PCI Express High Performance Reference Design

2017.04.20

AN-456-2.5 Subscribe Send Feedback

The *PCI Express High-Performance Reference Design* highlights the performance of the Altera's PCI Express® products. The design includes a high-performance chaining direct memory access (DMA) that transfers data between the a PCIe Endpoint in the FPGA, internal memory and the system memory. The reference design includes a Windows-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 PCI Express protocol in the following devices:

- Arria II GX
- Arria V
- Arria 10
- Cyclone IV GX
- Cyclone V
- Stratix IV GX
- Stratix V

Altera offers the IP Compiler for PCI Express IP core in both hard IP and soft IP implementations, and the Arria V, Arria 10, Cyclone V, and Stratix V Hard IP for PCI Express in hard IP. The hard IP implementation is available as a Root Port or Endpoint. Depending on the device used, the hard IP implementation is compliant with *PCI Express Base Specification 1.1, 2.0,* or *3.0.* The soft IP implementation is available only as an Endpoint. It is compliant with *PCI Express Base Specification 1.0a* or *1.1.*

Related Information

PCI Express Base Specification 1.1, 2.0, or 3.0.

Understanding Throughput in PCI Express

The throughput in a PCI Express system depends on the following factors:

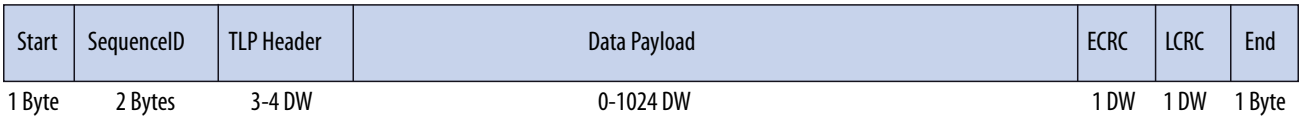
- Protocol overhead
- Payload size
- Completion latency
- Flow control update latency
- Characteristics of the devices that form the link

Protocol Overhead

PCI Express Gen1 and Gen2 IP cores use 8B/10B encoding. Each 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 and consist of SKP Ordered Sets. The DLLPs are two dwords 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). If link is transmitting and receiving high bandwidth traffic, the DLLP activity can be significant. The DLLPs and PLPs reduce the effective bandwidth available for TLPs. The format of the TLP illustrates that the overhead if a TLP is seven dwords. The overhead is five dwords if the optional ECRC is not included.

Figure 1: TLP Format



The overhead includes the following fields:

- Start and End framing symbols
- A Sequence ID
- A TLP header that is three or four dwords long,
- The link cyclic redundancy check (LCRC).

The rest of the TLP contains 0–1024 dwords of data payload.

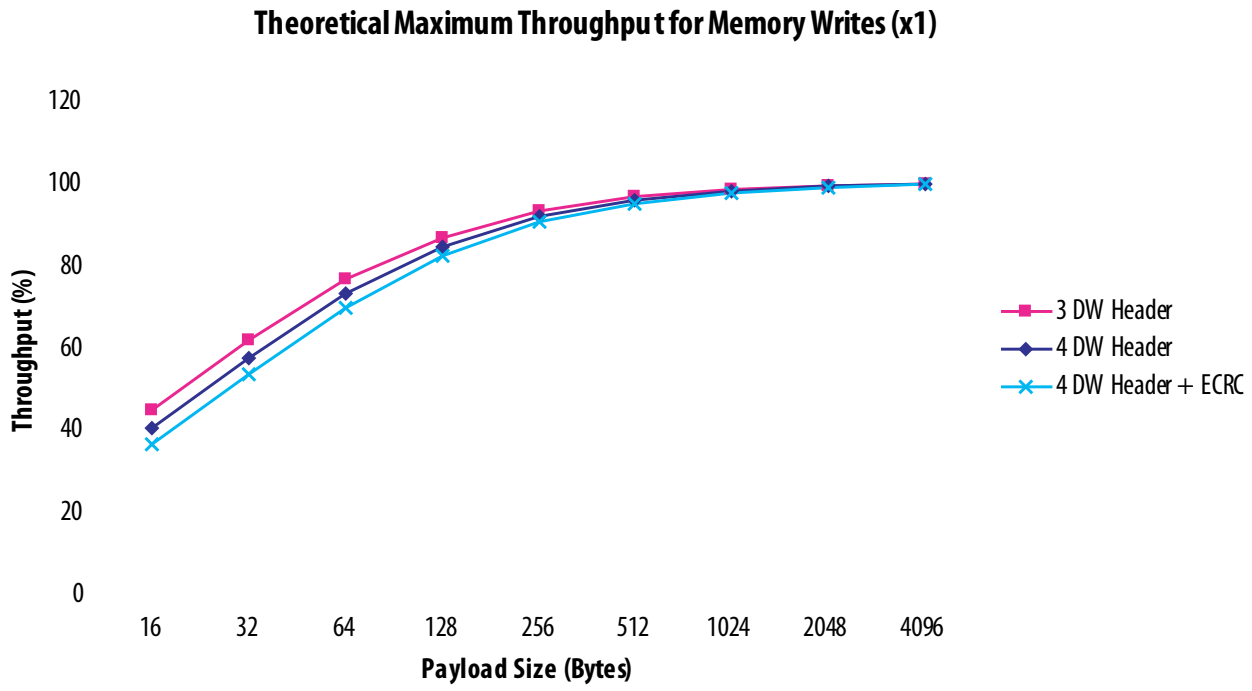
Throughput for Posted Writes

The theoretical maximum throughput is calculated using the following formula:

```
Throughput % = payload size / (payload size + overhead)
```

