunction
```

For more information, see the SolvNetPlus article at

https://solvnet.synopsys.com/retrieve/2367808.html

1. How do I retrieve credit information from the PCIe VIP?

   The `svt_pcie_tl_status` class has some attributes that can be used to check credits:

   ```
   int attribute
   svt_pcie_tl_status::tx_credits_available[48] = '{48{0}}
   ```

   Credits available for the link partner to transmit TLPs on a given VC. It is an array where individual element in an array represents initial credit limit for a specified VC and for a specified credit type. The array is indexed using `pre_defined` macros which specify VC number and credit type.

   For example, element located at position 46 specifies credits available of type CPL header for VC7. If infinite credits are available then the value returned is -1.

Note Available only when acting as an active component.

```
int attribute
svt_pcie_tl_status::credits_consumed[48] = '{48{0}}
```

Credits consumed: It is an array where individual element in an array represents credits consumed for a specified VC and for a specified credit type. The array is indexed using `pre_defined` macros which specify VC number and credit type.

For example, element located at position 5 specifies credits consumed of type CPL data for VC0.

Note Available only when acting as an active component.

```
svt_pcie_tl_service class:

typedef enum  svt_pcie_tl_service::service_type_enum
```

CHECK_FINAL_CREDITS(16) compares initial allocated credits to final allocated—received credit values. Compares initial limit credits to final limit —consumed credit values. Warnings are issued if any credits are lost.

2. How do I configure VIP to stop sending any UPDATEFC packet for error scenario?

   You can configure svt_pcie_dl_configuration attribute vc0_updatefc_interval_ns to a higher value to block VIP from sending UPDATEFC packets.

   For example,

   ```
   cfg.rc_cfg.pcie_cfg.dl_cfg.vc0_updatefc_interval_ns = 999999999;
   ```

3. How do I set infinite credits for the initfc packets?

   You can control the initial P/NP/CPL header/data type credits sent to link partner for each VC using the following svt_pcie_tl_configuration attributes.The array index is the VC number, 0-7

   Setting header/data credits to 0 indicates infinite credits.

```
virtual function void create_cfg();
`uvm_info("create_cfg", "Enter...", UVM_HIGH)
super.create_cfg();
// Disable check for error Msgs transmitted by DUT
cfg.check_sb_dut_tx_err_msg = 1'b0;
//Configure initial credits
for(int i=0; i<8; i++) begin
   cfg.rc_cfg.pcie_cfg.tl_cfg.init_p_hdr_tx_credits[i] = 0;
   cfg.rc_cfg.pcie_cfg.tl_cfg.init_p_data_tx_credits[i] = 0;
   cfg.rc_cfg.pcie_cfg.tl_cfg.init_np_hdr_tx_credits[i] = 0;
   cfg.rc_cfg.pcie_cfg.tl_cfg.init_np_data_tx_credits[i] = 0;
   cfg.rc_cfg.pcie_cfg.tl_cfg.init_cpl_hdr_tx_credits[i] = 0;
   cfg.rc_cfg.pcie_cfg.tl_cfg.init_cpl_data_tx_credits[i] = 0;
end
endfunction: create_cfg
```

3 Data Link Layer

This section provides a list of frequently asked Data Link Layer (DLL) questions for the VC VIP for PCIe.

3.1 DLL Flow Control

1. How do I access the status of the DLCMSM state machine?

   Use the `dl_state` attribute in the DL status class (`svt_pcie_dl_status`) to determine the current state of the DLCMSM.

   The following are the DLCMSM states:

   - `FC_INIT1`
   - `FC_INIT2`
   - `FC_UPDATE`
   - `DL_DOWN`
   - `DL_INIT`
   - `DL_ACTIVE`

2. How do I enable and disable transmission of InitFC1 and InitFC2 initialization DLLP packet for a particular VC?

   Use the `GATE_TX_INITFC1` and `GATE_TX_INITFC2` services in the DL service class (`svt_pcie_dl_service`) to enable or disable the transmission of InitFC1 and InitFC2.

   For example:

```
svt_pcie_dl_service gate_tx_initfc1_req;
`uvm_do_with(gate_tx_initfc1_req,{
service_type ==  svt_pcie_dl_service::GATE_TX_INITFC1;
vc == local::vc;
fc_type == local::fc_type;
gate_enable == local::gate_enable;})// A value of gate_enable=1: Disable
// A value of gate_enable=0: Enable
svt_pcie_dl_service gate_tx_initfc1_req;

`uvm_do_with(gate_tx_initfc1_req,
{
service_type ==  svt_pcie_dl_service::GATE_TX_INITFC2;
vc == local::vc;
```

```
fc_type == local::fc_type;
gate_enable == local::gate_enable;
})
```

3. How do I generate a timeout for the UpdateFC timer in the DUT?

   The VIP does not have any feature to stop UpdateFC transmission. However, you can create a timeout by setting the following attributes of the DLL configuration class (`svt_pcie_dl_configuration`) to a very large value:

   ```
   min_updatefc_delay = <a large numerical value>
   max_updatefc_delay = <a large numerical value>
   ```

   The VIP waits for the specified large number of clock cycles before sending any UpdateFC packet. This causes the UpdateFC timer in the DUT to timeout.

   For example:

   ```
   env_cfg.m_pcie_vip_cfg.pcie_cfg.dl_cfg.min_updatefc_delay=300000;
   env_cfg.m_pcie_vip_cfg.pcie_cfg.dl_cfg.max_updatefc_delay=300000;
   ```

4. How can I generate a flow control dllp packet?

   VIP automatically generates flow control packets to exchange the Flow Control information during VC Initialization through InitFc1/2 dllp packet and regular updates of credits with updatefc dllp packet. To generate a dllp packet manually, create a sequence extending from `svt_pcie_dllp_base_sequence` class and sequencing it on the dllp sequencer.

   For more information, see the SolvNetPlus article at

   https://solvnet.synopsys.com/retrieve/2082099.html

Generally, FC packets should not be sent manually as it is an automated process performed by DL layer after `dl_link` up is high. User defined dllp may cause blocking of TLPs in case if the credits are insufficient.

5. How do I enable scaled flow control DLLPs from the VIP?

   FC scaling is enabled by the `enable_dl_feature_handshake` DL configuration variable.

   Example:

   ```
   cfg.rc_cfg.pcie_cfg.dl_cfg.enable_dl_feature_handshake=1;
   ```

6. How to setup internal delays in the VIP?

   VIP provides the following configuration for controlling the internal delay in symbol times:

   a. `svt_pcie_dl_configuration::internal_delay_2_5g`: Default value is 19 symbol times and is a constant value as per the specification.

   b. `svt_pcie_dl_configuration::internal_delay_5g`: Default value is 70 symbol times and is a constant value as per the specification.

   c. `svt_pcie_dl_configuration::internal_delay_8g`: Default value is 115 symbol times and is a constant value as per the specification.

   These variables specify the VIP's internal delay in symbol times used in "Ack Transmit Latency Limit" equation for replay timer and AckNak latency timer.

   Example code for configuration:

```
vip_cfg.pcie_cfg.dl_cfg.internal_delay_2_5g= 15;
vip_cfg.pcie_cfg.dl_cfg.internal_delay_5g= 50;
vip_cfg.pcie_cfg.dl_cfg.internal_delay_8g= 100;
```

7. What is the best test method or error injection method to mimic a bit flip scenario that can happen on a wire?

   The PCIe VIP supports flipping a bit error injection and retaining the LCRC, but flipping a bit pre scrambling is not same as mimicking a bit flip on the wire and making sure it is caught. There is an option to attach an exception for disparity error and illegal codeword, but nothing for bit flips. The problem with bit flips is that they result in three possible errors that is very difficult to predict when using 8b/10b encoding:

   a. It can result in a disparity error and that disparity error may be detected immediately or it may not be detected until a later time. If the error is detected immediately, then the packet should be discarded without even LCRC calculation and the DUT should respond with a NAK. If the error is not detected immediately, the LCRC will probably fail but might not if the bit flip caused data representation to remain the same but disparity to flip. Additionally, a later packet may be discarded if the disparity error is detected in that packet, resulting in an unexpected NAK.

   b. It can result in an illegal codeword. These are detected immediately and the packet should be discarded without checking LCRC and the DUT should NAK the packet with the exception.

   c. It can result in a new legal 8b/10b codeword. In this case, the LCRC must be used to find the error and the DUT should respond with a NAK.

   All of these scenarios are dependent on the data inside the TLP—that is, how it is encoded, which byte in the TLP gets the exception, and how that TLP data is striped across lanes at varying link widths. Because it is difficult to predict, we do not recommend using bit flip exceptions. It is very hard to write a randomized test that can correctly predict DUT behavior using this exception.

   Rather than using a bit flip exception, it is recommended to use two different types of exception to verify the correct DUT behavior. These exceptions result in errors that are always detected on the packet where the exception was inserted, making them ideal for randomized testing:

   - An illegal codeword exception: This can be attached to a packet at the DL. An illegal codeword is caught immediately by the decoder on the packet where the error was injected. This covers the scenario where a decoder error is detected and the VIP. Note that you can also try using a disparity error, but depending on the data passing through the decoder the disparity error may not be detected until some time later on a different packet. Changing the disparity does not change the underlying codeword, so an LCRC error will not be detected.
   - A corrupt LCRC: Functionally, there is no difference between flipping a bit inside the TLP and corrupting the LCRC. Going through the trouble of randomly corrupting individual bits adds little to the verification of the DUT unless you are concerned about the correct implementation of the LCRC calculator. The end result of an LCRC error is to discard the packet, and so the contents of that packet are irrelevant.

3.2 LCRC

1. How to inject LCRC ERROR?

   PCIe VIP has exception `AUTO_TX_FORCE_LCRC` to achieve this feature. The `AUTO_TX_FORCE_LCRC` causes the corrupted_data value to be forced into the LCRC field.

   Code snippet for injecting LCRC error:

```
svt_pcie_dl_tlp_exception my_dl_tlp_exc;
```

```
dl_tlp_exception_list_via_callback_exc_list callback_dl_tlp_exc_list;
virtual function void post_tlp_framed_in_get(svt_pcie_dl dl,  svt_pcie_dl_tlp
transaction, ref bit drop);
       my_dl_tlp_exc.corrupted_data = 32'hbaad_baad; // This is the forced value
for the LCRC.
      my_dl_tlp_exc.error_kind = svt_pcie_dl_tlp_exception::AUTO_TX_FORCE_LCRC;
       // Put the exception into the exception list, then copy it into the
transaction
       callback_dl_tlp_exc_list.add_exception(my_dl_tlp_exc);
  endfunction: post_tlp_framed_in_get
```

2. Why the VIP model does not replay TLPs injected with exception "TX_NULLIFIED_TLP_CORRUPT_LCRC"?

   As per the specification, when a nullify TLP is transmitted, the transmitter does not increment NEXT_TRANSMIT_SEQ (stores the packet sequence number applied to TLPs).

   Since copies of transmitted nullified TLPs are not stored in Data link layer retry buffer, it cannot be replayed. This is the reason RC VIP is not replaying Nullified TLP with respective sequence number.

3.3 ACK/NAK and Retry Mechanism

1. How do I create Ack or Nak with a bad sequence number from the VIP?

   Use the dllp_type and ack_nak_seq_num attributes in the DLLP class (svt_pcie_dllp) to generate Ack or Nak with a bad sequence number.

   The dllp_type can be specified as Ack or Nak.

   The ack_nak_seq_num can be specified with a user-defined value.

   // Trigger one Nak/Ack with ack_nak_sequence number does not correspond to an

   // unacknowledged TLP or to the value in ACKD_SEQ.

```
`uvm_do_on_with(dllp_req_seq,p_sequencer.root_virt_seqr.pcie_virt_seqr.dllp_seqr,
{
dllp_req_seq.dllp_type inside { svt_pcie_dllp::ACK,svt_pcie_dllp::NAK};
dllp_req_seq.ack_nak_seq_num inside {[local_seq_len+4:4095]};

// BAD Sequence number:
//     Since there are 4 transactions in this sequence, seq no=
local_seq_len+4:4095 is
//     specified to create sequence number that is not part of the /transaction.
});
```

2. How do I gate a transmission of an ACK or a NAK to allow the VIP to send a collapsed ACK or NAK?

   Use the GATE_TX_ACKNAK service in the DLL service class (svt_pcie_dl_service) to gate transmission of ACK or NAK from the VIP.

   If an ACK or a NAK is gated, then the ACK or NAK is not transmitted. This allows the VIP to receive TLPs from the link partner without transmitting an ACK or a NAK. When enough TLPs are received

from the link partner, the gating of ACK or NAK can be disabled. Then, the VIP sends a collapsed ACK or NAK.

For example,

```
svt_pcie_dl_service gate_tx_acknak_req;
`uvm_do_with(gate_tx_acknak_req,
{
service_type == svt_pcie_dl_service::GATE_TX_ACKNAK;
gate_enable == local::gate_enable;
})
```

3. How do I match the VIP `AckNak_LATENCY_TIMER` with the DUT to prevent late ACK reception warning messages issued by the VIP?

   The `Max_Payload_Size`, rate of transmission and link width are used to calculate the `AckNak_LATENCY_TIMER` value of the VIP and the attached ACK/NAK latency timer value of the DUT.

   Use the `max_attached_ack_nak_latency` attribute of the DL configuration class (`svt_pcie_dl_configuration`) to adjust the attached ACK/NAK latency timer value to match the DUT internal delays.

4. How do I change the sequence number of the first received TLP to create an error?

   You can use the `initial_receive_sequence_value` attribute of the DL configuration class (`svt_pcie_dl_configuration`) to specify the sequence number of the first received TLP to a non-zero value.

   For example,

   ```
   cfg.rc_cfg.pcie_cfg.dl_cfg.initial_receive_sequence_value= 100;
   ```

   If the `initial_transmit_sequence_number` attribute in the DUT is specified to a non-zero value for the first TLP transmitted from the DUT, then the VIP issues an error for this first received TLP. A non-zero sequence number for the first TLP violates the data Integrity check for sequence number.

5. How do I manage VIP ACK/NAK latency?

   Sometimes the following warning is issued by the VIP

   ```
   "ACK rcvd late, expected attached acknak_latency value =<integral number>"
   ```

   The specification defines a timer called the `Ack Nak Latency Timer` at the

   receiver side of the PCIe port. Every time this timer expires the receiver is supposed to send an Ack/Nak (as applicable) from the Ack-Nak generator. The check in the VIP tries to check if this happens correctly from the DUT.

   VIP sets the limit for this timer using the following equation:

   ```
   attached_acknak_latency_timer_limit = attached_max_acknak_latency +
   attached_internal_delay + internal_delay;
   ```

   1) `attached_max_acknak_latency` is the latency value defined by the PCIe

   specification in tables 3-7, 3-8 and 3-9. The VIP has the `SetMaxAttachedAckNakLatency` to set this.

   2) `attached_internal_delay` is to factor in any processing delay at the DUT (Tx). There are configuration parameters at the DL (*svt_pcie_dl_configuration.sv*) to set this:
   `ATTACHED_INTERNAL_DELAY_*G_VAR`

3) `internal_delay` is to factor in any processing delays at the VIP (Rx). There are configuration parameters at the DL (*svt_pcie_dl_configuration.sv*) to set this: `INTERNAL_DELAY_*G_VAR`

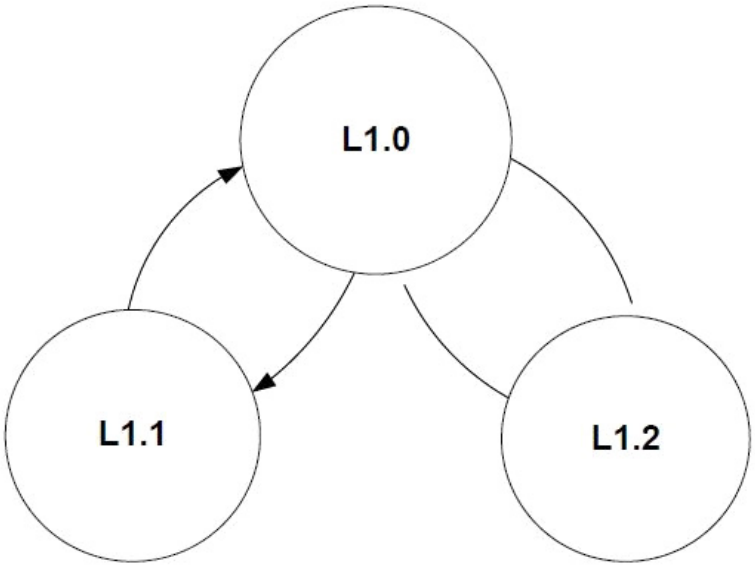
3.4 Low Power States

3.4.1 ASPM

The low power LTSSM states under ASPM are as follows:

- L0s
- L1

  L1 state is further divided into the following substate:



1. How do I initiate ASPM L0s state entry from VIP acting as an endpoint?

   You can initiate ASPM L0s entry from VIP as EP using the following DL service sequence and by setting the DL configuration parameter for L0s entry from the VIP.

- `svt_pcie_dl_service_initiate_aspm_l0s_entry_sequence`
- `svt_pcie_dl_configuration::enable_aspm_l0s_entry`

  For example,

```
// Configuration parameter required to set for L0s_State from testcase:

cfg.root_cfg.pcie_cfg.dl_cfg.enable_aspm_l0s_entry = 1'b1 ;
cfg.endpoint_cfg.pcie_cfg.dl_cfg.enable_aspm_l0s_entry = 1'b1 ;
// Initializing entry to l0s Sequence

l0s_entry_sequence.start(env.endpoint0.pcie_agent.dl_seqr);

//wait for both RC/EP comes to L0S
```

```
wait(root_device.pcie_agent.pl.status.ltssm_state == svt_pcie_types::L0S );
wait(endpoint_device.pcie_agent.pl.status.ltssm_state ==svt_pcie_types::L0S);
```

2. How to enter ASPM L1 state from VIP?

   You can initiate ASPM L1 entry from VIP acting as an EP using the following DL service sequence and DL configuration parameter.

- `svt_pcie_dl_service_initiate_aspm_l1_entry_sequence`
- `svt_pcie_dl_configuration::enable_aspm_l1_entry`

For example,

```
//Configuration parameter required to set for L1_state from testcase:

cfg.root_cfg.pcie_cfg.dl_cfg.enable_aspm_l1_entry = 1'b1 ;
cfg.endpoint_cfg.pcie_cfg.dl_cfg.enable_aspm_l1_entry = 1'b1 ;


svt_pcie_dl_service_initiate_aspm_l1_entry_sequence l1_entry_sequence;

// Initiate entry to L1_state :
l1_entry_sequence.start(env.endpoint0.pcie_agent.dl_seqr);

//wait for both RC/EP comes to L1_IDLE

wait(root_device.pcie_agent.pl.status.ltssm_state == svt_pcie_types::L1_IDLE);
wait(endpoint_device.pcie_agent.pl.status.ltssm_state ==
svt_pcie_types::L1_IDLE);
```

3. How to exit from the ASPM low power states?

   The `svt_pcie_dl_service_initiate_aspm_exit_sequence` DL service sequence is the common sequence to exit from ASPM low power state.

Example 3-1 Exiting from L0s state

```
svt_pcie_dl_service_initiate_aspm_exit_sequence  l0s_exit_sequence;
// Initializing exit from l0s Sequence

l0s_exit_sequence.start(env.endpoint0.pcie_agent.dl_seqr);

wait(root_device.pcie_agent.pl.status.ltssm_state == svt_pcie_types::L0);
wait(endpoint_device.pcie_agent.pl.status.ltssm_state == svt_pcie_types::L0);
```

Example 3-2 Exiting from L1 state

```
svt_pcie_dl_service_initiate_aspm_exit_sequence L1_exit_sequence;
// Initiate exit from L1_state :
L1_exit_sequence.start(env.endpoint0.pcie_agent.dl_seqr)
//Wait until LTSSM is in Recovery.RcvrLock
wait(root_device.pcie_agent.pl.status.ltssm_state == svt_pcie_types::
RECOVERY_RCVRLOCK);
wait(endpoint_device.pcie_agent.pl.status.ltssm_state ==
svt_pcie_types::RECOVERY_RCVRLOCK);
```

4. How to enter L1.1 substate?

   To enter L1.1 substate, set the following DL configuration parameter from VIP:

   ```
   svt_pcie_dl_configuration::enable_aspm_l1_1_entry
   ```

   L1.1 substate has been entered from L1 state, therefore you must initialize L1 state first, then by setting the above configuration parameter it can enter L1.1 state.

   The following example shows how to enter L1.1 substate from VIP.

   ```
   //Set the following DL configuration parameter from testcase
   cfg.root_cfg.pcie_cfg.dl_cfg.enable_aspm_l1_entry = 1'b1 ;
   cfg.endpoint_cfg.pcie_cfg.dl_cfg.enable_aspm_l1_entry = 1'b1 ;

   cfg.root_cfg.pcie_cfg.dl_cfg.dl_cfg.enable_aspm_l1_1_entry = 1'b1 ;
   cfg.endpoint_cfg.pcie_cfg.dl_cfg.dl_cfg.enable_aspm_l1_1_entry = 1'b1 ;
   //Initiate L1 entry from EP VIP
   l1_entry_sequence.start(env.endpoint0.pcie_agent.dl_seqr);

   wait(root_device.pcie_agent.pl.status.ltssm_state == svt_pcie_types::L1_IDLE);
   wait(endpoint_device.pcie_agent.pl.status.ltssm_state == svt_pcie_types::L1_IDLE);

