RapidIO Interoperability with TI 6488 DSP Reference Design

Introduction

The Altera® RapidIO interoperability reference design provides a sample interface between the Altera RapidIO MegaCore® function and the Texas Instruments TMS320TCI6488 Communications Infrastructure Digital Signal Processor (TI 6488 DSP or TI 6488). Altera offers this reference design to demonstrate the installation and operation of Altera's RapidIO MegaCore function with the TI 6488. The reference design enables you to evaluate the RapidIO MegaCore function for integration into an Altera FPGA.

In addition to demonstrating basic interoperability, the design includes support to measure link utilization at all data rates, for all supported packet sizes. The statistics support helps you to determine the optimal payload size for transfers across the Serial RapidIO link from the Stratix® IV GX device to the TI 6488 DSP for the three baud rates. Because the TI6488 supports only 1× link modes, the design tests only 1× modes.

The reference design has the following features:

- Directs the RapidIO MegaCore function to initiate the following RapidIO transaction types, targeted to the TI 6488:

  - `NWRITE`

  - `NWRITE_R`

  - `SWRITE`

  - `NREAD`

  - `Maintenance Write`

  - `Maintenance Read`

  - `Doorbell` messages

- Configures the FPGA with any 1× RapidIO MegaCore variation. The comprehensive set of design variations includes 1× mode at all data rates (1.250, 2.500, and 3.125 gigabaud (GBaud)) and all supported packet sizes.

- Supports testing simultaneous traffic in both directions, including transactions initiated by both the RapidIO MegaCore function and the TI 6488 DSP.

- Includes support for gathering throughput statistics.

- Includes support for automatic data integrity checking.

This application note demonstrates how to install and run the RapidIO interoperability reference design. It describes the system used for this design, tells you how to use the hardware and software, and reports the performance of the tests run using this design. This application note contains the following sections:

General Description

The reference design is a sample application that connects the Altera RapidIO MegaCore function to the TI 6488 DSP through the reference design interface circuitry. The design uses the Altera Stratix IV GX FPGA development board and the TMS320TCI6488 Evaluation Module (EVM). The Stratix IV GX FPGA development board includes a Stratix IV GX EP4SGX230 device. Spectrum Digital, Inc. (www.spectrumdigital.com) provides the TMS320TCI6488 EVM, which includes the Texas Instruments Code Composer Studio (CCS) Integrated Development Environment (IDE) to program and monitor the TMS320TCI6488 EVM. Altera provides software that implements each variation of the reference design on the FPGA and programs the EP4SGX230 device on the Stratix IV GX FPGA development board.

Reference Design Overview

Figure 1 is a diagram of the complete system.

Figure 1. Complete System for RapidIO Interoperability Reference Design
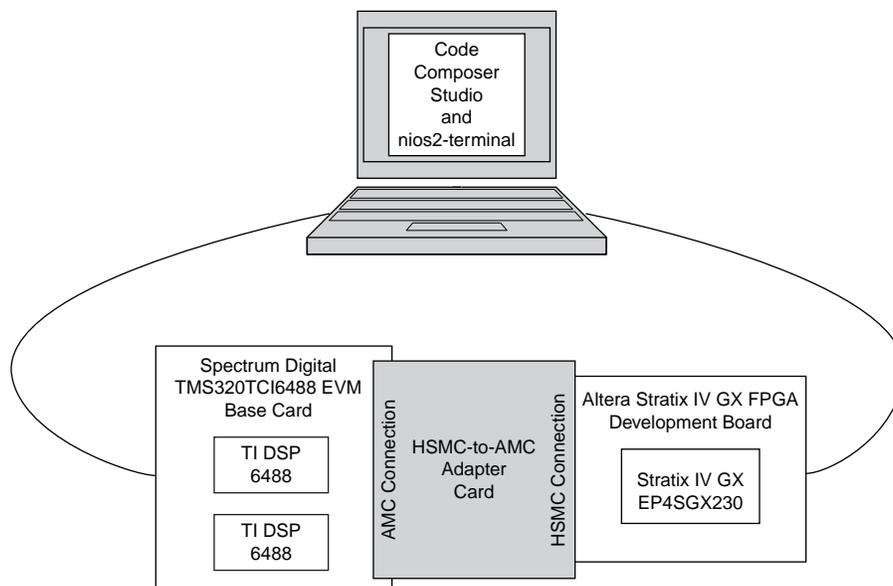
The C code running on the TI 6488 DSP performs the following tasks:

■ Configures the TI 6488 internal PLLs

■ Configures the TI 6488 SERDES blocks

■ Programs the TI 6488 RapidIO Command and Status registers (CSR)

- Responds to transactions initiated by the link partner (the Altera FPGA)

- Initiates write transactions to the link partner, focussing on performance

- Performs write-and-read operations to the Altera FPGA, checking for data integrity

- Initiates Doorbell messages to the Altera FPGA

- Displays the DDR2 memory space

- Checks the status of the RapidIO link

The nios2-terminal session allows you to perform the following equivalent tasks on the Altera FPGA:

- Programs the RapidIO CSRs

- Initiates transactions to the link partner (the TI DSP card)

- Responds to transactions initiated by the link partner

- Monitors data throughput on the link

- Monitors link status

Functional Description

A version of the design is available for each data rate of the RapidIO MegaCore function. The design for the 1× 1.250 Gbaud variation uses direct memory access (DMA) to transfer data from a local memory block to the RapidIO MegaCore I/O write Avalon-MM slave port. The designs for the other two RapidIO MegaCore function variations—the 1× variations at 2.500 and 3.125 Gbaud—use a packet generator with an Avalon® Memory-Mapped (Avalon-MM) master port to generate I/O bursts and send them to the RapidIO MegaCore I/O write Avalon-MM slave port.

The design can saturate the link with line rate traffic using the DMA approach only in the 1× 1.250 Gbaud design variation. A packet-generator approach allows the design to saturate the link for the other variations of the RapidIO protocol.

The following sections describe the DMA-based and the packet-generator-based versions of the design, as well as the TI 6488 DSP function and the clocks in all of the design variations.

DMA-Based RapidIO Design Implemented in FPGA Device

Figure 2 shows a top-level block diagram of the DMA-based RapidIO interoperability reference design implemented in the Altera FPGA.

Figure 2. DMA-Based Reference Design Block Diagram
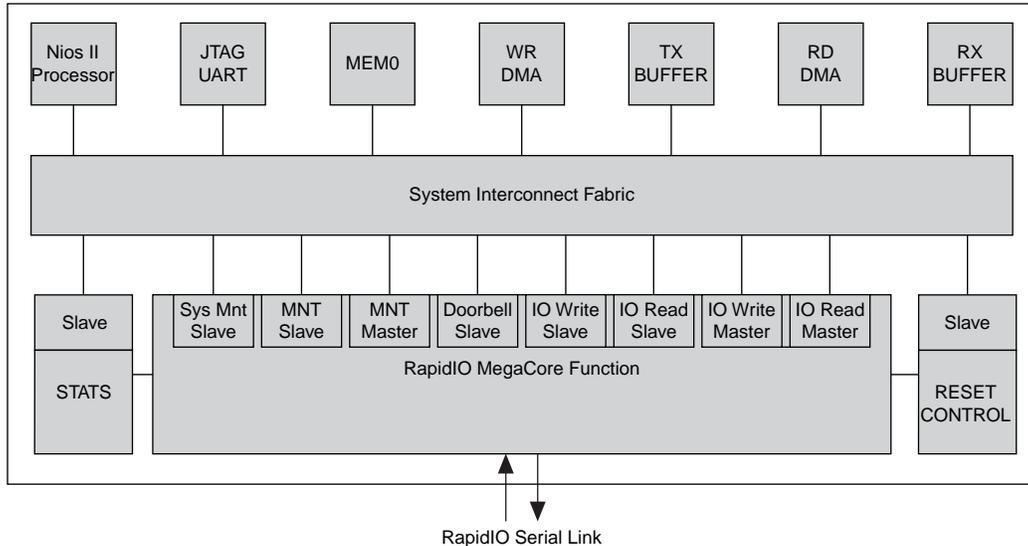
The system in Figure 2 is designed using SOPC Builder. The following sections describe the roles of the main system components in the reference design.

Nios II Embedded Processor

You can implement a Nios® II embedded processor in the FPGA device. The processor executes the RapidIO driver software included in this reference design. Using this driver, you can program the RapidIO MegaCore function and control the transactions that it performs. You can program the RapidIO MegaCore function to send transactions to and receive transactions from a remote link partner.

JTAG UART

This component provides the mechanism for the FPGA to communicate with the nios2-terminal. The nios2-terminal is the user interface to the RapidIO driver.

MEM0 Memory Segment

This memory stores the executable program. In this reference design, the program code is the RapidIO driver. After the program is compiled, it is stored in this memory and executed by the Nios II embedded processor.

WR DMA

This DMA component transfers data from the TX BUFFER to the RapidIO MegaCore function I/O write Avalon-MM slave port.

TX BUFFER Memory Segment

This memory stores the data to be transferred to the RapidIO MegaCore function I/O write Avalon-MM slave port. The WR DMA component uses this data for the payload in the different RapidIO write transactions.

RD DMA

This DMA component transfers data from the RapidIO MegaCore function I/O read Avalon-MM slave port to RX BUFFER memory. This memory supports the NREAD transactions initiated by the RapidIO MegaCore function. The RD DMA component receives the data read from a remote processing endpoint or link partner and stores it in RX BUFFER memory.

RX BUFFER Memory Segment

This memory stores the data read from a remote processing endpoint or link partner. The RD DMA component stores the data in this memory.

This memory is also addressable by a remote processing endpoint, which can write to and read from this memory segment. The RX BUFFER memory segment services the following transactions initiated by a remote processing endpoint and targeted to the FPGA:

■ NWRITE

■ SWRITE

■ NWRITE_R

■ NREAD

The buffer has 64 Kbytes of addressable memory.

STATS

This custom SOPC Builder component maintains the following statistics:

■ TX throughput

■ RX throughput

■ Packets transmitted

■ Bytes transmitted

■ Packets received

■ Bytes received

■ Packets not accepted

■ Packets cancelled

■ Packets retried

■ Packets with CRC errors

■ Packets dropped by transport layer

■ Symbol errors

■ Character errors

You can program this component to assert an interrupt every time its sample window expires. In response to the interrupt, the driver can read the statistics counters and display them in the nios2-terminal session.

☞ The 1× 1.250 Gbaud RapidIO MegaCore variation cannot display statistics dynamically while it is running the bidirectional data integrity test. To view statistics from a bidirectional data integrity test run on this RapidIO variation, you must run dit_stats after the run completes.

RESET CONTROL

This custom SOPC Builder component allows the user to execute a soft reset to the RapidIO MegaCore function.

RapidIO MegaCore Function

The RapidIO MegaCore function is the main component in this reference design. This component establishes a RapidIO link with the link partner. It also converts the transactions presented to it on the Avalon-MM interface to the corresponding RapidIO transactions and transmits them on the RapidIO serial link. It converts the RapidIO transactions from the RapidIO serial link to I/O burst transfers and presents these burst transfers to the corresponding Avalon-MM slave or master ports.

