# RapidIO Interoperability with TI 6482 DSP Reference Design

© November 2008 AN513-2.0

## Introduction

The Altera® RapidIO interoperability reference design provides a sample interface between the Altera RapidIO MegaCore® function and the Texas Instruments TMS320TCI6482 Communications Infrastructure Digital Signal Processor (TI 6482 DSP or TI 6482). Altera offers this reference design to demonstrate the installation and operation of Altera's RapidIO MegaCore function with the TI 6482. 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 in all modes, 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® II GX device to the TI 6482 DSP, given a set of operating constraints such as lane width and baud rate.

The reference design has the following features:

- Programs the RapidIO MegaCore function to initiate the following RapidIO transactions types, targeted to the TI 6482:
  - `NWRITE`
  - `NWRITE_R`
  - `SWRITE`
  - `NREAD`
  - `Maintenance Write`
  - `Maintenance Read`
- Programs the RapidIO MegaCore function to any Serial RapidIO variation. The comprehensive set of design variations includes all lane width (1× and 4×) and data rate (1.25, 2.5, and 3.125 gigabaud (GBaud)) combinations and all supported packet sizes.
- Supports testing simultaneous traffic in both directions, including transactions initiated by both the RapidIO MegaCore function and the TI 6482 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

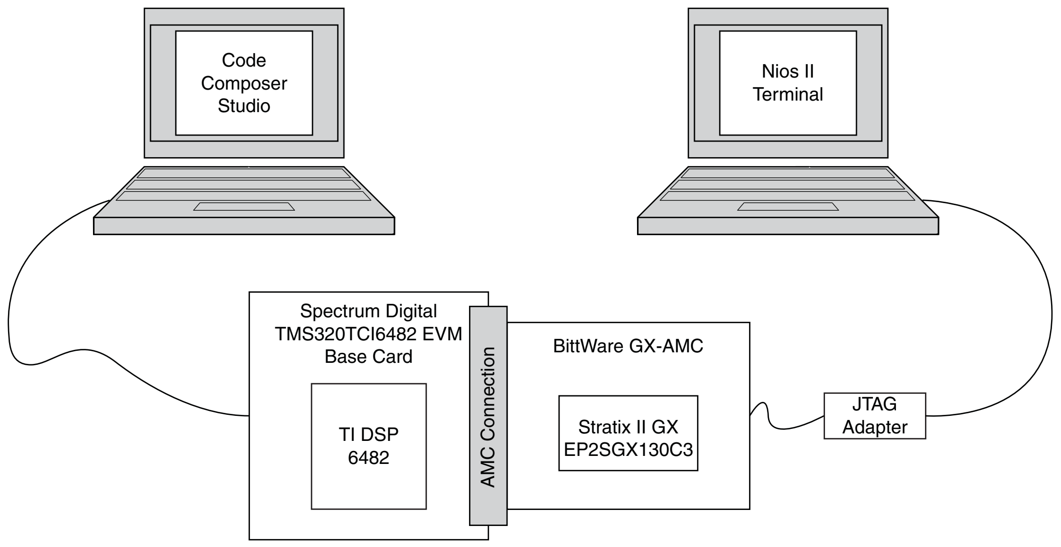
The reference design is a sample application that connects the Altera RapidIO MegaCore function to the TI 6482 DSP through the reference design interface circuitry. The design uses two third-party boards. BittWare, Inc. (www.bittware.com) provides the GX-AMC Card (GXAM), which includes an Altera Stratix II GX EP2SGX130C device, an Advanced Mezzanine Card (AMC) connector, and an optional JTAG adapter board. Spectrum Digital, Inc. (www.spectrumdigital.com) provides the TMS320TCI6482 Evaluation Module (EVM), which includes the Texas Instruments Code Composer Studio (CCS) software to program and monitor the TMS320TCI6482 EVM. Altera provides software that implements each variation of the reference design on the FPGA and programs the EP2SGX130C device on the BittWare GXAM.

The TMS320TCI6482 EVM includes a base card with a primary TI 6482 DSP (TI DSP card), a mezzanine card with a secondary TI 6482 DSP, assorted cables, the CCS software, and documentation. The TI DSP card is a Spectrum Digital board that includes the primary TI 6482 and an AMC connector.

## Reference Design Overview

Figure 1 is a diagram of the complete system.

**Figure 1.** Complete System for RapidIO Interoperability Reference Design



The CCS software running on the far left computer in Figure 1 performs the following tasks:

- Configures the TI 6482 internal PLLs

- Configures the SERDES blocks
- Programs the 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
- Displays the DDR2 memory space
- Checks the status of the RapidIO link

The **nios2-terminal** session on the far right computer in Figure 1 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 every mode and rate of the RapidIO MegaCore function. The design for the 1× 1.25 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 RapidIO MegaCore function variations—the 4× variation at 1.25 Gbaud and the 1× and 4× variations at 2.5 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.

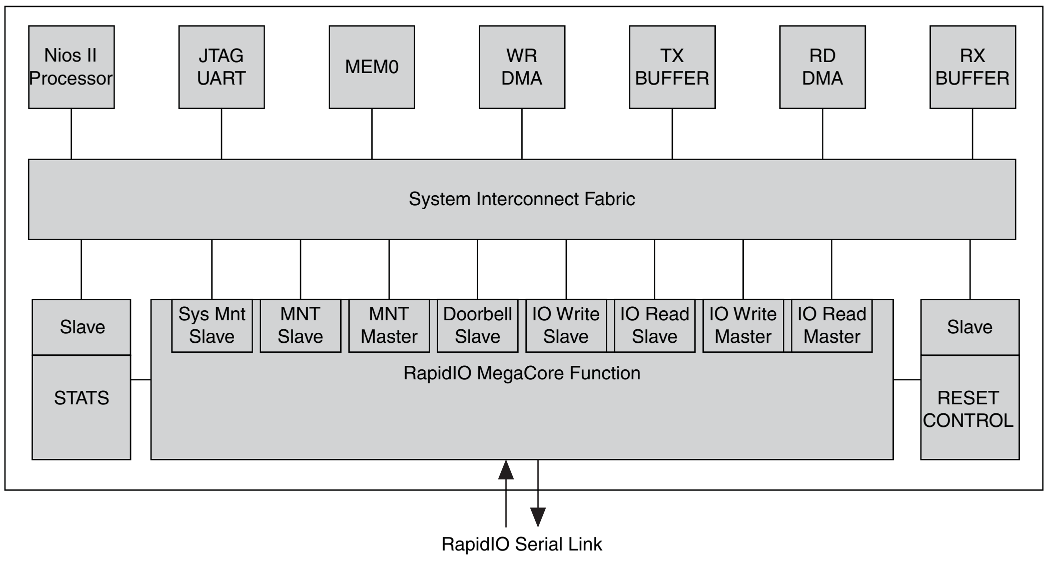
The design can saturate the link with line rate traffic using the DMA approach only in the 1× 1.25 Gbaud design variation. A packet-generator approach allows the design to saturate the link for the other variations of the RapidIO protocol.

The following sections describe the DMA-based and the packet-generator-based versions of the design, as well as the TI 6482 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 retries
- 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 data integrity test initiated by the RapidIO MegaCore function does not support dynamic statistics display. After the data integrity test completes, you can run the `dit_stats` command to report statistics for the test run.

