

## Introduction

The *PCI Express High-Performance Reference Design* highlights the performance of the hard IP implementation of Altera® PCI Express MegaCore® function. The design includes a high-performance chaining direct memory access (DMA) that transfers data between the Stratix® IV GX FPGA or Arria® II GX internal memory and the system memory. The reference design includes a Windows XP-based software application that sets up the DMA transfers. The software application also measures and displays the performance achieved for the transfers. This reference design enables you to evaluate the performance of the hard IP implementation of PCI Express protocol in a Stratix IV GX or Arria II GX device.

Altera offers the PCI Express MegaCore function in both hard IP and soft IP implementations. The hard IP implementation is available as a root port or endpoint. The hard IP implementation is compliant with *PCI Express Base Specification 1.1 or 2.0*. The soft IP implementation is available only as an endpoint. It is compliant with *PCI Express Base Specification 1.0a or 1.1*.

The remainder of this application note includes a tutorial on calculating the throughput of the PCI Express MegaCore function and instructions for running the chaining DMA design example. The chaining DMA in this reference design is the chaining DMA example generated by the PCI Express Compiler. This example is explained in detail in the *PCI Express Compiler User Guide*.

This application note includes the following sections:

- “Understanding Throughput in PCI Express” on page 1
- “Deliverables Included with the Reference Design” on page 6
- “Reference Design Functional Description” on page 6
- “Design Walkthrough” on page 10
- “Performance Benchmarking Results” on page 17

## Understanding Throughput in PCI Express

The throughput in a PCI Express system depends on several factors, including protocol overhead, payload size, completion latency, and flow control update latency. The throughput also depends on the characteristics of the devices that form the link. This section discusses the various factors that you must consider when analyzing throughput. This example assumes an  $\times 1$  link operating at 2.5 Gbps. The same theory applies to a Gen2 link running at 5.0 Gbps.

### Protocol Overhead

PCI Express uses 8b/10b encoding, in which every byte of data is converted into a 10-bit data code, resulting in a 25% overhead. The effective data rate is therefore reduced to 2 Gbps or 250 MBps per lane.

An active link also transmits Data Link Layer Packets (DLLPs) and Physical Layer Packets (PLPs). The PLPs are four bytes or one dword in size and consist of SKP ordered sets. The DLLPs are two dwords in size and consist of the ACK/NAK and flow control DLLPs. The ACKs and flow control update DLLPs are transmitted in the opposite direction from the Transaction Layer Packet (TLP). In cases where the link is transmitting and receiving high bandwidth traffic, the DLLP activity can be significant and on the order of one DLLP for every TLP. The DLLPs and PLPs reduce the effective bandwidth available for TLPs. The format of the TLP is shown in [Figure 1](#).

**Figure 1.** TLP Format

Start	SequenceID	TLP Header	Data Payload	ECRC	LCRC	End
1 Byte	2 Bytes	3-4 DW	0-1024 DW	1 DW	1 DW	1 Byte

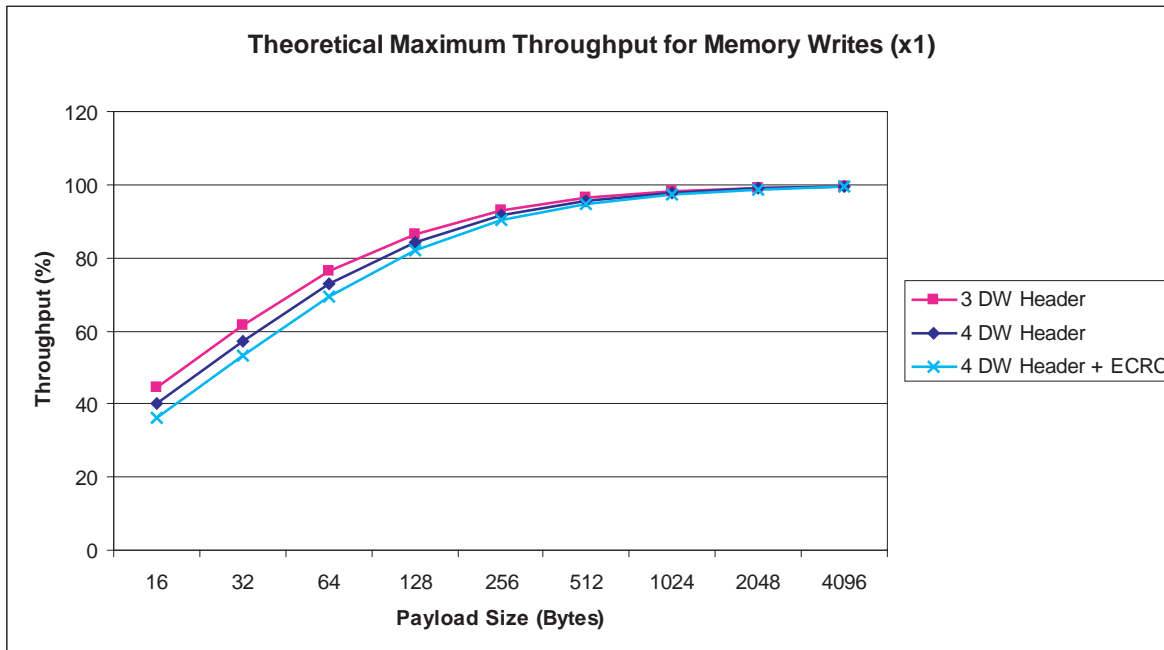
The overhead associated with a single TLP varies between five and seven dwords if the optional ECRC is not included. The overhead includes the Start and End framing symbols, the Sequence ID, a TLP header that is three or four dwords long, and the link cyclic redundancy check (LCRC). The TLP header size depends on the TLP type and can change from one TLP to another. The rest of the TLP contains 0–1024 dwords of data payload.

