Application Note

AM64x and AM243x: MCU+ SDK-Based PCIe End Point

Ashwani Goel *Sitara MPU*

ABSTRACT

This document serves as application report for the PCIe EP driver extension of the MCU+SDK 9.2.1 or newer for the Texas Instruments ARM-based SOCs AM64x and AM243x.

Table of Contents

Trademarks

1 Abbreviations

PCIe	Peripheral Component Interconnect Express
PCI-SIG	PCI Special Interest Group
EP	End Point
RC	Root Complex
SSC	Spread Spectrum
BIOS	Basic Input Output Software
CCS	Code Composer Studio
TI	Texas Instruments
BAR	Base Address Register
MSI	Message Signal Interrupt
MSI-X	Message Signal Interrupt X
SBL	Secondary Bootloader
VFIO	Virtual Function I/O
IOMMU	I/O Memory Management Unit
ATU	Address Translation Unit
FLR	Function Level Reset

2 Introduction

2.1 Peripheral Component Interconnect Express

Peripheral Component Interconnect Express (PCIe) is a motherboard expansion bus standard introduced in 2003 to enable high-speed serial communication between the Central Processing Unit (CPU) and the peripheral components. Today, the PCIe is the primary motherboard expansion bus standard and a popular communication method for many other onboard applications. PCIe is often used for Graphics Processing Unit (GPU) and solid-state drives (SSD) to send and receive data with the CPU.

- Texas Instruments TIPL video: What is PCIe?

2.1.1 Components of PCIe Communication

PCIe communication consists of three main components: root complex, repeaters, and PCIe endpoints. PCIe communication is hierarchical so there is a single source, which is the root complex, through which all the data passes. The data goes to the root complex from multiple PCIe endpoints and vice versa.

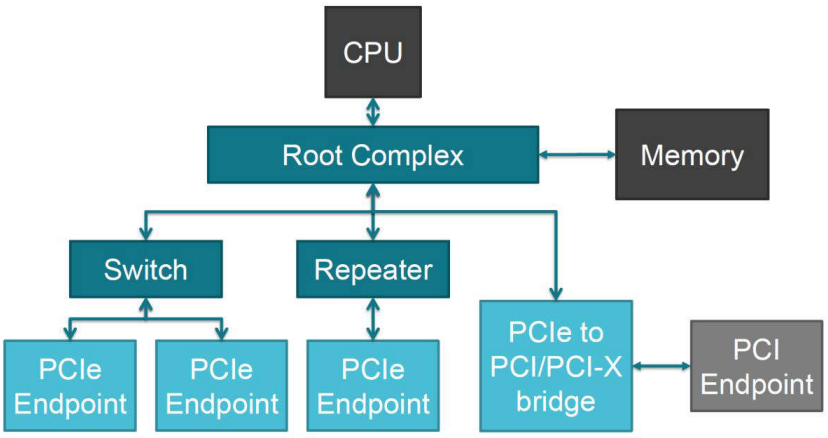


Figure 2-1. PCIe Topology

2.1.1.1 Root Complex

A root complex is the interface between the system CPU, memory, and the remainder of the PCIe interface. The root complex is either integrated into the CPU directly, or is external to the CPU as a discrete component. This interface also acts as a single source where all the data from various PCIe endpoints pass through. Figure 2-1 shows the root complex as the dark blue box that connects CPU, memory, and PCIe components and is referred as *Root Complex*.

For more details AM64x as RC

2.1.1.2 Repeater

A repeater is a signal conditioning device that makes sure a good signal to reach to and from the root complex and PCIe endpoints. Repeaters can fall into two categories: retimers and redrivers. Both are common PCIe components used to maintain signal quality of high speed links and compensate for the loss of signal quality over the traces. Figure 2-1 shows the repeater as the dark blue box that connects the root complex and PCIe endpoint and is referred as *Repeater*.

2.1.1.3 Endpoints

An endpoint is a general term for a PCIe end component. This can represent many different types of PCIe devices such as M.2 solid state drive (SSD) or graphics processing unit (GPU). The endpoint can be either a PCIe component or a PCI component with PCIe to PCI/PCI-X bridge. Figure 2-1 shows PCI endpoint as light blue box and gray box that is connected to either bridge, switch, or repeater and is referred as either *PCIe Endpoint* or *PCI Endpoint*.

For more details AM64x as EP

2.1.2 Signaling

Each component of PCIe communication (except for redrivers) have the following control signals: PERST, WAKE, CLKREQ, and REFCLK. These signals work to generate high-speed signals and communicate with other PCIe devices. Figure 2-2 shows the diagram of PCIe devices with the control signals. This diagram shows that all of the control signals except REFCLK are active low signals.

In Figure 2-2, the repeater is referred as retimer

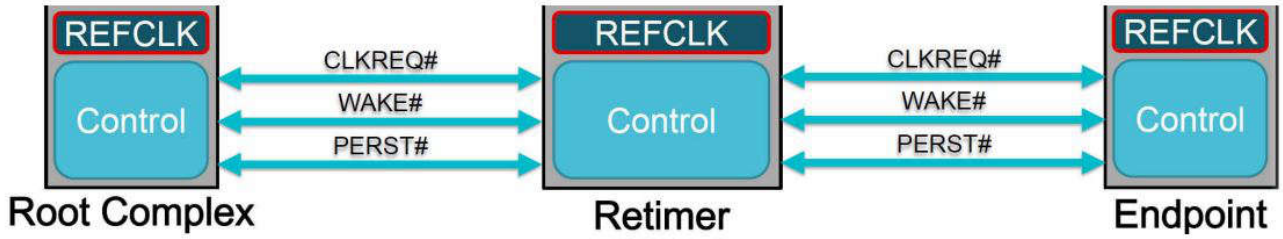


Figure 2-2. PCIe Signaling

2.1.2.1 PERST

PERST is referred to as a fundamental reset. PERST can be held low until all the power rails in the system and the reference clock are stable. A transition from low to high in this signal usually indicates the beginning of link initialization. In Figure 2-2, PERST is referred as *PERST#*.

2.1.2.2 WAKE and CLKREQ

WAKE and CLKREQ signals are both used for transitioning to and from low power states. WAKE signal is an active-low signal that is used to return the PCIe interface to an active state when in a low-power state. CLKREQ signal is also an active-low signal and is used to request the reference clock. In Figure 2-2, these are referred as *WAKE#* and *CLKREQ#*, respectively.

2.1.2.3 REFCLK

A REFCLK, or reference clock signal, is a prerequisite for a PCIe device to begin data transmission. This 100MHz reference clock signal is used by the PCIe device to generate the high-speed PCIe data within the link and is shared by the PCIe devices within the link. In Figure 2-2, REFCLK is referred as *REFCLK*.

2.1.3 PCIe Common Usage

- **Computer Hardware**
  - **Graphics Cards**
    PCIe is the common motherboard interface for graphics cards. PCIe allows high-speed communication between the GPU and the remainder of the system.
  - **Sound Cards**
    Sound cards use PCIe slots for audio processing and output.
  - **Storage Devices**
    SSDs (Solid State Drives) connect through PCIe for fast data transfer.
  - **Network Interface Cards (NICs)**
    PCIe enables high-speed networking connections.
- **Industrial Systems**
  - In industrial automation and control systems, PCIe is used for high-speed communication between sensors, actuators, and controllers.
  - Industrial PCs often rely on PCIe for expansion cards and peripherals.
- **Data Centers**
  - Servers and storage systems in data centers use PCIe for connecting storage devices, network adapters, and accelerators (such as GPUs or FPGAs).
  - PCIe provides low-latency communication, essential for data center workloads.
- **Automotive Technology**
  - **Infotainment Systems**
    PCIe interfaces are used for connecting multimedia components, such as displays, audio systems, and navigation units.
  - **Advanced Driver Assistance Systems (ADAS)**
    PCIe connects sensors, cameras, and processing units for real-time data processing.
- **Laptop and Mini-PCs**
  - PCIe is used to connect built-in peripherals and add-in cards.
  - **Mini PCIe** uses the same topology and specifications as regular PCIe and is electrically compatible.
  - The now-common M.2 SSD interface also uses PCIe topology.

2.1.4 PCIe Aggregate Throughput

A PCI Express interconnect is referred to as a Link, and connects two devices. A link consists of either 1, 2, 4, 8, 12, 16 or 32 signals in each direction (note, because the system uses full-differential signaling, each signal actually needs two wires). These signals are referred to as Lanes. A designer determines how many lanes to implement based on the targeted performance benchmark required on a given link. In the nomenclature, the width of a link is shown with an ***x*** in front of a number, where the ***x*** is pronounced as *by*, so that a link with 4 signals in each direction, for example, is referred to as *by four* link.

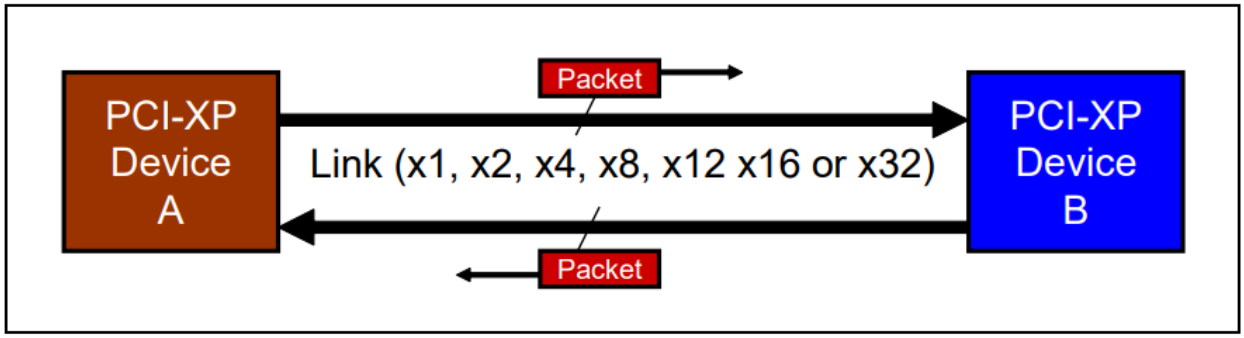


Figure 2-3. PCIE Express Link

Table 2-1 shows the aggregate bandwidth numbers for various Link width implementations. As is apparent from this table, the peak bandwidth achievable with PCI Express is significantly higher than most existing buses today.

Consider how these bandwidth numbers are calculated. The transmission/reception rate is currently 2.5 Gbits/sec per Lane per direction. To support a greater degree of robustness during data transmission and reception, each byte of data to be transmitted is converted into a 10-bit code (via an 8b/10b encoder in the transmitter device). In other words, for every Byte of data to be sent, 10-bits of encoded data are actually transmitted. The result is a 25% overhead to transmit a byte of data. PCI Express implements a dual-simplex Link which implies that data is both transmitted and received simultaneously.

The aggregate bandwidth assumes simultaneous traffic in both directions. To obtain the aggregate bandwidth numbers in Table 2-1, multiply 2.5 Gbits per second by 2 (to account for both directions), then multiply by the number of Lanes, and finally divide by 10-bits per Byte (to account for the 8-to-10 bit encoding) to arrive at a bytes per second result.

Table 2-1. PCIe Link Speed

PCIExpress Link Width	x1	x2	x4	x8	x12	x16	x32
AggregateBand- width (GBytes/sec)	0.5	1	2	4	6	8	16

2.2 PCIe Features on AM64x and AM243x

There is one instance of PCIe subsystem. Following are some of the main features:

- EP and RC operation
- Gen 1 and 2 operation speed
- x1 lane support
- Legacy interrupts
- MSI (Message Signaled Interrupt)

EVM

There is one instance of the PCIe subsystem on the EVM. Following are some of the details for that instance:

Table 2-2. PCIe on AM64x and AM243x EVM

Instance	Supported lanes	Supported Connector
PCIE0	1 lane	Standard female connector

For more info : PCIe

3 X86 as RC and AM64x as EP

Here, we are describing the steps to make PCIe connection between generic x86 PC as RC and AM64x as EP.

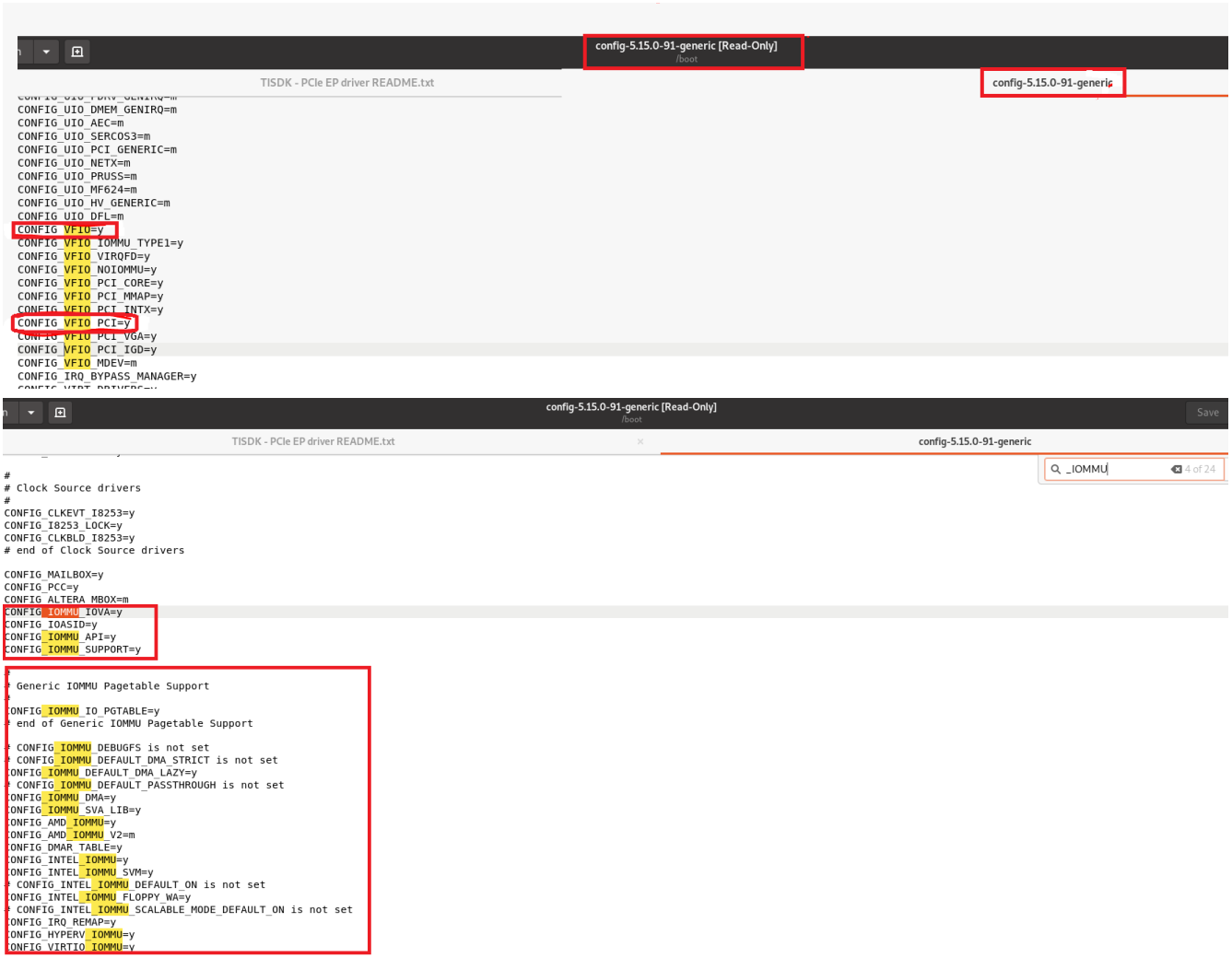
3.1 Hardware Environment

The following hardware is used to perform the specified function tests:

- x86 system with an available PCIe slot
  - The PCIe slot does NOT need to be connected to a PCIe switch but needs to rather be connected directly to the x86 CPU or PCH.
  - For development and testing the following systems have been used:

```
root@sitarampuapps-ThinkStation-P620:/home/sitara_mpu_apps# uname -a
Linux sitarampuapps-ThinkStation-P620 5.15.0-91-generic #101~20.04.1-Ubuntu SMP Thu Nov 16 14:22:28 UTC 2023 x86_64 x86_64 x86_64 GNU/Linux
```

- The x86 needs to be installed with a recent Linux version that supports VFIO, VFIO-PCI and IOMMU.