   //Wait for both RC/EP to enter L1.1 substate
   wait(ep_agent.pcie_agent.pl.status.ltssm_state == svt_pcie_types::L1_1);
   wait(root_agent.pcie_agent.pl.status.ltssm_state == svt_pcie_types::L1_1);
   ```

5. How to enter L1.2 substate?

   You can initiate L1.2 entry from VIP as EP using the following DL configuration parameter.

   ```
   svt_pcie_dl_configuration::enable_aspm_l1_2_entry
   ```

   L1.2substate has been entered from L1 state, therefore you must initialize L1 state first, then by setting the above configuration parameter it can enter to L1.2 state.

   ```
   //Set the following DL parameter from VIP
   cfg.root_cfg.pcie_cfg.dl_cfg.enable_aspm_l1_entry = 1'b1 ;
   cfg.endpoint_cfg.pcie_cfg.dl_cfg.enable_aspm_l1_entry = 1'b1 ;

   cfg.root_cfg.pcie_cfg.dl_cfg.enable_aspm_l1_2_entry = 1'b1 ;
   cfg.endpoint_cfg.pcie_cfg.dl_cfg.enable_aspm_l1_2_entry = 1'b1 ;

   //Initiate L1 entry from EP VIP
   l1_entry_sequence.start(env.endpoint0.pcie_agent.dl_seqr);

   //Wait for both RC/EP to come to L1_IDLE
   wait(root_device.pcie_agent.pl.status.ltssm_state == svt_pcie_types::L1_IDLE);
   wait(endpoint_device.pcie_agent.pl.status.ltssm_state == svt_pcie_types::L1_IDLE);

   //Wait for both RC/EP to enter  L1.2 substate.
   wait(ep_agent.pcie_agent.pl.status.ltssm_state == svt_pcie_types::L1_2);
   wait(root_agent.pcie_agent.pl.status.ltssm_state == svt_pcie_types::L1_2);
   ```

6. How to exit from L1.1/L1.2 substate?

   You can initiate exit from L1.1/L1.2 substate to L1 state by asserting clkreq_n signal from VIP using the following DL service sequence.

   ```
   svt_pcie_pl_service_assert_clkreq_n_sequence.
   ```

Example 3-3

```
svt_pcie_pl_service_assert_clkreq_n_sequence rc_clkreq_asserted;
//Initiate exit from L1.1/L1.2 state
`uvm_do_on_with(rc_clkreq_asserted,p_sequencer.root_virt_seqr.pcie_virt_seqr.pl_seqr, {
rc_clkreq_asserted.assert_clkreq_n == 1'b1 ;} )

wait(ep_agent.pcie_agent.pl.status.ltssm_state == svt_pcie_types::L1_IDLE);
 or
wait(ep_agent.pcie_agent.pl.status.ltssm_state == svt_pcie_types::L2_IDLE);
```

7. How to de-assert `clkreq` when LTSSM transition to L1?

   VIP model has a configuration attribute that controls `clkreq`# in the L1 low power state. This attribute needs to be asserted when the VIP enters L1 State:

   ```
   svt_pcie_pl_configuration::enable_l1_clk_power_management =
   SVT_PCIE_ENABLE_L1_CLK_POWER_MANAGEMENT_DEFAULT
   ```

   When this attribute is set to 0, LTSSM will keep `clkreq`# asserted in the L1 low power state.

   When set to a 1, the LTSSM will de-assert `clkreq`# when entering the L1 low power state.

As per the PIPE spec, if `clkreq`# is de-asserted then the LTSSM should move to P2 instead of P1. This control is valid only when L1 substate support is not enabled.

8. How to control entry and exit delay of phystatus assertion in various PowerDown states of MAC?

   If you use `svt_pcie_pipe_phystatus_response_configuration` class, then you can control the phystatus assertion using following attributes:

- `p2_entry_delay`: Controls delay in ns for phystatus assertion in response to PowerDown change from P0/P1 to P2.
- `p2_exit_delay`: Controls delay in ns for phystatus assertion in response to PowerDown change from P2 to P1/P0.
- `p0s_entry_delay`: Controls delay in ns for phystatus assertion in response to PowerDown change from P0 to P0s.
- `p0s_exit_delay`: Controls delay in ns for phystatus assertion in response to PowerDown change from P0s to P0.
- `p1_entry_delay`: Controls delay in ns for phystatus assertion in response to PowerDown change from P0 to P1.
- `p1_exit_delay`: Controls delay in ns for phystatus assertion in response to PowerDown change from P1 to P0.

  Example code:

  ```
  cfg.rc_cfg.pcie_cfg.pl_cfg.phystatus_response_cfg[0].p1_exit_delay = 500;
  ```

9. How to control assertion/de-assertion of pclk in P2 state.

   Set the following configuration attribute of the `svt_pcie_pl_configuration` to turn-off pclk in P2 state.

   Example code:

   ```
   cfg.rc_cfg.pcie_cfg.pl_cfg.pclk_turned_off_in_p2 = 0;
   cfg.ep_cfg.pcie_cfg.pl_cfg.pclk_turned_off_in_p2 = 0;
   ```

3.4.2 PCI-PM

1. How to enter PM L1 state?

   You can initiate PM L1 state from VIP as EP using the following DL service sequence.

   `svt_pcie_dl_service_initiate_pm_l1_entry_sequence`

Example 3-4

```
svt_pcie_dl_service_initiate_pm_l1_entry_sequence l1_entry_sequence
// Initiate entry to L1 State:
l1_entry_sequence.start(env.endpoint0.pcie_agent.dl_seqr);
wait(root_device.pcie_agent.pl.status.ltssm_state == svt_pcie_types::L1_IDLE );
wait(endpoint_device.pcie_agent.pl.status.ltssm_state ==
svt_pcie_types::L1_IDLE);
```

2. How to enter PM L23 state?

   You can initiate PM L1 state from VIP as EP using the following DL service sequence.

   `svt_pcie_dl_service_initiate_pm_l23_entry_sequence`

Example 3-5

```
// Initiate entry to L23 State:

l23_entry_sequence.start(env.endpoint0.pcie_agent.dl_seqr);

wait(root_device.pcie_agent.pl_status.ltssm_state == svt_pcie_types::L2_IDLE );
wait(endpoint_device.pcie_agent.pl_status.ltssm_state ==
svt_pcie_types::L2_IDLE);
```

3. How to exit from PCI_PM state?

   The `svt_pcie_dl_service_initiate_pm_exit_sequence` DL service sequence is the common sequence to exit from PCI-PM low power state.

Example 3-6

```
svt_pcie_dl_service_initiate_pm_exit_sequence exit_from_l1;

// Initiate exit from L1 State:

exit_from_l1.start(env.endpoint0.pcie_agent.dl_seqr);

wait(root_device.pcie_agent.pl.status.ltssm_state == svt_pcie_types::L0) ;
wait(endpoint_device.pcie_agent.pl.status.ltssm_state == svt_pcie_types::L0)
```

Example 3-7

```
svt_pcie_dl_service_initiate_pm_exit_sequence exit_from_l23;

// Initiate exit from L23 State:

exit_from_l23.start(env.endpoint0.pcie_agent.dl_seqr);

wait(root_device.pcie_agent.pl.status.ltssm_state == svt_pcie_types::L0) ;
wait(endpoint_device.pcie_agent.pl.status.ltssm_state == svt_pcie_types::L0);
```

4. How to keep VIP in L2 state after Beacon is received?

   As per the PCIe specification (section 4.2.6.8.1 L2.Idle), once the beacon condition is satisfied device will move from L2 state to Detect state.

   You can keep the pcie_svt VIP's `tx_elec_idle` asserted for longer time to avoid this beacon condition. This can be achieved by setting `tx_idle_min_timer_ns` from `svt_pcie_pl_configuration` class.

   Example code:

   ```
   cfg.root_cfg.pcie_cfg.pl_cfg.tx_idle_min_timer_ns = 20;
   ```

4 Physical Layer

This section provides a list of frequently asked Physical Layer (PL) questions for the VC VIP for PCIe.

1. How do I configure the VIP to generate lane-to-lane skews?

   Use the `min_tx_lane_skew_<data_rate>` and `max_tx_lane_skew_<data_rate>` attributes of the PL configuration class (`svt_pcie_pl_configuration`) to specify the lane-to-lane skews.

   For example,

   //Sets the minimum and maximum lane skews between transmit lanes in symbol times

   //at the specified data rate.

   //Skew is randomized on every link down event, speed change, and at initial startup.

Table 4-1 Data Rate

Data rate of 2.5Gb/s
`rand int unsigned min_tx_lane_skew_2_5g = `SVT_PCIE_MIN_TX_LANE_SKEW_2_5G_DEFAULT;`
`rand int unsigned max_tx_lane_skew_2_5g = `SVT_PCIE_MAX_TX_LANE_SKEW_2_5G_DEFAULT;`
Data rate of 5Gb/s
`rand int unsigned min_tx_lane_skew_5g = `SVT_PCIE_MIN_TX_LANE_SKEW_5G_DEFAULT;`
`rand int unsigned max_tx_lane_skew_5g = `SVT_PCIE_MAX_TX_LANE_SKEW_5G_DEFAULT;`
Data rate of 8Gb/s
`rand int unsigned min_tx_lane_skew_8g = `SVT_PCIE_MIN_TX_LANE_SKEW_8G_DEFAULT;`
`rand int unsigned max_tx_lane_skew_8g = `SVT_PCIE_MAX_TX_LANE_SKEW_8G_DEFAULT;`
Data rate of 16Gb/s
`rand int unsigned min_tx_lane_skew_16g = `SVT_PCIE_MIN_TX_LANE_SKEW_16G_DEFAULT;`
`rand int unsigned max_tx_lane_skew_16g = `SVT_PCIE_MAX_TX_LANE_SKEW_16G_DEFAULT;`

2. How to increase or decrease allowable receive skew in the VIP?

   The default maximum allowable skew is 1 symbol time so you will see the following UVM_ERROR in your simulation:

```
RxDeskewLanes: Lane skew of N has exceeded the maximum allowed lane skew of 1 symbol
times.
```

However, based on your DUT you can configure skew tolerance in the VIP's receive side (even transmit side)

Set the following PL configuration variables of *svt_pcie_pl_configuration.sv* class in your VIP configuration file:

```
===========
/** Maximum allowed lane skew before the SVC flags a skew violation on the receive side
at 2.5G.  */
  max_rx_lane_skew_2_5g

  /** Maximum allowed lane skew before the SVC flags a skew violation on the receive
side at 5G. */
  max_rx_lane_skew_5g

/** Maximum allowed lane skew before the SVC flags a skew violation on the receive side
at 8G. */
int unsigned max_rx_lane_skew_8g

  /** Maximum allowed lane skew before the SVC flags a skew violation on the receive
side at 16G. */
  int unsigned max_rx_lane_skew_16g
===========
```

For example,

```
root_cfg.pcie_cfg.pl_cfg.max_rx_lane_skew_2_5g= 4;
```

3. How to skip link training?

   The `skip_initial_link_training` variable in `svt_pcie_pl__configuration` class enables the pcie_svt model to bypass link training. It allows link training to be completely skipped.

   Example code:

   ```
   cfg.pl_cfg.skip_initial_link_training = 1;
   ```

4. How to set the phy_status response time in the VIP model?

   You can set the following configuration fields of `svt_pcie_pl_configuration` class to change `phy_status` time.

   ```
   svt_pcie_pl_configuration::max_spipe_phystatus_delay
   svt_pcie_pl_configuration::min_spipe_phystatus_delay
   ```

   - `max_spipe_phystatus_delay`: Maximum number of pipe clk cycles the VIP will wait before asserting `phystatus` indicating completion of a PHY function (for example, power state change or rate change). Valid only for SPIPE models.
   - `min_spipe_phystatus_delay`: Minimum number of pipe clk cycles the VIP will wait before asserting `phystatus` indicating completion of a PHY function (for example, power state change or rate change). Valid only for SPIPE models.

5. How do I update the phystatus timeout value?

   You can use the following attributes in the PHY layer configuration class (`svt_pcie_pl_configuration`) to update phystatus timeout value.

   `svt_pcie_pl_configuration::phystatus_timeout_ns` has a default value of 1000.

   Example code to increase the phystatus timeout value:

```
root_cfg.pcie_cfg.pl_cfg.phystatus_timeout_ns= 20000;
```

6. How to define the bit period of VIP's clocks?

   You can write to the following registers to define the bit-period of the VIPs clocks.

   ```
   tb_top.endport0.model.clkgen_txclk0.PCIE_2_5G_BIT_CLK_PERIOD_VAR = <value>
   ```

   A frequency higher than the standard is used, the clock compensation logic in the DUT should remove SKP's more frequently to prevent the elastic buffer from overflowing

   Example code:

   ```
   tb_top.endport0.model.clkgen_txclk0.PCIE_2_5G_BIT_CLK_PERIOD_VAR = <value>//for 2.5GT/s
   tb_top.endport0.model.clkgen_txclk0.PCIE_5G_BIT_CLK_PERIOD_VAR = <value>//for 5GT/s
   tb_top.endport0.model.clkgen_txclk0.PCIE_8G_BIT_CLK_PERIOD_VAR = <value>//for 8GT/s
   tb_top.endport0.model.clkgen_txclk0.PCIE_16G_BIT_CLK_PERIOD_VAR = <value>//for 16GT/s
   ```

7. Can I increase beacon signal transmission time?

   VIP beacon signal pulse width is controlled by PL configuration variable. `svt_pcie_pl_configuration::beacon_signal_pulse_width`.

   Example:

   ```
   cfg.pcie_cfg.pl_cfg.beacon_signal_pulse_width = 2 //in ns
   ```


Note It must be under valid range (2ns to 33us).

4.1 Encoding/Decoding

1. How do I inject an 8b/10b decode error?

   Use the symbol exception class (`svt_pcie_symbol_exception`) and the `pre_symbol_out_put` attribute of the PL callback class (`svt_pcie_pl_callback`) to inject an 8b/10b decode error.

   For example,

   - If VIP is transmitting data with 8b/10b encoding, then use the `error_kind` `CODE_VIOLATION` of the symbol exception class (`svt_pcie_symbol_exception`) to inject an error.

   ```
   //This Callback class is extended from svt_pcie_pl_callback class.
   class svt_pcie_pl_pre_symbol_out_put_corrupt_8b_10b_config_lw_start_callback
   extends
   svt_pcie_pl_callback;
   //Facroty registration
   `uvm_object_utils(svt_pcie_pl_pre_symbol_out_put_corrupt_8b_10b_config_lw_star
   t_callback)
   int  i;

   //Single error indicator
   static int error_count = 0;
   function new(string name =
   "svt_pcie_pl_pre_symbol_out_put_corrupt_8b_10b_config_lw_start_callback");
   super.new();
   endfunction
   virtual function void pre_symbol_out_put(svt_pcie_pl pl, svt_pcie_symbol
   symbols[]);
   ```

```
`uvm_info("pre_symbol_out_put",$sformatf("\n Link width = %0d. Symbols ready
for
transmission.\n",symbols.size()));

If (symbols[0].symbol_type == svt_pcie_symbol::DATA_8B_10B &&
pl.status.ltssm_state ==
svt_pcie_types::CONFIGURATION_LINKWIDTH_START) begin
//Test case decision to determine how many errors to introduce.
if(error_count < 1 ) begin
for(i=0; i< symbols.size(); i++) begin
svt_pcie_symbol_exception_list sym_exc_list = new();
svt_pcie_symbol_exception sym_exc = new();
sym_exc.error_kind = svt_pcie_symbol_exception::CODE_VIOLATION;
sym_exc.scrambler_control = svt_pcie_symbol_exception::FORCE_SCRAMBLE;
sym_exc_list.add_exception(sym_exc);
symbols[i].exception_list = sym_exc_list;
`svt_note("TEST: pre_symbol_out_put_callback",$sformatf("\nCorrupting 8b_10b
encode error on lane %d.\n", i));
end
error_count = error_count + 1;
end
end
endfunction
endclass //
svt_pcie_pl_pre_symbol_out_put_corrupt_8b_10b_config_lw_start_callback
```

- Register the callback in the test case

```
//End of elaboration phase
function void end_of_elaboration_phase(uvm_phase phase);
`uvm_info("end_of_elaboration_phase", "Entry");
rc_sym_cb = new("rc_sym_cb");
uvm_callbacks#(svt_pcie_pl,svt_pcie_pl_callback)::add(rc_vip_agent.pcie_agent.
pl,rc_sym_cb);
`uvm_info("end_of_elaboration_phase", "Exit");
endfunction
```

4.2 Scrambling

1. How do I enable/disable scrambling in the PCIe SVT VIP?

   Enable/disable scrambling by setting the configuration member `disable_scrambling` in the PHY configuration class.

   If by default the variable is set to 0, then the model is scrambling data:

   Disabling of scrambling is possible only with 2.5 GT/s and 5.0 GT/s data rates, Gen3 and above the data is always scrambled.

   For example,

   ```
   cfg.pcie_cfg.pl_cfg.disable_scrambling = 1 //disable scrambling
   ```

   ```
   cfg.pcie_cfg.pl_cfg.disable_scrambling = 0 //enable scrambling
   ```

   For more information, see the SolvNetPlus article at

   https://solvnet.synopsys.com/retrieve/2349545.html

4.3 Link Speed

1. How do I specify the link speed?

There are two methods to specify the supported and target link speed.

- Set the supported and target link speed variables in the PL configuration class.

  For example,

  ```
  svt_pcie_pl_cfg.supported_speeds = `SVT_PCIE_SPEED_8_0G
  svt_pcie_pl_cfg.target_speed = `SVT_PCIE_SPEED_8_0G
  ```

  Call the `set_link_speed_values` and `set_link_width_values` functions in the PL configuration class.

  For example,

  ```
  svt_pcie_pl_cfg.supported_speeds(`SVT_PCIE_SPEED_8_0G | `SVT_PCIE_SPEED_5_0G | `SVT_PCIE_SPEED_2_5G)
  svt_pcie_pl_cfg.set_link_speed_values(`SVT_PCIE_SPEED_8_0G)
  ```

- Using functions in the PL configuration class, is recommended. In addition to setting the supported and target link speed, these functions set the required attributes for calculating the link up speed.

For more information, see the SolvNetPlus article at

https://solvnet.synopsys.com/retrieve/1904896.html

2. How do I reconfigure the target speed?

   You can use the `REFRESH_CFG` service in the device agent service class (`svt_pcie_device_agent_service`) to reconfigure the target speed and other configuration attributes in the PL configuration class.

   For example:

   ```
   // Reconfigure the target speed to Gen2 speed
   vip_cfg.pcie_cfg.pl_cfg.set_link_speed_values(32'h0000_0006,`SVT_PCIE_SPEED_5_0G,
   `SVT_PCIE_SPEED_5_0G );
   `uvm_do_on_with (reconfigure_vip_cfg,vip_seqr.device_agent_service_seqr,
   {reconfigure_vip_cfg.service_type ==
   svt_pcie_device_agent_service::REFRESH_CFG;});
   ```

3. How do I obtain the actual data transfer rate of a link?

   Obtain the data transfer rate of a link from the `current_speed` attribute in the PL status class (`svt_pcie_pl_status`).

   Example:

   ```
   svt_pcie_device_status root_status, endpoint_status;
   wait( vip_status.pcie_status.pl_status.current_speed == svt_pcie_pl_status::SPEED_8_0G);
   ```

   Note

   - `<vip_status>` can be declared by the user.
   - `<vip_status>` is an object of `svt_pcie_device_status` class.

4. How to block automatic speed change by the VIP when configured as endpoint?

   VIP model will try to negotiate to the highest speed that is mutually supported between itself and its link partner. Since your intension is to block automatic speed change by the VIP, you can set `enable_auto_directed_speed_change` variable in the `svt_pcie_pl_configuration` class.

Example code:

```
cfg.pcie_cfg.pl_cfg.enable_auto_directed_speed_change=0;
```

5. How to check if the VIP has linked in configured speed and when link training is completed?

   PCIe VIP asserts the `svt_pcie_pl_status` variable `link_up` status when Gen1 training is finished.

   To make sure that VIP has linked to configured speed and link training has completed, wait until `svt_pcie_dl_status` variable `dl_link_up` is high.

   Once `dl_link_up` is high, DLCMSM state moves to `DL_ACTIVE` state and this indicates that VIP has linked to the configured speed, and LTSSM state enters L0 of the configured speed.

6. How to reconfigure using the VIP's task?

   You can use the reconfigure task:

```
root_agent.reconfigure(rc_cfg);
ep_agent.reconfigure(ep_cfg);
```

   Basic Example:

```
svt_pcie_pl_service_initiate_speed_change_sequence init_speed_change_seq;
svt_pcie_device_agent root_agent, ep_agent;
`uvm_do_on(init_speed_change_seq,p_sequencer.root_virt_seqr.pcie_virt_seqr.pl_seqr);
rc_cfg.pcie_cfg.pl_cfg.set_link_speed_values(`SVT_PCIE_SPEED_5_0G|`SVT_PCIE_SPEED_2_5G);
ep_cfg.pcie_cfg.pl_cfg.set_link_speed_values(`SVT_PCIE_SPEED_5_0G|`SVT_PCIE_SPEED_2_5G);
root_agent.reconfigure(rc_cfg);
ep_agent.reconfigure(ep_cfg);
```

4.4 Link Width

1. How do I disable all or particular lanes and transmit Electrical Idle on lanes?

   You can set the `phy_enable` variable in the PL service class to 0. This setting directs the LTSSM to transition from the Config.LinkWidth.Start or Recovery.Idle state to the Disabled state. In the Disabled state, the VIP transmits Electrical Idle on all required lanes.

2. How do I specify the randomized values for the link width?

   Use the `set_link_width_values()` method to specify the randomized values for the link width. This method sets the `supported_link_width_vector`, `expected_link_width_value` and `link_width_value`.

You can constraint the `link_width_value` to either 1, 2, 4, 8 ,16 or 32.

For example,