Packet-Based RapidIO Design Implemented in FPGA Device

Figure 3 shows a top-level block diagram of the packet-generator based RapidIO interoperability reference design implemented in the Altera FPGA. This version of the design is based on a custom Avalon-MM traffic generator.

Figure 3. Packet-Generator Based Reference Design Block Diagram
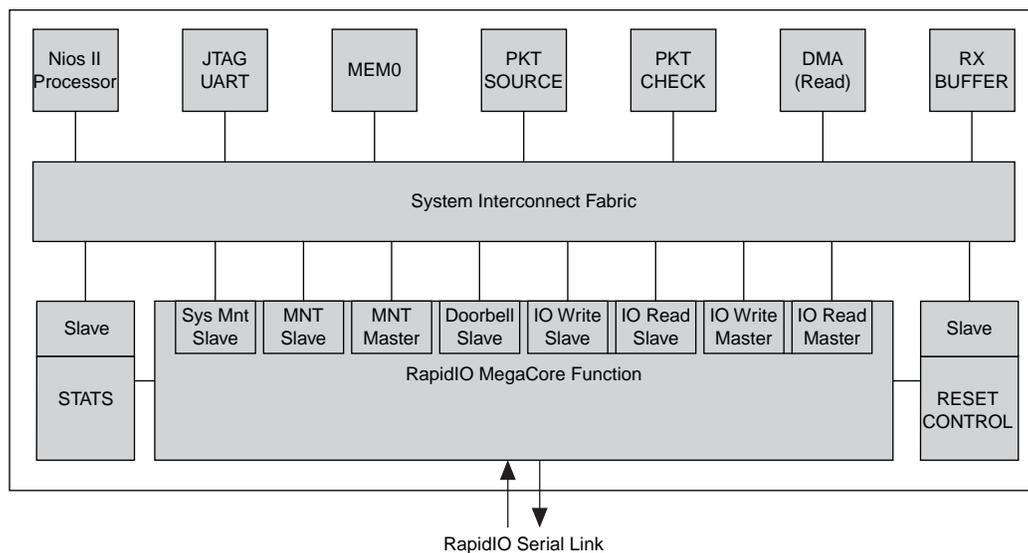
The pkt_source and pkt_check components appear in the packet-generator based design but not in the DMA-based design. All other system components function identically in the DMA-based and packet-generator based versions of the design.

The pkt_source component is a custom SOPC Builder component with an Avalon-MM master port capable of generating I/O burst transfers. In this design the transfers are sent to the RapidIO MegaCore I/O write Avalon-MM slave port.

During the data integrity tests, the pkt_source component pushes a packet descriptor to a packet-descriptor FIFO in the pkt_check component, for every write packet it transmits. The pkt_check component initiates NREAD transactions to the write addresses of the transmitted packets and compares the retrieved read data with the expected data.

The pkt_check component is a custom SOPC Builder component with an Avalon-MM master port capable of generating burst reads to the RapidIO MegaCore I/O read Avalon-MM slave port. This component contains a packet descriptor FIFO. When the pkt_check component detects that the FIFO is not empty, it pulls a packet descriptor from the FIFO and initiates a read burst to the RapidIO read Avalon-MM slave port. The read burst targets the address and data length that the packet descriptor specifies. When it receives the read data, the pkt_check component performs a data integrity check by comparing the retrieved read data with the packet-descriptor data, and then pulls the next packet descriptor from the FIFO.

TI 6488 DSP Function

NWRITE, NWRITE_R, SWRITE, and NREAD transactions initiated by the Altera RapidIO MegaCore target the DDR2 SDRAM connected to the TI 6488 DSP on the TI DSP card. The DDR2 SDRAM address space starts at address 0x80000000. The data integrity tests write to this address space and then perform NREAD transactions to read the contents of the write addresses, to check data integrity across the system.

To verify that data is being written in the DDR2 SDRAM, start the Memory Editor in the CCS IDE and read the contents of address 0x80000000 and beyond.

Clocks

Table 1 shows the relationship of the Avalon system clock rate to the data rate, the mode, and the transceiver reference clock.

Table 1. Clock Relationships in the RapidIO–TI 6488 Interoperability Reference Design

Data rate (Gbaud)	Mode	ALT4GXB Reference Clock Rate	Avalon System Clock
1.250	1x	125 MHz	40 MHz
2.500	1x	125 MHz	75 MHz
3.125	1x	125 MHz	84 MHz

The reference clock is dictated by the clocks available on the Stratix IV GX FPGA development board. The 125 MHz clock is the reference clock for all design variations.

Using the Reference Design

The following sections describe how to set up and use the reference design:

■ "Hardware Requirements"

■ "Software Requirements" on page 10

Hardware Requirements

The reference design application requires the following hardware:

■ One computer running the Windows XP operating system, capable of running the Code Composer Studio IDE v3.3 and a nios2-terminal session simultaneously

■ Altera Stratix IV GX FPGA development board (contains an Altera EP4SGX230 device)

■ Spectrum Digital Dual TMS320TCI6488 AMC Mezzanine Board

■ Altera USB-Blaster™ download cable

■ HSMC-to-AMC adapter card

The performance results reported in "Performance Summary" on page 38 were obtained using an assembly version 509310 Rev D Dual TMS320TCI6488 Mezzanine Board and a Rev A Stratix IV GX FPGA development board.

Figure 4 shows the Stratix IV GX FPGA development board connected to the TI DSP card.

Figure 4. Altera Stratix IV GX FPGA Development Board and Spectrum Digital TMS320TCI6488 AMC Mezzanine Board
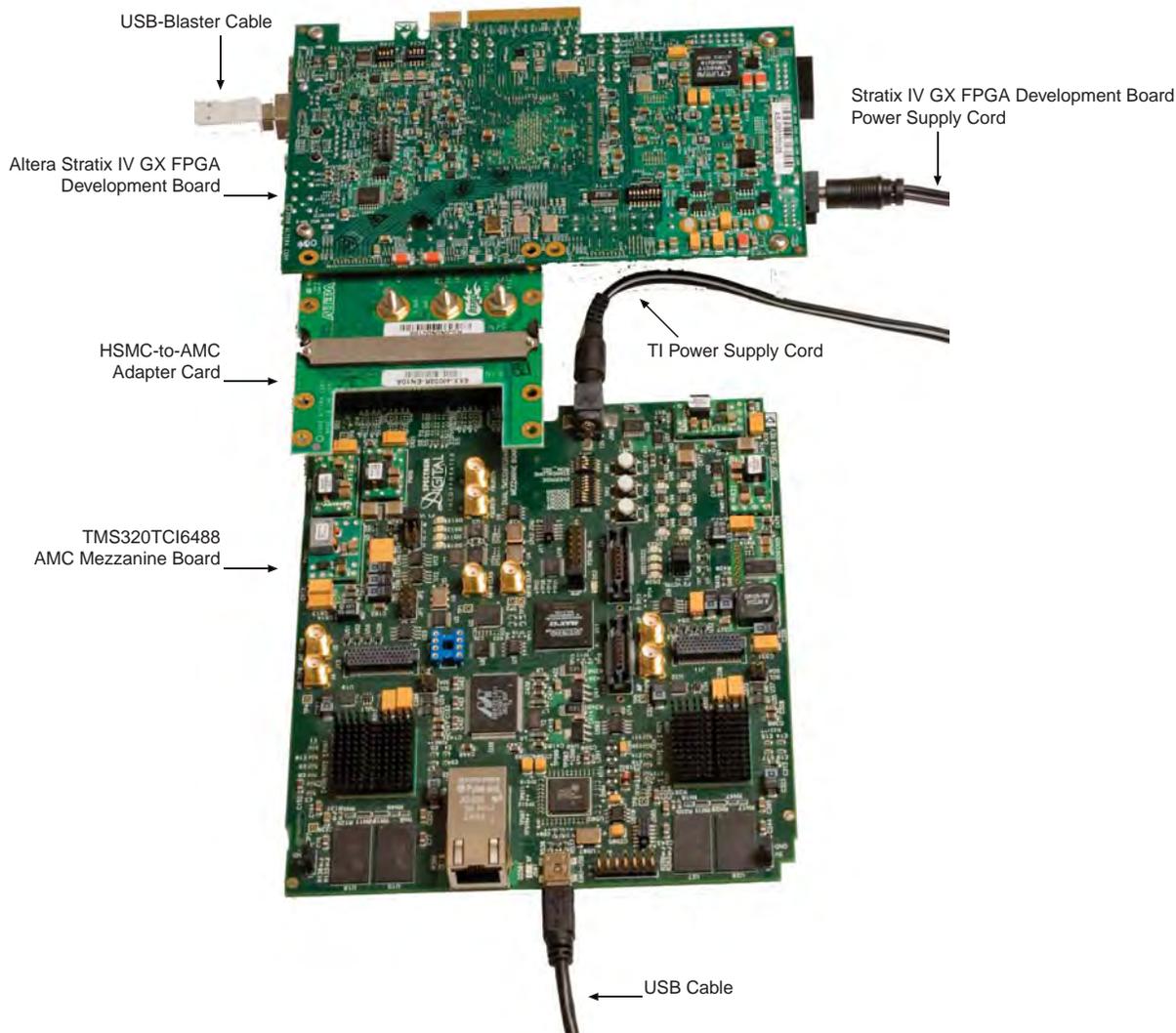
Figure 4 also shows the following connectors:

■ Altera USB-Blaster download cable

■ USB Cable

■ TI 12-volt power supply cord

■ Altera 16-volt power supply cord

The Altera USB-Blaster download cable is used to configure the FPGA on the Stratix IV GX FPGA development board and also to communicate between a nios2-terminal session and the FPGA. The nios2-terminal session is the standard input, output, and error location (stdio) for all of the tests performed in this reference design. It is the user interface to the RapidIO driver.

The Code Composer Studio IDE communicates with the TI DSP card through the USB cable. The TI 12-volt power supply provides power to the TI DSP card and the 16-volt power supply provides power to the Stratix IV GX FPGA development board.

Software Requirements

The reference design application requires the following software:

- Altera Quartus® II v9.0 software, including the USB-Blaster driver, SOPC Builder, and the RapidIO MegaCore function

- Altera Nios II Embedded Design Suite (EDS) v9.0

- TI Code Composer Studio IDE v3.3 (CCS)

Reference Design Installation

This section describes how to install the reference design.

Downloading the Package

All of the files necessary for this reference design are included in the **altera_6488_srio_ref_design.zip** file. This file is available for you to download from the Serial RapidIO To TI 6488 DSP Reference Design web page.

Extracting the Altera FPGA Package Files

Unzip the **altera_6488_srio_ref_design.zip** file in the working directory you designated for this project. After you unzip the file, your working directory contains two subdirectories named **altera_pcie_srio_ref_design** and **altera_ti_srio_ref_design**. The **altera_pcie_srio_ref_design** directory contains the files for programming the Altera FPGA. Figure 5 shows the **altera_pcie_srio_ref_design** directory structure.