- TMDS243EVM or TMDS64EVM: is termed as a single EVM going forward
- PCIE_FLEX_NOCLK: Adex Electronics PCIe flexible extender cable PE-FLEX1-G2.MMCX-12-TI1
- PCIE_FLEX_CLK: Modified Adex Electronics PCIe flexible extender cable PE-FLEX1-G2.MMCX-12-TI1 with connected reference clock REFCLK+/-
- SPEC_ANA: Spectrum Analyzer
- The AM24x EVM needs to be modified to support a common reference clock for PCIe: - remove Resistors R661, R662, R667 & R668 - populate Resistors R665, R666, R679 & R680 (all 0 ohm)

3.2 Software Environment

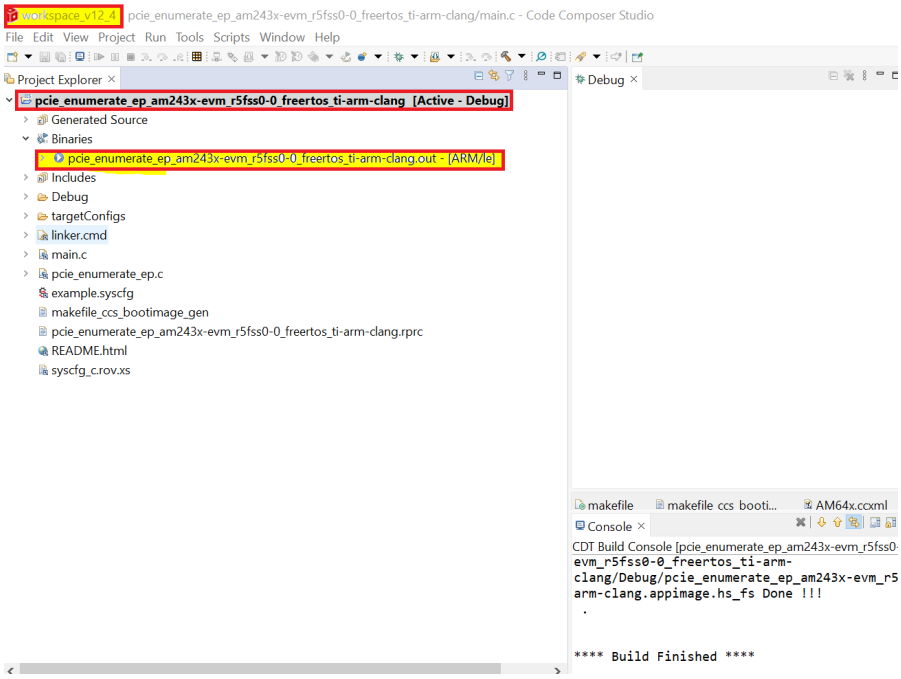
The following software is used to perform the specified function tests:

1. CCS12: TI Code Composer Studio, Version: 12.4.0.00007 2.
2. SysConfig: Version: 1.17.0 3.
3. SER_TER: Serial Terminal Emulator Program for example, Tera Term or Putty
4. LIN: Lenovo ThinkStation-P620 having Ubuntu 20.04
5. WIN: Windows 10 22H2
6. MCU+ SDK
7. EP sample application
   a. The PCIe Enumeration (EP) example demonstrates an EP that supports enumeration through an RC that is running Windows or Linux.

3.2.1 Building Application

Location: examples/drivers/pcie/pcie_enumerate_ep

The example can be imported into CCS 12.4 and built as a regular CCS project.



3.2.2 Usage

- You can refer MCUSDK documentation to flash generated binary into OSPI using section
- If the pcie_enumerate_ep example is started on an AM24x EVM that is NOT connected to an x86 RC, or if the x86 is not powered, the only output is going to be:
  – Power off AM64x and Linux-PC
  – Power on AM64x having *pcie_enumerate_ep* flashed

```
ÿ
DMSC Firmware Version 9.0.7--v09.00.07 (Kool Koala)
DMSC Firmware revision 0x9
DMSC ABI revision 3.1

[BOOTLOADER_PROFILE] Boot Media       : NOR SPI FLASH
[BOOTLOADER_PROFILE] Boot Media Clock : 166.667 MHz
[BOOTLOADER_PROFILE] Boot Image Size  : 0 KB
[BOOTLOADER_PROFILE] Cores present    :
r5f0-0
[BOOTLOADER_PROFILE] SYSFW init                       :      12192us
[BOOTLOADER_PROFILE] System_init                      :        548us
[BOOTLOADER_PROFILE] Drivers_open                     :        285us
[BOOTLOADER_PROFILE] Board_driversOpen                :      21966us
[BOOTLOADER_PROFILE] Sciclient Get Version            :       9840us
[BOOTLOADER_PROFILE] CPU Load                         :     186155us
[BOOTLOADER_PROFILE] SBL Total Time Taken             :     230991us

Image loading done, switching to application ...
PCIe: initialized and waiting for link
```

Afterward, the sample application is waiting for a PCIe link to be established, which requires the RC. Once the cable is connected and the RC is powered, the application outputs the state changes:

- Now power ON Linux PC

```
ÿ
DMSC Firmware Version 9.0.7--v09.00.07 (Kool Koala)
DMSC Firmware revision 0x9
DMSC ABI revision 3.1

[BOOTLOADER_PROFILE] Boot Media       : NOR SPI FLASH
[BOOTLOADER_PROFILE] Boot Media Clock : 166.667 MHz
[BOOTLOADER_PROFILE] Boot Image Size  : 0 KB
[BOOTLOADER_PROFILE] Cores present    :
r5f0-0
[BOOTLOADER_PROFILE] SYSFW init                       :      12192us
[BOOTLOADER_PROFILE] System_init                      :        548us
[BOOTLOADER_PROFILE] Drivers_open                     :        285us
[BOOTLOADER_PROFILE] Board_driversOpen                :      21966us
[BOOTLOADER_PROFILE] Sciclient Get Version            :       9840us
[BOOTLOADER_PROFILE] CPU Load                         :     186155us
[BOOTLOADER_PROFILE] SBL Total Time Taken             :     230991us

Image loading done, switching to application ...
PCIe: initialized and waiting for link
PCIe: link detected
PCIe Link Parameter: PCIe Gen2 with 5.0 GT/s speed, Number of Lanes: 1
EP is in D0 state
PCIe: signaling APPL ready
APPL: pcie ready
```

4 Test Setup

4.1 Common Setup for LINUX and WIN

To perform function tests with the TMDS243EVM/TMDS64EVM as a PCIe EP and either a Linux-based PCIe RC or a Windows-based RC, the following test setup needs to be performed:

1. Perform hardware modification on TMDS243EVM:
   a. Remove resistors R661, R662, R667 and R668
   b. Populate 0Ω resistors R665, R666, R679 and R680
2. Remove Jumper J34 on TMDS243EVM, as we want neither the AM24x to driver the **PERST** signal (we are an EP, this is an input) nor do we want the x86's PERST signal to reset our processor, **because we want to boot the AM24x BEFORE the x86** to make sure the startup and reset timing requirements are met.
3. Enable Intel Virtualization Technology in BIOS settings of X86 Linux-based RC hardware for IOMMU usage.
4. Connect TMDS243EVM with X86_10TH or X86_ADLN using the modified Cable PCIE_FLEX_CLK.
5. Note, the blue circled zero-ohm resistors in the following image:

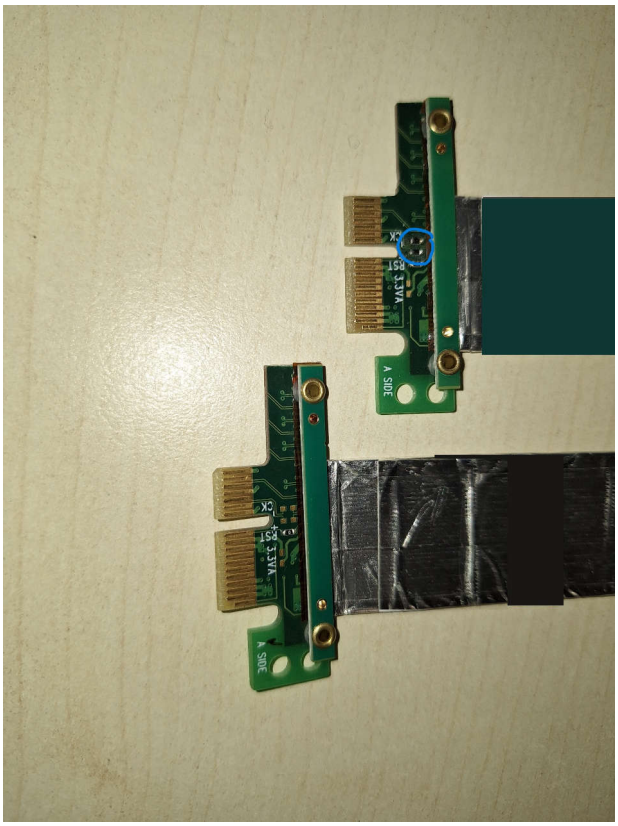


Figure 4-1. Modified PCIE_FLEX_CLK cable

6. Boot TMDS243EVM with *NULL* SBL from SD card.
7. Connect to TMDS243EVM through FTDI USB for UART port using serial terminal emulator program SER_TER.

Note

The serial connection is intended to provide various status messages of the PCIe EP, which can be used for test verification as described in Section 5 and Section 6

Each function test described in Section 5 requires modifications of the *pcie_enumerate_ep* example application. These modifications are either performed on the Sysconfig file or within the source code on CCS. On executing these modifications, continue with the following test setup:

1. On changing Sysconfig file or source code save files and build project.

2. Open designed for target configuration, connect with target. Perform system reset, load and run the *pcie_enumerate_ep* example application on TMDS243EVM.
3. Boot X86 based PCIe RC.

Note

The PCIe EP needs to always run before the PCIe RC hardware boots.

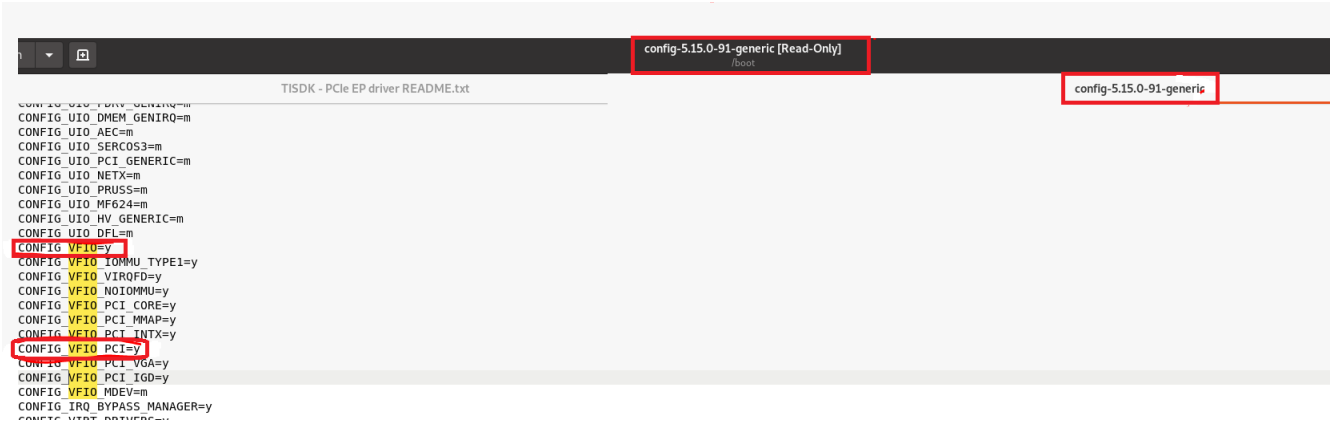
4. Perform test validation on successful boot up via terminal commands as descriptive in Section 5.

4.2 Linux Driver (VFIO)

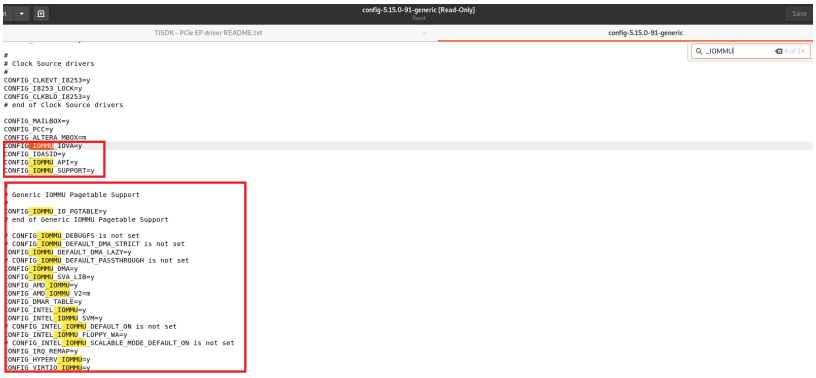
4.2.1 Prerequisites

To use the ti-sample-vfio example driver the target system's Linux needs to support the following features:

- VFIO, VFIO_PCI
  - CONFIG_VFIO, CONFIG_VFIO_PCI



- IOMMU
  - CONFIG_IOMMU, CONFIG_INTEL_IOMMU
  - enabled via kernel command line argument intel_iommu=on
  - Intel VT-d support needs to be enabled in the BIOS



4.2.2 Building

The ti-sample-vfio example driver consists of a single C file that can be compiled on-target using a simple GCC command:

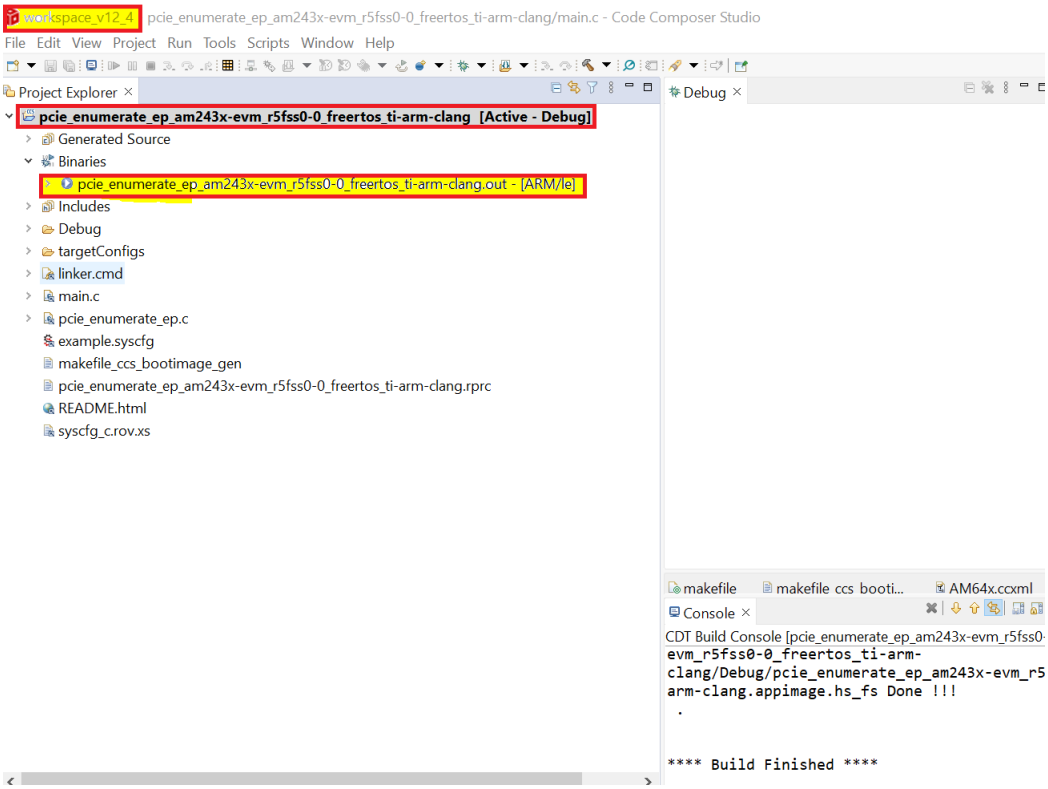
```
gcc ti-sample-vfio.c -o ti-sample-vfio -g -O2
```

4.2.3 Deploying

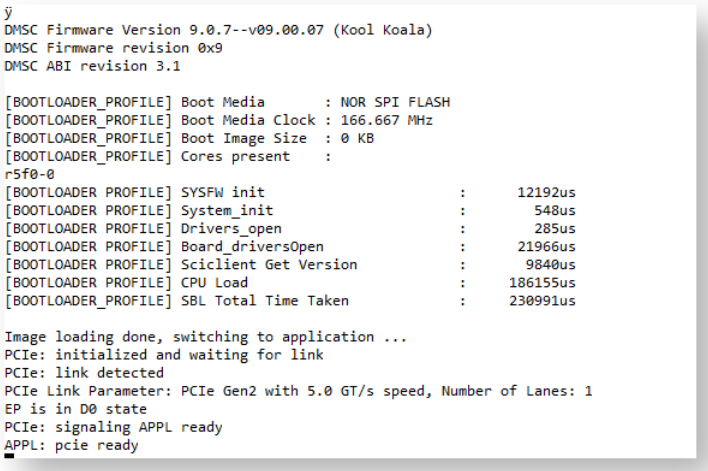
Since the driver can be compiled self-hosted on Linux there is no separate deployment step.

4.3 Test Application Usage

- To run the pcie_enumerate_ep example application the AM24x EVM needs to be booted with the NULL bootloader (SOC Initialization Binary).
- The pcie_enumerate_ep example can then be loaded via CCS 12.4 and the onboard XDS110.



- The pcie_enumerate_ep example prints the output on the EVM's debug UART.



