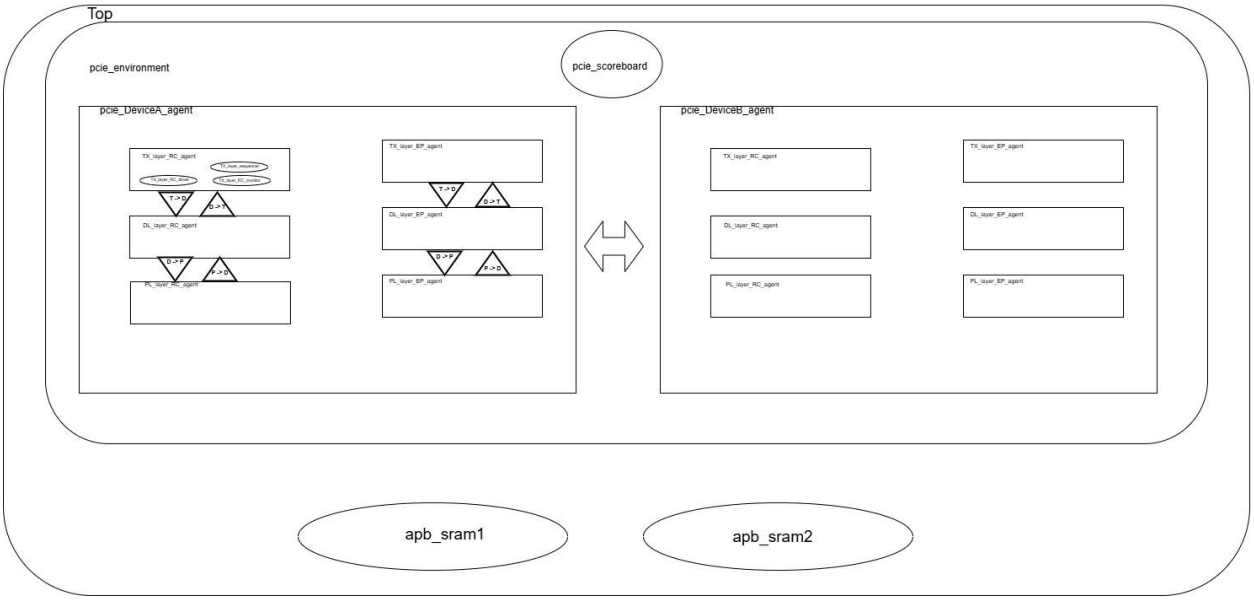
GEN6 VIP DEVELOPMENT

Gen6 VIP structure:



CODING STRUCTURE:

Configuration space and enumeration process through APB:

1) PCIe Agents:
   - RC & EP agents :for accessing configuration space from both EP and RC side

   - **RC APB Agent**:
   - APB_SRAM1 -> APB SRAM RTL
   1) Hierarchy: RC_APB_TEST -> RC_APB_AGENT -> RC_APB_ENV -> RC_APB_AGENT -> RC_APB_DRV & RC_APB_SQR

   2) RC_APB_SEQ_LIB: Sequences are defined for enumeration process which comprises of different memory based txns as follows:

- rc_apb_seq

- rc_apb_wr_rd_seq
- rc_apb_wr_seq
- rc_apb_rd_seq
- rc_apb_wr_rd_spec_seq
- rc_apb_rd_wr_seq
- rc_apb_wr_error_seq
- rc_apb_rd_error_seq
- rc_apb_n_wr_n_rd_seq
- rc_apb_wr_rd_byte_seq

3) Apb1_top

4) RC_APB_TEST_LIB1 -> It Consist of following tests
   - rc_apb_test rc_apb_wr_rd_test
   - rc_apb_wr_test
   - rc_apb_rd_test rc_apb_wr_rd_spec_test
   - rc_apb_rd_wr_test
   - rc_apb_wr_error_test
   - rc_apb_rd_error_test
   - rc_apb_n_wr_n_rd_test
   - rc_apb_wr_rd_byte_test

5) RC_APB_ENV:
- RC_APB_AGENT is instance inside this class currently

6) RC_APB_AGENT:
- RC_APB_SQR & RC_APB_DRV are present inside rc agent currently

7) RC_APB_SQR

8) RC_APB_DRV:
- For enumeration process, sending configuration txn via APB signals from the sequencer and driving to APB1 slave
- This is done through apb_intf1(RC_APB_INTF)
- Here its just getting the vif handle from top and driving apb signals for enumeration process

9) Coverage and assertion class is separately implemented

10) RC side SBD is also implemented which is comparing expected and actual APB transaction

EP APB Agent:

1) Hierarchy: EP_APB_ENV -> EP_APB_AGENT -> EP_APB_DRV & EP_APB_SQR
- APB_SRAM2 -> APB SRAM RTL
2) Hierarchy: EP_APB_TEST -> EP_APB_AGENT -> EP_APB_ENV -> EP_APB_AGENT -> EP_APB_DRV & EP_APB_SQR

3) EP_APB_SEQ_LIB: Sequences are defined for enumeration process which comprises of different memory based txns similar to RC APB