The following figure 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 `device control` register (bits 7:5) in the PCI Express Configuration Space specifies maximum TLP payload size. The parameter **maximum payload size** sets the read-only value of the `Maximum Payload Size Supported` field of the `Device Capabilities` register (bits 2:0). The payload size you specify for your variant may be reduced based on the system maximum payload size. This **maximum payload size** parameter affects the resource utilization. To maximize resources, do not specify a the maximum payload size that is greater than the system maximum payload size.

PCI Express uses flow control. A TLPs is not transmitted unless the receiver has enough free buffer space to accept it. Header and data credits track available buffer space. When the application in the completer accepts the TLP, it frees 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. After the device uses all of its initial credits, 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.

Related Information

Stratix V Avalon-ST Interface for PCIe Solutions User Guide

For more information about the flow control update loop and the associated latencies, refer to the *Throughput Optimization* chapter. The information in the *Throughput Optimization* chapter is not specific to a particular device.

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. This example uses a read request for 512 bytes and a completion packet size of 256 bytes. The maximum possible throughput is calculated as follows:

Number of completion packets = 512/256 = 2

Overhead for a 3 dword TLP Header with no ECRC = 2*20 = 40 bytes

Maximum Throughput % = 512/(512 + 40) = 92%.

These calculations do not take into account any DLLPs and PLPs. The *PCI Express Base Specification* defines a read completion boundary (RCB) parameter. The RCB parameter 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.

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

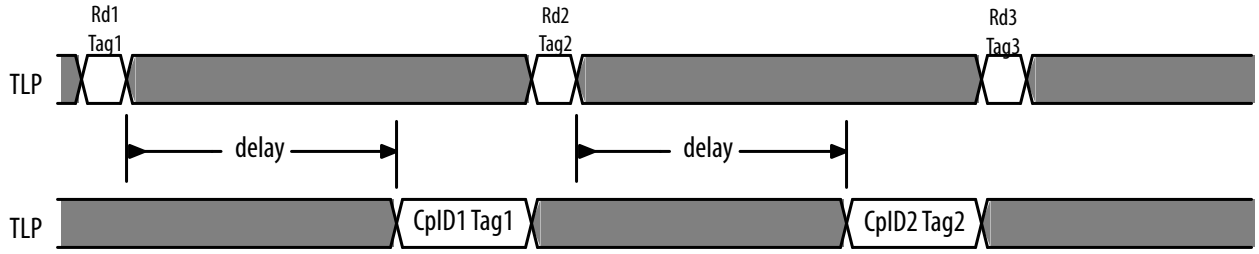
Read throughput depends on the round-trip delay between the following two times:

- The time when the application logic issues a read request
- 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, the application must issue enough non-posted header credits to cover this delay.

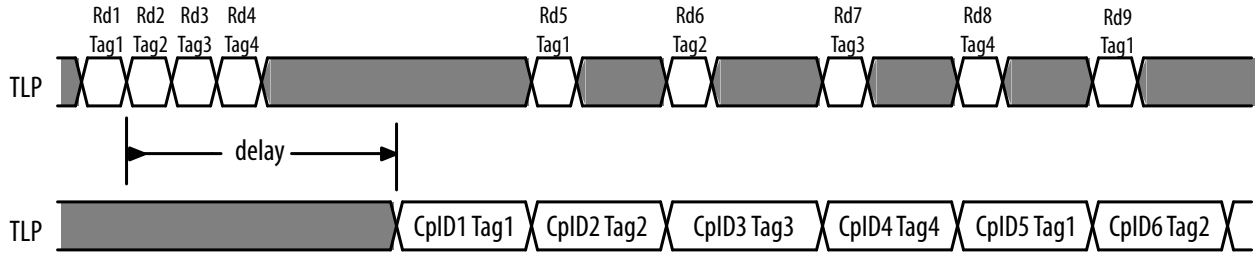
The following figure shows timing diagram for memory read requests (`MRd`) and completions (`CplD`). The requester waits for a completion before making a subsequent read request, resulting in lower throughput.

Figure 3: Low Performance Reads Request Timing Diagram



The following timing diagram eliminates the delay for completions with the exception of the first read. This strategy maintains a high throughput.

Figure 4: High Performance Read Request TimingDiagram



The requester must maintain maximum throughput for the completion data packets by selecting appropriate settings for completions in the RX buffer. All versions of Altera's PCIe IP cores offer five

settings for the **RX Buffer credit allocation performance for requests** parameter. This parameter specifies the distribution of flow control header, data, and completion credits in the RX buffer. You should use this parameter to allocate credits to optimize for the anticipated workload.

