PCIe Host Driver Software for CertusPro-NX AXI MM DMA

Technical Note

FPGA-TN-02386-1.0

December 2024

Disclaimers

Lattice makes no warranty, representation, or guarantee regarding the accuracy of information contained in this document or the suitability of its products for any particular purpose. All information herein is provided AS IS, with all faults, and all associated risk is the responsibility entirely of the Buyer. The information provided herein is for informational purposes only and may contain technical inaccuracies or omissions, and may be otherwise rendered inaccurate for many reasons, and Lattice assumes no obligation to update or otherwise correct or revise this information. Products sold by Lattice have been subject to limited testing and it is the Buyer's responsibility to independently determine the suitability of any products and to test and verify the same. LATTICE PRODUCTS AND SERVICES ARE NOT DESIGNED, MANUFACTURED, OR TESTED FOR USE IN LIFE OR SAFETY CRITICAL SYSTEMS, HAZARDOUS ENVIRONMENTS, OR ANY OTHER ENVIRONMENTS REQUIRING FAIL-SAFE PERFORMANCE, INCLUDING ANY APPLICATION IN WHICH THE FAILURE OF THE PRODUCT OR SERVICE COULD LEAD TO DEATH, PERSONAL INJURY, SEVERE PROPERTY DAMAGE OR ENVIRONMENTAL HARM (COLLECTIVELY, "HIGH-RISK USES"). FURTHER, BUYER MUST TAKE PRUDENT STEPS TO PROTECT AGAINST PRODUCT AND SERVICE FAILURES, INCLUDING PROVIDING APPROPRIATE REDUNDANCIES, FAIL-SAFE FEATURES, AND/OR SHUT-DOWN MECHANISMS. LATTICE EXPRESSLY DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY OF FITNESS OF THE PRODUCTS OR SERVICES FOR HIGH-RISK USES. The information provided in this document is proprietary to Lattice Semiconductor, and Lattice reserves the right to make any changes to the information in this document or to any products at any time without notice.

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

Figures

Tables

Abbreviations in This Document

A list of abbreviations used in this document.

Abbreviation	Definition
API	Application Programming Interface
CPNX	CertusPro-NX
CPU	Central Processing Unit
DMA	Direct Memory Access
OS	Operating System
PCIe	Peripheral Component Interconnect Express
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 X4 IP Core User Guide (FPGA-IPUG-02126) for more details about the IP core.

This guide describes how to use the PCIe host driver software with the CertusPro™-NX AXI-MM 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 CertusPro-NX devices. The technical guide assumes readers have expertise in embedded systems and FPGA technologies.

1.3. Driver Versioning

1.3.1. 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	3.2.0

Table 1.2. Quick Facts of Driver Tested Environment

Driver tested on HW devices	PC environment
CertusPo-NX PCIe Bridge 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\cpnxdma_24.02.00* before you install the driver.
3. Double click on *cpnxdma-24.02.00-win64.exe* to install the driver. Select **Next** to continue with the installation.