- If the pcie_enumerate_ep example is started on an AM24x EVM that is NOT connected to an x86 RC, or if the x86 is not powered, the only output is going to be:

```
PCIe: initialized and waiting for link
```

- Afterward, the sample application is waiting for a PCIe link to be established, which requires the RC. Once the cable is connected and the RC is powered, the application outputs the state changes:

```
PCIe: link detected
PCIe Link Parameter: PCIe Gen1 with 2.5 GT/s speed, Number of Lanes: 1
EP is in D0 state
PCIe: signaling APPL ready
APPL: pcie ready
```

- At this point, the application is ready for configuration through the RC driver, either ti-sample-vfio or ti-sample-kmdf.

4.4 Setup Steps for LINUX PC

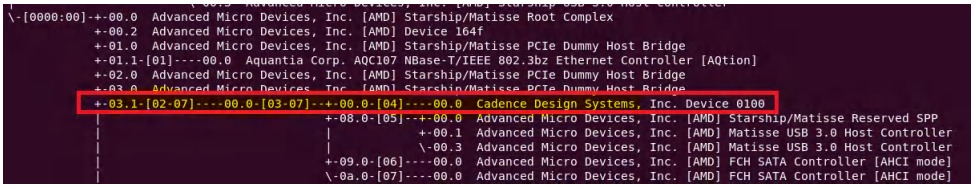
Since a RC sample application *ti-sample-vfio* based on Linux VFIO driver is implemented for testing and verification purposes, some tests descripted on Section 5 require the usage. To use *ti-sample-vfio*, the following setup needs to be implemented:

1. On successful PCIe boot up open Linux terminal and acquire root privileges:

```
sudo su
```

2. Determine bus-, device-, and function number of TMDS243EVM PCIe EP device using lspci command in Linux terminal. Use the vendor and device ID as set in Sysconfig. Search for it with the following command which shows related information of all PCIe devices numerically:

```
lspci -vtn
```



3. The output of the previous command is shown in the following figure. In that case, the PCIe EP is assigned the bus 4, device 00 and function 0.
4. Load the VFIO-PCI driver using modprobe:

```
modprobe vfio-pci
```

5. Assign TMDS243EVM PCIe EP vendor and device ID to the VFIO driver.

```
echo "17cd 0100" > /sys/bus/pci/drivers/vfio-pci/new_id
```

```
root@sitarampuapps-ThinkStation-P620:/home/sitara_mpu_apps# modprobe vfio-pci
root@sitarampuapps-ThinkStation-P620:/home/sitara_mpu_apps# echo "17cd 0100" > /sys/bus/pci
pci/         pci-epf/     pci_express/
root@sitarampuapps-ThinkStation-P620:/home/sitara_mpu_apps# echo "17cd 0100" > /sys/bus/pci/drivers/vfio-pci/new_id
```

6. Check which IOMMU group the PCIe EP is assigned to:

```
readlink /sys/bus/pci/devices/0000:04:00.0/iommu_group
```

   a. The output of the previous command can provide: PCIe EP assigned the IOMMU group.
7. Make sure the EP is the only device in this IOMMU group:

```
ls -l /sys/bus/pci/devices/0000:04:00.0/iommu_group/devices
```

```
root@sitarampuapps-ThinkStation-P620:/home/sitara_mpu_apps# ls -l /sys/bus/pci/devices/0000\:04\:00.0/iommu_group/devices/
total 0
lrwxrwxrwx 1 root root 0 Jan 15 14:39 0000:04:00.0 -> ../../../../devices/pci0000:00/0000:00:03.1/0000:02:00.0/0000:03:00.0/0000:04:00.0
```

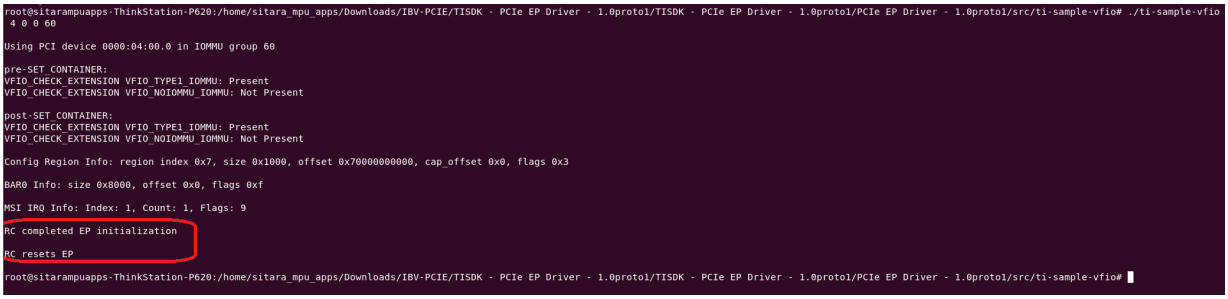
As can be seen the TMDS243EVM PCIe EP is the only device in IOMMU group 60. In case of additional PCIe devices within the same IOMMU group, these PCIe devices must be bind to VFIO driver as well.

8. Compile *ti-sample-vfio* (**root privileges are no longer needed for the next steps**):

```
gcc ti-sample-vfio.c -o ti-sample-vfio -g -O2
```

9. Execute *ti-sample-vfio* application with previously determined parameter:

```
sudo ./ti-sample-vfio 40 0 60 1 wait
```



Note

The *ti-sample-vfio* application requires the following parameter for execution:

1. [bus]: PCIe EP bus number
2. [device]: PCIe EP device number
3. [function]: PCIe EP function number
4. [IOMMU group]: PCIe EP IOMMU group
5. [test_mode] This parameter is only required for test case 4.3.2 which refers to an extended inbound ATU/BAR configuration (see the corresponding description). To start the test case, the parameter *testbars* must be passed. Otherwise, this parameter can be omitted, and the input is interpreted as the subsequent parameter [Number of MSI IRQs].
6. [Number of MSI IRQs]: Number of to be tested MSI IRQs as descripted on Test 4.5.2. This parameter needs to be set to 1 if test 4.5.2 is not performed.
7. [Number of loops]: Number of loops the test program can execute. This parameter is optional and can be left empty. The default value is 10.
8. ['wait'] This parameter instructs the test program to wait for user input throughout the execution of the test application.

4.4.1 UART Console Output

Running the sample application puts the device from D3hot into D0 state.

The application outputs further state changes while the sample executes until finally the EP is put back into D3hot state:

```
EP is in D0 state
PCIe: signaling APPL ready
APPL: pcie ready
PCIe: lost PCIe link
PCIe: hot reset detected
PCIe: signaling APPL halt
APPL: pcie not ready
PCIe: link detected
PCIe Link Parameter: PCIe Gen2 with 5.0 GT/s speed, Number of Lanes: 1
PCIe: signaling APPL ready
APPL: pcie ready
PCIe: MSI enabled with 1 vector(s) using address fee00538 and data 0
APPL: EP configured
APPL: EP unconfigured
PCIe: lost PCIe link
PCIe: hot reset detected
PCIe: signaling APPL halt
APPL: pcie not ready
PCIe: link detected
PCIe Link Parameter: PCIe Gen2 with 5.0 GT/s speed, Number of Lanes: 1
PCIe: signaling APPL ready
```

```
APPL: pcie ready
PCIe: power state entry
EP is in D3hot state
PCIe: signaling APPL halt
APPL: pcie not ready
```

```
r5t0-0
[BOOTLOADER PROFILE] SYSFW init                        :      12192us
[BOOTLOADER PROFILE] System_init                       :        547us
[BOOTLOADER PROFILE] Drivers_open                      :        285us
[BOOTLOADER PROFILE] Board_driversOpen                 :      22008us
[BOOTLOADER PROFILE] Sciclient Get Version             :       9845us
[BOOTLOADER PROFILE] CPU Load                          :     184217us
[BOOTLOADER_PROFILE] SBL Total Time Taken              :     229098us

Image loading done, switching to application ...
PCIe: initialized and waiting for link
PCIe: link detected
PCIe Link Parameter: PCIe Gen2 with 5.0 GT/s speed, Number of Lanes: 1
EP is in D0 state
PCIe: signaling APPL ready
APPL: pcie ready
PCIe: power state entry
EP is in D3hot state
PCIe: signaling APPL halt
APPL: pcie not ready
EP is in D0 state
PCIe: signaling APPL ready
APPL: pcie ready
PCIe: lost PCIe link
PCIe: hot reset detected
PCIe: signaling APPL halt
APPL: pcie not ready
PCIe: link detected
PCIe Link Parameter: PCIe Gen2 with 5.0 GT/s speed, Number of Lanes: 1
PCIe: signaling APPL ready
APPL: pcie ready
PCIe: MSI enabled with 1 vector(s) using address fee00000 and data 0
APPL: EP configured
APPL: EP unconfigured
PCIe: lost PCIe link
PCIe: hot reset detected
PCIe: signaling APPL halt
APPL: pcie not ready
PCIe: link detected
PCIe Link Parameter: PCIe Gen2 with 5.0 GT/s speed, Number of Lanes: 1
PCIe: signaling APPL ready
APPL: pcie ready
PCIe: power state entry
EP is in D3hot state
PCIe: signaling APPL halt
APPL: pcie not ready
```

4.5 MSI Example

This test setup is intended for testing the PCIe MSI RC project *pcie_msi_irq_rc_am243x-evm_r5fss0-0_nortos_ti-arm-clang* in combination with the PCIe EP example project *pcie_msi_irq_ep_am243x-evm_r5fss0-0_nortos_ti-arm-clang*.

1. Connect two TMDS243EVM or two TMDS64EVM through unmodified cable PCIE_FLEX_NOCLK.

Figure 4-2. Two AM64x Connected Though Unmodified PCIe Cable

2. For first TMDS243EVM open CCS, import and build *pcie_msi_irq_rc_am243x-evm_r5fss0-0_nortos_ti-arm-clang*. Open designed for target configuration and connect with target. Perform system reset and load the example application.
3. Connect to first TMDS243EVM through FTDI USB for UART port using serial terminal emulator program SER_TER.
4. For second TMDS243EVM open second CCS application (this can require an additional workspace), import and build *pcie_msi_irq_ep_am243x-evm_r5fss0-0_nortos_ti-arm-clang*. Open designed for target configuration and connect with target. Perform system reset and load the example application.
5. Connect to second TMDS243EVM through FTDI USB for UART port using serial terminal emulator program SER_TER.

4.6 Setup Steps for WINDOWS PC

To make use of the driver the ti-sample-console application can be run. This application opens the driver, sends IOCTLs to the driver, and waits for the IOCTLs to return.

The KMDF driver uses the *pattern* sent with the IOCTL to fill the Bar0 data area of the EP and then triggers a *downstream* interrupt in the EP. The EP copies the data area from the Bar0 to the RC driver's DMA buffer and triggers an MSI in the RC. The RC handles this interrupt and replies to the IOCTL with the data sent back via DMA

This test setup is intended for:

- Windows-PC as PCIe Rc
- AM243x as PCIe EP

4.6.1 Prerequisites

To perform function tests with the TI TMDS243EVM as a PCIe EP and a Windows-based RC, the following test setup can be performed:

Building the Windows driver requires a Windows host with the following software packages:

- WDK for windows 10, version 2004
- Windows SDK 10.0.19041.685

Visual Studio 2019 (Professional or Community edition)

- spectre mitigation libraries need to be added as an individual component using Visual Studio's installer
- ti-sample-kmdf and ti-sample-console source code

The target machine needs to have the following software installed:

- Windows 10 22H2
- WDK for windows 10, version 2004
- Microsoft Visual C++ Redistributable

4.6.2 Building

The ti-sample-kmdf solution contains two projects, the kernel mode driver ti-sample-kmdf and a console application ti-sample-console.

Both projects can be built by opening the ti-sample-kmdf solution in Visual Studio 2019 and building the entire solution.

- The driver currently only supports 64-bit mode on x86 machines, so the platform *x64* needs to be selected, along with the configuration *Release*.
- In the solution explorer, select the solution *ti-sample-kmdf* and build the solution.

The build output is located in a new folder *x64\Release* below the solution directory (ti-sample-kmdf\x64\Release).

4.6.3 Deploying

The following files from the solution's build output are required on the target machine:

- ti-sample-kmdf.inf
- ti-sample-kmdf.sys
- ti-sample-kmdf.cat
- ti-sample-console.exe

Windows by default only accepts signed drivers. An installation can be modified to accept so-called test signed drivers. The Windows KMDF sample driver ti-sample-kmdf uses this approach and is built as a test signed driver.

To allow Windows to use test signed drivers, open an administrator prompt (cmd, Run as administrator) and enter the following command:

```
Bcdedit.exe -set TESTSIGNING ON
```

Enabling test signing the system needs to be rebooted. At this point the AM24x EVM with the pcie_enumerate_ep application needs to be started as described above.

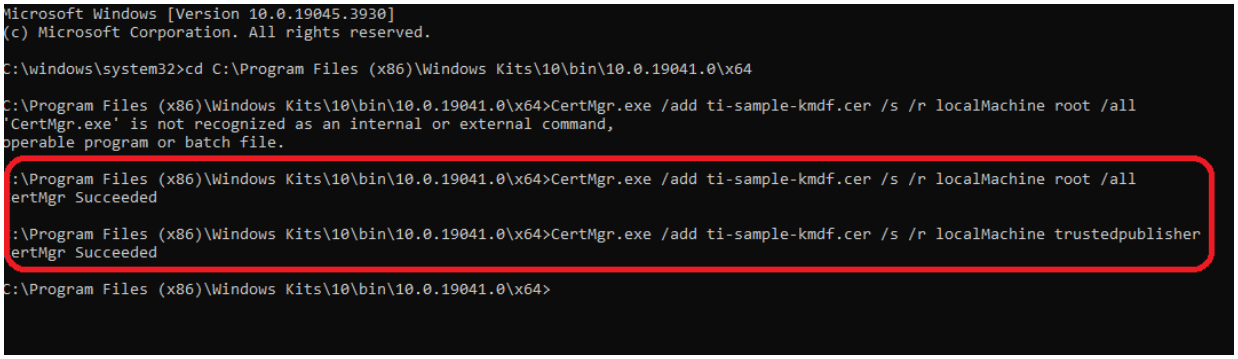
You then need to install the certificate used to test sign the driver on the target computer. This certificate is placed in the solution output folder along with the driver and is named ti-sample-kmdf.cer.

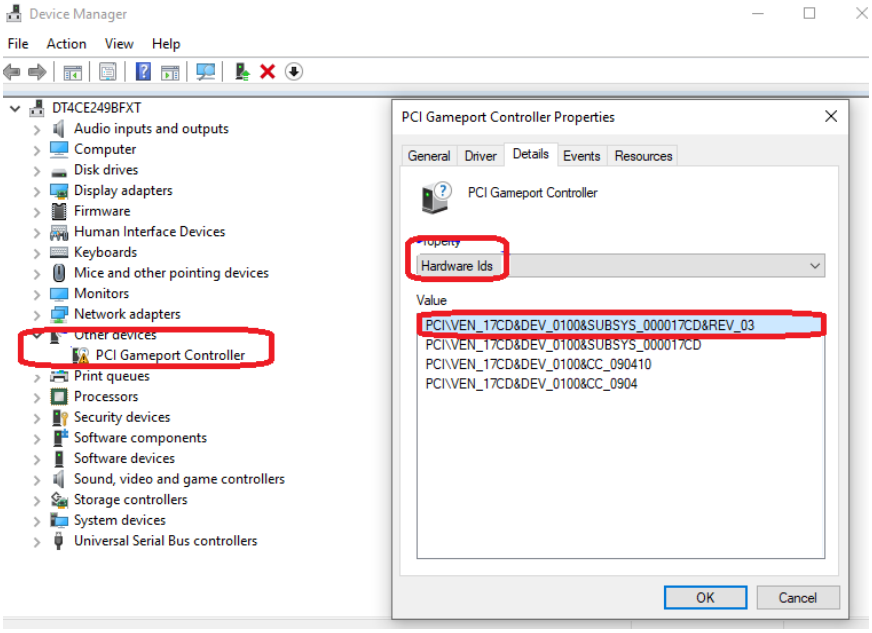
It can be installed using the CertMgr.exe tool that comes with the WDK from an administrator prompt:

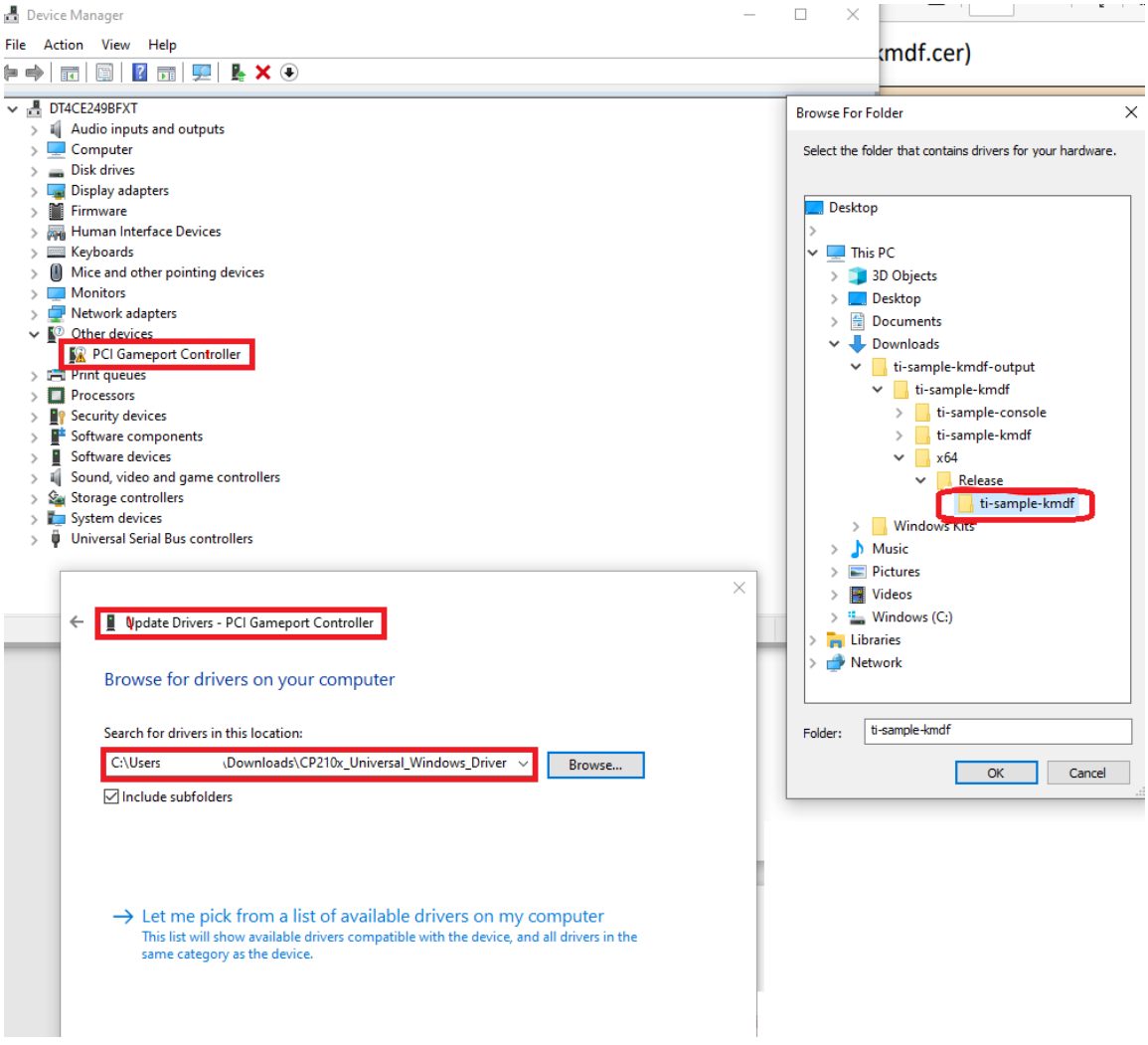
- cd C:\Program Files (x86)\Windows Kits\10\bin\10.0.19041.0\x64\
- CertMgr.exe /add ti-sample-kmdf.cer /s /r localMachine root /all
- CertMgr.exe /add ti-sample-kmdf.cer /s /r localMachine trustedpublisher

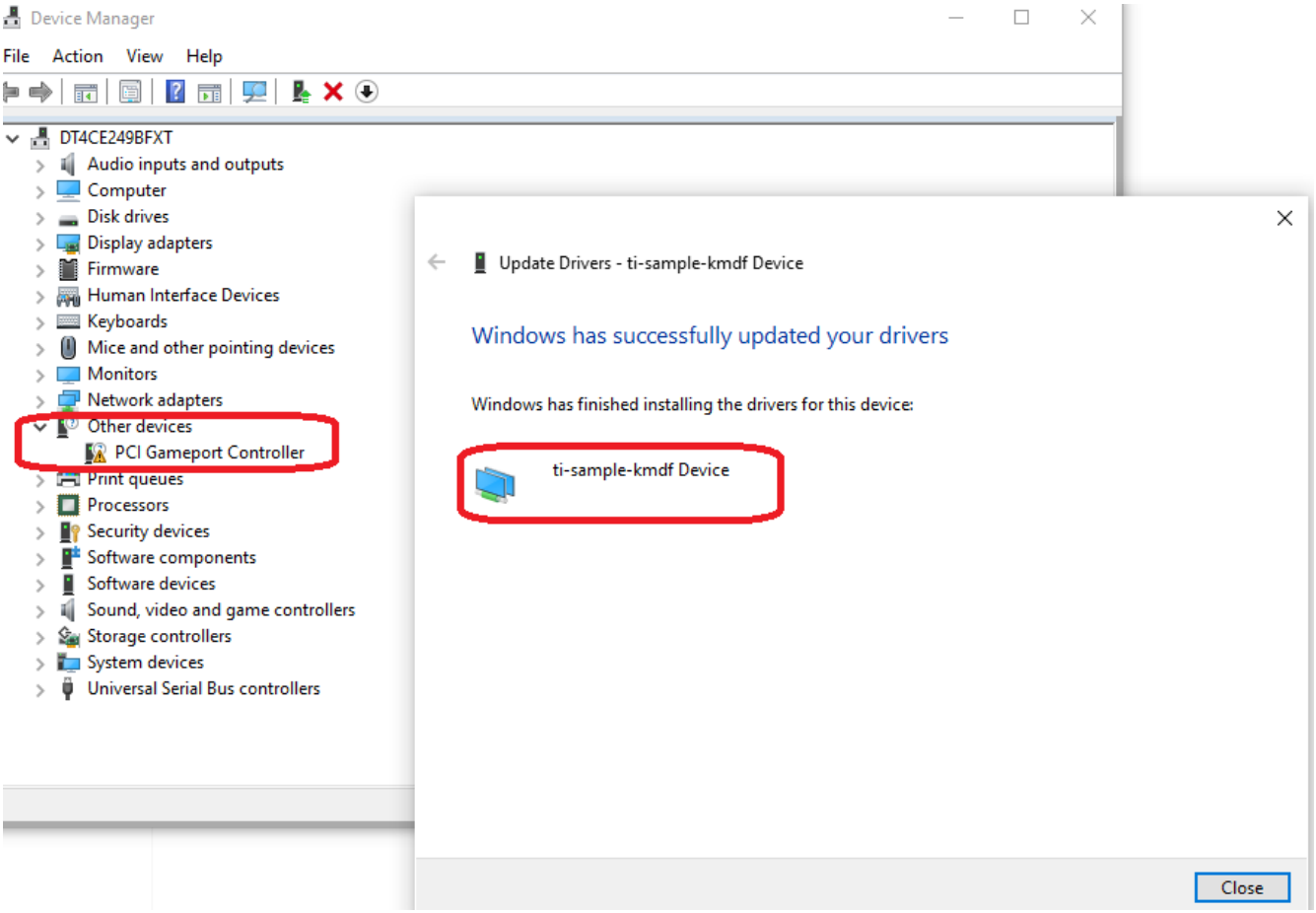
Note

- Use full path to ti-sample-kmdf.cer
- PCI device and verify the hardware ID is PCI\VEN_17cd&DEV_0100. Right-click the device









Windows installs the driver and then notifies you when finished installing the driver for the ti-sample-kmdf device.

5 PCIe Test Specification

This chapter defines and specifies various PCIe function tests. Based on a test description, detailed instructions are given on how to carry out a test as well as an explanation of the desired results. The following test specification shown in Section 4.4 assumes environment.

5.1 Identification and Configuration Functionalities

Table 5-1. Identification and Configuration Functionalities

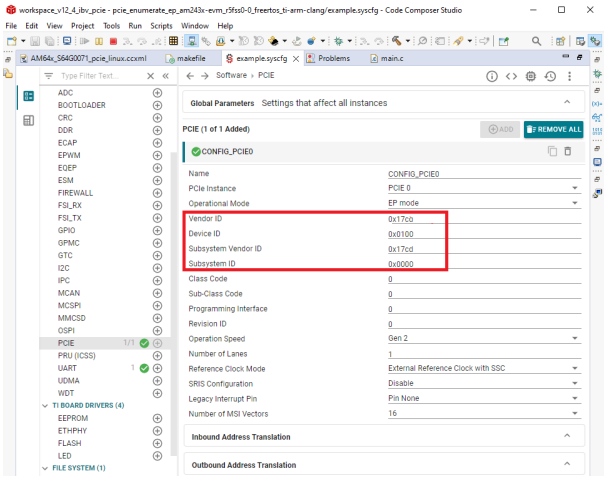
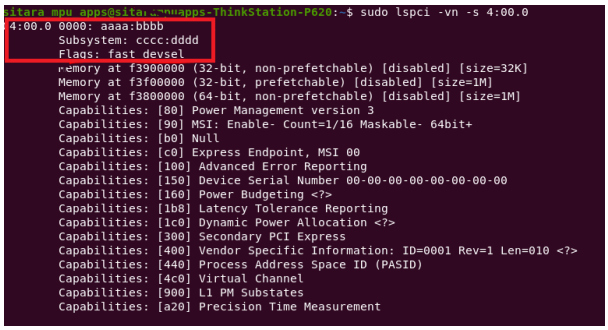
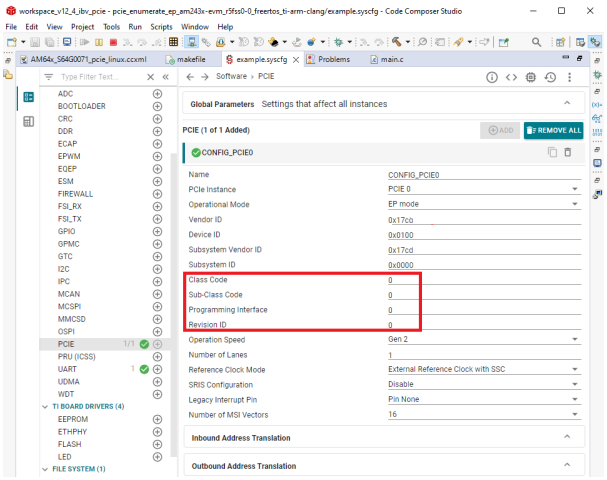
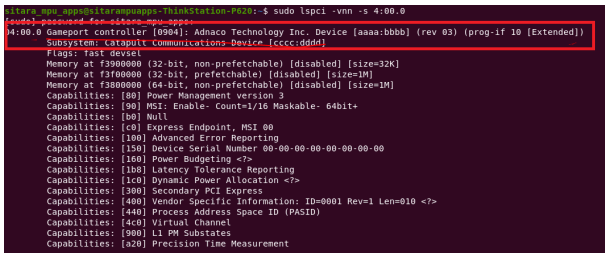
S.No.	Test Specification
1	**Description**: Test to verify if desired PCIe vendor ID, device ID, subsystem ID & subsystem vendor ID can be set and configured correctly in the TMDS243EVM PCIe EP. **Execution**: 1. Configure desired IDs through Sysconfig, and so on: a. Vendor ID:0xAAAA b. Device ID: 0xBBBB c. Subsystem Vendor ID: 0xCCCC d. Subsystem ID: 0xDDDD  2. Determine the PCIe EP bus-, device- and function number and verify the configured IDs in Linux terminal, and so on. a. sudo lspci -vn -s 4:00.0 b. The desired IDs are displayed as configured in Sysconfig. The following figure shows the expected result. 

Table 5-1. Identification and Configuration Functionalities (continued)

S.No.	Test Specification
2	**Description**: Test to verify if a desired PCIe class code, sub-class code, the programming interface & the revision-ID can be set and configured correctly in the PCIe EP. Just for testing purposes the base class code needs to be set as “Input device (09h)” with a sub-class code set as “Gameport controller (10h)” and a programming interface set to “10h”. The revision ID is set to “03h”. For additional information regarding the encoding of above parameter refer to “PCI Code and ID Assignment Specification” as published by PCI-SIG. **Execution**: 1. Configure desired settings through Sysconfig. a. Classe Code: 0x09 b. Sub-Class Code:0x04 c. Programming Interface: 0x10 d. Revision ID: 0x03  2. Determine the PCIe EP bus-, device- and function number and verify the configured settings in Linux terminal, and so on. a. sudo lspci -vnn -s 4:00.0 b. The desired parameter is displayed as configured in Sysconfig. The following figure shows the expected result. 

5.1.1 Test Case

Test

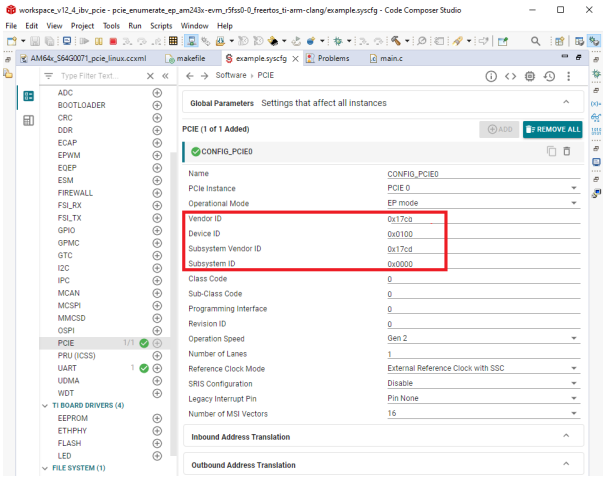
Description:

Test to verify if desired PCIe vendor ID, device ID, subsystem ID and subsystem vendor ID can be set and configured correctly in the TMDS243EVM PCIe EP.

Execution:

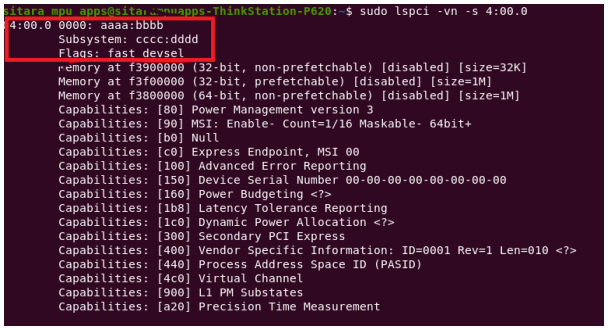
1. Configure desired IDs via Sysconfig, and so on.

a. Vendor ID:0xAAAA
b. Device ID: 0xBBBB
c. Subsystem Vendor ID: 0xCCCC
d. Subsystem ID: 0xDDDD



2. Determine the PCIe EP bus-, device- and function number and verify the configured IDs in Linux terminal, and so on.:

```
sudo lspci -vn -s 4:00.0
```



The desired IDs is displayed as configured in Sysconfig.

Test

Description:

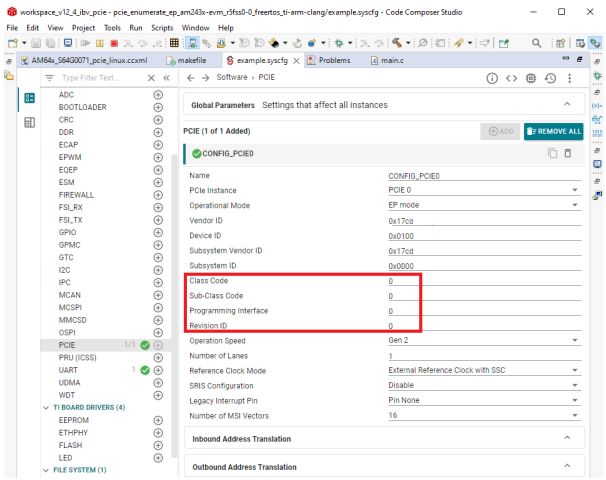
Test to verify if a desired PCIe class code, sub-class code, the programming interface and the revision-ID can be set and configured correctly in the PCIe EP.

Just for testing purposes. the base class code needs to be set as *Input device (09h)* with a sub-class code set as *Gameport controller (10h)* and a programming interface set to *10h*. The revision ID needs to be set to *03h*.

For additional information regarding the encoding of above parameter refer to *PCI Code and ID Assignment Specification* as published by PCI-SIG.

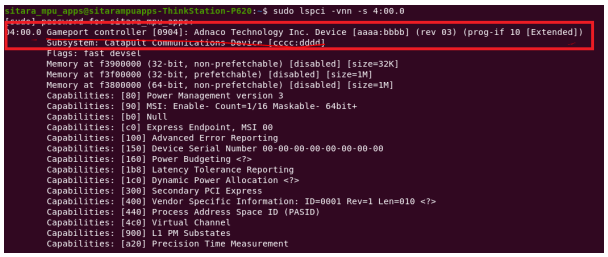
Execution:

1. Configure desired settings through Sysconfig.
   a. Class Code: 0x09
   b. Sub-Class Code:0x04
   c. Programming Interface: 0x10
   d. Revision ID: 0x03



2. Determine the PCIe EP bus-, device- and function number and verify the configured settings in Linux terminal, and so on:

```
sudo lspci -vnn -s 4:00.0
```



The desired parameter is displayed as configured in Sysconfig.