#### RESET CONTROL

This custom SOPC Builder component allows the user to execute a soft reset to the RapidIO MegaCore function. Because the BittWare GXAM does not have a reset button, this component is very useful.

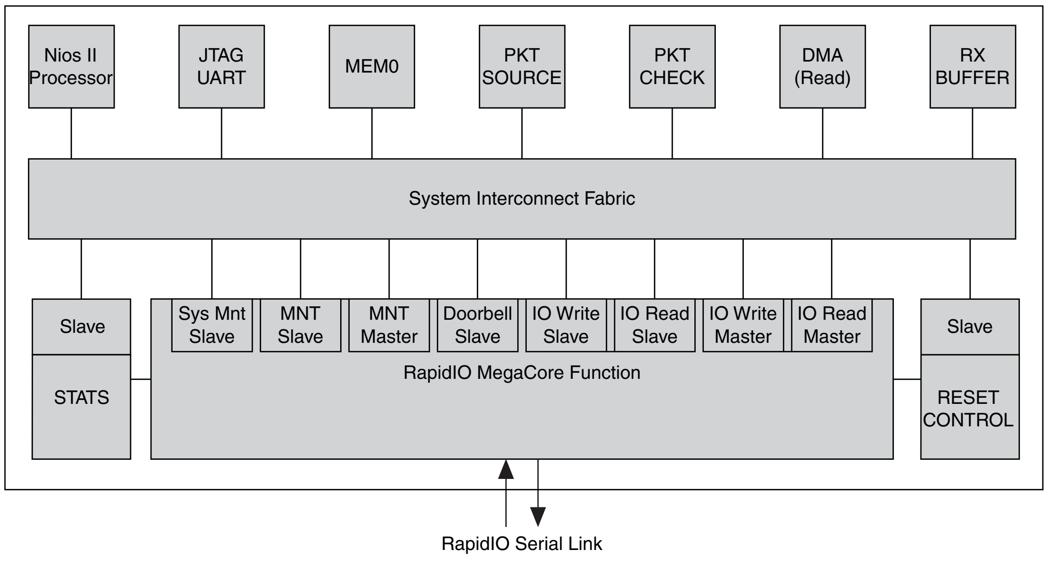
#### RapidIO MegaCore Function

The RapidIO MegaCore function is the main component in this reference design. This component is responsible for establishing 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 6482 DSP Function

`NWRITE`, `NWRITE_R`, `SWRITE`, and `NREAD` transactions initiated by the Altera RapidIO MegaCore target the DDR2 SDRAM connected to the TI 6482 DSP on the TI DSP card. The DDR2 SDRAM address space starts at address `0xE0000000`. 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 software and read the contents of address `0xE0000000` 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 6482 Interoperability Reference Design

| Data rate (Gbaud) | Mode | ALT2GXB Reference Clock Rate | Avalon System Clock |
|---|---|---|---|
| 1.25 | 1x | 125 MHz | 41.5 MHz |
| | 4x | 125 MHz | 83 MHz |
| 2.50 | 1x | 125 MHz | 83 MHz |
| | 4x | 125 MHz | 125 MHz |
| 3.125 | 1x | 125 MHz | 83 MHz |
| | 4x | 125 MHz | 156 MHz |

The reference clock is dictated by the clocks available on the BittWare GXAM. 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 9
- "Reference Design Installation" on page 9
- "Preparing to Run the Applications" on page 13

### 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 v3.2 software
- One computer running the WIndows XP operation system, capable of running a **nios2-terminal** session
- BittWare GXAM (contains an Altera EP2SGX130C device)
- Spectrum Digital TMS320TCI6482 EVM
- Altera USB-Blaster™ download cable

The performance results reported in "Performance Summary" on page 34 were obtained using an assembly version 508550-2002 Rev C TI DSP card and a GXAM Rev 0.

Figure 4 shows the GXAM connected to the TI DSP card. The BittWare card has a male AMC connector that plugs into the female AMC connector on the TI DSP card.

**Figure 4.** TMS320TCI6482 EVM Base Card and BittWare GX-AMC Card

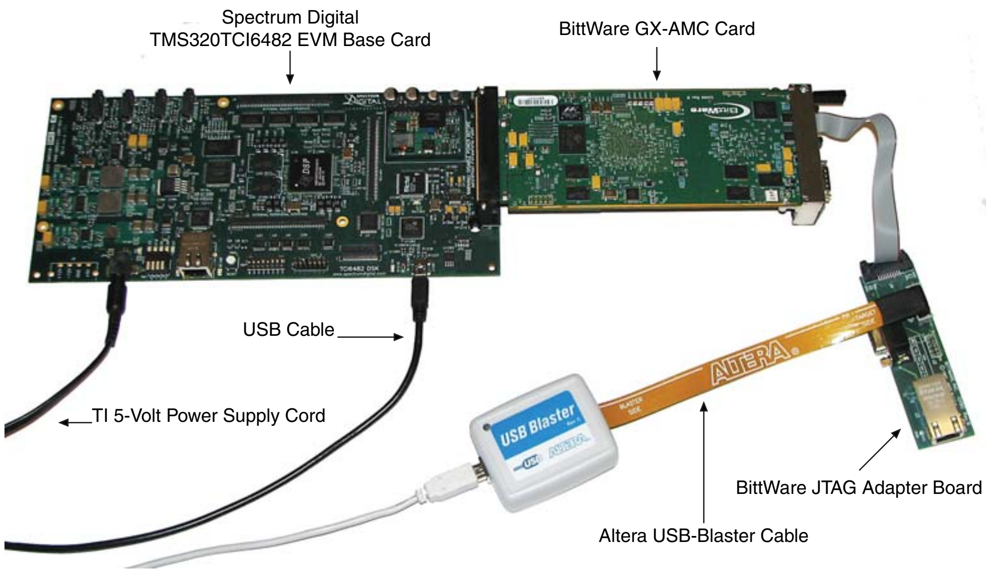

Spectrum Digital
TMS320TCI6482 EVM Base Card
BittWare GX-AMC Card
USB Cable
TI 5-Volt Power Supply Cord
BittWare JTAG Adapter Board
Altera USB-Blaster Cable

Figure 4 also shows the following connectors:

- BittWare JTAG adapter board

- Altera USB-Blaster download cable
- USB Cable
- TI 5-volt power supply cord

The BittWare JTAG adapter board connects the Altera JTAG USB-Blaster download cable to the BittWare card's front panel `Utility` port.

The Altera USB-Blaster download cable is used to configure the FPGA on the BittWare card and also to communicate between a computer running 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 computer running the Code Composer Studio software communicates with the TI DSP card through the USB cable.

The TI 5-volt power supply provides power to the TI DSP card. The TI DSP card provides power to the BittWare GXAM across the AMC connector.

## Software Requirements

The reference design application requires the following software:

- Altera Quartus® II 8.0 software, including the USB-Blaster driver, SOPC Builder, the Nios II Embedded Design Suite (EDS), and the RapidIO MegaCore function
- TI Code Composer Studio IDE v3.2 (CCS)

## Reference Design Installation

This section describes how to install the reference design.