## Throughput for Posted Writes

The theoretical maximum throughput is calculated using the following formula:

$$\text{Throughput \%} = \text{payload size} / (\text{payload size} + \text{overhead})$$

[Figure 2](#) shows the maximum throughput possible with different TLP header sizes and ignores any DLLPs and PLPs. For a 256-byte maximum payload size and a three dword TLP header (or five dword overhead), the maximum possible throughput is  $(256 / (256 + 20))$ , or 92%.

**Figure 2.** Maximum Throughput for Memory Writes

The maximum TLP payload size is controlled by the device control register (bits 7:5) in the PCI Express configuration space. The MegaCore function parameter **maximum payload size** sets the read-only value of the maximum payload size supported field of the device capabilities register (bits 2:0) and optimizes the MegaCore function for this payload size. You can configure the MegaCore function for a maximum payload size, which can then be reduced by the system based on the maximum payload size supported by the system. This MegaCore function parameter affects the resource utilization, so the parameter must be set such that it is not greater than the maximum payload size supported by the system in which the MegaCore function is used.

PCI Express uses flow control, in which a TLP is not transmitted unless the receiver has enough free buffer space to accept that TLP. A device needs sufficient header and data credits before sending a TLP. When the application logic in the completer accepts the TLP, it frees up the Rx buffer space in the completer's transaction layer. The completer sends a flow control update (FC Update DLLP) that returns the credits consumed by the originating TLP. When the device uses up all of its initial credits, the link bandwidth is limited by how fast it receives credit updates. Flow control updates depend on the maximum payload size and the latencies in the transmitting and receiving devices.



For more information about the flow control update loop, refer to the *PCI Express Compiler User Guide*. This user guide provides detailed information about the flow control update loop and the associated latencies.

## Throughput for Reads

PCI Express uses a split-transaction for reads. A requester first sends a memory read request. The completer then sends an ACK DLLP to acknowledge the memory read request. It subsequently returns a completion data that can be split into multiple completion packets. Read throughput is somewhat lower than write throughput because the data for the read completions may be split into multiple packets rather than being returned in a single packet. The following example illustrates this point. Assuming a read request for 512 bytes and a completion packet size of 256 bytes, the maximum possible throughput can be calculated as follows:

$$\text{Number of completion packets} = 512/256 = 2$$

$$\text{Overhead for a 3 dword TLP Header with no ECRC} = 2*20 = 40 \text{ bytes}$$

$$\text{Maximum Throughput \%} = 512/(512 + 40) = 92\%.$$

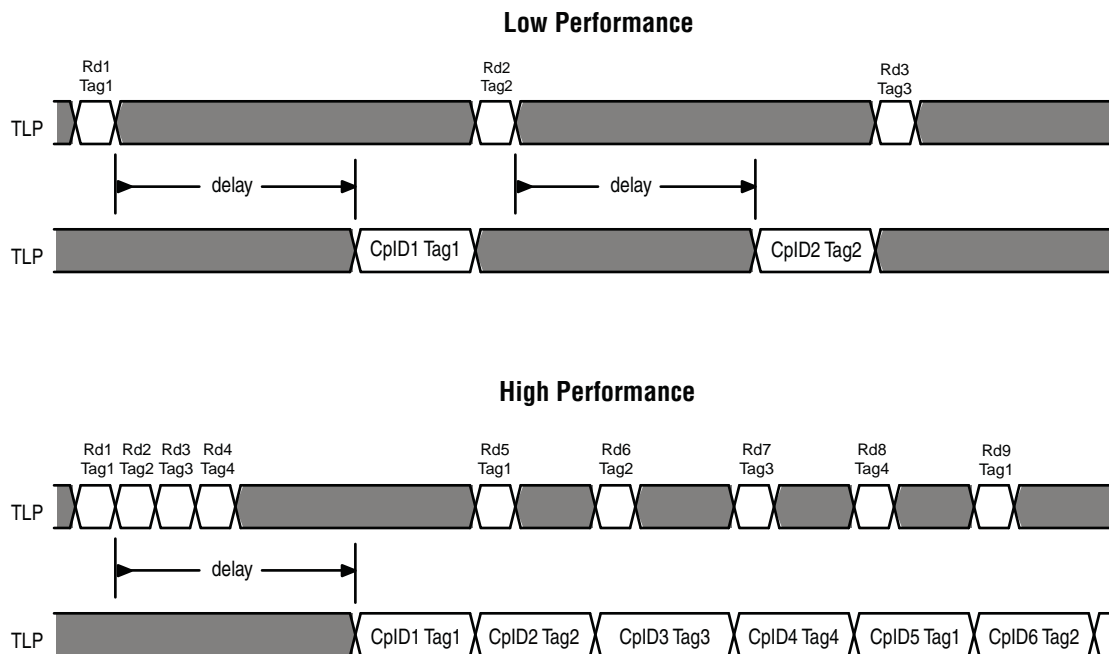
These calculations do not take into account any DLLPs and PLPs. The read completion boundary (RCB) parameter specified by the *PCI Express Base Specification* determines the naturally aligned address boundaries on which a read request may be serviced with multiple completions. For a root complex, the RCB is either 64 bytes or 128 bytes. For all other PCI Express devices, the RCB is 128 bytes.




A non-aligned read request may experience a further throughput reduction.

Read throughput depends on the round-trip delay between the time when the application logic issues a read request and the time when all of the completion data has been returned. To maximize throughput, the application must issue enough read requests and process enough read completions, or offer enough non-posted header credits to cover this delay.

