PCIe Logical Device Driver
Verification Plan

Change History

Version No	Date	Change Detail	Author
0.1	02nd Nov 2020	Initial version	Vayavya Labs

Table of Contents

1. Introduction

This document describes the verification of a PCIe host controller interface using a logical device driver. The verification will be achieved by having unit level test cases for

a. PCIe host
b. PCIe host connected to PCIe devices

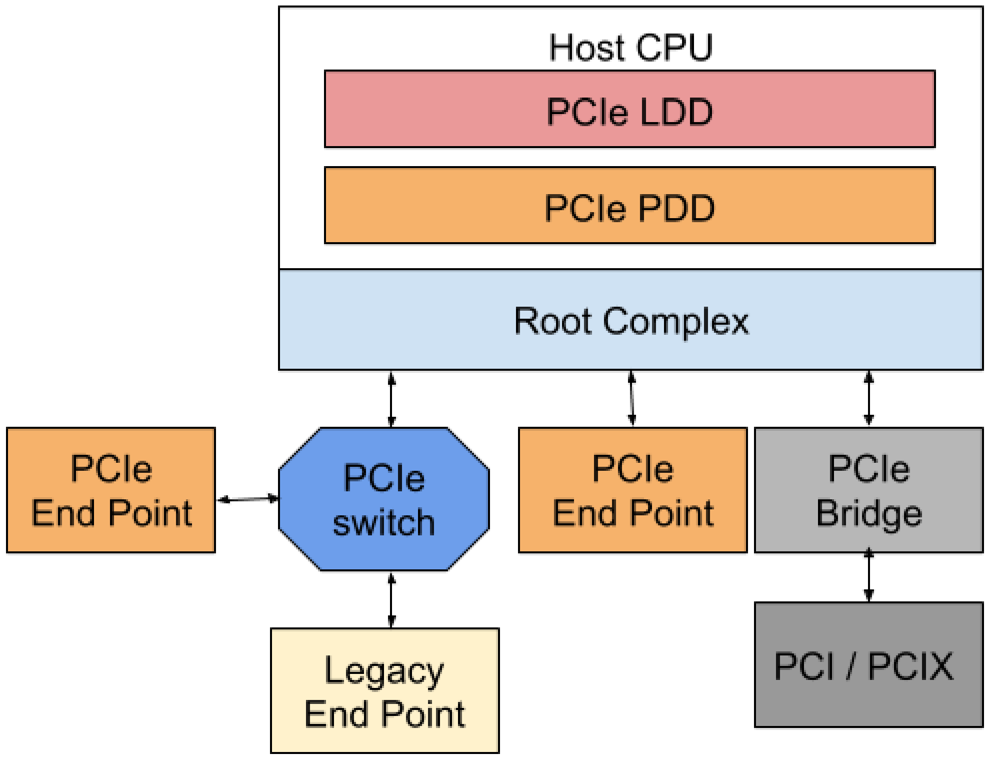
The testing will happen in its respective environment

