Avant-G/X PCIe Host DMA Driver Software User Guide

Technical Note

FPGA-TN-02405-1.0

March 2025

Disclaimers

Inclusive Language

This document was created consistent with Lattice Semiconductor’s inclusive language policy. In some cases, the language in underlying tools and other items may not yet have been updated. Please refer to Lattice’s inclusive language FAQ 6878 for a cross reference of terms. Note in some cases such as register names and state names it has been necessary to continue to utilize older terminology for compatibility.

Contents

Contents 3
Abbreviations in This Document 5
1. Introduction 6
1.1. Purpose 6
1.2. Audience 6
1.3. Driver Version 6
1.4. Driver and IP Compatibility 6
2. Software Setup 7
2.1. Driver Installation on a Windows Machine 7
2.2. Driver Installation on a Linux Machine 11
3. Application Overview 13
3.1. DMA Functionalities 13
3.1.1. DMA Transfer 14
3.1.2. DMA Transfer and Compare Data 15
3.1.3. DMA Transfer With Incremented Data and Compare Data 16
3.1.4. DMA Performance Measure 17
3.2. Read/Write Memory and I/O Addresses 18
3.2.1. Read/Write Mode 19
3.2.2. Transfer Type: Non-block or Block Transfers 20
3.2.3. Write, Read, and Compare Values 20
4. APIs 22
5. Software Flow Diagrams 23
5.1. AVANTDMA_DmaOpen 23
5.2. DmaPerformanceSingleDir 24
5.3. DmaKpPerfDevThread 25
Appendix A. Limitations 26
References 28
Technical Support Assistance 29
Revision History 30

Figures

Figure 2.1. Setup 7
Figure 2.2. Setup License Agreement 8
Figure 2.3. Setup Install Location 8
Figure 2.4. Setup Install 9
Figure 2.5. Device Manager 10
Figure 2.6. lsmod 11
Figure 2.7. Run avantdma 12
Figure 3.1. AVANTDMA DMA Menu 13
Figure 3.2. DMA Transfer Direction 14
Figure 3.3. DMA Data Compare 15
Figure 3.4 DMA Data Compare With Incremented Data 16
Figure 3.5. PCIe Link Status Register 17
Figure 3.6. Measure DMA Performance for Single Direction Transfer 17
Figure 3.7. Measure DMA Performance for Bi-Directional Simultaneous Transfer 18
Figure 3.8 Read/Write Memory and I/O Addresses 18
Figure 3.9 Change Active Address Space for Read/Write 19
Figure 3.10 Change Active Read/Write Mode 19
Figure 3.11 Non-Block or Block Transfers 20
Figure 3.12 Non-Block Transfer – Write, Read, and Compare Values 21
Figure 3.13 Block Transfer – Write, Read, and Compare Values 21
Figure 5.1. AVANTDMA_DmaOpen 23
Figure 5.2. DmaPerformanceSingleDir 24
Figure 5.3. DmaKpPerfDevThread 25
Figure A.1. Message-Signaled Interrupt Incorrectly Labeled 26
Figure A.2. DMA Performance Test Failure 27

Tables

Table 1.1. Driver and IP Compatibility 6
Table 1.2. Quick Facts of Driver Tested Environment 6
Table 4.1. List of APIs 22

Abbreviations in This Document

A list of abbreviations used in this document.

Abbreviation	Definition
API	Application Programming Interface
CPU	Central Processing Unit
DMA	Direct Memory Access
FPGA	Field Programmable Gate Array
OS	Operating System
PCIe	Peripheral Component Interconnect Express
RAM	Random Access Memory
RO	Read-Only
RW	Read-Write
SGDMA	Scatter-Gather Direct Memory Access
SPI	Serial Peripheral Interface
SRAM	Static Random Access Memory

1. Introduction

PCI Express® is a high performance, fully scalable, and well-defined standard for a wide variety of computing and communications platforms. Refer to the PCIe X8 IP Core User Guide (FPGA-IPUG-02243) for more details about the IP core.

This guide describes how to use the PCIe host driver software with the Lattice Avant™ Hardened DMA module.

The software driver is developed using Jungo WinDriver software toolkit. You can develop your own driver or use the Jungo WinDriver for driver development. To use Jungo WinDriver, contact Jungo to obtain a valid paid annual subscription. For more information, contact Jungo.

1.1. Purpose

This document is intended to act as a reference guide for developers by providing details of the C language driver APIs and function call flows.

1.2. Audience

The intended audience for this document includes embedded system designers and embedded software developers using Lattice Avant-G/X devices. The technical guide assumes readers have expertise in embedded systems and FPGA technologies.

1.3. Driver Version

PCIe Host DMA driver version 24.02.00.

1.4. Driver and IP Compatibility

Table 1.1. Driver and IP Compatibility

Driver Version	IP Version
24.02.00	2.2.0

Refer to the PCIe x8 IP Release Notes (FPGA-RN-02061) for more information on the driver and IP versions.

Table 1.2. Quick Facts of Driver Tested Environment

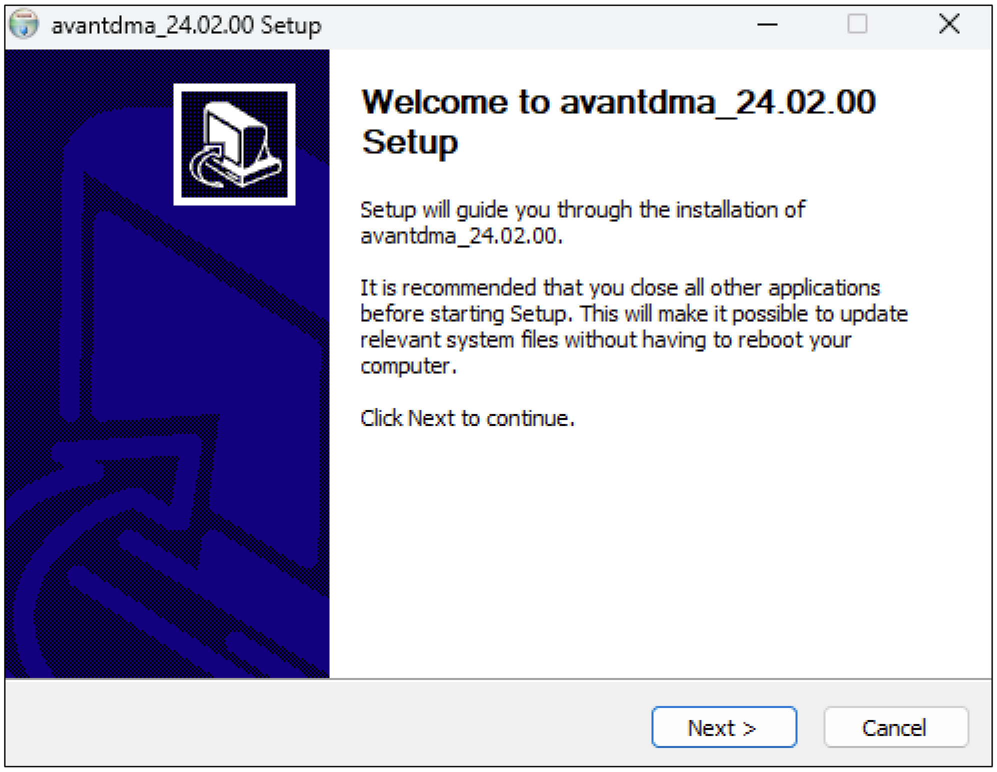
Driver Tested on Hardware Devices	PC Environment
Avant-X Versa Board	OS: Windows 10, Windows 11, Linux
	Supported architecture: x86_64
	CPU: Intel, AMD

2. Software Setup

This section describes the steps to install the software driver onto the host machine.

2.1. Driver Installation on a Windows Machine

1. If you are installing the driver for the first time, skip step 2 and go to step 3.
2. If the driver is already installed previously, run *Uninstall.exe* in the installation folder: *C:\Program Files\avantdma_24.02.00* before you install the driver.
3. Double click on *avantdma-24.02.00-win64.exe* to install the driver. Select **Next** to continue with the installation.



Welcome to avantdma_24.02.00 Setup

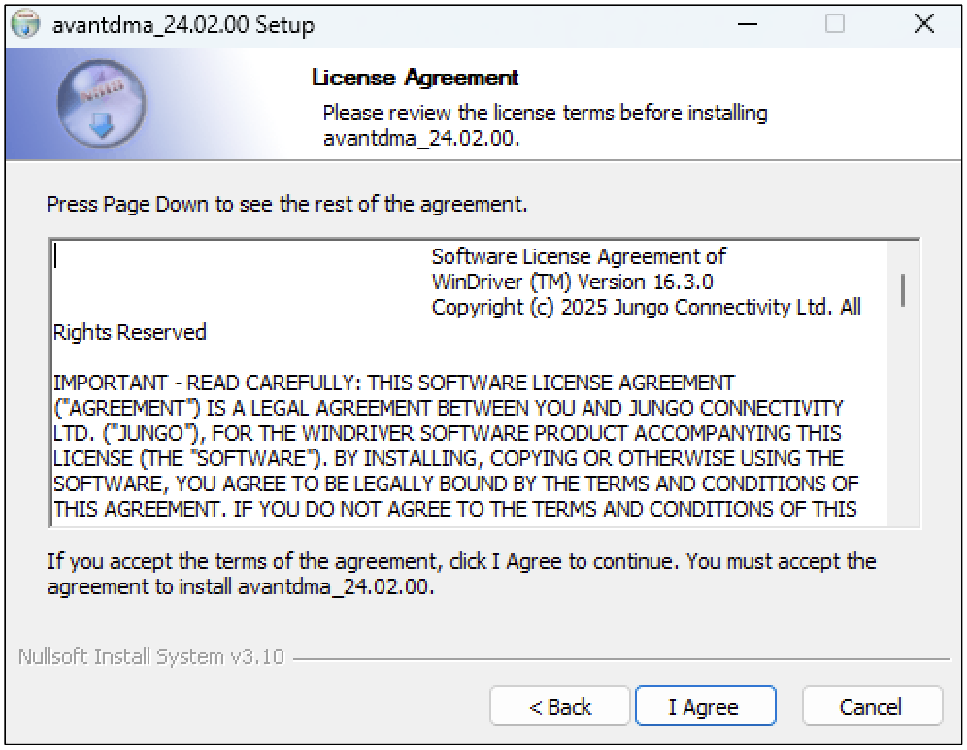
Setup will guide you through the installation of avantdma_24.02.00.

It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.

Click Next to continue.

Figure 2.1. Setup

4. Click **I Agree** to continue with the installation.



avantdma_24.02.00 Setup

License Agreement

Please review the license terms before installing avantdma_24.02.00.

Press Page Down to see the rest of the agreement.

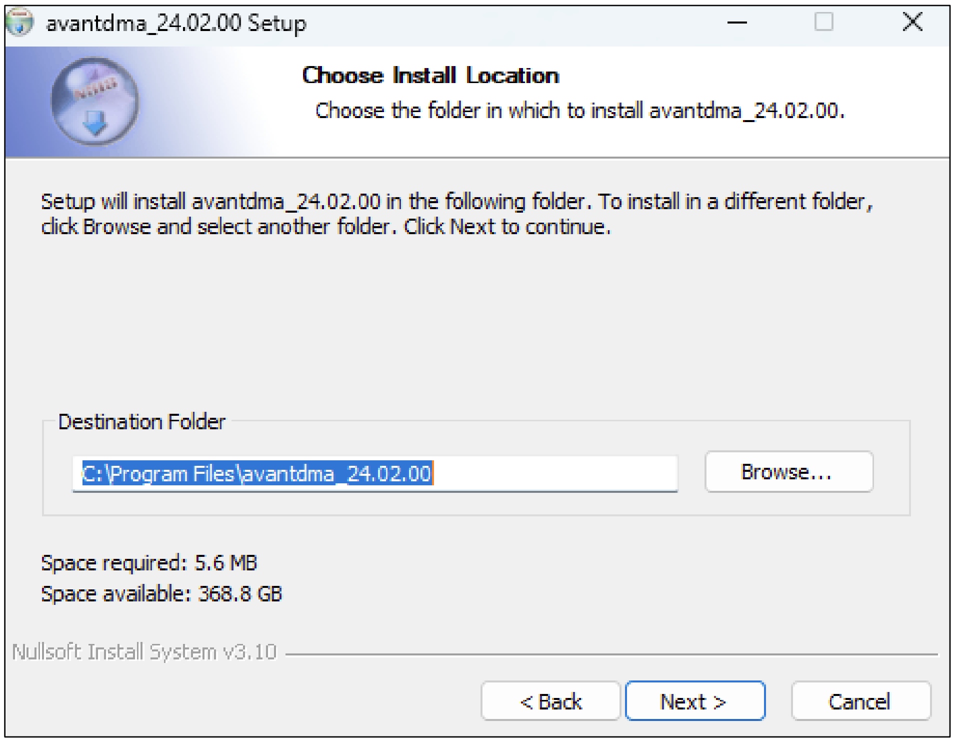
Software License Agreement of WinDriver (TM) Version 16.3.0
Copyright (c) 2025 Jungo Connectivity Ltd. All Rights Reserved

IMPORTANT - READ CAREFULLY: THIS SOFTWARE LICENSE AGREEMENT ("AGREEMENT") IS A LEGAL AGREEMENT BETWEEN YOU AND JUNGO CONNECTIVITY LTD. ("JUNGO"), FOR THE WINDRIVER SOFTWARE PRODUCT ACCOMPANYING THIS LICENSE (THE "SOFTWARE"). BY INSTALLING, COPYING OR OTHERWISE USING THE SOFTWARE, YOU AGREE TO BE LEGALLY BOUND BY THE TERMS AND CONDITIONS OF THIS AGREEMENT. IF YOU DO NOT AGREE TO THE TERMS AND CONDITIONS OF THIS

If you accept the terms of the agreement, click I Agree to continue. You must accept the agreement to install avantdma_24.02.00.