Figure 5. altera_pcie_srio_ref_design Directory Structure
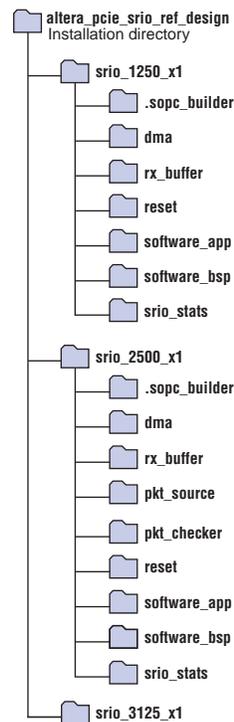
Package Content Description

The **altera_pcie_srio_ref_design** directory has three subdirectories, **srio_<*rate*>_×1**, one for each RapidIO ×1 variation.

The **srio_1250_×1** directory contains the FPGA design for the 1.250 Gbaud 1x RapidIO MegaCore function. This design uses DMA to transfer data from a local memory block to the RapidIO MegaCore I/O write Avalon-MM slave port. These data transfers are the basis for the NWRITE, NWRITE_R, and SWRITE transactions.

The **srio_2500_×1** and **srio_3125_×1** directories contain the FPGA designs for the remaining RapidIO MegaCore function variations—the 1× variations at 2.500 and 3.125 Gbaud. These designs use a packet generator with an Avalon-MM master port to generate I/O burst transfers and send them to the RapidIO MegaCore I/O write Avalon-MM slave port. These bursts are the basis for the NWRITE, NWRITE_R, and SWRITE transactions. The design cannot saturate the link with line rate traffic in these design variations using the DMA approach. The packet-generator approach allows the design to saturate the link for this mode and these rates of the RapidIO protocol.

Each **srio_<*rate*>_<*mode*>** directory contains the SRAM Object File (**.sof**) for programming the Altera FPGA with the relevant design. In addition, each of these directories contains all of the files necessary to regenerate the complete design, including the following files:

- **srio_<*rate*>_<*mode*>.qpf**
- **srio_<*rate*>_<*mode*>.qsf**
- **srio_<*rate*>_<*mode*>_sys.ptf**
- **srio_<*rate*>_<*mode*>_sys.sopc**
- **srio_<*rate*>_<*mode*>_top.v**

The **srio_<*rate*>_<*mode*>_top.v** file is a top-level wrapper for the SOPC design. It implements the clocking methodology. For further details about the clocking methodology, refer to .

Each **srio_<*rate*>_<*mode*>** directory contains the relevant subdirectories — for the DMA-based or the packet-generator based design — from the following set:

**Table 2.** Subdirectories Contained in srio_<*rate*>_<*mode*> Directory   (Part 1 of 2)

Directory Name	In DMA or Packet Based Case?	Description
.sopc_builder	Both	Contains Peripheral Template (**.ptf)** files used by SOPC Builder. These files list the components available in SOPC Builder.
dma	Both	Contains the Verilog HDL code for the DMA controller used in this reference design. It also contains the TCL file for SOPC Builder that describes the function and parameters of the component.
rx_buffer	Both	Contains the Verilog HDL code for the Avalon-MM traffic sink, and the TCL file for SOPC Builder.
pkt_source	Packet-based	Contains the Verilog HDL code for the Avalon-MM traffic generator, and the TCL file for SOPC Builder.

Table 2. Subdirectories Contained in srio_<*rate*>_<*mode*> Directory   (Part 2 of 2)

Directory Name	In DMA or Packet Based Case?	Description
pkt_checker	Packet-based	Contains the Verilog HDL code for the data integrity checker, including a packet-descriptor FIFO, an NREAD transaction generator, and a data comparison block, as well as the TCL file for SOPC Builder.
reset	Both	Contains the Verilog HDL code for the RESET CONTROL block, and the TCL file for SOPC Builder.
software_app	Both	Contains the following three files. ■ **create-this-app** – script used to compile the driver ■ **srio_main_full.c** – C code implementing the RapidIO driver ■ **srio_regs.h** – mnemonics for the RapidIO MegaCore internal registers
software_bsp	Both	Contains the file **create-this-bsp**. This script creates the HAL drivers that are used by the application, in this case the RapidIO driver.
srio_stats	Both	Contains the Verilog HDL code for the STATS component, and the TCL file for SOPC builder.

Extracting the TI 6488 DSP Package Files

After you download the reference design zip file, as described in "Downloading the Package" on page 10, and unzip the file, your working directory contains an **altera_ti_srio_ref_design** subdirectory. This directory contains the files for programming the TI 6488 DSP. Figure 6 shows the **altera_ti_srio_ref_design** directory structure.

Figure 6. altera_ti_srio_ref_design Directory Structure
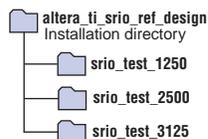
Package Content Description

The **altera_ti_srio_ref_design** directory contains three subdirectories, one for each serial RapidIO data rate.

Each directory contains the following files:

■ **lnk.cmd**

■ **main_unidirectional.c**

■ **main_bidirectional_perf.c**

■ **main_bidirectional_dit.c**

■ **main_tx_dbell.c**

■ **srio_test_<*rate*>.pjt**

Table 3 describes these files.

Table 3. Files in altera_ti_srio_ref_design subdirectories

File	Description
lnk.cmd	The linker command file for the TI 6488 DSP.
main_unidirectional.c **main_bidirectional_perf.c** **main_bidirectional_dit.c** **main_tx_dbell.c**	The code to program the TI 6488 DSP.
srio_test_<*rate*>.pjt	The Code Composer Studio project file. It contains all of the relevant information for compiling the **.c** files.

The **srio_test_<*rate*>.pjt** file includes paths to targeted libraries. You must edit this file and modify the paths shown in Example 1 to reflect the location of your libraries.

Example 1. Paths in the srio_test_<*rate*>.pjt File

```
ProjectDir="C:\srio_ref_design\PCIe_a_6488\srio_test_<rate>\"
Source="..\..\..\CCStudio_v3.3\C6000\cgtools\lib\rts64plus.lib"
Source="C:\CCStudio_v3.3\C6000\csl_c6488\csl_c6488.lib"
Options=-g -pdv -fr"$(Proj_dir)\Debug" \
  -i"c:\srio_ref_design\PCIe_a_6488\ti6488_srio_source\default_package\csl_c6488\inc" \
  -d"RATE_<rate>" -d"_DEBUG" -d"CHIP_64XX" -mv6400+
```

Preparing to Run the Applications

To run the application, you must install the required software, connect the hardware, program the Altera FPGA, select your application, program the TI 6488 DSP, and run the selected performance, data integrity, and Doorbell message tests. The following sections teach you how to connect the hardware, program the Altera FPGA, select your application, and program the TI 6488 DSP.

Connecting the Hardware

Before connecting the hardware, you must install the required software listed in "Software Requirements" on page 10.

To connect the hardware, perform the following steps:

1. Install CCS v3.3.

2. Power up and initialize the TI DSP card according to the instructions in the *TMS320TCI6488 EVM Quick Start Installation Guide*, supplied with your EVM.

3. Disconnect the power to the TI DSP card.

4. Remove the TI USB cable from the TI DSP card.

5. Connect the Stratix IV GX FPGA development board to the TI DSP card using the HSMC-to-AMC connector.

6. Reconnect the TI USB cable to the TI DSP card.

7. Connect the USB-Blaster cable to the Stratix IV GX FPGA development board.

8. Provide power to the Stratix IV GX FPGA development board.

9. Provide power to the TI DSP card.

10. Turn on the Stratix IV GX FPGA development board.

Programming the Altera FPGA for the 1× 1.250-Gbaud Design Variation
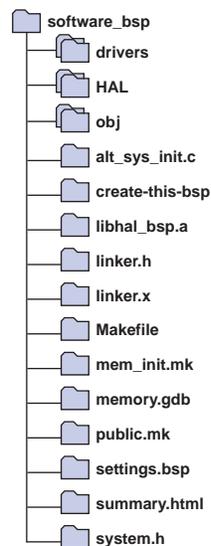
To program the Altera FPGA for the 1× 1.250-Gbaud variation of the RapidIO interoperability reference design, perform the following steps:

1. On the Windows Start menu, point to **All Programs** > **Altera** > **Nios II EDS** *<version_number>*, and click **Nios II** *<version_number>* **Command Shell**, to start a Nios II command shell.

2. Navigate to the directory **altera_pcie_srio_ref_design\srio_1250_x1**.

3. Navigate to the directory **software_bsp**. The **create-this-bsp** script file should be the only file in this directory.

4. To execute the script that builds all of the HAL drivers required by this reference design, type the following command:

   ```
   ./create-this-bsp ↵
   ```

   The **software_bsp** directory should now contain the subdirectories and files shown in Figure 7.

Figure 7. software_bsp Directory Structure After Running create-this-bsp Script


```
software_bsp
├── drivers
├── HAL
├── obj
├── alt_sys_init.c
├── create-this-bsp
├── libhal_bsp.a
├── linker.h
├── linker.x
├── Makefile
├── mem_init.mk
├── memory.gdb
├── public.mk
├── settings.bsp
├── summary.html
└── system.h
```

5. Navigate to the directory **..\software_app**.

   The directory contains the following three files:

   ■ **srio_regs.h**

   ■ **srio_main_full.c**

   ■ **create-this-app**

6. To execute the script that compiles the driver software in **srio_main_full.c**, type the following command:

   ```
   ./create-this-app ↵
   ```

☞ The **srio_regs.h** file contains mnemonics for the Serial RapidIO registers in the Altera RapidIO MegaCore function.

The **software_app** directory should now contain the subdirectories and files shown in Figure 8.

Figure 8. software_app Directory Structure After Running create-this-app Script


```
software_app
├── obj
├── create-this-app
├── Makefile
├── srio_main_full.c
├── srio_regs.h
├── srio_test.elf
├── srio_test.map
└── srio_test.objdump
```
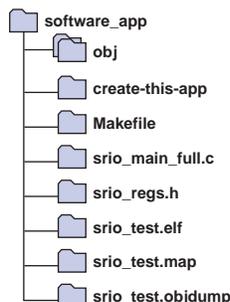
If no compilation errors occur when the `create-this-app` script executes, you can now program the FPGA and run the RapidIO MegaCore driver.

To program the FPGA, download the software image, and start a nios2-terminal session, perform the following steps:

1. To program the FPGA, in the command shell, type the following command:
   `nios2-configure-sof -d 1 ../srio_1250_x1.sof` ↵

2. To download the software image, type the following command:
   `nios2-download --device=1 -g srio_test.elf` ↵

   The Altera RapidIO MegaCore function is downloaded to the FPGA.

3. To start a nios2-terminal session, type the following command:
   `nios2-terminal --device=1` ↵

   A nios2-terminal session opens, allowing you to communicate with the Nios II processor on your FPGA.

Before you execute any of the RapidIO interoperability reference design tests, you must bring up and program the TI 6488 DSP. The following sections describe this procedure.

Programming the Altera FPGA for the Remaining RapidIO Design Variations

To program the Altera FPGA for any other variation of the RapidIO interoperability reference design, follow the procedure in "Programming the Altera FPGA for the 1× 1.250-Gbaud Design Variation" on page 14, with the following exceptions:

■ Replace the path **altera_pcie_srio_ref_design\srio_1250_x1** with the path **altera_pcie_srio_ref_design\srio_<rate>_x1**.