Figure 3 shows timing diagrams for memory read requests (MRd) and completions (Cp1D). The timing diagram in the top part of the figure shows the requester waiting for a completion before making a subsequent read request, resulting in lower throughput. The timing diagram in the bottom part shows the requester making enough memory read requests to eliminate the delay for the completions with the exception of the first read, thus maintaining higher throughput.

**Figure 3.** Read Request Timing

In addition, the requester must maintain maximum throughput for the completion data packets by selecting appropriate settings for completions in the Rx buffer and by managing the rate at which the application logic issues read requests and processes the completion data.

 For more recommendations for the Rx buffer settings, refer to the *PCI Express Compiler User Guide*.

A final constraint on the throughput is the amount of read request data that can be outstanding at one time. This amount is limited by the number of header tags and the maximum read request size that can be issued. The maximum read request size is controlled by the device control register (bits 14:12) in the PCI Express configuration space. The header tag is a number that is assigned to a non-posted request by the application layer to distinguish completions for that request from other requests. The maximum number of tags that the application uses is set in the MegaCore function configuration, because the MegaCore function tracks read requests and completions for error checking. For the soft IP implementation of the MegaCore function, specifying a large number of tags increases the resource utilization and so can decrease performance ( $f_{MAX}$ ) and the throughput of the MegaCore function. A minimum number of tags are required to maintain sustained read throughput. This number is system dependent. On a Windows system, eight tags are usually enough to ensure continuous read completion with no gap for a 4 KByte read request. The high performance system shown in [Figure 3](#) uses 4 tags. The first tag is reused for the fifth read.

## Deliverables Included with the Reference Design

The reference design includes the following components:

- Software application and windows drivers (32-bit, Windows XP)
- FPGA programming files for the Stratix IV GX FPGA Development Kit for ×1, ×4, and ×8 operation
- FPGA programming files for the Arria II GX FPGA Development Kit for ×1, ×4, and ×8 operation
- Quartus® II Archives Files (.qar) for the development boards and configurations, including SRAM Object File (.sof), Programmer Object File (.pof), and SignalTap® II Files (.stp).

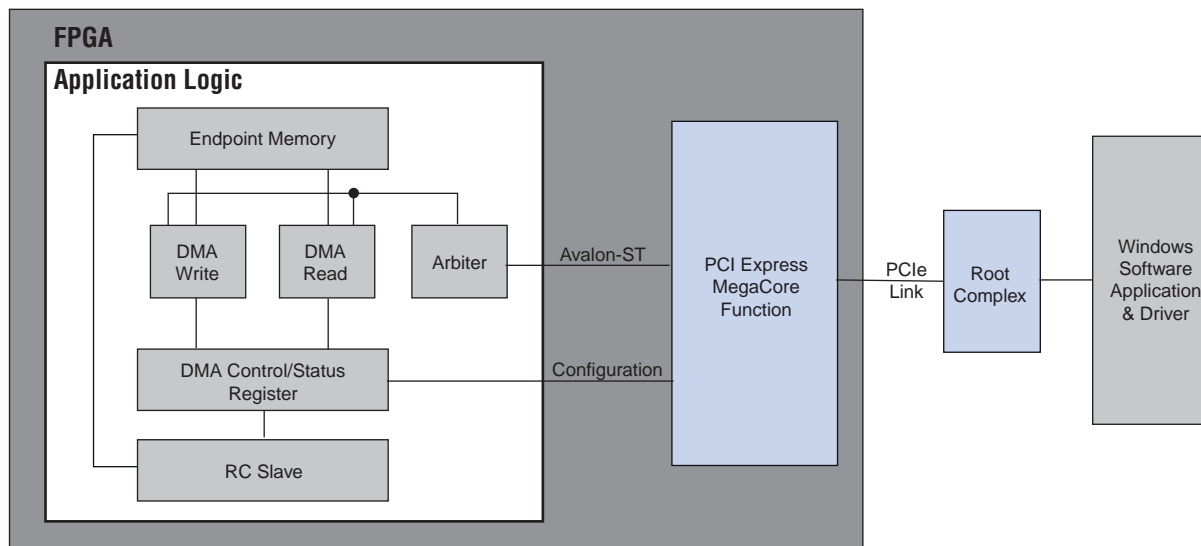
## Reference Design Functional Description

The reference design consists of the following components:

- An application layer that consists of the chaining DMA example generated by the MegaCore function
- The MegaCore function variation
- A software application and Windows XP drivers

These components are shown in [Figure 4](#).

**Figure 4.** Reference Design Components



 For more information about this reference design, refer to the *Testbench and Design Example* chapter of the *PCI Express Compiler User Guide*.

The chaining DMA example consists of two DMA modules in the application logic and an internal memory (referred to as the endpoint memory). The design can support simultaneous DMA read and DMA write transactions. The DMA write module implements write operations in which the data is transferred from the endpoint memory to the root complex (system memory) across the PCI Express link. The DMA read module implements read operations in which data is transferred from the root complex (system memory) across the PCI Express link to the endpoint memory.

The reference design itself is completely contained within the FPGA and relies on no other hardware interface except the PCI Express link. A chaining DMA provides higher performance compared to a simple DMA when transferring a large amount of non-contiguous memory between the system memory and endpoint memory. In a simple DMA, the software application programs the DMA registers for every transfer. The chaining DMA uses descriptor tables for each memory page. These descriptor tables contain the following information:

- Transfer length
- Source and destination addresses for the transfer
- Control information that sets the handshaking behavior between the software application and the DMA module

Each descriptor consists of four dwords. The descriptors are staggered in a contiguous memory page.

