The PCI Express® (PCIe®) to External Memory reference design provides a sample interface between the Altera® IP Compiler for PCI Express MegaCore® function and 64-bit external memory. Altera offers this reference design to demonstrate the operation of the IP Compiler for PCI Express MegaCore function and either a DDR2 or DDR3 SDRAM memory controller. The reference design has the following features:

- Supports PCIe endpoint direct memory access (DMA) read and write transactions
- Uses the IP Compiler for PCI Express hard IP MegaCore function
- Uses the High-Performance SDRAM Controller MegaCore function for DDR2
- Uses the High-Performance SDRAM Controller II MegaCore function for DDR3
- Uses the Altera DDR3 SDRAM Controller with UniPHY in Qsys
- Uses either an Arria® II GX or Stratix® IV GX device with internal transceivers
- Supports the Qsys system integration tool

This application note describes the following topics:

- Reference Design Overview
- Using the Reference Design

Reference Design Overview

The reference design connects the Altera IP Compiler for PCI Express MegaCore function to external memory using the reference design interface circuitry. The design runs on Altera’s Arria II GX FPGA Development Kit or Altera’s Stratix IV GX FPGA Development Kit. Both kits include a PCIe development board. Altera also provides a software driver, programming information, and a GUI to run the application.

The PCI Express to External Memory reference design interfaces to the system side of the Altera IP Compiler for PCI Express IP core (refer to Figure 1). The external side of the IP Compiler for PCI Express MegaCore function forms half of the PCIe link. The memory controller accesses the external memory. The IP Compiler for PCI Express MegaCore function generally operates as a PCIe master or initiator. When the IP Compiler for PCI Express MegaCore function operates as a PCIe master, the DMA engine initiates the transaction, monitors the status, and manages the progress of the data transfers.

This reference design is similar to the chaining DMA design example that you automatically generate when you create an IP Compiler for PCI Express MegaCore function using the MegaWizard® Plug-In Manager.
A Qsys version of this reference design is also available. The Qsys version showcases PCIe and DDR3 as the standard Qsys components that supports the Avalon® Memory Map (Avalon-MM) interface. The performance-enhanced mSGDMA component also has an Avalon-MM interface that you can smoothly integrate with other blocks.

For a detailed description of the chaining DMA design example, refer to Chapter 15 Testbench and Design Example in the IP Compiler for PCI Express User Guide.

Figure 1. Block Diagram for the PCIe to External Memory Reference Design

The difference between the chaining DMA design example and the PCI Express to External Memory reference design is that the reference design uses external memory to store data instead of internal memory in the FPGA. Consequently, the files in the chaining DMA design example that define the DMA controller and memory accesses are modified. In addition, the top-level Verilog HDL file, <name>_example_chaining_top.v, points to the external memory version of these files.

Table 1 lists the files in the DMA design example and PCI Express to External Memory reference design that access memory.

Table 1. Files that Access Memory (Part 1 of 2)

<table>
<thead>
<tr>
<th>File Name in the Chaining DMA Design Example</th>
<th>File Name in the PCIe to External Memory Reference Design</th>
</tr>
</thead>
<tbody>
<tr>
<td>altpcie_dma_requester_128.v</td>
<td>altpcie_dma_requester_128_ddr.v</td>
</tr>
<tr>
<td>altpcie_dma_dt.v</td>
<td>altpcie_dma_dt_ddr.v</td>
</tr>
<tr>
<td>altpcie_dma_pr_reg.v</td>
<td>altpcie_dma_pr_reg_ddr.v</td>
</tr>
<tr>
<td>altpcie_cdma_app_icm.v</td>
<td>altpcie_cdma_app_icm_ddr.v</td>
</tr>
</tbody>
</table>
Figure 1 shows the modules that include these files in dark blue.

Be aware that the Qsys version of this reference design has a different file and directory structure. The structure follows the standard Qsys-generated design’s file and directory structure. For more information about the Qsys directory structure for this reference design, refer to Chapter 5 Creating a System with Qsys in volume 1 of the Quartus II Handbook.

The following sections provide an overview of the modules in this reference design that differ from the comparable modules in the chaining DMA design example that you automatically generate when you create an IP Compiler for PCI Express MegaCore function.

### PCIe to Memory Interface Block

The memory interface block interfaces to DDR3 memory on the Stratix IV GX development board and DDR2 memory on the Arria II GX development board. Figure 2 shows the PCIe Express to External Memory subsystem.

Table 1. Files that Access Memory (Part 2 of 2)

<table>
<thead>
<tr>
<th>File Name in the Chaining DMA Design Example</th>
<th>File Name in the PCIe to External Memory Reference Design</th>
</tr>
</thead>
<tbody>
<tr>
<td>altpcie_r_example_app_chaining.v</td>
<td>altpcie_r_example_app_chaining_ddr.v</td>
</tr>
<tr>
<td><code>&lt;name&gt;_example_chaining_pipen1b.v</code></td>
<td><code>&lt;name&gt;_example_chaining_pipen1b_ddr.v</code></td>
</tr>
</tbody>
</table>

Figure 2. Datapath from IP Compiler for PCI Express MegaCore Function to External Memory

On the Stratix IV GX development board, DDR3 memory transfers 64 bits on each clock edge with a burst size of eight for a total of 512 bits per burst transfer. The local interface to the DDR3 High Performance SDRAM Controller II is 256 bits wide and runs at half rate, or 266.667 MHz, so that the local interface must complete two transfers to match the external memory transfer size of 512 bits.