Figure 2.2. Setup License Agreement

5. Use the default location in the **Destination Folder** as the installation location and click **Next** to continue.



avantdma_24.02.00 Setup

Choose Install Location

Choose the folder in which to install avantdma_24.02.00.

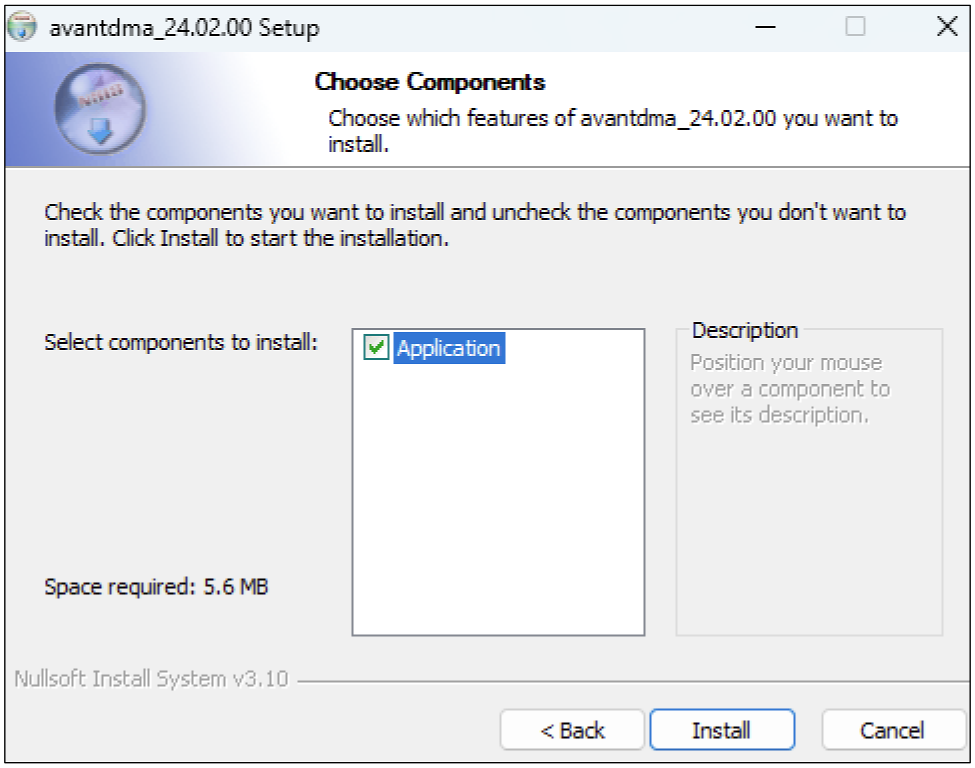
Setup will install avantdma_24.02.00 in the following folder. To install in a different folder, click Browse and select another folder. Click Next to continue.

Destination Folder: C:\Program Files\avantdma_24.02.00

Space required: 5.6 MB
Space available: 368.8 GB

Figure 2.3. Setup Install Location

6. Click **Next** until you see the window as shown in Figure 2.4. Click **Install**.


avantdma_24.02.00 Setup

Choose Components

Choose which features of avantdma_24.02.00 you want to install.

Check the components you want to install and uncheck the components you don't want to install. Click Install to start the installation.

Select components to install: [✓ Application]

Space required: 5.6 MB

Description

Position your mouse over a component to see its description.

Nullsoft Install System v3.10

Figure 2.4. Setup Install

7. When installation is complete, open **Device Manager**. Two new devices are listed under **Jungo Connectivity**:
   - avantdma Driver
   - PCIe (Gen4 - Lattice – Device ID : 0x9c25 (Requiring driver: avantdma)

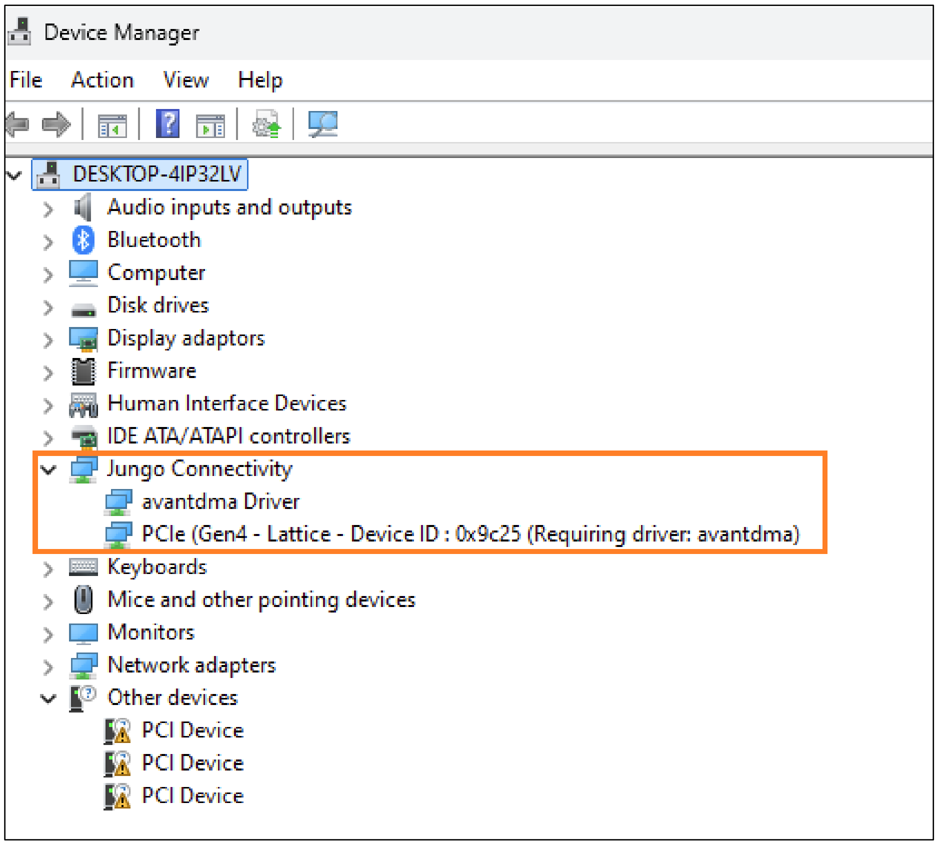


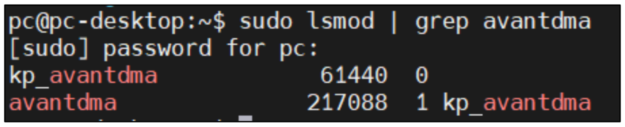
Figure 2.5. Device Manager

8. A new folder is created in *C:\Program Files\avantdma_24.02.00*.
9. A shortcut: *avantdma_24.02.00* is added to your desktop.
10. You can run the user app by running the desktop shortcut or by running: *C:\Program Files\avantdma_24.02.00\bin\avantdma.exe*.

Note: This driver is built with a demo license and is limited to 30 days. After 30 days, you need to replace the license with a valid license and rebuild the driver. For more information, refer to the Jungo WinDriver documentation. To continue using WinDriver drivers, you can get a valid paid annual subscription from Jungo. For more information, contact Jungo.

2.2. Driver Installation on a Linux Machine

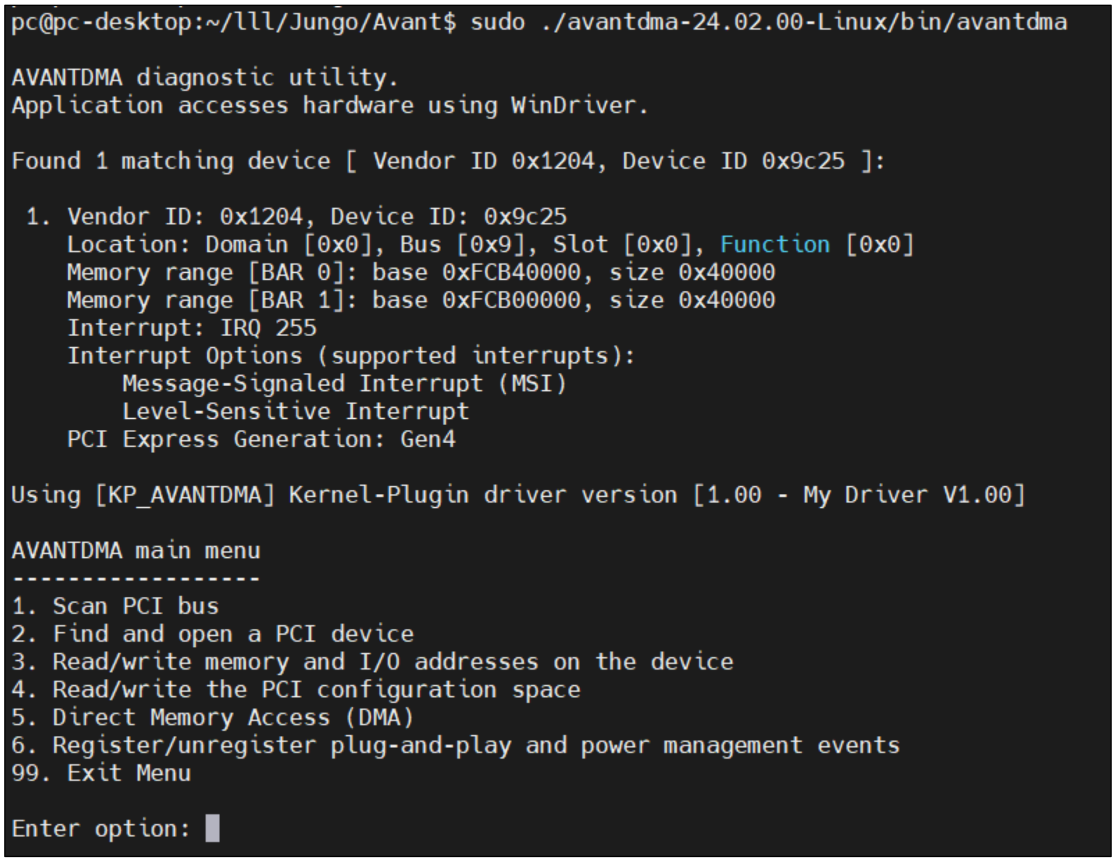
1. Before installing the driver, check if you have gcc installed by typing the following command in your terminal:
   ```
   gcc --version
   ```
2. If you see a message indicating that the command is not found, install *gcc*:
   ```
   sudo apt update
   sudo apt install gcc
   gcc --version
   ```
3. Go to the directory where file *avantdma-24.02.00-Linux.sh* is located. Give executable permission to the *.sh* file:
   ```
   sudo chmod +x avantdma-24.02.00-Linux.sh
   ```
4. Then, run the command below to build the driver and user application and install the driver.
   ```
   sudo ./avantdma-24.02.00-Linux.sh
   ```
5. Enter *Y* when prompted.
6. Two new modules are installed. Run the command below to verify that *avantdma* is installed:
   ```
   sudo lsmod | grep avantdma
   ```



```
pc@pc-desktop:~$ sudo lsmod | grep avantdma
[sudo] password for pc:
kp_avantdma              61440  0
avantdma                217088  1 kp_avantdma
```

Figure 2.6. lsmod

7. A new folder is created in your current directory *avantdma-24.02.00-Linux*.
8. Give permission to all the files in the directory:
   ```
   sudo chmod +x -R *
   ```
9. Run the user app:
   ```
   sudo ./avantdma-24.02.00-Linux/bin/avantdma
   ```



```
pc@pc-desktop:~/lll/Jungo/Avant$ sudo ./avantdma-24.02.00-Linux/bin/avantdma

AVANTDMA diagnostic utility.
Application accesses hardware using WinDriver.

Found 1 matching device [ Vendor ID 0x1204, Device ID 0x9c25 ]:

 1. Vendor ID: 0x1204, Device ID: 0x9c25
    Location: Domain [0x0], Bus [0x9], Slot [0x0], Function [0x0]
    Memory range [BAR 0]: base 0xFCB40000, size 0x40000
    Memory range [BAR 1]: base 0xFCB00000, size 0x40000
    Interrupt: IRQ 255
    Interrupt Options (supported interrupts):
        Message-Signaled Interrupt (MSI)
        Level-Sensitive Interrupt
    PCI Express Generation: Gen4

Using [KP_AVANTDMA] Kernel-Plugin driver version [1.00 - My Driver V1.00]

AVANTDMA main menu
------------------
1. Scan PCI bus
2. Find and open a PCI device
3. Read/write memory and I/O addresses on the device
4. Read/write the PCI configuration space
5. Direct Memory Access (DMA)
6. Register/unregister plug-and-play and power management events
99. Exit Menu

Enter option:
```

Figure 2.7. Run avantdma

Note: This driver is built with a demo license and is limited to one hour. You need to reinstall the driver after one hour. To continue using WinDriver drivers without having to reinstall every hour, contact Jungo to obtain a valid paid annual subscription. Contact Jungo for more information.