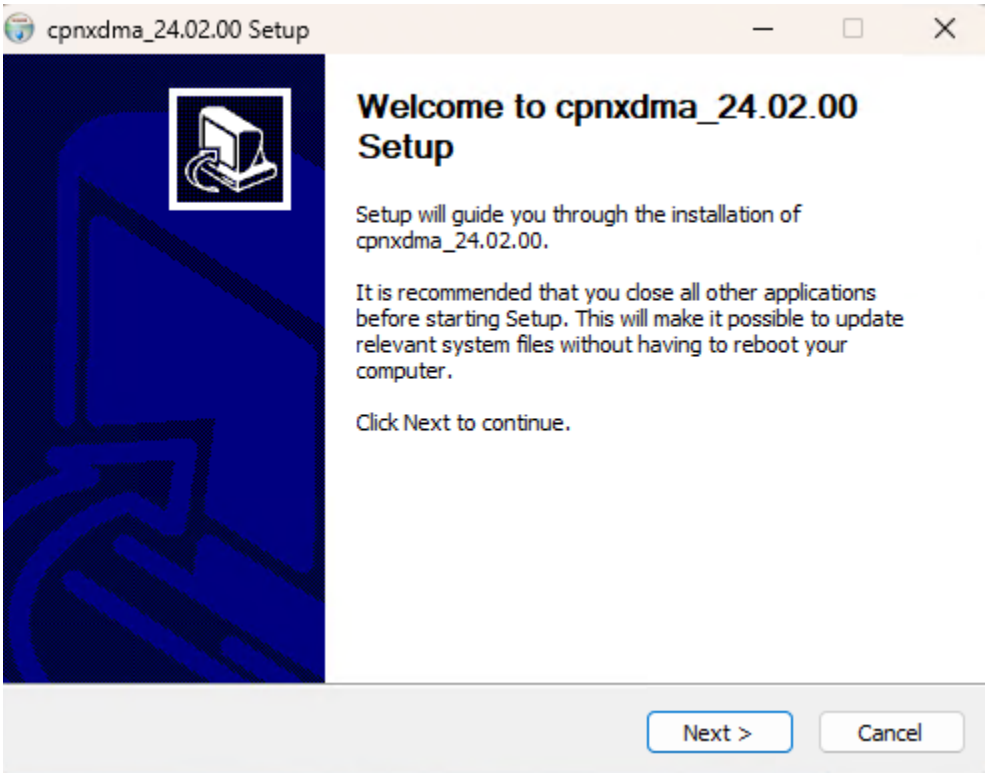

Figure 2.1. cpnxdma_24.02.00 Setup

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

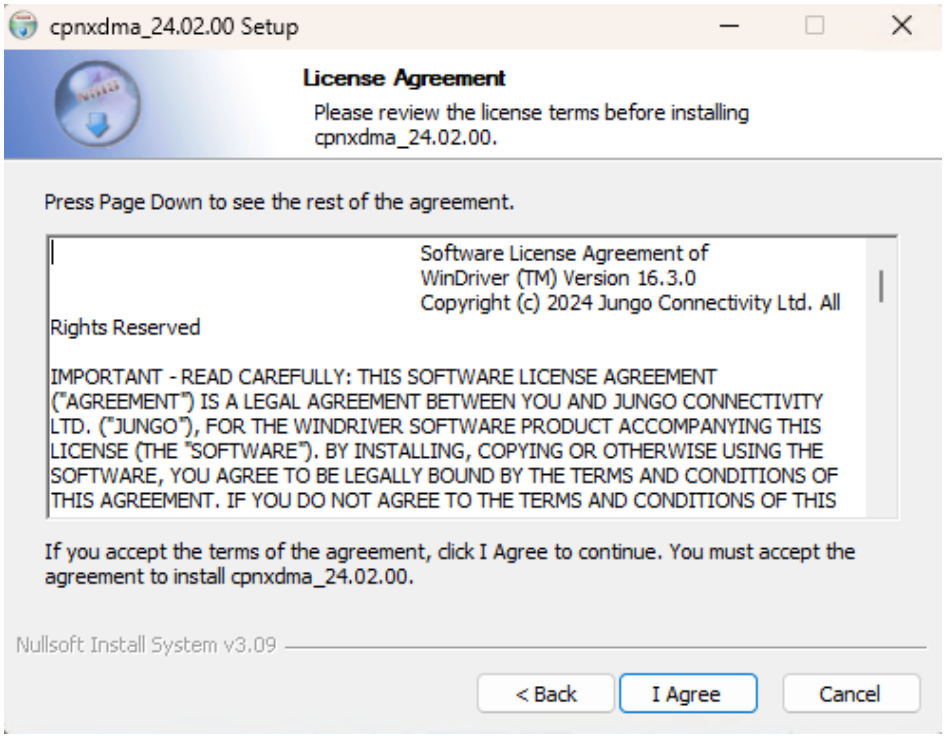

Figure 2.2. cpnxdma_24.02.00 Setup License Agreement

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

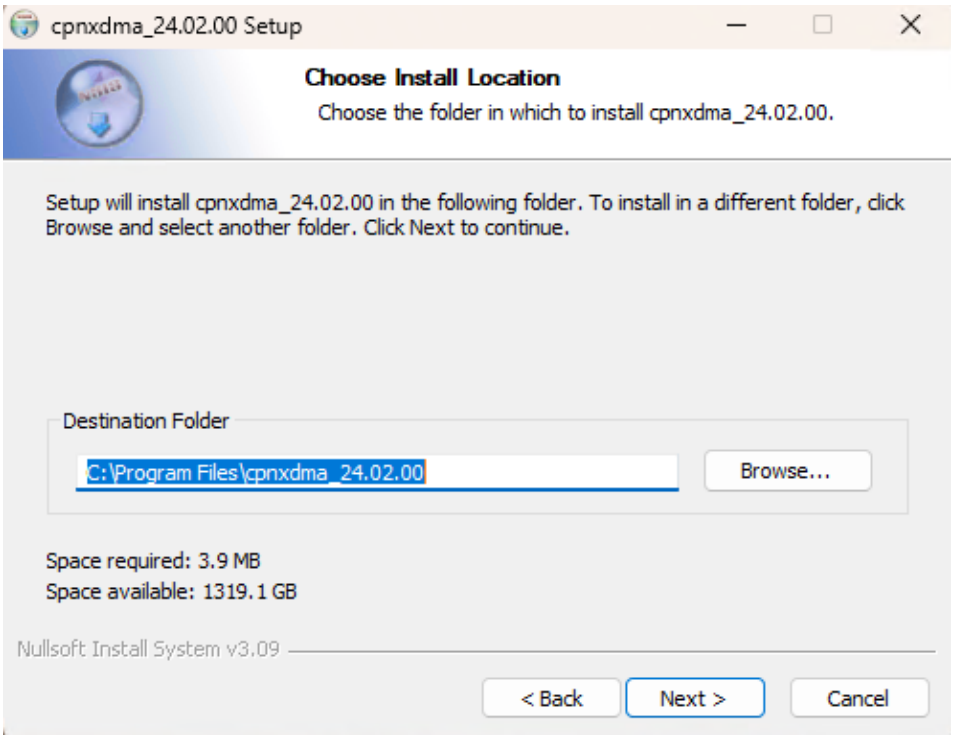

Figure 2.3. cpnxdma_24.02.00 Setup Install Location