On the Arria II GX development board, the DDR2 memory transfers 64 bits on each clock edge with a burst size of four for a total of 256 bits per burst transfer. The local interface of the DDR2 High Performance Controller is 256 bits wide and runs at half rate, or 150 MHz.
The reference design uses three FIFOs to pass data between the external memory interface clock domain and the PCIe clock domain. The three separate FIFOs handle access commands, write data, and read data. The FIFOs are instantiated in a block called DDR2_Fifo_Interfaces.vhd or DDR3_Fifo_Interfaces.vhd for DDR2 and DDR3, respectively. This file also defines a state machine to control the timing of the data and commands to the memory controller’s local interface and monitor the status signals from it.

The following pseudo code summarizes the algorithm that the state machine implements:

1. Wait for the command FIFO to not be empty.
2. Decode the read or write command.
3. For reads, ensure the memory controller is ready to receive a command, then send the read.
4. Writes are different for the Arria II GX and Stratix IV GX development boards:
   a. For the Arria II GX development board, retrieve two consecutive words before sending a 256-bit word to the memory controller. The two PCIe write commands must write to two consecutive addresses. These constraints are built into the state machine.
   b. For the Stratix IV GX development board, because the PCIe word size is 128 bits, retrieve four consecutive words before sending two 256-bit words on consecutive clock cycles. To match throughput, four write commands are required by the IP Compiler for PCI Express MegaCore function for each external memory write in this particular design application. The write commands must be two consecutive addresses. These constraints are built into the state machine. When all the data is collected and the memory controller is ready, the state machine writes the data to the memory controller and then decodes the next command.

There are several alternatives when designing hardware that interfaces to the external memory. For more information, refer to Volume 3: Implementing Altera Memory Interface IP in the External Memory Interface Handbook.

**External Memory**

The following sections describe DDR2 and DDR3 memory performance on Stratix IV GX and Arria II GX development boards.

**DDR2 SDRAM IP Block**

An updated Arria II GX FPGA Development Kit will soon be released that will use a 64-bit Micron DDR2 SODIMM MT8HTF1284H. The entire interface runs at the maximum supported speed of 300 MHz to ensure that the DDR2 memory system does not limit the bandwidth of the reference design.
Example 1 describes the bandwidth calculations for the PCIe and DDR2 SDRAM interfaces.

### Example 1. Bandwidth Calculations

<table>
<thead>
<tr>
<th>Description</th>
<th>Bandwidth</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gen1 PCI Express</td>
<td>128 bits @ 125 MHz = 16 Gbps (minus some overhead)</td>
</tr>
<tr>
<td>DDR2</td>
<td>64 bits @ 300 MHz = 38.4 Gbps (minus overhead due to controller and interface inefficiencies)</td>
</tr>
</tbody>
</table>

As these calculations indicate, by running the DDR2 interface at 300 MHz, the available bandwidth easily accommodates the bandwidth required for Gen1 operation. The reference design uses Altera’s DDR2 High Performance Controller which consists of a memory PHY (ALTMEMPHY) and controller code.

### DDR3 SDRAM IP Block

The Stratix IV GX FPGA Development Kit uses a bank of four, 16-bit Micron MT41J64M16 DDR3 components to create a 64-bit memory interface. The interface uses fly-by topology as specified in JEDEC Standard for DDR3 SDRAM, JES79-3C. The entire interface runs at the maximum supported speed of 533.333 MHz to ensure that the DDR3 memory system does not limit the bandwidth of the reference design. This speed accommodates the bandwidth required for Gen2 operation. The reference design uses Altera’s DDR3 SDRAM Controller with the UniPHY IP found in the Qsys component library. This IP consists of the memory PHY (UniPHY) and the controller (High Performance Controller II) code.

The memory interface block interfaces to the DDR3 memory on the Stratix IV GX development board. As mentioned, DDR3 memory runs at 533.333 MHz. This means that the local interface (being a half rate design) runs at 266.667 MHz. The maximum burst length is set to 32, so the DDR3 memory transfers 256 bits on each clock edge with a burst length of 32.

### Example 2. Bandwidth Calculations

<table>
<thead>
<tr>
<th>Description</th>
<th>Bandwidth</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gen2 PCI Express</td>
<td>128 bits @ 250 MHz = 32 Gbps (minus some overhead)</td>
</tr>
<tr>
<td>Gen1 PCI Express</td>
<td>128 bits @ 125 MHz = 16 Gbps (minus some overhead)</td>
</tr>
<tr>
<td>DDR3</td>
<td>64 bits @ 533 MHz = 68.4 Gbps (minus overhead due to controller and interface inefficiencies)</td>
</tr>
</tbody>
</table>

As these calculations indicate, by running the DDR3 interface at 533 MHz, the available bandwidth easily accommodates the bandwidth required for Gen2 operation. The reference design uses Altera’s High-Performance SDRAM Controller II MegaCore function which consists of a memory PHY (ALTMEMPHY) and controller code.