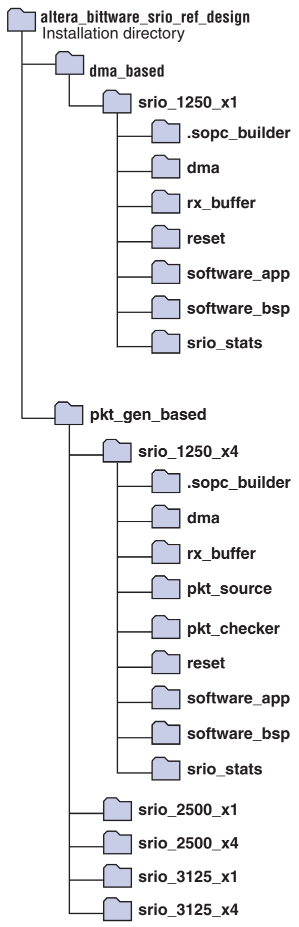
### Downloading the Package

All of the files necessary for this reference design are included in the **altera_6482_srio_ref_design.zip** file. This file is available for you to download from the Serial RapidIO To TI 6482 DSP Reference Design web page.

### Extracting the Altera FPGA Package Files

Unzip the **altera_6482_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_bittware_srio_ref_design** and **altera_ti_srio_ref_design**. The **altera_bittware_srio_ref_design** directory contains the files for programming the Altera FPGA. Figure 5 shows the **altera_bittware_srio_ref_design** directory structure.

**Figure 5.** altera_bittware_srio_ref_design Directory Structure



## Package Content Description

The **altera_bittware_srio_ref_design** directory has two subdirectories, the **dma_based** directory and the **pkt_gen_based** directory.

The **dma_based** directory contains the FPGA design for the 1.25 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 **pkt_gen_based** directory contains the FPGA designs for the remaining RapidIO MegaCore function variations—the 4× variation at 1.25 Gbaud and the 1× and 4× variations at 2.5 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 these modes and 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 "Clocks" on page 7.

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

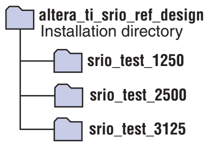
**Table 2.** Subdirectories Contained in srio_<*rate*>_<*mode*> Directory

| 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. |
| **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.<br>■ **create-this-app** – script used to compile the driver<br>■ **srio_main_full.c** – C code implementing the RapidIO driver<br>■ **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. |
| **srio_<*rate*>_<*mode*>** | Packet-based | Contains all of the files that are associated with the RapidIO MegaCore function supporting a <*mode*> <*rate*> serial link. |

## Extracting the TI 6482 DSP Package Files

After you download the reference design zip file, as described in "Downloading the Package" on page 9, and unzip the file, your working directory contains an **altera_ti_srio_ref_design** subdirectory. This directory contains the files for programming the TI 6482 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**
- **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 6482 DSP |
| **main_unidirectional.c**<br>**main_bidirectional_perf.c**<br>**main_bidirectional_dit.c** | The code to program the TI 6482 DSP. Each program supports both 1× and 4× modes. |
| **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\srio_test_<rate>\"
Source="..\..\..\TI\CCStudio_v3.2\boards\dsktci6482_v2\csl_c6482\ lib\csl_c6482.lib"
Source="..\..\..\TI\CCStudio_v3.2\C6000\cgtools\lib\rts64plus.lib"
Options=-g -pdv -fr"$(Proj_dir)\Debug" \
                  -i"c:\ti\ccstudio_v3.2\boards\dsktci6482_v2\csl_c6482\inc" \
                  -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 6482 DSP, and run the selected performance and data integrity tests. The following sections teach you how to connect the hardware, program the Altera FPGA, select your application, and program the TI 6482 DSP.

### Connecting the Hardware

Before connecting the hardware, you must install the required software listed in "Software Requirements" on page 9.

☞ Install CCS on the machine to which you intend to connect the TI DSP card, and install the Quartus II software on the other machine.

To connect the hardware, perform the following steps:

1. Install CCS v3.2 on one of the computers.
2. Power up and initialize the TI  DSP card according to the instructions in the *TMS320TCI6482 EVM Quick Start Installation Guide*, supplied with your EVM.
3. Run the diagnostic tests to verify the basic functionality of the TI DSP card.

   If you cannot run the diagnostics, the reference design tests will not complete.
4. After completing the TI 6482 diagnostic tests successfully, disconnect the power to the TI DSP card.
5. Remove the TI USB cable from the TI DSP card.
6. Disconnect the mezzanine card from the TI DSP card.
7. Remove diode D6, labeled Presence Detect, from the BittWare GXAM, to prevent CCS from searching for the secondary TI 6482 on the mezzanine card.
8. Connect the GXAM to the TI DSP card using the AMC connector.
9. Provide power to the TI DSP card.
10. Connect the USB cable from the computer with CCS to the TI DSP card.
11. Connect the USB-Blaster cable to the computer and to the BittWare JTAG adapter card.

    ☞ The JTAG adapter card must connect to the BittWare card at the `Utility` port on the BittWare card front panel. Refer to Figure 4 on page 8 for an illustration of the required configuration.

### Programming the Altera FPGA for the 1× 1.25-Gbaud Design Variation

To program the Altera FPGA for the 1× 1.25-Gbaud variation of the RapidIO interoperability reference design, perform the following steps:

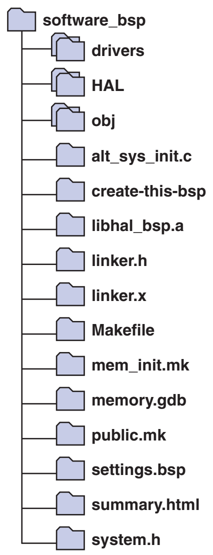
1. On the program menu, point to **Altera**, then **Nios II EDS** <*version_number*>, and click **Nios II** <*version_number*> **Command Shell**, to start a Nios II command shell.
2. Navigate to the directory **altera_bittware_srio_ref_design\dma_based\srio_1250_x1**.

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



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

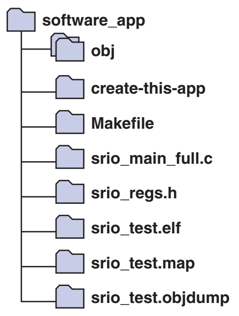
   The directory contains the following three files:
   - **srio_regs.h**
   - **srio_main_full.c**
   - **create-this-app**
2. To execute the script that compiles the driver software in **srio_main_full.c**, type the following command:

```
./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



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 ELF file, 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 ELF file and start a **nios2-terminal** session, type the following command:
   ```
   nios2-download --device=1 -g srio_test.elf && \
   nios2-terminal --device=1 ↵
   ```
   The Altera RapidIO MegaCore function is now downloaded to the FPGA.

Before you execute any of the RapidIO interoperability reference design tests, you must bring up and program the TI 6482 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× 1.25-Gbaud Design Variation" on page 13, with the following exceptions:

- Replace the path **altera_bittware_srio_ref_design\dma_based\srio_1250_x1** with the path **altera_bittware_srio_ref_design\pkt_gen_based\srio_<*rate*>_<*mode*>**.
- Replace the **.sof** file name **srio_1250_x1.sof** with **srio_<*rate*>_<*mode*>.sof**.

## Selecting the Program to Run on the TI 6482 DSP

This reference design provides three programs that you can load on the TI 6482 DSP. They are found in the following three files:

- **main_unidirectional.c**
- **main_bidirectional_perf.c**
- **main_bidirectional_dit.c**

The program in **main_unidirectional.c** configures the TI 6482 DSP as a slave only. This program is used to measure performance for data flowing from the FPGA to the TI 6482 DSP.

The program in **main_bidirectional_perf.c** configures the TI 6482 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 6482 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.

Before you can load and run the TI 6482 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 three 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_bidirectional_dit.c"
#Source="main_bidirectional_perf.c"
Source="main_unidirectional.c"
```

### Programming the TI 6482 DSP for the 1× 1.25-Gbaud Design Variation

After you edit the **.pjt** file to specify the program you want, you can load and run the TI 6482 DSP driver. This section describes how to load and run the TI 6482 DSP driver for the 1× 1.25-Gbaud design variation.