A final constraint on the throughput is the number of outstanding read requests supported. The outstanding requests are limited by the number of header tags and the maximum read request size. The maximum read request size is controlled by the `device control` register (bits 14:12) in the PCIe Configuration Space. The Application Layer assign header tags to non-posted requests to identify completions data. The **Number of tags supported** parameter specifies number of tags available. 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 Request Timing Diagram* uses 4 tags. The first tag is reused for the fifth read.

Related Information

PCI Express Base Specification.
For more information the read completion boundary.

Deliverables Included with the Reference Design

The reference design includes the following components:

- Software application and Windows driver configured specifically for this reference design
- FPGA programming files for the Arria II GX FPGA Development Kit for x1, x4, and x8 Gen1 operation
- FPGA programming files for the Cyclone IV GX FPGA Development Kit for x1 and x4 Gen1 operation
- FPGA programming files for the Stratix IV GX FPGA Development Kit for x1, x4, and x8 Gen1 and Gen2 operation
- FPGA programming files for Arria V GT FPGA Development Kit for x1 and x4 Gen1 and Gen2 operation. Also included are x8 Gen1 programming files
- FPGA programming files for Cyclone V GT FPGA Development Kit for x1 and x4 Gen1 operation
- FPGA programming files for Stratix V GX FPGA Development Kit for x1, x4 and x8 Gen1 and Gen2 operation. Also x1 and x4 Gen3 andx1 programming files are included
- FPGA programming files for Arria 10 GX FPGA Development Kit for x1 Gen1, x8 Gen2, and x4 Gen3
- 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`)

Related Information

- **Arria 10 Avalon-ST Interface for PCIe Solutions user Guide**
  For more information about this example design in Arria 10 devices
- **Arria V Avalon-ST Interface for PCIe Solutions User Guide**
  For more information about this example design in Arria V devices
- **Arria V GZ Avalon-ST Interface for PCIe Solutions User Guide**
  For more information about this example design in Arria V GZ devices
- **Cyclone V Avalon-ST Interface for PCIe Solutions User Guide**
  For more information about this example design in Cyclone V devices
- **Stratix V Avalon-ST Interface for PCIe Solutions User Guide**
  For more information about this example design in Stratix V devices

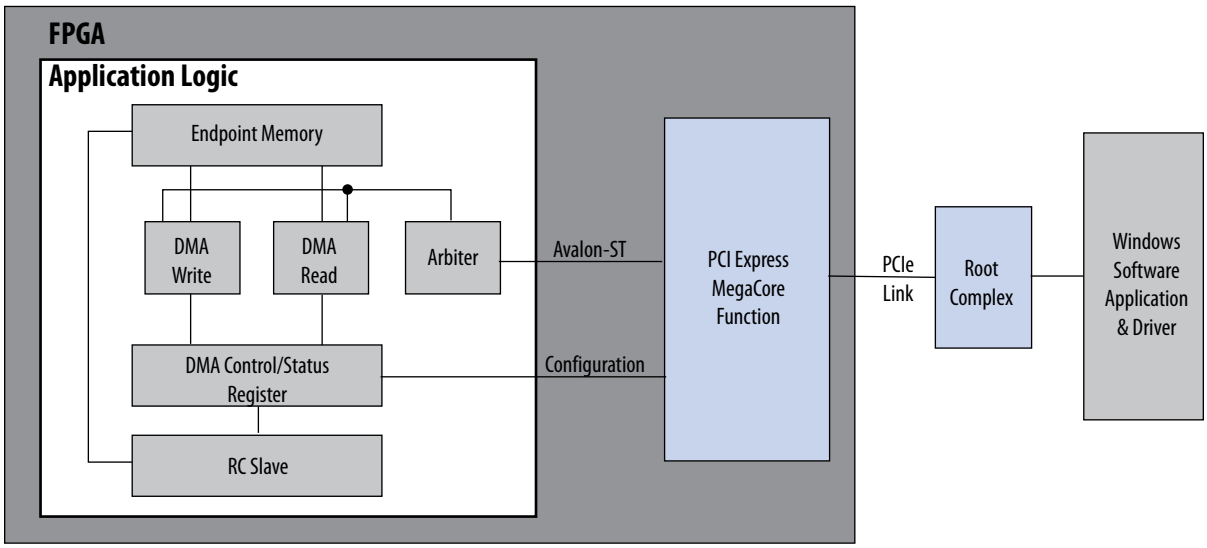
- **IP Compiler for PCI Express User Guide**
  For earlier device families

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 IP core
- The IP core variation
- A software application and Windows driver configured specifically for this reference design

Figure 5: Reference Design Components



The chaining DMA example consists of two DMA modules in the application logic and an internal Endpoint memory. The design supports simultaneous DMA read and DMA write transactions. The DMA write module transfers data from Endpoint memory to the Root Complex system memory across the PCIe link. The DMA read module implements read transfers data from the Root Complex system memory across the PCIe link to Endpoint memory.

The reference design is included the FPGA and relies on no other hardware interface except the PCIe link. A chaining DMA provides higher performance than a simple DMA for non-contiguous memory transfers between the system and Endpoint memory. For 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 stored in a contiguous memory page.