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

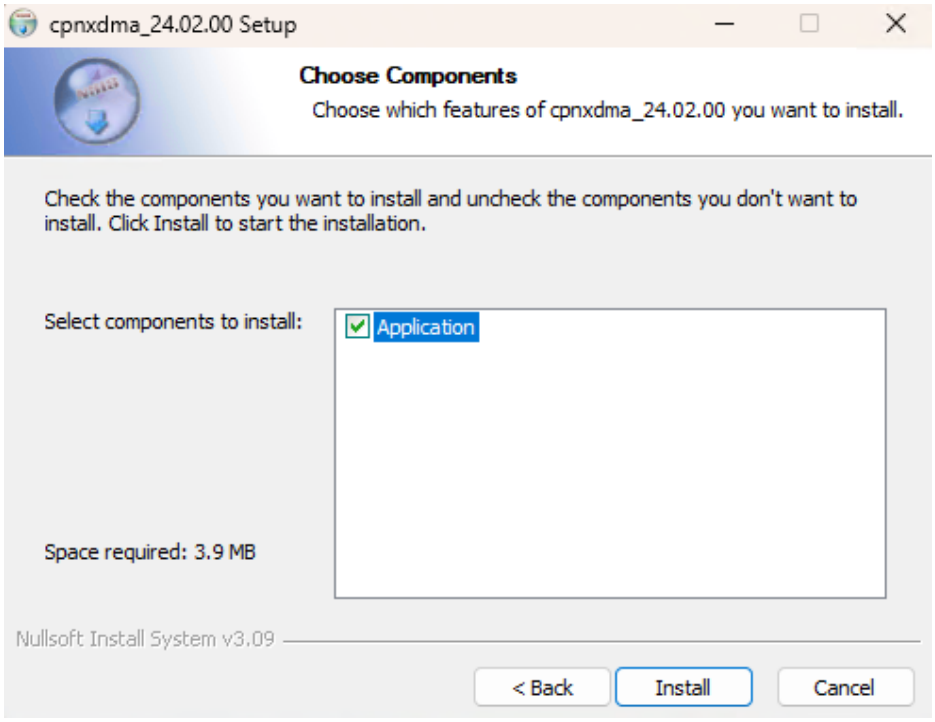

Figure 2.4. cpnxdma_24.02.00 Setup Install

7. When installation is complete, open **Device Manager**. Two new devices are listed under **Jungo Connectivity**:
   cpnxdma Driver
   PCIe (Gen3 – Lattice-Device: 0x9c25 (Requiring driver: cpnxdma)