5.2 Reference Clock Functionalities

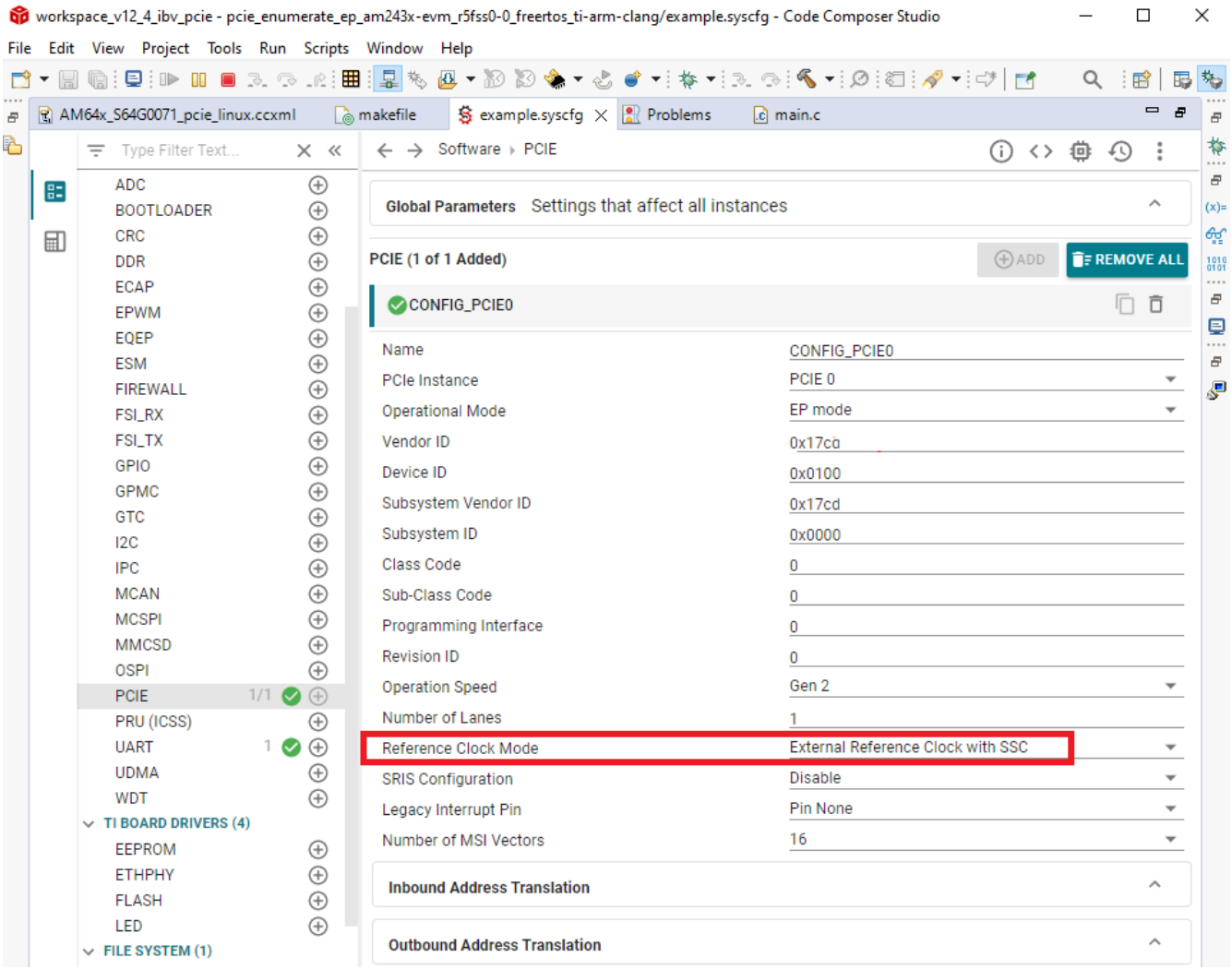
Test

Description:

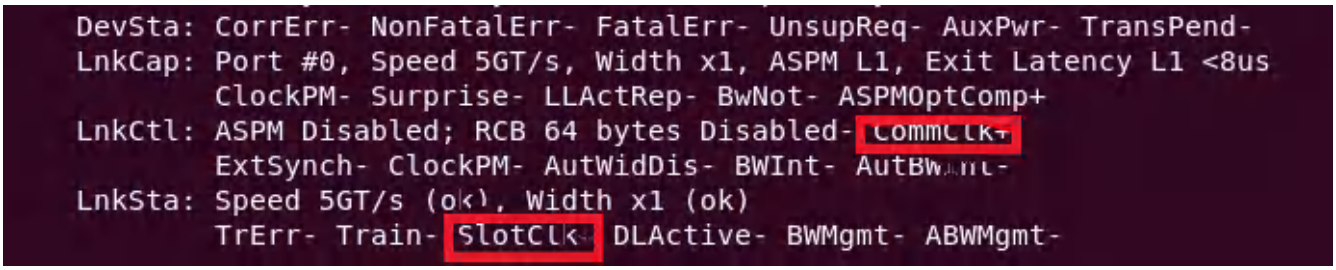
Test to verify if external reference clock can be configured on TMDS243EVM PCIe EP and if PCIe EP works correctly with external reference clock.

Execution:

1. Check if the following setting is configured on Sysconfig:
   a. Reference Clock Mode: External Reference Clock, no SSC



2. Verify if Common Clock and Slot Clock mechanisms are enabled as shown in the following figure.



Test

Description:

Test to verify if internal reference clock configuration of AM243X/AM64X can be used with enabled output and SSC configuration.

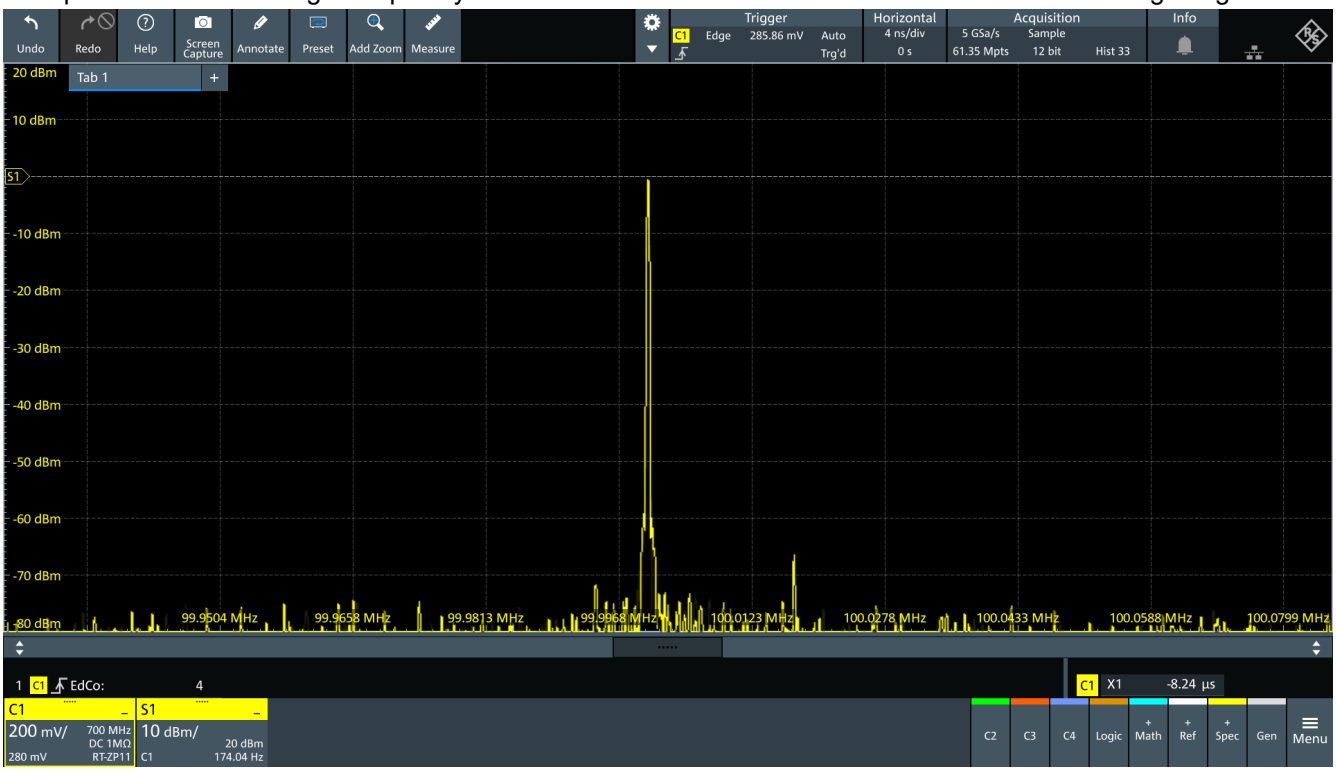
Execution:

1. Remove PCIe cable PCIE_FLEX_CLK from TMDS243EVM PCIe EP

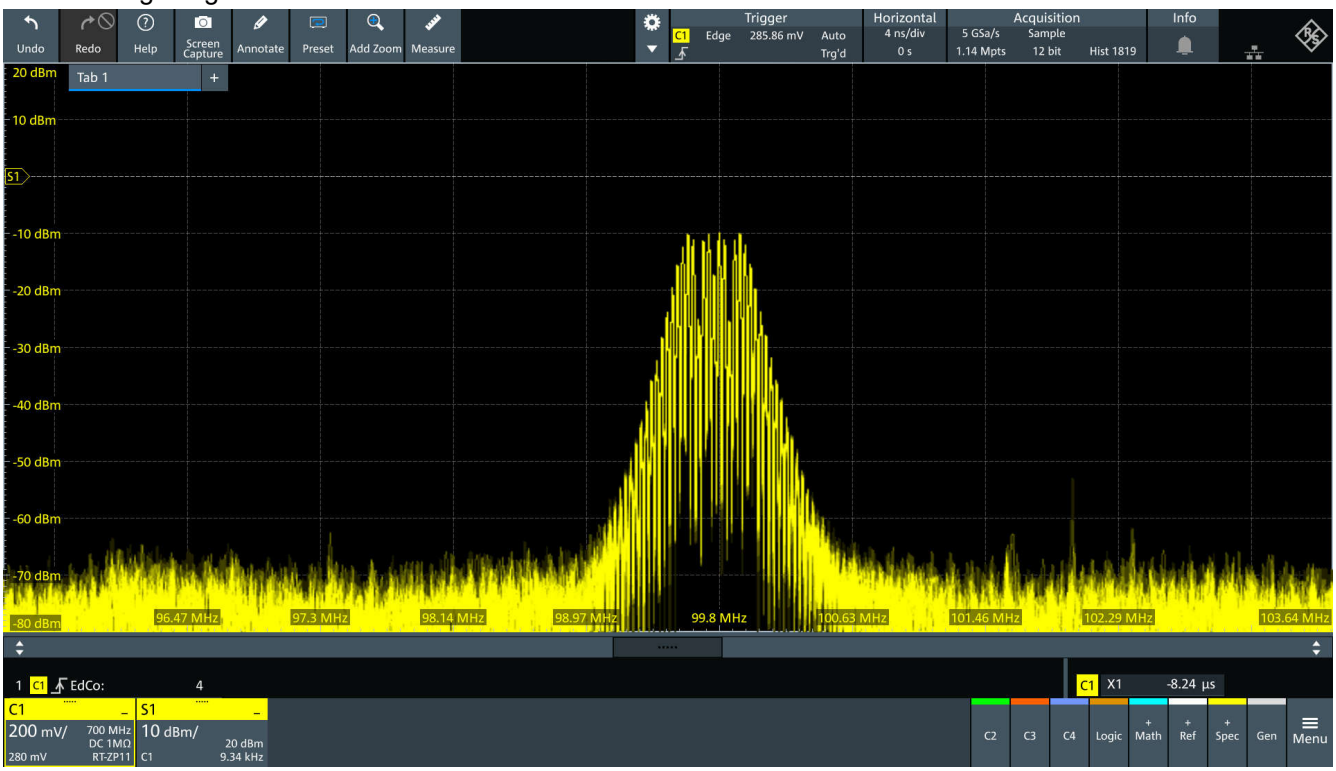
Note

This is important since having two driving PCIe reference clock sources from EP and RC can damage the hardware.

2. Configure the following setting via Sysconfig:
   a. Internal Reference Clock, no SSC, Output enabled.
3. Measure the PCIe reference clock spectrum on PCIe connector. Since SCC is disabled, the measured spectrum shows a single frequency above the noise level at 100MHz as shown in the following image.



4. Configure the following setting via Sysconfig:
   a. Internal Reference Clock, with SSC, Output enabled.
5. Since SCC is enabled, the measured spectrum can be spread one around 100MHz as shown in the following image.



We probe REFCLK+ and respective GROUND pins to get waveform on oscilloscope

https://en.wikipedia.org/wiki/PCI_Express

PCI Express connector pinout (x1, x4, x8 and x16 variants)

Pin	Side B	Side A	Description	Pin	Side B	Side A	Description
1	+12 V	PRSNT1#	Must connect to farthest PRSNT2# pin	50	HSOp(8)	Reserved	Lane 8 transmit data, + and −
2	+12 V	+12 V	Main power pins	51	HSOn(8)	Ground	
3	+12 V	+12 V		52	Ground	HSIp(8)	Lane 8 receive data, + and −
4	Ground	Ground		53	Ground	HSIn(8)	
5	SMCLK	TCK	SMBus and JTAG port pins	54	HSOp(9)	Ground	Lane 9 transmit data, + and −
6	SMDAT	TDI		55	HSOn(9)	Ground	
7	Ground	TDO		56	Ground	HSIp(9)	Lane 9 receive data, + and −
8	+3.3 V	TMS		57	Ground	HSIn(9)	
9	TRST#	+3.3 V		58	HSOp(10)	Ground	Lane 10 transmit data, + and −
10	+3.3 V aux	+3.3 V	Aux power & Standby power	59	HSOn(10)	Ground	
11	WAKE#	PERST#	Link reactivation; fundamental reset [23]	60	Ground	HSIp(10)	Lane 10 receive data, + and −
Key notch				61	Ground	HSIn(10)	
12	CLKREQ#[24]	Ground	Clock Request Signal	62	HSOp(11)	Ground	Lane 11 transmit data, + and −
13	Ground	REFCLK+	Reference clock differential pair	63	HSOn(11)	Ground	
14	HSOp(0)	REFCLK−	Lane 0 transmit data, + and −	64	Ground	HSIp(11)	Lane 11 receive data, + and −
15	HSOn(0)	Ground		65	Ground	HSIn(11)	
16	Ground	HSIp(0)	Lane 0 receive data, + and −	66	HSOp(12)	Ground	Lane 12 transmit data, + and −
17	PRSNT2#	HSIn(0)		67	HSOn(12)	Ground	
18	Ground	Ground		68	Ground	HSIp(12)	Lane 12 receive data, + and −
PCI Express x1 cards end at pin 18				69	Ground	HSIn(12)	

5.3 Inbound ATU and BAR Functionalities

Test

Description:

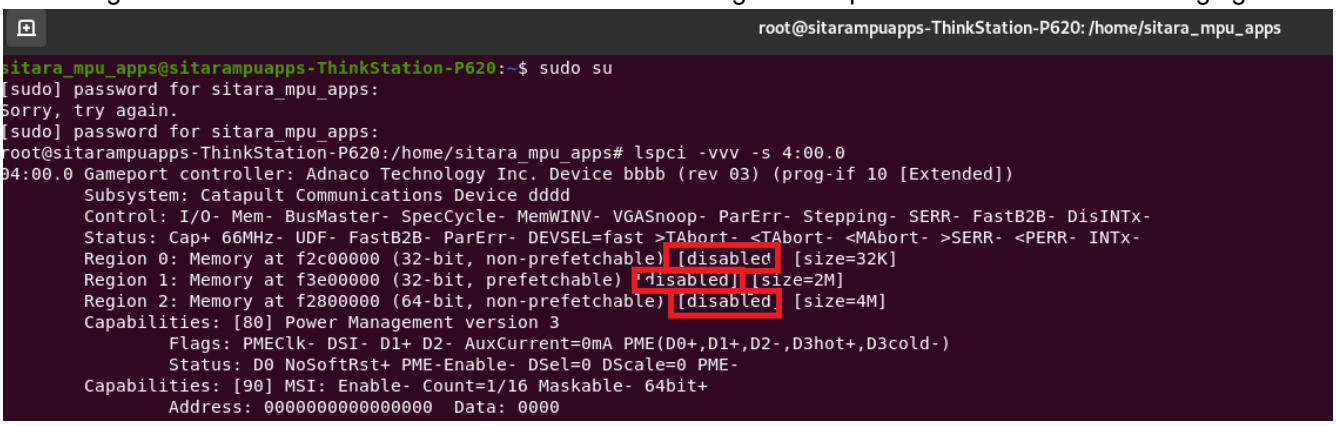
Test to verify if PCIe inbound ATU and BAR configurations work correctly for the TMDS243EVM/TMDS64EVM PCIe EP.

