Xilinx Answer 65444

Xilinx PCI Express Windows DMA Drivers and Software Guide

Important Note: This downloadable PDF of an Answer Record is provided to enhance its usability and readability. It is important to note that Answer Records are Web-based content that are frequently updated as new information becomes available. You are reminded to visit the Xilinx Technical Support Website for the latest version of this Answer.

Introduction

The Xilinx PCI Express DMA IP provides high-performance direct memory access (DMA) via PCI Express. The PCIe DMA can be implemented in Xilinx 7 Series XT and UltraScale devices. This answer record provide drivers and software that can be run on a PCI Express root port host PC to interact with the DMA endpoint IP via PCI Express. The drivers and software provided with this answer record are designed for Windows 7 operating systems and can be used for lab testing or as a reference for driver and software development. Through the use of the PCIe DMA IP and the associated drivers and software you will be able to generate high-throughput PCIe memory transactions between a host PC and a Xilinx FPGA.

PCIe DMA Driver for Windows Operating Systems

The following operating system is supported:

-   Windows 7 Enterprise 64-bit

Driver Installation

Follow the steps below to install the PCIe Xilinx DMA driver on Windows. Note that you will need administrator privileges to complete installation:

1. Download and unzip the 'Xilinx_Answer_65444_Windows_Files.zip' zip file supplied in this Answer Record.
2. The Unzipped directory should have the following content:
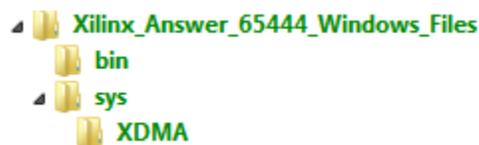
3. Open the *Device Manager* (*Control Panel -> System-> Device Manager*)
4. Initially, the device will be displayed as a *PCI Memory Controller* device.
5. Right-Click on the device and select *Update Driver Software* and select the folder of the built XDMA driver (located in *sys/XDMA/)*.
6. If prompted about unverified driver publisher, select '*Install this driver software anyway'* (see note below).

Note: The driver does not provide a certified signature and uses a test signature instead. Please be aware that you may need to enable test-signed drivers in your windows boot configuration in order to enable installation of this driver. See [MSDN](#) for further information.

7. *Xilinx Drivers -> Xilinx DMA* should now be visible in the *Device Manager*

Sample Applications

Some basic applications that use the PCIe DMA kernel module driver have been included for reference. They can be found in the *bin/* directory of the supplied Answer Record zip file. They are further detailed in the sections below.

Xdma_test

This application is designed to run with the PCIe example design which implements a 4KByte BRAM buffer in the user portion of the design. As such DMA transfers should be limited to 4 KByte transfers. For a 4 channel design this script transfers 4096 bytes on each channel. The following functions are performed:

- Determines how many h2c and c2h channels are enabled in the PCIe DMA IP
- Determines if the PCIe DMA core is configured for memory mapped (AXI-MM) or streaming (AXI-ST) modes
- Performs data transfers on all available h2c and c2h channels
- Verifies that the data written to the device matches the data that was read from the device
- Reports pass or fail completion status to the user

Usage

```
xdma_test.exe
```

Xdma_info

This application opens the XDMA *control* device node via *CreateFile()* and executes *ReadFile()* to read status and control registers of the XDMA IP core. These register values are then interpreted according to the register map in the IP Documentation (PG195). The IP core configuration and status is then printed to console in human readable format.

Usage

```
xdma_info.exe
```

Xdma_rw

This application can be used to open any of the device nodes and perform read/write operations. Typically this is useful for reading memory space of the *control* or *user* PCIe BARs. However it can also be used to perform aligned DMA transfers via the h2c_* *and c2h_* nodes, where the asterisk denotes the channel index (0-3). By default the host-side data buffer that this application allocates is memory aligned to the PAGE_SIZE boundary (typically 4kB).

Usage

```
xdma_rw.exe <DEVNODE> <read|write> <ADDR> [OPTIONS] [DATA]
    - DEVNODE : One of: control | user | event_* | hc2_* | c2h_*,
               where the * is a numeric wildcard (0-15 for events, 0-3 for hc2 and c2h).
    - ADDR :    The target offset address of the read/write operation.
               Can be in hex or decimal.
    - OPTIONS :
               -l (length of data to read/write)
```

```
            -b open file as binary
            -f use data file as input/output (for write/read respectively).
            -v more verbose output
            -a set host-side buffer alignment in bytes (default: PAGE_SIZE)
  - DATA :   Space separated byte data in decimal or hex,
            e.g.: 17 34 51 68
            or:   0x11 0x22 0x33 0x44
```

Examples

Read a 4 Byte control register at offset 0x1000:
```
xdma_rw.exe control read 0x1000 -l 4
```

Write 2 Bytes (0x1234) to a control register at offset 0x2004 (*Note that -l option is not required. When specifying data in byte sequence)*
```
xdma_rw.exe control write 0x2004 0x34 0x12
```