3. Application Overview

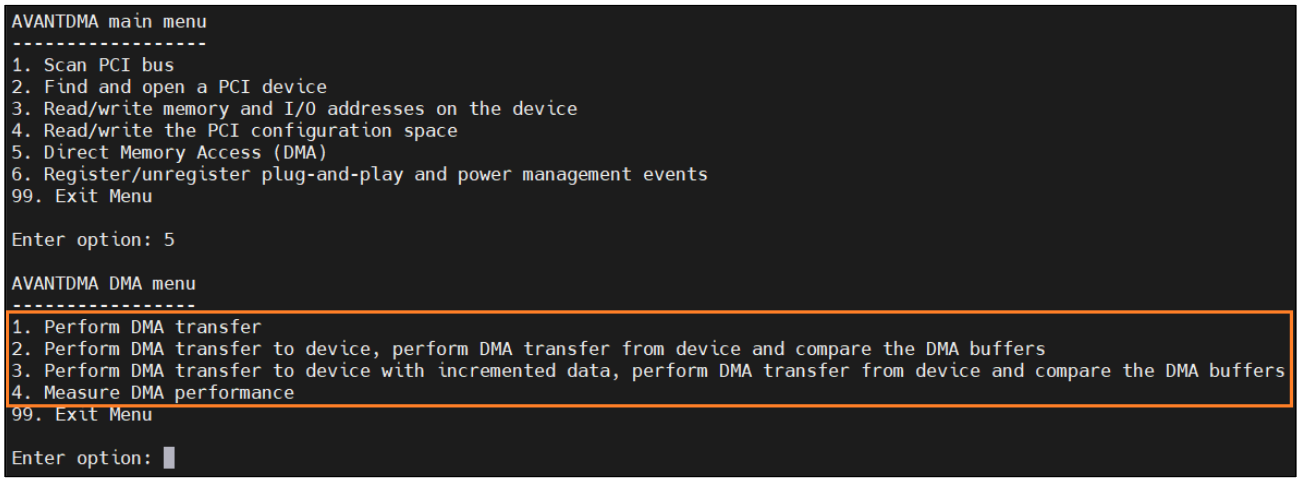
The PCIe software driver can run on both Windows and Linux operating systems. To run the software driver, you need to program the FPGA the example design to SPI flash or SRAM on the board.

The PCIe software driver is developed using the Jungo WinDriver software toolkit. It comes with a console-based user application that provides the following functionalities:

- Scan PCI bus
- Find and open a PCI device
- Read/write memory and I/O addresses on the device
- Read/Write the PCI configuration space
- Direct Memory Access (DMA)

3.1. DMA Functionalities

Below are the DMA functionalities provided by the user application.



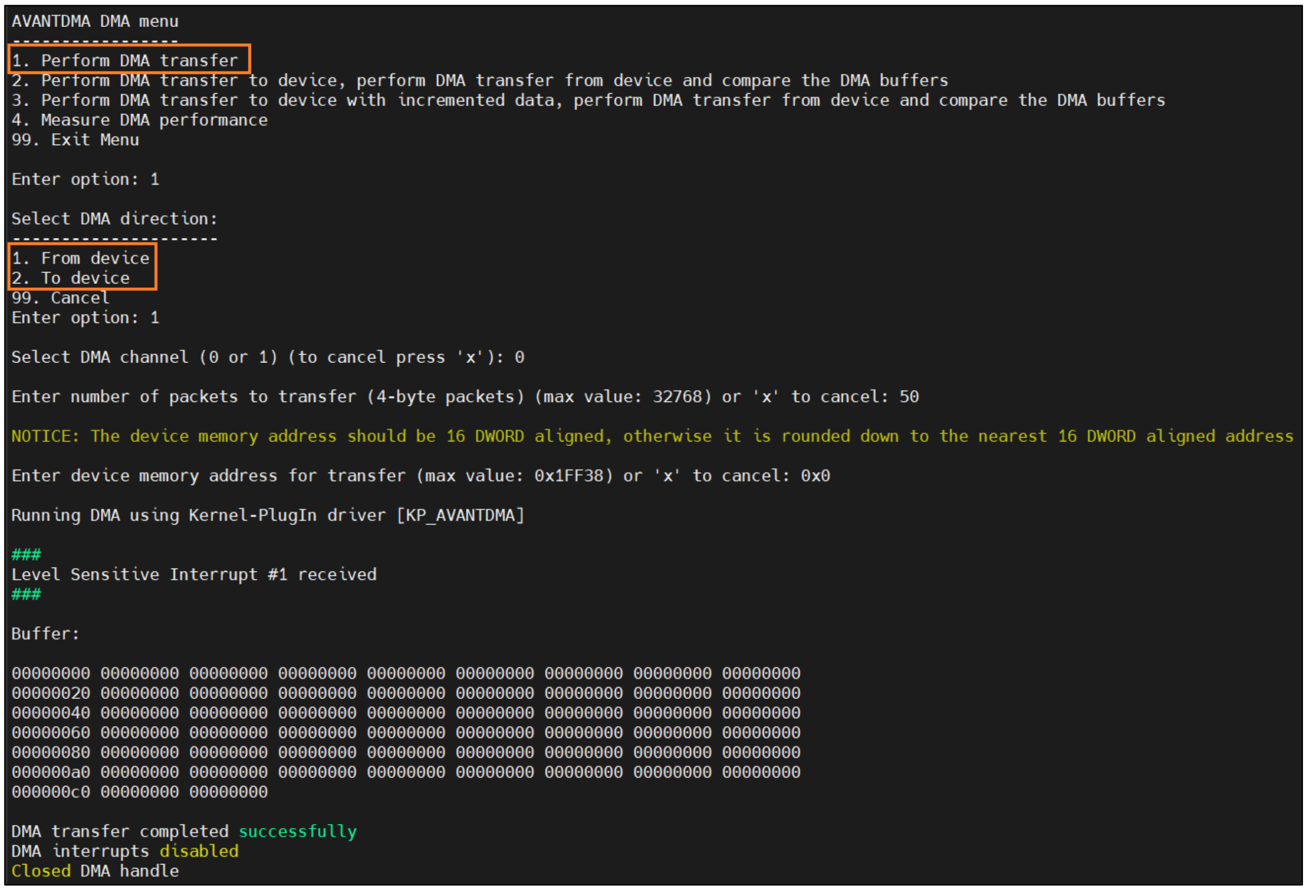
```
AVANTDMA main menu
------------------
1. Scan PCI bus
2. Find and open a PCI device
3. Read/write memory and I/O addresses on the device
4. Read/write the PCI configuration space
5. Direct Memory Access (DMA)
6. Register/unregister plug-and-play and power management events
99. Exit Menu

Enter option: 5

AVANTDMA DMA menu
-----------------
1. Perform DMA transfer
2. Perform DMA transfer to device, perform DMA transfer from device and compare the DMA buffers
3. Perform DMA transfer to device with incremented data, perform DMA transfer from device and compare the DMA buffers
4. Measure DMA performance
99. Exit Menu

Enter option:
```

Figure 3.1. AVANTDMA DMA Menu

3.1.1. DMA Transfer

When selecting **Perform DMA transfer**, you will be prompted to select the DMA direction.

DMA direction **From device** indicates from the FPGA to the host device, while **To device** indicates from the host device to the FPGA.



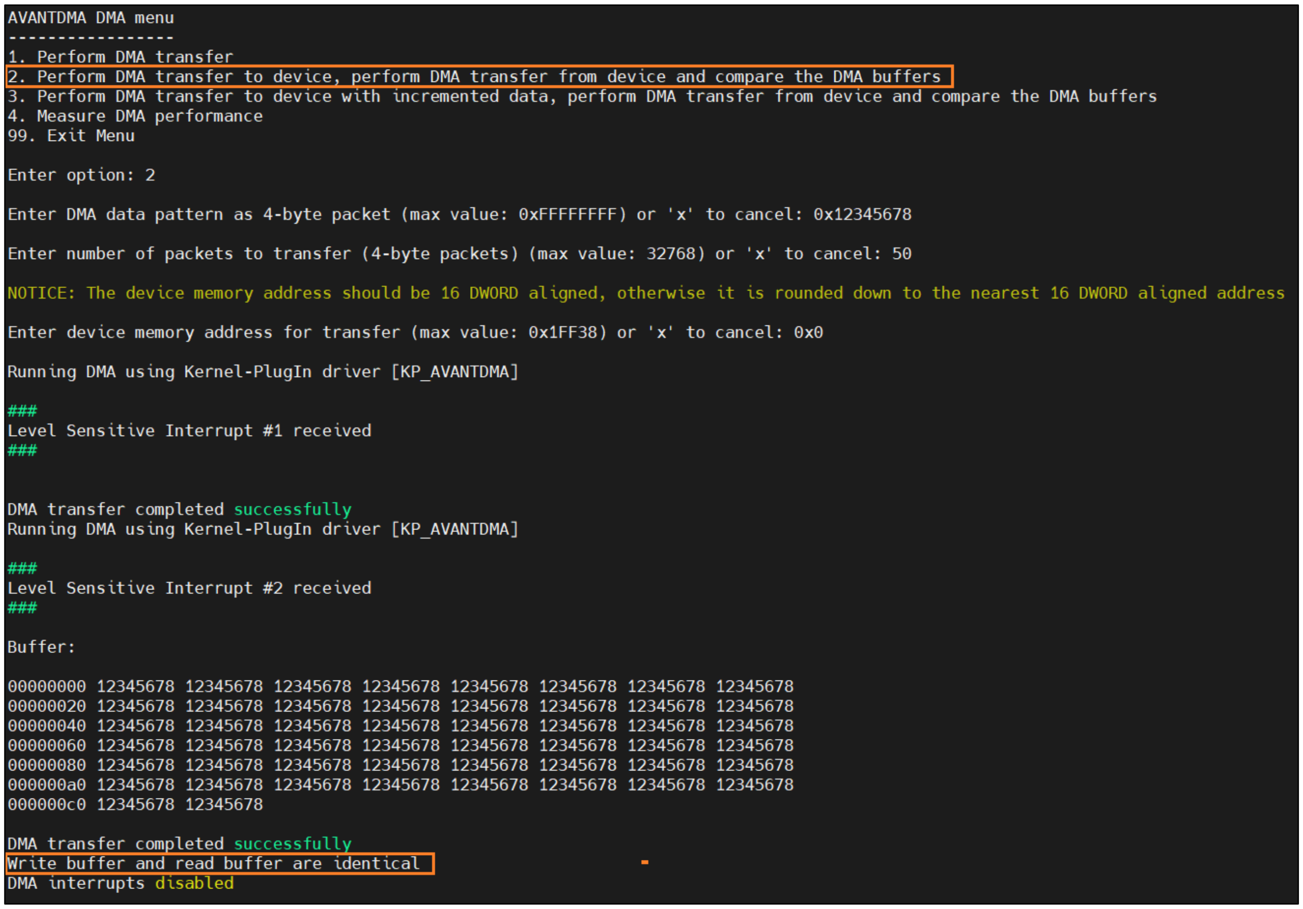
```
AVANTDMA DMA menu
-----------------
1. Perform DMA transfer
2. Perform DMA transfer to device, perform DMA transfer from device and compare the DMA buffers
3. Perform DMA transfer to device with incremented data, perform DMA transfer from device and compare the DMA buffers
4. Measure DMA performance
99. Exit Menu

Enter option: 1

Select DMA direction:
---------------------
1. From device
2. To device
99. Cancel
Enter option: 1

Select DMA channel (0 or 1) (to cancel press 'x'): 0

Enter number of packets to transfer (4-byte packets) (max value: 32768) or 'x' to cancel: 50

NOTICE: The device memory address should be 16 DWORD aligned, otherwise it is rounded down to the nearest 16 DWORD aligned address

Enter device memory address for transfer (max value: 0x1FF38) or 'x' to cancel: 0x0

Running DMA using Kernel-PlugIn driver [KP_AVANTDMA]

###
Level Sensitive Interrupt #1 received
###

Buffer:

00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000020 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000040 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000060 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000080 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
000000a0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
000000c0 00000000 00000000

DMA transfer completed successfully
DMA interrupts disabled
Closed DMA handle
```

Figure 3.2. DMA Transfer Direction

The example design is built with an FPGA internal RAM of 128 kilobytes (131072 bytes). Therefore, when prompted for the device memory address, enter a range from 0 to 0x1FFFF (131071 in decimal). However, consider the size of data to transfer and ensure that the device memory address + data size does not exceed 0x1FFFF.

3.1.2. DMA Transfer and Compare Data

When selecting **Perform DMA transfer to device, perform DMA transfer from device and compare the DMA buffers**, the driver application duplicates the user-entered data pattern to fill the write buffer, then initiates the DMA transfer from the host to device. This will be followed by a DMA transfer from the device to host, after which the values in the read buffer is compared to the write buffer. The status of the DMA buffer compare is reported on the console.