■ Replace the **.sof** file name **srio_1250_x1.sof** with **srio_<rate>_x1.sof**.

Selecting the Program to Run on the TI 6488 DSP

This reference design provides four programs that you can load on the TI 6488 DSP. They are found in the following four files:

- **main_unidirectional.c**

- **main_bidirectional_perf.c**

- **main_bidirectional_dit.c**

- **main_tx_dbell.c**

The program in **main_unidirectional.c** configures the TI 6488 DSP as a slave only. This program is used to measure performance for data flowing from the FPGA to the TI 6488 DSP and to test Doorbell message passing from the FPGA to the TI 6488 DSP.

The program in **main_bidirectional_perf.c** configures the TI 6488 DSP to be both a slave and a master. In this configuration, the TI DSP processes incoming read and write transactions and initiates write transactions to the FPGA.

The program in **main_bidirectional_dit.c** configures the TI 6488 DSP to be both a slave and a master. In this configuration, the TI DSP processes incoming read and write transactions. It also initiates write transactions to the FPGA and reads data from the write location for data integrity checking.

☞ The program in **main_bidirectional_dit.c** cannot saturate the link. Therefore, it should not be used to measure performance.

The program in **main_tx_dbell.c** configures the TI 6488 DSP to transmit Doorbell messages to the RapidIO core on the Altera FPGA.

Before you can load and run the TI 6488 DSP driver, you must edit the **srio_test_<*rate*>.pjt** file to specify the program you want. The **srio_test_<*rate*>.pjt** file provided in the reference design **.zip** file contains lines to source all four programs. Open the file in a text editor to uncomment the line that corresponds to the program you want, and comment out the others.

Example 2 shows the relevant source lines in a **srio_test_<*rate*>.pjt** file that specifies the program in the **main_unidirectional.c** file.

Example 2. Section of srio_test_<*rate*>.pjt File Specifying That main_unidirectional.c Be Compiled

```
Source="main_unidirectional.c"
#Source="main_bidirectional_dit.c"
#Source="main_bidirectional_perf.c"
#Source="main_tx_dbell.c"
```

Running the CCStudio IDE v3.3

After you edit the **.pjt** file to specify the program you want, you can load and run the TI 6488 DSP driver. This section describes how to begin or reset a CCStudio v3.3 (CCS) session in which to load and run the TI 6488 DSP driver for any RapidIO variation.

To prepare the CCS IDE session to run a new project, perform the following steps:

1. If a TI 6488 DSP program is running in a CCS session, perform the following steps to stop the program and close the previous session completely:

    a. In the CCS driver window for the TMS320TCI6488 EVM, on the left side, click the **Halt** icon to stop the program. Figure 9 shows the icon.

    b. On the Project menu, click **Close**.

    c. On the Debug menu, click **Disconnect**.

    d. On the File menu, click **Close Session**.

    e. In the Parallel Debug Manager, on the File menu, click **Exit**. Figure 10 shows the Parallel Debug Manager.

Figure 9. Code Composer Studio Window With Halt Icon Active
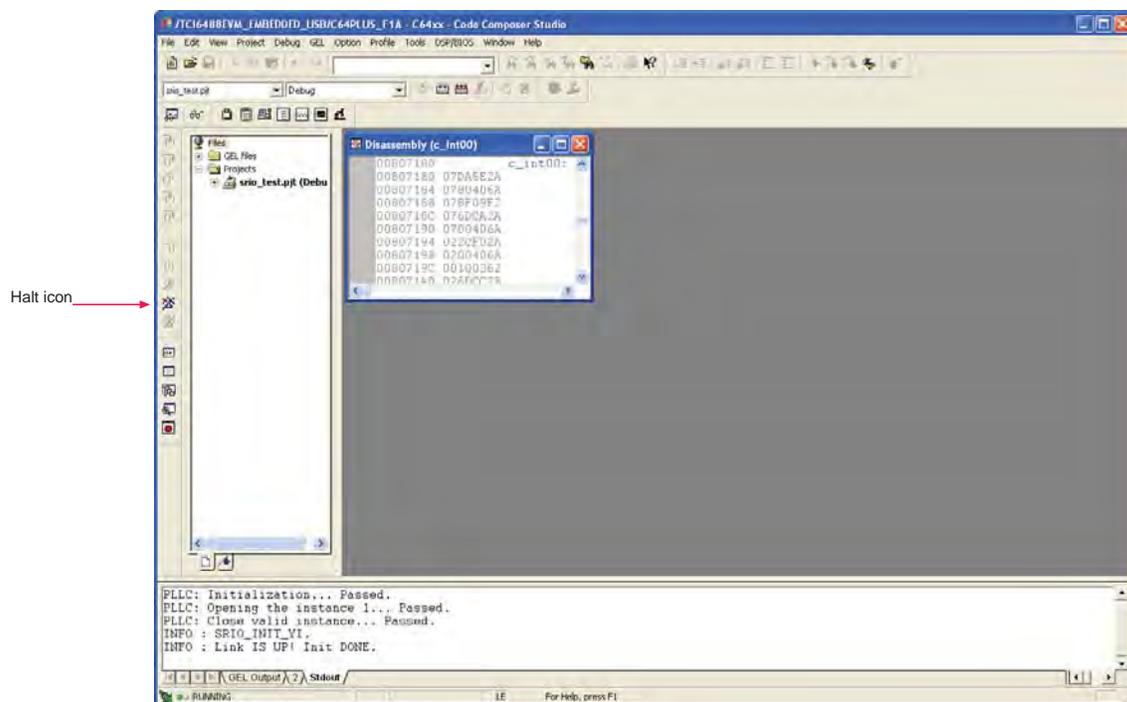
2. Double-click the **TCI6488 EVM CCStudio v3.3** icon. The Parallel Debug Manager window appears.

3. Under **System**, under **TCI6488EVM_EMBEDDED_USB**, double-click **C64PLUS_F1A** to start a CCS session on core A of the F1 TI 6488 DSP on the EVM board. Figure 10 shows the Parallel Debug Manager after you select this CPU.

4. On the Debug menu, click **Connect**. A connection is established between the driver and the F1 TI 6488 DSP core A.

Figure 10. Parallel Debug Manager After CPU Selection
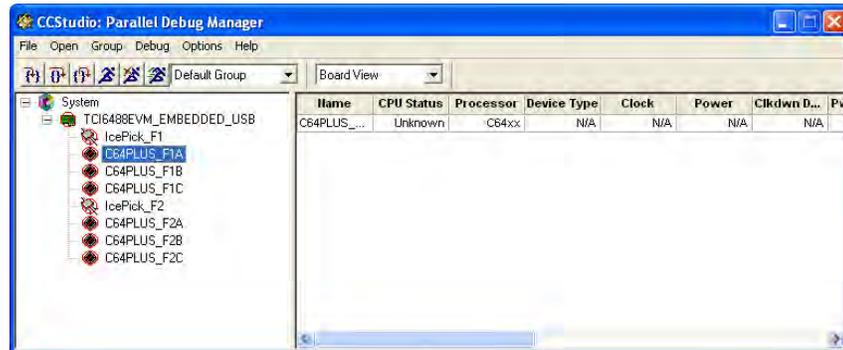
Programming the TI 6488 DSP for the 1× 1.250-Gbaud Design Variation

After you edit the **.pjt** file to specify the program you want and reset your CCS session, you can load and run the TI 6488 DSP driver. This section describes how to load and run the TI 6488 DSP driver for the 1× 1.250-Gbaud design variation.

To load and run this design variation, perform the following steps:

1. On the Project menu, click **Open**.

2. In the **Project Open** dialog box, navigate to the **srio_test_1250** directory.

3. Select the file **srio_test_1250.pjt**.

4. Click **Open**.

5. In the File View window, in the **Projects** directory, select the **srio_test_1250.pjt** file.

6. Right-click the file name and click **Clean**.

7. With the file name still highlighted, click **Build**. The code to configure the TI 6488 DSP compiles.

8. On the File menu, click **Load Program**.

9. In the Load Program window, navigate to the **Debug** directory.

10. Select **srio_1250_test.out**.

11. Click **Open**.

12. In the CCS driver window, on the left side, click the **Run** icon to run the program. Figure 11 shows the icon.

In the stdout window panel, messages embedded in the code are displayed. The final message displayed states that the link is established.

Figure 11 shows the **Run** icon and the final message that states the link is established.

Figure 11. Code Composer Studio Window
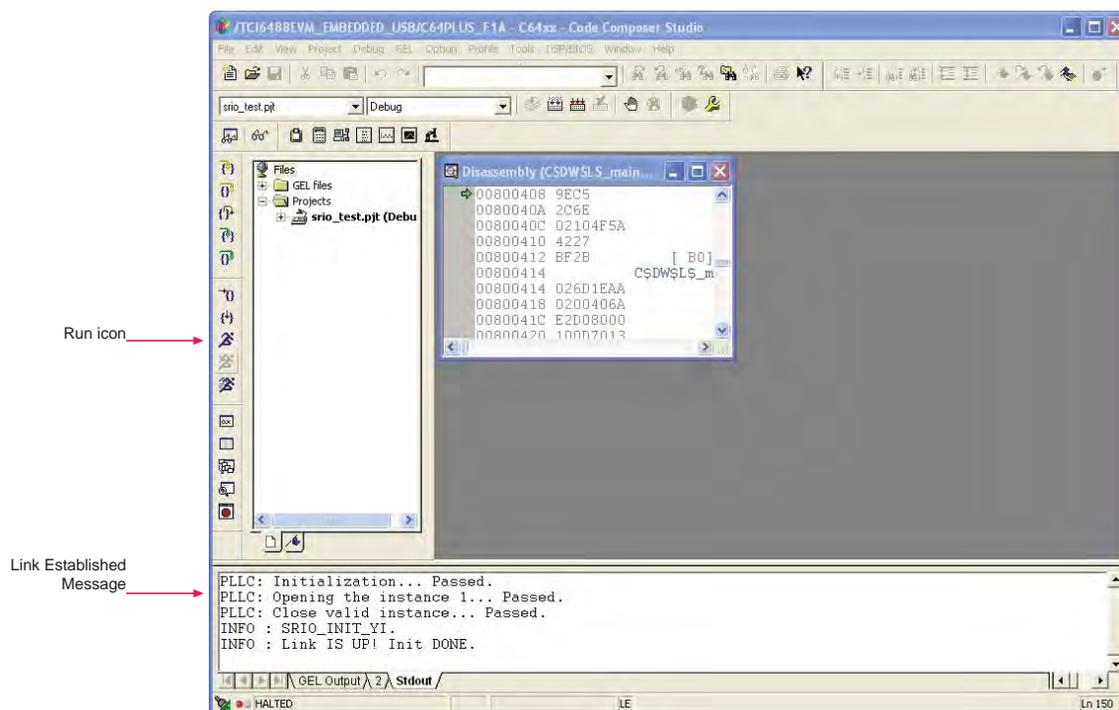
Programming the TI 6488 DSP for the Remaining Design Variations

To program the TI 6488 DSP for any other variation of the RapidIO interoperability reference design, follow the procedure in "Programming the TI 6488 DSP for the 1× 1.250-Gbaud Design Variation" on page 18, with the following exceptions:

- Replace the directory **srio_test_1250** with the directory **srio_test_<*rate*>**.