On the computer on which you installed the Code Composer Studio v3.2 software, perform the following steps:

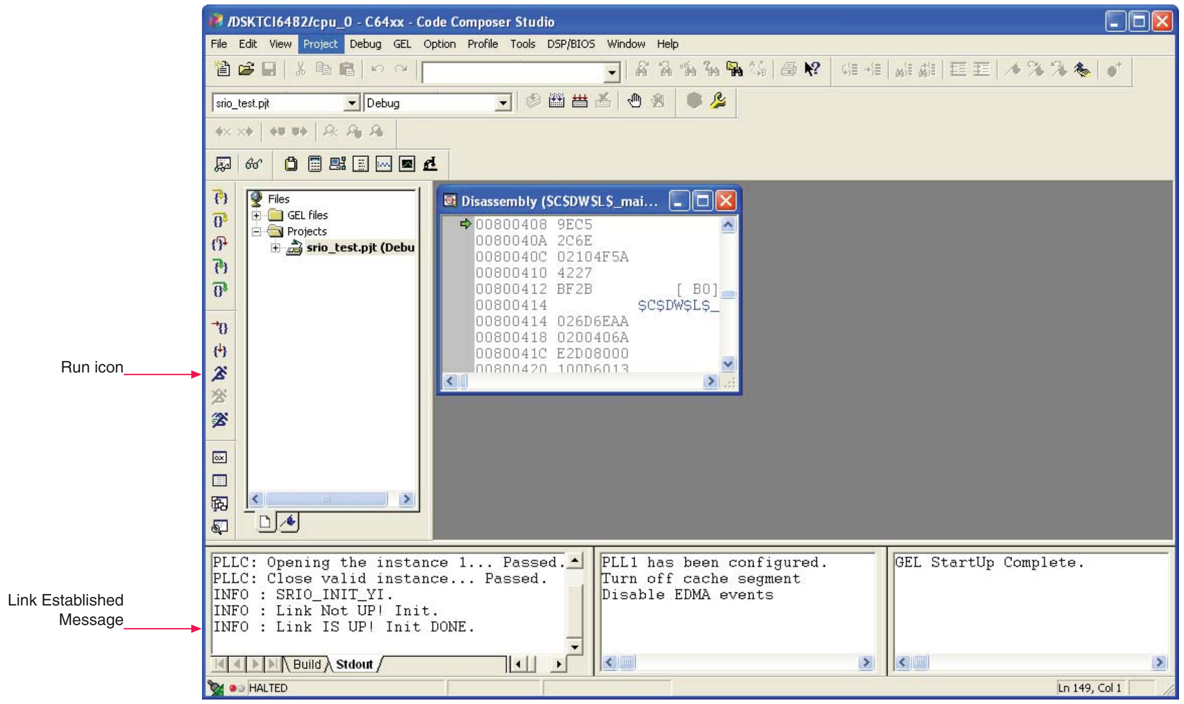
1. Double-click the **6482 DSK CCStudio** icon to start a CCS session. The driver window for the TMS320TCI6482 EVM displays.
2. In this window, on the Debug menu, click **Connect**. A connection is established between the driver and the TI 6482 DSP.
3. On the Project menu, click **Open**.
4. In the **Project Open** dialog box, navigate to the **srio_test_1250** directory.
5. Select the file **srio_test_1250.pjt**.
6. Click **Open**.
7. In the File View window, in the **Projects** directory, select the **srio_test_1250.pjt** file.
8. Right-click the file name and click **Build**. The code to configure the TI 6482 DSP compiles.
9. On the Edit menu, click **Load Program**.
10. In the Load Program window, navigate to the **Debug** directory.
11. Select **srio_1250_test.out**.

12. Click **Open**.
13. In the CCS driver window for the TMS320TCI6482 EVM, on the left side, click the **Run** icon to run the program. Figure 9 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 9 shows the **Run** icon and the final message that states the link is established.

**Figure 9.** Code Composer Studio Window


/DSKTCI6482/cpu_0 - C64xx - Code Composer Studio
File Edit View Project Debug GEL Option Profile Tools DSP/BIOS Window Help
srio_test.pjt
Debug
Files
GEL files
Projects
srio_test.pjt (Debu
Disassembly ($C$DW$L$_mai...
Build
Stdout
HALTED
Ln 149, Col 1
Run icon
Link Established Message

```
00800408 9EC5
0080040A 2C6E
0080040C 02104F5A
00800410 4227
00800412 BF2B                [ B0]
00800414                 $C$DW$L$_
00800414 026D6EAA
00800418 0200406A
0080041C E2D08000
00800420 100D6013
```

```
PLLC: Opening the instance 1... Passed.
PLLC: Close valid instance... Passed.
INFO : SRIO_INIT_YI.
INFO : Link Not UP! Init.
INFO : Link IS UP! Init DONE.
```

```
PLL1 has been configured.
Turn off cache segment
Disable EDMA events
```

```
GEL StartUp Complete.
```

## Programming the TI 6482 DSP for the Remaining Design Variations

To program the TI 6482 DSP for any other variation of the RapidIO interoperability reference design, follow the procedure in "Programming the TI 6482 DSP for the 1× 1.25-Gbaud Design Variation" on page 16, 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**.

☞ The procedure is identical for the two RapidIO modes at the same rate.

## Running the Applications

After you configure the Altera FPGA and load your selected application in the TI 6482 DSP, you can run the application, which is one of three kinds of performance tests.

The steps to run the performance tests are different for the RapidIO variation implemented in a DMA-based system—the 1× 1.25-Gbaud variation—and the remaining variations, which are implemented in a packet-generator based system.

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

### Running the Unidirectional Performance Test for the 1× 1.25-Gbaud Design Variation

Before you can run the unidirectional performance test for the 1× 1.25-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 6482 DSP with the unidirectional performance test for the 1× 1.25-Gbaud variation, perform the following steps:

1. Follow the instructions in "Programming the Altera FPGA for the 1× 1.25-Gbaud Design Variation" on page 13.

   If you have not reprogrammed the FPGA since programming it for this design variation, you can skip this step.

   If you have reprogrammed the FPGA 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 "Programming the TI 6482 DSP for the 1× 1.25-Gbaud Design Variation" on page 16.

To run the unidirectional performance test, return to the **nios2-terminal** session. 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 6482 DSP, and monitors performance. The default `init` setting generates `NWRITE` transactions.

```
stop
r
rate 1250
clk 42
pload 256
init
start
stats
z
stop
```

Type carefully, because the **nios2-terminal** does not echo the characters. After you press Enter, the reference design program displays the command that you typed.