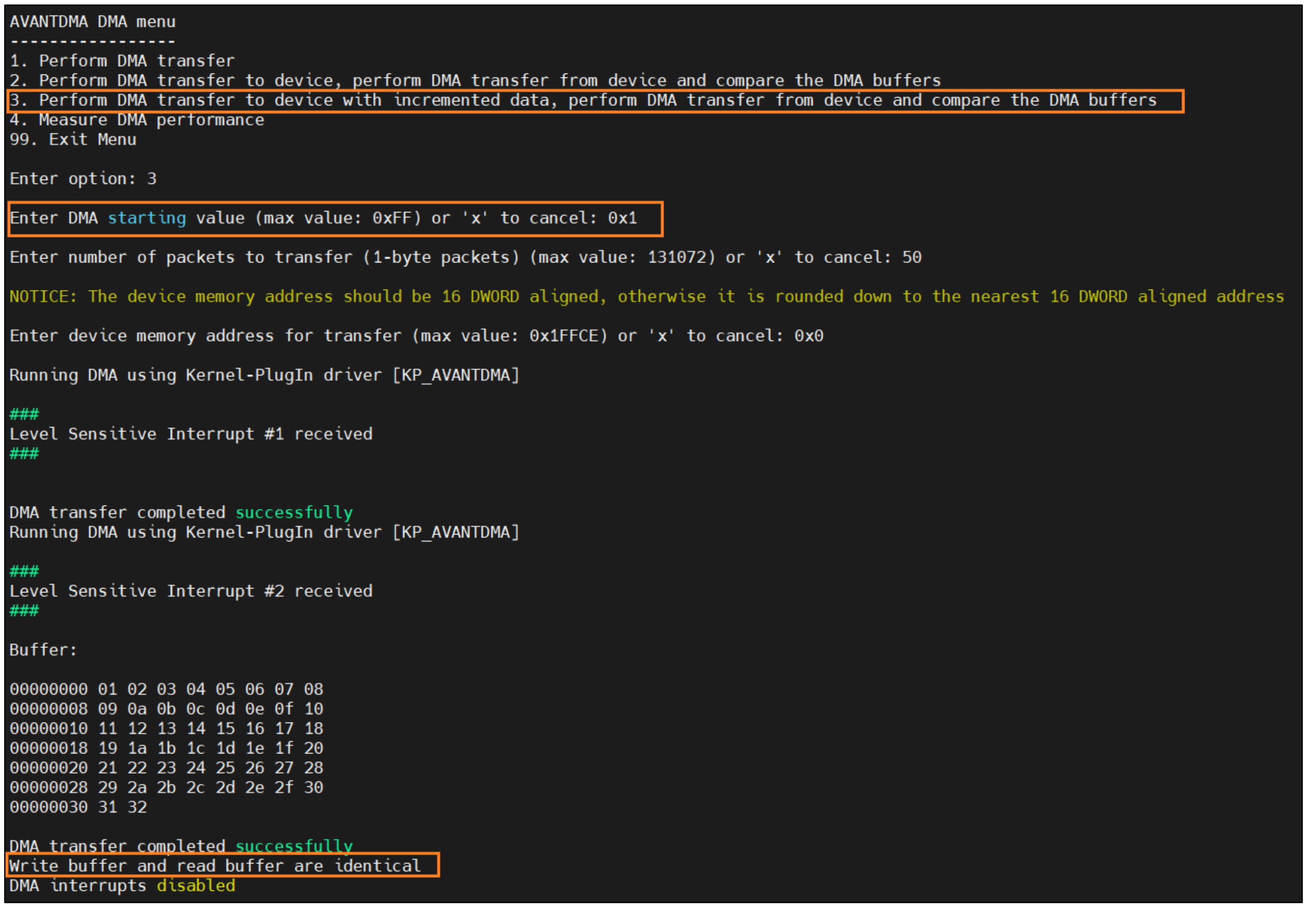


```
AVANTDMA DMA menu
-----------------
1. Perform DMA transfer
2. Perform DMA transfer to device, perform DMA transfer from device and compare the DMA buffers
3. Perform DMA transfer to device with incremented data, perform DMA transfer from device and compare the DMA buffers
4. Measure DMA performance
99. Exit Menu

Enter option: 2

Enter DMA data pattern as 4-byte packet (max value: 0xFFFFFFFF) or 'x' to cancel: 0x12345678

Enter number of packets to transfer (4-byte packets) (max value: 32768) or 'x' to cancel: 50

NOTICE: The device memory address should be 16 DWORD aligned, otherwise it is rounded down to the nearest 16 DWORD aligned address

Enter device memory address for transfer (max value: 0x1FF38) or 'x' to cancel: 0x0

Running DMA using Kernel-PlugIn driver [KP_AVANTDMA]

###
Level Sensitive Interrupt #1 received
###


DMA transfer completed successfully
Running DMA using Kernel-PlugIn driver [KP_AVANTDMA]

###
Level Sensitive Interrupt #2 received
###

Buffer:

00000000 12345678 12345678 12345678 12345678 12345678 12345678 12345678 12345678
00000020 12345678 12345678 12345678 12345678 12345678 12345678 12345678 12345678
00000040 12345678 12345678 12345678 12345678 12345678 12345678 12345678 12345678
00000060 12345678 12345678 12345678 12345678 12345678 12345678 12345678 12345678
00000080 12345678 12345678 12345678 12345678 12345678 12345678 12345678 12345678
000000a0 12345678 12345678 12345678 12345678 12345678 12345678 12345678 12345678
000000c0 12345678 12345678

DMA transfer completed successfully
Write buffer and read buffer are identical
DMA interrupts disabled
```

Figure 3.3. DMA Data Compare

3.1.3. DMA Transfer With Incremented Data and Compare Data

When selecting **Perform DMA transfer to device with incremented data, perform DMA transfer from device and compare the DMA buffers**, the driver application fills the write buffer with incremented data, starting from the user-entered *starting value*, then initiates the DMA transfer from the host to device. This will be followed by a DMA transfer from the device to host, after which the values in the read buffer is compared to the write buffer. The status of the DMA buffer compare is reported on the console.



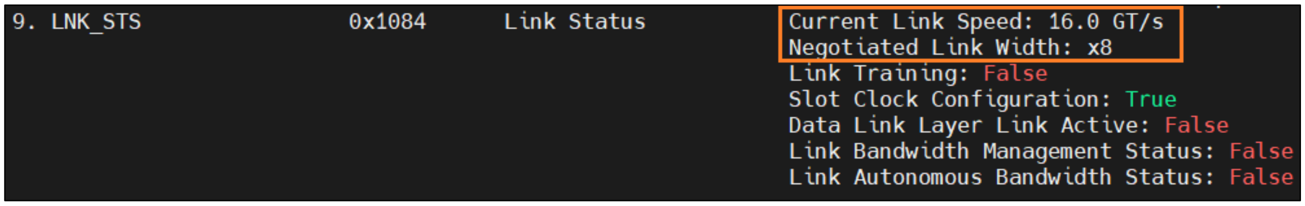
```
AVANTDMA DMA menu
-----------------
1. Perform DMA transfer
2. Perform DMA transfer to device, perform DMA transfer from device and compare the DMA buffers
3. Perform DMA transfer to device with incremented data, perform DMA transfer from device and compare the DMA buffers
4. Measure DMA performance
99. Exit Menu

Enter option: 3

Enter DMA starting value (max value: 0xFF) or 'x' to cancel: 0x1

Enter number of packets to transfer (1-byte packets) (max value: 131072) or 'x' to cancel: 50

NOTICE: The device memory address should be 16 DWORD aligned, otherwise it is rounded down to the nearest 16 DWORD aligned address

Enter device memory address for transfer (max value: 0x1FFCE) or 'x' to cancel: 0x0

Running DMA using Kernel-PlugIn driver [KP_AVANTDMA]

###
Level Sensitive Interrupt #1 received
###


DMA transfer completed successfully
Running DMA using Kernel-PlugIn driver [KP_AVANTDMA]

###
Level Sensitive Interrupt #2 received
###

Buffer:

00000000 01 02 03 04 05 06 07 08
00000008 09 0a 0b 0c 0d 0e 0f 10
00000010 11 12 13 14 15 16 17 18
00000018 19 1a 1b 1c 1d 1e 1f 20
00000020 21 22 23 24 25 26 27 28
00000028 29 2a 2b 2c 2d 2e 2f 30
00000030 31 32

DMA transfer completed successfully
Write buffer and read buffer are identical
DMA interrupts disabled
```

Figure 3.4 DMA Data Compare With Incremented Data

3.1.4. DMA Performance Measure

When selecting **Measure DMA performance**, ensure the PCIe link status is showing *Link Speed* of 16GT/s and *Link Width* of ×8 to get the maximum throughput. You can check this by using this application driver to read the configuration space.

Select **Read/Write PCI configuration space**, followed by **Read all configuration registers defined**. Then, look for register *LINK_STS*.



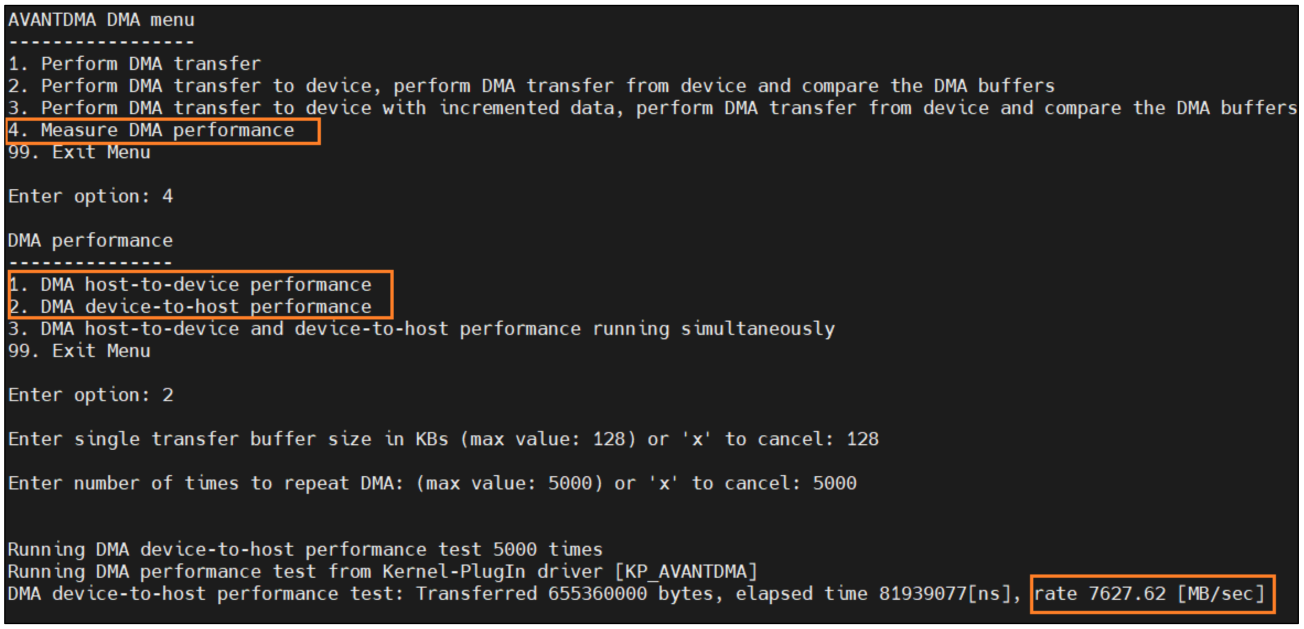
```
9. LNK_STS                0x1084      Link Status         Current Link Speed: 16.0 GT/s
                                                          Negotiated Link Width: x8
                                                          Link Training: False
                                                          Slot Clock Configuration: True
                                                          Data Link Layer Link Active: False
                                                          Link Bandwidth Management Status: False
                                                          Link Autonomous Bandwidth Status: False
```

Figure 3.5. PCIe Link Status Register

If the link is not 16.0 GT/s, verify that the FPGA card is connected to a suitable slot that supports PCIe Gen4 and that the slot is properly configured to support Gen4. You may need to configure the PCIe speed in BIOS.

DMA performance measurement is available for single-direction DMA transfer (host to device or device to host) and bi-directional simultaneous DMA transfer.