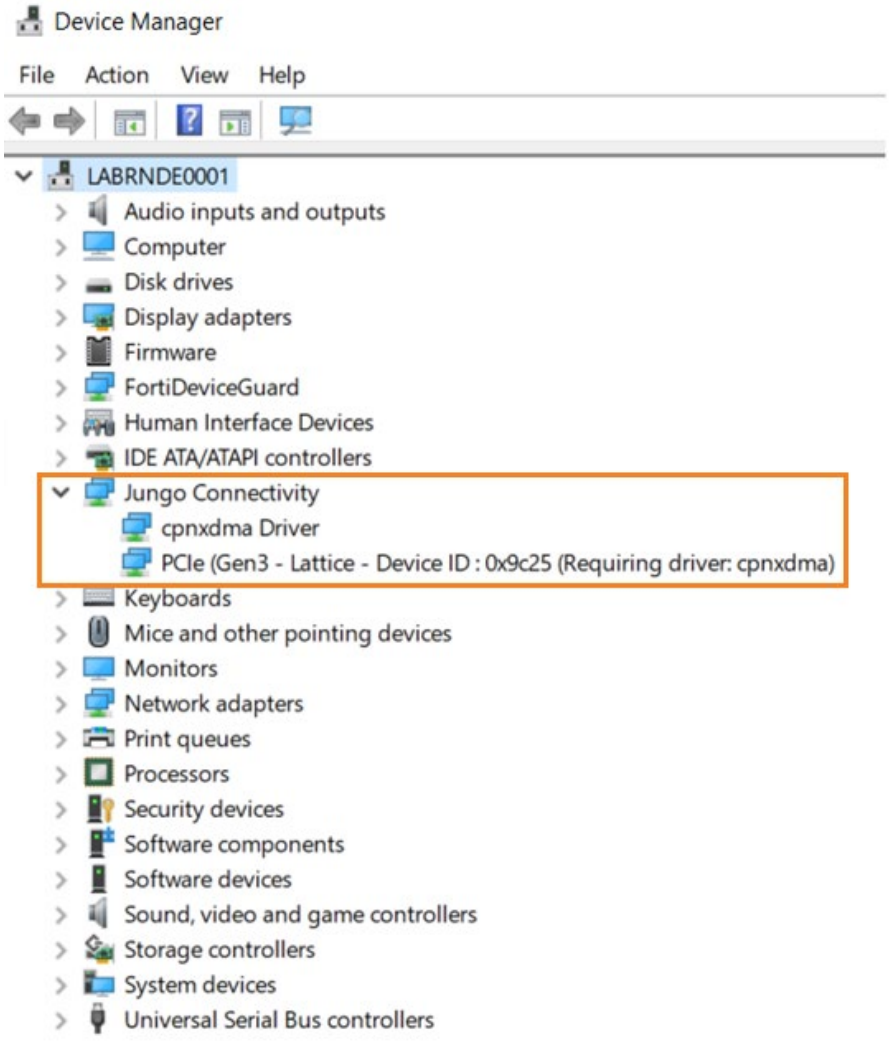

Figure 2.5. Device Manager

8. A new folder is created in *C:\Program Files\cpnxdma_24.02.00*.
9. A shortcut: *cpnxdma_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\cpnxdma_24.02.00\bin\cpnxdma.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 *cpnxdma-24.02.00-Linux.sh* is located. Give executable permission to the *.sh* file:
   ```
   sudo chmod +x cpnxdma-24.02.00-Linux.sh
   ```
4. Then, run the command below to build the driver and user application and install the driver.
   ```
   sudo ./cpnxdma-24.02.00-Linux.sh
   ```
5. Enter *Y* when prompted.
6. Two new modules are installed. Run the command below to verify that *cpnxdma* is installed:
   ```
   sudo lsmod | grep cpnx
   ```