Based on the attributes set in the Parameter Editor, the software application creates the necessary descriptor tables in the system memory. The software application also creates a descriptor header table.

This table specifies 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 write and performs the transfers specified.

The DMA module also includes a performance counter. The counter starts when the software writes a descriptor header table to the DMA registers. It 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. Consequently, the throughput reported by the software application is less than the actual throughput.

File Naming Conventions

This reference design is available in many different configurations as shown in the following table. The file name for each configuration is created by concatenating the following variables:

<type>_<device_family>_<data_rate>_x<lanes>_<user_interface><datapath_width>

Table 1: Naming Conventions for Reference Design Variations

Variable	Abbreviation	Value
Type	hip	Hard IP implementation
	sip	Soft IP implementation
Device_family	civgx	Cyclone IV GX
	cvgt	Cyclone V GT
	cvgx	Cyclone V GX
	aiigx	Arria II GX
	avgt	Arria V GT
	a10gx	Arria 10 GX
	sivgx	Stratix IV GX
	svgx	Stratix V GX
Data_rate	g1	Gen1
	g2	Gen2
	g3	Gen3
Lanes	N/A	1, 4, or 8
User_interface	avst	Avalon® Streaming (Avalon-ST)
Datapath_width	64	64 bit interface to the Application Layer
	128	128 bit interface to the Application Layer

For example, the filename **hip_sivgx_g2_x8_avst128** specifies a reference design for the following configuration:

- Hard IP implementation
- Stratix IV GX device
- Gen2
- Eight lanes
- Avalon-ST interface with a 128 interface to the Application Layer.

Project Hierarchy

The directory structure used for the Arria V, Cyclone V, and Stratix V reference design differs from the earlier device families.

Arria V, Cyclone V, and Stratix V Directory Structure

Arria V, Cyclone V, and Stratix V devices use the following directory structure:

- **top**—the project directory. The top-level entity is **top_example_chaining_top**.
- **pcie_lib**—includes all design files. If you modify the design, you must copy the modified files to the **pcie_lib** directory before recompiling the design.

Arria II GX, Cyclone IV GX, and Stratix IV GX Directory Structure

These devices use the following directory structure:

- **top** or **top_**<*n*>**gx**—**top** is the top-level project directory for the hard IP implementation. The soft IP implementation may have a <*n*>**gx** suffix where <*n*> indicates the number of lanes. In both cases, the top-level entity is **top_example_chaining_top**.
- **top_examples/chaining_dma**—includes design files to implement the chaining DMA.
- **ip_compiler_for_pci_express**—includes library files for PCI Express.

IP Core Settings

The reference design supports a maximum payload size of 512 Bytes. The desired performance for received completions and requests is set to **Maximum**. The following tables show the settings for supported devices.

Table 2: System Settings for PCI Express IP Core–Stratix V GX Device

Parameter	Value
PCIe IP core type	**PCI Express hard IP**
PCIe System Parameters	
PHY type	**Stratix V GX**
PHY interface	**Serial**
System Settings	
Number of lanes	**x8**
Lane rate	**Gen2 (5.0 Gbps)**
Port type	**Native endpoint**
PCI Express Base Specification version	**2.1**

Parameter	Value
Application interface	Avalon-ST 128-bit
RX buffer credit allocation	Low
Reference clock frequency	100 MHz
Use 62.5 MHz application clock	OFF
Use deprecated RX Avalon-ST data byte enable port(rx_st_be)	ON
Enable byte parity ports on Avalon-ST interface	OFF
Enable multiple packets per cycle	OFF
Enable configuration via the PCIe link	OFF
Use credit consumed selection port tx_cons_cred_sel	OFF
Enable Configuration Bypass	OFF
Enable Hard IP reconfiguration	OFF

Table 3: PCI Register Settings–Stratix V GX Device

PCI Base Address Registers (Type 0 Configuration Space)		
BAR	BAR Type	BAR Size
0	32-bit Non-Prefetchable Memory	256 MBytes - 28 bits
1	Disabled	N/A
2	32-bit Non-Prefetchable Memory	1 KBytes - 10 bits
Base and Limit Registers for Root Ports		
Input/Output	Disabled	
Prefetchable memory	Disabled	
PCI Read-Only Registers		
Register Name	Value	Additional Information
Vendor ID	0x1172	The Vendor ID can be either 0x1172 or 0xB0D8. This parameter has no effect on design behavior.
Device ID	0xE001	N/A
Revision ID	0x1	N/A
Class Code	0x00FF0000	N/A
Subsystem Vendor ID	0xA8	For this design, the**Subsystem Vendor ID** is the encoded value used by the GUI to identify the device family and configuration. Consequently, it changes depending on the settings specified.