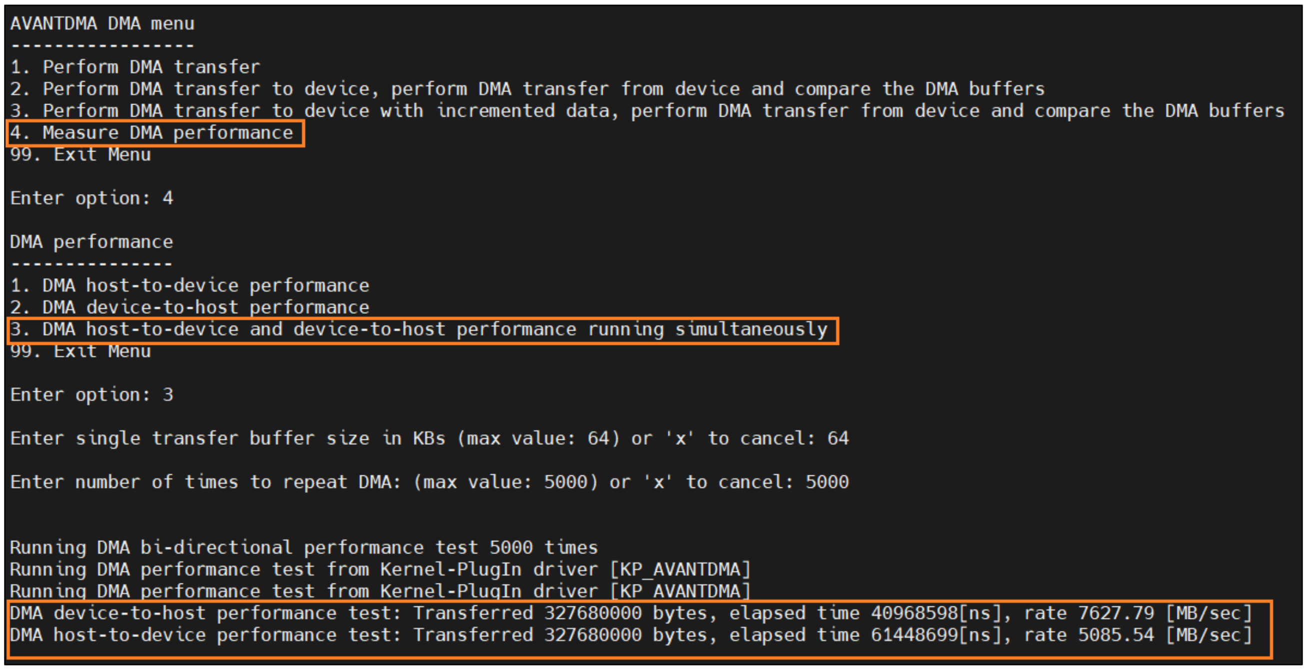
For the single-direction DMA transfers, the maximum buffer size available is 128 kB.



```
AVANTDMA DMA menu
-----------------
1. Perform DMA transfer
2. Perform DMA transfer to device, perform DMA transfer from device and compare the DMA buffers
3. Perform DMA transfer to device with incremented data, perform DMA transfer from device and compare the DMA buffers
4. Measure DMA performance
99. Exit Menu

Enter option: 4

DMA performance
---------------
1. DMA host-to-device performance
2. DMA device-to-host performance
3. DMA host-to-device and device-to-host performance running simultaneously
99. Exit Menu

Enter option: 2

Enter single transfer buffer size in KBs (max value: 128) or 'x' to cancel: 128

Enter number of times to repeat DMA: (max value: 5000) or 'x' to cancel: 5000


Running DMA device-to-host performance test 5000 times
Running DMA performance test from Kernel-PlugIn driver [KP_AVANTDMA]
DMA device-to-host performance test: Transferred 655360000 bytes, elapsed time 81939077[ns], rate 7627.62 [MB/sec]
```

Figure 3.6. Measure DMA Performance for Single Direction Transfer

For bi-directional simultaneous DMA transfer, the maximum buffer size available is 128 ÷ 2 = 64 kB. The internal FPGA RAM is divided into two halves: one half for host-to-device DMA transfers and the other half for device-to-host DMA transfers.

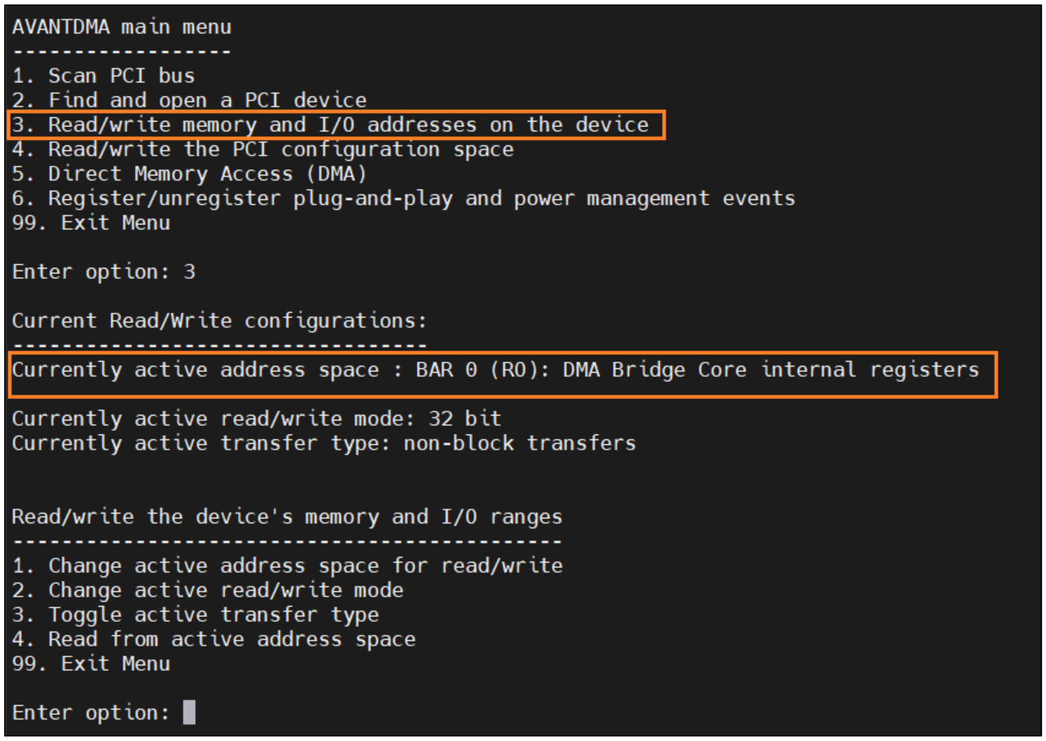


```
AVANTDMA DMA menu
-----------------
1. Perform DMA transfer
2. Perform DMA transfer to device, perform DMA transfer from device and compare the DMA buffers
3. Perform DMA transfer to device with incremented data, perform DMA transfer from device and compare the DMA buffers
4. Measure DMA performance
99. Exit Menu

Enter option: 4

DMA performance
---------------
1. DMA host-to-device performance
2. DMA device-to-host performance
3. DMA host-to-device and device-to-host performance running simultaneously
99. Exit Menu

Enter option: 3

Enter single transfer buffer size in KBs (max value: 64) or 'x' to cancel: 64

Enter number of times to repeat DMA: (max value: 5000) or 'x' to cancel: 5000


Running DMA bi-directional performance test 5000 times
Running DMA performance test from Kernel-PlugIn driver [KP_AVANTDMA]
Running DMA performance test from Kernel-PlugIn driver [KP_AVANTDMA]
DMA device-to-host performance test: Transferred 327680000 bytes, elapsed time 40968598[ns], rate 7627.79 [MB/sec]
DMA host-to-device performance test: Transferred 327680000 bytes, elapsed time 61448699[ns], rate 5085.54 [MB/sec]
```

Figure 3.7. Measure DMA Performance for Bi-Directional Simultaneous Transfer

3.2. Read/Write Memory and I/O Addresses

When selecting **Read/write memory and I/O addresses on the device**, the default active address space is BAR 0. In the DMA example design, BAR 0 is mapped to the DMA internal registers for DMA configuration and status. Therefore BAR 0 memory is read-only (RO) to prevent any interference with the internal registers.

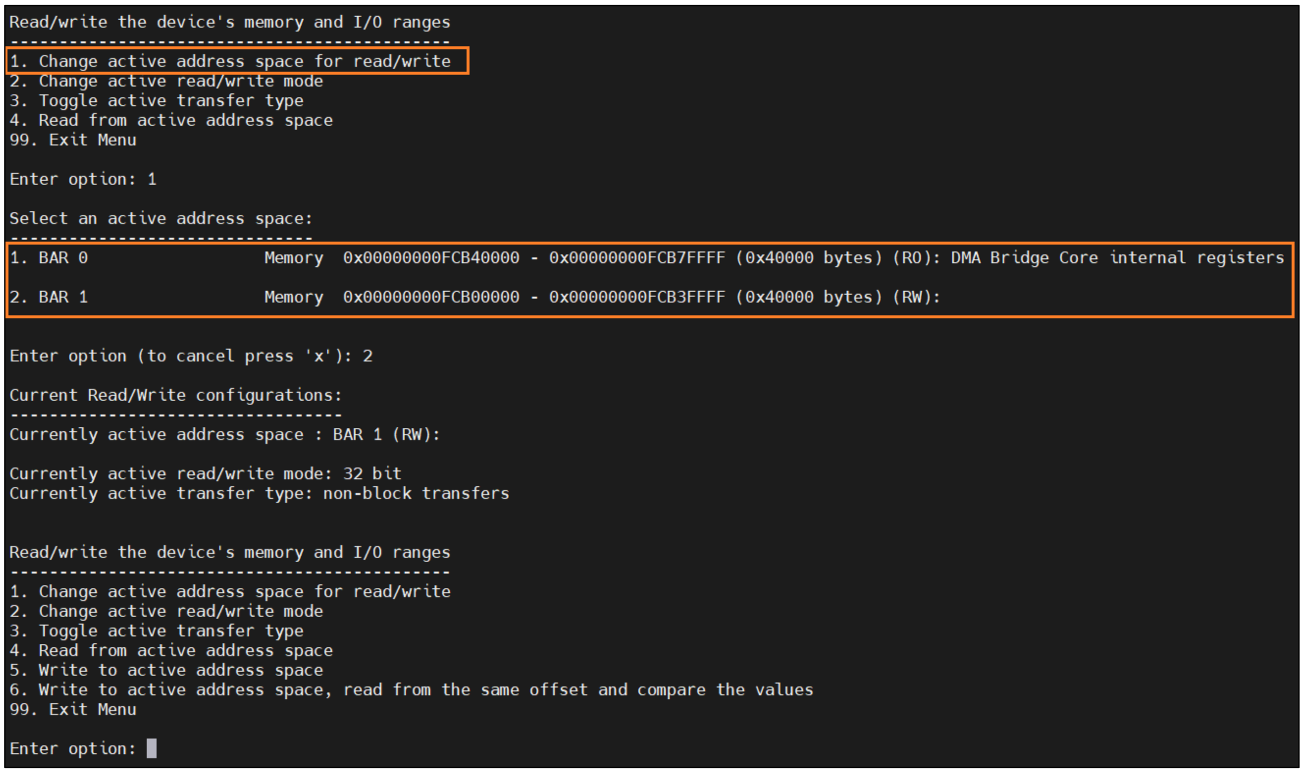


```
AVANTDMA main menu
------------------
1. Scan PCI bus
2. Find and open a PCI device
3. Read/write memory and I/O addresses on the device
4. Read/write the PCI configuration space
5. Direct Memory Access (DMA)
6. Register/unregister plug-and-play and power management events
99. Exit Menu

Enter option: 3

Current Read/Write configurations:
----------------------------------
Currently active address space : BAR 0 (RO): DMA Bridge Core internal registers

Currently active read/write mode: 32 bit
Currently active transfer type: non-block transfers


Read/write the device's memory and I/O ranges
---------------------------------------------
1. Change active address space for read/write
2. Change active read/write mode
3. Toggle active transfer type
4. Read from active address space
99. Exit Menu

Enter option:
```

Figure 3.8 Read/Write Memory and I/O Addresses

The DMA example design has 2 BARs enabled: BAR 0 for DMA registers and BAR 1 for memory or I/O read/write operations. BAR 0 memory is read-only (RO), while BAR 1 memory is read-write (RW).

To change the active address space to BAR1, select **Change active address space for read/write.** When BAR 1 is selected, both read and write operations can be performed on the memory.

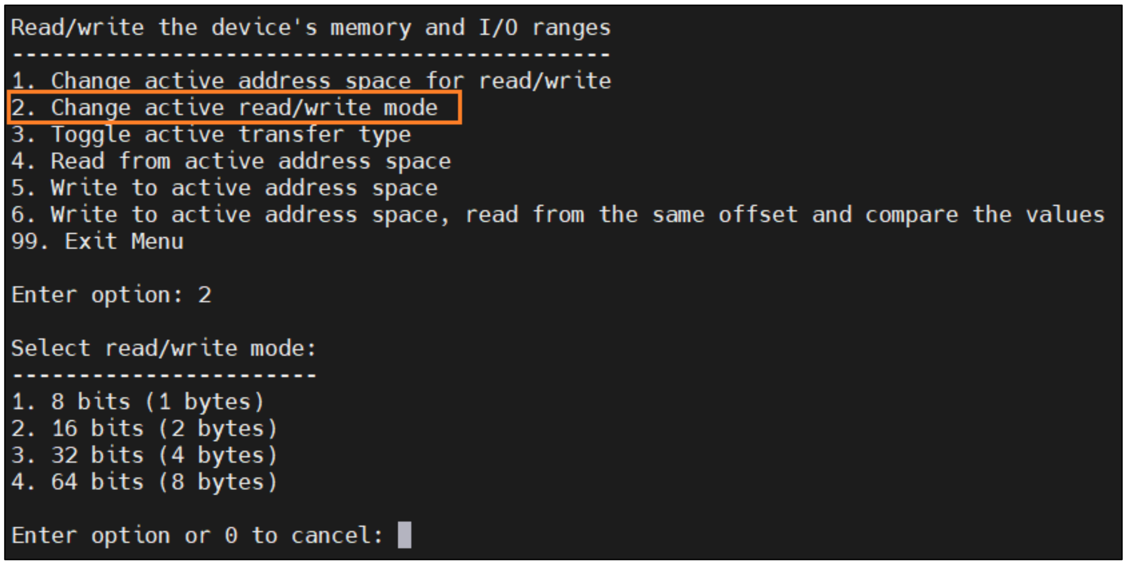


```
Read/write the device's memory and I/O ranges
---------------------------------------------
1. Change active address space for read/write
2. Change active read/write mode
3. Toggle active transfer type
4. Read from active address space
99. Exit Menu

Enter option: 1

Select an active address space:
------------------------------
1. BAR 0                    Memory  0x00000000FCB40000 - 0x00000000FCB7FFFF (0x40000 bytes) (RO): DMA Bridge Core internal registers

2. BAR 1                    Memory  0x00000000FCB00000 - 0x00000000FCB3FFFF (0x40000 bytes) (RW):


Enter option (to cancel press 'x'): 2

Current Read/Write configurations:
----------------------------------
Currently active address space : BAR 1 (RW):

Currently active read/write mode: 32 bit
Currently active transfer type: non-block transfers


Read/write the device's memory and I/O ranges
---------------------------------------------
1. Change active address space for read/write
2. Change active read/write mode
3. Toggle active transfer type
4. Read from active address space
5. Write to active address space
6. Write to active address space, read from the same offset and compare the values
99. Exit Menu

Enter option: 
```