- Replace the **.pjt** file name **srio_test_1250.pjt** with **srio_test_<*rate*>.pjt**.

- Replace the **.out** file name **srio_test_1250.out** with **srio_test_<*rate*>.out**.

Running the Applications

After you configure the Altera FPGA and load your selected application in the TI 6488 DSP, you can run the application, which is one of four kinds of tests.

The steps to run the performance and data integrity tests are different for the RapidIO MegaCore variation implemented in a DMA-based system—the 1× 1.250-Gbaud variation—and the remaining variations, which are implemented in a packet-generator based system.

The following sections teach you how to run the four application programs.

Running the Unidirectional Performance Test

The following sections teach you how to run the unidirectional performance test.

Running the Unidirectional Performance Test for the 1× 1.250-Gbaud Design Variation

Before you can run the unidirectional performance test for the 1× 1.250-Gbaud design variation, you must configure the Altera FPGA and load the unidirectional performance test if you have not already done so. To configure the Altera FPGA and load the TI 6488 DSP with the unidirectional performance test for the 1× 1.250-Gbaud variation, perform the following steps:

1. If you have reprogrammed the FPGA since programming it for this design variation, follow the instructions in "Programming the Altera FPGA for the 1× 1.250-Gbaud Design Variation" on page 14.

    ☞ If you have reprogrammed the FPGA since programming it for this design variation, but have already executed `create-this-bsp` and `create-this-app` for this variation, then you need only perform steps 1 and 2 on page 15.

2. Edit the **srio_test_1250.pjt** file to specify that the **main_unidirectional.c** file be compiled.

3. Follow the instructions in "Running the CCStudio IDE v3.3" on page 16.

4. Follow the instructions in "Programming the TI 6488 DSP for the 1× 1.250-Gbaud Design Variation" on page 18.

To run the unidirectional performance test, return to the nios2-terminal session you opened in step 1. For a list of supported RapidIO driver commands, type h.

The following sequence of commands, explained in Table 4, programs the RapidIO MegaCore function, generates traffic to the TI 6488 DSP, and monitors performance. The default init setting generates NWRITE transactions.

```
stop
r
rate 1250
clk 40
pload 256
init
start
stats
z
stop
```

Table 4 explains the individual commands in the sequence.

Table 4. nios2-terminal Commands for NWRITE Performance Test on 1× 1.250-Gbaud RapidIO Variation   (Part 1 of 2)

Command	Description
stop	Stops the DMA transfers from the WR DMA block to the RapidIO I/O write Avalon-MM slave port. Start your command sequence with this command to ensure you've stopped any previous test activity, and end your command sequence with this command to stop your current test activity.
r	Resets the RapidIO MegaCore function.
rate 1250	Sets the rate to 1.250 Gbaud.
clk 40	Tells the driver that the Avalon-MM system clock runs at 40 MHz. For a list of RapidIO MegaCore function variations and clock rates and corresponding Avalon-MM system clock frequencies, refer to Table 1 on page 7. The clk value you program is used for calculating throughput.

Table 4. nios2-terminal Commands for NWRITE Performance Test on 1× 1.250-Gbaud RapidIO Variation   (Part 2 of 2)

Command	Description
pload 256	Programs the Write DMA to burst 256-byte payloads. The allowed values are multiples of 8 bytes, with minimum value 8 and maximum value 256.
init	Initializes a subset of the CSRs in the RapidIO MegaCore function. For information about the specific configuration registers that are programmed, refer to the file **srio_main_full.c**.
start	Starts the DMA transfers from the WR DMA block to the RapidIO I/O write Avalon-MM slave port.
stats	Enables the statistics-gathering operation.
z	Stops the statistics-gathering operation.

Figure 12 shows a sample unidirectional statistics-gathering cycle for this RapidIO variation.

Figure 12. Sample Unidirectional Statistics-Gathering Cycle at 1× 1.250 Gbaud

If you type the command dit in the command sequence instead of the command start, you enable data integrity checking. In this test, the FPGA initiates write commands, as in the regular unidirectional performance test. However, in the dit test, the FPGA also initiates read commands to test the success of the write operations.

During the dit test, the nios2-terminal displays a running count of packets transmitted and checked. The 1× 1.25-Gbaud RapidIO MegaCore variation does not support dynamic statistics gathering. After the packets are transmitted and checked, the Serial RapidIO> prompt is available again. To view a summary of the packet activity during this test, type dit_stats at the Serial RapidIO> prompt.

Table 5 provides information about these data-integrity checking commands.

Table 5. nios2-terminal Commands for Data Integrity Testing

Command	Description
dit N	Runs a data integrity test. The dit command takes a decimal number argument that specifies the number of packets that the FPGA writes to the TI DSP. In the data integrity test, the FPGA writes each packet and then reads the data back to confirm it was written correctly.
dit_stats	Displays a summary of statistics gathering during the completed run. You must use this command to display the statistics after a data integrity test run on the 1× 1.250 Gbaud RapidIO variation, because this RapidIO MegaCore variation cannot display statistics dynamically while it is running a data integrity test.

Figure 13 shows a sample unidirectional statistics-gathering cycle for this RapidIO variation, after the command dit 100 is inserted in the command sequence in place of the command start. The dit_stats display shows that read data is received by the RapidIO MegaCore function for data integrity checking.

Figure 13. Sample Unidirectional Statistics With Data Integrity Testing at 1× 1.250 Gbaud

Running the Unidirectional Performance Test for the 1× 2.500-Gbaud Design Variation

Before you can run the unidirectional performance test for the 1× 2.500-Gbaud design variation, you must configure the Altera FPGA and load the unidirectional performance test if you have not already done so. To configure the Altera FPGA and load the TI 6488 DSP with the unidirectional performance test for the 1× 2.500-Gbaud variation, perform the following steps:

1. If you have reprogrammed the FPGA since programming it for this design variation, follow the instructions in "Programming the Altera FPGA for the Remaining RapidIO Design Variations" on page 15.

   ☞ If you have reprogrammed the FPGA since programming it for this design variation, but have already executed create-this-bsp and create-this-app for this variation, then you need only perform steps 1 and 2 on page 15 with the **.sof** file name substitution indicated in "Programming the Altera FPGA for the Remaining RapidIO Design Variations" on page 15.

2. Edit the **srio_test_2500.pjt** file to specify that the **main_unidirectional.c** file be compiled.

3. Follow the instructions in "Running the CCStudio IDE v3.3" on page 16.

4. Follow the instructions in "Programming the TI 6488 DSP for the Remaining Design Variations" on page 19.

To run the unidirectional performance test, return to the nios2-terminal session you opened in step 1. For a list of supported RapidIO driver commands, type h.

The following sequence of commands, explained in Table 6, programs the RapidIO MegaCore function, generates traffic to the TI 6488 DSP, and monitors performance. The default `init` setting generates `NWRITE` transactions.

```
stop
r
rate 2500
mode 1
gap 4
clk 75
pload 256
init
start
stats
z
stop
```

Table 6 explains the individual commands in the sequence:

Table 6. nios2-terminal Commands for NWRITE Performance Test on 1× 2.500-Gbaud RapidIO Variation

Command	Description
stop	Stops the packet generator. Start your command sequence with this command to ensure you've stopped any previous test activity, and end your command sequence with this command to stop your current test activity.
r	Resets the RapidIO MegaCore function.
rate 1250	Sets the rate to 2.500 Gbaud.
mode 1	Sets the mode. This interoperability reference design supports only mode 1, because the TI 6488 DSP supports only ×1 RapidIO variations.
gap 4	Sets the gap between Avalon-MM write bursts to the RapidIO MegaCore I/O write Avalon-MM slave port to specified number of Avalon-MM system clock cycles. Smaller gap values cause the packet generator to push traffic to the RapidIO MegaCore I/O write Avalon-MM slave port with shorter delays between bursts, increasing throughput.
clk 75	Tells the driver that the Avalon-MM system clock runs at 75 MHz. For a list of RapidIO MegaCore variations and clock rates and corresponding Avalon-MM system clock frequencies, refer to Table 1 on page 7.
pload 256	Programs the packet generator to burst 256-byte payloads. The allowed values are multiples of 8 bytes, with minimum value 8 and maximum value 256.
init	Initializes a subset of the CSRs in the RapidIO MegaCore function. For information about the specific configuration registers that are programmed, refer to the file **srio_main_full.c**.
start	Enables the packet generator to start transmitting traffic to the RapidIO MegaCore I/O write Avalon-MM slave port.
stats	Enables the statistics-gathering operation.
z	Stops the statistics-gathering operation.

Figure 14 shows a sample unidirectional statistics-gathering cycle for this RapidIO variation.

Figure 14. Sample Unidirectional Statistics-Gathering Cycle at 1× 2.500 Gbaud

If you type the command `dit` in the command sequence instead of the command `start`, you enable data integrity checking by the pkt_check component. In this test, the FPGA initiates write commands, as in the regular unidirectional performance test. However, in the dit test, the FPGA also initiates read commands to test the success of the write operations. Table 5 on page 21 provides information about this data-integrity checking command.

Figure 15 shows a sample unidirectional statistics-gathering cycle for this RapidIO variation, after the command `dit 80000000` is inserted in the command sequence before the command `stats`. The display shows that read data is received by the RapidIO MegaCore function for data integrity checking.

Figure 15. Sample Unidirectional Statistics With Data Integrity Testing at 1× 2.500 Gbaud

Running the Unidirectional Performance Test for the 1× 3.125-Gbaud Design Variation

Before you run the unidirectional performance test for the 1× 3.125-Gbaud design variation, you must program the FPGA for the correct variation and program the TI 6488 DSP for the unidirectional performance test and the correct rate setting.

Program and run the 1× 3.125-Gbaud variation according to the instructions in "Running the Unidirectional Performance Test for the 1× 2.500-Gbaud Design Variation", with the appropriate modifications. When you run the program, specify the correct `rate` (3125), `mode` (1), and `clk` (84), and specify the `gap`. Refer to Table 1 on page 7 for the correct `clk` setting for this rate. The `gap` setting is measured in clock cycles and can remain the same for all packet-generator based variations.

Running the Bidirectional Performance Test

The following sections teach you how to run the bidirectional performance test.

Running the Bidirectional Performance Test for the 1× 1.250-Gbaud Design Variation

Before you can run the application, you must configure the Altera FPGA and load the bidirectional performance test. To configure the Altera FPGA and load the TI 6488 DSP with the bidirectional performance test for the 1× 1.250-Gbaud variation, perform the following steps:

1. If you have reprogrammed the FPGA since programming it for this design variation, follow the instructions in "Programming the Altera FPGA for the 1× 1.250-Gbaud Design Variation" on page 14.