```
rand int local_link_width;
constrain tc_local_link_width {local_link_width inside {1, 2, 4,8,16,32}; }

local_link_width.randomize();
env_cfg.m_pcie_vip_cfg.pcie_cfg.pl_cfg.set_link_width_values(local_link_width);
```

For more information, see the SolvNetPlus article at

https://solvnet.synopsys.com/retrieve/2263409.html

3. What is a potential cause for data transmitting on 2 lanes in a x1 Link configuration?

By default, the data is transmitted on Lane 0 when a VIP is configured as x1 Link. The `CONFIGURE_LANE service_type` of the PL service class (`svt_pcie_pl_service`) can be used to enable transmission on an additional lane in a x1 Link configuration. Therefore, data can be transmitted on 2 lanes in a x1 Link configuration if the `CONFIGURE_LANE` service type is used.

4. How to make VIP MAC drive turn off signaling (Tx Compliance high) to unused lanes of the link?

   Set PL configuration attribute `assert_turn_off_signaling_continuously` from *svt_pcie_configuration.sv* file to 1 will make MPIPE drive Tx Compliance signal continuously to 1 for several PCLK cycles.

   Description of `assert_turn_off_signaling_continuously`:

   When the `mpipe` decides to use signaling to turn off lanes that are not part of the link, setting this to 1 makes the `mpipe` drive the turn off signaling continuously. A value of zero will cause the turn off signaling to occur for one pclk cycle.

5. Does PCIe VIP SERDES PHY support two-lane mode?

   The VIP model does support the two lanes.

   PCIe SVT VIP supports the two lanes, VIP has instantiation models for x4, x8, and x32. You need to pick the model as per your requirement.

   For example, you can use two lanes with x32 instantiation model, and ensure to tie off the other unused lanes.

   For more details, see “Instantiation Model Checklist” in the *VC Verification IP PCI Express UVM User Guide*.

   To use the VIP in x2 using your x4 instantiation model, you must modify your configuration parameter `set_link_width_values`. The HTML snippet is as follows:

   ```
   function void svt_pcie_pl_configuration::set_link_width_values (int unsigned
   link_width_value , bit [31:0] supported_link_width_vector_value = 32'hffffffff,
   int expected_link_width_value = -1  )
   ```

   In the Unified TB, you just need to use the macros to instantiate the correct configuration.

   Example:

   Command line: `SVT_PCIE_TARGET_LINK_WIDTH`

   Valid Values as per PCIe Spec are 1,2,4,8,12,16, and 32.

6. How to reconfigure the `link_width` using `pl_cfg.dut_receiver_present`?

   The `dut_receiver_present` PL configuration variable is used to disable the receiver detection on the respective lanes to downsize or upsize link width.

   To set the required lanes, the VIP will see as present from the DUT when the VIP performs a receiver detect in the `Detect.Active` state, use the following configuration member:

   ```
   svt_pcie_pl_configuration::dut_receiver_present.
   ```

   Each bit corresponds to each lane, with bit 0 corresponding to lane 0, and with bit 1 corresponding to lane 1 and so on.

   Example code:

   ```
   vip_cfg.pcie_cfg.pl_cfg.dut_receiver_present = 32'h0000_0001
   ```

7. How can upconfigure be supported in the VIP?

To enable upconfigure support, set the `enable_upconfigure_support` configuration attribute from `svt_pcie_pl_configuration` class.

Example code:

```
cfg.pcie_cfg.pl_cfg.enable_upconfigure_support = 1'b1;
```

8. How can up-down link width be supported in VIP?

   To enable up-down link width support, set the following configuration attribute from `svt_pcie_dut_capabilities` class.

   Example code:

```
cfg.ep_cfg.pcie_cfg.dut_capabilities.enable_up_down_linkwidth_config_support  = 1;
```

9. How to enable upconfigure support using VIP configurations?

   VIP provides the following configuration attributes to enable the upconfiguration support:

- `svt_pcie_dl_configuration::enable_upconfigure_support`: Enables support for upconfiguration.
- `svt_pcie_dut_capabilities::enable_up_down_linkwidth_config_support`: DUT has full linkwidth up/down configure capability support according to specification.

Example code for configuration:

```
cfg.ep_cfg.pcie_cfg.dut_capabilities.enable_up_down_linkwidth_config_support = 1'b1;  //
for endpoint DUT
cfg.rc_cfg.pcie_cfg.pl_cfg.enable_upconfigure_support = 1'b1; // for RC VIP
```

4.5 Lane Reversal

1. How to perform lane reversal check?

   To check lane reversal check, VIP provides the following configuration parameter:

```
svt_pcie_pl_configuration::lane_reversal_mode
```

Set the parameter to the following values and perform lane reversal checking:

- UNSUPPORTED (32'h0): Lane reversal is unsupported.
- SUPPORTED (32'h1): Lane reversal is supported.
- FORCED (32'h2): Lane reversal is forced during configuration. The VIP reverses its lanes from 0-(N-1) to (N-1)-0, where N is the maximum supported link width as set using `set_link_width_values()` task.

FORCED mode is intended for testing purposes only. This mode allows you to test the DUT's lane reversal support without changing the wiring.

For example,

```
cfg.root_cfg.pcie_cfg.pl_cfg.lane_reversal_mode =
svt_pcie_pl_configuration::FORCED;
cfg.endpoint_cfg.pcie_cfg.pl_cfg.lane_reversal_mode =
svt_pcie_pl_configuration::FORCED;
```

For more information, see the SolvNetPlus article at

https://solvnet.synopsys.com/retrieve/1831170.html

2. How do I test the lane polarity inversion feature?

   The configuration parameter `invert_tx_polarity` of the `svt_pcie_pl_configuration` is used to invert polarity.

   `svt_pcie_pl_configuration::invert_tx_polarity`

   This parameter is a 32-bit variable where each bit position corresponds to a lane number in the link.

   Example code to enable polarity inversion:

```
endpoint_cfg.pcie_cfg.pl_cfg.invert_tx_polarity = 1'b1;     // Invert polarity on lane0
endpoint_cfg.pcie_cfg.pl_cfg.invert_tx_polarity = 'b1111;  // Invert polarity on lane3,
lane2, lane1 and lane0
```

   Example code to disable polarity inversion:

```
endpoint_cfg.pcie_cfg.pl_cfg.invert_tx_polarity = 1'b0;     // Invert polarity on lane0
endpoint_cfg.pcie_cfg.pl_cfg.invert_tx_polarity = 'b0000;  // Invert polarity on lane3,
lane2, lane1 and lane0
```

3. How to enable SRIS mode?

   VIP provides the following configuration parameter to enable the SRIS mode:

   Example code for configuration:

```
endpoint_cfg.port_cfg.pl_cfg.sris_mode_enabled = 1;
```

   User Guide description: Currently, this is only used to control transmission/reception of modified compliance pattern at 128/130b in SRIS and non-SRIS modes. If set to "1", then VIP will transmit/receive SRIS mode modified compliance pattern else non-SRIS one.

4.6 Order Set

1. How do I modify a TS Ordered Set?

   Use the following methods to modify a TS Ordered Set.

   - Use the `tx_ts_os_started()` callback to modify the fields of a TS Ordered set.
   - Use the `user_ts services` to modify the TS Ordered Set symbols.

   For example:

```
// Specify user tx TS enable sequence
`uvm_do_on_with(set_tx_ts1_pattern_seq,
vip_seqr.root_virt_seqr.pcie_virt_seqr.pl_seqr,
{
set_tx_ts1_pattern_seq.user_tx_ts_enable == 1'b1;
set_tx_ts1_pattern_seq.min_user_tx_ts_burst == 27;
set_tx_ts1_pattern_seq.max_user_tx_ts_burst == 28;
set_tx_ts1_pattern_seq.min_user_tx_ts_spacing == 1;
set_tx_ts1_pattern_seq.max_user_tx_ts_spacing == 1;
})


// Specify user tx TS1 with lane and link number as non-PAD
for (int lane =0; lane < total_lanes; lane++) begin
`uvm_do_on_with(set_tx_ts1_seq,
```

```
vip_seqr.root_virt_seqr.pcie_virt_seqr.pl_seqr,
{
set_tx_ts1_seq.lane_num == lane;
set_tx_ts1_seq.enable_symbol_constraint == 16'hFFF9;
set_tx_ts1_seq.symbol1_link_num == 8'h14;
set_tx_ts1_seq.symbol2_lane_num == 8'h12;
})
end
```

- Replace the TS OS with a primitive sequence

  For example,

```
// Replace TS2 with TS1 OS
// Specify user tx TS1 with lane and link number as non-PAD
for (int lane =0; lane < total_lanes; lane++) begin
`uvm_do_on_with(replace_with_ts1_seq,

vip_seqr.root_virt_seqr.pcie_virt_seqr.pl_seqr, {
replace_with_ts1_seq.lane_num == lane;
replace_with_ts1_seq.enable_symbol_constraint[15:0] == 16'hFFE9;
replace_with_ts1_seq.symbol1_link_num[7:0] == 8'h02;
replace_with_ts1_seq.symbol2_lane_num[7:0] == 8'h05;
replace_with_ts1_seq.symbol4_data_rate_identifier[7:0] == 8'h0e;
})
end
```

  For more information, see the SolvNetPlus article at

  https://solvnet.synopsys.com/retrieve/2343320.html

2. How do I configure the VIP to send Ordered Sets?

   Configure the VIP to transmit user-defined TS1 and TS2 by using the `SET_USER_TX_TS1` and `SET_USER_TX_TS2` services in the PL service class. The number of TS1 and TS2 to be transmitted are specified by the `SET_USER_TX_TS1_PATTERN` and `SET_USER_TX_TS2_PATTERN` services.

   For example:

```
// Trigger user tx TS enabled sequence
`uvm_do_on_with(set_tx_ts2_pattern_seq,vip_seqr.pcie_virt_seqr.pl_seqr,
{
set_tx_ts2_pattern_seq.user_tx_ts_enable == 1'b1;
set_tx_ts2_pattern_seq.min_user_tx_ts_burst == 1;
set_tx_ts2_pattern_seq.max_user_tx_ts_burst == 1;
set_tx_ts2_pattern_seq.min_user_tx_ts_spacing == 1;
set_tx_ts2_pattern_seq.max_user_tx_ts_spacing == 1;
})
```

```
// Trigger user tx TS2 with lane and link number set to PAD
for (int lane = 0; lane < vip_cfg.pcie_cfg.pl_cfg.link_width; lane++)
begin
`uvm_do_on_with(set_tx_ts2_seq,vip_seqr.pcie_virt_seqr.pl_seqr,
{
set_tx_ts2_seq.lane_num == lane;
set_tx_ts2_seq.enable_symbol_constraint[15:0] == 16'hFFF9;
set_tx_ts2_seq.symbol1_link_num[7:0] == 8'hF7;
set_tx_ts2_seq.symbol2_lane_num[7:0] == 8'hF7;
})
end
```

3. How do I inject error in the transmit Ordered Sets?

   Use callbacks in the PL class to inject an error in the transmit Ordered Sets.

   For example,

```
virtual function void tx_os_started (svt_pcie_plpl, svt_pcie_os transaction);
if (transaction.os_type == svt_pcie_os::SDS_OS)
begin
`svt_note("tx_sds_os_started",$sformatf("\nSDS OS of type %0d will be transmitted
on lane %0d.\n",transaction.os_type, transaction.logical_lane_num));
// Test case decision to determine how many errors to introduce.
if (error_count< 1 )
begin
if (transaction.logical_lane_num == 0)
begin
my_sds_os_exc_list.xact_exc.NO_ERROR_wt = 0;
my_sds_os_exc_list.xact_exc.INVALID_DATABYTE_wt = 1000;
end
else if(transaction.logical_lane_num == 1)
begin
my_sds_os_exc_list.xact_exc.NO_ERROR_wt = 0;
my_sds_os_exc_list.xact_exc.INVALID_DATABYTE_wt = 1000;
end
else if(transaction.logical_lane_num == 2)
begin
my_sds_os_exc_list.xact_exc.NO_ERROR_wt = 0;
my_sds_os_exc_list.xact_exc.INVALID_DATABYTE_wt = 1000;
end
else if(transaction.logical_lane_num == 3)
begin
```

```
my_sds_os_exc_list.xact_exc.NO_ERROR_wt = 0;
my_sds_os_exc_list.xact_exc.INVALID_DATABYTE_wt = 1000;
end
end
else
begin
my_sds_os_exc_list.xact_exc.NO_ERROR_wt = 100;
my_sds_os_exc_list.xact_exc.INVALID_DATABYTE_wt = 0;
end
my_sds_os_exc_list.randomized_exception = my_sds_os_exc_list.xact_exc;
my_sds_os_exc_list.setup_randomized_exception(dummy_pl_cfg,transaction);
success = my_sds_os_exc_list.num_exceptions_first_randomize();
error_count += my_sds_os_exc_list.num_exceptions;
`svt_note("pre_sds_os_xact_transmission",$sformatf(" Attaching exception
list.\n."));
$cast(transaction.exception_list, my_sds_os_exc_list.`SVT_DATA_COPY());
end
endfunction
```

4. How do I create sequences for TS Ordered Set?

   Use the `get_os()` method and the sequence constraints in the `svt_pcie_os` class to create sequences for TS Ordered Set. Extend the PL service class to create primitive sequences.

   For example:

```
// Constraints for the symbol0 field
constraint symbol0_header {
if (enable_symbol_constraint[0]) {
symbol0[8] == 1'b1;

if (os_transaction.encoding_type == svt_pcie_types::ENC_8B_10B)
symbol0[7:0] == 8'hBC;

if (os_transaction.encoding_type == svt_pcie_types::ENC_128B_130B)
symbol0[7:0] == 8'h1E;
}
symbol0[8] == 1'b1;
}

// Specify a sequence
`uvm_do_with(set_user_tx_ts1_seq,{
service_type == svt_pcie_pl_service::SET_USER_TX_TS1;
lane_num     == local::lane_num;
symbol0      == local::symbol0;
```

```
symbol1       == local::symbol1_link_num;
…
})
```

A set of primitive sequences that meet all protocol requirements is available in the following file:

$DESIGNWARe_HOME/vip/svt/pcie_svt/src/svt_pcie_pl_service_sequence_collection.sv

5. How do I delay SKP using minimum/maximum settings?

   The SKP interval transmission and reception can be controlled in the PCIE VIP through the following attributes in the svt_pcie_pl_configuration class.

   The VIP will transmit a SKP Ordered Sets based on the following settings:

   Gen1 and Gen2

   ```
   min_tx_skp_interval_in_symbol_times
   max_tx_skp_interval_in_symbol_times
   ```

   Gen3

   ```
   min_tx_skp_interval_in_blocks
   max_tx_skp_interval_in_blocks
   ```

   The VIP will check the reception of the SKP Ordered Sets from the DUT based on the following settings:

   Gen1 and Gen2

   ```
   min_rx_skp_interval_in_symbol_times
   max_rx_skp_interval_in_symbol_times
   ```

   Gen3

   ```
   min_rx_skp_interval_in_blocks
   max_rx_skp_interval_in_blocks
   ```

   The following table shows the allowed ranges and the default setting for the `skp_interval` attributes.

Table 4-2 Range and Settings

Attribute	Type	Range	Default Value	Description
MIN_TX_SKP_INTERVAL_IN_SYMBOL_TIMES	Integer	32 - large value	1180	Minimum number of symbol times before the upper phy will schedule the insertion of a SKP ordered set (2.5GT/s and 5 GT/s)
MAX_TX_SKP_INTERVAL_IN_SYMBOL_TIMES	Integer	32 - large value	1538	Maximum number of symbol times before the upper phy will schedule the insertion of a SKP ordered set (2.5GT/s and 5 GT/s)
MIN_TX_SKP_INTERVAL_IN_BLOCKS	Integer	2 - large value	375	Minimum number of blocks before the upper phy must schedule the insertion of a SKP ordered set (8GT/s)
MAX_TX_SKP_INTERVAL_IN_BLOCKS	Integer	2 - large value	375	Maximum number of blocks before the upper phy must schedule the insertion of a SKP ordered set (8GT/s)

Table 4-2 Range and Settings

MIN_RX_SKP_INTERVAL_IN_SYMBOL_TIMES	Integer	32 - large value	1180	Minimum number of symbol times before the upper phy flag an error due to the lack of a SKP ordered set (2.5GT/s and 5 GT/s)
MAX_RX_SKP_INTERVAL_IN_SYMBOL_TIMES	Integer	32 - large value	1538	Maximum number of symbol times before the upper phy will flag an error due to the lack of a SKP ordered set (2.5GT/s and 5 GT/s)
MIN_RX_SKP_INTERVAL_IN_BLOCKS	Integer	2 - large value	375	Minimum number of blocks before the upper phy flags an error due to the lack of a SKP ordered set (8GT/s)
MAX_RX_SKP_INTERVAL_IN_BLOCKS	Integer	2 - large value	375	Maximum number of blocks before the upper phy flags an error due to lack of a SKP ordered set (8GT/s)
MIN_TX_SKP_SYMBOLS_IN_ORDERED_SET	Integer	1-5	3	Minimum number of SKP symbols in an ordered set at 2.5GT/s and 5GT/s.
MAX_TX_SKP_SYMBOLS_IN_ORDERED_SET	Integer	1-5	3	Maximum number of SKP symbols in an ordered set at 2.5GT/s and 5GT/s.
MIN_TX_SKP_SYMBOLS_IN_ORDERED_SET_8G	Integer	1-5	3	Minimum number of SKP symbols in an ordered set at 8GT/s. Acceptable values: 1 – 5.
MAX_TX_SKP_SYMBOLS_IN_ORDERED_SET_8G	Integer	1-5	3	Maximum number of SKP symbols in an ordered set at 8GT/s. Acceptable values: 1 – 5.

The `min/max_tx_skp_interval_in_xxx` settings are randomized in the VIP to the minimum and maximum settings of the attributes. For example, the default setting for `min_tx_skp_interval_in_symbol_times` is 1180 symbol times. The default setting for `max_tx_skp_interval_in_symbol_times` is 1538 symbol times. The PCIE VIP will transmit the SKP OS based on these 2 settings. The SKP interval transmission will be randomized between the min and max values.

If the SKP OS interval must be set to a specific value, then set the min and max values to the same number. For example, to have the PCIE VIP transmit the SKP interval at 1275, the following setting is required.

```
//Configure min/max skip interval to a set value.
cfg.rc_cfg.pcie_cfg.pl_cfg.min_tx_skp_interval_in_symbol_times = 1275;
cfg.rc_cfg.pcie_cfg.pl_cfg.max_tx_skp_interval_in_symbol_times = 1275;
```

Similarly, for the SKP interval in blocks for gen3, the following setting is required.