Figure 3.9 Change Active Address Space for Read/Write

3.2.1. Read/Write Mode

The application supports read/write mode of 8 bits, 16 bits, 32 bits, and 64 bits. You can change the read/write mode by selecting **Change active read/write mode.**

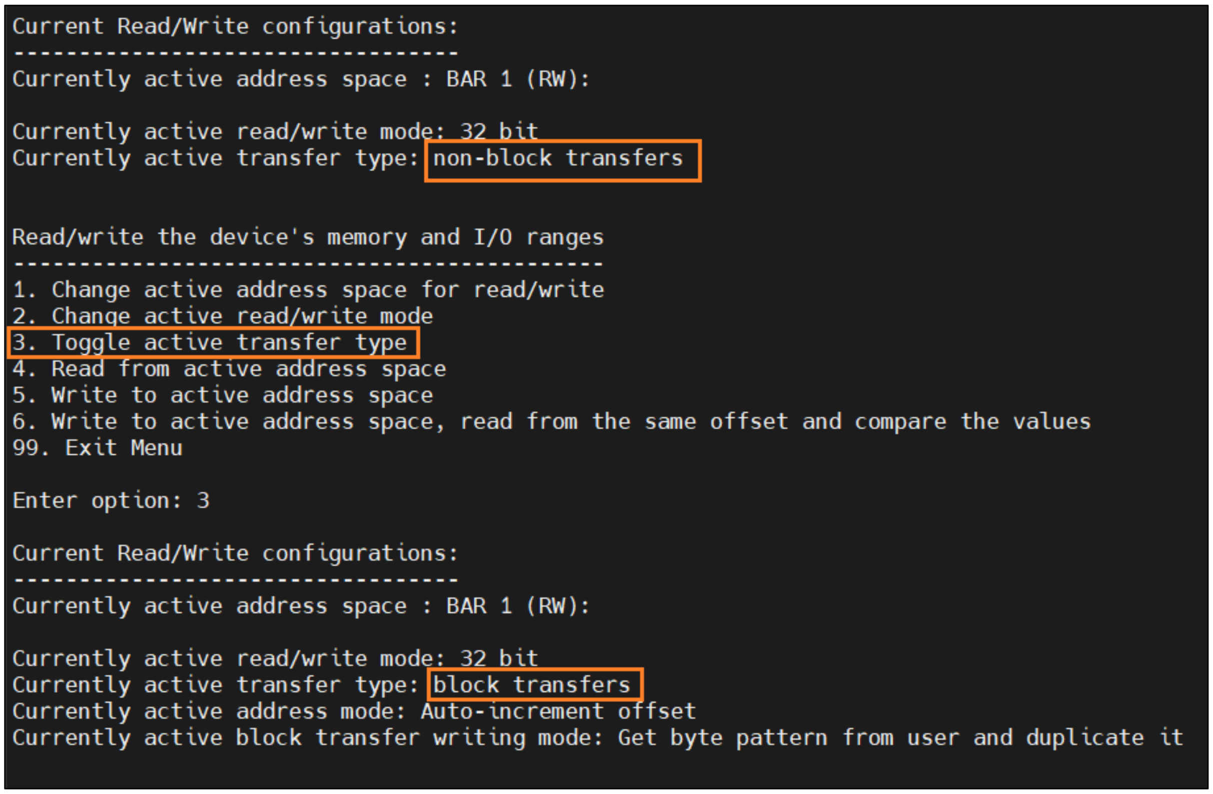


```
Read/write the device's memory and I/O ranges
---------------------------------------------
1. Change active address space for read/write
2. Change active read/write mode
3. Toggle active transfer type
4. Read from active address space
5. Write to active address space
6. Write to active address space, read from the same offset and compare the values
99. Exit Menu

Enter option: 2

Select read/write mode:
----------------------
1. 8 bits (1 bytes)
2. 16 bits (2 bytes)
3. 32 bits (4 bytes)
4. 64 bits (8 bytes)

Enter option or 0 to cancel: 
```

Figure 3.10 Change Active Read/Write Mode

3.2.2. Transfer Type: Non-block or Block Transfers

You can toggle between non-block transfers and block transfers by selecting **Toggle active transfer type**.



```
Current Read/Write configurations:
----------------------------------
Currently active address space : BAR 1 (RW):

Currently active read/write mode: 32 bit
Currently active transfer type: non-block transfers


Read/write the device's memory and I/O ranges
---------------------------------------------
1. Change active address space for read/write
2. Change active read/write mode
3. Toggle active transfer type
4. Read from active address space
5. Write to active address space
6. Write to active address space, read from the same offset and compare the values
99. Exit Menu

Enter option: 3

Current Read/Write configurations:
----------------------------------
Currently active address space : BAR 1 (RW):

Currently active read/write mode: 32 bit
Currently active transfer type: block transfers
Currently active address mode: Auto-increment offset
Currently active block transfer writing mode: Get byte pattern from user and duplicate it
```

Figure 3.11 Non-Block or Block Transfers

3.2.2.1. Non-block Transfer

Non-block transfer refers to read/write to a single unit of the memory depending on the read/write mode:

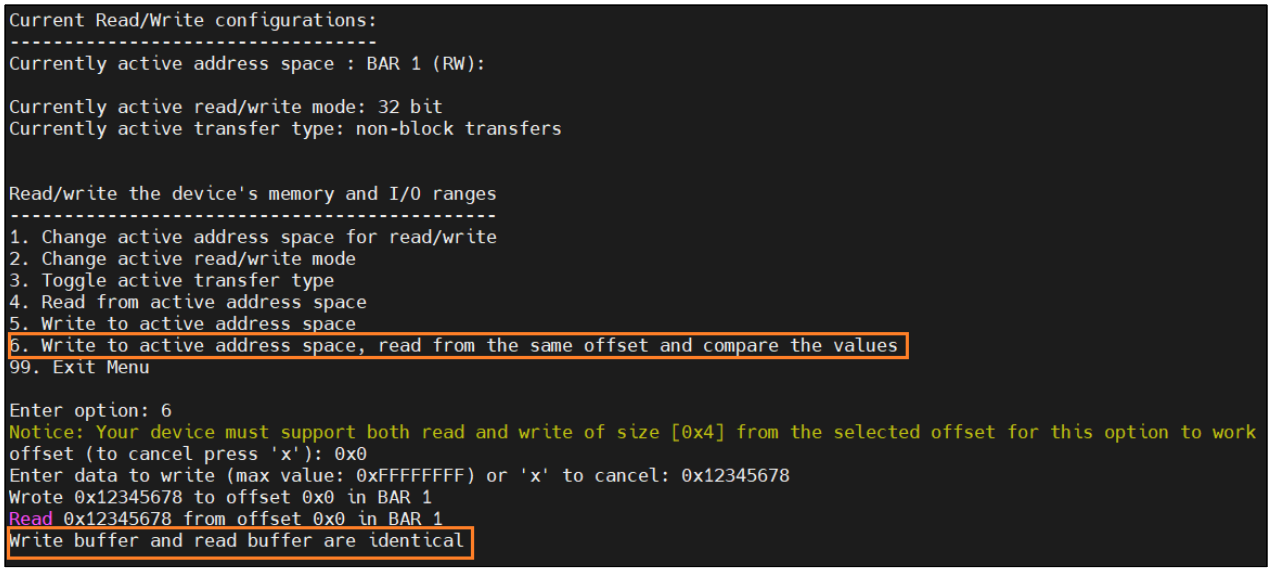
- If 8-bit mode is selected, non-block transfer reads/writes 1 byte of data.
- If 16-bit mode is selected, non-block transfer reads/writes 2 bytes of data.
- If 32-bit mode is selected, non-block transfer reads/writes 4 bytes of data.
- If 64-bit mode is selected, non-block transfer reads/writes 8 bytes of data.

3.2.2.2. Block Transfer

Block transfer allows you to read/write data to/from a block of memory.

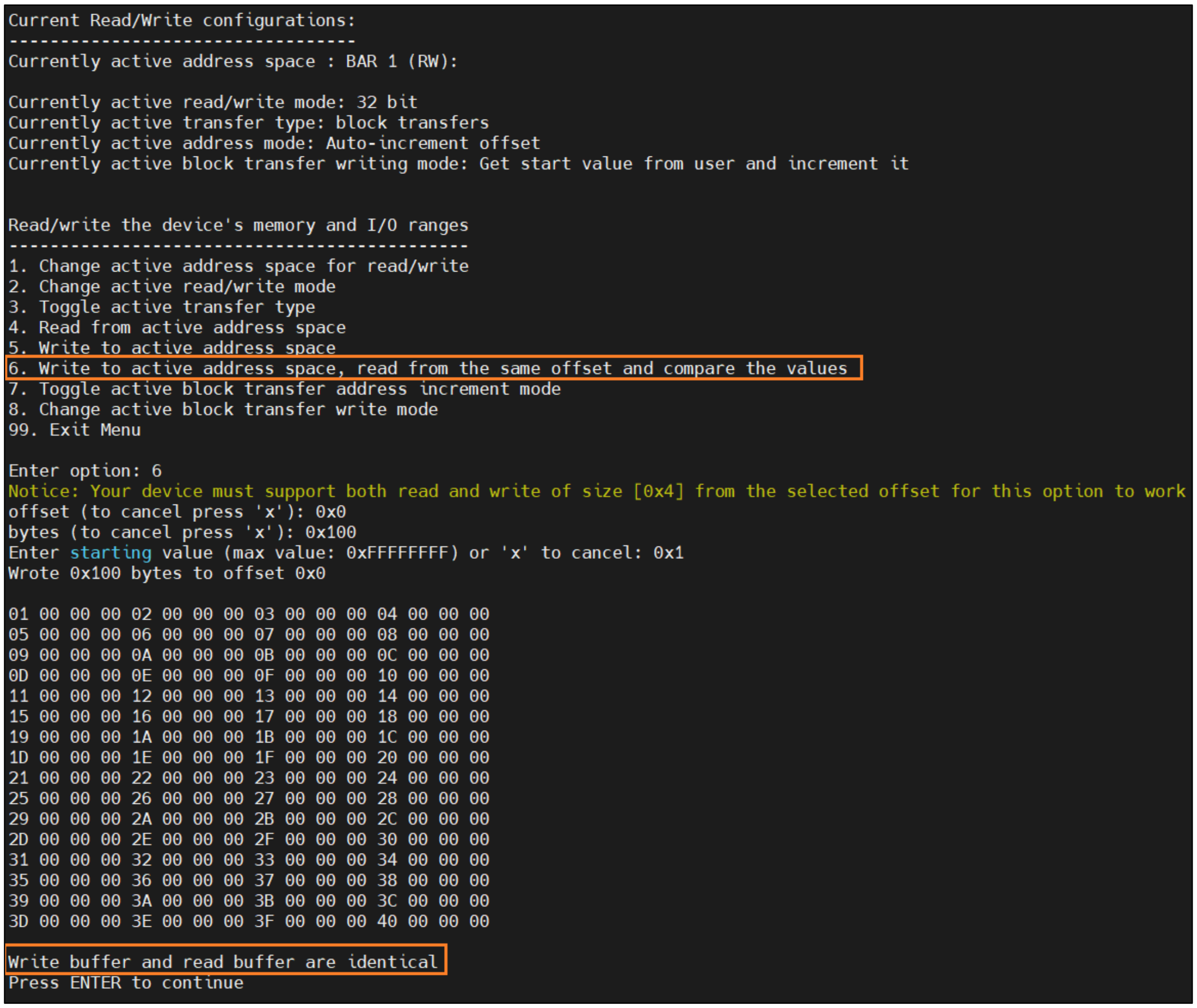
3.2.3. Write, Read, and Compare Values

For both block and non-block transfers, there is a function that writes your entered data, then reads back, and compares the values. Any data mismatch are reported on the console.



```
Current Read/Write configurations:
----------------------------------
Currently active address space : BAR 1 (RW):

Currently active read/write mode: 32 bit
Currently active transfer type: non-block transfers


Read/write the device's memory and I/O ranges
---------------------------------------------
1. Change active address space for read/write
2. Change active read/write mode
3. Toggle active transfer type
4. Read from active address space
5. Write to active address space
6. Write to active address space, read from the same offset and compare the values
99. Exit Menu

Enter option: 6
Notice: Your device must support both read and write of size [0x4] from the selected offset for this option to work
offset (to cancel press 'x'): 0x0
Enter data to write (max value: 0xFFFFFFFF) or 'x' to cancel: 0x12345678
Wrote 0x12345678 to offset 0x0 in BAR 1
Read 0x12345678 from offset 0x0 in BAR 1
Write buffer and read buffer are identical
```