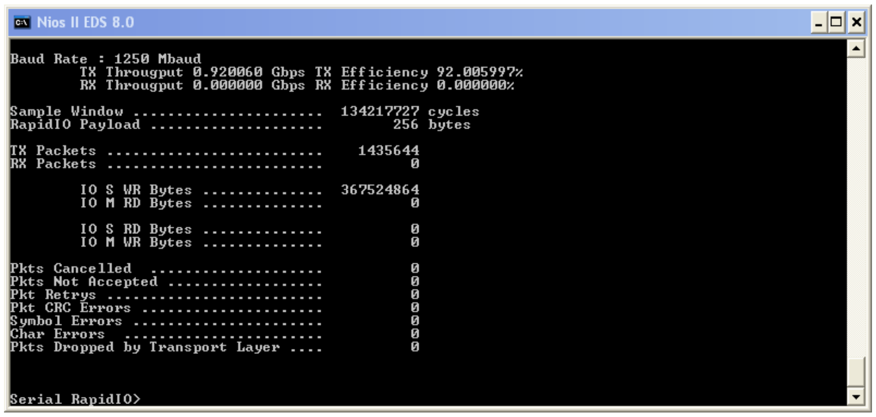
Table 4 explains the individual commands in the sequence.

**Table 4.** nios2-terminal Commands for NWRITE Performance Test on 1× 1.25-Gbaud RapidIO Variation

| 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 1.25 Gbaud. |
| `clk 42` | Tells the driver that the Avalon-MM system clock runs at 42 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. |
| `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 10 shows a sample unidirectional statistics-gathering cycle for this RapidIO variation.

**Figure 10.** Sample Unidirectional Statistics-Gathering Cycle at 1× 1.25 Gbaud



```
Baud Rate : 1250 Mbaud
        TX Througput 0.920060 Gbps TX Efficiency 92.005997%
        RX Througput 0.000000 Gbps RX Efficiency 0.000000%

Sample Window .......................  134217727 cycles
RapidIO Payload .....................        256 bytes

TX Packets ..........................    1435644
RX Packets ..........................          0

        IO S WR Bytes ...............  367524864
        IO M RD Bytes ...............          0

        IO S RD Bytes ...............          0
        IO M WR Bytes ...............          0

Pkts Cancelled  .....................          0
Pkts Not Accepted ...................          0
Pkt Retrys ..........................          0
Pkt CRC Errors ......................          0
Symbol Errors .......................          0
Char Errors  ........................          0
Pkts Dropped by Transport Layer ....           0


Serial RapidIO>
```

The previous command sequence generates `NWRITE` transactions only. To test `SWRITE` or `NWRITE_R` transactions in the reference design, 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*.

### Running the Unidirectional Performance Tests for the 4× 1.25-Gbaud Design Variation

Before you can run the unidirectional performance test for the 4× 1.25-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 6482 DSP with the unidirectional performance test for the 4× 1.25-Gbaud variation, perform the following steps:

1. Follow the instructions in "Programming the Altera FPGA for the Remaining RapidIO Design Variations" on page 15.

   If you have not reprogrammed the FPGA since programming it for this design variation, you can skip this step.

   If you have reprogrammed the FPGA 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_1250.pjt** file to specify that the **main_unidirectional.c** file be compiled.
3. Follow the instructions in "Programming the TI 6482 DSP for the 1× 1.25-Gbaud Design Variation" on page 16.

To run the unidirectional performance test, return to the **nios2-terminal** session. For a list of supported RapidIO driver commands, type `h`.

The following sequence of commands, explained in Table 5, programs the RapidIO MegaCore function, generates traffic to the TI 6482 DSP, and monitors performance. The default `init` setting generates `NWRITE` transactions.

```
stop
r
rate 1250
mode 4
gap 24
clk 83
pload 256
init
start
stats
z
stop
```

Type carefully, because the **nios2-terminal** does not echo the characters. After you press Enter, the reference design program displays the command that you typed.

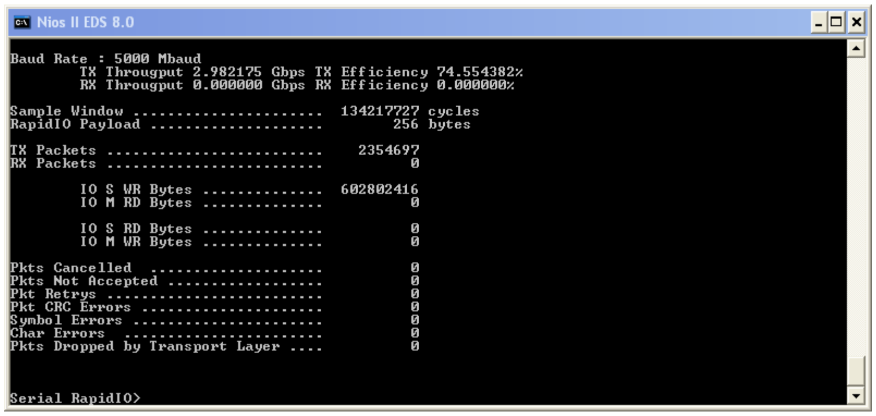
Table 5 explains the individual commands in the sequence:

**Table 5.** nios2-terminal Commands for NWRITE Performance Test on 4× 1.25-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 1.25 Gbaud. |
| `mode 4` | Sets the mode. The allowed values are `1` and `4`. |
| `gap 24` | 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 83` | Tells the driver that the Avalon-MM system clock runs at 83 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 11 shows a sample unidirectional statistics-gathering cycle for this RapidIO variation.

**Figure 11.** Sample Unidirectional Statistics-Gathering Cycle at 4× 1.25 Gbaud



```
Baud Rate : 5000 Mbaud
        TX Througput 2.982175 Gbps TX Efficiency 74.554382%
        RX Througput 0.000000 Gbps RX Efficiency 0.000000%

Sample Window ......................  134217727 cycles
RapidIO Payload ....................        256 bytes

TX Packets .........................    2354697
RX Packets .........................          0

        IO S WR Bytes ..............  602802416
        IO M RD Bytes ..............          0

        IO S RD Bytes ..............          0
        IO M WR Bytes ..............          0

Pkts Cancelled  ....................          0
Pkts Not Accepted ..................          0
Pkt Retrys .........................          0
Pkt CRC Errors .....................          0
Symbol Errors ......................          0
Char Errors  .......................          0
Pkts Dropped by Transport Layer ....          0


Serial RapidIO>
```

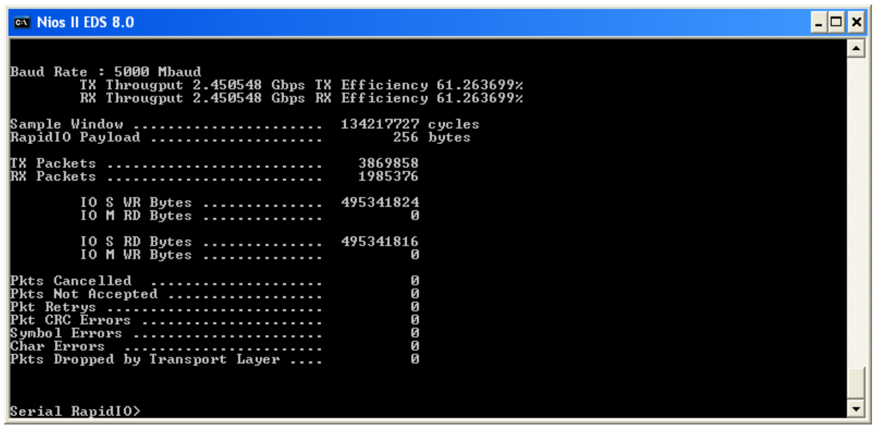
If you type the command `dit` in the command sequence instead of the command `start`, you enable data integrity checking by the pkt_check component. In this test, the FPGA initiates write commands, as in the regular unidirectional performance test. However, in the dit test, the FPGA also initiates read commands to test the success of the write operations. Table 6 provides information about this data-integrity checking command.

