XAPP1289 (v1.0) June 20, 2016

Using DMA with Zynq UltraScale+ MPSoC Controller for PCI Express as Root Port

Authors: Bharat Kumar G., Sunita Jain, Jason Lawley

Summary

In a typical system with PCI Express® architecture, Endpoints often contain a DMA engine controlled by the system host to transfer data between system memory and the Endpoint. This removes the task from the processor, enabling the processor to have increased cycles to perform other operations. The downside of this type of system is the increased latency of the Endpoint having to be told where to fetch the data when moving data from the system to the card (Endpoint).

The Zynq® UltraScale+™ Controller for PCI Express has a built-in DMA engine that can be used in Endpoint as well as Root Port mode. By using the built-in DMA engine while in Root Port mode, designers can reduce latency and possibly increase system performance in a way not available in many other processing subsystems

This application note provides an example that demonstrates how to configure and use the DMA in the Controller for PCI Express when the controller is configured as a Root Port. In addition, performance observations for moving data from system to card and card to system are also shown. This design targets the ZCU102 hardware platform allowing for development of a PCIe system ranging from Gen1 x1 to Gen2 x4 operating as a Root Complex.

Using this application note as a starting point, developers and system architects now have more tools to determine the most efficient way to move data. Whether using the DMA in the Controller for PCI Express on its own, or employing more complex schemes such as having DMA engines in both Endpoints and Root Ports, Zynq UltraScale+ provides the functionality and tools to enable these types of systems.

You can download the Reference Design Files for this application note from the Xilinx® website. For detailed information about the design files, see Reference Design, page 8.

Introduction

This design shows the use of DMA on the PCI Express Root Port to push data into an Endpoint or pull data from it. The Controller for PCI Express on Zynq UltraScale+ is used in Root Port mode along with the integrated DMA block.

The design uses a KCU105 board based design as Endpoint. The Endpoint design contains Memory Interface Generator IP (MIG) targeting DDR4 on KCU105) mapped to a PCIe BAR via Xilinx IP - AXI Bridge for PCI Express Gen3 v2.0. See Figure 1, page 2 for an overview of the design.

The address translation for PCIe BAR to DDR4 and AXI Performance Monitor is setup during IP customization. DMA on a Root Port is used for custom applications where the Endpoint can be used as a co-processor or accelerator block.

Requirements

Hardware

1. ZCU102 Board with power supply, USB-UART cables, SD card
2. KCU105 Board with power supply cable, USB-JTAG cable

Software

1. Vivado® Design Suite 2016.1
2. PetaLinux 2016.1

Design Overview

An overview of the design is depicted in Figure 1. The bitfile to be programmed on the KCU105 Endpoint is provided with the reference design package. Terminology used in this document is as follows:

- AXI memory refers to PS-DDR on the Zynq UltraScale+ MPSoC
- EP memory or PCIe memory refers to DDR on the KCU105 Endpoint