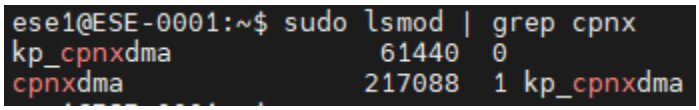

Figure 2.6. lsmod

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

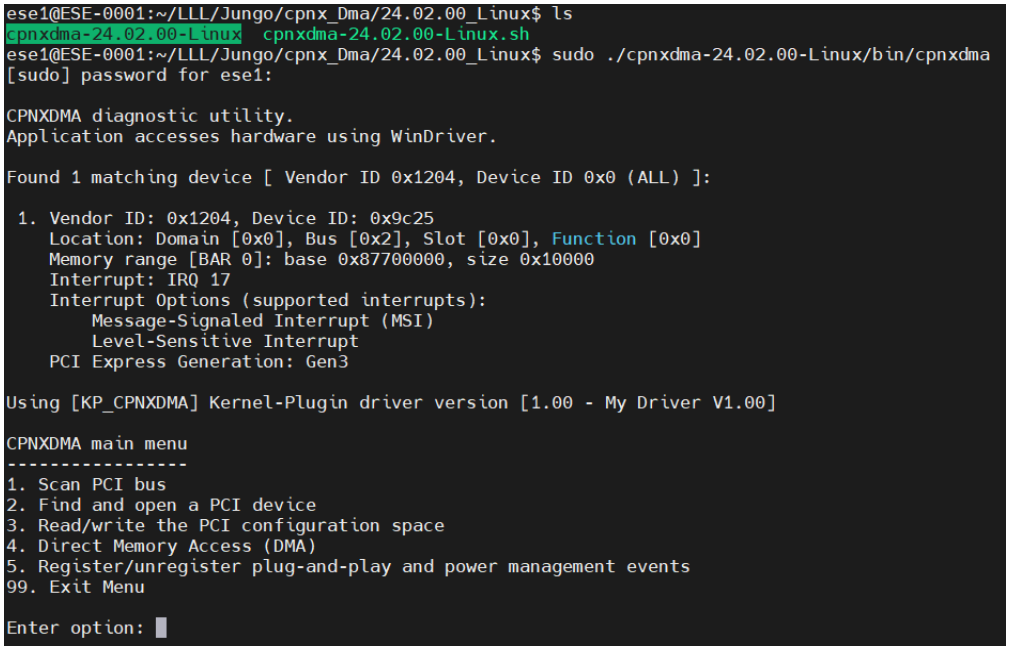

Figure 2.7. Run cpnxdma

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 the PCI configuration space
- Direct Memory Access (DMA)

3.1. DMA Functionalities

Below are the DMA functionalities provided by the user application.

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

Enter option: 4

CPNXDMA DMA menu
----------------
1. Perform DMA transfer
2. Perform DMA transfer to device, perform DMA transfer from device and compare the DMA buffers
3. Measure DMA performance
99. Exit Menu