```
//Configure min/max skip interval to a set block value.
cfg.rc_cfg.pcie_cfg.pl_cfg.min_tx_skp_interval_in_blocks = 372;
cfg.rc_cfg.pcie_cfg.pl_cfg.max_tx_skp_interval_in_blocks = 372;
```

For the `min/max_rx_skp_interval_in_symbol_times` and `min/max_rx_skp_interval_in_blocks`, the PCIE VIP will check for the reception of the SKP OS from the DUT. Again, the SKP interval will be checked based on the randomized minimum and maximum values. If the VIP is required to check the SKP interval reception for a particular value, then set the minimum and maximum values to the required value.

The PCIe VIP also allows for the number of SKP symbols to be included in the SKP OS. The default setting for both minimum and maximum is 3, therefore, 3 SKP symbols will be sent in the SKP OS. The SKP OS can be adjusted to send a random number of SKP symbols and set to another value such as 5 by setting the min and max numbers to be different or the same number.

6. How do I determine the number of OS received and transmitted in a particular state?

   The following attributes in the PL status class (`svt_pice_pl_status`) record the number of OS received and transmitted in a particular state.

   - `num_rx_ts1[31:0]`
   - `num_rx_ts2[31:0]`
   - `num_tx_ts1[31:0]`
   - `num_tx_ts2[31:0]`

   Usage:

   ```
   root_status.pcie_status.pl_status.lane_status[1].num_tx_ts1;
   root_status.pcie_status.pl_status.lane_status[1].num_tx_ts2;
   ```

   Usage:

   ```
   root_status.pcie_status.pl_status.lane_status[1].num_rx_ts1;
   root_status.pcie_status.pl_status.lane_status[1].num_rx_ts2;
   ```

7. How do I access the fields (symbols) of TS OS received from the DUT?

   Use the `rcvd_ts_os [$]` array of `svt_pcie_os` class objects in the PL status class (`svt_pcie_pl_status`) to access the fields of the TS Ordered Sets received from the DUT.

   For example,

   ```
   if (vip_status.pcie_status.pl_status.rcvd_ts_os[0].ts_link ==  8'hf7)
   ```

8. How do I access the status of receiver detection at a particular lane?

   Use the `receiver_detected` attribute in the PL status class to monitor the receiver detection at a particular lane.

9. How do I count the number of transmitted or received logical idles?

   Use the `num_rx_logical_idle` and `num_tx_logical_idle` variables in the PL status class (`svt_pcie_pl_status`) to count the number of transmitted and received logical Idles.

   ```
   int num_rx_logical_idle[31:0];
   int num_tx_logical_idle[31:0];
   ```

   For example,

   ```
   // Wait for LTSSM to enter the Recovery.RcvrLock  state
   wait (dut_status.pcie_status.pl_status.ltssm_state ==
   svt_pcie_types::RECOVERY_IDLE);


   // Store the total number of LOG_IDLE transmitted and received
   current_tx_idle = vip_status.pcie_status.pl_status.num_tx_logical_idle[0];
   current_rx_idle = vip_status.pcie_status.pl_status.num_rx_logical_idle[0];
   ```

10. How do I block SKP symbol transmission in the VIP?

Use the `min_tx_skp_symbols_in_ordered_set` and `max_tx_skp_symbols_in_ordered_set` attributes in the PL configuration class (`svt_pcie_pl_configuration_class`) to block SKP symbol transmission in the VIP.

For example,

```
// Configure min/max skip interval to a larger value.
cfg.rc_cfg.pcie_cfg.pl_cfg.min_tx_skp_interval_in_symbol_times = 0;
cfg.rc_cfg.pcie_cfg.pl_cfg.max_tx_skp_interval_in_symbol_times = 0;

// Configure min/max skip interval to a larger value.
cfg.rc_cfg.pcie_cfg.pl_cfg.min_tx_skp_interval_in_blocks = 0;
cfg.rc_cfg.pcie_cfg.pl_cfg.max_tx_skp_interval_in_blocks = 0;
```

For more information, see the SolvNetPlus article at

https://solvnet.synopsys.com/retrieve/2335767.html

11. How do I modify an Ordered Set transaction in the VIP?

    The `tx_ts_os_started`, `tx_os_started` and `pre_symbol_out_put` callbacks in the PL class (`svt_pcie_pl`) can be used to modify an Ordered Set transaction.

    For example,

```
virtual function void pre_symbol_out_put(svt_pcie_pl pl, svt_pcie_symbol
symbols[]);
if (error_count < 1)
begin
for ( int lane = 0 ; lane < symbols.size() ; lane = lane + 1)
begin
if (symbols[lane].symbol_event == svt_pcie_symbol::EDS_STARTED)
symbol_found_flag = 1;
end

if (symbol_found_flag == 1 && pl.status.ltssm_state == svt_pcie_types::L0)
begin
for (i=0; i< symbols.size(); i++)
begin
svt_pcie_symbol_exception_list sym_exc_list = new();
svt_pcie_symbol_exception sym_exc = new();
sym_exc.error_kind = svt_pcie_symbol_exception::CORRUPT_DATA_VALUE_ONLY;
sym_exc.corrupted_data = 8'h3c;
sym_exc.scrambler_control = svt_pcie_symbol_exception::NONE;
sym_exc_list.add_exception (sym_exc);
symbols[i].exception_list = sym_exc_list;
```

```
`svt_note("TEST: pre_symbol_out_put_callback",$sformatf("\n Corrupting EDS on
lane %d.\n", i));
end
error_count = error_count + 1;
end
end
endfunction
```

12. How do I skip Polling.Active LTSSM state in the VIP?

    Use attribute `skip_polling_active` present in the PL configuration *svt_pcie_pl_configuration.sv* class. Skipping Polling.Active helps speed up link training for cases not specifically testing the ltssm states.

13. How to corrupt the EDS token?

    You can use the following callback to corrupt the EDS token:

    ```
    pre_tx_symbols_put(svt_pcie_pl pl, svt_pcie_symbol symbols[])
    ```

    This is the last point at which you can intercept the data before it goes out of the wire. This will give you the current symbol striping (one per lane)—hence the symbols[].

    Within the callback you can check for the EDS staring on a particular symbol with:

    ```
    if(symbols[n].symbol_event == svt_pcie_symbol::EDS_STARTED)
       ...
    ```

    You can then count the byte offsets 0–3 and when on the 3rd byte you would use:

    ```
    sym_exc.error_kind = svt_pcie_symbol_exception::CORRUPT_DATA_VALUE_ONLY;  // Corrupt the
    data
       sym_exc.corrupted_data = 'h80;
       sym_exc.scrambler_control = svt_pcie_symbol_exception::NONE; // Do not corrupt
    scrambling
    ```

    The *ts.symbol_exception_via_callback_pipe_test.sv* example shows how to corrupt symbols: It is available in the unified level examples.

14. How to corrupt EIOS pattern?

    You can use the `pre_symbol_out_put` attribute of the PL callback class (`svt_pcie_pl_callback`) to corrupt the EIOS pattern.

    For example,

    ```
    virtual function void pre_symbol_out_put(svt_pcie_pl pl, svt_pcie_symbol symbols[]);
        if(symbols[0].symbol_event == svt_pcie_symbol::EIOS_STARTED) begin
          for(i=0; i< symbols.size(); i++) begin
            svt_pcie_symbol_exception_list sym_exc_list = new();
            svt_pcie_symbol_exception sym_exc = new();
            sym_exc.error_kind = svt_pcie_symbol_exception::CORRUPT_DATA_VALUE_ONLY;
            sym_exc.corrupted_data = 'h3c;
            sym_exc.scrambler_control = svt_pcie_symbol_exception::NONE;
            sym_exc_list.add_exception(sym_exc);
            symbols[i].exception_list = sym_exc_list;
             `uvm_note("TEST: pre_symbol_out_put_callback",$sformatf("\n Corrupting EIOS
             .\n"))

          end
        end
    ```

```
endfunction
```

You can see the corruption from simulation log.

This is the code from simulation log of a back-to-back example:

```
UVM_INFO ./vip/src/sverilog/vcs/svt_pcie_pl_proxy.sv(4040) @ 19588856.40 ps:
uvm_test_top.env.rc_env.rc_agent.port0.pl0 [process_exception_list_to_get_symbol_errors]
Exception with error_kind = 10 and corrupted_data = 3cH and scrambler_control = 0H.
```

15. How do I enable EIEOS pattern for 16GT?

    You can use the following attributes in the PHY layer configuration class (`svt_pcie_pl_configuration`) to enable EIEOS pattern for 16GT.

    Example code to enable EIEOS pattern:

    ```
    root_cfg.pcie_cfg.pl_cfg.enable_eieos_16g_0000_ffff=1
    endpoint_cfg.pcie_cfg.pl_cfg.enable_eieos_16g_0000_ffff=1
    ```

16. How to control the number of garbage symbols sent after an EIOS is transmitted?

    The PCIe SVT VIP model has a configuration parameter that controls the number of garbage symbols sent after an EIOS is transmitted. The default value is 6.

    ```
    svt_pcie_pl_configuration::tx_shutdown_latency = 6
    ```

    Example code to set `tx_shutdown_latency` value:

    ```
    cust_cfg.root_cfg.pcie_cfg.pl_cfg.tx_shutdown_latency=0;
    cust_cfg.endpoint_cfg.pcie_cfg.pl_cfg.tx_shutdown_latency=0;
    ```

17. How to truncate the number of symbols in the last EIOS before electrical idle when using 128b130b encoding?

    The `svt_pcie_pl_configuration` class has two variables that control the number of symbols in the last EIOS.

    ```
    min_num_symbols_in_last_eios_before_elec_idle_128b130b
    max_num_symbols_in_last_eios_before_elec_idle_128b130b
    ```

    The minimum value is 2 and the maximum allowed value is 16.

    Example code:

    ```
    cfg.pcie_cfg.pl_cfg.min_num_symbols_in_last_eios_before_elec_idle_128b130b=2;
    cfg.pcie_cfg.pl_cfg. max_num_symbols_in_last_eios_before_elec_idle_128b130b=16;
    ```

18. How to set different values of FTS using PCIe VIP?

    The PCIe VIP provides the following `pcie_pl_configuration` configuration attribute to set the values of FTS:

- TX side

  ```
  cfg.pcie_cfg.pl_cfg.num_tx_fts_required_2_5g = < Values>;
  cfg.pcie_cfg.pl_cfg.num_tx_fts_required_5g = < Values>;
  cfg.pcie_cfg.pl_cfg.num_tx_fts_required_8g = <Values>;
  ```

- RX side

  ```
  cfg.pcie_cfg.pl_cfg.num_rx_fts_required_2_5g = <Values>;
  cfg.pcie_cfg.pl_cfg.num_rx_fts_required_5g = <Values>;
  cfg.pcie_cfg.pl_cfg.num_rx_fts_required_8g = <Values>;
  ```

  For more details about the these attributes, see the HTML class reference documentation:

https://solvnet.synopsys.com/dow_retrieve/latest/verification_compiler/doc/nvme_svt_uvm_class_reference/html/class_svt_pcie_pl_configuration.html

19. How to set compliance receive bit in TS Order set?

    The `set_ts_compliance_receive` PL configuration attribute is used to set the compliance receive bit of the outgoing TS1 OS.

- When set to 1, outgoing training sets will have the compliance receive bit set.
- When set to 0, compliance receive bit in outgoing training sets is not set.

Example code:

```
vip_cfg.pcie_cfg.pl_cfg.set_ts_compliance_receive=1;
```

20. How to control EIEOS before FTS?

    Set the following PL configuration variable to specify the minimum/maximum number of EIEOS ordered sets transmitted before transmitting FTS. The VIP will randomly pick a value within the range such that `min<=value<=max`.

- `min_num_tx_eieos_before_fts`
- `max_num_tx_eieos_before_fts`
- `min_num_rx_eieos_before_fts`
- `max_num_rx_eieos_before_fts`

Example code:

```
vip_cfg.pcie_cfg.pl_cfg.min_num_tx_eieos_before_fts = 6;
```

21. How to disable VIP model to insert/remove SKP symbols to avoid Rx lane skew error?

    The `ENABLE_RX_ELASTIC_BUFFER` switch in the PCS enables the receive elastic buffer. This will cause the model to insert/remove skip symbols. Setting this model configuration to 0 will stop VIP model to insert/remove skew. The default value of this switch is set to 1.

    Example code:

```
defparam test_top.root0.port0.pcs0.ENABLE_RX_ELASTIC_BUFFER = 0;
defparam test_top.endpoint0.port0.pcs0.ENABLE_RX_ELASTIC_BUFFER = 0;
```

22. How to control tx_elecidle assert timing after transmission of EIOS?

    The `tx_shutdown_latency` control sets how many symbols of garbage are sent after EIOS. A DUT should ignore everything after the EIOS. The latency can be set to 0 so that no garbage data will be sent after EIOS.

```
cfg.pcie_cfg.pl_cfg. tx_shutdown_latency=0;
```

23. How to control electrical idle time of transmitter before transitioning to another state?

    The tx_idle_min_timer_ns control variable specifies the minimum time in ns spent in Electrical Idle. The time a Tx must spend in Electrical Idle before transitioning to another state.

```
cfg.pcie_cfg.pl_cfg. tx_idle_min_timer_ns=<value>
```

24. How to disable SKP order set checking?

    In some cases, where the SKP order sets are less than or greater than 3 SKP symbols, VIP will issue the following warning:

```
UVM_WARNING: uvm_test_top.env.ep_env.ep_agent.port0.pl0
[register_fail:ACTIVE_PL_LANE_OS:SKP_OS:ordered_set_checker_undersize_skip_0] :
```

```
Detected undersize SKP ordered set with 1 COM and 1 SKP symbols. Expecting 3 SKP
symbols.
```

To avoid this warning and to disable the SKP checking, set the following configuration attribute:

```
cfg.rc_cfg.pcie_cfg.pl_cfg.enable_relaxed_skp_checking = 1;
```

25. How to increase or decrease the ctrl skp ordered sets or standard skp ordered sets?

    You can use the following configuration attribute to increase or decrease the skip symbol ordered sets:

    ```
    rand int unsigned  attribute
     svt_pcie_pl_configuration::min_tx_skp_symbols_in_ordered_set_8g
    ```

    Usage:

    ```
    <vip_cfg>.pcie_cfg.pl_cfg.max_tx_skp_symbols_in_ordered_set_8g = 3;
    <vip_cfg>.pcie_cfg.pl_cfg.min_tx_skp_symbols_in_ordered_set_8g = 3;
    ```

    Note

    - `<vip_cfg>` can be declared by the user.
    - `<vip_cfg>` is an object of `svt_pcie_device_configuration` class.

    Set the values based on requirement. You can set the values from 1 to 5. Here, in the example, the configuration values are set to 3 which indicates 12 skip symbols with 1 skip end symbol and 3 other symbols.

4.7 LTSSM

1. How do I view changes in the LTSSM states?

   Changes in the LTSSM states can be viewed in the following ways:

   - Specify the +uvm_set_verbosity option with `UVM_HIGH` verbosity in the command line to enable LTSSM state information to be displayed during simulation.
   - Set the `enable_symbol_logging` attribute in the PL configuration class to enable symbol logging. With symbol logging enabled, a symbol log file is generated in the same directory where the simulation is run. Changes in the LTSSM states are recorded for each received and transmitted symbol from the VIP.

     For example,

     ```
     sim timerx_lane ->  0   1   2   3       0   1    2    3 <-tx_lane LTSSM State
     12232:  root0       z   z   z   z   |   z   z    z    z        ->Det.Active
     12236:  root0       z   z   z   z   |   z   z    z    z        ->Det.Active
     12240:  root0       z    z   z   z    | 00 00 00 00        ->Poll.Active
     12244:  root0       z    z   z   z    | 00 00 00 00        ->Poll.Config
     12248:  root0       z   z   z  z    | COM COM COM COM   ->Poll.Config
     12252:  root0       z    z   z   z    | PAD PAD  PAD  PAD  ->Poll.Config
     12256:  root0       z    z   z   z    | PAD PAD  PAD  PAD ->Poll.Config
     ```

   - View the PL ASCII signals, `ascii_ltssm_rx_state` and `ascii_ltssm_tx_state` on a waveform viewer.

2. How do I access the previous LTSSM state of the VIP?

   The `last_ltssm_state` variable in the `svt_pcie_pl_status` class stores the previous LTSSM state of the VIP.

For example,

```
if (vip_status.pcie_status.pl_status.last_ltssm_state ==
svt_pcie_types::RECOVERY_RCVRLOCK)
begin …
```

3. How do I access the current LTSSM state of the DUT?

   Use cross module reference to map the LTSSM state of the DUT and the `dut_ltssm_state` variable in the PL status class. Once the mapping is established, the LTSSM state of the DUT can be access through the PL status class.

   For example,

```
if (dut_status.pcie_status.pl_status.ltssm_state ==
svt_pcie_types::RECOVERY_RCVRLOCK)
begin …
```

4. How do I monitor LTSSM state timeouts?

   The `svt_timer` and `svt_pcie_timer` classes include features to start a timer for a user-specified duration and to monitor the timer until timeout occurs. Timers can be controlled by the LTSSM states of the DUT if the state variable is mapped to the `dut_ltssm_state` variable in the PL status class.

   For example,

```
svt_timer config_lw_start_timer;
…
config_lw_start_timer.start_timer
(vip_cfg.pcie_cfg.pl_cfg.configuration_linkwidth_start_timeout_ns,"Configuration
Linkwidth Start Timer Started",1,0);
```

5. How do I configure the VIP for directed state transitions?

   If a VIP-VIP environment is used, services from the `svt_pcie_dl_service` and `svt_pcie_pl_service` classes can be used to configure the VIP for directed state transitions:

   For example,

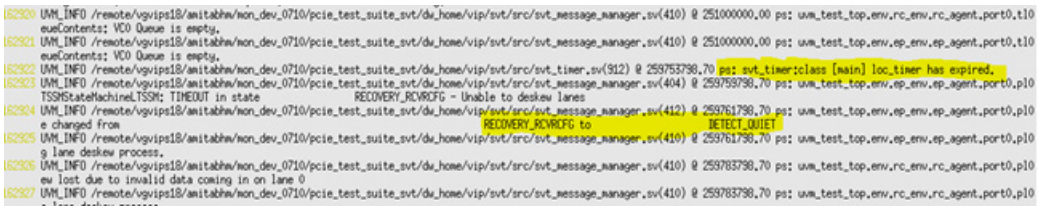
```
INITIATE_ASPM_L0S_ENTRY//< Initiates VIP to enter ASPM Tx L0s low power state.
INITIATE_ASPM_L1_ENTRY//< Initiates VIP to enter ASPM L1 low power state.
INITIATE_PM_L1_ENTRY//< Initiates VIP to enter PM L1 low power state.
INITIATE_PM_L23_ENTRY//< Initiates VIP to enter PM L2/L3 low power states.
INITIATE_ASPM_EXIT//< Initiates VIP to transition back to L0 from ASPM low
//power state.
INITIATE_PM_EXIT//< Initiates VIP to transition back to L0 from PM Low
//power state
SET_PHY_ENABLE//< When clear, LTSSM will attempt to enter the
//DISABLED state through the exchange of TS.
INITIATE_LOOPBACK//< Instructs the VIP to enter loopback as a loopback
//master.
// This sequence Initiates entry to ASPM Low Power state.
svt_pcie_dl_service_initiate_aspm_l1_entry_sequence aspm_l1_entry;
```

```
// This sequence Initiates exit to ASPM Low Power state.
svt_pcie_dl_service_initiate_aspm_exit_sequence aspm_l1_exit;
// Initializing entry to l1 Sequence
`uvm_do_on(entry_to_l1,p_sequencer.endpoint_virt_seqr.pcie_virt_seqr.dl_seqr);
// Initializing exit to l1 Sequence
`uvm_do_on(exit_to_l1,p_sequencer.endpoint_virt_seqr.pcie_virt_seqr.dl_seqr);
```

6. Are state transitions displayed before or after the timeout value specified for a state transition timeout?

   In the simulation log file, state transitions are displayed after the timeout valued has reached.

Figure 4-1 Simulation Log File



7. What is a potential cause of endless Link Training in the `CONFIGURATION_LINK_WIDTH_START` state?

   The VIP is configured as an Endpoint and the DUT is the PCIe upstream device.

   The VIP uses the link_number attribute to qualify the transition from the `CONFIGURATION_LINK_WIDTH_START` state to the `CONFIGURATION_LINK_WIDTH_ACCEPT` state.

   When the upstream device uses a non-zero link number in the TS1 Order Sets and the VIP continues to send TS1 Ordered Sets with the link number symbol PAD, the mismatched link numbers prevents the state transition. As a result, Link Training continues endlessly in the `CONFIGURATION_LINK_WIDTH_START` state.

   Use the `link_number` attribute of the PL configuration class (`svt_pcie_pl_configuration`) to specify a link number value that matches the link number value of the upstream device.

   For example,

   ```
   endpoint_cfg.port_cfg.pl_cfg.link_number = 4;
   ```

8. What is a potential cause for the Endpoint VIP to timeout in the `RECOVERY_EQUALIZATION_3` state?

   A common reason for an Endpoint VIP to timeout in the `RECOVERY_EQUALIZATION_3` state is an unknown value (x) on the `attached_powerdown` signal. This unknown value can impact the phystatus timers and causes the VIP to timeout.

   `attached_powerdown`

   This is an input PIPE signal used to power up or power down the transceiver.

   The power states are:

   - P0 (normal operation)
   - P0s (low time recovery latency)
   - P1 (longer recovery time latency)
   - P2 (lowest recovery time latency)

   phystatus

This is an active high output signal used to communicate completion of several PHY functions including power management state transitions, rate change, and receiver detection. When this signal transitions during entry and exit from the P2 power state and the PCLK is not running, then the signaling is asynchronous.

9. How do I enable the link to transmit DLLPs in the LTSSM L0 state?

   The transmission of DLLPs in the LTSSM L0 state requires the data link layer and the PHY layer to be enabled. Use the `svt_pcie_dl_service_set_link_en_sequence` sequence class to enable the data link layer. The PHY layer is enabled by default. If the PHY layer is disabled in a test, then use the `svt_pcie_pl_service_set_phy_en_sequence` sequence class to enable the PHY layer.

   For example,

```
//Enable the PHY layer in each component
fork
`uvm_do_on_with(vip_pl_link_en_seq,vip_seqr.pcie_virt_seqr.pl_seqr, {
vip_pl_link_en_seq.phy_enable == 1'b1; })
`uvm_do_on_with(dut_pl_link_en_seq,dut_seqr.pcie_virt_seqr.pl_seqr, {
dut_pl_link_en_seq.phy_enable == 1'b1; })
join

//Trigger DL link up sequence
fork
`uvm_do_on_with(vip_dl_link_en_seq,vip_seqr.pcie_virt_seqr.dl_seqr, {
vip_dl_link_en_seq.enable == 1'b1; })
`uvm_do_on_with(dut_dl_link_en_seq,dut_seqr.pcie_virt_seqr.dl_seqr, {
dut_dl_link_en_seq.enable == 1'b1; })
join

//Wait until LTSSM is in L0
wait(vip_status.pcie_status.pl_status.ltssm_state == svt_pcie_types::L0);
`svt_note("body",$sformatf("Current state = %s, \n",
dut_status.pcie_status.pl_status.ltssm_state));
```

10. What is a potential cause for the VIP not entering Electrical Idle based on the request from the DUT?

    In the L2 power state, an endpoint transmits a `PM_ENTER_L23` and the VIP responds with a `PM_REQUEST_ACK`. After the endpoint receives the `PM_REQUEST_ACK`, it transmits an EIOS. The VIP continues to transmit a `PM_REQUEST_ACK` and does not transmit an EIOS.

    A potential reason for the above behavior is that the DUT does not transmit an EIOS before entering an electrical idle. This scenario causes the VIP to continue transmitting `PM_REQUEST_ACK` for the power management DLLPs and not to enter the L0s power state.

11. How to increase the time of VIP in `loopback.entry`, if DUT is taking more time to transmit TS1 OS in `loopback.entry`.

    There are two configuration attributes of `svt_pcie_pl_configuration` class which can be used to update the time of VIP in `loopback.entry` before entering the `loopback.active` or Detect.

- `loopback_master_enter_compliance_tx_ts1_timeout_ns`:

  Time before the loopback master enters `loopback.active` if no TS1s are received with the enter compliance bit set.

- `loopback_master_no_enter_compliance_tx_ts1_timeout_ns`:

  Time before the loopback master enters detect if no TS1s are received with the enter compliance bit clear.

  Example code:

  ```
  cfg.pcie_cfg.pl_cfg.loopback_master_enter_compliance_tx_ts1_timeout_ns = 10;
  ```

12. How can `pipe_clk` be stopped when VIP is effectively unplugged?

    To turn off PCLK when VIP is effectively unplugged, you can use this PL configuration attribute:

    ```
    rand bit turn_off_pclk_in_hot_plug_unplug_mode = 1;
    ```

    Default value of this attribute is 0.

- 1 – PCLK turned off in `HOT_PLUG_UNPLUG` mode
- 0 – PCLK not turned off in `HOT_PLUG_UNPLUG` mode

  The control applies to the SPIPE VIP running PCLK in output mode and the MPIPE VIP running PCLK in input mode when using `HOT_PLUG_UNPLUG`.

Note This is not applicable for the SPIPE VIP when PCLK is used in input mode and for the MPIPE VIP when PCLK is used in output mode.

13. How can I set up the PCIe VIP to use the full timeout values for all the LTSSM states specified in the PCIe specification?

    The VIP has a function in the `svt_pcie_pl_configuration` class that will set all LTSSM timeout values to the values stated in the specification.

    ```
    extern function void set_full_ltssm_state_timeout_values();
    ```

    Example:

    ```
    cfg.pcie_cfg.pl_cfg.set_full_ltssm_state_timeout_values();
    ```

14. How to forcefully exit L0 state regardless of any packet activity on the link?

    PCIe VIP provides a configuration variable from PL configuration `allow_l0_exit_truncate_tx_packet` which can be set to 0, so that VIP will exit L0 state once link is idle.

    ```
    rand bit allow_l0_exit_truncate_tx_packet =
    `SVT_PCIE_ALLOW_L0_EXIT_TRUNCATE_TX_PACKET_DEFAULT;
    ```

    Example code:

    ```
    cfg.rc_cfg.pcie_cfg.pl_cfg.allow_l0_exit_truncate_tx_packet=0;
    ```

    When set to 0 the LTSSM will exit L0 only when the transmitter is idle and not sending a TLP/DLLP.

    When set to 1 the LTSSM will exit L0 regardless of whether or not transmitter is idle.

15. How to configure PCIe VIP to transition to L0 state after framing error detected in the `Cfg.Idle` state?

The `enable_framing_error_in_config_idle_and_recovery_idle_check` attribute in `svt_pcie_dut_capabilities` checks for the optional feature.

This attribute must be enabled from the test itself if a DUT chooses to recover through L0 state.

By default, this attribute is set to zero causing LTSSM to move from `Cfg.Idle` to `recovery_idle`.

16. How to adjust tolerance timeout in LTSSM states?

    Set the controllable tolerance value for all LTSSM time outs. You must provide the percentage of the tolerance limit (maximum allowable tolerance is +50% as per specification).

    This is how you can control the `ltssm_timeout_tolerance_percentage` from the test case.

    Example:

```
uvm_config_db#(int)::set(this,"*", "ltssm_timeout_tolerance_percentage",50);
```

17. In lane turnoff scenario, should the VIP keep `Tx_Compliance` signal high until it goes to `Detect` state (P1 Power state)?

    PHY should ignore all signaling from the MAC until it is turned on again, so it is not required to keep `tx_compliance` asserted.

    If there is too much dependency/constraint on PHY, then set the `assert_turn_off_signaling_continuously` PL configuration variable to 1. This keeps the `Tx_compliance` signal asserted.

18. VIP does not change the state from Detect to Polling.

    Following are the possibilities for this scenario:

- Scenario 1: Check DUT link width at PIPE interface. VIP's initial link width configuration should match with DUT's. Ensure that it is correctly done using PL configuration method `set_link_width_values()`.

  For example,

```
cust_cfg.root_cfg.pcie_cfg.pl_cfg.set_link_width_values(4);
```

- Scenario 2: Check if the PCLK rate, width signal values correctly correspond to the actual clock rate and bus width.
- Scenario 3: Check if `txdetectrx` is asserted after PHY asserts `Phystatus` for one clock cycle and drives `rxstatus` set to 011b(if receiver is present).
- Scenario 4: Check timeout interval for the `Detect.Quiet` state and `Detect.Active` state.
- Scenario 5: Check if `Phystatus` is deasserted after MAC stopped holding the PHY in reset.

4.8 Equalization

1. How to set symbol 7 of TS2 while in `Recovery.RcvrCfg` to communicate Tx preset and Rx preset hint values from RC?

   Configure following variables in test on per lane basis.

   Example:

```
for(int lane=0; lane< vip_cfg.pcie_cfg.pl_cfg.link_width; lane++) begin
  vip_cfg.pcie_cfg.pl_cfg.upstream_receiver_preset_hint_16G[lane] = 0;
  vip_cfg.pcie_cfg.pl_cfg.downstream_receiver_preset_hint_16G[lane] = 0;
  vip_cfg.pcie_cfg.pl_cfg.upstream_preset_value_16g[lane] = 0;
  vip_cfg.pcie_cfg.pl_cfg.downstream_preset_value_16g[lane] = 0;
end
```

2. How to set symbol 7 and symbol 8 of TS1 while in `Recovery.Equalization` Phase0 of EP/Phase1 of RC to communicate FS/LF values to link partner?

   Configure the following variables in test on per lane basis.

   Example:

```
for(int i=0; i< vip_cfg.pcie_cfg.pl_cfg.link_width; i++) begin
  vip_cfg.pcie_cfg.pl_cfg.lf_value_16g[i] = 24;
  vip_cfg.pcie_cfg.pl_cfg. fs_value_16g[i] = 48;
end
```

3. How to set symbol 6 of TS1 while in `Recovery.Equalization` Phase2 of EP /Phase3 of RC to communicate new preset request to link partner?

   VIP transmits new preset requests in Phase2 EP /Phase3 RC by using `svt_pcie_pl_service_queue_eq_tx_request_preset_coeff` PL service request. But queue the preset/Coeff request before entering Phase2/Phase3(means in Phase1/Phase2) so that VIP will send these requests once it entered Phace2/Phase3.

   Example:

```
svt_pcie_pl_service_queue_eq_tx_request_preset_coeff preset_req;
for(int i=0; i< vip_cfg.pcie_cfg.pl_cfg.link_width; i++) begin
      preset_req =
svt_pcie_pl_service_queue_eq_tx_request_preset_coeff::type_id::create("preset_req");
     preset_req.eq_lane_num = lane;
      preset_req.preset_valid = 1;//use preset bit 1
      preset_req.preset_value = 2;//preset value 2
      preset_req.expect_reject = 0; //If we configure illegal preset(10 to 15) in
preset_value then VIP flags warning so this warning we can aviod by setting this
variable 1 as warning is expected.
      preset_req.start(p_sequencer.endpoint_virt_seqr.pcie_virt_seqr.pl_seqr );
    end.
```

4. How to set symbol 6, symbol 7, symbol 8 and symbol 9 of TS1 while in Recovery. Equalization Phase2 of EP /Phase3 of RC to communicate new coefficient requests to link partner?

   VIP transmits new coefficient requests in Phase2 EP DUT/Phase3 RC DUT by using `svt_pcie_pl_service_queue_eq_tx_request_preset_coeff` PL service request.

   Example:

```
svt_pcie_pl_service_queue_eq_tx_request_preset_coeff coeff_req;
for(int i=0; i< vip_cfg.pcie_cfg.pl_cfg.link_width; i++) begin
      coeff_req =
svt_pcie_pl_service_queue_eq_tx_request_preset_coeff::type_id::create("coeff_req");
        coeff_req.eq_lane_num == lane ;
        coeff_req.preset_valid == 1'b0 ; //use preset bit 0
        coeff_req.precursor_coeff == test_eq_tx_precursor_coeff[lane];
        coeff_req.cursor_coeff == test_eq_tx_cursor_coeff[lane];
        coeff_req.postcursor_coeff ==  test_eq_tx_postcursor_coeff[lane];
        coeff_req.expect_reject = 0; //If VIP transmits illegal coeff then VIP flag
warning so if user set this variable as 1 then VIP will not flag warning.
        coeff_req.start(p_sequencer.endpoint_virt_seqr.pcie_virt_seqr.pl_seqr );
    end
```

5. Is there any provision in VIP to reject incoming new Preset/coefficient requests sent by EP DUT in Phase2/RC DUT in Phase3?

   You can use the following configuration variable in VIP PL configuration class. You can set this configuration in the test so that VIP will reject incoming request from the DUT.

Example:

```
/* Configure VIP to enable reject received coefficient request from link partner */
vip_cfg.pcie_cfg.pl_cfg.reject_preset_coefficient_request = 32'hffffffff;
```

6. How to adjust evaluation delay of `Recovery. Equalization` Phase2 of EP/Phase3 of RC VIP?

   The following configuration variables are available in VIP PL configuration class. You can configure desired values.

   Represents time in NS to evaluate attached transmitter setting changes during equalization Phase2/Phase3. VIP picks random value between `min_eq_evaluation_delay` and `max_eq_evaluation_delay`.

   Example:

```
for(int i=0; i< vip_cfg.pcie_cfg.pl_cfg.link_width; i++) begin
  vip_cfg.pcie_cfg.pl_cfg. min_eq_evaluation_delay [i] = 100;
  vip_cfg.pcie_cfg.pl_cfg. max_eq_evaluation_delay [i] = 1000;
end
```

7. How to adjust response delay for preset/coefficient of Recovery. Equalization Phase2 of EP/Phase3 of RC VIP?

   The following configuration variables are available in VIP PL configuration class. You can configure desired values.

   Represents amount of time VIP takes to either accept or reject a received preset/coefficient request. VIP picks random value between `min_eq_preset_coeff_validation_dela` and `max_eq_preset_coeff_validation_delay`.

   Example:

```
for(int i=0; i< vip_cfg.pcie_cfg.pl_cfg.link_width; i++) begin
  vip_cfg.pcie_cfg.pl_cfg.min_eq_preset_coeff_validation_delay [i] = 50;
  vip_cfg.pcie_cfg.pl_cfg.max_eq_preset_coeff_validation_delay [i] = 100;
end
```

8. How to enable EQ Phases (Phase0/Phase1/Phase2/Phase3) in VIP?

   The following configuration variable is preset in VIP PL configuration class. You can configure desired values.

   Specifies highest enabled equalization phase.

- When set to 1, enables equalization phase0 and phase 1.
- When set to 3, enables equalization phases 0, 1, 2, and 3.

   Example:

```
cust_cfg.root_cfg.pcie_cfg.pl_cfg.highest_enabled_equalization_phase = 3;
cust_cfg.endpoint_cfg.pcie_cfg.pl_cfg.highest_enabled_equalization_phase = 3;
```

9. How to set “Reset EIEOS” count bit in Phase2 of EP /Phase3 of RC transmit TS1 VIP?

   The following configuration variable is preset in VIP PL configuration class. You can configure desired values.

   When transmitting TS1s at 8GT/s speeds, and this bit is set to 1'b1, it forces VIP to set `reset_eieos_interval_count` bit in transmitted TS1s. Each lane can be set individually, for example to set this bit on lanes 0–3 only, set this to 32'h0000_000f.

   Example:

```
cust_cfg.root_cfg.pcie_cfg.pl_cfg.tx_ts1_reset_eieos_interval_count_bit = 32'h0000_000f;
```

10. How to set “Reset EIEOS” interval in Phase2 of RC /Phase3 of EP?

    The following configuration variable is present in VIP PL configuration class. You can configure desired values.

    During equalization phase 2 and 3, when the reset EIEOS interval bit is set on incoming TS1 this value controls the interval at which EIEOS are transmitted.

    Example: As per specification VIP transmits EIEOS after every 32 TS1/TS2 OS but in this case VIP transmits EIEOS after every 65536 TS1/TS2 OS.

    ```
    cust_cfg.root_cfg.pcie_cfg.pl_cfg.eq_rx_reset_eieos_interval = 65536;
    ```

11. Whether VIP supports redo equalization?

    Yes, VIP supports redo equalization. VIP performs redo equalization using the following PL service request `svt_pcie_pl_service_request_equalization_sequence` `request_equalization`.

    Example:

    There are two possible situations:

    i. RC is initiating redo equalization: Use above mentioned PL service request from RC VIP.

    ii.EP is initiating redo equalization: You must invoke different preset/coefficient values (these values must not be equal to RC Phase2 concluded values) in TS1 in Recovery.RcvrLock from RC VIP so that EP receives these different preset and coefficient values in 8 consecutive TS1 and then EP transmits TS2 in Recovery.RcvrCfg by setting request equalization bit 1 in transmit TS1 to RC.

    ```
    request_equalization =
    svt_pcie_pl_service_request_equalization_sequence::type_id::create("request_equalization
    ");
      request_equalization.start(p_sequencer.root_virt_seqr.pcie_virt_seqr.pl_seqr );
    ```

12. Is there any check available in VIP to check received coefficients legality? If yes, then how to enable?

    The following configuration variable is present in VIP PL configuration class. You can enable equalization coefficient rule checks.

    Phase2: EP transmits presets/coefficient request to RC

    Phase3: RC transmits presets/coefficient request to EP

    You must configure following variables based on EP DUT/RC DUT

    Example:

    ```
    cust_cfg.root_cfg.pcie_cfg.pl_cfg.enable_equalization_verification_mode = 1;
    cust_cfg.root_cfg.pcie_cfg.pl_cfg.enable_equalization_coefficients_checks = 1;
    ```

13. How to configure VIP to respond with coefficients to preset request from link partner in Phase2 of EP/Phase3 of RC?

    The following configuration variables are present in VIP PL configuration class. You can configure desired values.

    A bit mapped variable where bit N corresponds to entry N in `preset_to_coefficients_mapping_table_16g`. When bit is set to 1'b1, it indicates that corresponding entry in `preset_to_coefficients_mapping_table` is valid for 16G. Need 0 through 10 valid per PIPE 4.x specification.

    Example:

```
cust_cfg.root_cfg.pcie_cfg.pl_cfg. preset_to_coefficients_mapping_entry_valid_16g =
16'h7FF;
```

Preset to coefficients mapping table is used to map received preset requests to coefficients for use in local transmitter settings. For a passive VIP, the table must be programmed exactly like preset table of the link partner of monitored device.

The table is indexed by a preset value.

Mapping table stores coefficients values packed in the following format.

```
{ postcursor_coeff, cursor_coeff, precursor_coeff }
```

Default value = { 6'h0c, 6'h24, 6'h00} == 18'h0c900. For 16G speed only.

Example: Following different coefficients are mapped for different preset values based on local FS/LF values, here assumed local FS='d48 and local LF='d24.

```
cust_cfg.root_cfg.pcie_cfg.pl_cfg.preset_to_coefficients_mapping_table_16g[0]  =
{6'h0C,6'h24,6'h00};
cust_cfg.root_cfg.pcie_cfg.pl_cfg.preset_to_coefficients_mapping_table_16g[1]  =
{6'h00,6'h28,6'h08};
cust_cfg.root_cfg.pcie_cfg.pl_cfg.preset_to_coefficients_mapping_table_16g[2]  =
{6'h0C,6'h24,6'h00};
cust_cfg.root_cfg.pcie_cfg.pl_cfg.preset_to_coefficients_mapping_table_16g[3]  =
{6'h0C,6'h24,6'h00};
cust_cfg.root_cfg.pcie_cfg.pl_cfg.preset_to_coefficients_mapping_table_16g[4]  =
{6'h0C,6'h24,6'h00};
cust_cfg.root_cfg.pcie_cfg.pl_cfg.preset_to_coefficients_mapping_table_16g[5]  =
{6'h0C,6'h24,6'h00};
cust_cfg.root_cfg.pcie_cfg.pl_cfg.preset_to_coefficients_mapping_table_16g[6]  =
{6'h0C,6'h24,6'h00};
cust_cfg.root_cfg.pcie_cfg.pl_cfg.preset_to_coefficients_mapping_table_16g[7]  =
{6'h0C,6'h24,6'h00};
cust_cfg.root_cfg.pcie_cfg.pl_cfg.preset_to_coefficients_mapping_table_16g[8]  =
{6'h0C,6'h24,6'h00};
cust_cfg.root_cfg.pcie_cfg.pl_cfg.preset_to_coefficients_mapping_table_16g[9]  =
{6'h00,6'h28,6'h08};
cust_cfg.root_cfg.pcie_cfg.pl_cfg.preset_to_coefficients_mapping_table_16g[10] =
{6'h0C,6'h24,6'h00};
```

14. How to configure VIP to check received coefficients are same as DUT transmitted coefficients in response to preset request in Phase2 of EP/Phase3 of RC?

    The following configuration variables are present in VIP PL configuration class. You can configure desired values.

    A bit mapped variable where bit N corresponds to entry N in `expected_preset_to_coefficients_mapping_entry_enable_16g`.

    When bit is set to 1'b1, it indicates that corresponding entry in `expected_preset_to_coefficients_mapping_entry_enable_16g` is valid and VIP should check incoming TS1s to show the correct coefficient values when a particular preset has been requested. For 16G speed only.

    Example:

    ```
    cust_cfg.root_cfg.pcie_cfg.pl_cfg.expected_preset_to_coefficients_mapping_entry_enable_1
    6g = 16'h7FF;
    ```

    Preset to coefficients mapping table used to verify DUT's preset to coefficient mappings. The table must be programmed exactly like the DUT's preset table.

For a passive VIP, the table should be programmed exactly like the monitored device's preset table.

The table is indexed by a preset value.

Mapping table stores coefficients values packed in the following format.

```
{ postcursor_coeff, cursor_coeff, precursor_coeff }
```

Default value = { 6'h0c, 6'h24, 6'h00} == 18'h0c900. For 8G only.

```
cfg.rc_cfg.pcie_cfg.pl_cfg.expected_preset_to_coefficients_mapping_entry_enable_16g[0]
= {6'h0C,6'h24,6'h00};
cfg.rc_cfg.pcie_cfg.pl_cfg.expected_preset_to_coefficients_mapping_entry_enable_16g[1]
= {6'h00,6'h28,6'h08};
cfg.rc_cfg.pcie_cfg.pl_cfg.expected_preset_to_coefficients_mapping_entry_enable_16g[2]
= {6'h0C,6'h24,6'h00};
cfg.rc_cfg.pcie_cfg.pl_cfg.expected_preset_to_coefficients_mapping_entry_enable_16g[3]
= {6'h0C,6'h24,6'h00};
cfg.rc_cfg.pcie_cfg.pl_cfg.expected_preset_to_coefficients_mapping_entry_enable_16g[4]
= {6'h0C,6'h24,6'h00};
cfg.rc_cfg.pcie_cfg.pl_cfg.expected_preset_to_coefficients_mapping_entry_enable_16g[5]
= {6'h0C,6'h24,6'h00};
cfg.rc_cfg.pcie_cfg.pl_cfg.expected_preset_to_coefficients_mapping_entry_enable_16g[6]
= {6'h0C,6'h24,6'h00};
cfg.rc_cfg.pcie_cfg.pl_cfg.expected_preset_to_coefficients_mapping_entry_enable_16g[7]
= {6'h0C,6'h24,6'h00};
cfg.rc_cfg.pcie_cfg.pl_cfg.expected_preset_to_coefficients_mapping_entry_enable_16g[8]
= {6'h0C,6'h24,6'h00};
cfg.rc_cfg.pcie_cfg.pl_cfg.expected_preset_to_coefficients_mapping_entry_enable_16g[9]
= {6'h00,6'h28,6'h08};
cfg.rc_cfg.pcie_cfg.pl_cfg.expected_preset_to_coefficients_mapping_entry_enable_16g[10]
= {6'h0C,6'h24,6'h00};
```

15. How to configure user-defined DUT timeout values in Phase0/Phase1/Phase2 and Phase3?

    The following configuration variables are present in VIP PL configuration class. You can configure desired values.

    Example: Timeout in NS

    ```
    cust_cfg.root_cfg.pcie_cfg.pl_cfg.upstream_lanes_recovery_eq_phase0_timeout_ns = 12000;
    cust_cfg.root_cfg.pcie_cfg.pl_cfg.upstream_lanes_recovery_eq_phase1_timeout_ns = 12000;
    cust_cfg.root_cfg.pcie_cfg.pl_cfg.upstream_lanes_recovery_eq_phase2_timeout_ns = 24000;
    cust_cfg.root_cfg.pcie_cfg.pl_cfg.upstream_lanes_recovery_eq_phase3_timeout_ns = 32000;
    cust_cfg.root_cfg.pcie_cfg.pl_cfg.downstream_lanes_recovery_eq_phase1_timeout_ns =
    24000;
    cust_cfg.root_cfg.pcie_cfg.pl_cfg.downstream_lanes_recovery_eq_phase2_timeout_ns =
    32000;
    cust_cfg.root_cfg.pcie_cfg.pl_cfg.downstream_lanes_recovery_eq_phase3_timeout_ns =
    24000;
    ```

16. What are the primary settings to consider if DUT timed out in Phase0/Phase1?

    You must check SERDES lock and Block alignment at DUT end and also EC bits transmitted by DUT.

17. How to set Quiesce Guarantee bits in transmit TS2?

    The following configuration variables are present in VIP PL configuration class. you can configure desired values.

    Example:

    ```
    cust_cfg.endpoint_cfg.pcie_cfg.pl_cfg.quiesce_guarantee = 1;
    ```

18. How to enable/disable software based equalization?

    VIP can enable soft ware based equalization by using the following PL service request.

    ```
    svt_pcie_pl_service_perform_equalization_sequence
    ```

    Example:

    You must perform the following steps in sequence build task.

    ```
    perform_equalization =
    svt_pcie_pl_service_perform_equalization_sequence::type_id::create("perform_equalization
    ");
      perform_equalization.start(p_sequencer.root_virt_seqr.pcie_virt_seqr.pl_seqr );
    ```

19. How to enable/disable autonomous equalization?

    The following configuration variable is present in VIP PL configuration class. You can configure desired values.

    When set to 1, VIP maintains default behavior of carrying out autonomous 16G equalization procedure.

    When set to 0, VIP does not carry out autonomous 16G equalization procedure.

When set to 0, you must call `PERFORM_EQUALIZATION` service sequence from VIP (Downstream Port) to execute at least software based 16G equalization procedure.

    Example:

    ```
    cust_cfg.root_cfg.pcie_cfg.pl_cfg.enable_autonomous_16g_equalization = 1;
    ```

20. How do I set `lf` and `fs` values?

    You can use the following attributes in the PHY layer configuration class (`svt_pcie_pl_configuration`) to specify the LF and FS values.

    Example code:

    ```
    root_cfg.pcie_cfg.pl_cfg.lf_value = '{32{'d9}};
    root_cfg.pcie_cfg.pl_cfg.fs_value = '{32{'d24}};
    ```

21. How to reject preset or coefficient values requested by the DUT?

    You can use the following attribute in the PHY layer configuration class (`svt_pcie_pl_configuration`) to reject preset or coefficient values requested by the DUT.

    Example code:

    ```
    root_cfg.pcie_cfg.pl_cfg.reject_preset_coefficient_request = 32'h0;
    ```

    Each bit of this attribute maps to a corresponding lane. When a bit is set to 1'b1, the corresponding lane rejects the new preset and coefficient values. This bit is applicable only in equalization phase 2 for downstream ports and equalization phase 3 for upstream ports.

22. How do I disable Equalization in the VIP?

    You can use the following parameters to disable Equalization in the VIP:

    ```
    rand bit  attribute
     svt_pcie_pl_configuration::enable_equalization_verification_mode = 0
    ```

    Enables equalization verification mode (this enables equalization checking).