**Table 6.** nios2-terminal Commands for Data Integrity Testing in Packet-Generator Based Designs

| 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. |

Figure 12 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 12.** Sample Unidirectional Statistics With Data Integrity Testing at 4× 1.25 Gbaud



```
Baud Rate : 5000 Mbaud
        TX Througput 2.450548 Gbps TX Efficiency 61.263699%
        RX Througput 2.450548 Gbps RX Efficiency 61.263699%

Sample Window ........................  134217727 cycles
RapidIO Payload ......................        256 bytes

TX Packets ...........................    3869858
RX Packets ...........................    1985376

        IO S WR Bytes ..............  495341824
        IO M RD Bytes ..............          0

        IO S RD Bytes ..............  495341816
        IO M WR Bytes ..............          0

Pkts Cancelled  ......................          0
Pkts Not Accepted ....................          0
Pkt Retrys ...........................          0
Pkt CRC Errors .......................          0
Symbol Errors ........................          0
Char Errors  .........................          0
Pkts Dropped by Transport Layer ....            0


Serial RapidIO>
```

The previous command sequence generates `NWRITE` transactions only. To test `SWRITE` or `NWRITE_R` transactions in the reference design, 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*.

## Running the Unidirectional Performance Tests for the Remaining Design Variations

Program and run the remaining design variations—the 1× and 4× variations at 2.5 and at 3.125 Gbaud—according to the instructions in "Running the Unidirectional Performance Tests for the 4× 1.25-Gbaud Design Variation", using the appropriate rate and mode settings. You must program the FPGA for the correct variation and program the TI 6482 DSP for the unidirectional performance test and the correct rate setting.

When you run the program, specify the correct `rate`, `mode`, and `clk`, and specify the `gap`. Refer to Table 1 on page 7 for the correct `clk` setting for your rate and mode. The `gap` setting is measured in clock cycles and can remain the same for all variations.

## Running the Bidirectional Performance Tests for the 1× 1.25-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 6482 DSP with the bidirectional performance test for the 1× 1.25-Gbaud variation, perform the following steps:

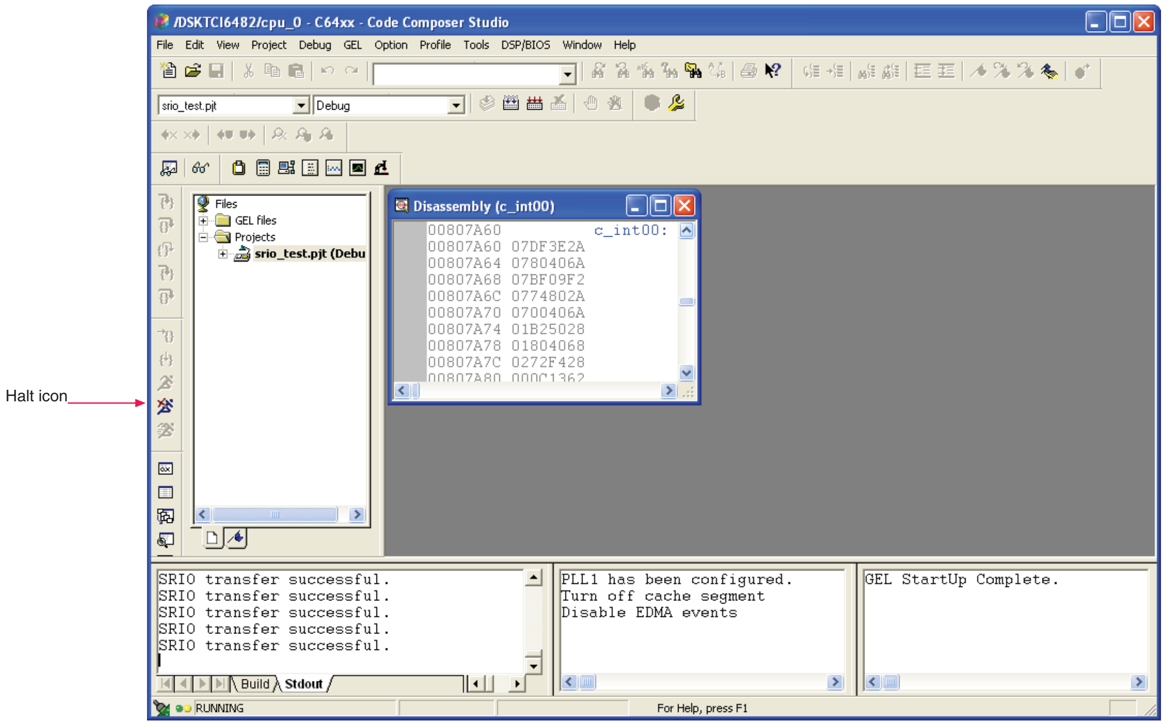
1. Follow the instructions in "Programming the Altera FPGA for the 1× 1.25-Gbaud Design Variation" on page 13.

   ☞ If you have not reprogrammed the FPGA since programming it for this design variation, you can skip this step.

   ☞ If you have reprogrammed the FPGA 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. To stop the TI 6482 DSP program that was running previously, if you have not already done so, perform the following steps:
   a. In the CCS driver window for the TMS320TCI6482 EVM, on the left side, click the **Halt** icon to stop the program. Figure 13 shows the icon.
   b. On the Project menu, click **Close**.