    ☞ If you have reprogrammed the FPGA since programming it for this design variation, but have already executed `create-this-bsp` and `create-this-app` for this variation, then you need only perform steps 1 and 2 on page 15.

2. Edit the **srio_test_1250.pjt** file to specify that the **main_bidirectional_perf.c** file be compiled.

3. Follow the instructions in "Running the CCStudio IDE v3.3" on page 16.

4. Perform steps 1 to 11 in "Programming the TI 6488 DSP for the 1× 1.250-Gbaud Design Variation" on page 18.

To run the bidirectional performance test on the FPGA and on the TI 6488 DSP, perform the following steps:

1. Return to the nios2-terminal session you opened in step 1.

2. At the `Serial RapidIO>` prompt, type the following sequence of commands:

   ```
   stop
   r
   rate 1250
   clk 40
   pload 256
   init
   start
   stats
   ```

3. Perform step 12 in "Programming the TI 6488 DSP for the 1× 1.250-Gbaud Design Variation" on page 18. (In the CCS driver window, on the left side, click the **Run** icon to run the program).

4. To stop statistics gathering, at the nios2-terminal, type `z`.

5. To restart statistics gathering and display, type `stats`.

6. To stop the traffic initiated from the FPGA—but not the FPGA responses to requests from the TI DSP—type `stop`.

Initially, your nios2-terminal displays traffic from the FPGA to the TI 6488 DSP only. After you perform step 3, traffic flows in both directions, and the statistics display shows non-zero numbers for transactions initiated from the FPGA and from the TI 6488 DSP.

Figure 16 shows a sample bidirectional performance statistics-gathering cycle for this RapidIO variation.

Figure 16. Sample Bidirectional Performance Statistics-Gathering Cycle at 1× 1.250 Gbaud

After you type `stop`, you can try a different burst size, *<payload value>*, by typing the following command sequence:

```
pload <payload value>
start
```

The allowed values for *<payload value>* are specified in Table 4 on page 20.

Running the Bidirectional Performance Test for the 1× 2.500-Gbaud Design Variation

Before you can run the application, you must configure the Altera FPGA and load the bidirectional performance test. To configure the Altera FPGA and load the TI 6488 DSP with the bidirectional performance test for the 1× 1.250-Gbaud variation, perform the following steps:

1. If you have reprogrammed the FPGA since programming it for this design variation, follow the instructions in "Programming the Altera FPGA for the Remaining RapidIO Design Variations" on page 15.

   ☞ If you have reprogrammed the FPGA since programming it for this design variation, but have already executed `create-this-bsp` and `create-this-app` for this variation, then you need only perform steps 1 and 2 on page 15.

2. Edit the **srio_test_2500.pjt** file to specify that the **main_bidirectional_perf.c** file be compiled.

3. Follow the instructions in "Running the CCStudio IDE v3.3" on page 16.

4. Perform steps 1 to 11 in "Programming the TI 6488 DSP for the Remaining Design Variations" on page 19.

To run the bidirectional performance test on the FPGA and on the TI 6488 DSP, perform the following steps:

1. Return to the nios2-terminal session you opened in step 1.

2. At the `Serial RapidIO>` prompt, type the following sequence of commands:

```
stop
r
rate 2500
mode 1
gap 4
clk 75
pload 256
init
start
stats
```

3. Perform step 12 in "Programming the TI 6488 DSP for the 1× 1.250-Gbaud Design Variation" on page 18. (In the CCS driver window, on the left side, click the **Run** icon to run the program).

4. To stop statistics gathering, at the nios2-terminal, type z.

5. To restart statistics gathering and display, type `stats`.

6. To stop the traffic initiated from the FPGA—but not the FPGA responses to requests from the TI DSP—type `stop`.

Initially, your nios2-terminal displays traffic from the FPGA to the TI 6488 DSP only. After you perform step 3, traffic flows in both directions, and the statistics display shows non-zero numbers for transactions initiated from the FPGA and from the TI 6488 DSP.

Figure 17 shows a sample bidirectional performance statistics-gathering cycle for this RapidIO variation.

Figure 17. Sample Bidirectional Performance Statistics-Gathering Cycle at 1× 2.500 Gbaud

Running the Bidirectional Performance Test for the 1× 3.125-Gbaud Design Variation

Before you run the bidirectional performance test for the 1× 3.125-Gbaud design variation, you must program the FPGA for the correct variation and program the TI 6488 DSP for the bidirectional performance test and the correct rate setting.

Program and run the 1× 3.125-Gbaud variation according to the instructions in "Running the Bidirectional Performance Test for the 1× 2.500-Gbaud Design Variation" on page 26, with the appropriate modifications. When you run the program, specify the correct rate (3125), mode (1), and clk (84), and specify the gap. Refer to Table 1 on page 7 for the correct clk setting for this rate. The gap setting is measured in clock cycles and can remain the same for all packet-generator based variations.

Running the Bidirectional Data Integrity Test

The following sections teach you how to run the bidirectional data integrity test.

Running the Bidirectional Data Integrity Test for the 1× 1.250-Gbaud Design Variation

Before you run the application, you must configure the Altera FPGA and load the bidirectional data integrity test. To configure the Altera FPGA and load the TI 6488 DSP with the bidirectional data integrity test for the 1× 1.250-Gbaud variation, perform the following steps:

1. If you have reprogrammed the FPGA since programming it for this design variation, follow the instructions in "Programming the Altera FPGA for the 1× 1.250-Gbaud Design Variation" on page 14.

    ☞ If you have reprogrammed the FPGA since programming it for this design variation, but have already executed create-this-bsp and create-this-app for this variation, then you need only perform steps 1 and 2 on page 15.

2. Edit the **srio_test_1250.pjt** file to specify that the **main_bidirectional_dit.c** file be compiled.

3. Follow the instructions in "Running the CCStudio IDE v3.3" on page 16.

4. Perform steps 1 to 11 in "Programming the TI 6488 DSP for the 1× 1.250-Gbaud Design Variation" on page 18.

To run the bidirectional data integrity test on the FPGA and on the TI 6488 DSP, perform the following steps:

1. Return to the nios2-terminal session you opened in step 1.

2. At the Serial RapidIO> prompt, type the following sequence of commands:

```
stop
r
rate 1250
clk 40
pload 256
init
```

3. Perform step 12 in "Programming the TI 6488 DSP for the 1× 1.250-Gbaud Design Variation" on page 18. (In the CCS driver window, on the left side, click the **Run** icon to run the program).

4. At the `Serial RapidIO>` prompt, type the following command:

```
dit 10000000
```

The nios2-terminal displays a running count of packets transmitted and checked. The 1× 1.25-Gbaud RapidIO MegaCore variation does not support dynamic statistics gathering. After the 10000000 packets are transmitted and checked, the `Serial RapidIO>` prompt is available again. To view a summary of the packet activity during this test, type `dit_stats` at the `Serial RapidIO>` prompt.

You can use a smaller number of packets as the value for the `dit` command. The value 10000000 allows you to view the packet activity with the SignalTap® II Embedded Logic Analyzer. Refer to "Viewing the RapidIO-Avalon Interface with the SignalTap II Embedded Logic Analyzer" for more information.

Figure 18 shows a sample nios2-terminal window after the following steps are performed:

1. In the nios2-terminal, type the following command sequence:

```
stop
r
rate 1250
clk 40
pload 256
init
```

2. In the CCS driver window, on the left side, click the **Run** icon to run the program.

3. In the nios2-terminal, type the following command sequence:

```
dit 20000
dit_stats
```

Figure 18. Sample Bidirectional Data Integrity Test and dit_stats Display at 1× 1.250 Gbaud

In Figure 18, the bidirectional_dit test tests `0x4E20` (decimal 20000) packets for data integrity, to implement the `dit` command. Packet numbering is from `0x0` to `0x4E1F`. The `dit_stats` command outputs the statistics gathered during the run. The FPGA writes and then reads 5120000 bytes of data, the total payload in 20000 transactions that each transfer 256 bytes. To implement these data transfers, the FPGA transmits 20000 write packets and 20000 read request packets, and receives 20000 read response

packets. The TI 6488 DSP also writes and reads data as it performs data integrity checking. At the time of the statistics report, the TI DSP had transmitted 1884 write packets with a total payload of 482304 bytes, and 1884 read request packets to check the integrity of the data it wrote. In response, the FPGA transmitted 482304 bytes of data in 1884 read response packets.

☞ In contrast to the other two RapidIO MegaCore variations, the 1× 1.250 Gbaud RapidIO MegaCore variation cannot display statistics dynamically while it is running a data integrity test. To view statistics from a bidirectional data integrity test run on this RapidIO variation, you must run dit_stats after the run completes.

Running the Bidirectional Data Integrity Test for the 1× 2.500-Gbaud Design Variation

Before you run the bidirectional data integrity test for the 1× 2.500-Gbaud design variation, you must program the FPGA for the correct variation and program the TI 6488 DSP for the bidirectional data integrity test and the correct rate setting.

Program and run the 1× 2.500-Gbaud variation according to the instructions in "Running the Bidirectional Data Integrity Test for the 1× 1.250-Gbaud Design Variation" on page 28, with the following modifications when you run the program at the Serial RapidIO> prompt:

1. Add the mode 1 command following the rate command.

   The gap setting is not significant, because this test does not measure performance.

2. When you run the program, specify the correct rate (2500), mode (1), and clk (75) values.

3. Following the dit command, replace the dit_stats command with the stats command.

Figure 19 shows a sample nios2-terminal window after the FPGA and the TI 6488 DSP are programmed for the bidirectional data integrity test on the 1× 2.500-Gbaud design variation, and the following steps are performed:

1. In the nios2-terminal, at the Serial RapidIO> prompt, type the following command sequence:

```
stop
r
rate 2500
mode 1
clk 75
pload 256
init
```

2. In the CCS driver window, on the left side, click the **Run** icon to run the program.

3. At the Serial RapidIO> prompt, type the following command:

   dit 80000000 ↵

4. To view dynamic statistics gathering, at the Serial RapidIO> prompt, type the following command:

   stats ↵

5. To stop the dynamic statistics display, at the Serial RapidIO> prompt, type the following command:

   z ↵

Figure 19 shows a sample of the output from the `stats` command. It shows traffic initiated from both the FPGA and from the TI DSP.

Figure 19. Sample Bidirectional Data Integrity Test stats Display at 1× 2.500 Gbaud

Running the Bidirectional Data Integrity Test for the 1× 3.125-Gbaud Design Variation

Before you run the bidirectional data integrity test for the 1× 3.125-Gbaud design variation, you must program the FPGA for the correct variation and program the TI 6488 DSP for the bidirectional data integrity test and the correct rate setting.

Program and run the 1× 3.125-Gbaud variation according to the instructions in "Running the Bidirectional Data Integrity Test for the 1× 2.500-Gbaud Design Variation" on page 30, with the appropriate modifications. When you run the program, specify the correct `rate` (3125), `mode` (1), and `clk` (84) values. Refer to Table 1 on page 7 for the correct `clk` setting for this rate. The `gap` setting is not significant, because this test does not measure performance.

Running the Doorbell Message Tests

The Doorbell message test has the following two parts:

■ FPGA-to-DSP: The RapidIO MegaCore function on the FPGA sends Doorbell messages to the TI DSP and the test reports their Doorbell Sent status.