```
rand int unsigned  attribute
 svt_pcie_pl_configuration::highest_enabled_equalization_phase = 3
```

Specifies highest enabled equalization phase. When set to 1, enables equalization phase0 and phase 1. When set to 3, enables equalization phases 0, 1, 2, and 3 (0 will disable equalization).

23. How do I use the following variables of *svt_pcie_pl_configuration.sv* for equalization?

    - enable_equalization_coefficients_checks
    - enable_equalization_verification_mode

    These variables give us information on how VIP is processing equalization values. By default, the checking is not enabled.

    ```
    svt_pcie_pl_configuration::enable_equalization_coefficients_checks = 0
    ```

    Enables equalization coefficients check. By setting this variable to 1, VIP checks equalization coefficients.

    The default value is 0, therefore you can assume that there is no coefficient error.

    ```
    svt_pcie_pl_configuration::enable_equalization_verification_mode = 0
    ```

    Enables equalization verification mode (this enables equalization checking).

    Use case:

    ```
    root_cfg.pcie_cfg.pl_cfg.enable_equalization_verification_mode = 1;
    root_cfg.pcie_cfg.pl_cfg.enable_equalization_coefficients_checks = 1;

    endpoint_cfg.pcie_cfg.pl_cfg.enable_equalization_verification_mode = 1;
    endpoint_cfg.pcie_cfg.pl_cfg.enable_equalization_coefficients_checks = 1;
    ```

    For more details, see the following documentation:

    - User Guide: *$DESIGNWARE_HOME/vip/svt/pcie_svt/latest/doc/pcie_svt_uvm_user_guide.pdf*
    - HTML Class Reference: *$DESIGNWARE_HOME/vip/svt/pcie_svt/latest/doc/pcie_svt_uvm_class_reference/html/class_svt_pcie_pl_configuration.html*

24. How to check the preset and preset_hint values that are received by VIP?

    You can check the received preset/preset_hint values with the following lane status variable for VIP acting as downstream port and upstream port.

    - `rcvd_upstream_preset_value`
    - `rcvd_upstream_preset_value_16g`
    - `rcvd_downstream_preset_value`
    - `rcvd_downstream_preset_value_16g`

    Example code:

    ```
    rx_preset =
    root_agent.pcie_agent.status.pl_status.lane_status[lane_num].rcvd_upstream_preset_value;
    ```

    For more details, see the following HTML class reference documentation:

    https://solvnet.synopsys.com/dow_retrieve/latest/verification_compiler/doc/nvme_svt_uvm_class_reference/html/class_svt_pcie_lane_status.html#item_rcvd_upstream_preset_value

25. How to let VIP to assert `rx_eq_eval_0` to initiate equalization?

    In order to assert rx_eq_eval to initiate equalization, set the following configuration attribute from `svt_pcie_pl_configuration`.

    Example code:

    ```
    cfg.rc_cfg.pcie_cfg.pl_cfg.enable_rxeqeval_default_settings_vector = 2'b11;
    cfg.ep_cfg.pcie_cfg.pl_cfg.enable_rxeqeval_default_settings_vector = 2'b11;
    ```

26. How to make `rx_eq_eval` de-assert on the same/next edge of phystatus?

    Set the following configuration attribute of the `svt_pcie_pl_configuration` to adjust the de-assertion timing of `rx_eq_eval`.

    Example code:

    ```
    for(int i=0; i< vip_cfg.pcie_cfg.pl_cfg.link_width; i++) begin
    cfg.ep_cfg.pcie_cfg.pl_cfg.pipe_rxeqeval_deassertion_delay[i] = 0; //
    end
    ```

27. How to bypass equalization in VIP for different data rates?

    VIP provides the following function to control the link equalization behavior of VIP:

    ```
    function void svt_pcie_pl_configuration::set_link_eq_attribute_values
    ( svt_pcie_pl_configuration :: link_eq_mode_enum link_eq_mode_value =
    LINK_EQ_MODE_FULL_EQUALIZATION_REQUIRED, bit
    enable_direct_speed_up_from_2_5g_to_16g_value = 0, int
    unsigned highest_enabled_eq_phase_value = 3 )
    ```

    This function determines whether the link will carry out the full sequential equalization for all speeds, or the link will carry out equalization for only the highest supported speed or the link will skip the equalization completely.

    - `link_eq_mode_value`: This attribute determines whether the full equalization will be carried out, or the equalization will be bypassed for lower supported data rates, or the equalization will be bypassed altogether. This attribute is ONLY applicable when VIP supports 32 GT/s speed.
    - `enable_direct_speed_up_from_2_5g_to_16g_value`: This attribute enables the VIP specific behavior and allows the VIP to bypass equalization of 8GT/s speed. This attribute is ONLY applicable when 16 GT/s is the highest supported speed.
    - `highest_enabled_eq_phase_value`: This attribute enables the VIP specific behavior and controls the number of phases while in equalization state for 8 GT/s, 16 GT/s, and 32 GT/s equalization process. When 32 GT/s is NOT the highest supported link speed, then you can bypass the equalization for 8 GT/s or 16 GT/s speeds by setting this attribute to 0.

    Use this function in conjunction with `set_link_speed_values` to achieve various speed change scenarios without equalization.

    - GEN1 -> GEN3 (without equalization)

      ```
      rc_cfg[0].pcie_cfg.pl_cfg.set_link_eq_attribute_values(,,0/highest_enabled_eq_
      phase/);
      ep_cfg[0].pcie_cfg.pl_cfg.set_link_eq_attribute_values(,,0/highest_enabled_eq_
      phase/);
      ```

    - GEN1 -> GEN3 -> GEN4 (without equalization)

```
rc_cfg[0].pcie_cfg.pl_cfg.set_link_eq_attribute_values(,,0/highest_enabled_eq_
phase/)
ep_cfg[0].pcie_cfg.pl_cfg.set_link_eq_attribute_values(,,0/highest_enabled_eq_
phase/)
```

- GEN1 -> GEN4 (without equalization)

```
rc_cfg[0].pcie_cfg.pl_cfg.set_link_eq_attribute_values(,1/enable_direct_speed_
up_from_2_5g_to_16g/, 0
/highest_enabled_eq_phase/);
ep_cfg[0].pcie_cfg.pl_cfg.set_link_eq_attribute_values(,1/enable_direct_speed_
up_from_2_5g_to_16g/, 0
/highest_enabled_eq_phase/)
```

- GEN1 -> GEN5 (without equalization)

```
rc_cfg[0].pcie_cfg.pl_cfg.set_link_eq_attribute_values(svt_pcie_pl_configurati
on::LINK_EQ_MODE_NO_EQUALIZATION_NEEDED);
ep_cfg[0].pcie_cfg.pl_cfg.set_link_eq_attribute_values(svt_pcie_pl_configurati
on::LINK_EQ_MODE_NO_EQUALIZATION_NEEDED);
```

28. How to set the equalization to "DIRECTION_CHANGE" mode?

    For "DIRECTION_CHANGE" mode set up, VIP provides a service type,

    "QUEUE_EQ_DIRECTION_CHANGE_RESPONSE". This sets the direction change response from equalization evaluation. If no responses are queued, the default response used indicates no change on any of the coefficients.

    For example:

```
/** Lane number for the equalization request */

      rand bit [31:0] eq_lane_num = 32'h00;

/** Direction change response */

rand bit[5:0] direction_change_response = 6'h00;

svt_pcie_pl_service queue_eq_direction_change_response_req;
`uvm_do_with(queue_eq_direction_change_response_req

{service_type==         svt_pcie_pl_service::QUEUE_EQ_DIRECTION_CHANGE_RESPONSE;
eq_lane_num == local::eq_lane_num;             direction_change_response ==
local::direction_change_response;}
)
```

The request shown in the example is created for an SPIPE model. This service call is not valid for MPIPE/ Serdes model.

4.9 Gen4

1. How to enable control SKP support feature for Gen4 data rate?

   VIP provides the following configuration attributes to enable control SKP support feature for Gen4 data rate:

   `svt_pcie_pl_configuration::enable_ctrl_skp_support`

   When set, the VIP enables support for CTRL-SKP OS at 16G (Gen4) and above. This feature will also enable support for Rx Margining and `FaultIsolation`. By default, it is disabled.

Note Only utilized by the Active component.

   Example code for configuration:

   `root_cfg.port_cfg.pl_cfg.enable_ctrl_skp_support = 1;`

   Gen4 tests with:

   symbol_log snippet here.

   Try with one device enabled and second with disabled.

   Standard SKP format:

```
*aa  *aa  *aa  *aa  *aa  *aa  *aa  *aa  *aa  *aa  *aa  *aa  *aa  *aa  *aa  *aa   -> tx=L0, rx=L0
 aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   -> tx=L0, rx=L0
 aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   -> tx=L0, rx=L0
 aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   -> tx=L0, rx=L0
 aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   -> tx=L0, rx=L0
 aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   -> tx=L0, rx=L0
 aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   -> tx=L0, rx=L0
 aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   -> tx=L0, rx=L0
 aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   -> tx=L0, rx=L0
 aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   -> tx=L0, rx=L0
 aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   -> tx=L0, rx=L0
 aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   -> tx=L0, rx=L0
 e1   e1   e1   e1   e1   e1   e1   e1   e1   e1   e1   e1   e1   e1   e1   e1   -> tx=L0, rx=L0
 fc   83   d7   54   48   1c   e3   ff   fc   83   d7   54   48   1c   e3   7f   -> tx=L0, rx=L0
 4c   9a   83   19   72   6b   bd   d6   4c   9a   83   19   72   6b   bd   d6   -> tx=L0, rx=L0
 cb   a6   4d   eb   9e   75   18   6d   cb   a6   4d   eb   9e   75   18   6d   -> tx=L0, rx=L0
```

Control SKP format:

```
*aa  *aa  *aa  *aa  *aa  *aa  *aa  *aa  *aa  *aa  *aa  *aa  *aa  *aa  *aa  *aa  -> tx=L0, rx=L0
 aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa  -> tx=L0, rx=L0
 aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa  -> tx=L0, rx=L0
 aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa  -> tx=L0, rx=L0
 aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa  -> tx=L0, rx=L0
 aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa  -> tx=L0, rx=L0
 aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa  -> tx=L0, rx=L0
 aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa  -> tx=L0, rx=L0
 aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa  -> tx=L0, rx=L0
 aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa  -> tx=L0, rx=L0
 aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa  -> tx=L0, rx=L0
 aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa   aa  -> tx=L0, rx=L0
 78   78   78   78   78   78   78   78   78   78   78   78   78   78   78   78  -> tx=L0, rx=L0
 19   19   f9   19   f9   f9   f9   19   19   19   f9   f9   19   f9   f9   19  -> tx=L0, rx=L0
 38   38   38   38   38   38   38   38   38   38   38   38   38   38   38   38  -> tx=L0, rx=L0
 9c   9c   9c   9c   9c   9c   9c   9c   9c   9c   9c   9c   9c   9c   9c   9c  -> tx=L0, rx=L0
```

2. How to enable simplified replay timer?

   VIP provides the following configuration attribute to enable simplified replay timer with Gen4 data rate when `extended_sync` is set to 0.

   `svt_pcie_dl_configuration::replay_timeout`

   VIP supports `replay_timer` value from 24,000 to 31,000 symbol times when Gen4 is supported.

   Example code:

```
cfg.rc_cfg.pcie_cfg.dl_cfg.replay_timeout = 31000; (or some more value)
cfg.rc_cfg.pcie_cfg.dl_cfg.attached_replay_timeout = 31000; (or some more value)
```

5 Score-boarding

This section provides frequently asked Score-boarding questions for the VC VIP for PCIe.

1. What is Global Shadow?

   PCIe VIP provides shadow of the DUT's address space for Read/Write transaction checking. The shadow transaction handler captures the DUT transactions on the bus. Class name in SVT to access is `svt_pcie_global_shadow`.

   Global shadow needs to be explicitly declared and instantiated at the top when used:

   ```
   `define EXPERTIO_PCIESVC_GLOBAL_SHADOW_PATH  test_top.global_shadow0
   pciesvc_global_shadow #( .DISPLAY_NAME( "global_shadow0." ) ) global_shadow0();
   ```

   For more information, see the SolvNetPlus article at

   https://solvnet.synopsys.com/retrieve/1588012.html

2. How to disable Global Shadow?

   Disable the shadow memory checking by setting the `enable_shadow_memory_checking` attribute in the configuration class to 0.

   For example,

   ```
   root_cfg.driver_cfg[0].enable_shadow_memory_checking = 1'b0;
   ```

   For more information, see the SolvNetPlus article at

   https://solvnet.synopsys.com/retrieve/2061397.html

3. How to enable access for sent and received TLPs and DLLPs from the VIP for scoreboard?

   To have access to TLPs or DLLPs for external scoreboarding, enable the model to output these transactions by updating the DL layer configuration object.

   To enable this, set the following configuration variable:

   ```
   root_cfg.port_cfg.dl_cfg.sent_tlp_interface_mode = 1;
   root_cfg.port_cfg.dl_cfg.received_tlp_interface_mode = 1;
   root_cfg.port_cfg.dl_cfg.sent_dllp_interface_mode = 1;
   root_cfg.port_cfg.dl_cfg.received_dllp_interface_mode = 1;
   ```

   For more information, see the SolvNetPlus article at

   https://solvnet.synopsys.com/retrieve/1497240.html

4. How to bypass check for comparison in scoreboard for completions generated by VIP in `pcie_svt`?

   There is a configuration to enable comparison of completion which are generated by VIP. Set this variable to 0 in the test case so that scoreboard will not compare CPL/CPLD.

Example code:

```
cfg. enable_sb_dut_rx_cpl_compare = 0 ;
```

6 Licensing

This section provides frequently asked Licensing question for the VC VIP for PCIe.

1. How do I set the `DW_LICENSE_OVERRIDE` environment variable with multiple license features?

   The `DW_LICENSE_OVERRIDE` environment variable enables you to force usage of particular license features during verification.

   In case of multiple license features, the variable is set as follows:

   ```
   % setenv DW_LICENSE_OVERRIDE "VIP-LIBRARY-SVT DESIGNWARE-REGRESSION
   ```

   For more information, see the SolvNetPlus article at

   https://solvnet.synopsys.com/retrieve/2125477.html

7 Error Injections

This section provides a list of frequently asked Error Injection questions for the VC VIP for PCIe.

1. What are exceptions?

   Exception classes are provided to inject error in specific fields of TLPs, DLLPs or PLPs before sending the packets to the next layer or to the PCIe Bus. Error injections can be implemented using factory, sequence or callback.

   For example,

   TL-> `CORRUPT_FMT` (Corrupt the FMT field of TLP before sending a packet to the DL)

   DL -> `AUTO_TX_CORRUPT_CRC` (Corrupt the CRC of a DLLP)

   The following are the layer specific exception classes:

Table 7-1 Layer Specific Exception Classes

Layer	Exception Classes
TL	`svt_pcie_tlp_exception`
DL	`svt_pcie_dllp_exception`
PL	(Ordered Set exception) `svt_pcie_os_exception`

2. What are the types of callback provided in the VIP and the usage of these callbacks?

   Several callbacks are provided in each protocol layer. Virtual methods to override callbacks are defined in layer specific proxy classes. These virtual methods provide handles to received packets for validating protocol specific parameters in the packets, and handles to packets before transmission for injecting error.

   For example,

   Use callback to access the CplD of TLPs received at the DL and validate the value of the completer ID field.

   Use callback to corrupt the `first_dw_be` field of TLPs transmitted from the VIP before sending the packets to the DL.

   Use callback to access the requester_id of PCIe error packets received and validate the value.

   The following are callback classes:

Table 7-2 Callback Classes Description

Layer	Classes	Description
TL	svt_pcie_tl_callback	Virtual method definition
DL	svt_pcie_dl_callback	Virtual method definition
PL	svt_pcie_pl_callback	Virtual method definition

For more information, see the SolvNetPlus articles at

https://solvnet.synopsys.com/retrieve/2372765.html

https://solvnet.synopsys.com/retrieve/2315907.html

https://solvnet.synopsys.com/retrieve/2369278.html

https://solvnet.synopsys.com/retrieve/2235954.html

https://solvnet.synopsys.com/retrieve/2024177.html

3. What is the virtual method definition of the TL callback?

<callback_func> Describe the function used to implement the callback

<callback_id> Describe the objective of the callback

```
class svt_pcie_tl_<callback_func>_<callback_id>_callback extends
svt_pcie_tl_callback

// Callback issued by the component once the TLP is received completely and
// prior to putting received TLP on the rx port.
//  <tl>      A reference to the component object issuing this callback.
// <tlp>     A reference to the svt_pcie_tlp descriptor object of interest.
// <drop> A reference bit which indicates to drop the transaction when set to 1.

virtual function void pre_tlp_out_put (svt_pcie_tl tl, svt_pcie_tlp tlp, ref bit
drop);
/**
*
*/
endfunction

// Callback issued by the component after pulling a TLP out of its
// TLP input, but before acting on the TLP in any way.
// <tl>       A reference to the component object issuing this callback.
// <tlp>     A reference to the svt_pcie_tlp descriptor object of interest.
// <drop> A reference bit argument that, if set by the user's implementation,
causes the
```

```
// component to discard the svt_pcie_tlp descriptor without further action.
virtual function void post_seq_item_get (svt_pcie_tl tl, svt_pcie_tlp tlp, ref
bit drop);
/**
*
*/
endfunction

endclass
```

4. What is the virtual method definition of the DL callback?

<callback_func> Describe the function used to implement the callback

<callback_id> Describe the objective of the callback

```
class svt_pcie_dl_<callback_func>_<callback_id>_callback extends
svt_pcie_dl_callback

// Callback issued by the component after scheduling a TLP transaction
// for transmission on the link, just prior to framing.
//  <dl> A reference to the component object issuing this callback.
// <transaction>  A reference to the svt_pcie_dl_tlp descriptor object of interest
// <drop> A reference, if set, causes the transaction to be dropped prior
// to transmission.

virtual function void pre_tlp_framed_out_put (svt_pcie_dl dl, svt_pcie_dl_tlp
transaction, ref bit drop);
/**
*
*/
endfunction

// Callback issued by the component after recognizing a TLP transaction
// received immediately from a link.
//  <dl> A reference to the component object issuing this callback.
// <transaction>  A reference to the svt_pcie_dl_tlp_transaction descriptor
object of interest.
// <drop> A reference, if set, causes the transaction to be dropped prior to
// transmission.

virtual function void post_tlp_framed_in_get (svt_pcie_dl dl, svt_pcie_dl_tlp
transaction, ref bit drop);
/**
*
```

```
*/
endfunction

// Callback issued by the component after building a DLLP transaction and prior to
further
// processing of the transaction.
//  <dl> A reference to the component object issuing this callback.
// <transaction>  A reference to the svt_pcie_dllp descriptor object of interest.
// <drop> A reference, if set, causes the transaction to be dropped prior
// to transmission.

virtual function void tx_dllp_started (svt_pcie_dl dl, svt_pcie_dl_tlp
transaction, ref bit drop);
/**
*
*/
endfunction

// Callback issued by the component after receiving DLLP transaction from an input
port.
//  <dl> A reference to the component object issuing this callback.
// <transaction>  A reference to the svt_pcie_dllp descriptor object of interest..
// <drop> A reference bit, if set to 1, indicates that the transaction is dropped.

virtual function void rx_dllp_started (svt_pcie_dl dl, svt_pcie_dl_tlp
transaction, ref bit drop);
/**
*
*/
endfunction
endclass
```

For more information, see the SolvNetPlus articles at

https://solvnet.synopsys.com/retrieve/2315907.html

https://solvnet.synopsys.com/retrieve/2372765.html

https://solvnet.synopsys.com/retrieve/2369278.html

5. What is the virtual method definition of the PL callback?

   <callback_func> Describe the function used to implement the callback

   <callback_id> Describe the objective of the callback

   ```
   class svt_pcie_pl_<callback_func>_<callback_id>_callback extends
   svt_pcie_pl_callback
   ```

```
// Callback issued by the component after building a TS OS Transaction.
// The callback can be used to attach an exception list to the TS OS
// transaction prior to the transmission of the transaction on the link.
//  <pl> A reference to the component object issuing this callback.
// <transaction>  A reference to the svt_pcie_os descriptor object of interest.