By default, the following BAR configurations are set in Sysconfig for the *pcie_enumerate_ep* example application:

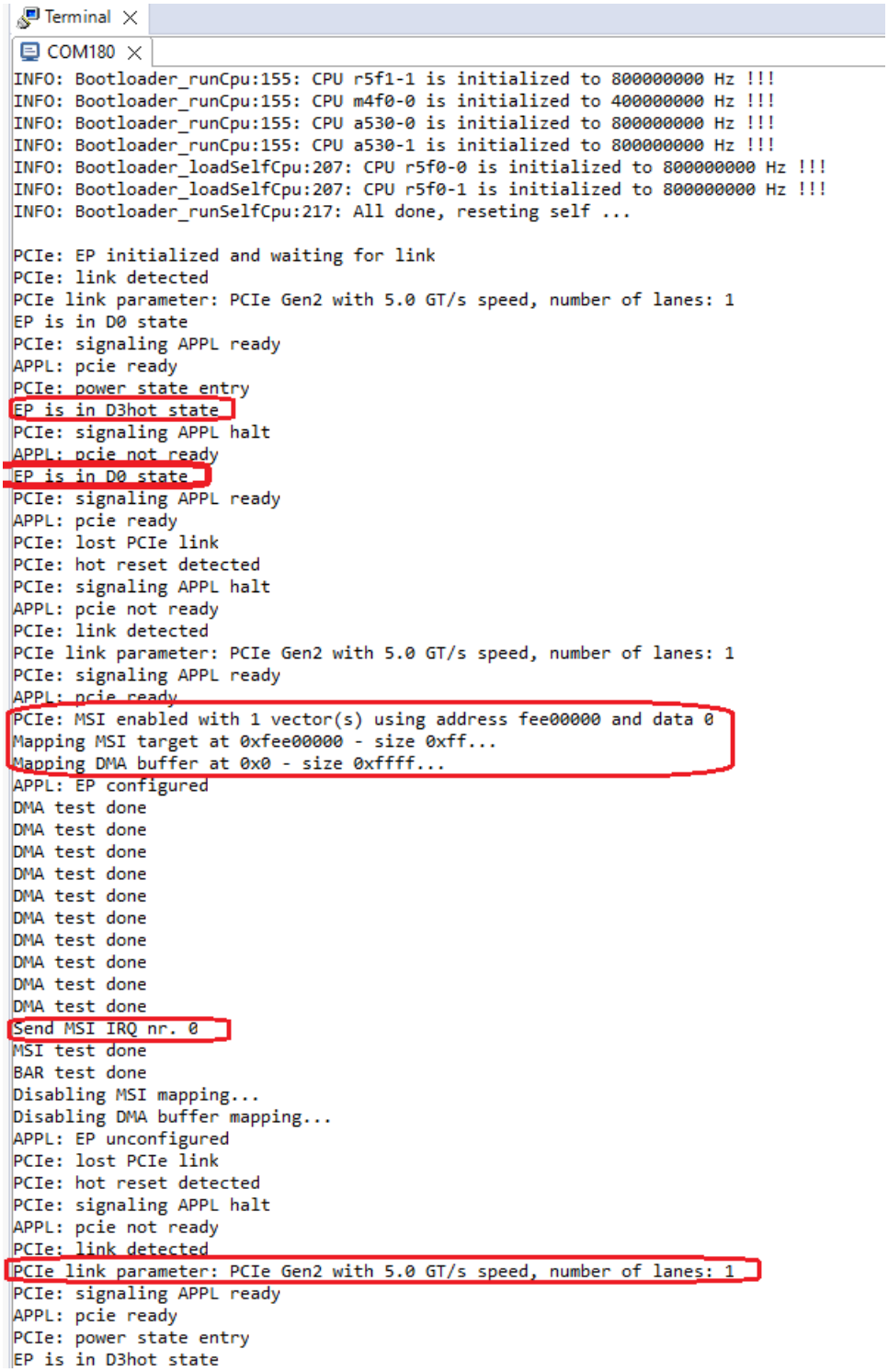
1. Inbound Address Translation 0:
   This ATU configuration uses region index 0 with a 32 Kbyte non-prefetchable 32bit memory BAR linked to an external struct bar0_mem. This inbound ATU configuration can not be modified for this test as it is required to ensure functionality with the RC VFIO based sample application *ti-sample-vfio*.
2. Inbound Address Translation 1:
   This ATU configuration uses region index 1 with a 64 Mbyte prefetchable 32bit memory BAR linked to an external data buffer bar1_data. This inbound ATU may be modified for this test as it is specifically implemented to test various BAR configurations.
3. Inbound Address Translation 2:
   This ATU configuration uses region index 2 with a 1 Gbyte non-prefetchable 64bit memory BAR linked to an external data buffer bar2_data. This inbound ATU may be modified for this test as it is specifically implemented to test various BAR configurations.

Execution:

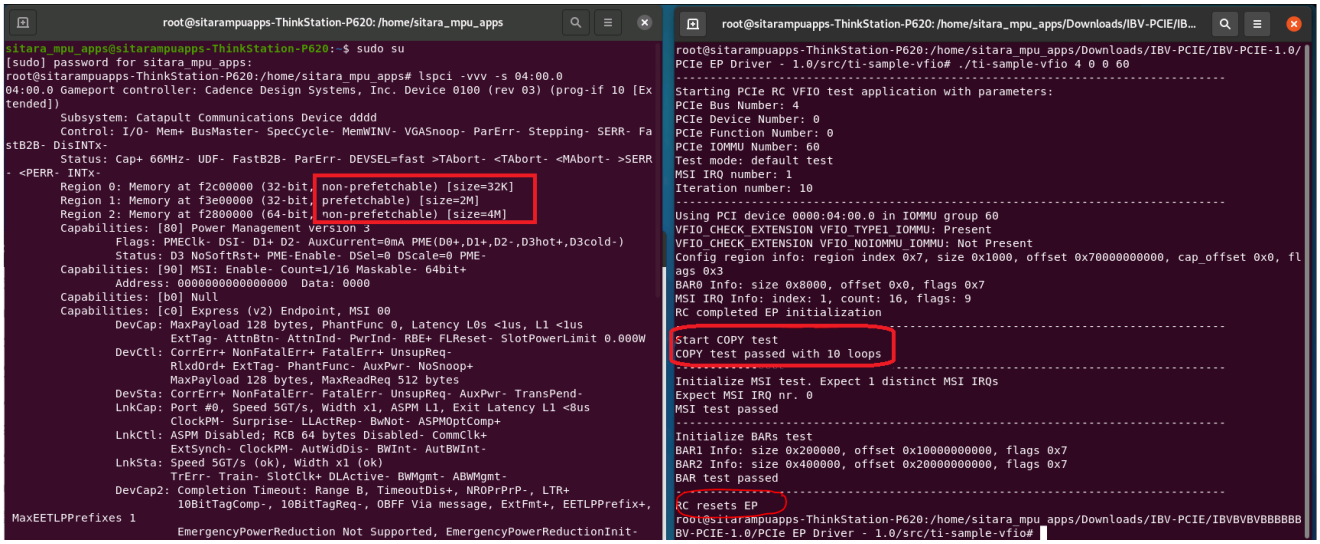
1. Set desired BAR configurations in Sysconfig file for Inbound Address Translation 1 and 2 for *pcie_enumerate_ep* application.
2. Check if desired PCIe EP BARs are configured correctly on Linux-based RC hardware. On boot up the configured BARs can be shown as disabled on PCIe configuration space as shown in the following figure:



3. Run RC sample application *ti-sample-vfio*. Open a second Linux terminal and check PCIe EP configuration space. As the program halts after EP initialization and BAR mapping, corresponding BARs can now be enabled (not shown as disabled).



4. Continue RC sample application *ti-sample-vfio*. The program can continue normally and end without any failure.



Test

Description

Test to verify if PCIe inbound ATU and extended BAR configurations work correctly for the TMDS243EVM/ TMDS64EVM PCIe EP.

For this purpose, up to 6 different BAR configurations are defined for the PCIe EP:

1. Inbound Address Translation 0:
   This ATU configuration uses region index 0 with a 32 Kbyte non-prefetchable 32bit memory BAR.
2. Inbound Address Translation 1:
   This ATU configuration uses region index 1 with a 32 Mbyte prefetchable 32bit memory BAR.
3. Inbound Address Translation 2:
   This ATU configuration uses region index 2 with a 512 Mbyte non-prefetchable 32bit memory BAR.
4. Inbound Address Translation 3:
   This ATU configuration uses region index 3 with a 128 byte 32bit I/O BAR.
5. Inbound Address Translation 4:
   This ATU configuration uses region index 4 with a 1 Kbyte 32bit I/O BAR.
6. Inbound Address Translation 5:
   This ATU configuration uses region index 5 with an 8 Kbyte 32bit. I/O BAR.

Execution:

1. Set described BAR configurations in Sysconfig file for *pcie_enumerate_ep* application.
2. Check if desired PCIe EP BARs are configured correctly on Linux-based RC hardware. On boot up the configured BARs can be shown as disabled on PCIe configuration space as shown in the following figure:

```
root@sitarampuapps-ThinkStation-P620:/home/sitara_mpu_apps# lspci -vvv -s 04:00.0
04:00.0 Gameport controller: Cadence Design Systems, Inc. Device 0100 (rev 03) (prog-if 10 [Extended])
        Subsystem: Cadence Design Systems, Inc. Device 0000
        Control: I/O- Mem- BusMaster- SpecCycle- MemWINV- VGASnoop- ParErr- Stepping- SERR- FastB2B- DisINTx-
        Status: Cap+ 66MHz- UDF- FastB2B- ParErr- DEVSEL=fast >TAbort- <TAbort- <MAbort- >SERR- <PERR- INTx-
        Region 0: Memory at f3800000 (32-bit, non-prefetchable) [disabled] [size=32K]
        Region 1: Memory at f3f00000 (32-bit, prefetchable) [disabled] [size=1M]
        Region 2: Memory at f3600000 (32-bit, non-prefetchable) [disabled] [size=2M]
        Region 3: I/O ports at 3400 [disabled] [size=128]
        Region 4: I/O ports at 3000 [disabled] [size=1K]
        Region 5: I/O ports at 2000 [disabled] [size=4K]
        Capabilities: [80] Power Management version 3
                Flags: PMEClk- DSI- D1+ D2- AuxCurrent=0mA PME(D0+,D1+,D2-,D3hot+,D3cold-)
                Status: D3 NoSoftRst+ PME-Enable- DSel=0 DScale=0 PME-
        Capabilities: [90] MSI: Enable- Count=1/16 Maskable- 64bit+
                Address: 0000000000000000  Data: 0000
        Capabilities: [b0] Null
        Capabilities: [c0] Express (v2) Endpoint, MSI 00
                DevCap: MaxPayload 128 bytes, PhantFunc 0, Latency L0s <1us, L1 <1us
                        ExtTag- AttnBtn- AttnInd- PwrInd- RBE+ FLReset- SlotPowerLimit 0.000W
                DevCtl: CorrErr+ NonFatalErr+ FatalErr+ UnsupReq-
                        RlxdOrd+ ExtTag- PhantFunc- AuxPwr- NoSnoop+
                        MaxPayload 128 bytes, MaxReadReq 512 bytes
                DevSta: CorrErr- NonFatalErr- FatalErr- UnsupReq- AuxPwr- TransPend-
                LnkCap: Port #0, Speed 5GT/s, Width x1, ASPM L1, Exit Latency L1 <8us
                        ClockPM- Surprise- LLActRep- BwNot- ASPMOptComp+
                LnkCtl: ASPM Disabled; RCB 64 bytes Disabled- CommClk+
                        ExtSynch- ClockPM- AutWidDis- BWInt- AutBWInt-
                LnkSta: Speed 5GT/s (ok), Width x1 (ok)
                        TrErr- Train- SlotClk+ DLActive- BWMgmt- ABWMgmt-
                DevCap2: Completion Timeout: Range B, TimeoutDis+, NROPrPrP-, LTR+
                         10BitTagComp-, 10BitTagReq-, OBFF Via message, ExtFmt+, EETLPPrefix+, MaxEETLPPrefixes 1
                         EmergencyPowerReduction Not Supported, EmergencyPowerReductionInit-
                         FRS-, TPHComp-, ExtTPHComp-
                         AtomicOpsCap: 32bit- 64bit- 128bitCAS-
                DevCtl2: Completion Timeout: 50us to 50ms, TimeoutDis-, LTR+, OBFF Disabled
                         AtomicOpsCtl: ReqEn-
                LnkCtl2: Target Link Speed: 5GT/s, EnterCompliance- SpeedDis-
                         Transmit Margin: Normal Operating Range, EnterModifiedCompliance- ComplianceSOS-
                         Compliance De-emphasis: -6dB
                LnkSta2: Current De-emphasis Level: -3.5dB, EqualizationComplete-, EqualizationPhase1-
                         EqualizationPhase2-, EqualizationPhase3-, LinkEqualizationRequest-
```

3. Run RC sample application *ti-sample-vfio* with parameter *testbars*. Open a second Linux terminal and check PCIe EP configuration space. As the program halts after VFIO initialization, corresponding BARs can now be enabled (not shown as disabled) as shown in the following figure.

```
root@sitarampuapps-ThinkStation-P620:/home/sitara_mpu_apps# ./ti-sample-vfio 4 0 0 60
bash: ./ti-sample-vfio: No such file or directory
root@sitarampuapps-ThinkStation-P620:/home/sitara_mpu_apps# lspci -vvv -s 04:00.0
04:00.0 Gameport controller: Cadence Design Systems, Inc. Device 0100 (rev 03) (prog-if 10 [Extended])
        Subsystem: Cadence Design Systems, Inc. Device 0000
        Control: I/O+ Mem+ BusMaster- SpecCycle- MemWINV- VGASnoop- ParErr- Stepping- SERR- FastB2B- DisINTx-
        Status: Cap+ 66MHz- UDF- FastB2B- ParErr- DEVSEL=fast >TAbort- <TAbort- <MAbort- >SERR- <PERR- INTx-
        Region 0: Memory at f3800000 (32-bit, non-prefetchable) [size=32K]
        Region 1: Memory at f3f00000 (32-bit, prefetchable) [size=1M]
        Region 2: Memory at f3600000 (32-bit, non-prefetchable) [size=2M]
        Region 3: I/O ports at 3400 [size=128]
        Region 4: I/O ports at 3000 [size=1K]
        Region 5: I/O ports at 2000 [size=4K]
        Capabilities: [80] Power Management version 3
                Flags: PMEClk- DSI- D1+ D2- AuxCurrent=0mA PME(D0+,D1+,D2-,D3hot+,D3cold-)
                Status: D3 NoSoftRst+ PME-Enable- DSel=0 DScale=0 PME-
        Capabilities: [90] MSI: Enable- Count=1/16 Maskable- 64bit+
                Address: 0000000000000000  Data: 0000
        Capabilities: [b0] Null
        Capabilities: [c0] Express (v2) Endpoint, MSI 00
                DevCap: MaxPayload 128 bytes, PhantFunc 0, Latency L0s <1us, L1 <1us
                        ExtTag- AttnBtn- AttnInd- PwrInd- RBE+ FLReset- SlotPowerLimit 0.000W
                DevCtl: CorrErr+ NonFatalErr+ FatalErr+ UnsupReq-
                        RlxdOrd+ ExtTag- PhantFunc- AuxPwr- NoSnoop+
                        MaxPayload 128 bytes, MaxReadReq 512 bytes
                DevSta: CorrErr- NonFatalErr- FatalErr- UnsupReq- AuxPwr- TransPend-
                LnkCap: Port #0, Speed 5GT/s, Width x1, ASPM L1, Exit Latency L1 <8us
                        ClockPM- Surprise- LLActRep- BwNot- ASPMOptComp+
                LnkCtl: ASPM Disabled; RCB 64 bytes Disabled- CommClk+
                        ExtSynch- ClockPM- AutWidDis- BWInt- AutBWInt-
                LnkSta: Speed 5GT/s (ok), Width x1 (ok)
                        TrErr- Train- SlotClk+ DLActive- BWMgmt- ABWMgmt-
                DevCap2: Completion Timeout: Range B, TimeoutDis+, NROPrPrP-, LTR+
                         10BitTagComp-, 10BitTagReq-, OBFF Via message, ExtFmt+, EETLPPrefix+, MaxEETLPPrefixes 1
                         EmergencyPowerReduction Not Supported, EmergencyPowerReductionInit-
                         FRS-, TPHComp-, ExtTPHComp-
                         AtomicOpsCap: 32bit- 64bit- 128bitCAS-
                DevCtl2: Completion Timeout: 50us to 50ms, TimeoutDis-, LTR+, OBFF Disabled
                         AtomicOpsCtl: ReqEn-
                LnkCtl2: Target Link Speed: 5GT/s, EnterCompliance- SpeedDis-
                         Transmit Margin: Normal Operating Range, EnterModifiedCompliance- ComplianceSOS-
                         Compliance De-emphasis: -6dB
                LnkSta2: Current De-emphasis Level: -3.5dB, EqualizationComplete-, EqualizationPhase1-
                         EqualizationPhase2-, EqualizationPhase3-, LinkEqualizationRequest-
```

4. Continue RC sample application *ti-sample-vfio*. The program can perform an extended BAR test and output BAR information as shown in the following figure.

```
root@sitarampuapps-ThinkStation-P620:/home/sitara_mpu_apps/Downloads/IBV-PCIE/IBV-PCIE-1.0/PCIe EP Driver - 1.0/src/ti-sample-vfio# ./ti-sample-vfio 4 0 0 60 testbars
--------------------------------------------------------------------------------
Starting PCIe RC VFIO test application with parameters:
PCIe Bus Number: 4
PCIe Device Number: 0
PCIe Function Number: 0
PCIe IOMMU Number: 60
Test mode: extended BAR test
--------------------------------------------------------------------------------
Using PCI device 0000:04:00.0 in IOMMU group 60
VFIO_CHECK_EXTENSION VFIO_TYPE1_IOMMU: Present
VFIO_CHECK_EXTENSION VFIO_NOIOMMU_IOMMU: Not Present
Config region info: region index 0x7, size 0x1000, offset 0x70000000000, cap_offset 0x0, flags 0x3
BAR0 Info: size 0x8000, offset 0x0, flags 0x7
BAR1 Info: size 0x100000, offset 0x10000000000, flags 0x7
BAR2 Info: size 0x200000, offset 0x20000000000, flags 0x7
BAR3 Info: size 0x80, offset 0x30000000000, flags 0x3
BAR4 Info: size 0x400, offset 0x40000000000, flags 0x3
BAR5 Info: size 0x1000, offset 0x50000000000, flags 0x3
Extended BAR test passed
root@sitarampuapps-ThinkStation-P620:/home/sitara_mpu_apps/Downloads/IBV-PCIE/IBV-PCIE-1.0/PCIe EP Driver - 1.0/src/ti-sample-vfio#
```

5.4 Outbound ATU Functionalities

Test

Description:

Test to verify if several PCIe outbound ATU configurations work correctly for the TMDS243EVM/TMDS64EVM PCIe EP.