Read a 4 Byte control register at offset 0x1000:
```
xdma_rw.exe control read 0x1000 -l 4
```

Read 4kB from a C2H_0 at offset 0 into a binary file (my_data.bin).
```
xdma_rw.exe c2h_0 read 0 –l 0x1000 -b -f my_data.bin
```

Read 4kB of data from C2H channel 0 at offset 0x100
```
xdma_rw.exe c2h_0 read 0x100 -l 0x1000
```

Write 4 Bytes (0x12345678) to H2C channel 3 at offset 0x10.
```
xdma_rw.exe h2c_3 write 0x10 0x78 0x56 0x34 0x12
```

User_event

This application opens a user event device file and waits on the event to be triggered.

Usage

```
user_event.exe
```

Enabling the PCIe to AXI-Lite Master interface

During IP customization in Vivado the PCIe DMA IP can be customized to enable a DMA AXI-Lite Master interface. This selection is available on the PCIe:BARs tab of the PCIe customization GUI.
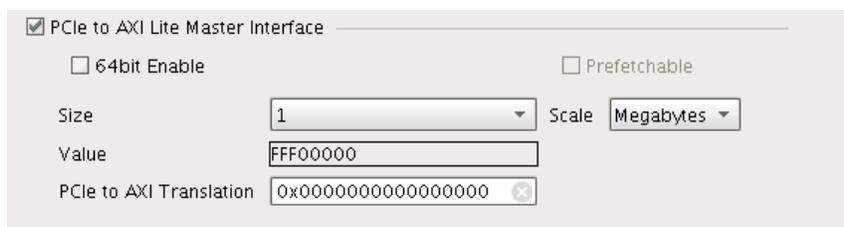
This interface exposes an AXI – Lite Memory Mapped interface to the user which can be used for control or other functions and also can be connected to AXI Interconnect IP. From example design this interface is connected to a 4Kbytes Bram. This memory region can be accessed through the following command using the provided software application.

Here is an example of how to read 4 bytes from AXI-Lite interface from offset (0x0000).

```
xdma_rw.exe user read 0 –l 4
```

Here is an example of how to write to the Bram at a specified offset (0x0000) with specific data (0x1234567).

```
xdma_rw.exe user write 0x0 0x67 0x45 0x23 0x01
```

Example to read 256 Bytes of user memory at offset 0x0:

```
xdma_rw.exe user read 0 -l 0x100
```

Example to get data from a binary file (my_data.bin) and write it into user memory at offset 0x0

```
xdma_rw.exe user write 0 -b -f my_data.bin
```

Enabling the PCIe to DMA Bypass interface in the PCIe DMA Driver

During IP customization in Vivado the PCIe DMA IP can be customized to enable a DMA bypass interface. This selection is available on the PCIe:BARs tab of the PCIe customization GUI.

This interface exposes an AXI Memory Mapped interface that bypasses the DMA and can be connected to an AXI system through the AXI Interconnect IP.

This memory region can be accessed through the following command using the provided software application.
Here is an example of how to read 4 bytes from the bypass channel at a specified offset (0x0000).

```
xdma_rw.exe bypass read 0 –l 4
```

Here is an example of how to write to the bypass channel at a specified offset (0x0000) with specific data (0x1234567).

```
xdma_rw.exe bypass write 0 0x67 0x45 0x23 0x01
```

Device ID Support

Below is a list of Xilinx Device IDs that are covered in the current windows driver.

PCI\VEN_10ee&DEV_9011	PCI\VEN_10ee&DEV_8011	PCI\VEN_10ee&DEV_7011
PCI\VEN_10ee&DEV_9012	PCI\VEN_10ee&DEV_8012	PCI\VEN_10ee&DEV_7012
PCI\VEN_10ee&DEV_9014	PCI\VEN_10ee&DEV_8014	PCI\VEN_10ee&DEV_7014
PCI\VEN_10ee&DEV_9018	PCI\VEN_10ee&DEV_8018	PCI\VEN_10ee&DEV_7018
PCI\VEN_10ee&DEV_9021	PCI\VEN_10ee&DEV_8021	PCI\VEN_10ee&DEV_7021
PCI\VEN_10ee&DEV_9022	PCI\VEN_10ee&DEV_8022	PCI\VEN_10ee&DEV_7022

PCI\VEN_10ee&DEV_9024	PCI\VEN_10ee&DEV_8024	PCI\VEN_10ee&DEV_7024
PCI\VEN_10ee&DEV_9028	PCI\VEN_10ee&DEV_8028	PCI\VEN_10ee&DEV_7028
PCI\VEN_10ee&DEV_9031	PCI\VEN_10ee&DEV_8031	PCI\VEN_10ee&DEV_7031
PCI\VEN_10ee&DEV_9032	PCI\VEN_10ee&DEV_8032	PCI\VEN_10ee&DEV_7032
PCI\VEN_10ee&DEV_9034	PCI\VEN_10ee&DEV_8034	PCI\VEN_10ee&DEV_7034
PCI\VEN_10ee&DEV_9038	PCI\VEN_10ee&DEV_8038	PCI\VEN_10ee&DEV_7038
PCI\VEN_10ee&DEV_903f		