virtual function void tx_ts_os_started (svt_pcie_pl pl, svt_pcie_os transaction);
/**
*
*/
endfunction

// Callback issued by the component after building an OS Transaction.
// The callback can be used to attach an exception list to the OS
// transaction prior to the transmission of the transaction on the link.
//  <pl> A reference to the component object issuing this callback.
// <transaction>  A reference to the svt_pcie_os descriptor object of interest.
virtual function void tx_os_started (svt_pcie_pl pl, svt_pcie_os transaction);
/**
*
*/
endfunction

// Callback issued by the component at every symbol time after gathering all
// the symbols to be transmitted on the PCIe link. This is the last opportunity
// to corrupt any symbol before the transmission of the symbol on the link.
//  <pl> A reference to the component object issuing this callback.
//  <symbols[]> A reference to the svt_pcie_symbol descriptor object queue of
interest..
virtual function void pre_symbol_out_put (svt_pcie_pl pl, svt_pcie_symbol
symbols[]);
/**
*
*/
endfunction
endclass
```

For more information, see the SolvNetPlus article at

https://solvnet.synopsys.com/retrieve/2024177.html

6. How do I inject error in packets when callbacks are not available?

Use constraints to inject error in packets when callbacks are not available. Valid constraints are defined for the fields of a packet. If you disable the constraints for a specific field or specify invalid constraints, then packets are generated with a corrupted field.

For example,

CfgWr TLP transactions with non-zero TC can be sent when constraints are disabled. In such case, the VIP transmits TLP with error injection in TC.

7. What are the DLL exceptions available in the VIP?

   The DLL exceptions are defined in the DLLP exception class (`svt_pcie_dllp_exception`) and the DL TLP exception (`svt_pcie_dl_tlp_exception`) class. The following is a list of error types. For more information on the error types, see the HTML class reference.

Table 7-3 Error Types

svt_pcie_dllp_exception	svt_pcie_dl_tlp_exception
NO_ERROR	NO_ERROR
AUTO_TX_RETAIN_CRC	AUTO_TX_ILLEGAL_SEQ_NUM
AUTO_TX_FORCE_CRC	AUTO_TX_DUPLICATE_SEQ_NUM
AUTO_TX_CORRUPT_CRC	AUTO_TX_NULLIFIED_TLP
AUTO_TX_UNKNOWN_TYPE	AUTO_TX_NULLIFIED_TLP_GOOD_LCRC
AUTO_TX_RSVD_NON_ZERO	AUTO_TX_NULLIFIED_TLP_CORRUPT_LCRC
AUTO_TX_DUPLICATE_ACK	TX_CORRUPT_DISPARITY
AUTO_TX_MISSING_START	AUTO_TX_CODE_VIOLATION
AUTO_TX_MISSING_END	TX_MISSING_START
AUTO_TX_CORRUPT_DISPARITY	AUTO_TX_MISSING_END
AUTO_TX_CODE_VIOLATION	TX_CORRUPT_8G_HEADER_CRC
AUTO_TX_CORRUPT_8G_HEADER	TX_CORRUPT_8G_HEADER_PARITY
	AUTO_TX_RETAIN_LCRC
	AUTO_TX_FORCE_LCRC
	AUTO_TX_CORRUPT_LCRC
	AUTO_RX_WITHHOLD_ACK_NAK
	AUTO_RX_NAK_GOOD_TLP
	AUTO_RX_REPLAY_COUNT_FAILURE

For more information, see the SolvNetPlus articles at

https://solvnet.synopsys.com/retrieve/2061209.html

https://solvnet.synopsys.com/retrieve/2372765.html

https://solvnet.synopsys.com/retrieve/2369278.html

8. How do I stop error injection in TS ordered sets?

   For TS1 ordered sets, you can set the `user_tx_ts1_lane_mask` variable in the `svt_pcie_pl_configuration` class to 0 to disable the user TS1 transmission and to start transmitting the intended TS1.

   For TS2 ordered sets, you can use the `user_tx_ts2_lane_mask` variable.

   For example,

```
ep_cfg.pcie_cfg.pl_cfg.user_tx_ts1_lane_mask = 32'h0;
ep_agent.reconfigure_via_task(ep_cfg);
```

9. How do I inject framing errors?

   Use the following methods to inject framing errors:

   - The OS exception class (`svt_pcie_os_exception`) and the `tx_os_started` callback of the PL callback class (`svt_pcie_pl_callback`)

     For example,

     - Create an exception list

```
//Extend the svt_pcie_os_exception_list class to generate an exception for injecting
//an error via a callback
class svt_pcie_clear_os_exception_list extends svt_pcie_os_exception_list;
svt_pcie_os_exception xact_exc = new("xact_exc");
function new(string name = "svt_pcie_clear_os_exception_list",
svt_pcie_os_exception xact_exc = null);
super.new(name,xact_exc);
xact_exc = this.xact_exc;
xact_exc.NO_ERROR_wt = 1;
xact_exc.set_constraint_weights(0);
this.randomized_exception = xact_exc;
endfunction : new
endclass
```

     - Attach the exception list to a callback

```
virtual function void tx_os_started(svt_pcie_pl pl, svt_pcie_os transaction);
if(transaction.os_type == svt_pcie_os::SDS_OS) begin
`svt_note("tx_sds_os_started",$sformatf("\nSDS OS of type %0d will be transmitted
on lane %0d.\n",transaction.os_type, transaction.logical_lane_num));

//Test case decision to determine how many errors to introduce.
if(error_count < 1 ) begin
if(transaction.logical_lane_num == 0) begin
my_sds_os_exc_list.xact_exc.NO_ERROR_wt = 0;
my_sds_os_exc_list.xact_exc.INVALID_DATABYTE_wt = 1000;
end else if(transaction.logical_lane_num == 1) begin
my_sds_os_exc_list.xact_exc.NO_ERROR_wt = 0;
my_sds_os_exc_list.xact_exc.INVALID_DATABYTE_wt = 1000;
end else if(transaction.logical_lane_num == 2) begin
```

```
my_sds_os_exc_list.xact_exc.NO_ERROR_wt = 0;
my_sds_os_exc_list.xact_exc.INVALID_DATABYTE_wt = 1000;
end else if(transaction.logical_lane_num == 3) begin
my_sds_os_exc_list.xact_exc.NO_ERROR_wt = 0;
my_sds_os_exc_list.xact_exc.INVALID_DATABYTE_wt = 1000;
end
end else begin
my_sds_os_exc_list.xact_exc.NO_ERROR_wt = 100;
my_sds_os_exc_list.xact_exc.INVALID_DATABYTE_wt = 0;
end

my_sds_os_exc_list.randomized_exception = my_sds_os_exc_list.xact_exc;
my_sds_os_exc_list.setup_randomized_exception(dummy_pl_cfg,transaction);
success = my_sds_os_exc_list.num_exceptions_first_randomize();
error_count += my_sds_os_exc_list.num_exceptions;
`svt_note("pre_sds_os_xact_transmission",$sformatf(" Attaching exception
list .\n."));
$cast(transaction.exception_list, my_sds_os_exc_list.`SVT_DATA_COPY());
end
endfunction
```

- Register the callback in a test case :

```
//End of elaboration phase
function void end_of_elaboration_phase(uvm_phase phase);
`uvm_info("end_of_elaboration_phase", "Entry");
endpoint_sds_os_cb = new("endpoint_sds_os_cb");
uvm_callbacks#(svt_pcie_pl,svt_pcie_pl_callback)::add(rc_vip_agent.pcie_age
nt.pl,
endpoint_sds_os_cb);
`uvm_info("end_of_elaboration_phase", "Exit");
endfunction
```

- The symbol exception class (`svt_pcie_symbol_exception`) and the `pre_symbol_out_put` callback of the PL callback class (`svt_pcie_pl_callback`)

  For example,

  - Check if the VIP is transmitting the Frame to be corrupted. If yes, use the `CORRUPT_DATA_VALUE_ONLY error_kind` of the symbol exception class to inject an error.

```
virtual function void pre_symbol_out_put(svt_pcie_pl pl, svt_pcie_symbol
symbols[]);
if(error_count < 2)
begin
for(int lane = 0 ; lane < symbols.size() ; lane = lane + 1)
begin
if(symbols[lane].symbol_event == svt_pcie_symbol::EDS_STARTED)
symbol_found_flag = 1;
end

if(symbol_found_flag == 1 && pl.status.ltssm_state == svt_pcie_types::L0)
```

```
begin
for(i=0; i< symbols.size(); i++) begin
svt_pcie_symbol_exception_list sym_exc_list = new();
svt_pcie_symbol_exception sym_exc = new();
sym_exc.error_kind = svt_pcie_symbol_exception::CORRUPT_DATA_VALUE_ONLY;
sym_exc.corrupted_data = svt_pcie_symbol::SKP;  //SKP OS Identifiers
sym_exc.scrambler_control = svt_pcie_symbol_exception::NONE;
sym_exc_list.add_exception(sym_exc);
symbols[i].exception_list = sym_exc_list;
`svt_note("TEST: pre_symbol_out_put_callback", $sformatf( "\n Corrupting
EDS on lane %d.with SKP OS Identifier.\n", i));
end
error_count = error_count + 1;
end
end
endfunction
```

- Register the callback in a test case

```
//End of elaboration phase
function void end_of_elaboration_phase(uvm_phase phase);
`uvm_info("end_of_elaboration_phase", "Entry");
rc_sym_cb = new("rc_sym_cb");
uvm_callbacks#(svt_pcie_pl,svt_pcie_pl_callback)::add(rc_vip_agent.pcie_age
nt.pl,
rc_sym_cb);
`uvm_info("end_of_elaboration_phase", "Exit");
Endfunction
```

10. How do I inject an error to cause a symbol lock failure between the DUT and the VIP?

    Use the `pre_symbol_out_put` callback of the PL callback class (`svt_pcie_pl_callback`) to corrupt the COM (Symbol0) of the TS1 Ordered Set. A corrupted COM in the Polling.Active state of the LTSSM causes a symbol lock failure.

    For example,

```
virtual function void pre_symbol_out_put(svt_pcie_plpl, svt_pcie_symbol
symbols[]);
`uvm_info("pre_symbol_out_put",$sformatf("\n Link width = %0d. Symbols ready for
transmission.\n",symbols.size()));

if(symbols[0].symbol_type == svt_pcie_symbol::COM &&pl.status.ltssm_state ==
svt_pcie_types::POLLING_ACTIVE)
begin
for(i=0; i<symbols.size(); i++)
begin
svt_pcie_symbol_exception_listsym_exc_list = new();
```

```
svt_pcie_symbol_exceptionsym_exc = new();
sym_exc.error_kind = svt_pcie_symbol_exception::CORRUPT_DATA_VALUE_ONLY;
sym_exc.corrupted_data  = 8'F7;     //Other than COM value(BC)
sym_exc.scrambler_control = svt_pcie_symbol_exception::NONE;
sym_exc_list.add_exception(sym_exc);
symbols[i].exception_list = sym_exc_list;
`svt_note("TEST: pre_symbol_out_put_callback",$sformatf("\nCorruptingCOM lane
%d.\n", i));
end
error_count = error_count + 1;
end
endfunction
```

For more information, see the SolvNetPlus article at

https://solvnet.synopsys.com/retrieve/2024177.html

11. How do I inject an error to cause a block misalignment in Gen3?

    Use the `pre_symbol_out_put` attribute of the PL callback class (`svt_pcie_pl_callback`) to corrupt the Sync Header. A corrupted Sync Header causes either a Data Block or an Ordered Set Block (130 bits) misalignment in Gen3.

    For example,

```
class svt_pcie_pl_pre_symbol_out_put_corrupt_sync_header_recovery_idle_callback
extends svt_pcie_pl_callback;
//Facroty registration
`uvm_object_utils(svt_pcie_pl_pre_symbol_out_put_corrupt_sync_header_recovery_idl
e_callback)
int i;
rand bit [1:0] corrupted_value;
static int error_count = 0;
constraint corrupt { !(corrupted_value inside {2'b10, 2'b01});}

function new(string name =
"svt_pcie_pl_pre_symbol_out_put_corrupt_sync_header_recovery_idle_callback");
super.new();
endfunction

virtual function void pre_symbol_out_put(svt_pcie_pl pl, svt_pcie_symbol
symbols[]);
`uvm_info("pre_symbol_out_put",$sformatf("\n Link width = %0d. Symbols ready
for transmission.\n",symbols.size()));
if(error_count < 1)
begin
```

```
if(symbols[0].symbol_type == svt_pcie_symbol::DATA_WITH_SYNC_HDR &&
pl.status.ltssm_state == svt_pcie_types::RECOVERY_IDLE)
begin
for(i=0; i< symbols.size(); i++)
begin
svt_pcie_symbol_exception_list sym_exc_list = new();
svt_pcie_symbol_exception sym_exc = new();
sym_exc.error_kind = svt_pcie_symbol_exception::CORRUPT_SYNC_HEADER_ONLY;
sym_exc.corrupted_data [1:0] = corrupted_value;
sym_exc.scrambler_control = svt_pcie_symbol_exception::NONE;
sym_exc_list.add_exception(sym_exc);
symbols[i].exception_list = sym_exc_list;
`svt_note("TEST: pre_symbol_out_put_callback",$sformatf("\nCorrupting
SYNC_HDR lane %d.\n", i));
end
error_count = error_count + 1;
end
end
endfunction
endclass //
svt_pcie_pl_pre_symbol_out_put_corrupt_sync_header_recovery_idle_callback
```

12. How do I inject BIT_FLIP error using pl callback?

    When using 8b/10b encoding, VIP randomly flips one of the 10 bits in the specified transmission character after being properly encoded and applies the specified scrambler control. When using 128b/130b encoding, VIP randomly flips one of the 8 bits in a symbol and applies the specified scrambler control.

    BIT_FLIP error injection is done as shown below:

    `pre_symbol_out_put`: Callback issued by the PL at every symbol time after gathering all the symbols to be transmitted on the PCIe link. This is the last chance the user has to corrupt any symbol before it goes on the link.

    Example_code:

```
  virtual function void pre_symbol_out_put(svt_pcie_pl pl,
svt_pcie_symbol symbols[]);
    // Test case decision to determine how many errors to introduce.
    if(error_count < 10) begin
      if(symbols[0].symbol_event == svt_pcie_symbol::TS2_STARTED) //event indicates what
type of symbol is starting
      begin
        // build an exception list
        svt_pcie_symbol_exception_list sym_exc_list = new();
        svt_pcie_symbol_exception sym_exc = new();
        sym_exc.error_kind = svt_pcie_symbol_exception::BIT_FLIP;
        sym_exc.scrambler_control = svt_pcie_symbol_exception::NONE; // Do not corrupt
scrambling
        sym_exc_list.add_exception(sym_exc);
```

```
        symbols[0].exception_list = sym_exc_list; // add the Exception to the symbol
        error_count++;
      end
    end
  endfunction
```

8 Coverage

This section provides a list of frequently asked Coverage questions for the VC VIP for PCIe.

1. How do check/enable functional coverage in PCIe VIP

   To check/enable functional coverage in PCIe VIP, perform the following:

   Enable the `enable_cov` option in the `svt_pcie_configuration` class.

   For example,

   ```
   root_cfg.pcie_cfg.enable_cov = 5'b11111;
   endpoint_cfg.pcie_cfg.enable_cov = 5'b11111;
   ```

   In the enable_cov variable, bit 0 enables PIPE related functional coverage, and bits 1, 2, and 3 enable functional coverage for the Physical Layer, Data Link Layer, and Transaction Layer respectively.

   Run any example with `enable_cov` option.

   In `./output` directory `*.vdb` database will be generated

   Run the following command to create the `coverage_report`.

   ```
   % urg -dir *.vdb
   ```

   For more information, see SolvNet article 2284910, "Viewing PCIe SVT Functional Coverage Information".

2. How do I exclude PCIe coverage which is not relevant to my PCIe configuration?

   To exclude the PCIe coverage which is not relevant, create an exclusion filter.

   Generate an exclusion filter with the following steps:

   a. Get the `*.vdb` database.

   b. Open the Exclusion Manager tool and load `*.vdb`. Select the covergroups which you want to disable/exclude.

   c. Save the exclusion file with extension as `.el` file.

   d. Generate the coverage report with the following command (for VCS tool)

      ```
      %  urg -dir <path to .vdb database> -elfile <path to .el file>.
      ```

      For more details, see Verdi® Coverage User Guide and Tutorial and SolvNet article 2595106, "Comments for Exclusions Using the Verdi Coverage Tool".

9 VIP Reset

This section provides a list of frequently asked VIP Reset questions for the VC VIP for PCIe.

1. How do I enable hot reset mode in the VIP?

   You can use the `HOT_RESET_FORCE` service in the DL service class to enable hot reset mode in the VIP. The hot reset mode causes the `phy_link_down` in the VIP and the DUT to enter the hot rest state.

   For example,

   // Trigger sequence for hot reset from VIP and force DUT to enter the hot reset state.

   // In the hot rest state, the physical linkup bit becomes zero and causes the DL state to

   // transition from `DL_Active` to `DL_Inactive`.

   ```
   `uvm_do_on_with(hot_reset_seq,vip_seqr.pcie_virt_seqr.pl_seqr,
   {hot_reset_seq.mode == svt_pcie_pl_service::HOT_RESET_FORCE;})
   ```

   For more information, see the following SolvNetPlus article:

   https://solvnet.synopsys.com/retrieve/1887516.html

2. How to unplug PCIe VIP?

   The PCIe VIP supports the HotPlug feature to plug and unplug the LTSSM.

   To completely disable the LTSSM, set the `hot_plug_mode` to `HOT_PLUG_UNPLUG`.

   In this mode, the VIP is unplugged. The PCIe VIP does not respond to the other side of the link until the HotPlug mode changes to `HOT_PLUG_DETECT`.

   Using the sequence to control HotPlug mode:

   ```
   // within sequence test
   // "Unplug" the VIP
      svt_pcie_pl_service_request_hot_plug_mode_sequence hot_plug_seq;
      `uvm_do_on_with(hot_plug_seq,
   p_sequencer.root_virt_seqr.pcie_virt_seqr.pl_seqr,
   {hot_plug_seq.hot_plug_mode==svt_pcie_pl_service::HOT_PLUG_UNPLUG;})
   ...
   // Whenever the DUT is ready to link up or user needs to start the Linkup process after waiting for a certain period of time
   // "Plug" in the VIP
   ```

```
  `uvm_do_on_with(hot_plug_seq,
p_sequencer.root_virt_seqr.pcie_virt_seqr.pl_seqr,
{hot_plug_seq.hot_plug_mode==svt_pcie_pl_service::HOT_PLUG_DETECT;})
...
```

For more information, see the following SolvNetPlus article:

https://solvnet.synopsys.com/retrieve/040545.html

3. How to transition to Hot Reset state?

   Use the PL service sequence class to generate a Hot Reset in the PCIe VIP.

   The following code snippet shows how the hot reset service sequence is initiated.