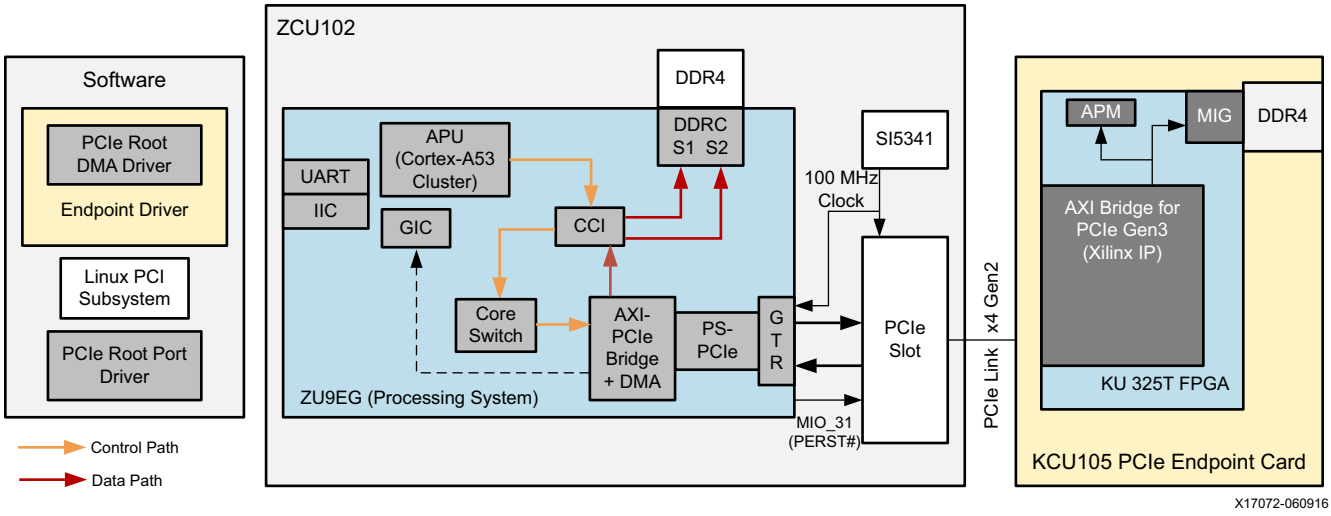

Figure 1: **Design Overview**

Note: S1 and S2 on DDRC represent slots through which the interface from the CCI connects to the DDRC.

Data Flow

Root Port-issued transactions (memory read or memory write) on the PCIe link are required to hit one of the Endpoint BARs (Base Address Register) to be accepted by the Endpoint. The DMA provides source and destination descriptors referred to as SRC-Q and DST-Q, respectively. For details on DMA operation see the Zynq UltraScale+ MPSoC TRM [Ref 1].

As shown in Figure 1, page 2 the controller for PCIe accesses PS-DDR via CCI (cache coherent interconnect). There are two interfaces connected to PS-DDR from CCI to PS-DDR, slot-1 and slot-2.

Note: In the following discussion, DMA refers to the DMA which is part of the controller for PCIe on the Zynq UltraScale+ MPSoC, and functioning as the Root Port.

Control Flow

The following occurs when the Root Port DMA driver executes on the APU SMP Linux:

1. Sets up the descriptor Qs (SRC and DST, and the respective status) in PS-DDR memory.
    a. The direction of data transfer is specified by the flags in the SRC and DST elements.
    b. All of the DMA descriptor Qs reside in PS-DDR (Root Port memory).
2. Programs the various DMA registers in the controller for PCIe required for DMA operation.

Data Flow

The following occurs after the DMA is set up by the Root Port DMA driver:

1. For S2C (Root Port-to-Endpoint) transfers,
    a. DMA issues reads in the AXI domain (to the address from the SRC-Q element) to fetch the data.
    b. The DMA then issues memory write transactions (to the address from the DST-Q element) on the PCIe link to transfer the data to the Endpoint.
    c. These writes are accepted by the Endpoint and the address is translated (from the PCIe BAR to the EP-DDR AXI domain address by the AXI Bridge for PCI Express Gen3 IP) to route the data to EP-DDR memory.
2. For C2S (Endpoint-to-Root Port) transfers,
    a. DMA on the Root issues reads on the PCIe link (to the address from the SRC-Q element). This downstream memory read gets translated to the EP-DDR AXI domain address by the AXI Bridge for PCI Express Gen3 IP.
    b. The data read from EP-DDR AXI domain address is received over the PCIe link as completion with data.
    c. DMA issues a write to the AXI domain (in the Root Port, to the address from the DST-Q element) to transfer the data received over the PCIe link to PS-DDR memory.

Components

The software components are shown in Figure 2.