Based on the attributes set in the GUI, the software application creates the necessary descriptor tables in the system memory. The software application also creates a descriptor header table that includes such information as the total number of descriptors and the address of the first descriptor table. At the beginning of the transfer, the software application programs the DMA registers with the descriptor header table. The DMA module continuously collects these descriptor tables for each DMA read and DMA write and performs the transfer for each descriptor.

The DMA module also includes a performance counter. The counter starts when the software writes a descriptor header table to the DMA registers and continues counting until the last data has been transferred by the DMA module. After the transfer is complete, the software application uses the counter value to compute the throughput for the transfer and reports it. The counter value includes latency for the initial descriptor read, so the throughput reported by the software application is less than the actual amount.



For more information about the chaining DMA example architecture and programming instructions, refer to the *PCI Express Compiler User Guide*.

### **MegaCore Function Settings**

The MegaCore functions used in the reference design support a maximum payload size of 512 Bytes. The desired performance for received completions and requests is set to **Maximum**.

Table 1 through Table 10 show the parameters used in the reference design, as set in the MegaWizard interface. The tables show the parameters for ×4 operation. The settings for ×1 and ×8 operations are identical except that the number of lanes is set to 1 or 8. Table 1 shows the parameters on the **System Settings** page of the MegaWizard interface.

**Table 1.** System Settings for PCI Express MegaCore Function

Parameter	Value
PCIe MegaCore Function Type	PCI Express hard IP
<b>PCIe System Parameters</b>	
PHY type	Stratix IV GX or Arria II GX
PHY interface	Serial
Configure transceiver block	
Enable fast recovery mode	Turn this option on
Enable rate match FIFO	Turn this option on
Starting channel number	0
Lanes	×4
Xcvr ref_clk	100 MHz
Application Interface	Avalon-ST 64-bit
Port type	Native Endpoint
PCI Express version	2.0
Application clock	250 MHz
Max rate	Gen2 (5.0 Gbps)
Test out width	9 bits
PCIe reconfig	Disable

Table 2 shows the settings for the **PCI Registers** page.

**Table 2.** PCI Register Settings

<b>PCI Base Address Registers (Type 0 Configuration Space)</b>		
BAR	BAR Type	BAR Size
0	32-bit Non-Prefetchable Memory	256 MBytes - 28 bits
1	32-bit Non-Prefetchable Memory	256 KBytes - 18 bits
2	32-bit Non-Prefetchable Memory	256 KBytes - 18 bits
<b>PCI Read-Only Registers</b>		
Register Name	Value	
DeviceID	0x0004	
VendorID	0x1172	
SubsystemID	0x0004	
Subsystem Vendor ID	0x1172	
RevisionId	0x01	
Class Code	0xFF0000	

Table 3 shows the settings for the Capabilities page.

**Table 3.** Capabilities Parameters

<b>Capability Registers</b>	
<b>Device Capabilities</b>	
Tags supported	32
Implement completion timeout disable	Turn this option on
Completion timeout range	ABCD
<b>Error Reporting</b>	
Implement advanced error reporting	Off
Implement ECRC check	Off
Implement ECRC generation	Off
Implement ECRC forwarding	Off
<b>MSI Capabilities</b>	
MSI messages requested	4
MSI message 64-bit address capable	On
<b>Link Capabilities</b>	
Link common clock	On
Data link layer active reporting	Off
Surprise down reporting	Off
Link port number	0x01
<b>Slot Capabilities</b>	
Enable slot capability	Off
Slot capability register	0x00000000
<b>MSI-X Capabilities</b>	
Implement MSI-X	Off
MSI-X Table size	0x000
MSI-X Table Offset	0x00000000
MSI-X Table BAR Indicator (BIR)	1:0
Pending Bit Array (PBA)	
Offset	0x00
BAR Indicator	1:0

Table 4 shows the settings for the Buffer Setup page.

**Table 4.** Buffer Setup Parameters (Part 1 of 2)

<b>Parameter</b>	<b>Value</b>
Maximum payload size	512 KBytes
Number of virtual channels	1
Number of low-priority VCs	None
Auto configure retry buffer size	On

**Table 4.** Buffer Setup Parameters (Part 2 of 2)

Parameter	Value
Retry buffer size	2 KBytes
Maximum retry packets	64
Desired performance for received requests	Maximum
Desired performance for received completions	Maximum

Table 5 shows the setting for the **Power Management** page.

**Table 5.** Power Management Parameters

Parameter	Value
<b>L0s Active State Power Management (ASPM)</b>	
Idle threshold for L0s entry	8,192 ns
Endpoint L0s acceptable latency	< 64 ns
<b>Number of fast training sequences (N_FTS)</b>	
Common clock	Gen2: 255
Separate clock	Gen2: 255
Electrical idle exit (EIE) before FTS	4
<b>L1s Active State Power Management (ASPM)</b>	
Enable L1 ASPM	Off
Endpoint L1 acceptable latency	< 1 $\mu$ s
L1 Exit Latency Common clock	> 64 $\mu$ s
L1 Exit Latency Separate clock	> 64 $\mu$ s



For more information about the MegaCore function parameters, refer to *Chapter 3: Parameters* in the *PCI Express Compiler User Guide*.

### Quartus II Settings

The .qar files in the reference design package has the recommended synthesis, Fitter, and timing analysis settings for the parameters chosen in the variation used in this reference design.