Enter option:
```

Figure 3.1. CPNXDMA 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.

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

Enter option: 1

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

Enter number of packets to transfer (32 bit packets) (to cancel press 'x'): 5

Enter device memory address for transfer (to cancel press 'x'): 0x0

Running DMA using Kernel-PlugIn driver [KP_CPNXDMA]

###
Message Signalled Interrupt #1 received
MSI data 0x22
###

Buffer:

0x12345678 0x12345678 0x12345678 0x12345678 0x12345678


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 will perform DMA transfer from host to device, followed by a DMA transfer from device to host, then compare the values of the data read to data written. The status of the DMA buffer compare will be reported on the console.

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

Enter option: 2

Enter DMA data pattern as 32 bit packet (to cancel press 'x'): 0x12345678

Enter number of packets to transfer (32 bit packets) (to cancel press 'x'): 10

Enter device memory address for transfer (to cancel press 'x'): 0x0

Running DMA using Kernel-PlugIn driver [KP_CPNXDMA]

###
Message Signalled Interrupt #1 received
MSI data 0x22
###


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

###
Message Signalled Interrupt #2 received
MSI data 0x22
###

Buffer:

0x12345678 0x12345678 0x12345678 0x12345678 0x12345678 0x12345678 0x12345678 0x12345678 0x12345678 0x12345678


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

Figure 3.3 DMA Data Compare

3.1.3. DMA Performance Measure

When selecting **Measure DMA performance**, ensure the PCIe link status is showing *Link Speed* of 8GT/s and *Link Width* of x4 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*.

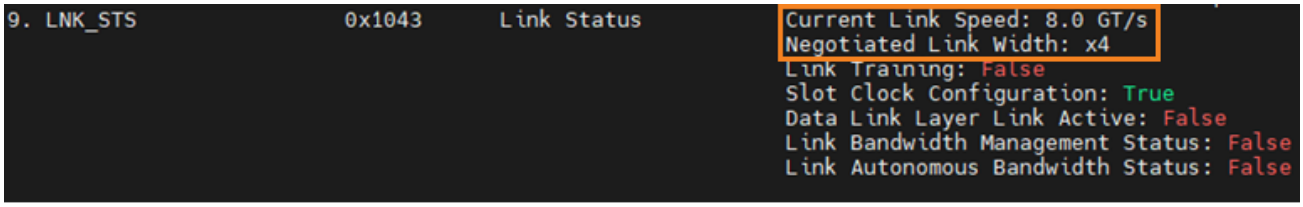

Figure 3.4. PCIe Link Status Register

If the link is not 8.0 GT/s, verify that the FPGA card is connected to a suitable slot that supports PCIe Gen3 and that the slot is properly configured to support Gen3. You may need to configure the PCIe speed in BIOS.

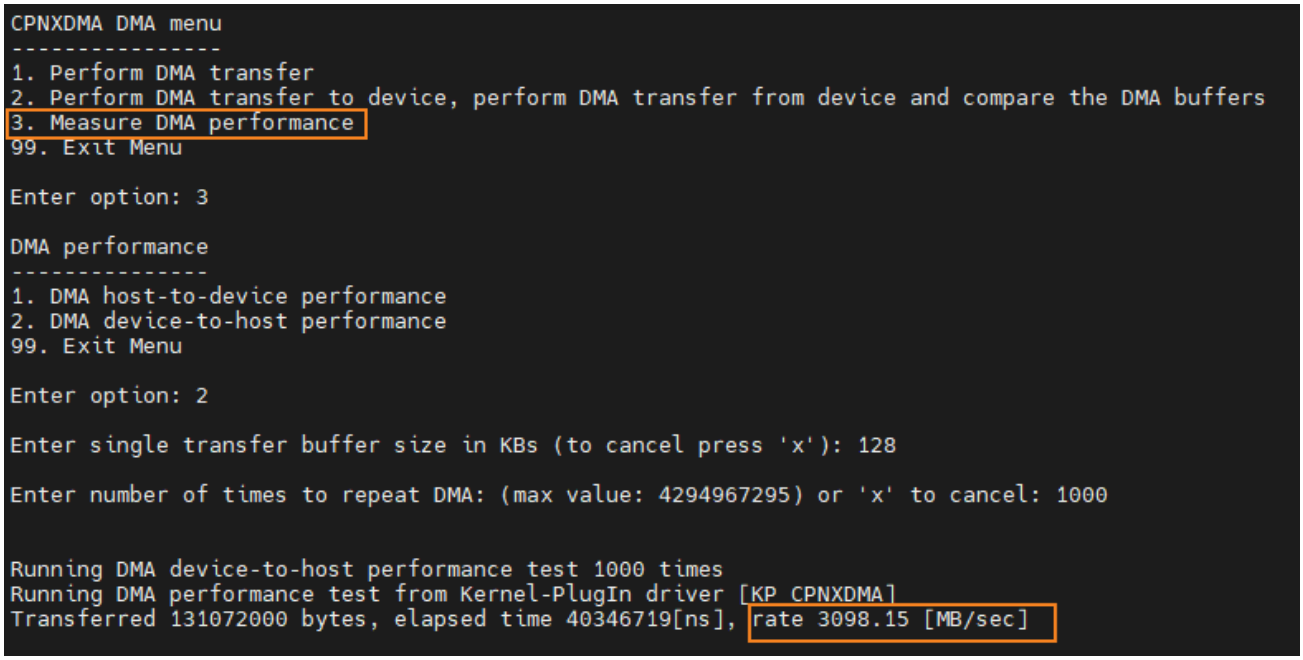

Figure 3.5 Measure DMA Performance

4. APIs

The APIs are defined by Jungo's 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. • Here, the software will pass in a user-mode interrupt handler callback function (DiagcpnxdmaTransferIntHandler), which will be called after an interrupt is received and processed in the kernel. • The last argument is set to TRUE as the driver uses a Kernel Plugin driver.
WDC_IntDisable	Disables interrupt handling for the device.
WDC_DMASGBufLock	• Locks a pre-allocated user-mode memory buffer for DMA which is passed in as 2nd argument of the API • Returns the corresponding physical mappings of the locked DMA pages as the 5th argument of the API.