References

[1] PCie LLD driver architecture document

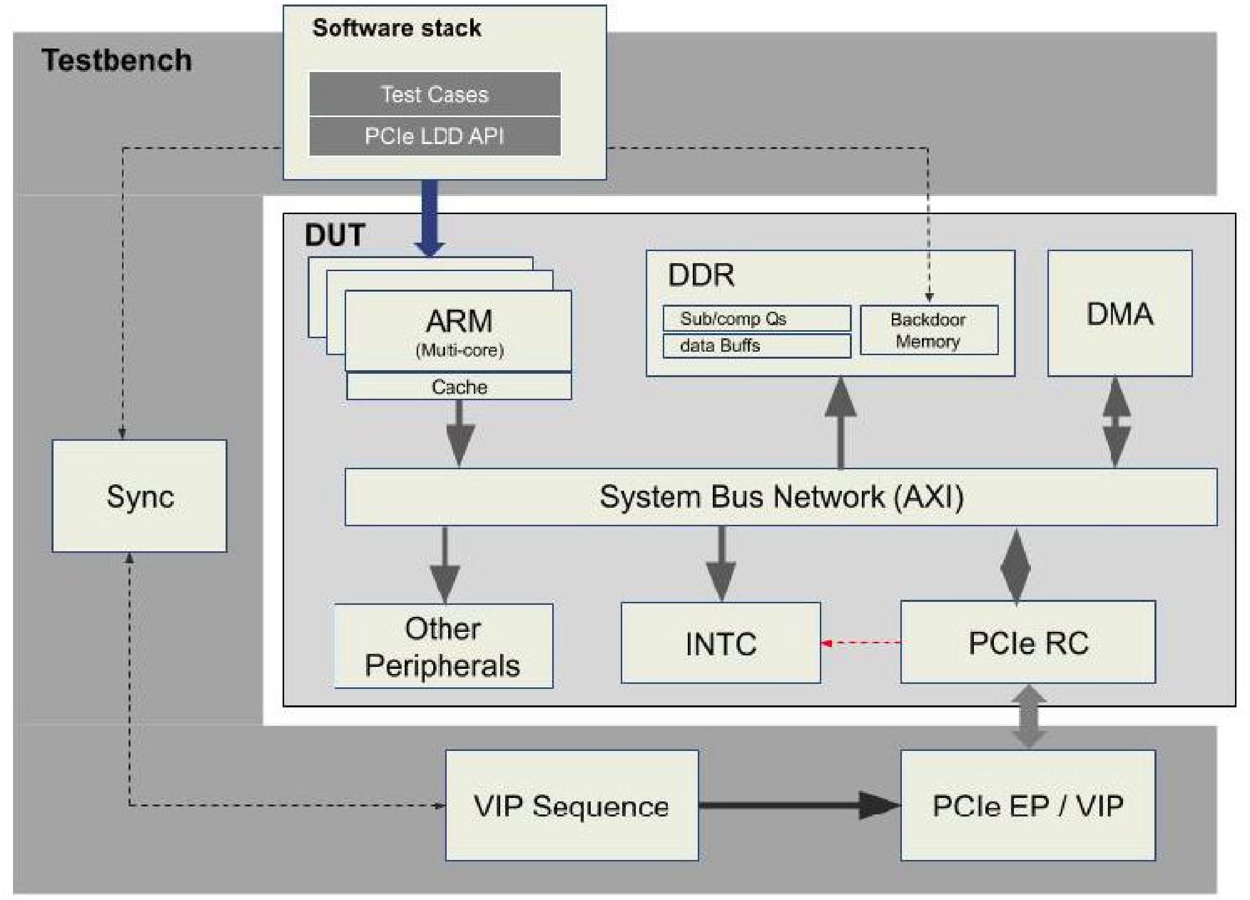
2. PCIe Logical Device Driver architecture overview

PCIe is a high-speed serial computer expansion bus standard. It is the common motherboard interface for personal computers' graphics cards, hard drives, SSDs, Wi-Fi and Ethernet hardware connections. PCIe has numerous improvements over the older standards, including higher maximum system bus throughput, lower I/O pin count and smaller physical footprint, better performance scaling for bus devices, a more detailed error detection and reporting mechanism (Advanced Error Reporting, AER) and native hot-swap functionality. More recent revisions of the PCIe standard provide hardware support for I/O virtualization.



A logical device driver is a high level abstraction of how a device behaves. Physical device driver is a driver for a specific piece of hardware. The logical device driver talks to the physical device driver for you to keep you abstracted from the underlying hardware.

3. UFS Host Controller Validation using LDD



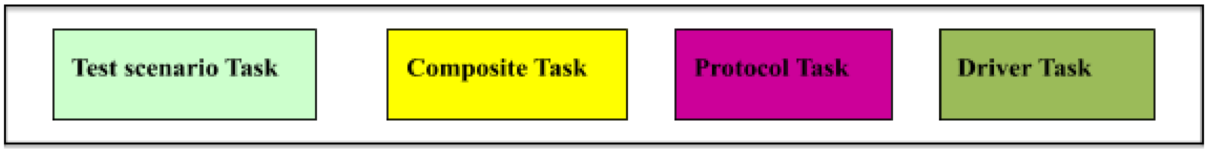
4. Test Scenarios

A test scenario is created using a tool called Platform Architecture Ultra (PAU) by connecting together tasks from the composite, protocol or driver layers to test the functionality of the PCIe subsystem. Each test scenario has a unique name and is saved in the repository as an XML file. To run the test scenario, it needs to be compiled successfully on the PAU platform. This will create an ".elf" file that can be executed on the VDK environment by specifying the executable file path. The test result will be displayed on the console window of the VDK.