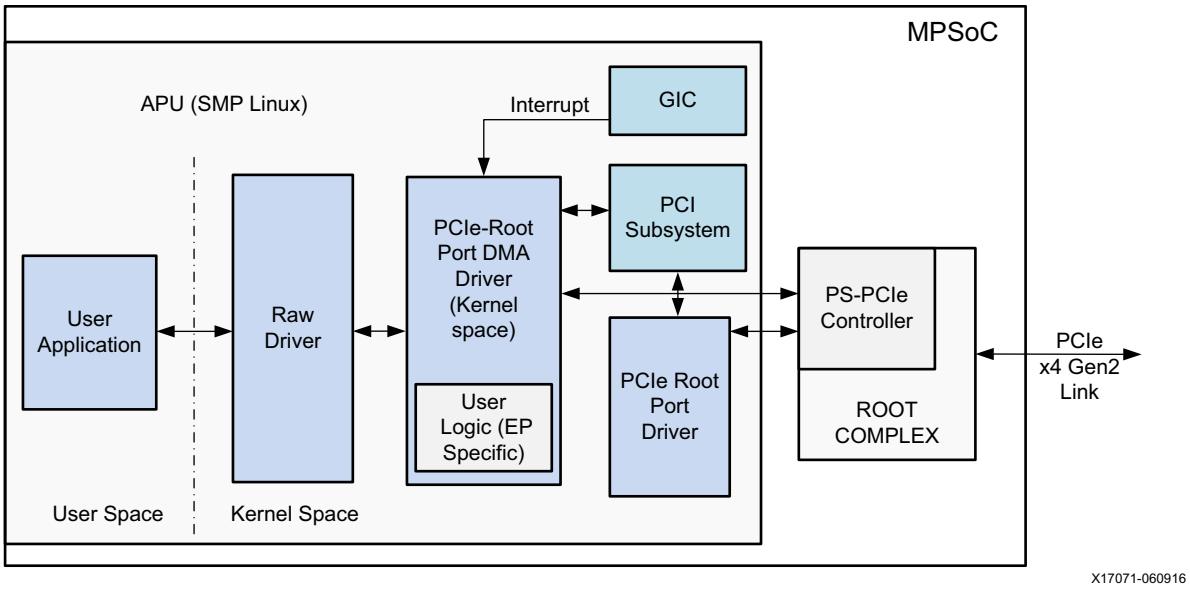

Figure 2: **Software Overview**

The main components of the design include:

1. **First Stage BootLoader (FSBL):** The FSBL sets up the ZCU102 with respect to various peripherals, clocks, PS-DDR, and programs the PCI Express controller for Root mode operation.
2. **Root Port Driver:** This driver is part of the kernel provided by the Petalinux build. For detailed instructions on using the ZCU102 as a PCIe Root Port see http://www.wiki.xilinx.com/ZynqMP+Linux+PCIe+Root+Port. This driver sets up the AXI-PCIe bridge in the PCI Express controller on the MPSoC and connects to the Linux PCI subsystem for enumeration.
3. **Root Port DMA Driver:** This driver manages DMA on the MPSoC's PCI Express controller.
    a. All DMA channel Qs (source, destination and corresponding status Qs) are managed by this driver.
    b. All Qs are resident on AXI (PS-DDR) memory.
    c. For downstream (Root Port-to-Endpoint) transfers, source buffers are in AXI memory and destination buffers in the Endpoint's PCIe memory.
    d. For upstream (Endpoint-to-Root) transfers, source buffers are in the Endpoint's PCIe memory and destination buffers are in AXI memory (PS-DDR).
    e. The DMA interrupt in the AXI domain from the MPSoC's PCI Express controller is used; no interrupts from the Endpoint are used.

f. There is a user-space application in the accompanying design that provides data buffers for downstream transfers and free buffers for receiving data from upstream transfers.

User Logic: This portion in the Root DMA driver performs the basic initialization task required for the Endpoint (specific to Endpoints). In the current design (KCU105 Endpoint), this driver sets up the AXI Performance Monitor which is mapped to BAR4.

The DMA driver calls the PCI probe twice – once for the PCIe Root Port and once for the PCIe Endpoint. The Raw driver is stacked atop the DMA driver and hooks up with the user space application. It invokes relevant DMA driver APIs for data movement based on the direction of the data transfer. The user space application is a traffic generator.

4. **Device-tree:** This lists relevant nodes for the PCI Express controller and is used by the Root Port driver and the Root DMA driver.

Performance Monitoring

The AXI Performance Monitor (APM) is used to gather throughput statistics using the read and write byte counts. For Zynq UltraScale+ MPSoC, the integrated APM in the PS is used. Throughput is monitored at the following locations:

1. Zynq UltraScale+ MPSoC's PS-DDR

   Traffic from the controller for PCI Express flows into the PS-DDR via slot-1 and slot-2. The statistics here include APU access to PS-DDR, Q fetch/update by DMA engine, apart from read and write access for actual data packets. See the Zynq UltraScale+ MPSoC TRM [Ref 1] for details on APM and PS-DDRC slot locations.

2. KCU105 EP-DDR

   This measures the actual data payload throughput (without any PCIe protocol or translation layer packet overheads) going into the EP-DDR.

Test Instructions

Refer to http://www.wiki.xilinx.com/XAPP1289+PCIe+Root+DMA for details on test setup and test instructions.