PCI Base Address Registers (Type 0 Configuration Space)		
Subsystem Device ID	0x2801	N/A

Table 4: Capabilities Parameters–Stratix V GX Device

Capability Registers	
Device Capabilities	
Maximum payload size	256 Bytes
Number of tags supported	32
Completion timeout range	ABCD
Implement completion timeout disable	ON
Error Reporting	
Advanced error reporting (AER)	Off
ECRC check	Off
ECRC generation	Off
ECRC forwarding	Off
Track receive completion buffer overflow	Off
Link Capabilities	
Link port number	1
Data link layer active reporting	Off
Surprise down reporting	Off
Slot clock configuration	On
MSI Capabilities	
MSI messages requested	4
MSI-X Capabilities	
Implement MSI-X	Off
MSI-X Table size	0
MSI-X Table Offset	0x0
MSI-X Table BAR Indicator (BIR)	1
Pending Bit Array (PBA)	0x0
Offset	0
BAR Indicator	0
Slot Capabilities	
Use slot register	Off
Slot power scale	0
Slot power limit	0

Capability Registers	
Slot number	**0**

Table 5: Power Management Parameters–Stratix V GX Device

Parameter	Value
Power Management	
Endpoint L0s acceptable latency	**Maximum of 64 ns**
Endpoint L1 acceptable latency	**Maximum of 1 us**
PHY Characteristics	
Gen2 transmit deemphasis	**6dB**

Quartus II Settings

The `.qar` files in the reference design package has the recommended synthesis, Fitter, and timing analysis settings. These settings are optimized for the parameters chosen in this reference design.

Hardware Requirements

The reference design requires the following hardware:

- The Arria II GX FPGA Development Kit, the Arria V GT FPGA Development Kit, the Arria 10 GX FPGA Development kit, the Cyclone IV GX FPGA Development Kit, the Cyclone V GT FPGA Development Kit, the Stratix IV GX FPGA Development Kit, or and Stratix V GX FPGA Development Kit.
- A computer running Windows driver with an x8/x4/ x1 PCI Express slot for the Arria II GX, Arria V GT, Arria 10 GX, Cyclone IV GX, Cyclone V GT, Stratix IV GX, or Stratix V 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 Arria II GX, Arria V GT, Arria 10 GX, Cyclone IV GX, Cyclone V GT, Stratix IV GX, or Stratix V GX development board, referred to as computer #2 in this document.
- A USB cable or other Altera download cable.
- A PCI Express x8-to-x4 lane converter for x4 operation or x8 to x1 lane converter for x1 operation.

Note: This reference design uses eight 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.
- The Quartus II software running on computer #2.

Related Information

- **PCI Express High-Performance Reference Design in Arria II GX Devices**
- **PCI Express High-Performance Reference Design in Arria V GT Devices**
- **PCI Express High-Performance Reference Design in Arria 10 GX Devices**
- **PCI Express High-Performance Reference Design in Cyclone IV GX Devices**
- **PCI Express High-Performance Reference Design in Cyclone V GT Devices**
- **PCI Express High-Performance Reference Design in Stratix IV GX Devices**
- **PCI Express High-Performance Reference Design in Stratix V GX Devices**

Software Installation

Before you begin

You must have Administrator privileges to install the software application.

The software installation is included with the design files and the application includes an executable driver. The driver configuration is specific to this reference design.

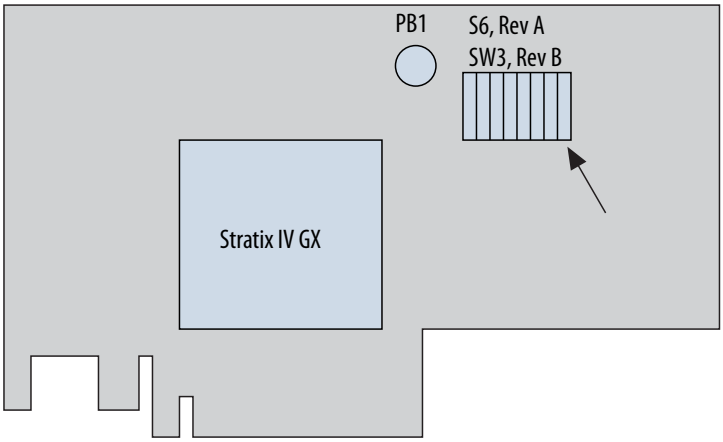
1. Download the appropriate `.zip` design files for your FPGA Development Kit and the demo driver. Extract the compressed files.
2. Before plugging in the PCI Express card, copy the `Windows_for_AVST_On_Chip_Mem` directory to computer #1. This software works with Quartus II release 10.0 SP1 or later.
3. Install the Windows driver. Follow the instructions in the `README.txt` file from the `Windows_for_AVST_On_Chip_Mem` directory to install and 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. The following figure highlights this component. The right-most position of this dip switch sets the operation to normal or PCI Express compliance base board (CBB) testing. To run the software included in this application note, this switch must be in the **off** position. The **off** position points towards the PCIe slot. The **on** position points away from the PCIe slot. When set 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.)

Note: The top-level RTL file has been modified to enable CBB testing. If you regenerate the IP core, 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



1. Power down computer #1 and plug the development board into the PCI Express slot. For an x1 or x4 operation, use a PCI Express lane converter.
2. The development kits include 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.
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.
4. Connect one end of the USB cable the USB port on the development board.
5. Connect the other end of the cable to the USB port on the computer running the Quartus II software on computer #2.

   To download the USB-Blaster driver, go to the Altera support site at **Cable and Adapter Drivers Information**.