4) Apb2_top
5) EP_APB_TEST_LIB1 -> It Consist of tests similar to RC side

6) EP_APB_ENV:
- EP_APB_AGENT is instance inside this class currently

7) EP_APB_AGENT:

- EP_APB_SQR & EP_APB_DRV are present inside rc agent currently

8) EP_APB_SQR
9) RC_APB_DRV:
- For enumeration process, sending configuration txn via APB signals from the sequencer and driving to APB1 slave
- This is done through apb_intf1(RC_APB_INTF)
- Here its just getting the vif handle from top and driving apb signals for enumeration process

10)Coverage and assertion class is separately implemented
11)SBD implemented similar to RC side, to compare expected and actual data

---

ACTUAL ENV & DATA FLOW:

1) **DEVICE_AGENT(DIRECTORY) -> PCIe_agent(TX_LAYER_RC_AGENT)**

-> It comprises of DeviceA and DeviceB agent. It comprises of the following class separately for TL, DL and PL layer which are as follows:

-> It consist of following instances:

a) TX_LAYER_SEQUENCER TL_RC_SEQUENCER

b) TX_LAYER_RC_DRIVER TL_RC_DRIVER

c) TX_LAYER_RC_MONITOR TL_RC_MONITOR

PCIe_agent consist of the following sub-agents:

- Tl layer rc agent -> Here tl rc driver is connected with tl rc monitor
- Dl layer rc agent -> Here dl rc driver is connected with dl rc monitor
- Pl layer rc agent -> Here pl rc driver is connected with pl rc monitor
- Tl layer EP agent -> Here tl rc driver is connected with tl rc monitor
- DL layer EP agent -> Here dl rc driver is connected with dl rc monitor
- PL layer EP agent -> Here pl rc driver is connected with pl rc monitor
- Pcie_deviceA_agent -> parameterized class with agent type passed as a paramater for differentiating RC & EP agents
- Pcie_deviceB_agent -> parameterized class with agent type passed as a paramater for differentiating RC & EP agents

2) **deviceA and deviceB** agents where all above RC and EP agents for all 3 layers are separately present

3) **ENV(directory) -> pcie_envrionment**

It is the top most environment which comprises of **DeviceA and DeviceB** agent types, and scoreboard

4) In the **pcie scoreboard(TX_layer_scoreboard)**, the RC and EP TLPs and configuration completion packets are compared for mismatches

- Here monitor ports from both tx and rx sides are connected to ep and rc ports in scoreboard separately which is implementing specific
- TLP packets from TX and RX sides monitors are checked here in pcie_scoreboard
- Write_rc and write_ep function is implemented from both monitors of
- In write function: ‘TX_layer_sequence_items’ are passed signifying transaction layer TLP packets comparison in scoreboard

5) **TX_layer_sequencer:** parameterized sequencer used

6) **TX_layer_sequence:** class present but not implemented anything, NYI

7) **TX_layer_sequence_item:**

- Here struct is defined for varios header types like config_header_field, mem_header_field, locked_memory_header_field, io_header_field, message_header_field, atomicop_3dw_header_field, atomicop_4dw_header_field
- Here all header types are defined which consist of all fields as per header defined
- Different packet_types are taken and are passed with some integers like:
  1) Config pkt
  2) Memory pkt
  3) Io packet
  4) Lock_mem pkt
  5) Msg_pkt
  6) 3dw_ap_pkt
  7) 4dw_ap_pkt
- Different tx are defined as per pkt_type defined like message transactions are defined
- 3dw mem txs
- Io txn
- 4dw mem tx
- Msg_tx(pkt_type == 5)
- 3dw atomicop tx
- 4dw atomic op tx
- Post_randomize function, several packets are created in TL layer which are transmitted to RX side
- Several constraints are implemented pertaining to several packet types which are implementing specific

8) **TL layer sequence**: Different TL layer sequences are already implemented

9) **TL_TOP_seq**: Includes all sequences for TL layer present in TL_layer_sequence directory

10) **RC_USER_DERIVED** -> callback class extended from PCIE_RC_DRV_CALLBACK. Here pre and post driver is implemented, post driver is empty and corruption is seen in pre driver

- Here we are corrupting various fields of TLP packet(mem, io and cfg packets) by passing some random numbers to various fields like format, type, first dw adn last dw byte enable, TC, attribute , TH , AT, etc

11) **PCIE_RC_DRV_CALLBACK**: empty class with virtual functions implemented which are overriden in child class RC_USER_DERIVED

12) **RC_AGENT:** A directory

- **TX_LAYER_RC_DRIVER:**

- **TX_LAYER_RC_MONITOR:**
  1) Its top level monitor, which consist of virtual interface handles of RC side
  2) The two handles are TL_to_DL_RC_interface(RC) for putting packets in scoreboard which are transmitted
  3) DL_to_TL_RC_interface handle for capturing packets to sbd which are rcvd basically and are completion packets as implemented here
  4) One analysis port is defined namingly **item_collected_port_rc_side**
  5) Here 2 tasks are defined(rc_side_pkt_collect & rc_side_cpl_collect) for collecting packets which are transmitted and rcvd(completion) separately, and written using write function which are sent scoreboard for comparing actual and expected packets