Results

Figure 3 summarizes system-to-card (S2C) and card-to-system (C2S) performance, respectively. It can be observed that:

- Throughput variation with packet size is seen as expected. (Packets are generated in the user space application.)
- For a given packet size, throughput for downstream traffic (S2C) is seen to be better than that for upstream traffic (C2S).
  - For downstream traffic, DMA issues memory write transactions on the PCIe link whereas for upstream traffic, there are memory read transactions on the PCIe link.

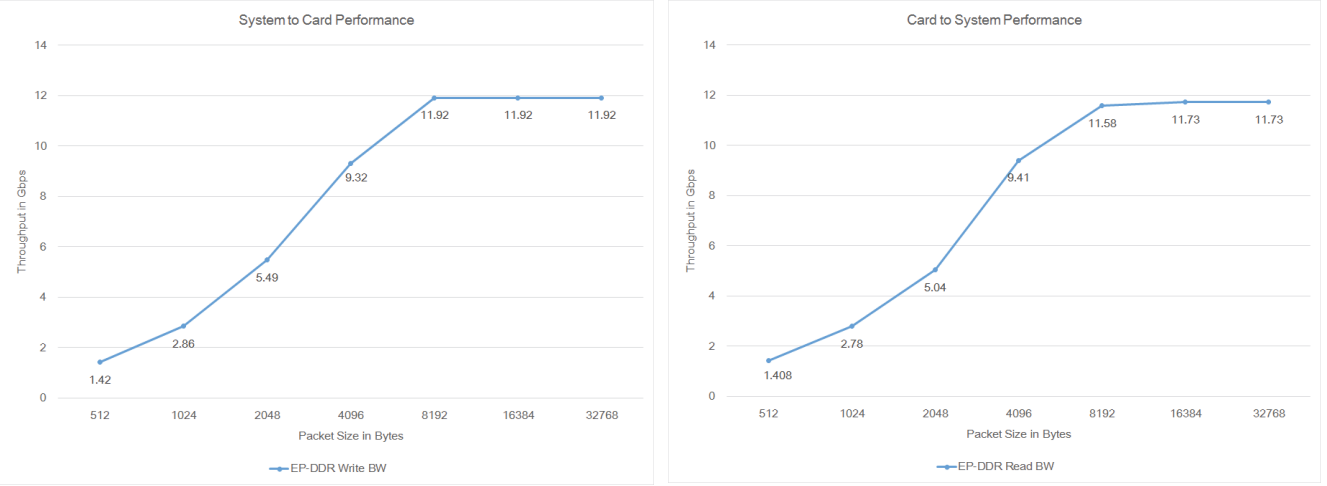

Figure 3: **Performance Summary**

Note: The Endpoint DDR bandwidth includes only the data payload performance.

The actual PCIe link throughput is expected to be similar to observed EP-DDR bandwidth.

Figure 4 plots DDR read bandwidth at the Root Port against packet size for C2S traffic. For C2S, reads to the PS-DDR predominantly include source and destination descriptor reads.

- It can be seen that for C2S traffic with smaller packet sizes, read bandwidth on the Root Port PS-DDR is high. This is because, for C2S, reads to the PS-DDR predominantly include source and destination descriptor reads. For smaller packet sizes, there are more frequent descriptor fetch operations when compared to larger packet sizes.