**Figure 13.** Code Composer Studio Window With Halt Icon Active



```
00807A60          c_int00:
00807A60 07DF3E2A
00807A64 0780406A
00807A68 07BF09F2
00807A6C 0774802A
00807A70 0700406A
00807A74 01B25028
00807A78 01804068
00807A7C 0272F428
00807A80 000C1362
```

```
SRIO transfer successful.
SRIO transfer successful.
SRIO transfer successful.
SRIO transfer successful.
SRIO transfer successful.
```

```
PLL1 has been configured.
Turn off cache segment
Disable EDMA events
```

```
GEL StartUp Complete.
```

4. Perform steps 3 to 12 in "Programming the TI 6482 DSP for the 1× 1.25-Gbaud Design Variation" on page 16.

To run the bidirectional performance test on the FPGA and on the TI 6482 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:

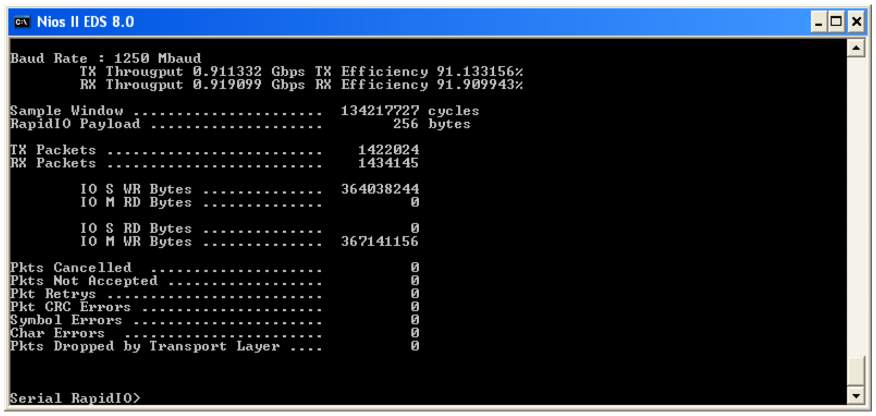
```
stop
r
rate 1250
clk 42
pload 256
init
start
stats
```

3. Perform step 13 in "Programming the TI 6482 DSP for the 1× 1.25-Gbaud Design Variation" on page 16.
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 6482 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 6482 DSP.

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

**Figure 14.** Sample Bidirectional Performance Statistics-Gathering Cycle at 1× 1.25 Gbaud



```
Baud Rate : 1250 Mbaud
        TX Througput 0.911332 Gbps TX Efficiency 91.133156%
        RX Througput 0.919099 Gbps RX Efficiency 91.909943%

Sample Window .........................  134217727 cycles
RapidIO Payload .......................        256 bytes

TX Packets ............................    1422024
RX Packets ............................    1434145

        IO S WR Bytes ...............  364038244
        IO M RD Bytes ...............          0

        IO S RD Bytes ...............          0
        IO M WR Bytes ...............  367141156

Pkts Cancelled  .......................          0
Pkts Not Accepted .....................          0
Pkt Retrys ............................          0
Pkt CRC Errors ........................          0
Symbol Errors .........................          0
Char Errors  ..........................          0
Pkts Dropped by Transport Layer ....           0


Serial RapidIO>
```

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 19.

The previous command sequence causes 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*.

### Running the Bidirectional Performance Tests for the 4× 1.25-Gbaud Design Variation

Before you run the application, you must configure the Altera FPGA and load the bidirectional performance test. To configure the Altera FPGA and load the TI 6482 DSP with the bidirectional performance test for the 4× 1.25-Gbaud variation, perform the following steps:

1. Follow the instructions in "Programming the Altera FPGA for the Remaining RapidIO Design Variations" on page 15.

   If you have not reprogrammed the FPGA since programming it for this design variation, you can skip this step.

If you have reprogrammed the FPGA 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_1250.pjt** file to specify that the **main_bidirectional_perf.c** file be compiled.
3. To stop the TI 6482 DSP program that was running previously, if you have not already done so, perform the following steps:
   a. In the CCS driver window for the TMS320TCI6482 EVM, on the left side, click the **Halt** icon to stop the program. Figure 13 on page 24 shows the icon.
   b. On the Project menu, click **Close**.
4. Perform steps 3 to 12 in "Programming the TI 6482 DSP for the 1× 1.25-Gbaud Design Variation" on page 16.

To run the bidirectional performance test on the FPGA and on the TI 6482 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
mode 4
gap 24
clk 83
pload 256
init
start
stats
```

3. Perform step 13 in "Programming the TI 6482 DSP for the 1× 1.25-Gbaud Design Variation" on page 16.
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, type `stop`.

Initially, your **nios2-terminal** displays traffic from the FPGA to the TI 6482 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 6482 DSP.

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 19.

The previous command sequence causes 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*.

### Running the Bidirectional Performance Tests for the Remaining Design Variations

Program and run the remaining design variations—the 1× and 4× variations at 2.5 and at 3.125 Gbaud—according to the instructions in "Running the Bidirectional Performance Tests for the 4× 1.25-Gbaud Design Variation" on page 25, using the appropriate rate and mode settings. You must program the FPGA for the correct variation and program the TI 6482 DSP for the bidirectional performance test and the correct rate setting.

When you run the program, specify the correct `rate`, `mode`, and `clk`, and specify the `gap`. Refer to Table 1 on page 7 for the correct `clk` setting for your rate and mode. The `gap` setting is measured in clock cycles and can remain the same for all variations.

### Running the Bidirectional Data Integrity Tests for the 1× 1.25-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 6482 DSP with the bidirectional data integrity test for the 1× 1.25-Gbaud variation, perform the following steps:

1. Follow the instructions in "Programming the Altera FPGA for the 1× 1.25-Gbaud Design Variation" on page 13.

   If you have not reprogrammed the FPGA since programming it for this design variation, you can skip this step.

   If you have reprogrammed the FPGA 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. To stop the TI 6482 DSP program that was running previously, if you have not already done so, perform the following steps:
   a. In the CCS driver window for the TMS320TCI6482 EVM, on the left side, click the **Halt** icon to stop the program. Figure 13 on page 24 shows the icon.
   b. On the Project menu, click **Close**.

4. Perform steps 3 to 12 in "Programming the TI 6482 DSP for the 1× 1.25-Gbaud Design Variation" on page 16.

To run the bidirectional data integrity test on the FPGA and on the TI 6482 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 42
pload 256
init
```

3. Perform step 13 in "Programming the TI 6482 DSP for the 1× 1.25-Gbaud Design Variation" on page 16.
4. At the `Serial RapidIO>` prompt, type the following command:

```
dit 10000000
```

The **nios2-terminal** displays a running count of packets transmitted and checked. This test 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 15 shows a sample **nios2-terminal** window after the following steps are performed:

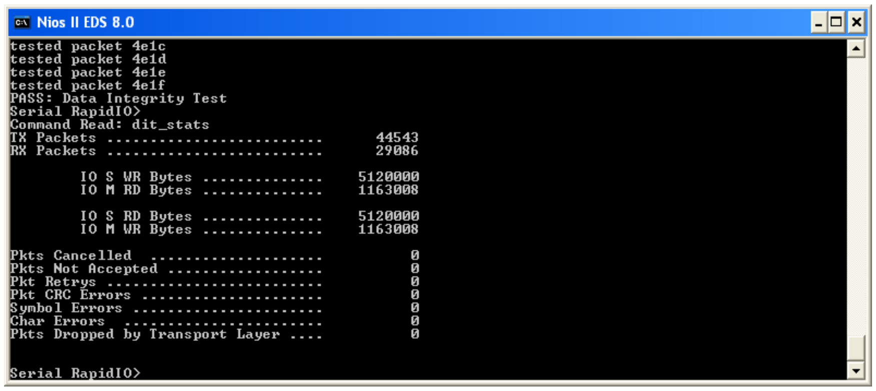
1. Type the following command sequence:

```
stop
r
rate 1250
clk 42
pload 256
init
```

2. Perform step 13 in "Programming the TI 6482 DSP for the 1× 1.25-Gbaud Design Variation" on page 16.
3. Type the following command sequence:

```
dit 20000
dit_stats
```