The different test scenarios implemented are:

1. Pcie_enum_scenario
2. Pcie_read_ep_config_scenario
3. Pcie_rc_register_read_write_scenario
4. Pcie_rc_read_ep_mem
5. Pcie_rc_link_management_scenario
6. Pcie_rc_band_width_management_scenario
7. Pcie_test_msi_intr
8. Pcie_test_msix_intr
9. Pcie_test_msix_properties
10. Pcie_rc_read_ep_mem_multi_bytes

Note: Color coding of the boxes in the flow diagrams has the following meaning:

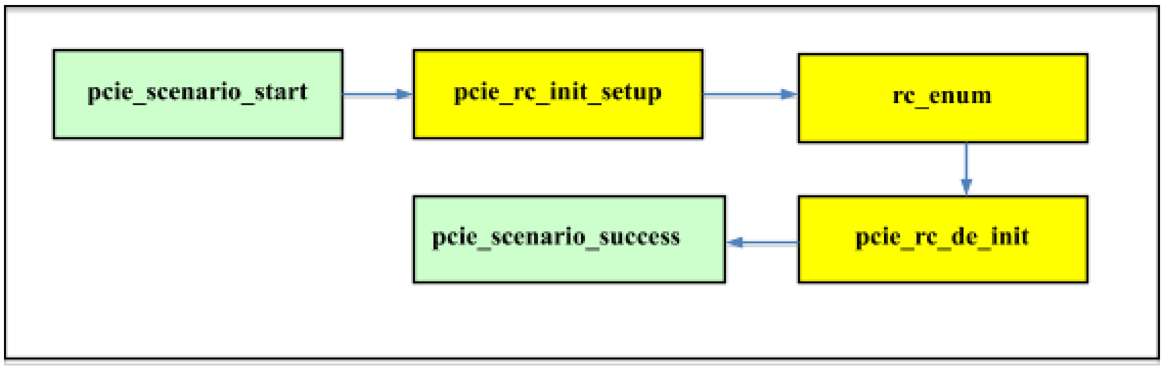


4.1. Test scenario - Pcie_enum_scenario.xml

Brief description: This test scenario is to test the PCIe enumeration.

It configures the PCIe controller driver and performs the PCIe enumeration by scanning the PCIe fabric to discover the machine topology and learn how the fabric is populated. Once the enumeration is completed, it is reported as a test passed.

Flow diagram of tasks:

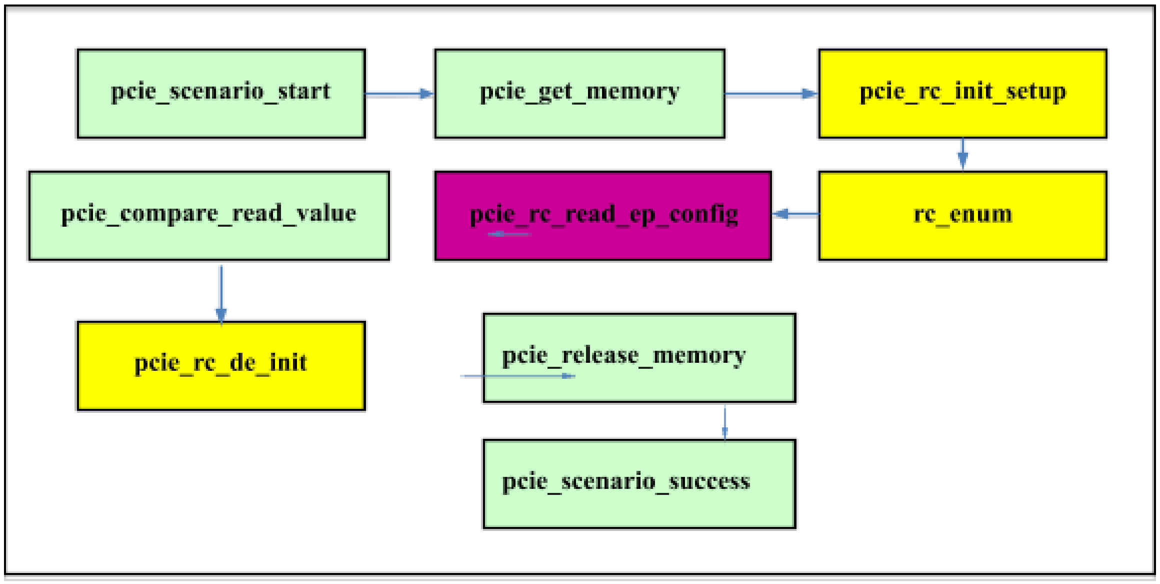


4.2. Test scenario - Pcie_read_ep_config_scenario.xml

Brief description: This test scenario is to test the PCIe endpoint configuration read. The read can be done only after enumeration is done.

It configures the PCIe controller driver and performs the PCIe enumeration. Once the enumeration is completed, It reads the configuration area (i.e.,Vendor ID & Device ID) of the EP & then read value is compared with the expected value to confirm configuration read is successful.