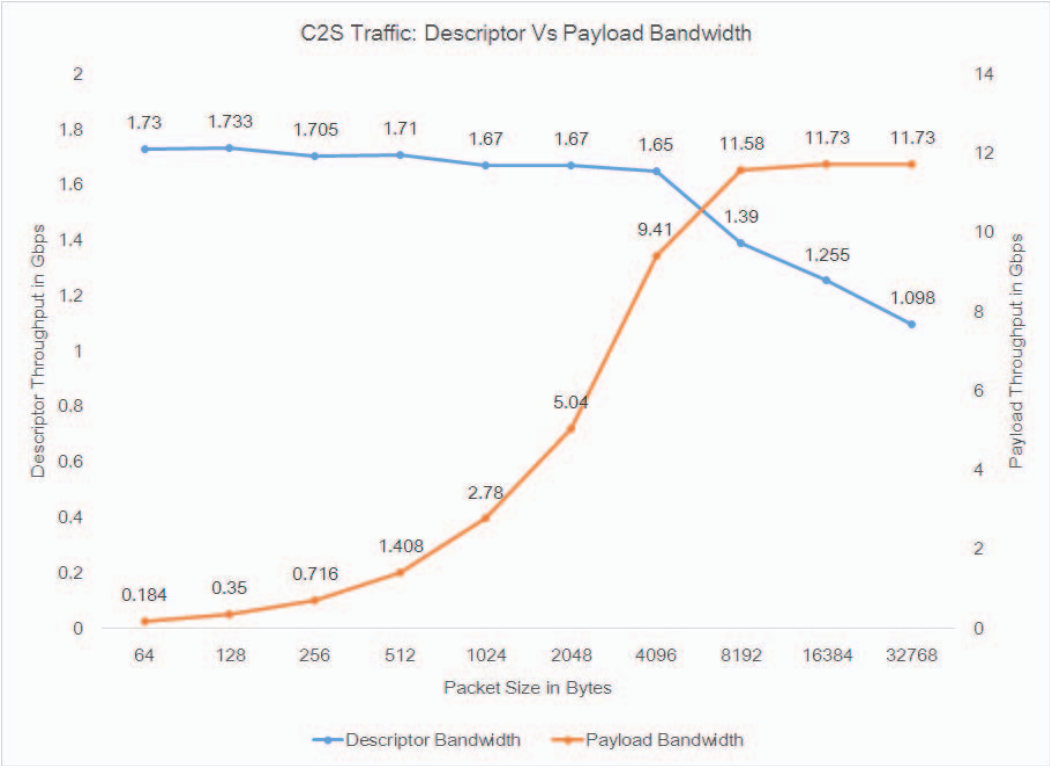

Figure 4: **RP DDR Read Bandwidth Variation (C2S Traffic)**

Figure 5 provides a performance comparison summary across use of DMA on the Root Port versus use of DMA on the EP. It provides the following observation:

- With the MPSoC as the Root Port, performance numbers with Root DMA and EP DMA are almost similar.

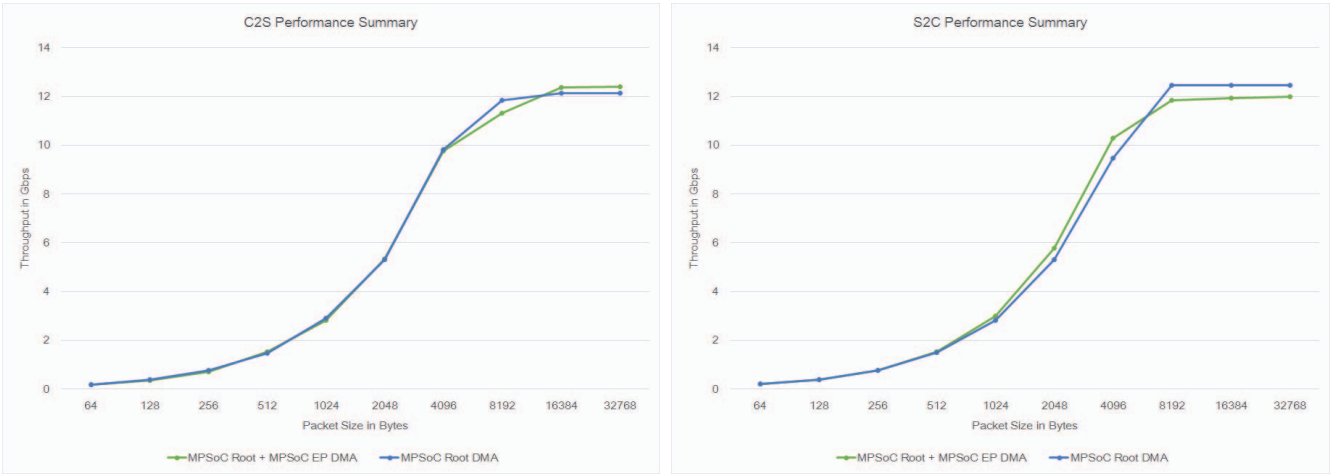

Figure 5: **Performance Comparison Root DMA vs EP-DMA**

Results of single packet transfer completion time between Root Port DMA and Endpoint DMA follows. These tests were conducted on a Zynq UltraScale+ MPSoC device as a Root Port.

- It can be seen in Figure 6 that single packet transfer latency with Root Port DMA is less than that with Endpoint DMA. This is due to fewer DMA setup overheads associated with Root Port DMA.

  Note: The latency computation is performed in software (in kernel space from start of transmission to completion interrupt reception) and has been averaged over multiple iterations.