■ DSP-to-FPGA: The TI DSP sends Doorbell messages to the RapidIO MegaCore function on the FPGA and the test reports the Doorbell messages the RapidIO MegaCore function receives.

You must program the TI DSP differently for the two parts of the test.

Before you can run the application, you must configure the Altera FPGA and load the TI 6488 DSP with the correct program for the part of the test you want to run.

To configure the Altera FPGA, follow the instructions in "Programming the Altera FPGA for the 1× 1.250-Gbaud Design Variation" on page 14 or in "Programming the Altera FPGA for the Remaining RapidIO Design Variations" on page 15, depending on your RapidIO design variation.

☞ If you have reprogrammed the FPGA since programming it for this design variation, but have already executed `create-this-bsp` and `create-this-app` for this variation, then you need only perform steps 1 and 2 on page 15.

Loading and Running the FPGA-to-DSP Doorbell Message Test

To load the TI 6488 DSP with the FPGA-to-DSP Doorbell message test for your RapidIO design variation, perform the following steps:

1. Edit the appropriate **srio_test_<*rate*>.pjt** file to specify that the **main_unidirectional.c** file be compiled.

2. Follow the instructions in "Running the CCStudio IDE v3.3" on page 16.

3. Follow the instructions in "Programming the TI 6488 DSP for the 1× 1.250-Gbaud Design Variation" on page 18 or in "Programming the TI 6488 DSP for the Remaining Design Variations" on page 19, depending on your RapidIO design variation.

To run the FPGA-to-DSP Doorbell message test on the FPGA and on the TI 6488 DSP, perform the following steps:

1. Return to the nios2-terminal session you opened when you configured and programmed the FPGA.

2. At the `Serial RapidIO>` prompt, type the following sequence of commands:

```
link
init
tx_dbell_test
```

The `tx_dbell_test` command directs the RapidIO MegaCore function in the FPGA to send five groups of 16 Doorbell messages to the TI DSP. After the test directs the RapidIO MegaCore function to transmit a group of Doorbell messages, the test verifies the status and content of each Doorbell message in the group. Figure 20 shows the test output to the nios2-terminal for the first group of Doorbell messages sent from the RapidIO MegaCore function in the FPGA to the TI DSP.

Figure 20. Doorbell Message Test Output for Successful First Group From FPGA to TI DSP

The TI DSP is programmed to handle only 64 outstanding messages. Therefore, the fifth group of Doorbell messages times out. The Doorbell Sent Status of each Doorbell message in the group indicates the timeout. Figure 21 shows the test output with Doorbell Sent Status indicating a timeout for each message in the fifth group of Doorbell messages sent from the RapidIO MegaCore function in the FPGA.

Figure 21. Doorbell Message Test Output Showing Timeout for Fifth Group From FPGA to TI DSP

Loading and Running the DSP-to-FPGA Doorbell Message Test

To load the TI 6488 DSP with the DSP-to-FPGA Doorbell message test and run this test on the FPGA and on the TI 6488 DSP, perform the following steps:

1. Return to the nios2-terminal session you opened when you configured and programmed the FPGA.

2. At the `Serial RapidIO>` prompt, type the following command:

   `init` ↵

3. Edit the appropriate **srio_test_<*rate*>.pjt** file to specify that the **main_tx_dbell.c** file be compiled.

4. Follow the instructions in "Running the CCStudio IDE v3.3" on page 16.

5. Follow the instructions in "Programming the TI 6488 DSP for the 1× 1.250-Gbaud Design Variation" on page 18 or in "Programming the TI 6488 DSP for the Remaining Design Variations" on page 19, depending on your RapidIO design variation.

   The TI 6488 DSP generates 16 Doorbell messages and transmits them to the RapidIO MegaCore function on the FPGA. Figure 22 shows the CCS window after the test completes.

6. At the `Serial RapidIO>` prompt, type the following command:

   `get_dbells` ↵

   The nios2-terminal displays information about the Doorbell messages received by the RapidIO MegaCore function on the FPGA, as shown in Figure 23.

Figure 22. Code Composer Studio Displays Doorbell Messages Sent From the TI DSP to the FPGA
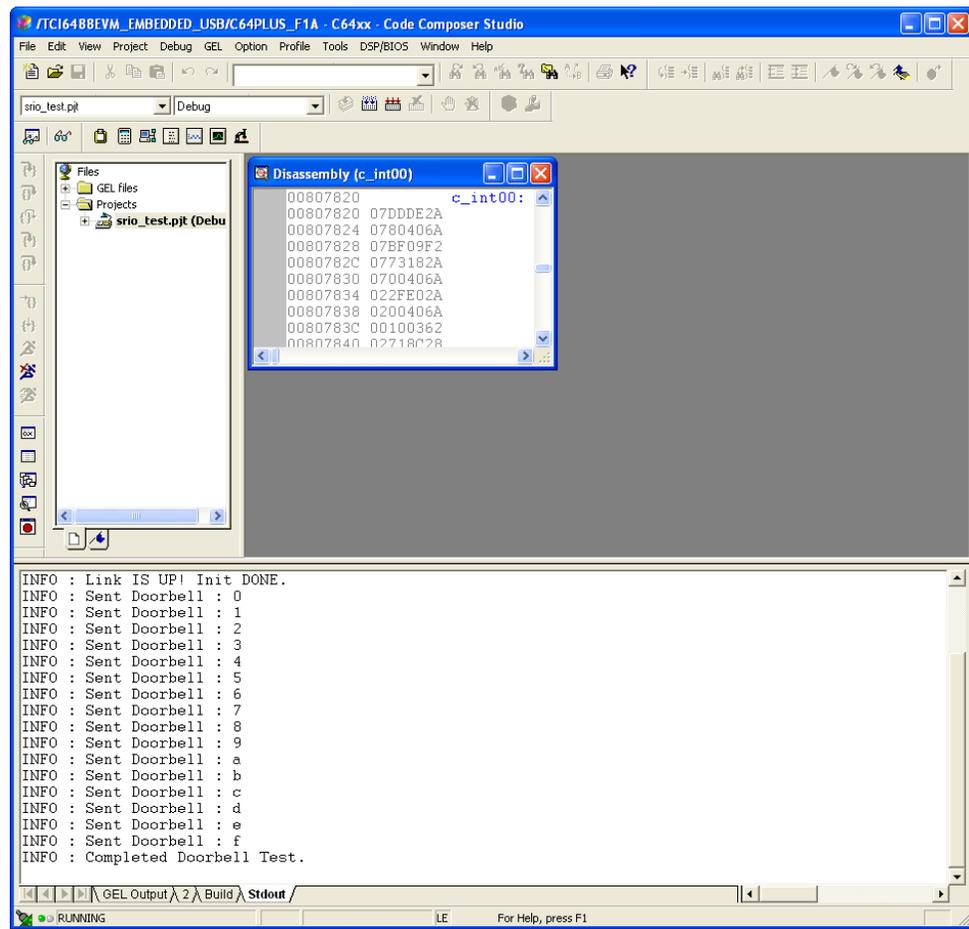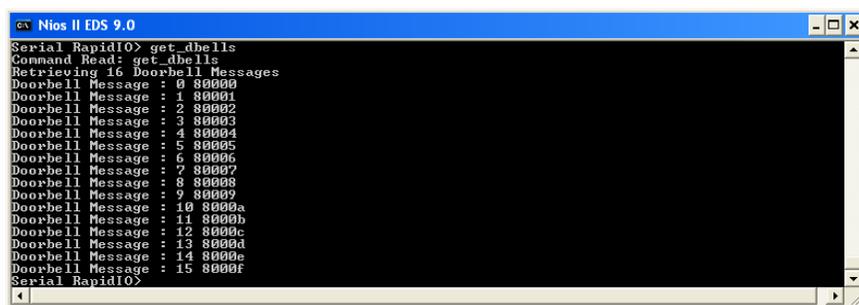

Figure 23. Doorbell Messages Received by the FPGA

Generating SWRITE or NWRITE_R Transactions

The command sequences described for the unidirectional performance, bidirectional performance, and bidirectional data integrity tests cause the RapidIO MegaCore function to generate NWRITE transactions only. To test SWRITE or NWRITE_R transactions from the FPGA, insert the following additional command in the command sequence, following the init command and preceding the start command:

■ To generate SWRITE transactions, type the following command:
   load 0x1040c 0x0008ff02 ↵

■ To generate NWRITE_R transactions, type the following command:
   load 0x1040c 0x0008ff01 ↵

For detailed information about using register 0x1040c, the Input/Output Slave Mapping Window 0 Control register, refer to the *RapidIO MegaCore Function User Guide*.

Generating a Maintenance Write Transaction

To generate a Maintenance Write transaction, at the Serial RapidIO> prompt, type the following command:

   rmw *<addr>* *<value>*

where *<addr>* is the address offset of the configuration register in the remote processing endpoint to which you are writing. For example, typing the command

   rmw 0x60 0x00AAFFFF

programs the Base Device ID of the TI 6488 DSP RapidIO peripheral to 0xAA.

Generating a Maintenance Read Transaction

To generate a Maintenance Read transaction, at the Serial RapidIO> prompt, type the following command:

   rmr *<addr>*

where *<addr>* is the address offset of the configuration register in the remote processing endpoint which you are reading. For example, typing the command

   rmr 0x60

after implementing the rmw command in the previous section should return the value 0x00AAFFFF.

Viewing the RapidIO-Avalon Interface with the SignalTap II Embedded Logic Analyzer

The reference design includes a predefined SignalTap II Embedded Logic Analyzer file, **stp1.stp**. This file defines a subset of the signals that communicate between system interconnect fabric and the RapidIO MegaCore function and specifies that the SignalTap II Embedded Logic Analyzer should capture them. You can add signals to this **.stp** file. This section teaches you how to start the SignalTap II Embedded Logic Analyzer and view the signals.

To use the SignalTap II Embedded Logic Analyzer to view the signals, perform the following steps:

1. Exit the nios2-terminal session.

2. Navigate to the main design directory for your current design variation, **altera_pcie_srio_ref_design/srio_<rate>_<mode>**.

3. Open the Quartus II project file **srio_<rate>_<mode>.qpf**.

4. On the Tools menu, click **SignalTap II Logic Analyzer**.

5. In the SignalTap II window, click **Setup**. The **Hardware Setup** dialog box appears.

6. In the **Hardware Setup** dialog box, for **Currently selected hardware**, select **USB-Blaster**.

7. Click **Close**.

8. Return to the Nios II command shell.

9. Reload the program, following the instructions in "Programming the Altera FPGA for the 1× 1.250-Gbaud Design Variation" on page 14 or in "Programming the Altera FPGA for the Remaining RapidIO Design Variations" on page 15, depending on your design variation.

10. Start a nios2-terminal session by typing the following commands:

```
nios2-download --device=1 -g srio_test.elf ↵
nios2-terminal --device=1 ↵
```

11. In the nios2-terminal, type the initial commands in the test command sequence, including the `start` command, by following the instructions in "Running the Unidirectional Performance Test for the 1× 1.250-Gbaud Design Variation" on page 20.