For more information about the high performance DDR3 or DD2 SDRAM memory controllers, refer to the Section 1. DDR and DDR2 High-Performance Controllers and ALTMEMPHY IP User Guide or the Section II. DDR3 High Performance Controller and ALTMEMPHY IP User Guide of the External Memory Interface Handbook.
For more information about using external memories with UniPHY Design Tutorials, refer to the *DDR3 SDRAM Controller with UniPHY Using Qsys* chapter in the UniPHY Design Tutorials section in the *External Memory Interface Handbook Volume 6*.

For more information about using external memories with Altera FPGAs, refer to the *Using DDR2 and DDR3 SDRAM Controller with UniPHY User Guide* chapter in the UniPHY Design Tutorials section in the *External Memory Interface Handbook Volume 3*.

**mSGDMA**

mSGDMA consists of three components—the dispatcher, read master, and write master. Figure 3 shows mSGDMA.

![Figure 3. Block Diagram of the mSGDMA](image)

The dispatcher contains the descriptor FIFO and controls both the read and write master operations. The dispatcher is connected to the PCIe IP using bar1_0 access; therefore, the PC has control over the PCIe bus. The dispatcher has a status register and a control register. The PCIe GUI accesses these registers to control operation.

The read master works as the Avalon-MM to Avalon-ST converter. The write master works as the Avalon-ST to Avalon-MM converter. Both the read and write masters have internal FIFOs buffering data between the Avalon-ST and Avalon-MM domains.

If you use your own software driver, you must ensure that the mSGDMA component buffer and the Avalon to PCIe Address Translation Table entries are compatible. The mSGDMA buffer must have size 16 Mbytes or smaller for this reference design, and the translation table entries must ensure that an mSGDMA read address source that crosses the window page boundary selects the correct page in the translation table.
mSGDMA Descriptor Fields

All mSGDMA descriptor fields are aligned on byte boundaries and, when necessary, span multiple bytes. Each byte lane of the descriptor slave port can be accessed independently of the others, allowing you the ability to populate the descriptor using any access size.

The control field of the descriptors is located at a different offset depending on the format you use. The last bit of the control field commits the descriptor to the dispatcher buffer when it is asserted. As a result, the control field is always located at the end of a descriptor to allow the host to write the descriptor sequentially to the dispatcher block.

Table 2 lists the standard descriptor format.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Byte Lanes</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0</td>
<td>Read Address[31..0]</td>
</tr>
<tr>
<td>0x4</td>
<td>Write Address[31..0]</td>
</tr>
<tr>
<td>0x8</td>
<td>Length[31..0]</td>
</tr>
<tr>
<td>0xC</td>
<td>Control[31..0]</td>
</tr>
</tbody>
</table>

Table 2. Standard Descriptor Format

Control Field

The control field is available for both standard and extended descriptor formats. Use this field to program the characteristics of the transfer; for example, parked descriptors, error handling, and interrupt masks. The interrupt masks are programmed into the descriptor so that the interrupt enables can be unique for each transfer. Table 3 lists the control field names and bit offset.

Table 3. Control Field Names and Bit Offset Values

<table>
<thead>
<tr>
<th>BitOffset</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>7-0</td>
<td>Transmit Channel</td>
</tr>
<tr>
<td>8</td>
<td>Generate SOP</td>
</tr>
<tr>
<td>9</td>
<td>Generate EOP</td>
</tr>
<tr>
<td>10</td>
<td>Park Reads</td>
</tr>
<tr>
<td>11</td>
<td>Park Writes</td>
</tr>
<tr>
<td>12</td>
<td>End on EOP</td>
</tr>
<tr>
<td>13</td>
<td>End on EOP or Length</td>
</tr>
<tr>
<td>14</td>
<td>Transfer Complete IRQ Mask</td>
</tr>
<tr>
<td>15</td>
<td>Early Termination IRQ Mask</td>
</tr>
<tr>
<td>23-16</td>
<td>Transmit Error/Error IRQ Mask</td>
</tr>
<tr>
<td>24</td>
<td>Early done enable</td>
</tr>
<tr>
<td>30-25</td>
<td>&lt;reserved&gt;</td>
</tr>
<tr>
<td>31</td>
<td>Go</td>
</tr>
</tbody>
</table>
Table 4 lists the dispatcher control and status registers (CSR).