The *pcie_enumerate_ep* application implements two outbound mappings. The first realizes a DMA mapping from internal “PCIE0_DAT0” window to a corresponding PCIe address. This DMA outbound mapping is used to write

back received data from the RC. The second serves the MSI mechanism and maps the local MSI address configured by the RC to a corresponding PCIe address.

Execution:

1. Run PCIe EP application *pcie_enumerate_ep*.
2. Run Linux-based RC test application *ti-sample-vfio*.
3. Check status of both programs. As *ti-sample-vfio* waits for MSI interrupts generated by the PCIe EP and checks the received data, it can terminate without any failure ensuring correct functionality of PCIe EP outbound mapping.

5.5 MSI Functionality

Test

Description:

Test to verify if MSI IRQs are sent correctly from the PCIe EP to the address configured by the RC.

Execution:

1. Run PCIe EP application *pcie_enumerate_ep*.
2. Run Linux-based RC test application *ti-sample-vfio*.
3. Check status of both programs. As *ti-sample-vfio* waits for MSI IRQs sent by PCIe EP on specified address, it can terminate without any failure ensuring correct functionality of PCIe EP MSI mechanism.

Test

Description:

Test to verify if the maximum number of different MSI IRQs (multi-message capable) available in the PCIe EP is determined correctly in the PCIe RC and if a reduced number of desired MSI IRQs (multi-message enable) can be requested from the RC.

Execution:

1. Run PCIe EP application *pcie_enumerate_ep* with the default number of MSI IRQs set as 16.
2. On Linux-based RC hardware check PCIe EP MSI capability on offset 90. The MSI capability can be disabled with a count set to 1 out of 16 as shown in the following figure.
3. Run Linux-based RC test application *ti-sample-vfio*. Pass the default number of configured MSI IRQs as the desired number of MSI IRQs being tested as the fifth parameter:

```
sudo ./ti-sample-vfio 9 0 0 19 16
```

4. Continue *ti-sample-vfio* with enter until the program halts at status output *Initialize MSI test. Expect 16 distinct MSI IRQs*.
5. Open a second Linux terminal and check the PCIe EP MSI capability at offset 90. The MSI mechanism can be enabled with a count of 16 as shown in the following figure.
6. Continue *ti-sample-vfio*. The program can continue normally, perform an extended MSI test with 16 MSI IRQs and end without any failure as shown in the following figure.
7. Run Linux-based RC test application *ti-sample-vfio*. Configure the number of MSI IRQs to be tested with less than the default number, for example, 8:

```
sudo ./ti-sample-vfio 9 0 0 19 8
```

8. Continue *ti-sample-vfio* with enter until the program halts at status output
   a. **Initialize MSI test. Expect 8 distinct MSI IRQs.**
9. Check the PCIe EP MSI capability at offset 90 on second Linux terminal. The MSI capability can be enabled with the count 8 out of 16 as shown in the following figure.
10. Continue *ti-sample-vfio*. The program can continue normally, perform an extended MSI test with 8 MSI IRQs and end without any failure as shown in the following figure.

11. On Linux-based RC hardware check PCIe EP MSI capability on offset 90. The MSI capability can be disabled with a count set to 1 out of 16 as shown in the following figure.

```
root@sitarampuapps-ThinkStation-P620:/home/sitara_mpu_apps/Downloads/IBV-PCIE/IBV-PCIE-1.0/PCIe EP Driver - 1.0/src/ti-sample-vfio# ./ti-sample-vfio 4 0 0 60 16
--------------------------------------------------------------------------------
Starting PCIe RC VFIO test application with parameters:
PCIe Bus Number: 4
PCIe Device Number: 0
PCIe Function Number: 0
PCIe IOMMU Number: 60
Test mode: default test
MSI IRQ number: 16
Iteration number: 10
--------------------------------------------------------------------------------
Using PCI device 0000:04:00.0 in IOMMU group 60
VFIO_CHECK_EXTENSION VFIO_TYPE1_IOMMU: Present
VFIO_CHECK_EXTENSION VFIO_NOIOMMU_IOMMU: Not Present
Config region info: region index 0x7, size 0x1000, offset 0x70000000000, cap_offset 0x0, flags 0x3
BAR0 Info: size 0x8000, offset 0x0, flags 0x7
MSI IRQ Info: index: 1, count: 16, flags: 9
RC completed EP initialization
--------------------------------------------------------------------------------
Start COPY test
COPY test passed with 10 loops
--------------------------------------------------------------------------------
Initialize MSI test. Expect 16 distinct MSI IRQs
Expect MSI IRQ nr. 0
Expect MSI IRQ nr. 1
Expect MSI IRQ nr. 2
Expect MSI IRQ nr. 3
Expect MSI IRQ nr. 4
Expect MSI IRQ nr. 5
Expect MSI IRQ nr. 6
Expect MSI IRQ nr. 7
Expect MSI IRQ nr. 8
Expect MSI IRQ nr. 9
Expect MSI IRQ nr. 10
Expect MSI IRQ nr. 11
Expect MSI IRQ nr. 12
Expect MSI IRQ nr. 13
Expect MSI IRQ nr. 14
Expect MSI IRQ nr. 15
MSI test passed
--------------------------------------------------------------------------------
Initialize BARs test
BAR1 Info: size 0x100000, offset 0x10000000000, flags 0x7
BAR2 Info: size 0x200000, offset 0x20000000000, flags 0x7
BAR test passed
```

Test

Description:

Test to verify is MSI per vector masking is disabled correctly at TMDS243EVM/TMDS64EVM PCIe EP.

Execution:

1. On Linux-based RC hardware, check PCIe EP MIS capability with Linux terminal. The maskable filed can be disabled as shown in the following figure.

```
root@sitarampuapps-ThinkStation-P620:/home/sitara_mpu_apps# lspci -vvv -s 04:00.0
04:00.0 Gameport controller: Cadence Design Systems, Inc. Device 0100 (rev 03) (prog-if 10 [Extended])
        Subsystem: Cadence Design Systems, Inc. Device 0000
        Control: I/O+ Mem+ BusMaster- SpecCycle- MemWINV- VGASnoop- ParErr- Stepping- SERR- FastB2B- DisINTx-
        Status: Cap+ 66MHz- UDF- FastB2B- ParErr- DEVSEL=fast >TAbort- <TAbort- <MAbort- >SERR- <PERR- INTx-
        Region 0: Memory at f3800000 (32-bit, non-prefetchable) [size=32K]
        Region 1: Memory at f3f00000 (32-bit, prefetchable) [size=1M]
        Region 2: Memory at f3600000 (32-bit, non-prefetchable) [size=2M]
        Region 3: I/O ports at 3400 [size=128]
        Region 4: I/O ports at 3000 [size=1K]
        Region 5: I/O ports at 2000 [size=4K]
        Capabilities: [80] Power Management version 3
                Flags: PMEClk- DSI- D1+ D2- AuxCurrent=0mA PME(D0+,D1+,D2-,D3hot+,D3cold-)
                Status: D3 NoSoftRst+ PME-Enable- DSel=0 DScale=0 PME-
        Capabilities: [90] MSI: Enable- Count=1/16 Maskable- 64bit+
                Address: 0000000000000000  Data: 0000
        Capabilities: [b0] Null
        Capabilities: [c0] Express (v2) Endpoint, MSI 00
                DevCap: MaxPayload 128 bytes, PhantFunc 0, Latency L0s <1us, L1 <1us
                        ExtTag- AttnBtn- AttnInd- PwrInd- RBE+ FLReset- SlotPowerLimit 0.000W
```

5.6 Downstream Interrupt Functionality

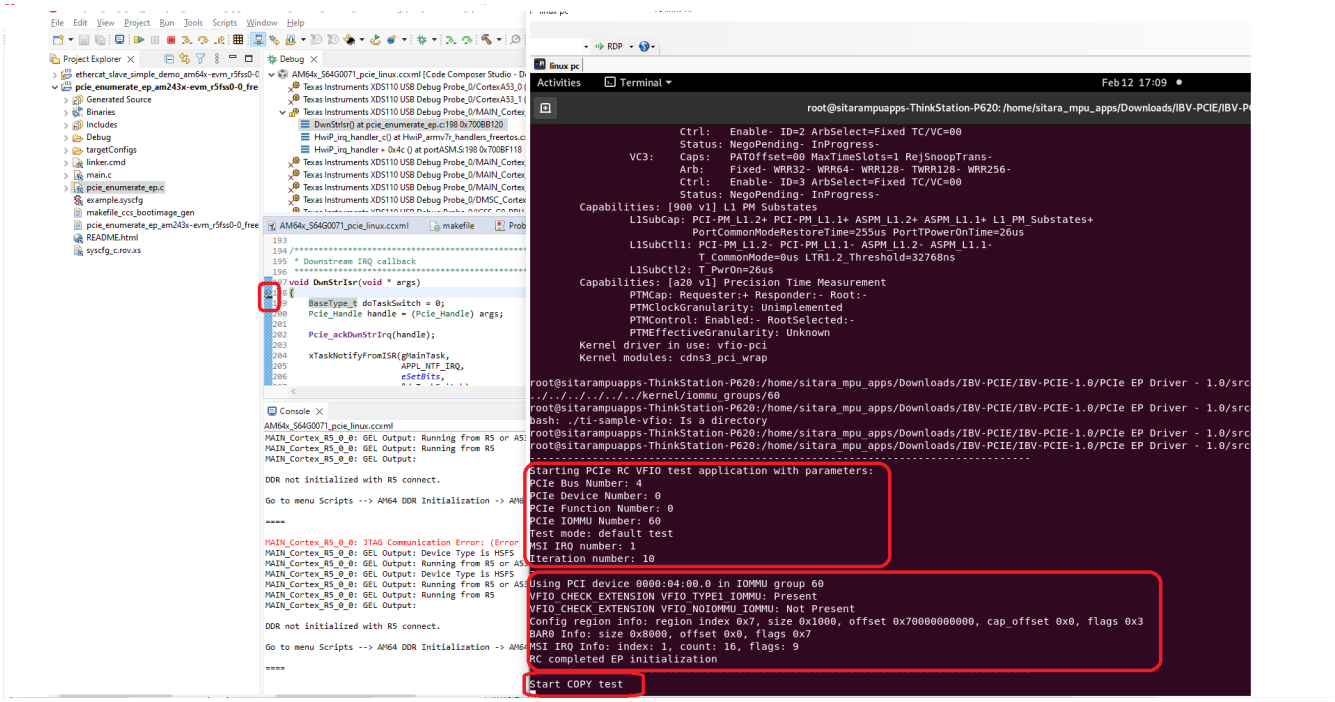
Test

Description:

Test to verify if downstream interrupt functionality can be configured and triggered at TMDS243EVM/ TMDS64EVM PCIe EP.

Execution:

1. On the *pcie_enumeraute_ep* sample application set a breakpoint within the downstream interrupt service routine *void DwnStrIsr(void *args)*, and so on at line 196.
2. Perform system reset, load and run *pcie_enumerate_ep* on TMDS243EVM PCIe EP.
3. On Linux-based RC hardware use GNU Debugger to start RC sample application *ti-sample-vfio*. Set a breakpoint at *sendDwnStrIrq* function, and so on at line 623.
4. Run *ti-sample-vfio*. As the program stops at *sendDwnStrIrq* continue and check if a downstream interrupt is triggered at *pcie_enumerate_ep*
5. On positive test result *pcie_enumerate_ep* can halt at *void DwnStrIsr(void *args)* indicating that a downstream interrupt is triggered successfully.



5.7 Device Power Management State Functionality

Test

Description:

Test to verify if power management states work correctly at TMDS243EVM/TMDS64EVM PCIe EP.

Execution:

1. On Linux-based RC hardware, check PCIe EP power management state with Linux terminal. On boot up, the PCIe EP can be in power management state D0, as shown in the following figure.

```
COM180
DMSC ABI revision 3.1

INFO: Bootloader_runCpu:155: CPU r5f1-0  is initialized to 800000000 Hz !!!
INFO: Bootloader_runCpu:155: CPU r5f1-1 is initialized to 800000000 Hz !!!
INFO: Bootloader_runCpu:155: CPU m4f0-0 is initialized to 400000000 Hz !!!
INFO: Bootloader_runCpu:155: CPU a530-0 is initialized to 800000000 Hz !!!
INFO: Bootloader_runCpu:155: CPU a530-1 is initialized to 800•ÁÖÓ&vÒ6-IW3¦KL•is
 initialized to 800000000 Hz !!!
INFO: Bootloader_runSelfCpu:217: All done, reseting self ...

PCIe: EP initialized and waiting for link
PCIe: link detected
PCIe link parameter: PCIe Gen2 with 5.0 GT/s speed, number of lanes: 1
EP is in D0 state
PCIe: signaling APPL ready
APPL: pcie ready
```

2. Bind VFIO driver to TMDS243EVM PCIe EP.
3. As VFIO is bind to PCIe EP as a kernel driver, the PCIe EP power management state can change to D3hot. Verify the power management state with Linux terminal as shown in the following figure:

```
root@sitarampuapps-ThinkStation-P620:/home/sitara_mpu_apps# modprobe vfio-pci
root@sitarampuapps-ThinkStation-P620:/home/sitara_mpu_apps# echo "17cd 0100" > /sys/bus/pci/drivers/vfio-pci/new_id
root@sitarampuapps-ThinkStation-P620:/home/sitara_mpu_apps# lspci -vvv -s 04:00.0
04:00.0 Gameport controller: Cadence Design Systems, Inc. Device 0100 (rev 03) (prog-if 10 [Extended])
        Subsystem: Cadence Design Systems, Inc. Device 0000
        Control: I/O- Mem- BusMaster- SpecCycle- MemWINV- VGASnoop- ParErr- Stepping- SERR- FastB2B- DisINTx-
        Status: Cap+ 66MHz- UDF- FastB2B- ParErr- DEVSEL=fast >TAbort- <TAbort- <MAbort- >SERR- <PERR- INTx-
        Region 0: Memory at f3800000 (32-bit, non-prefetchable) [disabled] [size=32K]
        Region 1: Memory at f3f00000 (32-bit, prefetchable) [disabled] [size=1M]
        Region 2: Memory at f3600000 (32-bit, non-prefetchable) [disabled] [size=2M]
        Region 3: I/O ports at 3400 [disabled] [size=128]
        Region 4: I/O ports at 3000 [disabled] [size=1K]
        Region 5: I/O ports at 2000 [disabled] [size=4K]
        Capabilities: [80] Power Management version 3
                Flags: PMEClk- DSI- D1+ D2- AuxCurrent=0mA PME(D0+,D1+,D2-,D3hot+,D3cold-)
                Status: D3 NoSoftRst+ PME-Enable- DSel=0 DScale=0 PME-
        Capabilities: [90] MSI: Enable- Count=1/16 Maskable- 64bit+
                Address: 0000000000000000  Data: 0000
        Capabilities: [b0] Null
        Capabilities: [c0] Express (v2) Endpoint, MSI 00
```

```
sitara_mpu_apps@sitarampuapps-ThinkStation-P620:~$ sudo su
[sudo] password for sitara_mpu_apps:
root@sitarampuapps-ThinkStation-P620:/home/sitara_mpu_apps# lspci -vvv -s 04:00.0
04:00.0 Gameport controller: Cadence Design Systems, Inc. Device 0100 (rev 03) (prog-if 10 [Extended])
        Subsystem: Cadence Design Systems, Inc. Device 0000
        Control: I/O- Mem- BusMaster- SpecCycle- MemWINV- VGASnoop- ParErr- Stepping- SERR- FastB2B- DisINTx-
        Status: Cap+ 66MHz- UDF- FastB2B- ParErr- DEVSEL=fast >TAbort- <TAbort- <MAbort- >SERR- <PERR- INTx-
        Region 0: Memory at f3800000 (32-bit, non-prefetchable) [disabled] [size=32K]
        Region 1: Memory at f3f00000 (32-bit, prefetchable) [disabled] [size=1M]
        Region 2: Memory at f3600000 (32-bit, non-prefetchable) [disabled] [size=2M]
        Region 3: I/O ports at 3400 [disabled] [size=128]
        Region 4: I/O ports at 3000 [disabled] [size=1K]
        Region 5: I/O ports at 2000 [disabled] [size=4K]
        Capabilities: [80] Power Management version 3
                Flags: PMEClk- DSI- D1+ D2- AuxCurrent=0mA PME(D0+,D1+,D2-,D3hot+,D3cold-)
                Status: D0 NoSoftRst+ PME-Enable- DSel=0 DScale=0 PME-
        Capabilities: [90] MSI: Enable- Count=1/16 Maskable- 64bit+
                Address: 0000000000000000  Data: 0000
        Capabilities: [b0] Null
        Capabilities: [c0] Express (v2) Endpoint, MSI 00
```

4. Open *ti-sample-vfio* with GNU debugger. Set a breakpoint after *initVFIO* function call within *main* at line 583. Run *ti-sample-vfio*.
5. As the *ti-sample-vfio* program halts after *initVFIO* and *initVFIO* initializes the PCIe EP device, the power management state can change to D0. Verify the power management state with Linux terminal as shown in the following figure:

```
COM180
INFO: Bootloader_runCpu:155: CPU m4f0-0 is initialized to 400000000 Hz !!!
INFO: Bootloader_runCpu:155: CPU a530-0 is initialized to 800000000 Hz !!!
INFO: Bootloader_runCpu:155: CPU a530-1 is initialized to 800•ÁÖÓ&▯vÒ▯6▯-▯IW3▯¦KL•is
 initialized to 800000000 Hz !!!
INFO: Bootloader_runSelfCpu:217: All done, reseting self ...

PCIe: EP initialized and waiting for link
PCIe: link detected
PCIe link parameter: PCIe Gen2 with 5.0 GT/s speed, number of lanes: 1
EP is in D0 state
PCIe: signaling APPL ready
APPL: pcie ready
PCIe: power state entry
EP is in D3hot state
PCIe: signaling APPL halt
APPL: pcie not ready
```

5.8 Function Level Reset Mechanism

Test

Description:

Test to verify if the Function Level Reset (FLR) mechanism is disabled correctly at TMDS243EVM/TMDS64EVM PCIe EP.