12. Return to the SignalTap II window and click the **Autorun Analysis** icon. Figure 24 shows the icon.

Figure 24. The SignalTap II Embedded Logic Analyzer Autorun Analysis Icon
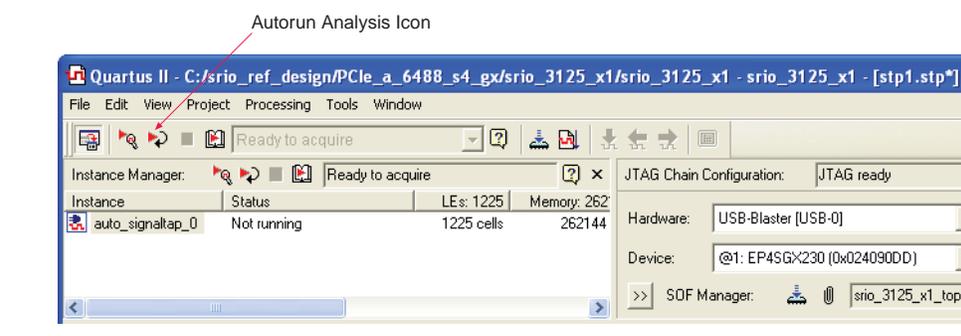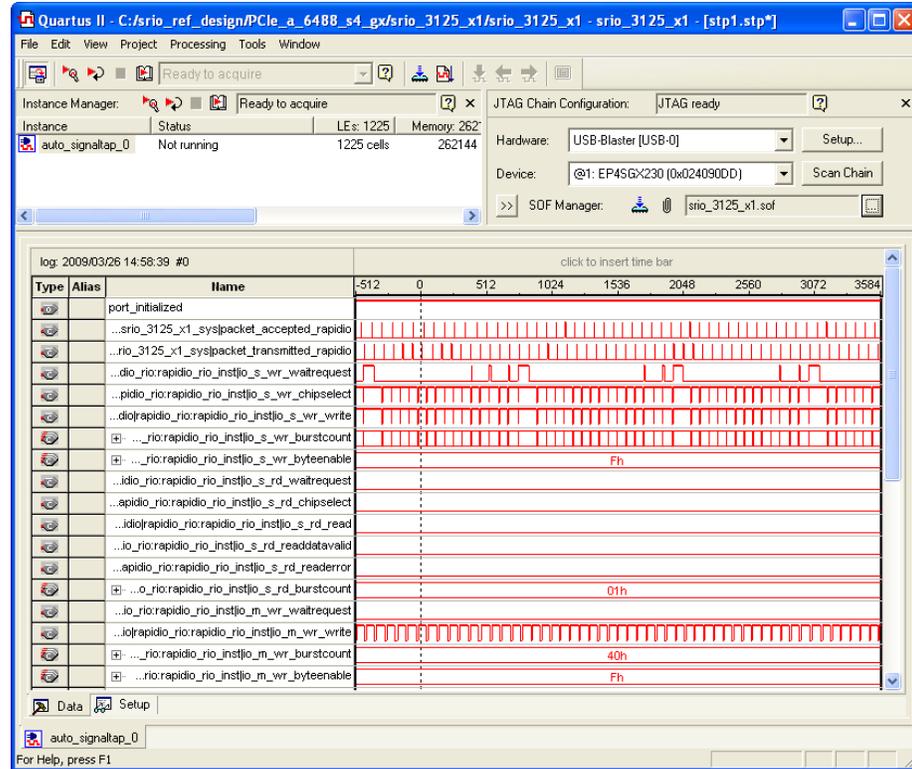
Figure 25 shows the SignalTap II Embedded Logic Analyzer. The window displays activity on the I/O slave write and I/O master write interfaces of the RapidIO MegaCore function.

Figure 25. SignalTap II Embedded Logic Analyzer Session

Performance Summary

This section contains three graphs that illustrate performance observations for each of the three design variations.

The performance results reported in this section were obtained using the Quartus II software v9.0 and the CCS IDE v3.3, and an assembly version 509310 Rev D TI DSP card and a Stratix IV GX FPGA development board Rev A.

The graphs in Figure 26 to Figure 28 show the performance numbers measured using this design. Each graph shows the performance for the three RapidIO WRITE transaction types.

Figure 26. Throughput with Increasing Payload in ×1 RapidIO MegaCore function at 1.250 Gbaud
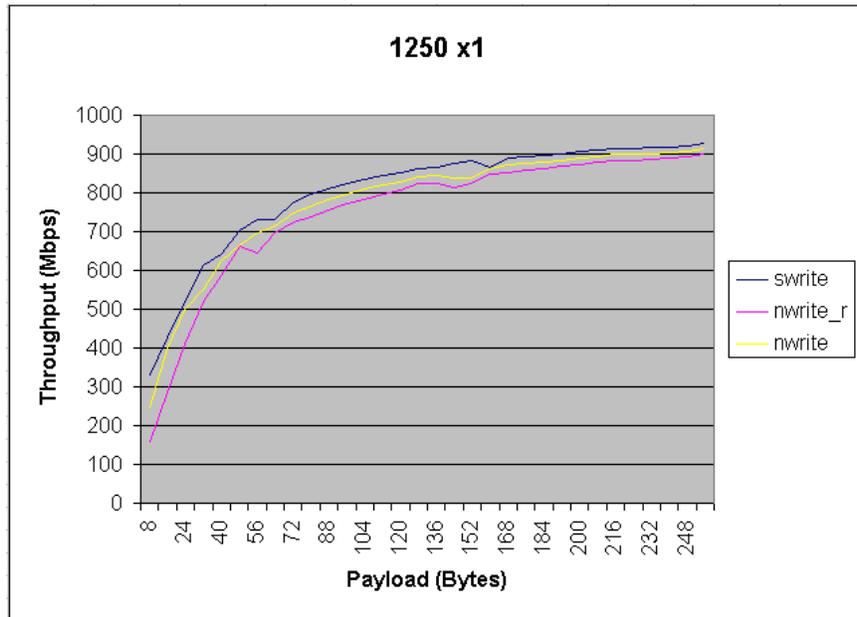
Figure 27. Throughput with Increasing Payload in ×1 RapidIO MegaCore function at 2.500 Gbaud
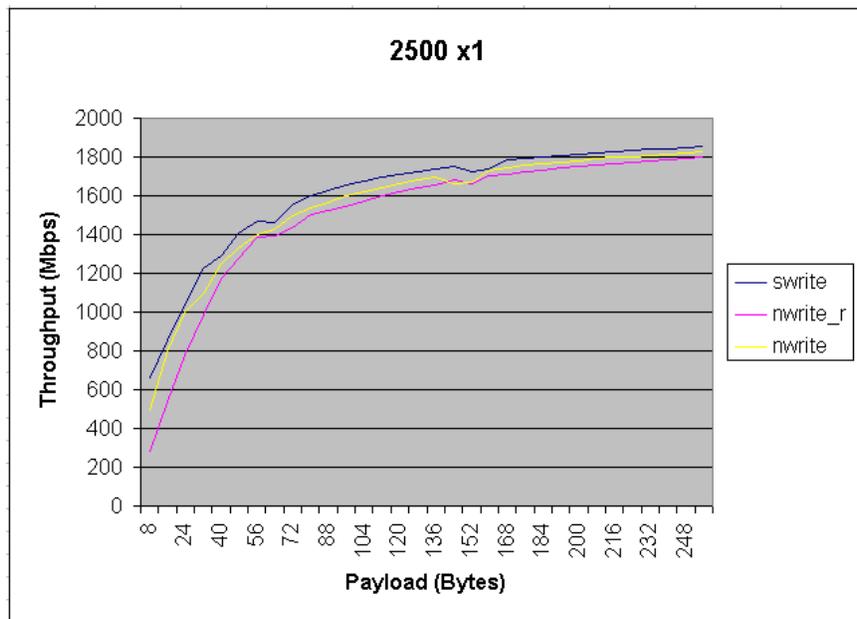
Figure 28. Throughput with Increasing Payload in ×1 RapidIO MegaCore function at 3.125 Gbaud
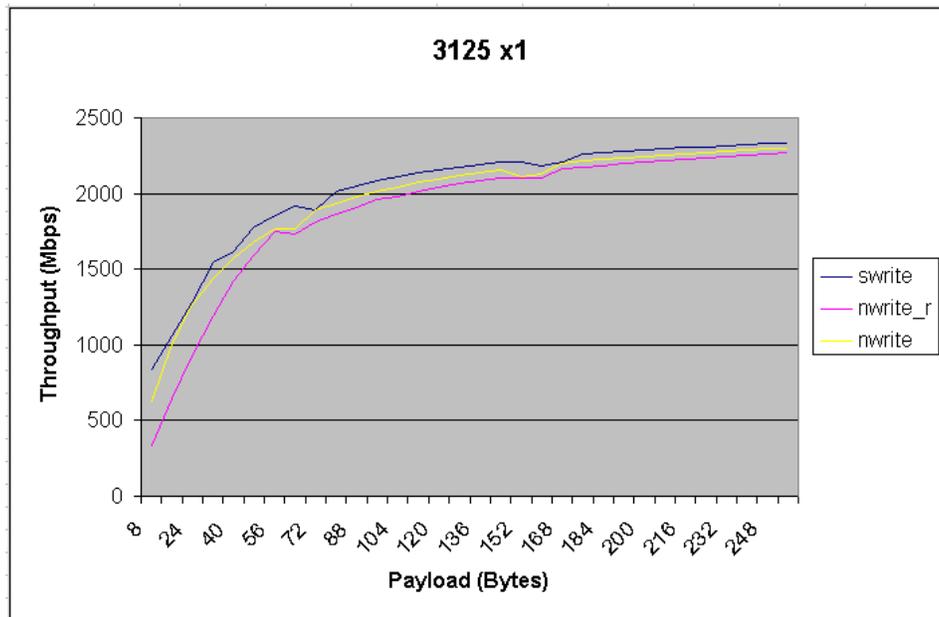
Design Limitations

This section lists the known limitations of the current version of this reference design.

Both the hardware and the RapidIO MegaCore function driver support the following transactions. However, the current version of this application note does not include results for these transactions:

■ Maintenance Port Write

■ Maintenance Port Read

The design can saturate the link with line rate traffic using the DMA approach only in the 1× 1.250 Gbaud design variation. A packet-generator approach allows the design to saturate the link for the other variations of the RapidIO protocol.

The current design does not generate a transaction mix. You must switch between transaction types manually, by setting the Input/Output Slave Mapping Window 0 Control register.

Referenced Documents

This application note references or contains information related to the following documents:

■ *RapidIO MegaCore Function User Guide*

■ *TMS320TCI6488 EVM Quick Start Installation Guide*, supplied with your EVM

Document Revision History

Table 7 shows the revision history for this application note.

Table 7. Document Revision History

Date and Document Version	Changes Made	Summary of Changes
May 2009 v1.0	Initial release.	—

.

101 Innovation Drive
San Jose, CA 95134
www.altera.com
Technical Support
www.altera.com/support

I.S. EN ISO 9001