13) **EP_AGENT**: A directory

- **TX_LAYER_EP_DRIVER**
- **TX_LAYER_EP_MONITOR:**
  1)Its top level monitor, which consist of virtual interface handles of EP side
  2) The two handles are TL_to_DL_EP_interface(EP) for putting packets in scoreboard which are transmitted
  3) DL_to_TL_EP_interface handle for capturing packets to sbd which are rcvd basically and are completion packets as implemented here
  4) One analysis port is defined namingly **item_collected_port_ep_side**
  5) Here 2 tasks are defined(ep_side_pkt_collect & ep_side_cpl_collect) for collecting packets which are transmitted and rcvd(completion) separately, and written using write function which are sent scoreboard for comparing actual and expected packets

  TL layer all implemented and checked in doc file, NOW DL_LAYER, PL_LAYER ENV -> pcie_env -> DONE

14) TOP:

- Here all interfaces are instantiated from APB to every layer wise
- Both apb memory sram is instantiated for configuration transactions required during enumeration process

15) TEST :

- TL_TEST: A directory consisting of all TL layer tests already implemented
- pcie_base_test: Here both rc_apb and ep_apb is taken, pcie_environment is taken and base test is created
- top_test: All TL layer tests are included in this file

16) DL_LAYER:

1) RC_AGENT(DIRECTORY)

- DL_layer_RC_monitor
- DL_layer_RC_driver

2) EP_AGENT (DIRECTORY)

- DL_layer_RC_monitor
- DL_layer_RC_driver

3) DL_layer_sequencer

4) DL_layer_sequence_item

5) DL_layer_sequence

6) DL_layer_scoreboard

17) INTF: It consist of all layer wise interfaces for TL, DL and PL for communication between all layers to and fro. All interfaces defined are as follows: Its description is code specific

- TL_to_DL_RC_interface
- TL_to_DL_EP_interface
- PL_to_DL_RC_interface
- PL_to_DL_EP_interface
- PL_T_to_PL_R_interface
- PL_R_to_PL_T_interface
- DL_to_TL_RC_interface
- DL_to_TL_EP_interface
- DL_to_PL_RC_interface
- DL_to_PL_EP_interface

18) Integrate file: The top file for entire PCIe env coded. All components developed are included here, including top file which is separtae file consisting of interface handles and design instantiation related stuffs, uvm_pkg, macros, etc

Different enums are defined for various message types, prefix type, etc. Also crc generator is implemented which could generate both 16 and 32 bit CRC

19) PL_LAYER: Most items are not implemented or either empty functions consisting of simple uvm_infos or displays, going forward will be planned and implemented accordingly

1) RC_AGENT(DIRECTORY):

- PL_layer_RC_monitor
- PL_layer_RC_driver

2) EP_AGENT(DIRECTORY):

- PL_layer_RC_monitor
- PL_layer_RC_driver

3) PL_layer_sequencer

4) PL_layer_sequence_item

5) PL_layer_sequence

6) PL_layer_scoreboard

20) Test flow implementation:

Testname: pcie_config_n_wr_rd_test

- Firstly LTSSM comes into picture but that is not implemented currently
- Then enumeration happens through APB for which normal APB SRAM is taken for now, and the configuration space needs to be updated targetting various registers, for BDF, etc. This is taken care here in curreny VIP development phase
- After that TLP packets are sent from TL layer once device is ready by enumeration
- Here from the test APB transactions are initiated both from RC and EP sides
- Parallelly TL/DL/PPL_RC/EP sequences are fired in their respective agents sequencer from both sides, after the configuration transactionn are initiated and device enumeration is done properly. Its proper device enumeration as several registers need to be implemented in configuration space which is NYI, which needs to be read and write usign configuration txns
- In the top, RC_GOOD_PKT is defined, the APB transactions initiated is controlled through this in the pcie_config_n_wr_rd_sequence
- First the drivers of the respective slaves from both RC & EP sides are driving the APB transactions for enumeration process to initiate after the LTSSM
- After the APB write transactions are done from both sides, the prints basically shows the good packets carried by TX_layer_RC_driver TX_LAYER_EP_DRIVER also
- The TX_layer_RC_monitor and TX_layer_EP_monitor captures the good packet from the interface
- After the APB transactions are done, completions are sent from both RC and EP sides and that is captured by their respective monitors(TX_layer_RC_monitor and TX_layer_EP_monitor to check in scoreboard

- The registers in the congif space are written with data sent via configuration transactions
- In the TX_layer_scoreboard, RC to EP configuration packets and EP to RC completions packets are compared
- Both RC to EP and completions from EP to RC side actual adn expected packets are compared. Here RC to EP TLP packet is taken for which ‘RC_TLP_pkt_exp’ and ‘EP_TLP_pkt_act’ are compared
- For completions, EP to RC completions are compared in scoreboard for which actual packet is RC_TLP_comp_act and expected one is EP_TLP_comp_exp.
- Upon successfull comparison, the test passes