Figure 3.12 Non-Block Transfer – Write, Read, and Compare Values



```
Current Read/Write configurations:
----------------------------------
Currently active address space : BAR 1 (RW):

Currently active read/write mode: 32 bit
Currently active transfer type: block transfers
Currently active address mode: Auto-increment offset
Currently active block transfer writing mode: Get start value from user and increment it


Read/write the device's memory and I/O ranges
---------------------------------------------
1. Change active address space for read/write
2. Change active read/write mode
3. Toggle active transfer type
4. Read from active address space
5. Write to active address space
6. Write to active address space, read from the same offset and compare the values
7. Toggle active block transfer address increment mode
8. Change active block transfer write mode
99. Exit Menu

Enter option: 6
Notice: Your device must support both read and write of size [0x4] from the selected offset for this option to work
offset (to cancel press 'x'): 0x0
bytes (to cancel press 'x'): 0x100
Enter starting value (max value: 0xFFFFFFFF) or 'x' to cancel: 0x1
Wrote 0x100 bytes to offset 0x0

01 00 00 00 02 00 00 00 03 00 00 00 04 00 00 00
05 00 00 00 06 00 00 00 07 00 00 00 08 00 00 00
09 00 00 00 0A 00 00 00 0B 00 00 00 0C 00 00 00
0D 00 00 00 0E 00 00 00 0F 00 00 00 10 00 00 00
11 00 00 00 12 00 00 00 13 00 00 00 14 00 00 00
15 00 00 00 16 00 00 00 17 00 00 00 18 00 00 00
19 00 00 00 1A 00 00 00 1B 00 00 00 1C 00 00 00
1D 00 00 00 1E 00 00 00 1F 00 00 00 20 00 00 00
21 00 00 00 22 00 00 00 23 00 00 00 24 00 00 00
25 00 00 00 26 00 00 00 27 00 00 00 28 00 00 00
29 00 00 00 2A 00 00 00 2B 00 00 00 2C 00 00 00
2D 00 00 00 2E 00 00 00 2F 00 00 00 30 00 00 00
31 00 00 00 32 00 00 00 33 00 00 00 34 00 00 00
35 00 00 00 36 00 00 00 37 00 00 00 38 00 00 00
39 00 00 00 3A 00 00 00 3B 00 00 00 3C 00 00 00
3D 00 00 00 3E 00 00 00 3F 00 00 00 40 00 00 00

Write buffer and read buffer are identical
Press ENTER to continue
```

Figure 3.13 Block Transfer – Write, Read, and Compare Values

4. APIs

The APIs are defined by Jungo WinDriver.

Table 4.1 shows the list of APIs used by the host PCIe driver to enable SGDMA data transfer and its operation. For more details, refer to the links provided in the table below.

Table 4.1. List of APIs

WinDriver APIs	Description
OsEventCreate	Creates an event object.
OsEventWait	Waits until a specified event object is in the signaled state or the time-out interval elapses.
OsEventSignal	Sets the specified event object to the signaled state.
OsEventClose	Closes a handle to an event object.
WDC_IntEnable	• Enables interrupt handling for the device. • The software provides a user-mode interrupt handler callback function, *AVANTDMA_IntHandler*, which is called after an interrupt is received and processed in the kernel. **Note:** For performance measurement, only the kernel plugin interrupt service routine is called. • The last argument is set to TRUE as the driver uses a Kernel Plugin driver.
WDC_IntDisable	Disables interrupt handling for the device.
WDC_IntIsEnabled	Checks if the interrupts of a device are enabled.
WDC_DMASGBufLock	• Locks a pre-allocated user-mode memory buffer for the DMA which is passed in as the second argument of the API. • Returns the corresponding physical mappings of the locked DMA pages as the fifth argument of the API.
WDC_DMAContigBufLock	• Allocates a contiguous descriptor buffer and locks it in the physical memory. • Returns mapping of the allocated descriptor buffer to the physical address and to the user-mode and kernel virtual address spaces.
WDC_DMABufUnlock	Unlocks and frees the memory allocated for a DMA buffer by a previous call to *WDC_DMASGBufLock* and *WDC_DMAContigBufLock*.
WDC_CallKerPlug	Sends a message from a user-mode application to a Kernel PlugIn driver. In this case, the software sends a message to start the DMA transfer.
kp_interlocked_init	Initializes a Kernel PlugIn interlocked counter.
kp_interlocked_read	Reads to the value of a Kernel PlugIn interlocked counter. In this case, the software driver reads the *intReceived* flag.
kp_interlocked_set	Sets the value of a Kernel PlugIn interlocked counter to the specified value. Here, it is used to set the *intReceived* flag inside the kernel plugin interrupt handler function.
kp_interlocked_uninit	Uninitialized a Kernel PlugIn interlocked counter.
WDC_DMASyncCpu	Synchronizes all CPU caches with the DMA buffer by flushing the data from the CPU caches.

5. Software Flow Diagrams

Some details of the software are omitted from the flowchart to simplify the diagrams and to describe the operation more clearly.

For details on the descriptors source scatter-gather queues, destination scatter-gather queues and status queues format, instructions on programming the DMA, and FPGA registers list related to the DMA, refer to the PCIe X8 IP Core User Guide (FPGA-IPUG-02243).

The software code includes an alternative DMA transfer method that allows reusing DMA queues. This alternative method is not enabled by default and can be enabled by recompiling the code with #define DMA_PERF_USING_TWO_SETS_OF_DESCRIPTORS. With this alternative method, when the queue pointer reaches the top of the queue, it wraps back to the first element at the bottom of the queue. This alternative method uses two sets of queues, and the software configures the DMA to alternate between the two sets of queues on each interrupt upon completion of a DMA transfer.

5.1. AVANTDMA_DmaOpen

This function opens a DMA handle and allocate and initialize the Avant DMA information structure, including the allocation of the scatter-gather DMA buffer.

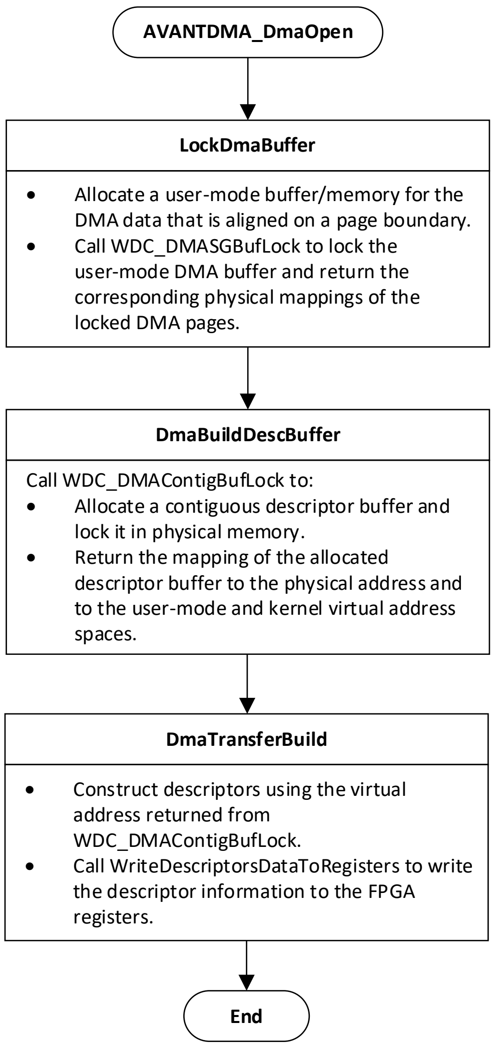

AVANTDMA_DmaOpen
LockDmaBuffer
Allocate a user-mode buffer/memory for the DMA data that is aligned on a page boundary.
Call WDC_DMASGBufLock to lock the user-mode DMA buffer and return the corresponding physical mappings of the locked DMA pages.
DmaBuildDescBuffer
Call WDC_DMAContigBufLock to:
Allocate a contiguous descriptor buffer and lock it in physical memory.
Return the mapping of the allocated descriptor buffer to the physical address and to the user-mode and kernel virtual address spaces.
DmaTransferBuild
Construct descriptors using the virtual address returned from WDC_DMAContigBufLock.
Call WriteDescriptorsDataToRegisters to write the descriptor information to the FPGA registers.
End

Figure 5.1. AVANTDMA_DmaOpen

5.2. DmaPerformanceSingleDir

This function initializes the DMA and starts a thread to initiate the DMA transfer and measure the DMA throughput.

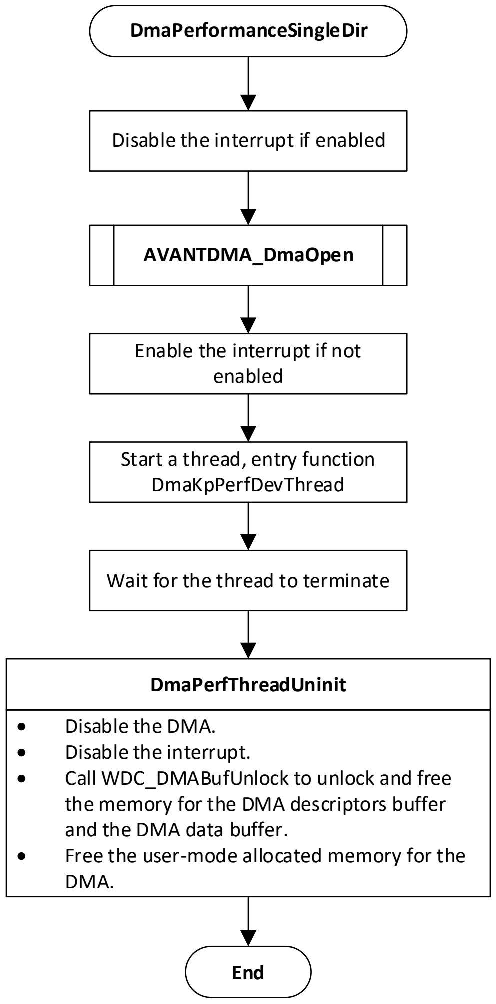

DmaPerformanceSingleDir
Disable the interrupt if enabled
AVANTDMA_DmaOpen
Enable the interrupt if not enabled
Start a thread, entry function DmaKpPerfDevThread
Wait for the thread to terminate
DmaPerfThreadUninit
Disable the DMA.
Disable the interrupt.
Call WDC_DMABufUnlock to unlock and free the memory for the DMA descriptors buffer and the DMA data buffer.
Free the user-mode allocated memory for the DMA.
End

Figure 5.2. DmaPerformanceSingleDir

5.3. DmaKpPerfDevThread

This thread function sends a message to kernel mode to start the DMA transfer, get the start time, poll for DMA completion, and get the end time for DMA throughput measurement.