## Design Walkthrough

This section describes how to install the reference design and provides instructions for running the software application. The following information is included:

- “Hardware Requirements”
- “Software Requirements”
- “Software Installation”
- “Hardware Installation”
- “Running the Software Application”

## Hardware Requirements

The reference design requires the following hardware:

- The Stratix IV GX FPGA Development Kit or Arria II GX FPGA Development Kit.
- A computer running 32-bit Windows XP with an  $\times 8/\times 4/\times 1$  PCI Express slot for the Stratix IV GX FPGA or Arria II GX development board. The software application and hardware are installed on this computer, referred to as computer #1 in this document.
- A computer with the Quartus II software for downloading FPGA programming files to the Stratix IV GX FPGA or Arria II GX development board, referred to as computer #2 in this document.
- A USB cable or other Altera download cable.
- A PCI Express  $\times 4$ -to- $\times 1$  lane converter for  $\times 1$  operation.



This reference design uses four lanes. If the PCI Express slot on your motherboard has fewer than four lanes, you must use a lane converter to transfer data from the higher lanes to the lower available lane.

## Software Requirements

To run the reference design application requires installation of the following software:

- Reference design software installed on computer #1.
- PCI Express High Performance Reference design package, available as a downloadable compressed file.



For more information refer to the [PCI Express High-Performance Reference Design](#) product page.

- The Quartus II software version 9.0 SP2 running on computer #2.

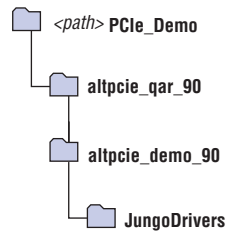
## Software Installation

You must have Administrator privileges to install the software application.

The software application only runs on 32-bit Windows XP and includes the WinDriver software from Jungo as part of the application.

Perform the following steps to install the software application and Windows drivers:

1. Download the PCI Express High Performance reference design package to computer #1, and extract the compressed files. [Figure 5](#) shows the directory structure.

**Figure 5.** Directory Structure

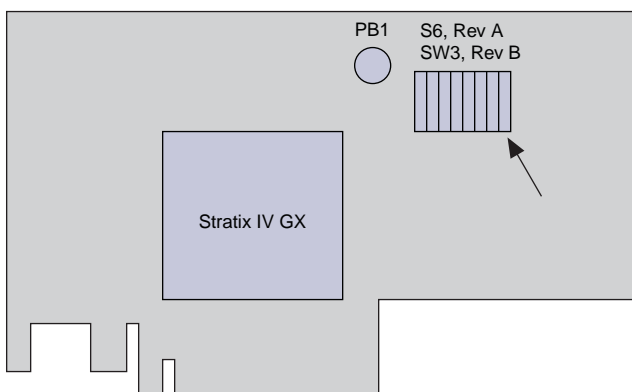
2. Before plugging in the PCI Express card, copy the **altpcie\_demo\_90** directory to computer #1.
3. In the **JungoDrivers** directory, double-click on **install.bat** to install the Windows XP driver for this application.
4. Run **altpcie\_demo.exe** from the **altpcie\_demo\_90** directory to run the software application.

## Hardware Installation

If you are using the Stratix IV GX card, you must check the settings on an eight-position dip switch which controls the PCI Express mode of operation. [Figure 6](#) highlights this component. The right-most position of this dip switch is used to change between normal operation and PCI Express compliance base board (CBB) testing. To run the software included in this application note, this switch must be in the **off** position. When it is in the **on** position, you can use the reset switch labeled PB1 to cycle through various modes required for CBB testing. (The dip switch labels the **on** side on the switch.)



The top-level RTL file has been modified to enable CBB testing. If you regenerate the MegaCore function, you may overwrite this top-level file and disable the CBB testing capability.

**Figure 6.** Location of Components that Control PCI Express Mode of Operation

Perform the following steps to install the hardware.

1. Power down computer #1 and plug the development board into the PCI Express slot. For an  $\times 1$  operation, use a PCI Express lane converter.
2. The Stratix IV GX FPGA and Arria II GX Development Kits includes integrated USB-Blaster circuitry for FPGA programming. However, for the host computer and development board to communicate, you must install the USB-Blaster on the host computer.

To download the USB-Blaster driver, go to the Altera support site at [www.altera.com/support/software/drivers/dri-index.html](http://www.altera.com/support/software/drivers/dri-index.html). For installation instructions, go to [www.altera.com/support/software/drivers/usb-blaster/dri-usb-blaster-xp.html](http://www.altera.com/support/software/drivers/usb-blaster/dri-usb-blaster-xp.html).

3. Program the FPGA with the reference design using the Quartus II software on computer #2 and an Altera USB-Blaster cable (or other download cable) connection between computer #2 and the development board on computer #1.

### Connecting the USB-Blaster Cable

To connect the USB-Blaster cable, perform the following steps:

1. Connect one end of the USB cable the USB port (J7) on the development board.
2. Connect the other end of the cable to the USB port on the computer running the Quartus II software on computer #2.

### Programming with the .sof File

To program the board with the .sof file provided, interrupt the boot sequence on computer #1 to bring up the BIOS System Setup interface. (Pressing the F2 key interrupts the boot sequence on many Windows PCs.)

Perform the following steps to program the FPGA with the .sof file:

1. Start the Quartus II programmer on computer #2.
2. Click **Hardware Setup** and select the **USB Blaster**. Click **Close**.
3. In the Quartus II Programmer, click **Auto Detect** to list the devices attached to the JTAG chain on the development board.
4. Right-click the Stratix IV GX device (EP4SGX230) or Arria II GX (EP2AGX125) and click **Change File**. Select the path to the appropriate .sof file.
5. Turn on the **Program/Configure** option for the added file.
6. Click **Start** to download the selected file to the Stratix IV GX or Arria II GX device. The device is configured when the **Progress** bar reaches 100%.
7. On computer #1 exit the BIOS System Setup or boot manager interface.
8. On computer #1, press Ctl-Alt-Delete to perform a soft reboot.
9. The operating system detects a new hardware device and displays the Found New Hardware Wizard. In the wizard, select **Install the software automatically (Recommended)**. Click **Next**.
10. Click **Finish** to close the wizard.