   For installation instructions, go to **USB-Blaster and USB-Blaster II Drivers for Windows 7 and Windows Vista**.

Programming using the .sof File

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.

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**. This command lists devices attached to the JTAG chain on the development board.
4. Right-click the Arria II GX (EP2AGX125), Arria V GT (5AGTFD7K3), Arria 10 GX (10AS066N3F40I2LG), Cyclone IV GX (EP4CGX150), Cyclone V GT (5CGTFD9E5), Stratix IV GX (EP4SGX230), or Stratix V (5SGXEA7K2F40) device 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 Altera development kit. 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

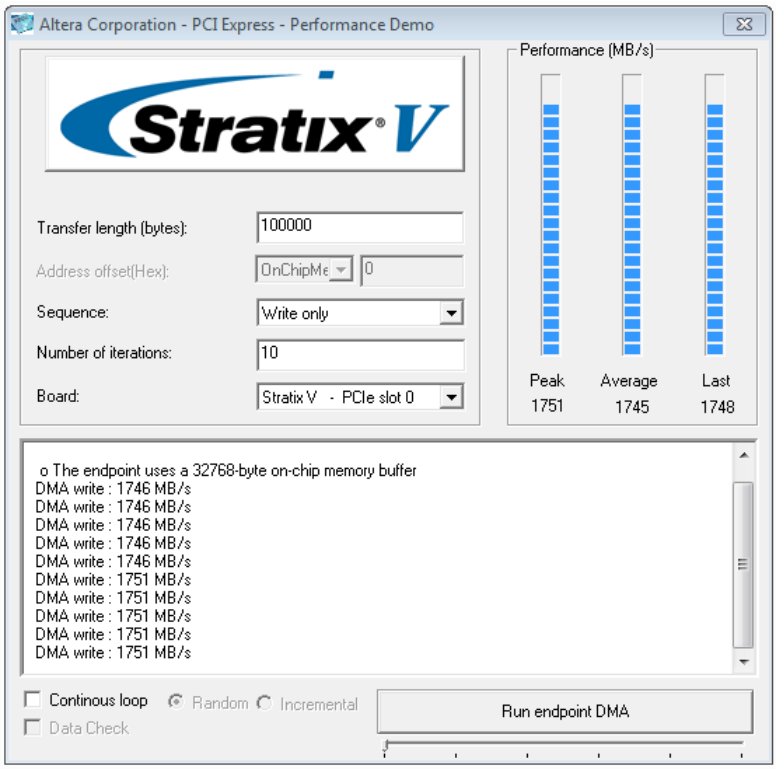
The software GUI has the following control fields:

- **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

1. 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.
2. You can use the GUI to change the **Transfer length** and **Sequence** and repeat the test.

Figure 7: Write-Only Options



3. Double-click on the application `Windows_for_AVST_On_Chip_Mem` in the `Windows_for_AVST_On_Chip_Mem` directory.
4. The application reports the board type, the number of active lanes, the maximum read request size, and the maximum payload size.

Additional Chaining DMA Commands

In addition to the parameter settings to control the chaining DMA, the GUI includes five other commands. The following table describes these commands. The position of the slider control changes the command.

Table 6: PCI Express Performance Demo GUI Commands and Options

Command	Options	Description
Run endpoint DMA	**Write only** **Read only** **Read then write** **Write then read** **Read and write**	Writes transfers data from the FPGA to system memory. Reads transfer data from system memory to the FPGA.
Scan the endpoint configuration space registers	**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.

Command	Options	Description
Scan the current PCI Express board settings	N/A	Reports the configuration settings for the development board.
Scan the motherboard PCI bus	N/A	Reports the vendor ID, device ID, slot, bus, and function numbers for all devices on the motherboard's PCI bus.
Run target read	**At endpoint address** **From 0x0 to endpoint address**	Programs the Root Port of the motherboard's PCI Express chipset to read from the FPGA's memory, as follows: • If you select **At Endpoint address**, you can type the starting read address in Endpoint memory in the **Endpoint address** field. • If you select **From 0x0 to Endpoint address**, the **Endpoint address** field specifies the transfer length.
Run target write	**At endpoint address** **From 0x0 to endpoint address**	Programs the Root Port of the motherboard's PCI Express chipset to write to the FPGA's memory, as follows: • If you select **At Endpoint address**, you can type the starting write address in Endpoint memory the **Endpoint address** field. • If you select **From 0x0 to Endpoint address**, the **Endpoint address** field specifies the transfer length.