Table 4. Dispatcher Control and Status Registers

<table>
<thead>
<tr>
<th>Offset</th>
<th>Access</th>
<th>Byte Lanes</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0</td>
<td>Read/Clear</td>
<td>Status</td>
</tr>
<tr>
<td>0x4</td>
<td>Read/Write</td>
<td>Control</td>
</tr>
<tr>
<td>0x8</td>
<td>Read</td>
<td>Write Fill Level[15..0] Read Fill Level[15..0]</td>
</tr>
<tr>
<td>0xC</td>
<td>Read</td>
<td>&lt;reserved&gt; Response Fill Level[15..0]</td>
</tr>
<tr>
<td>0x10</td>
<td>Read</td>
<td>Write Sequence Number[15..0] Read Sequence Number[15..0]</td>
</tr>
<tr>
<td>0x14</td>
<td>N/A</td>
<td>&lt;reserved&gt;</td>
</tr>
<tr>
<td>0x18</td>
<td>N/A</td>
<td>&lt;reserved&gt;</td>
</tr>
<tr>
<td>0x1C</td>
<td>N/A</td>
<td>&lt;reserved&gt;</td>
</tr>
</tbody>
</table>

Qsys Design Overview

The Qsys system contains PCIe, OnChipMem, DDR3, and mSGDMA.

- PCIe, OnChipMem, and mSGDMA uses a PCIe-generated 250 MHz clock as the system clock.
- DDR3 uses a 266 MHz clock as the system clock.
- PCIe uses 64 bits as the system data interface.
- OnChipMem, DDR3, and mSGDMA uses 256 bits as the system data interface.
- Bar1_0 connects to OnChipMem and DDR3 for quick reads and writes.
- Bar2 connects to PCIe and mSGDMA for the setup registers.

Figure 4 shows the Qsys System Contents page.

Figure 4. Qsys System Contents
Qsys uses two different clock domains, 250 MHz and 266 MHz, instead of using a clock-cross bridge. Figure 5 shows the Qsys Project Settings page.

**Figure 5. Qsys System Project Settings**

To handle clock crossing using Qsys, set the **Clock crossing adapter type** option, in the **Project Settings** tab, to either **Auto** or **FIFO**. To ease the fitter to meet timing, set the **Limit interconnect pipeline stages to** option to 4.

Outside of the Qsys system, there are two components required:

- One PLL to generate 50 MHz and 125 MHz
- One `altgx_reconfig` to control the PCIe

Figure 6 shows the two required components to control the transceiver physical medium attachment (PMA) settings and apply offset cancellation to enhance performance.

**Figure 6. Qsys Required Components**
The remaining circuitry is used only for the LED drive.

**Using the Reference Design**

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

- “Hardware Requirements” on page 10
- “Software Requirements” on page 11
- “Software Installation” on page 11
- “Hardware Installation” on page 12
- “Running the Software Application” on page 13
- “Running the Simulation with the Non-Qsys Version of the Design” on page 24
- “Running the Simulation with the Qsys Version Design” on page 27

**Hardware Requirements**

The reference design requires the following hardware:

- The Stratix IV GX FPGA Development Kit. The Arria II GX must be used with the Arria II GX FPGA Development Kit.
- A computer running 32-bit Windows XP with an x8/x4/x1 PCIe slot for the Arria II GX FPGA or Stratix IV GX FPGA development board. The software application and hardware are installed on this computer, referred to as computer #1 in this application note.
- A computer with the Quartus II software for downloading FPGA programming files to the development board, referred to as computer #2 in this application note.
- A USB cable or other Altera download cable.

The GUI requirements include the following:

- Microsoft Windows XP 32/64 OS
- Microsoft Windows 7 32/64 OS
- Motherboard with PCIe x8 or x16 slot(s) with version two support
- BIOS with support negotiate to x4
- Target board with a Stratix-IV PCIe Develop Kit (production silicon version)
Software Requirements

The reference design application requires the following software installation:

- Reference design software installed on computer #1.
- PCIe to External Memory reference design package, available as a downloadable compressed file.

For more information, refer to the PCI Express to External Memory reference design product page.

- The Quartus II software version 9.1 or later running on computer #2.

Software Installation

You must have Administrator privileges to install the software application.

The software application runs on both 32-bit and 64-bit Windows XP and Windows 7 and includes a Jungo WinDriver executable driver for evaluating. The driver configuration is specific to this reference design.

To install the software application and Windows drivers, follow these steps:

1. Download the PCIe to External Memory reference design software to computer #1 and extract the compressed files. Figure 7 shows the directory structure.

2. Before plugging in the PCIe card, copy the \altpcie_demo_91 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_Qsys_32.exe or altpcie_demo_Qsys_64.exe (depending on your operating system) from the \altpcie_demo_91 directory to start the software application.

Figure 7. Directory Structure


**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 PCIe mode of operation on computer #1 before plugging it in. Figure 8 shows this component. The right-most position of this dip switch is used to change between normal operation and PCIe 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 register transfer 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 8. Location of Components that Control PCIe Mode of Operation**

---

To install the hardware, follow these steps:

1. Power down computer #1 and plug the development card into the PCIe slot. Depending on your hardware, a PCIe lane converter may be required.

2. Both 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.
Connecting the USB-Blaster Cable
To connect the USB-Blaster cable, follow these steps:
1. Connect one end of the USB-Blaster cable to the USB port.
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 SRAM Object File (.sof) 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.)