## Running the Software Application

Perform the following steps to run the software application:

1. Double-click on the application **altpcie\_demo.exe** in the **altpcie\_demo\_90** directory.
2. The application reports the board type, the number of active lanes, the maximum read request size, and the maximum payload size.

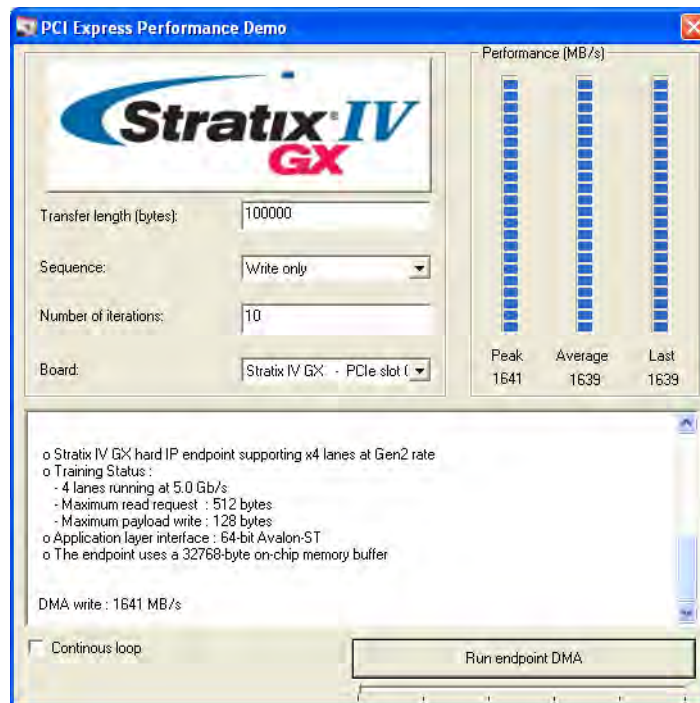
The software GUI has the following control fields:

- **Transfer length**—Specifies the transfer length in bytes
  - **Sequence**—Controls the sequence for data transfer or addressing
  - **Number of iterations**—Controls the number of iterations for the data transfer
  - **Board**—Specifies the development board for the software application
  - **Continuous loop**—When this option is turned on, the application performs the transfer continuously
3. Set the **Transfer length** to 100,000 bytes and the **Sequence** to **Write only**, Click **Run**.

When set for **Write only**, the software programs the DMA registers in the FPGA to transfer data from the FPGA to the system memory in chunks of 100,000 bytes. The performance bars report the peak, average, and last throughput. The average throughput is computed across all the iterations.

4. You can use the GUI to change the **Transfer length** and **Sequence** and repeat the test.

**Figure 7** illustrates the results of the **Write only** DMA test using the Stratix IV GX device.

**Figure 7.** Write-Only Options

In addition to the parameter settings to control the chaining DMA, the GUI includes five additional commands that you can use to obtain configuration information about the device and board and to perform root port reads and writes. Table 6 outlines all of the available commands. The position of the slider control changes the command.

**Table 6.** PCI Express Performance Demo GUI Commands and Options (Part 1 of 2)

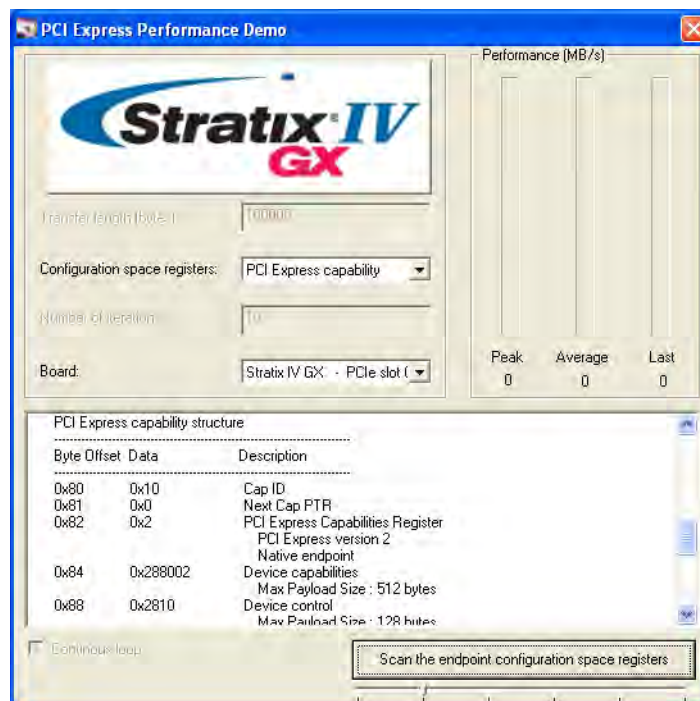
Command	Options	Description
Run endpoint DMA (Figure 7)	Write only Read only Read then write Write then read Read and write	Writes transfer data from the FPGA to system memory. Reads transfer data from system memory to the FPGA.
Scan the endpoint configuration space registers (Figure 8)	Type 0 Configuration PCI Express capability MSI capability Power management capability	Reports the byte address offset, value and a description of the selected register set.
Scan the current PCI Express board settings (Figure 9)	—	Reports the configuration settings for the development board.
Scan the motherboard PCI bus	—	Reports the vendor ID, device ID, slot, bus, and function numbers for all devices on the motherboard's PCI bus.