Flow diagram of tasks:

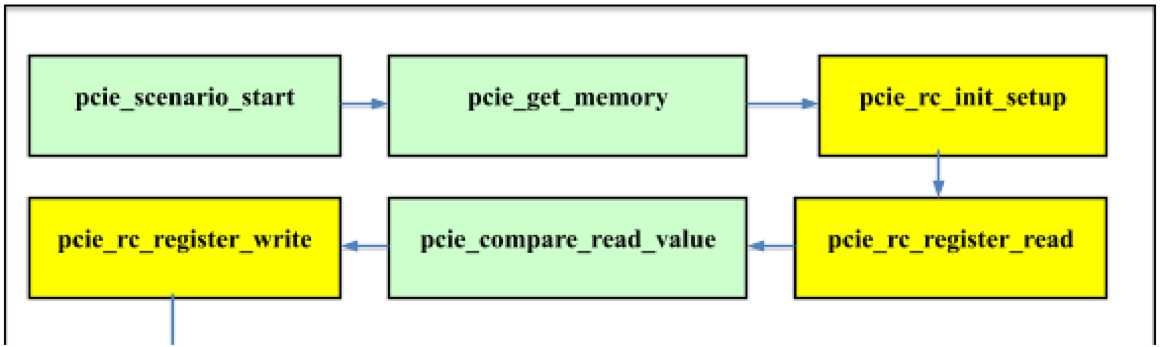


4.3. Test scenario - Pcie_rc_register_read_write_scenario.xml

Brief description: This test scenario is to test the PCIe Root complex Register read and write.

It writes a value into the root complex register and then reads the same root complex register. Theread value will be compared with the written value to confirm if the root complex register read and write happened successfully.

Flow diagram of tasks:



4.4. Test scenario - Pcie_rc_read_ep_mem.xml

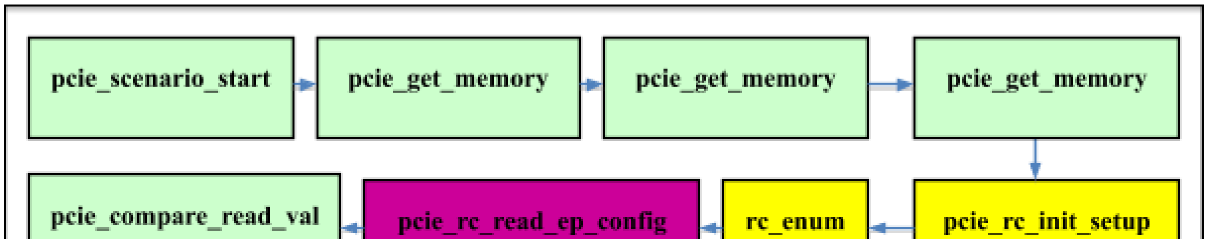
Brief description: This test scenario is to test the PCIe endpoint memory read.

It will read the memory region associated with the BAR registers of the EP.

Note: In order to ensure a EP memory read & write we have restricted the enumeration to one device.

MAX_DEV_CNT - Macro to change the device count .

Flow diagram of tasks:



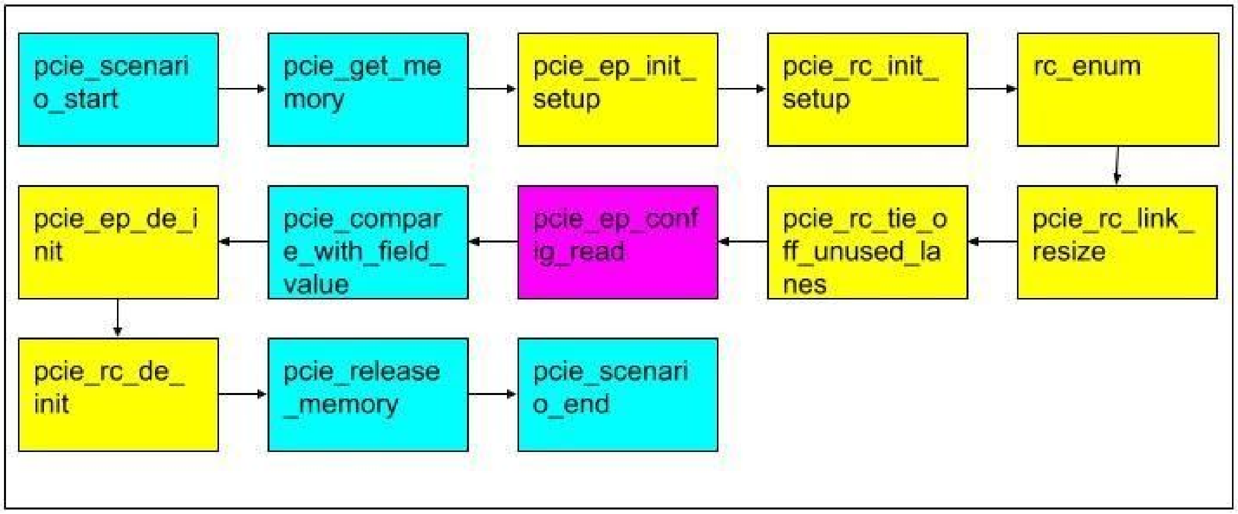
Note:

1. Please refer to the color coding information for type of tasks
2. Details of the different tasks executed in this test scenario as per the flow diagram of tasks are given in the subsequent chapters on “Test scenario tasks”, “Composite tasks”,“Protocol layer tasks” and “Driver layer tasks”.

4.5. Test scenario - Pcie_rc_link_management_scenario.xml

Brief description: This test scenario is for link management. Here RC is requesting for the lane resize & tying off the unused lanes. If EP supports the requested lane size it automatically does lane resize to the requested lane & tie off the unused lanes automatically at EP side else it will reject the lane resize request of the RC. Once this is done test will read & compare EP register field value (PCIE_CAP_NEGO_LINK_WIDTH of register LINK_CONTROL_LINK_STATUS_REG) with requested lane size to confirm lane resize request has been done or not.

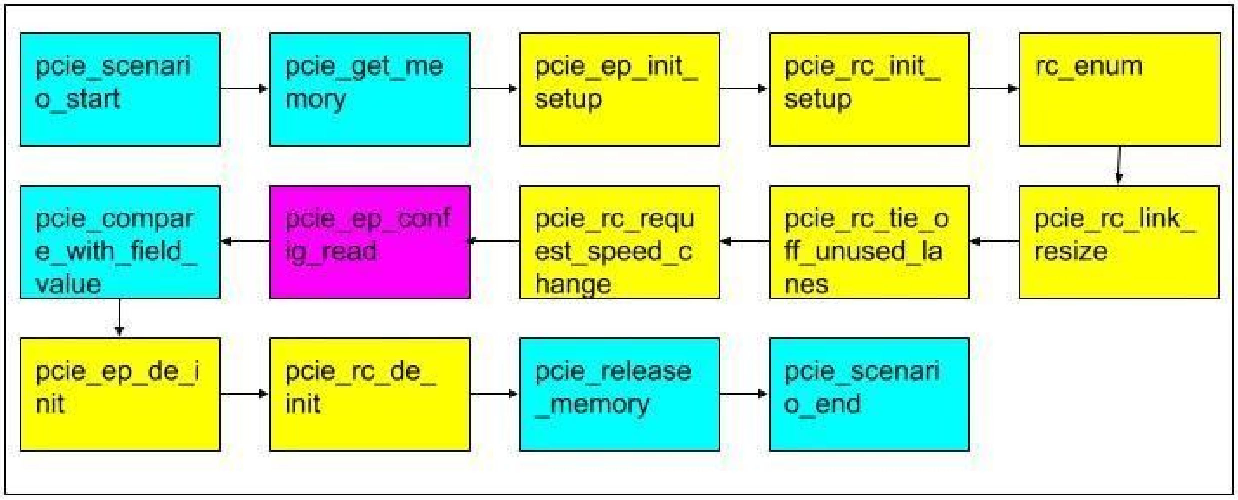
Flow diagram of tasks:



4.6. Test scenario - Pcie_rc_band_width_management_scenario.xml

<u>Brief description</u>: This test scenario is for bandwidth management. Bandwidth changes with change in the lane size & speed. Here RC is requesting for the lane resize & speed change. If EP supports the requested lane size & speed it automatically does lane resize to the requested lane & switches to that speed else it will reject the request of the RC. Once this is done test will read & compare RC register field value (PCIE_CAP_NEGO_LINK_WIDTH & PCIE_CAP_LINK_SPEED of register LINK_CONTROL_LINK_STATUS_REG) with requested lane size & speed to confirm lane resize & speed request has been changed or not.