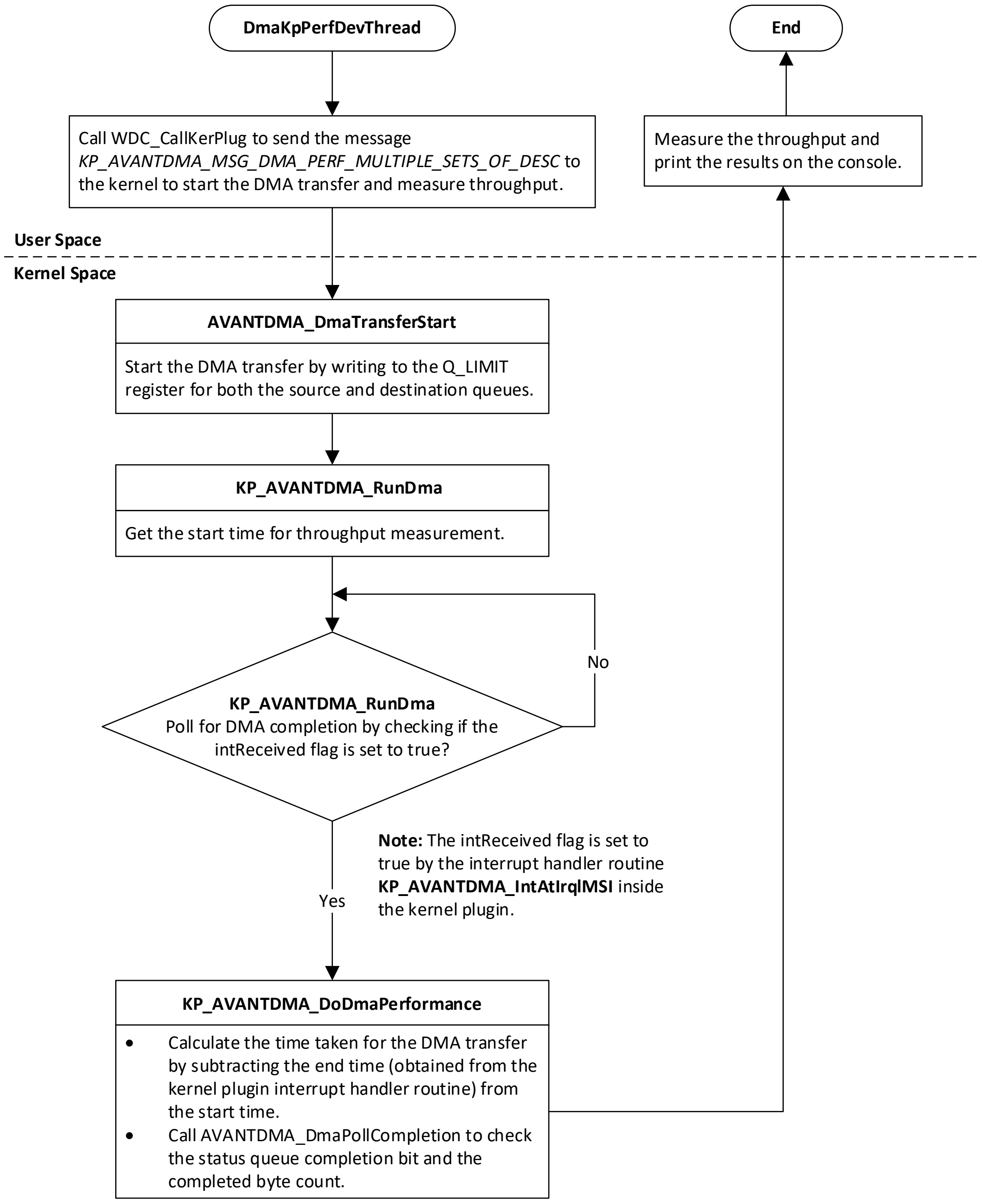

DmaKpPerfDevThread
Call WDC_CallKerPlug to send the message *KP_AVANTDMA_MSG_DMA_PERF_MULTIPLE_SETS_OF_DESC* to the kernel to start the DMA transfer and measure throughput.
User Space
Kernel Space
AVANTDMA_DmaTransferStart
Start the DMA transfer by writing to the Q_LIMIT register for both the source and destination queues.
KP_AVANTDMA_RunDma
Get the start time for throughput measurement.
KP_AVANTDMA_RunDma
Poll for DMA completion by checking if the intReceived flag is set to true?
No
Yes
Note: The intReceived flag is set to true by the interrupt handler routine KP_AVANTDMA_IntAtIrqlMSI inside the kernel plugin.
KP_AVANTDMA_DoDmaPerformance
Calculate the time taken for the DMA transfer by subtracting the end time (obtained from the kernel plugin interrupt handler routine) from the start time.
Call AVANTDMA_DmaPollCompletion to check the status queue completion bit and the completed byte count.
Measure the throughput and print the results on the console.
End

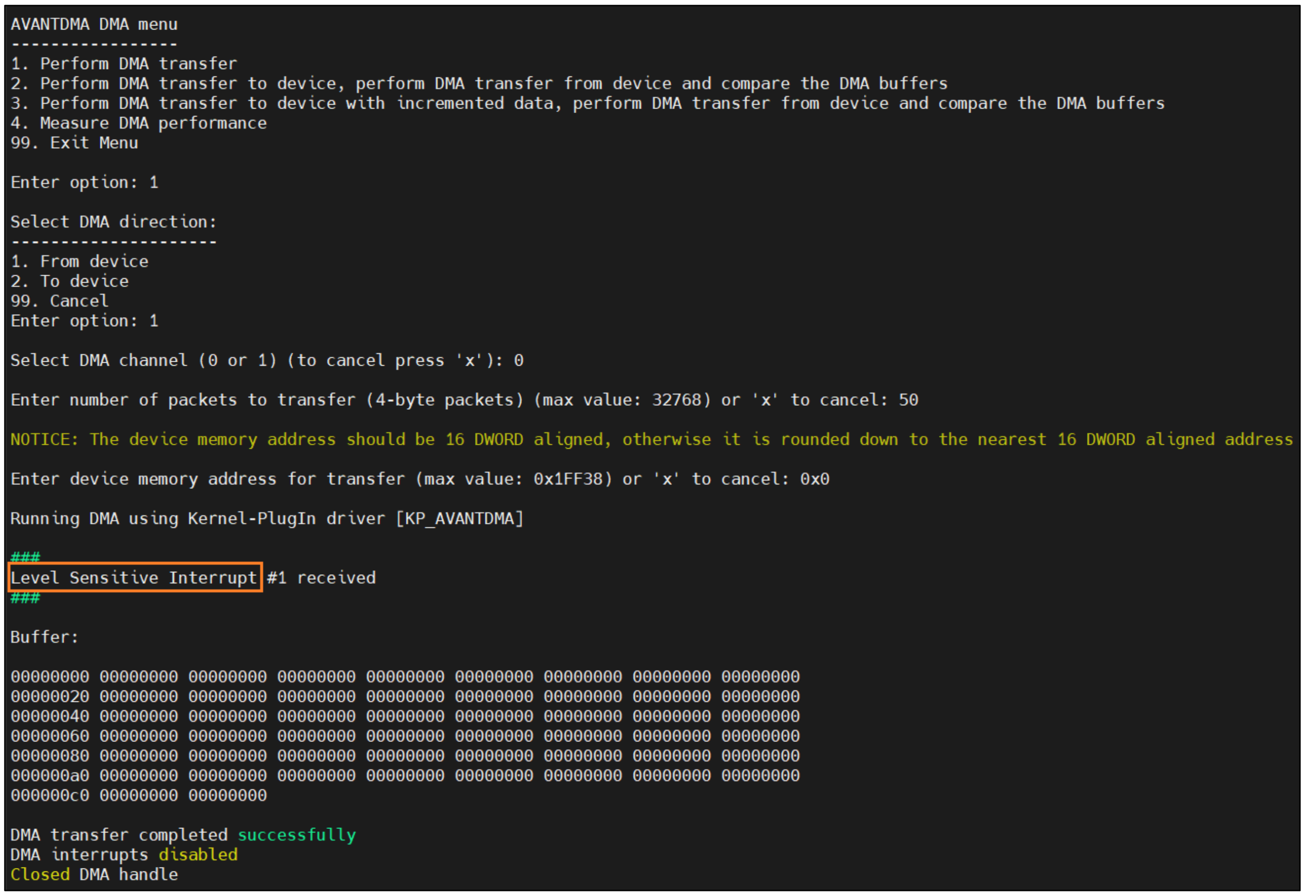
Figure 5.3. DmaKpPerfDevThread

Appendix A. Limitations

There are 2 known limitations with the Avant-G/X PCIe Host DMA driver version 24.02.00:

- The Message-Signaled Interrupt is incorrectly labeled as *Level Sensitive Interrupt* on the console.

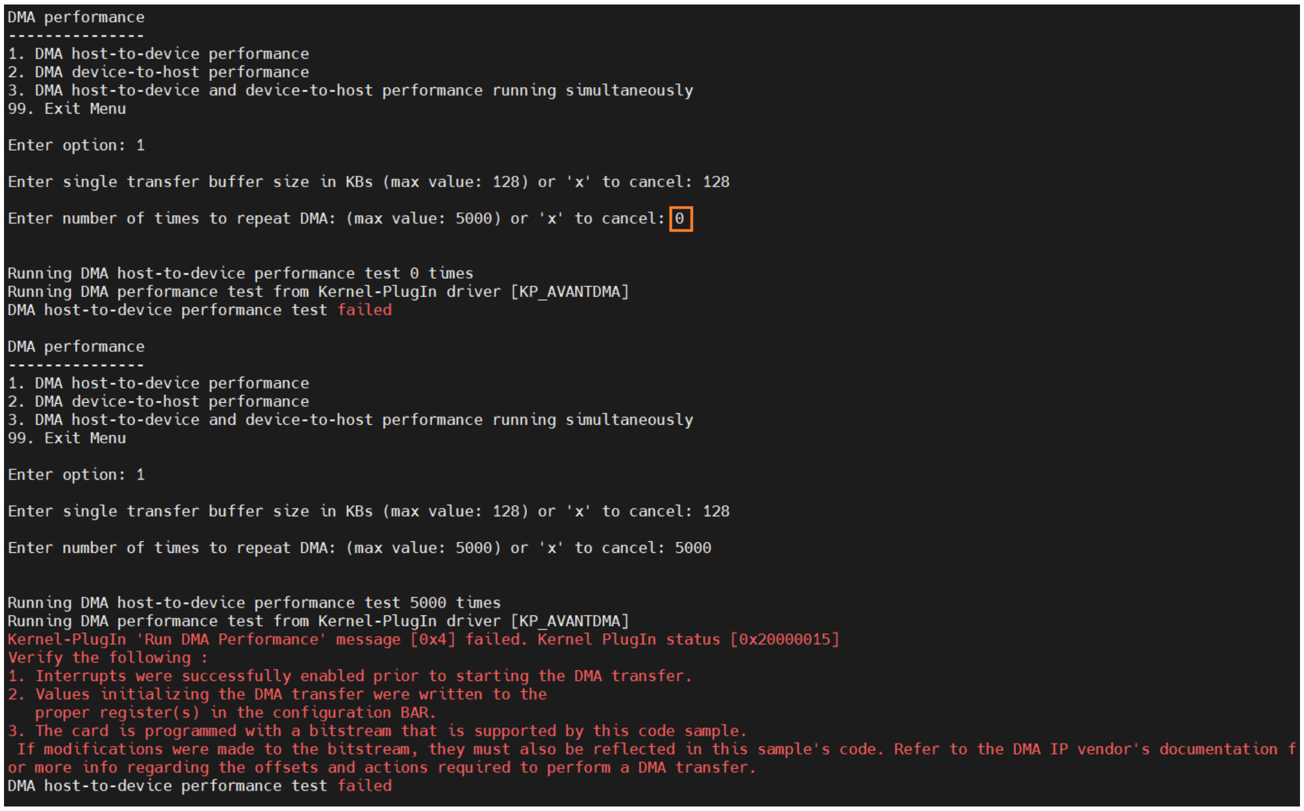
  Note: The DMA operation and the MSI are working as expected and are not affected by this limitation.



```
AVANTDMA DMA menu
-----------------
1. Perform DMA transfer
2. Perform DMA transfer to device, perform DMA transfer from device and compare the DMA buffers
3. Perform DMA transfer to device with incremented data, perform DMA transfer from device and compare the DMA buffers
4. Measure DMA performance
99. Exit Menu

Enter option: 1

Select DMA direction:
---------------------
1. From device
2. To device
99. Cancel
Enter option: 1

Select DMA channel (0 or 1) (to cancel press 'x'): 0

Enter number of packets to transfer (4-byte packets) (max value: 32768) or 'x' to cancel: 50

NOTICE: The device memory address should be 16 DWORD aligned, otherwise it is rounded down to the nearest 16 DWORD aligned address

Enter device memory address for transfer (max value: 0x1FF38) or 'x' to cancel: 0x0

Running DMA using Kernel-PlugIn driver [KP_AVANTDMA]

###
Level Sensitive Interrupt #1 received
###

Buffer:

00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000020 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000040 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000060 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000080 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
000000a0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
000000c0 00000000 00000000

DMA transfer completed successfully
DMA interrupts disabled
Closed DMA handle
```

Figure A.1. Message-Signaled Interrupt Incorrectly Labeled

- For the DMA performance function, when you enter *0* for the *number of times to repeat DMA*, the next DMA performance test fails even if you enter a valid input. However, if you continue on with the subsequent DMA performance test, it restores to normal.



```
DMA performance
---------------
1. DMA host-to-device performance
2. DMA device-to-host performance
3. DMA host-to-device and device-to-host performance running simultaneously
99. Exit Menu

Enter option: 1

Enter single transfer buffer size in KBs (max value: 128) or 'x' to cancel: 128

Enter number of times to repeat DMA: (max value: 5000) or 'x' to cancel: 0


Running DMA host-to-device performance test 0 times
Running DMA performance test from Kernel-PlugIn driver [KP_AVANTDMA]
DMA host-to-device performance test failed

DMA performance
---------------
1. DMA host-to-device performance
2. DMA device-to-host performance
3. DMA host-to-device and device-to-host performance running simultaneously
99. Exit Menu

Enter option: 1

Enter single transfer buffer size in KBs (max value: 128) or 'x' to cancel: 128

Enter number of times to repeat DMA: (max value: 5000) or 'x' to cancel: 5000


Running DMA host-to-device performance test 5000 times
Running DMA performance test from Kernel-PlugIn driver [KP_AVANTDMA]
Kernel-PlugIn 'Run DMA Performance' message [0x4] failed. Kernel PlugIn status [0x20000015]
Verify the following :
1. Interrupts were successfully enabled prior to starting the DMA transfer.
2. Values initializing the DMA transfer were written to the
   proper register(s) in the configuration BAR.
3. The card is programmed with a bitstream that is supported by this code sample.
 If modifications were made to the bitstream, they must also be reflected in this sample's code. Refer to the DMA IP vendor's documentation f
or more info regarding the offsets and actions required to perform a DMA transfer.
DMA host-to-device performance test failed
```

Figure A.2. DMA Performance Test Failure

References

- Jungo WinDriver Official Documentation: Introduction
- PCIe X8 IP Core User Guide (FPGA-IPUG-02243)
- PCIe x8 IP Release Notes (FPGA-RN-02061)
- PCI Express x1/x2/x4 Endpoint IP Core User Guide (FPGA-IPUG-02009)
- Avant-G web page
- Avant-X web page
- Lattice Solutions IP Cores web page
- Lattice Insights web page for Lattice Semiconductor training courses and learning plans

Technical Support Assistance

Submit a technical support case through www.latticesemi.com/techsupport.

For frequently asked questions, refer to the Lattice Answer Database at www.latticesemi.com/Support/AnswerDatabase.

Revision History

Revision 1.0, March 2025

Section	Change Summary
All	Initial release.

www.latticesemi.com