```
//Initiate hot reset
svt_pcie_pl_service_set_hot_reset_mode_sequence init_hot_reset_seq;
init_hot_reset_seq.mode = svt_pcie_pl_service::HOT_RESET_WAIT;
init_hot_reset_seq.start(p_sequencer.root_virt_seqr.pcie_virt_seqr.pl_seqr );
wait(root_device.pcie_agent.pl.status.ltssm_state == svt_pcie_types::HOT_RESET);

// Exit from HOT_RESET to DETECT
init_hot_reset_seq.mode = svt_pcie_pl_service::HOT_RESET_INACTIVE;
init_hot_reset_seq.start(p_sequencer.root_virt_seqr.pcie_virt_seqr.pl_seqr );
```

   For more information, see the following SolvNetPlus article:

   https://solvnet.synopsys.com/retrieve/1887516.html

4. How to perform mid-sim-reset in PCIe VIP?

   In PCIe VIP mid-sim reset is performed with a combination of hot plug control and application reset.

   For mid-sim-reset, perform the following steps:

   a. Initialize DUT and VIP, run to a point in the test where a mid-sim reset is to be performed.
   b. Unplug VIP from bus (`HOTPLUG_UNPLUG`).
   c. Assert Reset on the DUT.
   d. Reset apps (all, some – user choice).
   e. Re-enable the VIP on the bus (`HOTPLUG_DETECT`).
   f. De-assert Reset on the DUT.
   g. Continue with the test.

   For more information, see the following SolvNetPlus article:

   https://solvnet.synopsys.com/retrieve/2357728.html

5. How to disable LTSSM and keep VIP in `detect` state?

   In UVM, the HotPlug control is enabled through Physical Layer service(svt_pcie_pl_phy_service). By keeping `hot_plug` mode as `HOT_PLUG_UNPLUG`, LTSSM will move immediately to `detect`.

   The link will go down, and all transmitters shut off. LTSSM will remain in `detect` until the hot plug mode changes and not perform any checking on incoming data.

   For more information, see the following SolvNetPlus article:

https://solvnet.synopsys.com/retrieve/040545.html

6. How VIP works when using HOT_PLUG_MONITOR mode?

   `HOT_PLUG_MONITOR`: LTSSM will go to `detect`, bring down the link and shut off transmitters. But it will attempt to check incoming data if possible.

   In `HOT_PLUG_MONITOR`, PHY will check incoming training sets and make sure that the DUT is timing out correctly in `detect/polling.active`, but the PHY will not exit `detect` in this state.

   As PHY will not exit `detect`, your simulation is not continue.

   For more information, see the following SolvNetPlus article:

   https://solvnet.synopsys.com/retrieve/040545.html

10 Miscellaneous

This section provides a list of Miscellaneous questions for the VC VIP for PCIe.

1. How do I monitor the status and notification of a simulation timeout?

   Use the `wait_for_timeout` function in the `svt_timer` class to monitor the status and notification of a simulation timeout.

   For example,

   After starting a timer with a value predefined by the VIP, a flag is issued to indicate if a timer has expired or not. If the flag is assigned high when the required state transition has occurred, then it indicates that the timer has expired exactly when the state transition or a timeout in a state has happened.

   ```
   loc_timer.wait_for_timeout(timer_128us_flag);
   ```

2. How can I resolve the following memory allocation errors?

   ```
   ERROR: mem0.AllocateMemory: Could not allocate memory of size <bytes> for <scope>
   HandleMemReq: MemRd accessed uninitialized memory data at addr=<addr>, BE=<byte enables>, status=Memory addr invalid
   ReservePage32(addr=<addr>): 32-bit Page Tbl is full. Please increase the number of NUM_32_BIT_PAGES
   ReservePage64(addr=<addr>): 64-bit Page Tbl is full. Please increase the number of NUM_64_BIT_PAGES
   ```

   For more information, see the SolvNetPlus article at

   https://solvnet.synopsys.com/retrieve/2343641.html

3. How do I generate a Native PA FSDB dump for PCIe SVT?

   Protocol Analyzer (PA) is also supported as a feature in Verdi. This article will focus on various VIP controls to debug PA transactions.

   Configuration process includes the following three steps:

   Step 1:

   Enable layer-wise XML support for the VIP present in the *svt_pcie_configuration.sv* file

   - bit `enable_tl_xml_gen`: To enable/disable TL layer's transaction for PA
   - bit `enable_dl_xml_gen`: To enable/disable DL layer's transaction for PA
   - bit `enable_pl_xml_gen`: To enable/disable PL layer's transaction for PA

For example, taking VIP as an endpoint and setting its device configuration

```
svt_pcie_device_configuration  endpoint_cfg;
      ...
      ...
      endpoint_cfg.pcie_cfg.enable_tl_xml_gen = 1'b1;
      endpoint_cfg.pcie_cfg.enable_dl_xml_gen = 1'b1;
      endpoint_cfg.pcie_cfg.enable_pl_xml_gen = 1'b1;
```

Step 2:

`svt_xml_writer` is an SVT PA transaction's writer file which provides three different options for VIP transaction generation.

a) `endpoint_cfg.pcie_cfg.pa_format_type = svt_xml_writer::FSDB;`

The above setting generates a single FSDB dump which contains both standard waveform activity as well as the PA transaction activity inside it.

b) `endpoint_cfg.pcie_cfg.pa_format_type = svt_xml_writer::XML;`

The above setting generates XML files for each layer of the VIP for those enabled in Step 1. This is like a legacy control where a standalone Protocol Analyzer tool takes XML files as an input to display transaction activity.

c) `endpoint_cfg.pcie_cfg.pa_format_type = svt_xml_writer::BOTH;`

The above setting would generate FSDB dump along with separate VIP XML files for each layer for those enabled in Step 1. Here, you can view waveform and PA transaction activity by loading FSDB dump in Verdi as done in part a) of this step. It gives similar backward compatibility option to load standalone XML files in Protocol Analyzer tool.

Step 3:

The following switches must be added as compile-time defines on command line.

```
+define+SVT_FSDB_ENABLE
+define+SVT_PCIE_INCLUDE_AC_PA
```

Ensure that the Verdi tool version used should be 2015.09-1 or higher.

4. How do I configure various width and rate combinations for PIPE?

   See *svt_pcie_pl_configuration.sv* class:

   ```
   rand pclk_rate_enum  attribute
    svt_pcie_pl_configuration::pclk_rate[3] = '{ PCLK_250_MHZ, PCLK_500_MHZ, PCLK_1000_MHZ
   }
   ```

   The above specifies PCLK rate to be used at various PCIe data rates. pclk_rate[0] represents the rate used at 2.5 GT/s data rate. pclk_rate[1] represents the rate used at 5.0 GT/s data rate. pclk_rate[2] represents the rate used at 8.0 GT/s data rate.

   Also, see the description of `pipe_width` which summarizes various PIPE configurations and `pipe_width` and `pclk_rate` settings to achieve these PIPE configurations.

   ```
   rand pipe_width_enum  attribute
    svt_pcie_pl_configuration::pipe_width[4] = '{4{PIPE_8_BITS}}
   ```

   Specifies the PIPE width to be used at various data rates. pipe_width[0] represents the width used at 2.5 GT/s data rate. pipe_width[1] represents the width used at 5.0 GT/s data rate. pipe_width[2]

represents the width used at 8.0 GT/s data rate. pipe_width[3] represents the width used at 16.0 GT/s data rate.

A few examples of various legal various PIPE configurations and settings of `pipe_width` and `pclk_rate` to achieve these configurations.

1.Fixed width [8 bits] and variable PCLK PIPE configuration

```
[PIPE_8]
pipe_width[0] = PIPE_8_BITS; pclk_rate[0] = PCLK_250_MHZ; -- Achieve 2.5 GT/s
pipe_width[1] = PIPE_8_BITS; pclk_rate[0] = PCLK_500_MHZ; -- Achieves 5.0 GT/s
pipe_width[2] = PIPE_8_BITS; pclk_rate[0] = PCLK_1000_MHZ; -- Achieves 8.0 GT/s
```

2.Fixed width [16 bits] and variable PCLK PIPE configuration

```
 [PIPE_16]
pipe_width[0] = PIPE_16_BITS; pclk_rate[0] = PCLK_125_MHZ; -- Achieve 2.5 GT/s
pipe_width[1] = PIPE_16_BITS; pclk_rate[0] = PCLK_250_MHZ; -- Achieves 5.0 GT/s
pipe_width[2] = PIPE_16_BITS; pclk_rate[0] = PCLK_500_MHZ; -- Achieves 8.0 GT/s
```

3.Fixed width [32 bits] and variable PCLK PIPE configuration

```
[PIPE_32]
pipe_width[0] = PIPE_32_BITS; pclk_rate[0] = PCLK_62_5_MHZ; -- Achieve 2.5 GT/s
pipe_width[1] = PIPE_32_BITS; pclk_rate[0] = PCLK_125_MHZ; -- Achieves 5.0 GT/s
pipe_width[2] = PIPE_32_BITS; pclk_rate[0] = PCLK_250_MHZ; -- Achieves 8.0 GT/s
```

4.Fixed PCLK[250 MHz] and variable width PIPE configuration

```
[PIPE_250_MHZ]
pipe_width[0] = PIPE_8_BITS; pclk_rate[0] = PCLK_250_MHZ; -- Achieve 2.5 GT/s
pipe_width[1] = PIPE_16_BITS; pclk_rate[0] = PCLK_250_MHZ; -- Achieves 5.0 GT/s
pipe_width[2] = PIPE_32_BITS; pclk_rate[0] = PCLK_250_MHZ; -- Achieves 8.0 GT/s
```

5. Why does VIP give the following Warning?

```
"get_local_preset_coefficients asserted on multiple PCLK cycles on lane N before
previous coefficient request was completed"
```

Warning is issued by pcie_svt when the `getlocalpresetcoefficients` signal is asserted for more than one PCLK.

The PIPE spec says: A MAC holds this signal high for one PCLK cycle, requesting a preset to

co-efficient mapping for the preset on `LocalPresetIndex[3:0]` to coefficients on `LocalTxPresetCoefficient[17:0]`.

6. How do I resolve SERDES locking error in PCIe SVT?

```
UVM_ERROR: uvm_test_top.env.root.port0.pl0
[register_fail:AC_PL_LANE_SERDES:PROTOCOL:serdes_clk_slowdown_0] : New larger bit seen,
but not at least 2x old bit - clock has likely slowed down (was 0.071300, now is
0.072000 ns) - but SERDES wasn't locked yet.
```

This error can occur when jitter is added to SERDES and can be avoided by setting the following two parameters:

```
defparam *.port0.serdes0.ALLOW_RECOVERED_CLK_WIDTH_ADJUSTMENTS=1;

defparam *.port0.serdes0.CLK_TOLERANCE=0.000100; / / 100 PPM (default value)
```

This should be done for `*.port0.serdes0 through *.port0.serdesN`

where, N = number of lanes in the model minus 1 and "*" is the path one level below the VIP instance.

Note

The above mentioned approach to set the CLK_TOLERANCE may not work in some scenarios. The clock setting might not be taking the updated value always due to overridden.

In such scenarios, you can use an alternative approach to set the CLK_TOLERANCE value. You can perform the settings in the topology file along with "ALLOW_RECOVERED_CLK_WIDTH_ADJUSTMENTS" parameter setting. This ensures that the setting is complete for all lanes.

```
defparam spd_0.SVT_PCIE_UI_SERIAL_CLK_TOLERANCE = 0.0002;
```

For VIP to achieve serdes_lock, the following condition has to be satisfied:

```
(B/2) - (A/2)  < TOL                    ....... (a)
Where, TOL   =  (A/2) * CLK_TOLERANCE  ........ (b)
```

Value of A and B can be extracted from the following information available in the message log (where A=0.071300 and B=0.072000):

New larger bit seen, but not at least 2x old bit - clock has likely slowed down (it was 0.071300 and now it is 0.072000 ns)—but SERDES was not locked yet.

```
TOL in current case is (0.071300/2)*0.000100 = 0.000003565            ..... using
(b),
where, CLK_TOLERANCE is 100PPM as default value
........ (c)
Calculating LHS of (a): (B/2) - (A/2) => (0.072000/2)-(0.071300/2) =  0.00035
..... (d)
```

As 0.00035 !< 0.000003565, condition (a) is not satisfied, therefore, VIP serdes_lock is not achieved.

To make VIP serdes_lock, the CLK_TOLERANCE must be changed to 10,000 PPM using the following code:

```
defparam *.port0.serdes0.CLK_TOLERANCE = 0.010000;//10,000ppm
Which makes TOL as (0.071300/2)*0.010000 = 0.0003565 (>0.00035) satisfying
condition (a)
```

7. How to avoid lost valid signal error from serdes?

```
"UVM_ERROR :src/vcs/svt_message_manager.svp(349) @ 108302179980:
svt_message_manager.class [report_message] root0.port0.serdes0.: Lost valid signal level
on receiver, PLL (clock) recovery reset"
```

This message is issued by the serdes model of the PCIe SVT VIP whenever it loses a bit-lock with the stream of bits being received on the receiver. One common reason for this can be link speed change, in which case the PLL will lock to the new data rate.

In case of a speed change test, this error can be expected and must be suppressed.

8. How do I enable Spread Spectrum Clocking for PCIe VIP?

You can use ssc_mode attributes in the PHY layer configuration class (svt_pcie_pl_configuration) to set the SSC values:

Description of ssc_mode: Spread Spectrum Clocking mode for serial bit clock on the transmit path.

It is only applicable for VIP running with serial interface.

```
0 (default) - SSC_MODE_DISABLED => no ssc profile.
1 - SSC_MODE_DOWN => down spread spectrum.
```

```
2 - SSC_MODE_CENTER => center spread spectrum.
ssc_mode[0] - ssc_mode for Gen1 speed.
ssc_mode[1] - ssc_mode for Gen2 speed.
ssc_mode[2] - ssc_mode for Gen3 speed.
ssc_mode[3] - ssc_mode for Gen4 speed.
```

Example code to set the `ssc_mode` values:

```
cfg.ep_cfg.pcie_cfg.pl_cfg.ssc_mode[0] = svt_pcie_pl_configuration::SSC_MODE_DISABLED;
```

9. How to avoid "source code visibility" license check failures warning messages while compiling?

   By default, VCS checks for the necessary licenses for source code visibility feature. If this license is unavailable, then the following warning is displayed while compiling the source code:

   ```
   Warning-[VIPLCF] VIP license check failed
   VIP license check failed with "VIP-PCIe-VDB-SVT". Source code visibility will not be
   allowed to the protected code.
   ```

   The source code visibility license check failures warning messages can be suppressed by adding the following switch in VCS tool command line.

   ```
   +warn=noVIPLCF
   ```

   For example, in case of using VCS simulator in two-step flow,

   ```
   vcs +warn=noVIPLCF <other vcs options> <files to compile>
   ```

   
   Note This warning will not appear by default from 2017.03 or later versions of VCS compiler.

10. How to set SSC with 300PPM ~ -5300PPM?

    You can set SSC related configuration parameters to adjust SSC value.

    For more details, see `svt_pcie_pl_configuration::ssc_mode` attribute.

    ```
    cfg.rc_cfg.pcie_cfg.pl_cfg.ssc_mode[0] =
    svt_pcie_pl_configuration::SSC_MODE_DOWN_SPREAD; // Gen1
    cfg.rc_cfg.pcie_cfg.pl_cfg.ssc_mode[1] =
    svt_pcie_pl_configuration::SSC_MODE_DOWN_SPREAD; // Gen2
    cfg.rc_cfg.pcie_cfg.pl_cfg.ssc_mode[2] =
    svt_pcie_pl_configuration::SSC_MODE_DOWN_SPREAD; // Gen3
    cfg.rc_cfg.pcie_cfg.pl_cfg.ssc_mode[3] =
    svt_pcie_pl_configuration::SSC_MODE_DOWN_SPREAD; // Gen4
    // 5000 ppm
    cfg.rc_cfg.pcie_cfg.pl_cfg.ssc_max_spread[0] = 16'd5000; //Gen1
    cfg.rc_cfg.pcie_cfg.pl_cfg.ssc_max_spread[1] = 16'd5000; // Gen2
    cfg.rc_cfg.pcie_cfg.pl_cfg.ssc_max_spread[2] = 16'd5000; //Gen3
    cfg.rc_cfg.pcie_cfg.pl_cfg.ssc_max_spread[3] = 16'd5000; //Gen4
    ```

    To introduce the fixed variance in the Tx clocks of the VIP, you can program the following configuration attribute of the `svt_pcie_pl_configuration` class.

    Example code:

    ```
    cfg.rc_cfg.pcie_cfg.pl_cfg.fixed_ppm_due_to_tx_rx_xo
    ```

    
    Note Additionally, program the Tx SKP interval in the VIP or the Rx FIFO's in the DUT's receiver will run into an overflow or an underflow.

11. How to increase timeout of VIP?

    For most of the simulation, even if you increase `uvm_timeout`, test results in the testbench timeout.

    This is because most of the sequence will end with respect to `svt_timer` (if it is mentioned in the sequence).

    As the `svt_timer` expires at @1200000000.00ps, you may get the following error:

    ```
    @ 1200000000.00 ps: svt_timer:class [main] timer_total has expired.
    ```

    To run such test for more time, you can modify a particular sequence as shown in the following example:

    ```
    int test_timer = 2500000ns      //take a local variable
    timer_total.start_timer(test_timer,"Sequence Test Started",1,0); //when calling the
    start_timer pass the "test_timer"  (sequence timeout value ) instead of
    total_timeout_timer
    ```

12. How to change the supported PCIe specification version in the VIP?

    To change VIP PCIe version, set the following configuration attribute:

    ```
    vip_cfg.pcie_spec_ver = svt_pcie_device_configuration :: PCIE_SPEC_VER_2_0;
    ```

    Allowed values areas follows:

- `svt_pcie_device_configuration::PCIE_SPEC_VER_1_1, // configures the PCIE specification to version 1.1`
- `svt_pcie_device_configuration::PCIE_SPEC_VER_2_0, //  configures the PCIE specification to version 2.0`
- `svt_pcie_device_configuration::PCIE_SPEC_VER_2_1, //  configures the PCIE specification to version 2.1`
- `svt_pcie_device_configuration::PCIE_SPEC_VER_3_0, // configures the PCIE specification to version 3.0`
- `svt_pcie_device_configuration::PCIE_SPEC_VER_3_1, // configures the PCIE specification to version 3.1`
- `svt_pcie_device_configuration::PCIE_SPEC_VER_4_0, // configures the PCIE specification to version 4.0`

13. Is there a configurable parameter so that user can set the `total_timeout_timer` value through the test case rather than updating it in the sequence directly?

    Test timer `total_timeout_timer` is made configurable and added in configuration class. Now this can be controlled from the base test.

    ```
    /* Attribute to track total timer value of sequence */
      int total_timeout_timer = 1000000;
    ```

14. How to enable VIP to support Gen4?

    To enable Gen4 features, use the `SVT_PCIE_ENABLE_GEN4` define. The license manager checks for a Gen4 license when this define is set.

    To enable Gen4 protocol checks, set the specification version to 4.0.

    ```
    svt_pcie_device_configuration::pcie_spec_ver==svt_pcie_device_configuration::PCIE_SPEC_V
    ER_4_0;
    ```

    To initiate gen4 speed change:

    ```
    pl_cfg.set_link_speed_valuespl_cfg.set_link_speed_values(`SVT_PCIE_SPEED_16_0G |
    `SVT_PCIE_SPEED_8_0G |  `SVT_PCIE_SPEED_5_0G | `SVT_PCIE_SPEED_2_5G)
    ```

For more details on enabling Gen4 license, see the following SolvNetPlus article:

https://solvnet.synopsys.com/retrieve/2418353.html

15. How to disable and enable the VIP checks selectively for a certain time span while creating erroneous tests?

    While exercising any erroneous scenario, where link partner is expected to send some corrupted data, for which the VIP throws an error or warning. In that case, you may turn off the respective checks completely or may turn off the check for certain time span and again turn them on. This should be carried out during the run-time.

    You can turn off the check by selecting the appropriate group name, sub-group name, and protocol check instance name.

    For more details on the VIP's protocol checks, see the *HTML Class Reference* document.

    https://solvnet.synopsys.com/dow_retrieve/latest/snps_vip_lib/doc/pcie_svt_uvm_class_reference/html/protocolChecks.html

    VIP's checkers can be turned off or on selectively by using the following method:

    - For disabling check:

      ```
      void'(<vip_agent>.pcie_agent.err_check.disable_checks("<Group_name>","<Sub-group name>","^<protocol_check_instance_name>"));
      ```

    - For enabling check:

      ```
      void'(<vip_agent>.pcie_agent.err_check.enable_checks("<Group_name>","<Sub-group name>","^<protocol_check_instance_name>"));
      ```

    For example:

    ```
    //disabling the check.
    void'(root.pcie_agent.err_check.disable_checks("^ACTIVE_DL","^REPLAY","^dl_replay_timer_timeout"));
    //enabling the check.
    void'(root.pcie_agent.err_check.enable_checks("^ACTIVE_DL","^REPLAY","^dl_replay_timer_timeout"));
    ```

16. How do I reconfigure the key configuration attributes of VIP within a single simulation?

    You need to consider the following points when reconfiguring the key configuration attributes of VIP:

    - LTSSM state of VIP instance whose configuration is being changed must be in Detect Quiet, Hot Unplug state.
    - If the VIP instance is representing an SPIPE/ MPIPE instance, then the configuration should be done when `pipe_reset_n` is asserted.

    Code Snippet:

    ```
    // Example code to bring VIP into Detect_Quiet Unplug state
    svt_pcie_ts_pl_service_hot_plug_mode_sequence vip_hot_plug_unplug_sequence;
    `svt_xvm_do_on_with(vip_hot_plug_unplug_sequence,vip_seqr.pcie_virt_seqr.pl_seqr,

    { hot_plug_mode == svt_pcie_pl_service::HOT_PLUG_UNPLUG;}
    )
    wait (vip_status.pcie_status.pl_status.ltssm_state != svt_pcie_types::DETECT_QUIET)
    ```

```
// Example code to assert pipe_reset_n
svt_pcie_pl_service pl_service;
`svt_xvm_do_on_with(pl_service,vip_seqr.pcie_virt_seqr.pl_seqr,

{service_type == svt_pcie_pl_service::PIPE_INJECT_RESET;event_delay==0;event_width==-1;}
)

// Safe duration to reconfigure VIP

// Example code to de-assert pipe_reset_n
svt_pcie_pl_service pl_service;;
`svt_xvm_do_on_with(pl_service,vip_seqr.pcie_virt_seqr.pl_seqr,

{service_type == svt_pcie_pl_service::PIPE_INJECT_RESET;event_delay==0;event_width==10;}
// Example code to resume VIP LTSSM
svt_pcie_ts_pl_service_hot_plug_mode_sequence vip_hot_plug_unplug_sequence;
`svt_xvm_do_on_with(vip_hot_plug_unplug_sequence,vip_seqr.pcie_virt_seqr.pl_seqr,

{ hot_plug_mode == svt_pcie_pl_service::HOT_PLUG_DETECT;}
```

17. How do I resolve the "elastic buffer overflowed on lane" error?

    The basic reason for this error to occur is the improper setting of the SKP intervals.

    Approach 1:

    Set the SKP intervals as suggested for resolving the "elastic buffer overflowed on lane" error:

    - If SRIS is enabled, then the following settings are applicable:

      ```
      min_tx/rx_skp_interval_in_symbol_times = 1
      max_tx/rx_skp_interval_in_symbol_times = 153
      min_tx/rx_skp_interval_in_blocks = 1
      max_tx/rx_skp_interval_in_blocks = 153
      ```

    - If SRIS is enabled, then the following settings are applicable:

      ```
      min_tx/rx_skp_interval_in_symbol_times = 1180
      max_tx/rx_skp_interval_in_symbol_times = 1538
      min_tx/rx_skp_interval_in_blocks = 1180
      max_tx/rx_skp_interval_in_blocks = 1538
      ```

    For more information, see the SolvNetPlus article 000019114, "VC VIP: Blocking and Transmitting SKP Ordered Sets in the PCIe VIP".

    Approach 2:

    If all the above settings are correct and you still get this error, then you need to check if the following signals are working properly or not:

    - When "RxElecidle" is asserted, "Rxvalid" should de-assert and vice-versa.

      This is because the delay between RxElecIdle being asserted and RxValid being de-asserted is directly related to the depth of implementation of elastic buffer and symbol synchronization logic which leads to an overflow condition.