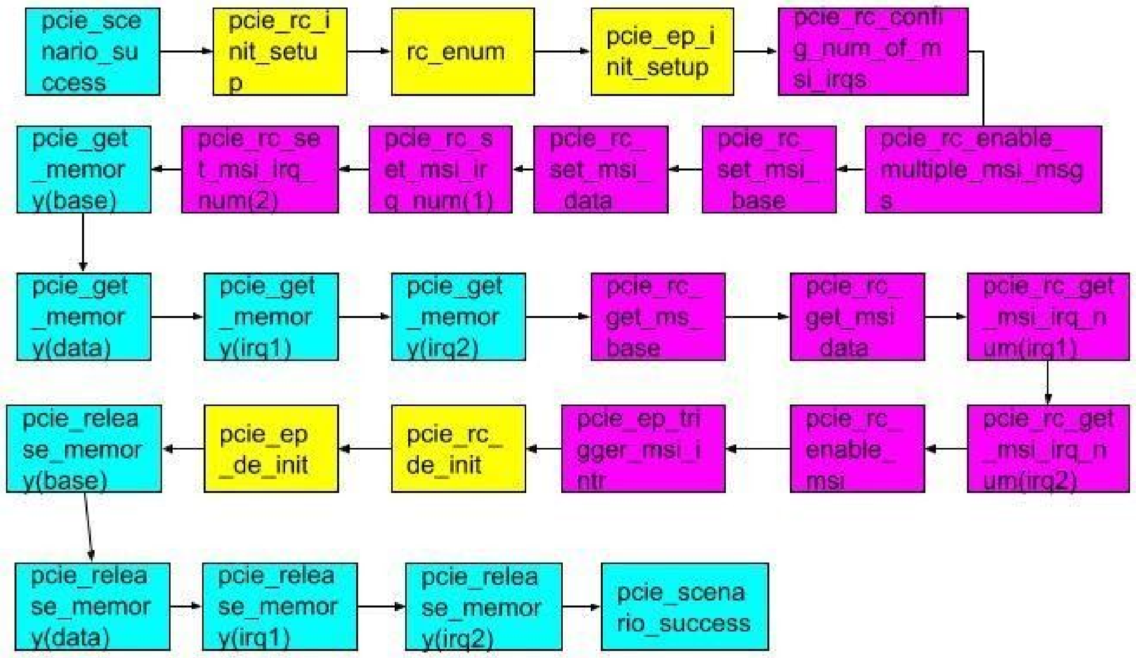
<u>Flow diagram of tasks:</u>



4.7. Test scenario - Pcie_test_msi_intr.xml

Brief description: Test scenario for MSI Interrupts. Configure the EP from RC and trigger the interrupt on EP.

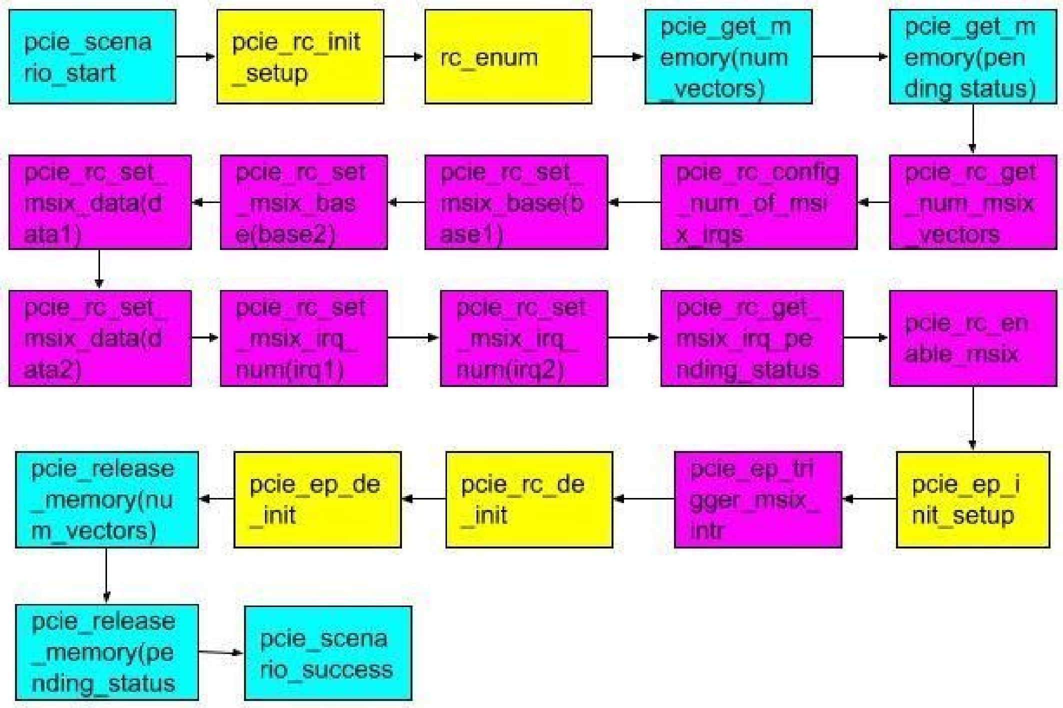
Flow diagram of tasks:



4.8. Test scenario - Pcie_test_msix_intr.xml

Brief description: Test scenario for MSIX Interrupts. Configure the EP from RC and trigger the interrupt on EP.