Figure 8: Scan the Endpoint Configuration Space Registers

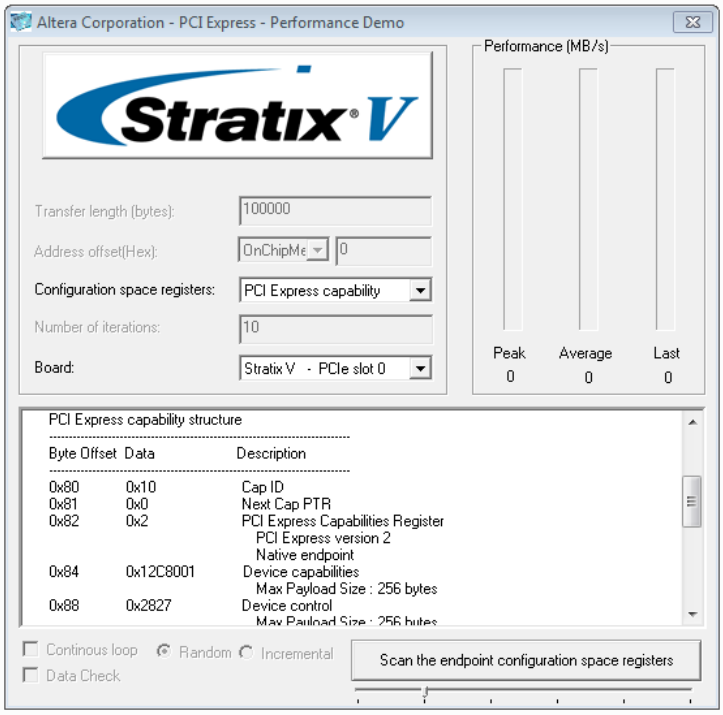
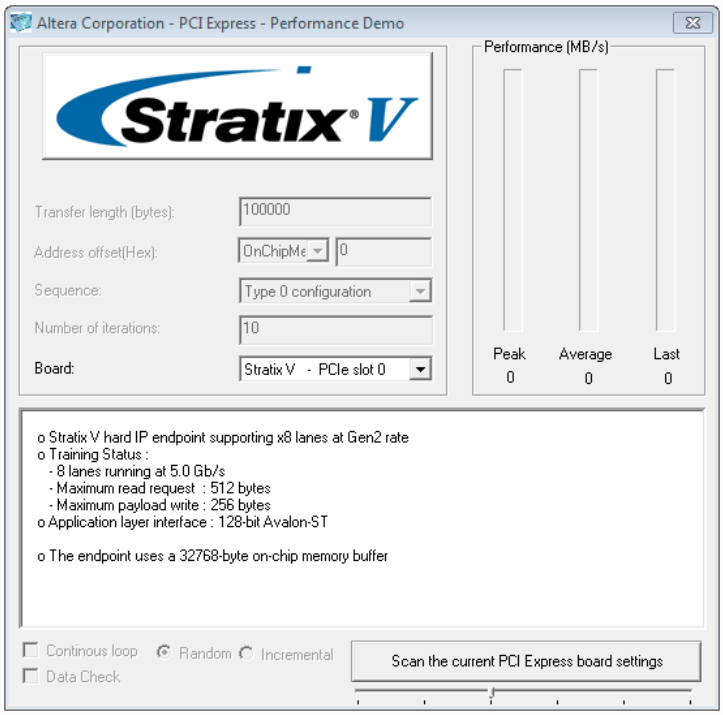


Figure 9: Scan the current PCI Express Board Settings



Using SignalTap II

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 IP core is not receiving enough credits from the device at the far end of the PCI Express link. It could also indicate that a x4 link has trained to x1. In the receive direction, the deassertion of `rx_st_valid0` indicates that the IP core is not receiving enough data.

Performance Benchmarking Results

The following tables list the performance of x1, x4, and x8 operations with the Stratix V GX FPGA development board for the Intel i7-3930K 3.8 GHz Sandy Bridge-E processor using this reference design. The table shows the average throughput with the following parameters:

- 100 KByte transfer
- 20 iterations
- A 256-byte payload
- Maximum 512-byte read request
- 256-byte read completion

Note: Refer to the following web page for other available reference designs and application notes for PCI Express.

- **PCI Express Reference Designs and Application Notes**

Table 7: Arria 10 Hard IP for PCI Express Performance

Configuration	DMA Read (MB/sec)	DMA Write (MB/sec)	Simultaneous DMA Read/ Write (MB/sec)	Theoretical maximum throughputs (MB/sec)	
				DMA Read (MB/sec)	DMA Write (MB/sec)
Gen3, X4	3362	3478	3270/2809	3710	3710
Gen2, X8	3337	3505	3227/2808	3710	3710
Gen1, X1	208	214	199/192	231	231

Table 8: Stratix V Hard IP for PCI Express Performance - Intel i7-3930K Processor

Configuration	DMA Reads (MB/s)	DMA Writes (MB/s)	Simultaneous DMA Read/ Writes (MB/s)	Theoretical Maximum Throughput (MB/s)	
				Read	Write
Gen3, x4	3324	3473	3212/2991	3710	3710
Gen2, x8	3326	3507	3267/2910	3710	3710

Configuration	DMA Reads (MB/s)	DMA Writes (MB/s)	Simultaneous DMA Read/ Writes (MB/s)	Theoretical Maximum Throughput (MB/s)	
				Read	Write
Gen2, x4	1704	1767	1653/1514	1855	1855
Gen2, x1	475	438	401/358	463	463
Gen1, x8	1676	1763	1647/1491	1855	1855
Gen1, x4	839	881	832/800	927	927
Gen1, x1	222	222	214/200	231	231