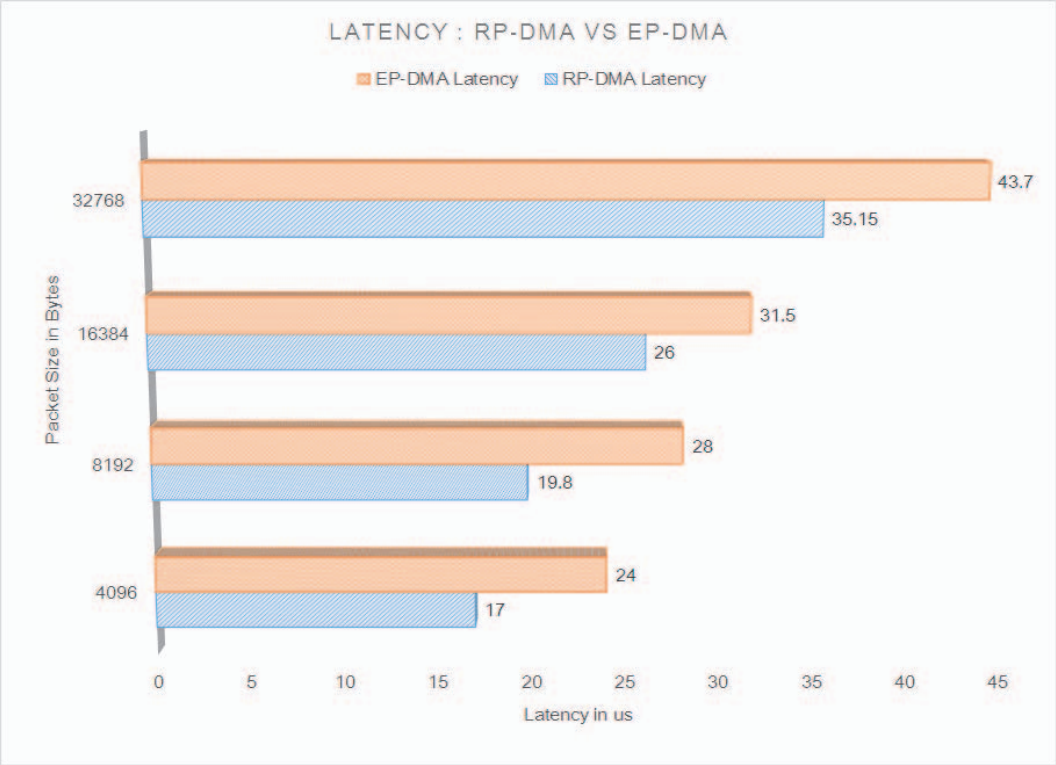

Figure 6: **Single Packet Transfer Latency**

Reference Design

You can download the Reference Design Files for this application note from the Xilinx website.

Table 1 shows the reference design matrix.

Table 1: **Reference Design Matrix**

Parameter	Description
General	
Developer Name	Bharat Kumar G., Sunita Jain, Jason Lawley
Target Devices	XCZU9EG-FFVB1156
Source code provided	Yes

Table 1: **Reference Design Matrix *(Cont'd)***

Parameter	Description
Source code format	C
Design uses code and IP from existing Xilinx application note and reference designs or third party	No
Implementation	
Synthesis software tools/versions used	Vivado 2016.1
Implementation software tools/version used	Vivado 2016.1
Static timing analysis performed	N/A
Hardware Verification	
Hardware verified	Yes
Hardware platform used for verification	ZCU102

Conclusion

Using the Zynq UltraScale+ Controller for PCI Express with DMA when implementing a Root Port design can meaningfully reduce the time it takes to send data from the system to the card (S2C) when compared to using a DMA located in an Endpoint. This application note and accompanying reference design provide an example of how to setup and use the DMA while in Root Port mode. The application note also measured and displayed time differences in latency of actual hardware designs demonstrating the lower latency possible for S2C transfers when using a DMA in the Root Port. Designers should consider DMA in the Root Port and might find that having DMAs in both the Endpoint and Root Port provide a unique value proposition for low latency designs.

References

This application note uses the following references:

1. *UG1085, Zynq UltraScale+ MPSoC TRM* (UG1085)
2. *AXI Performance Monitor Product Guide* (PG037)
3. *AXI Bridge for PCI Express Gen3 Subsystem v2.1 Product Guide* (PG194)

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
06/20/2016	1.0	Initial Xilinx release.

Please Read: Important Legal Notices