To program the FPGA with the .sof file, follow these steps:
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 Arria II GX (EP2AGX125EF35) device or Stratix IV GX (EP4SGX230) 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 Arria II GX or Stratix IV GX device. The device is configured when the Progress bar reaches 100%.
7. On computer #1, exit the BIOS System Setup or boot the manager interface.
8. On computer #1, press Ctrl+Alt+Delete to start 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
To run the software application, follow these steps:
1. Double-click on the application altpcie_demo_Qsys_32.exe or altpcie_demo_Qsys_64.exe (depending on your operating system) in the altpcie_demo_91 directory.
2. The application reports the board type, the number of active lanes, the maximum read request size, and the maximum payload size. Figure 9 shows the GUI for the Stratix IV GX device.

The software GUI has the following control fields:

- **Transfer length**—Specifies the transfer length (in bytes) from 32 bytes to 262,144 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**—Performs the transfer continuously when this option is turned on

3. Set the **Transfer length** to 99,968 bytes and the **Sequence** to **Write then Read**, and click **Run**.

When set to **Write then Read**, the software programs the DMA registers in the FPGA to transfer data from the FPGA to the external memory. 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 9 through Figure 22 on page 24 are for Stratix IV GX devices only. Arria II GX devices have different DMA descriptions.

---

Figure 9. Write then Read Options
In addition to the parameter settings to control the chaining DMA, the GUI includes commands that you can use to obtain configuration information about the device and board, and to perform root port reads and writes. Table 5 lists all of the available commands. The position of the slider control changes the command.

Table 5. PCIe Performance Demo GUI Commands and Options

<table>
<thead>
<tr>
<th>Command</th>
<th>Options</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run endpoint DMA (Figure 9)</td>
<td>Write only</td>
<td>Writes transfer data from the FPGA to the system memory.</td>
</tr>
<tr>
<td></td>
<td>Read only</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Read then write</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Write then read</td>
<td></td>
</tr>
<tr>
<td>Scan the endpoint configuration space registers (Figure 10)</td>
<td>Type 0 Configuration</td>
<td>Reports the byte address offset, value, and a description of the selected register set.</td>
</tr>
<tr>
<td></td>
<td>MSI capability</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Power management capability</td>
<td></td>
</tr>
<tr>
<td>Scan the current PCIe board settings (Figure 11)</td>
<td>—</td>
<td>Reports the configuration settings of the development board.</td>
</tr>
<tr>
<td>Scan the motherboard PCI bus</td>
<td>—</td>
<td>Reports the vendor ID, device ID, slot, bus, and function numbers for all devices on the motherboard’s PCI bus.</td>
</tr>
</tbody>
</table>

Note to Table 5:
(1) This software application is a different version of the software application used in AN 456: PCI Express High Performance Reference Design. To run this reference design, you must use the software version included with this reference design.

Figure 10 shows the output of the Scan the current PCI Express board settings command for the Stratix IV GX device.

Figure 10. Scan the Current PCIe Board Settings
Figure 11 shows the output of the **Scan the motherboard PCI bus** command for the Stratix IV GX device.

**Figure 11. Scan the Motherboard PCI Bus**

![Scan the Motherboard PCI Bus](image)

**GUI User Interface**

When you launch the GUI and the board detected properly, you will see the results shown in Figure 11. You can review the scan and test results in the `pcie_log.txt` file created each time you run the application.

Use the slide bar on the bottom of the page to set the current command that can be executed by clicking the button in the lower right-hand area of the GUI. The available commands are shown in the following sections.
Figure 12 shows the Qsys PCIe GUI for a Stratix IV GX device.

**Figure 12. Qsys PCIe GUI**
Scan the Endpoint Configuration Space Registers

Figure 13 shows the output of the Scan the endpoint configuration space registers command for the Stratix IV GX device.

Figure 13. Scan the Endpoint Configuration Space Registers
Scan the Current PCIe Board Settings

Figure 14 shows the output of the Scan the current PCI Express board settings command for the Stratix IV GX device.

Figure 14. Scan the Current PCIe Board Settings
Scan the Motherboard PCI Bus

Figure 15 shows the output of the Scan the motherboard PCI bus command for the Stratix IV GX device.

Figure 15. Scan the Motherboard PCI Bus
Run the Target Read (bar0 Read)

Figure 16 shows the output of the **Run target read** command for the Stratix IV GX device.

**Figure 16. Target Read (bar0 Read)**

You can select the target device by selecting **OnChipMem** or **DDR** from the **Address offset[Hex]**: option (Figure 17).

**Figure 17. Read Offset—OnChipMem or DDR**

You can change the read offset. For example, in Figure 17, bar0 access reads to OnChipMem with 0x07000000. You can read to DDR3 by setting the offset option to 0x08000000.

Table 6 lists the offset in hex value.

<table>
<thead>
<tr>
<th>Option</th>
<th>Start</th>
<th>End</th>
</tr>
</thead>
<tbody>
<tr>
<td>OnChipMem</td>
<td>0x0000.0000</td>
<td>0x0003.ffff</td>
</tr>
<tr>
<td>DDR</td>
<td>0x0000.0000</td>
<td>0x0fff.ffff</td>
</tr>
</tbody>
</table>
Run the Target Write (bar0 Write)

Figure 18 shows the output of the Run target write command for the Stratix IV GX device.

**Figure 18. Target Write (bar0 Write)**

![Image of the user interface showing the target write command output for the Stratix IV GX device.](image)

You can select the target device by selecting OnChipMem or DDR from the Address offset[Hex]: option (Figure 19).