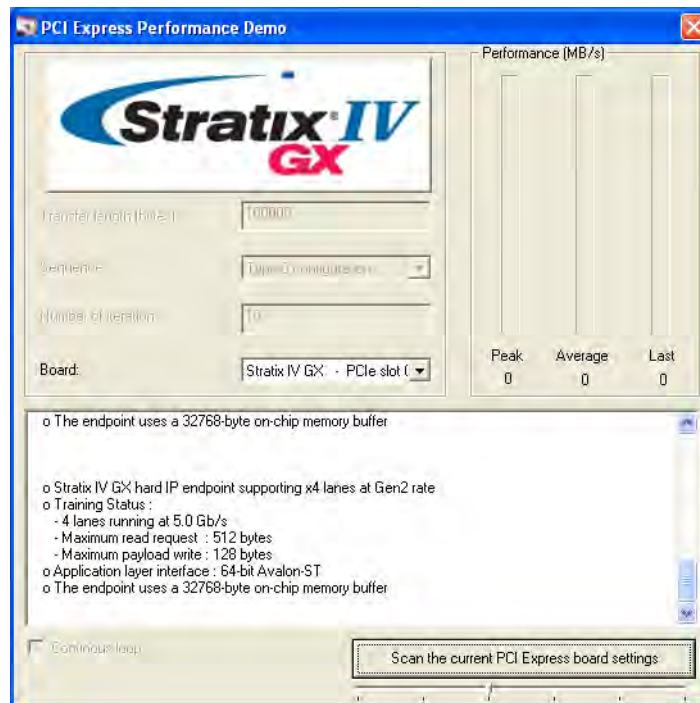
**Table 6.** PCI Express Performance Demo GUI Commands and Options (Part 2 of 2)

Command	Options	Description
Run root port memory read	At endpoint address From 0x0 to endpoint address	<p>Programs the root port of the motherboard's PCI Express chipset to read from the FPGA's memory, as follows:</p> <ul style="list-style-type: none"> <li>If you select <b>At endpoint address</b>, you can type the starting read address in endpoint memory in the <b>Endpoint address</b> field.</li> <li>If you select <b>From 0x0 to endpoint address</b>, the <b>Endpoint address</b> field specifies the transfer length.</li> </ul>
Run root port memory write	At endpoint address From 0x0 to endpoint address	<p>Programs the root port of the motherboard's PCI Express chipset to write to the FPGA's memory, as follows:</p> <ul style="list-style-type: none"> <li>If you select <b>At endpoint address</b>, you can type the starting write address in endpoint memory the <b>Endpoint address</b> field.</li> <li>If you select <b>From 0x0 to endpoint address</b>, the <b>Endpoint address</b> field specifies the transfer length.</li> </ul>

Figure 8 illustrates the output of the **Scan the Endpoint Configuration Space Registers** command. Figure 9 illustrates the output of the **Scan the current PCI Express board settings** command using the Stratix IV GX device.

**Figure 8.** Scan the Endpoint Configuration Space Registers



**Figure 9.** Scan the current PCI Express board settings

## SignalTap II Files

The reference design package also includes **.stp** files. The SignalTap II file can provide information on the performance of this design. The SignalTap II file includes the key signals from the application logic. The `init` signal in the DMA read and write modules transitions to zero at the beginning of the transfer. You can use the `init` signal as a trigger in the SignalTap II file to capture data.

The `tx_st_ready0` and `rx_st_valid0` are indications of link utilization and throughput. In the transmit direction, the frequent deassertion of the `tx_st_ready0` signal typically indicates that the MegaCore function is not receiving enough credits from the device at the far end of the PCI Express link. It could also indicate that a  $\times 4$  link has trained to  $\times 1$ . In the receive direction, the deassertion of `rx_st_valid0` indicates that the MegaCore function is not receiving enough data.

## Performance Benchmarking Results

[Table 7](#) gives the performance of the performance of  $\times 8$ ,  $\times 4$ , and  $\times 1$  operations with the Stratix IV GX FPGA development board for the Intel X58 Chipset using this reference design. The table shows the average throughput for a transfer size of 100 KBytes and 20 iterations with a maximum write payload size of 256 bytes, a maximum read request size of 512 bytes, a read completion size of 256 bytes using a clock multiplier unit (CMU) clock.

**Table 7.** Stratix IV GX Performance - Intel X58 Chipset

Configuration	DMA Reads (MB/s)	DMA Writes (MB/s)	Simultaneous DMA Read/Writes (MB/s)	Theoretical Maximum Throughput (MB/s)	
				Read	Write
<b>Hard IP Implementation—Stratix IV GX</b>					
Gen2 ×8, 128-bit	3304	3434	2956/2955	3710	3710
Gen2 ×4, 64-bit	1727	1775	1691/1631	1855	1855
Gen2 ×4, 128-bit	1708	1783	1684/1484	1855	1855
Gen2 ×1	448	450	438/425	463	463
Gen1 ×8, 64-bit	1706	1778	1680/1628	1855	1855
Gen1 ×8, 128-bit	1694	1778	1678/1480	1855	1855
Gen1 ×4	875	890	855/815	927	927
Gen1 ×1	224	225	219/211	231	231
<b>Soft IP Implementation—Stratix IV GX</b>					
Gen1 ×4	873	890	854/811	927	927
Gen1 ×1	222	225	219/209	231	231

**Table 8** gives the performance of the performance of ×1, ×4, and ×8 operations with the Arria II GX FPGA development board for the Intel x58 Chipset using this reference design. The table shows the average throughput for a transfer size of 100 KBytes and 20 iterations with a maximum write payload size of 256 bytes, a maximum read request size of 512 bytes, with the Avalon-ST interface using a clock multiplier unit (CMU) clock.