Execution:

1. On Linux-based RC hardware, check if the FLR mechanism is disabled at the PCIe EP. This can be checked at the device capabilities as shown in the following figure

```
root@sitarampuapps-ThinkStation-P620:/home/sitara_mpu_apps# lspci -vvv -s 04:00.0
04:00.0 Gameport controller: Cadence Design Systems, Inc. Device 0100 (rev 03) (prog-if 10 [Extended])
        Subsystem: Cadence Design Systems, Inc. Device 0000
        Control: I/O- Mem- BusMaster- SpecCycle- MemWINV- VGASnoop- ParErr- Stepping- SERR- FastB2B- DisINTx-
        Status: Cap+ 66MHz- UDF- FastB2B- ParErr- DEVSEL=fast >TAbort- <TAbort- <MAbort- >SERR- <PERR- INTx-
        Region 0: Memory at f3800000 (32-bit, non-prefetchable) [disabled] [size=32K]
        Region 1: Memory at f3f00000 (32-bit, prefetchable) [disabled] [size=1M]
        Region 2: Memory at f3600000 (32-bit, non-prefetchable) [disabled] [size=2M]
        Region 3: I/O ports at 3400 [disabled] [size=128]
        Region 4: I/O ports at 3000 [disabled] [size=1K]
        Region 5: I/O ports at 2000 [disabled] [size=4K]
        Capabilities: [80] Power Management version 3
                Flags: PMEClk- DSI- D1+ D2- AuxCurrent=0mA PME(D0+,D1+,D2-,D3hot+,D3cold-)
                Status: D0 NoSoftRst+ PME-Enable- DSel=0 DScale=0 PME-
        Capabilities: [90] MSI: Enable- Count=1/16 Maskable- 64bit+
                Address: 0000000000000000  Data: 0000
        Capabilities: [b0] Null
        Capabilities: [c0] Express (v2) Endpoint, MSI 00
                DevCap: MaxPayload 128 bytes, PhantFunc 0, Latency L0s <1us, L1 <1us
                        ExtTag- AttnBtn- AttnInd- PwrInd- RBE+ FLReset- SlotPowerLimit 0.000W
                DevCtl: CorrErr+ NonFatalErr+ FatalErr+ UnsupReq-
                        RlxdOrd+ ExtTag- PhantFunc- AuxPwr- NoSnoop+
                        MaxPayload 128 bytes, MaxReadReq 512 bytes
                DevSta: CorrErr- NonFatalErr- FatalErr- UnsupReq- AuxPwr- TransPend-
                LnkCap: Port #0, Speed 5GT/s, Width x1, ASPM L1, Exit Latency L1 <8us
                        ClockPM- Surprise- LLActRep- BwNot- ASPMOptComp+
                LnkCtl: ASPM Disabled; RCB 64 bytes Disabled- CommClk+
                        ExtSynch- ClockPM- AutWidDis- BWInt- AutBWInt-
                LnkSta: Speed 5GT/s (ok), Width x1 (ok)
                        TrErr- Train- SlotClk+ DLActive- BWMgmt- ABWMgmt-
```

5.9 Legacy Interrupt Mechanism

Test

Description:

Test to verify if the legacy interrupt (INTx) mechanism is disabled correctly at TMDS243EVM/TMDS64EVM PCIe EP.

Execution:

1. On Linux-based RC hardware check the device status of PCIe EP in Linux terminal. For a valid test case, the information *Interrupt: pin A routed to IRQ XXX* cannot occur.

For comparison, the following figure shows the device status of a different PCIe EP on the same Linux-based RC hardware where the legacy interrupt mechanism is active.

```
ibv@debian:~$ sudo lspci -vvv -s 00:02.0
00:02.0 VGA compatible controller: Intel Corporation CometLake-S GT2 [UHD Graphics 630] (rev 03) (prog-if 00 [VGA controller])
        DeviceName: Onboard - Video
        Subsystem: Micro-Star International Co., Ltd. [MSI] CometLake-S GT2 [UHD Graphics 630]
        Control: I/O+ Mem+ BusMaster+ SpecCycle- MemWINV- VGASnoop- ParErr- Stepping- SERR- FastB2B- DisINTx+
        Status: Cap+ 66MHz- UDF- FastB2B- ParErr- DEVSEL=fast >TAbort- <TAbort- <MAbort- >SERR- <PERR- INTx-
        Latency: 0, Cache Line Size: 64 bytes
        Interrupt: pin A routed to IRQ 147
        IOMMU group: 0
        Region 0: Memory at 6000000000 (64-bit, non-prefetchable) [size=16M]
        Region 2: Memory at 4000000000 (64-bit, prefetchable) [size=256M]
        Region 4: I/O ports at 4000 [size=64]
        Expansion ROM at 000c0000 [virtual] [disabled] [size=128K]
        Capabilities: [40] Vendor Specific Information: Len=0c <?>
        Capabilities: [70] Express (v2) Root Complex Integrated Endpoint, MSI 00
                DevCap: MaxPayload 128 bytes, PhantFunc 0
                        ExtTag- RBE+ FLReset+
                DevCtl: CorrErr- NonFatalErr- FatalErr- UnsupReq-
```

5.10 MSI-X Capability

Test

Description:

Test to verify if the MSI-X capability is disabled correctly at TMDS243EVM/TMDS64EVM PCIe EP.

Execution:

1. On Linux-based RC hardware, check if the MSI-X capability is disabled at the PCIe EP. Since the MSI-X capability is on offset B0, this field can contain *NULL* if disabled, as shown in the following figure.

```
root@sitarampuapps-ThinkStation-P620:/home/sitara_mpu_apps# lspci -vvv -s 04:00.0
04:00.0 Gameport controller: Cadence Design Systems, Inc. Device 0100 (rev 03) (prog-if 10 [Extended])
        Subsystem: Cadence Design Systems, Inc. Device 0000
        Control: I/O- Mem- BusMaster- SpecCycle- MemWINV- VGASnoop- ParErr- Stepping- SERR- FastB2B- DisINTx-
        Status: Cap+ 66MHz- UDF- FastB2B- ParErr- DEVSEL=fast >TAbort- <TAbort- <MAbort- >SERR- <PERR- INTx-
        Region 0: Memory at f3800000 (32-bit, non-prefetchable) [disabled] [size=32K]
        Region 1: Memory at f3f00000 (32-bit, prefetchable) [disabled] [size=1M]
        Region 2: Memory at f3600000 (32-bit, non-prefetchable) [disabled] [size=2M]
        Region 3: I/O ports at 3400 [disabled] [size=128]
        Region 4: I/O ports at 3000 [disabled] [size=1K]
        Region 5: I/O ports at 2000 [disabled] [size=4K]
        Capabilities: [80] Power Management version 3
                Flags: PMEClk- DSI- D1+ D2- AuxCurrent=0mA PME(D0+,D1+,D2-,D3hot+,D3cold-)
                Status: D0 NoSoftRst+ PME-Enable- DSel=0 DScale=0 PME-
        Capabilities: [90] MSI: Enable- Count=1/16 Maskable- 64bit+
                Address: 0000000000000000  Data: 0000
        Capabilities: [b0] Null
        Capabilities: [c0] Express (v2) Endpoint, MSI 00
```

5.11 Hot Reset Mechanism

Test

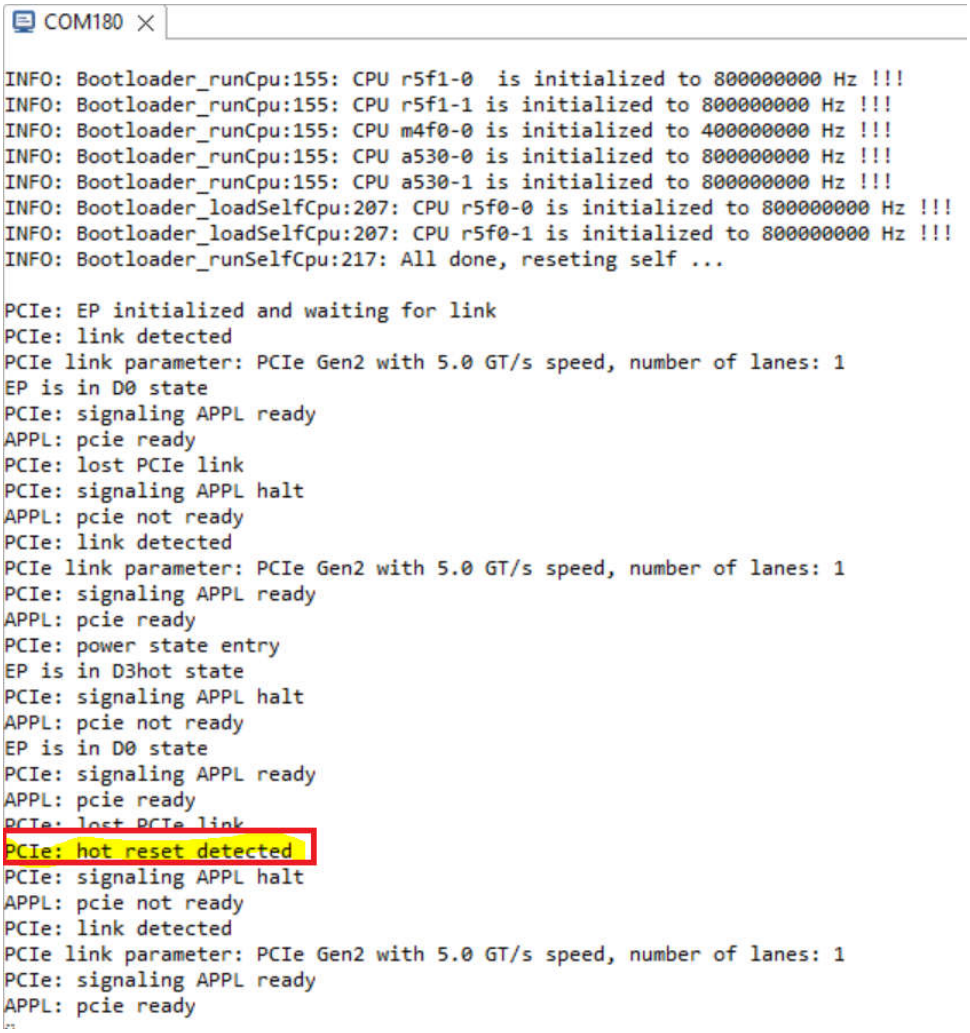
Description:

Test to verify if the hot reset mechanism works correctly at TMDS243EVM/TMDS64EVM PCIe EP.

Execution:

1. On PCIe EP sample application *pcie_enumerate_ep* set a breakpoint within the function *HotResetIsr*, for example, at line 178. As this function is the interrupt service routine for the corresponding mechanism the program can halt there.

2. Run *ti-sample-vfio* on Linux-based RC hardware.
3. Check if *pcie_enumerate_ep* halts at described breakpoint as shown in the following figure.
4. Continue *pcie_enumerate_ep* and check its status through SER_TER. The status can message *PCIe: hot reset detected* as shown in the following figure.



6 Windows Example Driver Verification

This chapter defines and specifies testing of the Windows example driver. Only a reduced subset of the EP's functionality is being tested, to make sure that functions previously tested on Linux work similarly on Windows. The following test specification assumes environment AM24_WIN.

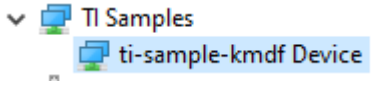
Test

Description:

Test to verify functionality of the Windows KMDF driver.

Execution:

1. Verify that the Windows driver for the pcie_enumerate_ep example EP has been loaded by searching for the ti-sample-kmdf Device in the Windows device manager:



2. Open a command prompt with administrator privileges and run the ti-sample-console.exe application:

```
C:\Users\ .Downloads\PCIe EP Driver - 1.0-updated-output\PCIe EP Driver - 1.0\src\ti-sample-kmdf\x64\Release>ti-s
ample-console.exe
--------------------------------------------------------------------------------
Starting PCIe RC KMDF test application
--------------------------------------------------------------------------------
Opening windows kernel mode driver \\.\SampleTI
--------------------------------------------------------------------------------
Start COPY test
IOCTL_TISAMPLEKMDF_TEST_DMA returned data, verifying...
COPY test passed
--------------------------------------------------------------------------------
Start MSI test
IOCTL_TISAMPLEKMDF_TEST_MSI returned, result: 0000ffff
MSI test passed
--------------------------------------------------------------------------------
Start Bar1/2 test
IOCTL_TISAMPLEKMDF_TEST_BARS returned, result: 00000001
BAR test passed
--------------------------------------------------------------------------------
Closing windows kernel mode driver
KMDF test application done
--------------------------------------------------------------------------------

C:\Users\ Downloads\PCIe EP Driver - 1.0-updated-output\PCIe EP Driver - 1.0\src\ti-sample-kmdf\x64\Release>
```

3. Verify that all tests have passed without errors as shown above.
4. Verify that the output on the EP's UART matches the expected output, indicating completion of the DMA test, the MSI test (sending 16 distinct interrupts) and the BAR test:

```
DMA test done
Send MSI IRQ nr. 0
Send MSI IRQ nr. 1
Send MSI IRQ nr. 2
Send MSI IRQ nr. 3
Send MSI IRQ nr. 4
Send MSI IRQ nr. 5
Send MSI IRQ nr. 6
Send MSI IRQ nr. 7
Send MSI IRQ nr. 8
Send MSI IRQ nr. 9
Send MSI IRQ nr. 10
Send MSI IRQ nr. 11
Send MSI IRQ nr. 12
Send MSI IRQ nr. 13
Send MSI IRQ nr. 14
Send MSI IRQ nr. 15
MSI test done
BAR test done
```

Rationale

The ti-sample-console application calls the ti-sample-kmdf driver and executes the following test steps:

- A COPY test where data previously written by the KMDF driver to the EP's Bar0 memory is correctly sent back to the Windows host's DMA buffer
- A MSI test where the EP triggers every enabled MSI vector (multiple message enable) once. The KMDF driver triggers this test in the EP and waits for the reception of all MSI vectors. If the test returns, all configured vectors have been received. Additionally, a bitmask of received MSI vectors is displayed (for example,. result: 0000ffff indicates vectors 0-15 have been received).
- A Bar1/2 test where the KMDF driver fills Bar1 and Bar2 of the EP with a known pattern, then triggers test execution in the EP. The EP verifies the known pattern in Bar1 and Bar2 and on success sends an MSI back to the RC. If the test returns, the verification was successful.

7 References

- Texas Instruments, *AM64x MCU+ SDK: PCIE*
- Texas Instruments, *PCIe End Point — Processor SDK AM64X Documentation*
- Texas Instruments, *PCIe Root Complex — Processor SDK AM64X Documentation*
- Texas Instruments, *TMDS243EVM Evaluation board*
- Texas Instruments, *TMDS64EVM Evaluation board*

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATA SHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, regulatory or other requirements.

These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products.

TI objects to and rejects any additional or different terms you may have proposed.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265