Flow diagram of tasks:



pcie_scenario_start
pcie_rc_init_setup
rc_enum
pcie_get_memory(num_vectors)
pcie_get_memory(pending status)
pcie_rc_get_num_msix_vectors
pcie_rc_config_num_of_msix_irqs
pcie_rc_set_msix_base(base1)
pcie_rc_set_msix_base(base2)
pcie_rc_set_msix_data(data1)
pcie_rc_set_msix_data(data2)
pcie_rc_set_msix_irq_num(irq1)
pcie_rc_set_msix_irq_num(irq2)
pcie_rc_get_msix_irq_pending_status
pcie_rc_enable_msix
pcie_ep_init_setup
pcie_ep_trigger_msix_intr
pcie_rc_de_init
pcie_ep_de_init
pcie_release_memory(num_vectors)
pcie_release_memory(pending_status
pcie_scenario_success

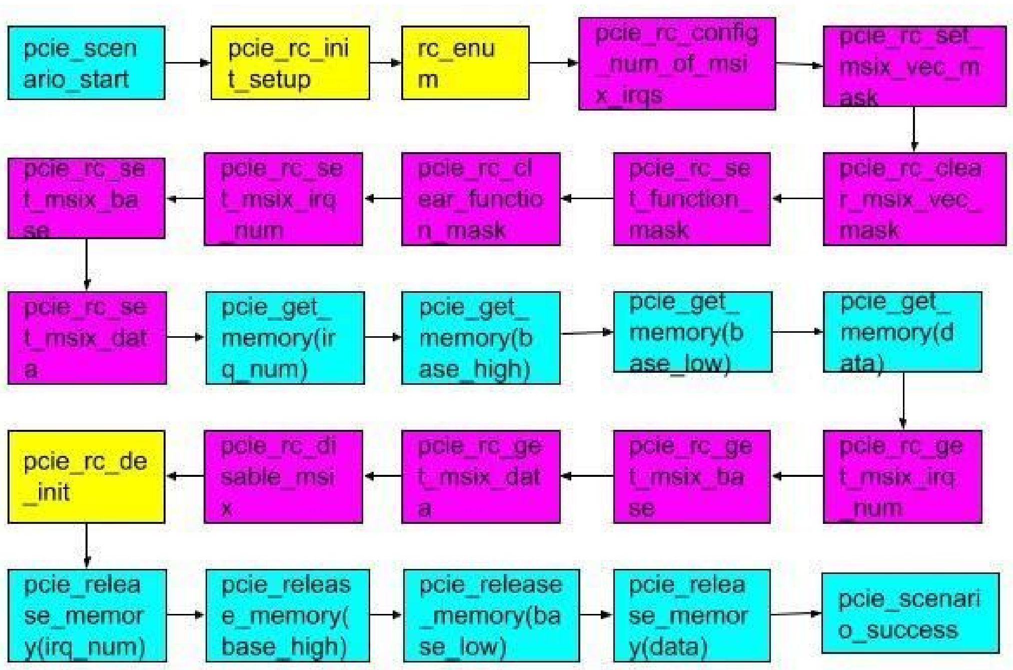
4.9. Test scenario - Pcie_test_msix_properties.xml

Brief description: Test msix functions.

Set and clear msix vector mask , function masks.

Set and get base, data and irq number.