**Table 8.** Arria II GX Performance - Intel X58 Chipset

Configuration	DMA Reads (MB/s)	DMA Writes (MB/s)	Simultaneous DMA Read/Writes (MB/s)	Theoretical Maximum Throughput (MB/s)	
				Read	Write
<b>Hard IP Implementation—Arria II GX</b>					
Gen1 ×8, 128-bit	1497	1775	1210/1331	1855	1855
Gen1 ×4, 64-bit	859	889	725/767	927	927
Gen1 ×1, 64-bit	220	225	217/204	231	231
<b>Soft IP Implementation—Arria II GX</b>					
Gen1 ×4, 64-bit	860	887	854/780	927	927
Gen1 ×1, 64-bit	220	225	203/203	231	231

**Table 9** gives the performance of the performance of ×4 and ×1 operations with the Arria GX FPGA development board for the Intel x58 Chipset using this reference design. The table shows the average throughput for a transfer size of 100 KBytes and 20 iterations with a maximum write payload size of 256 bytes, a maximum read request size of 512 bytes, with the Avalon-ST interface using a clock multiplier unit (CMU) clock.

**Table 9.** Arria GX Performance (Soft IP Implementation) - Intel X58 Chipset

Configuration	DMA Reads (MB/s)	DMA Writes (MB/s)	Simultaneous DMA Read/Writes (MB/s)	Theoretical Maximum Throughput (MB/s)	
				Read	Write
Gen1 ×4	866	889	855/784	927	927
Gen1 ×1	221	255	203/203	231	231

Table 10 shows the performance of the ×8, ×4, and ×1 operations with the PCI Express Development Kit, Stratix II GX Edition, for three chipsets from Intel, Dell, and NVIDIA using this reference design. The tables show the average throughput for a transfer size of 100 KBytes and 20 iterations. The Intel used a maximum payload of 256 bytes, a maximum read request of 512 bytes with the Avalon-ST interface. The Dell 490 testing used a maximum write payload 256 bytes and the read completion of 64 bytes. The NVIDIA testing used a maximum write payload of 128 bytes and a read completion of 64 bytes.

**Table 10.** Stratix II GX Performance (Soft IP Implementation)

Configuration and Platform	DMA Reads (MB/s)	DMA Writes (MB/s)	Simultaneous DMA Read/Writes (MB/s) (1)	Theoretical Maximum Throughput (MB/s)	
				Read	Write
Gen1 ×8, Intel X58 Chipset	1699	1775	1675/1627	1855	1855
Gen1 ×4, Intel X58 Chipset	873	890	855/816	927	927
Gen1 ×1, Intel X58 Chipset	222	225	219/209	231	231
Gen1 ×8, Dell 490 5000X	1295	1771	1287/1652	1523	1855
Gen1 ×8, NVIDIA CK804	1301	1632	1302/1400	1523	1729
Gen1 ×4, Dell 490 5000X	654	847	645/828	761	927
Gen1 ×4, NVIDIA CK804	654	819	652/757	761	864
Gen1 ×1, Dell 490 5000X	162	224	155/213	190	231
Gen1 ×1, NVIDIA CK804	185	207	177/199	190	216

**Note to Table 10:**

- (1) The simultaneous DMA read/write operation is very sensitive to the system (PC) memory controller. The measurements were obtained using four DIMMs.

Table 11 shows a comparison of the throughput measured with the Lecroy PCI Express analyzer and the software application for the Stratix II GX device (×8 operation) and 100 K Bytes transfer on a Dell 490. The numbers reported by the software application and the Lecroy PCI Express analyzer match very closely.

**Table 11.** Comparison with Lecroy PCI Express Analyzer

Operation	CATC Measurement (MB/s)	Altera Software Application Measurement (MB/s)
DMA Write	1799	1771
DMA Read	1316	1295

## References

*PCI Express Compiler User Guide*

*PCI Express High-Performance Reference*

*PCI Express Specifications*

*WinDriver information*

## Document Revision History

Table 12 shows the revision history for this application note.

**Table 12.** Document Revision History

Date and Document Version	Changes Made	Summary of Changes
August 2009, v1.2	<ul style="list-style-type: none"> <li>■ Updated to show the performance of the Arria II GX (EP2AGX125) device running on the Arria II GX FPGA Development Kit.</li> </ul>	Updated to show performance of Arria II GX device.
May 2009, v1.1	<ul style="list-style-type: none"> <li>■ Updated to show the performance of the Stratix IV GX (EP4SGX230) device running on the Stratix IV GX FPGA Development Kit.</li> <li>■ Added new commands to the PCI Express Performance Demo GUI.</li> </ul>	Updated for Stratix IV GX device and Arria GX. Also updated performance number for Stratix II GX device with the Intel X58 Chipset.
May 2007, v1.0	Initial release.	—

s



101 Innovation Drive  
San Jose, CA 95134  
[www.altera.com](http://www.altera.com)  
Technical Support  
[www.altera.com/support](http://www.altera.com/support)

Copyright © 2009 Altera Corporation. All rights reserved. Altera, The Programmable Solutions Company, the stylized Altera logo, specific device designations, and all other words and logos that are identified as trademarks and/or service marks are, unless noted otherwise, the trademarks and service marks of Altera Corporation in the U.S. and other countries. All other product or service names are the property of their respective holders. Altera products are protected under numerous U.S. and foreign patents and pending applications, maskwork rights, and copyrights. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera Corporation. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.



I.S. EN ISO 9001