WDC_DMAContigBufLock	• Allocates a contiguous descriptor buffer, locks it in physical memory • Returns mapping of the allocated descriptor buffer to physical address and to 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, 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.

5. Software Flow Diagrams

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

5.1. CPNXDMA_DmaOpen

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

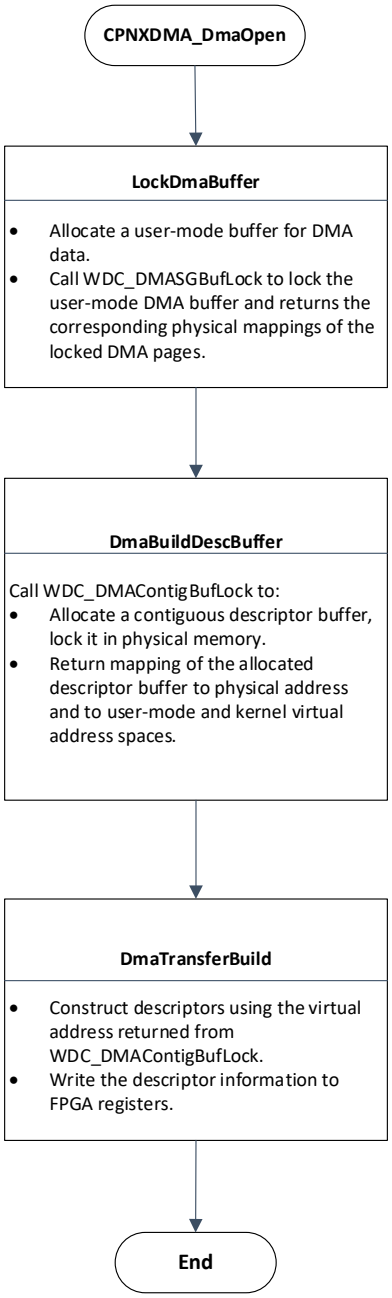

Figure 5.1. CPNXDMA_DmaOpen

5.2. DmaPerformanceSingleDir

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

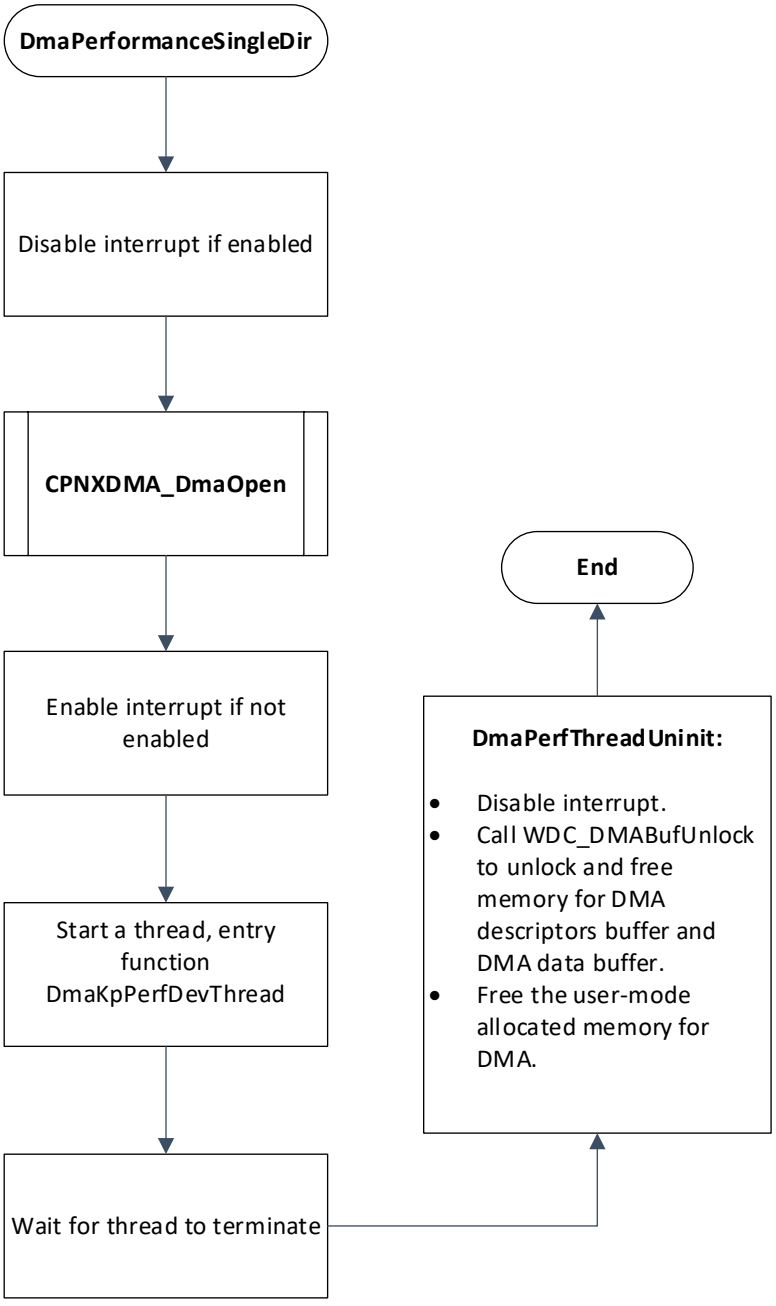

Figure 5.2. DmaPerformanceSingleDir

5.3. DmaKpPerfDevThread

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

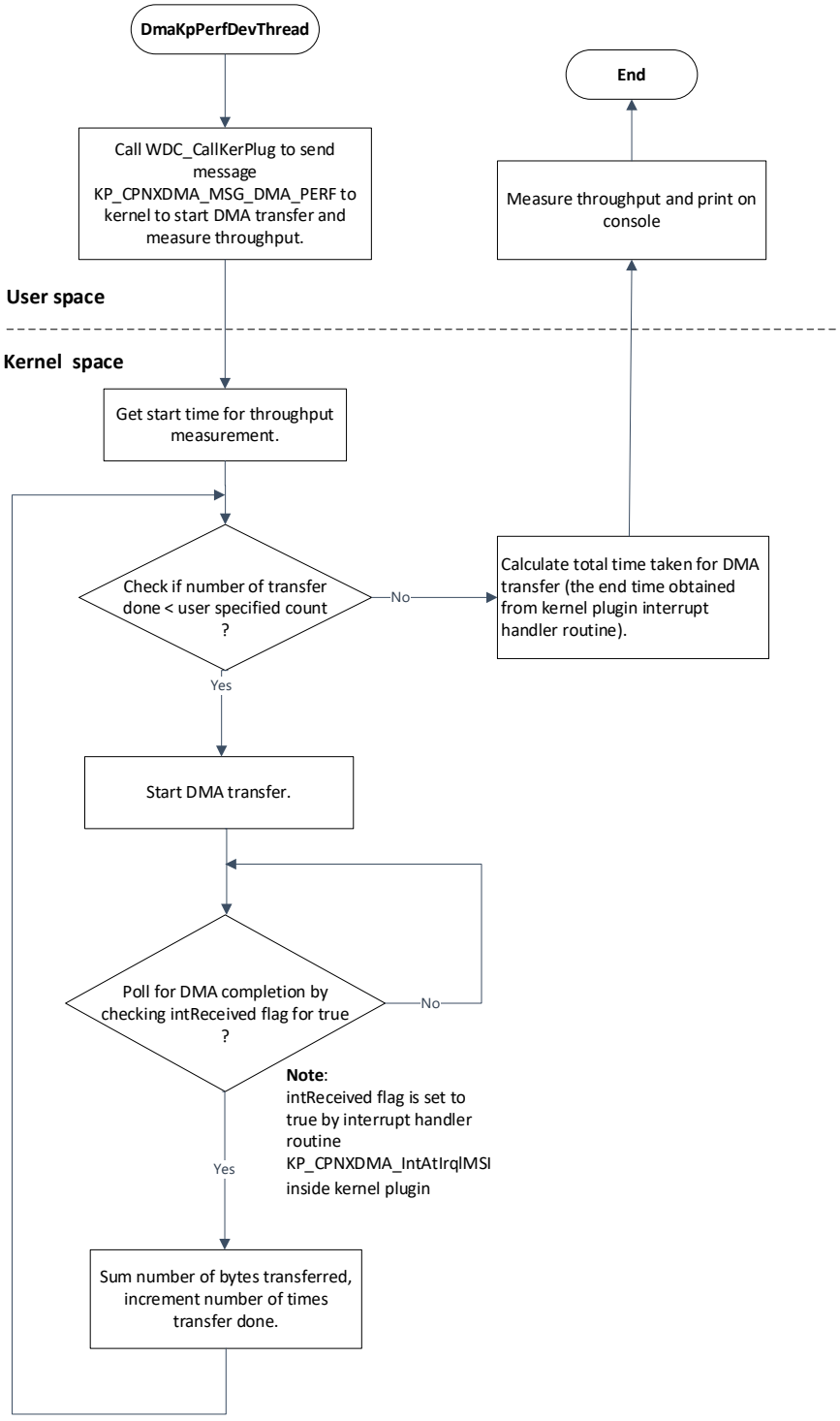

Figure 5.3. DmaKpPerfDevThread

Reference

- Jungo WinDriver: Introduction Jungo WinDriver official documentation
- PCIe X4 IP Core User Guide (FPGA-IPUG-02126)
- PCI Express x1/x2/x4 Endpoint IP Core User Guide (FPGA-IPUG-02009)
- CertusPro-NX web page
- Lattice Solutions IP Cores web page
- Lattice Insights 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, December 2024

Section	Change Summary
All	Initial release.

www.latticesemi.com