**Figure 19. Write Offset—OnChipMem or DDR**

![Image of the write offset options for OnChipMem and DDR.](image)

You can change the write by using the drop-down box to select the memory target to be either OnChipMem or DDR. Table 7 lists the valid address range within each target memory. For example, in Figure 19, bar0 access writes to OnChipMem with 0x07000000. You can write to DDR3 by setting the offset option to 0x08000000.

**Table 7. Memory Target Address Ranges**

<table>
<thead>
<tr>
<th>Option</th>
<th>Start</th>
<th>End</th>
</tr>
</thead>
<tbody>
<tr>
<td>OnChipMem</td>
<td>0x0000.0000</td>
<td>0x0003.ffff</td>
</tr>
<tr>
<td>DDR</td>
<td>0x0000.0000</td>
<td>0x0eff.ffff</td>
</tr>
</tbody>
</table>
OnChipMemory DMA Access

You can specify the transaction length in multiples of 32 bytes. The shortest length is 32 bytes; the longest length is 262,144 bytes (or 0x40000) (Figure 20).

Figure 20. DDR DMA

From the **Sequence** drop down menu, select one of the following settings (Figure 21):

- **PC -> FPGA**
- **FPGA -> PC**
- **FPGA -> PC -> FPGA**
- **PC -> FPGA->PC**

Figure 21. Sequence

You can select **Continuous loop** mode by enabling the bottom check box (Figure 20). This function allows you to run the GUI until you select the **stop** button.

You can select either to enable or disable **Data Check** verification by selecting the **Data Check** check box. If the software detects a data error within a DMA transfer, it reports the error to the screen.

You can also select either **Random** or **Incremental** data patterns by selecting the appropriate check box.
**Performance Monitor**

The peak column on the left records the highest measured performance. The average is considered the average of 500 samples. The last is considered the last measured performance value.

Figure 22 shows the output of the **Performance with Data Check** enabled command for the Stratix IV GX device.

**Figure 22. Performance Monitor**

![Performance Monitor](image)

---

**Running the Simulation with the Non-Qsys Version of the Design**

To simulate this PCIe reference design for either the Arria II GX or Stratix IV GX FPGA, follow these steps:

1. Download `altpcie_zip_91.zip` from the PCI Express to External Memory reference design product page.
2. Unzip the `altpcie_zip_91.zip` to your project directory. You will see the following directory: `<install_dir>\altpcie_zip_91\top_examples\chaining_dma\testbench`.
3. Choose **Programs > Altera > ModelSim <version>** (Windows Start menu) to run the ModelSim® software.
4. Select File, then Change Directory.
5. Browse to `<install_dir>\altpcie_zip_91\top_examples\chaining_dma\testbench`.
6. Click OK.
7. At the ModelSim command prompt, type `do runtb.do`

   This script compiles all of the Verilog modules necessary to run the simulation and loads the simulation files.

8. To set up the wave viewer, type `do wave.do`

9. To run the simulation, type `run -all`

   The simulation begins by initializing simulation models for both the PCIe hard IP MegaCore function, the external memory controller, and external memory. Then, it runs the DMA tests.
Example 3 shows excerpts from the transcript of a successful simulation.

**Example 3. Transcript from Successful Simulation**

```
#INFO:26440ns DMA: Read
#INFO:26440ns TASK:dma_rd_test
#INFO:26440ns TASK:dma_set_rd_desc_data
#INFO:26440ns TASK:dma_set_msi READ
#INFO:26440ns Message Signaled Interrupt Configuration
#INFO:26440ns msi_address (RC memory) = 0x07F0
#INFO:7040ns msi_control_register = 0x0084
#INFO:29440ns msi_expected = 0xB0FC
#INFO:29440ns msi_capabilities address = 0x0050
#INFO:29440ns multi_message_enable = 0x0002
#INFO:29440ns msi_number = 0000
#INFO:29440ns msi_traffic_class = 0000
#INFO:29440ns TASK:dma_set_header READ
#INFO:29480ns Writing Descriptor header
#INFO:29512ns TASK:msi_poll    Polling MSI Address:07F0---> Data:FADE......
#INFO:29696ns TASK:rcmem_poll Polling RC Address0000090C current data (0000FADE)
#INFO:29696ns task:msi_poll Received DMA Read MSI(0000) : B0FC
#INFO:47152ns Completed DMA Read
#INFO:47152ns TASK:chained_dma_test
#INFO:47152ns DMA: Write
#INFO:47152ns TASK:dma_wr_test
#INFO:47152ns TASK:dma_set_wr_desc_data
#INFO:47152ns TASK:dma_set_msi WRITE
#INFO:47152ns Message Signaled Interrupt Configuration
#INFO:47152ns msi_address (RC memory) = 0x07F0
#INFO:47760ns msi_control_register = 0x00A5
#INFO:50160ns msi_expected = 0xB0FD
#INFO:50160ns msi_capabilities address = 0x0050
#INFO:50160ns multi_message_enable = 0x0002
#INFO:50160ns msi_number = 0001
#INFO:50160ns msi_traffic_class = 0000
#INFO:50200ns Shared Memory Data Display:
#INFO:50200ns Address  Data
#INFO:50200ns 00000800 10100003 00000000 00000800 CAFEFADE
#INFO:50200ns TASK:dma_set_rclast
#INFO:50228ns TASK:msi_poll    Polling MSI Address:07F0---> Data:FADE......
#INFO:62132ns TASK:msi_poll Received DMA Write MSI(0000) : B0FD
#INFO:62212ns TASK:rcmem_poll Polling RC Address0000080C current data (0000FADE)
#INFO:62212ns TASK:rcmem_poll ---Received Expected Data (00000002)
#INFO:62220ns Completed DMA Write
#INFO:62220ns TASK:check_dma_data
#INFO:62220ns Passed : 4096 identical dwords.
```
Running the Simulation with the Qsys Version Design