Table 9: Cyclone V Hard IP for PCI Express Performance

Configuration	DMA Read (MB/sec)	DMA Write (MB/sec)	Simultaneous DMA Read/ Write (MB/sec)	Theoretical maximum throughputs (MB/sec)	
				DMA Read (MB/sec)	DMA Write (MB/sec)
Gen2, X4	1700	1762	1683/1485	1855	1855
Gen1, X4	832	882	849/801	927	927
Gen1, X1	222	225	220/209	231	231

Table 10: Arria V GT Hard IP for PCI Express Performance

Configuration	DMA Read (MB/sec)	DMA Write (MB/sec)	Simultaneous DMA Read/ Write (MB/sec)	Theoretical maximum throughputs (MB/sec)	
				DMA Read (MB/sec)	DMA Write (MB/sec)
Gen2, x4	1719	1784	1673/1450	1855	1855
Gen2, x1	446	451	439/421	463	463
Gen1, x8	1699	1782	1669/1461	1855	1855
Gen1, x4	865	892	806/802	927	927
Gen1, x1	222	225	220/209	231	231

The following tables list the performance of the performance of x8, x4, and x1 operations for development boards using the Intel X58 and using this reference design. The table shows the average throughput with the following parameters:

- 100 KByte transfer
- 20 iterations
- A 256-byte payload
- Maximum 512- byte read request
- 256-byte read completion

Table 11: 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
Hard IP Implementation—Stratix IV GX					
Gen2 x8, 128-bit	3304	3434	2956/2955	3710	3710
Gen2 x4, 128-bit	1708	1783	1684/1484	1855	1855
Gen2 x4, 64-bit	1727	1775	1691/1631	1855	1855
Gen2 x1	448	450	438/425	463	463
Gen1 x8, 128-bit	1694	1778	1678/1480	1855	1855
Gen1 x8, 64-bit	1706	1778	1680/1628	1855	1855
Gen1 x4	875	890	855/815	927	927
Gen1 x1	224	225	219/211	231	231
Soft IP Implementation—Stratix IV GX					
Gen1 x4	873	890	854/811	927	927
Gen1 x1	222	225	219/209	231	231

Table 12: 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)	
Hard IP Implementation—Stratix IV GX					
Gen1 x8, 128-bit	1497	1775	1210/1331	1855	1855
Gen1 x4, 64-bit	859	889	725/767	927	927
Gen1 x1, 64-bit	220	225	217/204	231	231
Soft IP Implementation—Stratix IV GX					
Gen1 x4, 64-bit	860	887	854/780	927	927
Gen1 x1, 64-bit	220	225	203/203	231	231

Table 13: Cyclone 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)	
Soft IP Implementation—Stratix IV GX					
Gen1 x1, 64-bit	220	225	217/203	231	231

Document Revision History

Table 14: Document Revision History

Date	Version	Changes
April 2017	2.5	Made the following changes: • Clarified that the software installation is included in the design (.zip) file in the "Software Installation" section.
October 2015	2.4	Made the following changes: • Added Arria 10 devices globally. • Changed the example filename given in the "File Naming Conventions" section. • Changed the application given in the "Running the Software Application" section. • Changed abbreviation in the "Naming Conventions for Reference Design Variations" table. • Changed the directory name in the "Software Installation" section. • Changed the directory name in the "Running the Software Application" section.
December 2014	2.3	Changed the note in the "Performance Benchmarking Results" section.
December 2014	2.2	Made the following changes: • Added links to **.zip** files for supported devices. • Corrected file name abbreviations. • Updated steps for *Software Installation*. • Changed to use 64-bit software driver.
October 2014	2.1	Made the following changes: • This reference design uses the Cyclone IV GX FPGA Development Kit, not the Transceiver Starter Kit. • Listed Arria V GT, Cyclone V GT, and Stratix V GX development kits that run this reference design. • Simplied language to facilitate understanding by non-native English audience.
January 2014	2.0	• Updated numbers for Stratix V Hard IP for PCI Express Performance • Added Cyclone V and Arria V GT Hard IP for PCI Express Performance numbers

Date	Version	Changes
February 2013	1.6	• Clarified that the only Jungo driver that Altera delivers with this reference design is an executable file configured for the specific reference design. Altera does not provide you a Jungo driver for use in any other application. • Updated with current IP core names. This document update includes no technical changes in the reference design.
August 2012	1.5	• Corrected bandwidths in *Stratix V Hard IP for PCI Express Performance - Intel i7-3930K Processor*.
July 2012	1.4	• Updated to show the performance of the Stratix V GX (5SGXEA7K2F40C2NES) device running on the Stratix V GX FPGA Development Kit. • Removed performance tables for legacy devices.
August 2010	1.3	• Updated to show the performance of the Cyclone IV GX (EP4CGX15) device running on the Cyclone IV GX Transceiver Starter Kit.
August 2009	1.2	• Updated to show the performance of the Arria II GX (EP2AGX125) device running on the Arria II GX FPGA Development Kit.
May 2009	1.1	• Updated to show the performance of the Stratix IV GX (EP4SGX230) device running on the Stratix IV GX FPGA Development Kit. • Added new commands to the PCI Express Performance Demo GUI.
May 2007	1.0	Initial release.