Debug Messages in the Driver

The driver provides debug messages via WPP tracing mechanism. A simple way to get access to these trace messages is via the TraceView program which is part of the Windows Driver Development Kit (WDK). The TraceView program can usually be found in '*tools/tracing/ARCH' folder of the WDK*. Please follow the steps below for

1. From the menu select *File-> Create New Log Session.*
2. Click on *Add Provider.*
3. Select *PDB (Debug Information) File* and navigate to the *XDMA.pdb* file in the driver binaries folder (*sys/*) and select OK.
4. Select *Next*.
5. Select *Real Time Display.*
6. Click on Set Flags and Level.
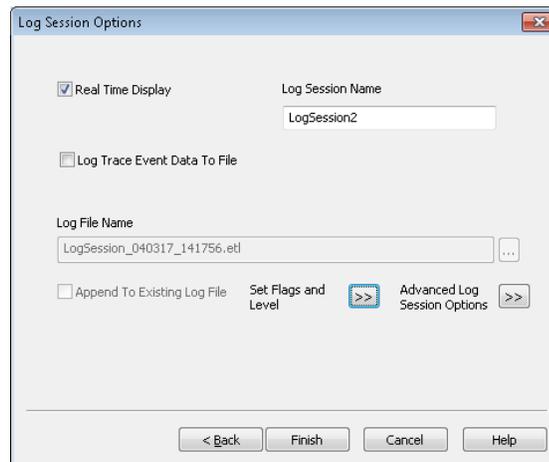
7. Double Click on Level and set to Verbose and click OK.
8. Double click on Flags and ensure all flags are selected.
9. Click on Finish.
10. Run any user-space application to generate driver activity. Debug messages should now appear in the message window as shown below.
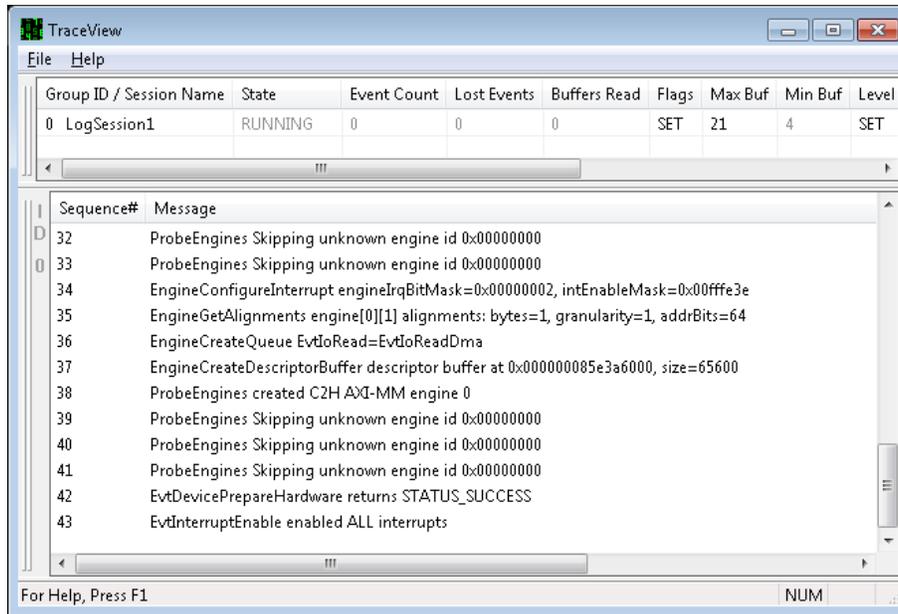
Uninstalling the PCIe DMA Driver

The Windows Device Manager should be used to uninstall the driver.

1. In the Device Manager, locate the 'Xilinx Drivers -> *Xilinx DMA*' entry.
2. Right-Click on the driver and select *Uninstall*.
3. Follow the on-screen instructions to complete the uninstallation.

Driver Source

The source files are available for the driver binary files provided with this answer record. Please request the source code access in the link below:

https://www.xilinx.com/member/xdma_windows_driver.html

Known Issues

*   Driver installation gives warning due to test signature.
*   Non-incremental mode not supported
*   Adjacent descriptor fields not used
*   Descriptor credits feature not supported

Revision History

04/20/2017
*   Initial Release
09/22/2017
*   Driver file update
01/22/2018
*   Windows 10 Support

- Support for non-incremental addressing mode
- Utilization of "adjacent descriptor" field of descriptors in the driver
- Architectural split of the driver into library component and driver component
- Bug fixes