To simulate this Qsys-based PCIe reference design for the Stratix IV FPGA, follow these steps:

1. Download the zip file `pcie_ddr3_s4gx_qsys.zip` from the PCI Express to External Memory reference design product page.
2. Unzip the file to your project directory. You will see the following directory:
   `<install_dir>\<zip>\q_sys\testbench`.
3. Choose Programs > Altera > Modelsim `<version>` (Windows Start Menu) to run the ModelSim software.
4. Select File, then Change Directory.
5. Browse to `<install_dir>\<zip>\q_sys\testbench`.
6. Click OK.
7. At the Modelsim command prompt, type `do msim_setup.tcl` ←
   
   This script compiles all of the modules necessary to run the simulation.
8. Type `ld` ← at the command prompt to load all the simulation libraries/files.
9. Type `run -all` ← to run the simulation.

Example 4.

```
#INFO:62220ns TASK:downstream_loop
#INFO:63116ns Passed: 0004 same bytes in BFM mem addr 0x00000040 and 0x00000840
#INFO:63988ns Passed: 0008 same bytes in BFM mem addr 0x00000040 and 0x00000840
#INFO:64868ns Passed: 0012 same bytes in BFM mem addr 0x00000040 and 0x00000840
#INFO:65756ns Passed: 0016 same bytes in BFM mem addr 0x00000040 and 0x00000840
#INFO:66652ns Passed: 0020 same bytes in BFM mem addr 0x00000040 and 0x00000840
#INFO:67556ns Passed: 0024 same bytes in BFM mem addr 0x00000040 and 0x00000840
#INFO:68468ns Passed: 0028 same bytes in BFM mem addr 0x00000040 and 0x00000840
#INFO:69388ns Passed: 0032 same bytes in BFM mem addr 0x00000040 and 0x00000840
#INFO:70324ns Passed: 0036 same bytes in BFM mem addr 0x00000040 and 0x00000840
#INFO:71268ns Passed: 0040 same bytes in BFM mem addr 0x00000040 and 0x00000840
# SUCCESS: Simulation stopped due to successful completion!
# Break in Function ebfm_log_stop_sim at ../../../common/testbench//altpcietb_bfm_log.v
line 96
```
Example 5 shows excerpts from the transcript of a successful simulation.