Flow diagram of tasks:



pcie_scenario_start
pcie_rc_init_setup
rc_enum
pcie_rc_config_num_of_msix_irqs
pcie_rc_set_msix_vec_mask
pcie_rc_clear_msix_vec_mask
pcie_rc_set_function_mask
pcie_rc_clear_function_mask
pcie_rc_set_msix_irq_num
pcie_rc_set_msix_base
pcie_rc_set_msix_data
pcie_get_memory(irq_num)
pcie_get_memory(base_high)
pcie_get_memory(base_low)
pcie_get_memory(data)
pcie_rc_get_msix_irq_num
pcie_rc_get_msix_base
pcie_rc_get_msix_data
pcie_rc_disable_msix
pcie_rc_de_init
pcie_release_memory(irq_num)
pcie_release_memory(base_high)
pcie_release_memory(base_low)
pcie_release_memory(data)
pcie_scenario_success

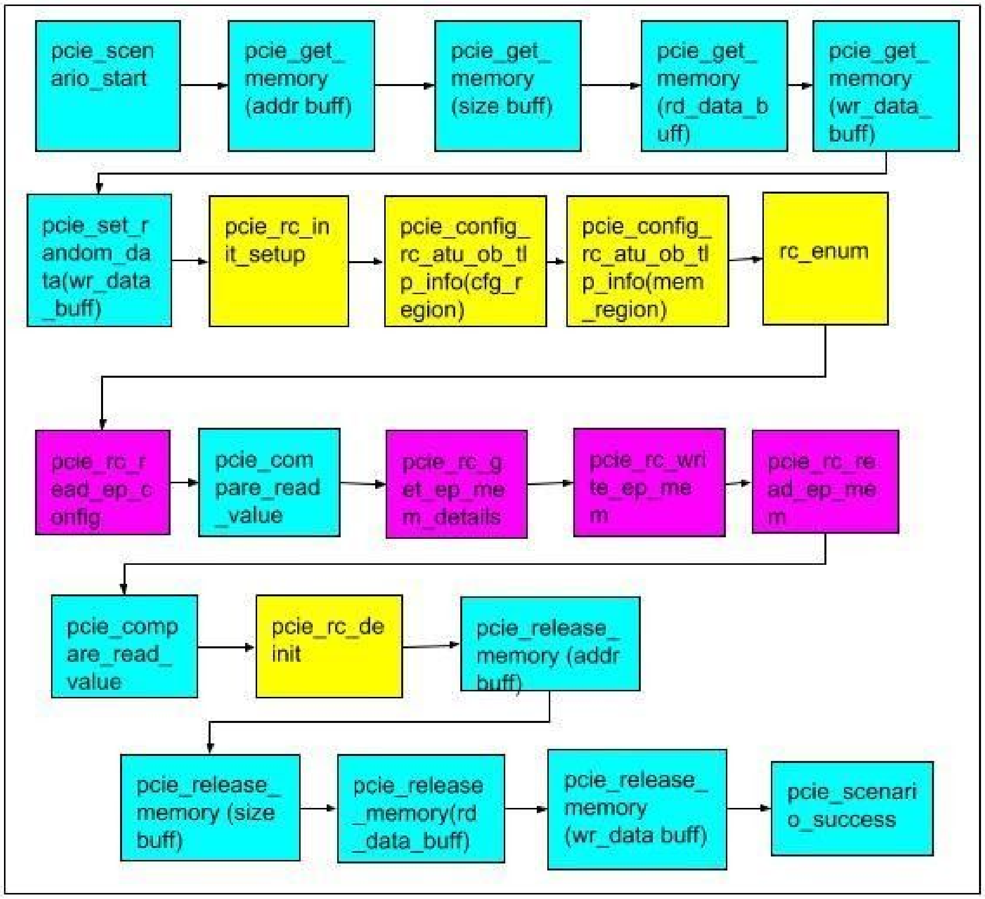
4.10. Test scenario - Pcie_rc_read_ep_mem_multi_bytes.xml

Brief description: This test scenario will read the memory region associated with the BAR registers of the EP. This is an extension of the test scenario Pcie_rc_read_ep_mem to write and read multiple numbers of bytes.

Note: In order to ensure a EP memory read & write we have restricted the enumeration to one device.

MAX_DEV_CNT - Macro to change the device count .

Flow diagram of tasks:


pcie_scenario_start
pcie_get_memory (addr buff)
pcie_get_memory (size buff)
pcie_get_memory (rd_data_buff)
pcie_get_memory (wr_data_buff)
pcie_set_random_data(wr_data_buff)
pcie_rc_init_setup
pcie_config_rc_atu_ob_tlp_info(cfg_region)
pcie_config_rc_atu_ob_tlp_info(mem_region)
rc_enum
pcie_rc_read_ep_config
pcie_compare_read_value
pcie_rc_get_ep_mem_details
pcie_rc_write_ep_mem
pcie_rc_read_ep_mem
pcie_compare_read_value
pcie_rc_deinit
pcie_release_memory (addr buff)
pcie_release_memory (size buff)
pcie_release_memory(rd_data_buff)
pcie_release_memory (wr_data buff)
pcie_scenario_success