Example 5. Transcript from Successful Simulation

```plaintext
1 | 1688842211 [VM0-0001]: Reading data 0000000000000000 @ 16b (CRC-4/8/16/16) burst: 3
2 | 1688842211 [VM0-0001]: Reading data 0000000000000000 @ 16b (CRC-4/8/16/16) burst: 3
3 | 1688842211 [VM0-0001]: Reading data 0000000000000000 @ 16b (CRC-4/8/16/16) burst: 3
4 | 1688842211 [VM0-0001]: Reading data 0000000000000000 @ 16b (CRC-4/8/16/16) burst: 3
5 | 1688842211 [VM0-0001]: Reading data 0000000000000000 @ 16b (CRC-4/8/16/16) burst: 3
6 | 1688842211 [VM0-0001]: Reading data 0000000000000000 @ 16b (CRC-4/8/16/16) burst: 3
7 | 1688842211 [VM0-0001]: Reading data 0000000000000000 @ 16b (CRC-4/8/16/16) burst: 3
8 | 1688842211 [VM0-0001]: Reading data 0000000000000000 @ 16b (CRC-4/8/16/16) burst: 3
9 | 1688842211 [VM0-0001]: Reading data 0000000000000000 @ 16b (CRC-4/8/16/16) burst: 3
10 | 1688842211 [VM0-0001]: Reading data 0000000000000000 @ 16b (CRC-4/8/16/16) burst: 3
11 | 1688842211 [VM0-0001]: Reading data 0000000000000000 @ 16b (CRC-4/8/16/16) burst: 3
12 | 1688842211 [VM0-0001]: Reading data 0000000000000000 @ 16b (CRC-4/8/16/16) burst: 3
13 | 1688842211 [VM0-0001]: Reading data 0000000000000000 @ 16b (CRC-4/8/16/16) burst: 3
14 | 1688842211 [VM0-0001]: Reading data 0000000000000000 @ 16b (CRC-4/8/16/16) burst: 3
15 | 1688842211 [VM0-0001]: Reading data 0000000000000000 @ 16b (CRC-4/8/16/16) burst: 3
16 | 1688842211 [VM0-0001]: Reading data 0000000000000000 @ 16b (CRC-4/8/16/16) burst: 3
17 | 1688842211 [VM0-0001]: Reading data 0000000000000000 @ 16b (CRC-4/8/16/16) burst: 3
18 | 1688842211 [VM0-0001]: Reading data 0000000000000000 @ 16b (CRC-4/8/16/16) burst: 3
19 | 1688842211 [VM0-0001]: Reading data 0000000000000000 @ 16b (CRC-4/8/16/16) burst: 3
20 | 1688842211 [VM0-0001]: Reading data 0000000000000000 @ 16b (CRC-4/8/16/16) burst: 3
21 | 1688842211 [VM0-0001]: Reading data 0000000000000000 @ 16b (CRC-4/8/16/16) burst: 3
22 | 1688842211 [VM0-0001]: Reading data 0000000000000000 @ 16b (CRC-4/8/16/16) burst: 3
23 | 1688842211 [VM0-0001]: Reading data 0000000000000000 @ 16b (CRC-4/8/16/16) burst: 3
24 | 1688842211 [VM0-0001]: Reading data 0000000000000000 @ 16b (CRC-4/8/16/16) burst: 3
25 | 1688842211 [VM0-0001]: Reading data 0000000000000000 @ 16b (CRC-4/8/16/16) burst: 3
26 | 1688842211 [VM0-0001]: Reading data 0000000000000000 @ 16b (CRC-4/8/16/16) burst: 3
27 | 1688842211 [VM0-0001]: Reading data 0000000000000000 @ 16b (CRC-4/8/16/16) burst: 3
28 | 1688842211 [VM0-0001]: Reading data 0000000000000000 @ 16b (CRC-4/8/16/16) burst: 3
29 | 1688842211 [VM0-0001]: Reading data 0000000000000000 @ 16b (CRC-4/8/16/16) burst: 3
30 | 1688842211 [VM0-0001]: Reading data 0000000000000000 @ 16b (CRC-4/8/16/16) burst: 3
31 | 1688842211 [VM0-0001]: Reading data 0000000000000000 @ 16b (CRC-4/8/16/16) burst: 3
```

SignalTap II Files

The reference design package also includes SignalTap® II files (.stp) that you can use with the SignalTap II Embedded Logic Analyzer to obtain information on the performance of this design. The SignalTap II files includes the key signals from the application logic. Figure 23 shows an example of the SignalTap II Embedded Logic Analyzer GUI.

The Qsys version of the reference design does not contain the SignalTap II circuitry or files.
For more information about the signals displayed in Figure 23 refer to the DDR and DDR2 High-Performance Controllers and ALTMEMPHY IP User Guide or the DDR3 High Performance Controller and ALTMEMPHY IP User Guide in volume 3 of the External Memory Interface Handbook and IP Compiler for PCI Express User Guide.

Figure 23. The SignalTap GUI

<table>
<thead>
<tr>
<th>Stratix IV GX and Arria II GX Hardware Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 8 lists the performance of the Stratix IV GX and Arria II GX FPGA development boards using this reference design with a motherboard that includes the Intel X58 Chipset. Table 8 lists the average throughput for a transfer size of 99,968 bytes and 20 iterations with a maximum write payload size of 256 bytes, a maximum read request size of 512 bytes, and a read completion size of 256 bytes using a clock multiplier unit (CMU) clock.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 8. Stratix IV GX and Arria II GX Performance - Intel X58 Chipset</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configuration</td>
</tr>
<tr>
<td>----------------</td>
</tr>
<tr>
<td>Stratix IV GX–Gen2 x4 64-bit interface in Qsys</td>
</tr>
<tr>
<td>Stratix IV GX–Gen1 x8 128-bit interface</td>
</tr>
<tr>
<td>Arria II GX–Gen1 x8 128-bit interface</td>
</tr>
</tbody>
</table>
Non-Qsys Design Limitations

This reference design has the following limitations:

- The DMA engine is not designed to transfer data with a size of less than 64 bytes.
- The DMA engine only transfers data in multiples of 64 bytes and the data must be aligned on a 64-byte address.
- The root port must use a DMA to access the SDRAM memory.

Document Revision History

Table 9 lists the revision history for this application note.

<table>
<thead>
<tr>
<th>Date</th>
<th>Version</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>February 2010</td>
<td>1.3</td>
<td>The design now uses the High-Performance SDRAM Controller II for DDR3 SDRAM.</td>
</tr>
<tr>
<td>December 2009</td>
<td>1.2</td>
<td>Updated to include the PCI Express hard IP implementation in an Arria II GX device using DDR2 memory. Added instructions for running a DMA simulation in ModelSim.</td>
</tr>
<tr>
<td>July 2009</td>
<td>1.1</td>
<td>Updated to use the PCI Express hard IP implementation in a Stratix IV GX device with DDR3 SDRAM.</td>
</tr>
<tr>
<td>August 2006</td>
<td>1.0</td>
<td>Initial release.</td>
</tr>
</tbody>
</table>