**Figure 15.** Sample Bidirectional Data Integrity Test and dit_stats Display at 1× 1.25 Gbaud



```
tested packet 4e1c
tested packet 4e1d
tested packet 4e1e
tested packet 4e1f
PASS: Data Integrity Test
Serial RapidIO>
Command Read: dit_stats
TX Packets .........................      44543
RX Packets .........................      29086

        IO S WR Bytes ..............    5120000
        IO M RD Bytes ..............    1163008

        IO S RD Bytes ..............    5120000
        IO M WR Bytes ..............    1163008

Pkts Cancelled  ....................          0
Pkts Not Accepted ..................          0
Pkt Retrys .........................          0
Pkt CRC Errors .....................          0
Symbol Errors ......................          0
Char Errors  .......................          0
Pkts Dropped by Transport Layer ....          0

Serial RapidIO>
```

In Figure 15, the bidirectional_dit test tests `0x4E20` (decimal 20000) packets for data integrity, to implement the `dit` command. Packet numbering is from 0x`0` 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 6482 DSP also writes and reads data as it performs data integrity checking. At the time of the statistics report, the TI DSP had transmitted 4543 write packets with a total payload of 1163008 bytes, and 4543 read request packets to check the integrity of the data it wrote. In response, the FPGA transmitted 1163008 bytes of data in 4543 read response packets.

In contrast to the other five RapidIO variations, the 1× 1.25 Gbaud RapidIO 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.

The previous command sequence causes 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*.

### Running the Bidirectional Data Integrity Tests for the 4× 1.25-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 6482 DSP with the bidirectional data integrity test for the 4× 1.25-Gbaud variation, perform the following steps:

1. Follow the instructions in "Programming the Altera FPGA for the Remaining RapidIO Design Variations" on page 15.

   ☞ If you have not reprogrammed the FPGA since programming it for this design variation, you can skip this step.

   ☞ If you have reprogrammed the FPGA 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_1250.pjt** file to specify that the **main_bidirectional_dit.c** file be compiled.
3. To stop the TI 6482 DSP program that was running previously, if you have not already done so, perform the following steps:
   a. In the CCS driver window for the TMS320TCI6482 EVM, on the left side, click the **Halt** icon to stop the program. Figure 13 on page 24 shows the icon.
   b. On the Project menu, click **Close**.
4. Perform steps 3 to 12 in "Programming the TI 6482 DSP for the 1× 1.25-Gbaud Design Variation" on page 16.

To run the bidirectional data integrity test on the FPGA and on the TI 6482 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
init
rate 1250
mode 4
gap 24
clk 83
pload 256
init
```

3. Perform step 13 in "Programming the TI 6482 DSP for the 1× 1.25-Gbaud Design Variation" on page 16.
4. Return to the **nios2-terminal** session and type the following command:

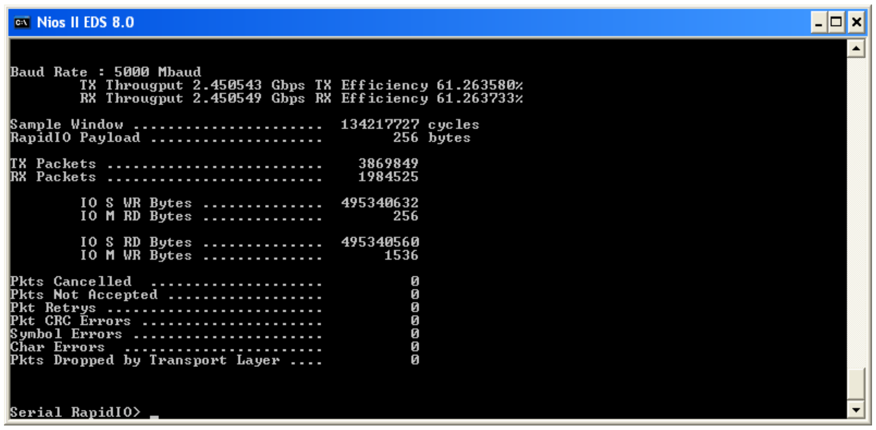
```
dit 80000000 ↵
```

5. To view dynamic statistics gathering, at the `Serial RapidIO>` prompt, type `stats`.

6. To stop the dynamic statistics display, at the **nios2-terminal**, type `z`.
7. To display the packet integrity testing results, type `dit_stats`.

Figure 16 shows a sample of the output from the `stats` command. It shows traffic initiated from both the FPGA and from the TI DSP.

**Figure 16.** Sample Bidirectional Data Integrity Test stats Display at 4× 1.25 Gbaud



```
Baud Rate : 5000 Mbaud
        TX Througput 2.450543 Gbps TX Efficiency 61.263580%
        RX Througput 2.450549 Gbps RX Efficiency 61.263733%

Sample Window ......................  134217727 cycles
RapidIO Payload ....................        256 bytes

TX Packets .........................    3869849
RX Packets .........................    1984525

        IO S WR Bytes ..............  495340632
        IO M RD Bytes ..............        256

        IO S RD Bytes ..............  495340560
        IO M WR Bytes ..............       1536

Pkts Cancelled  ....................          0
Pkts Not Accepted ..................          0
Pkt Retrys .........................          0
Pkt CRC Errors .....................          0
Symbol Errors ......................          0
Char Errors  .......................          0
Pkts Dropped by Transport Layer ....          0


Serial RapidIO> _
```

The previous command sequence causes 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*.

### Running the Data Integrity Tests for the Remaining Design Variations

Program and run the remaining design variations—the 1× and 4× variations at 2.5 and at 3.125 Gbaud—according to the instructions in "Running the Bidirectional Data Integrity Tests for the 4× 1.25-Gbaud Design Variation" on page 30, using the appropriate rate and mode settings. You must program the FPGA for the correct variation and program the TI 6482 DSP for the bidirectional data integrity test and the correct rate setting.

When you run the program, specify the correct `rate`, `mode`, and `clk`, and specify the `gap`. Refer to Table 1 on page 7 for the correct `clk` setting for your rate and mode. The `gap` setting is measured in clock cycles and can remain the same for all variations.

### 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 6482 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_bittware_srio_ref_design/<*type*>_based/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× 1.25-Gbaud Design Variation" on page 13 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 command:

```
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× 1.25-Gbaud Design Variation" on page 18 or in "Running the Unidirectional Performance Tests for the 4× 1.25-Gbaud Design Variation" on page 20, depending on your design variation.

12. Return to the SignalTap II window and click the **Autorun Analysis** icon. Figure 17 shows the icon.

**Figure 17.** The SignalTap II Embedded Logic Analyzer Autorun Analysis Icon

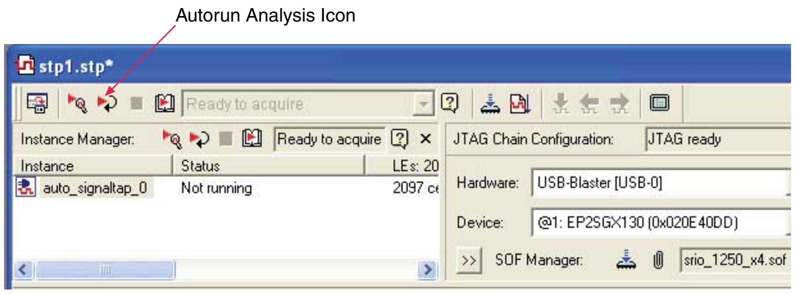
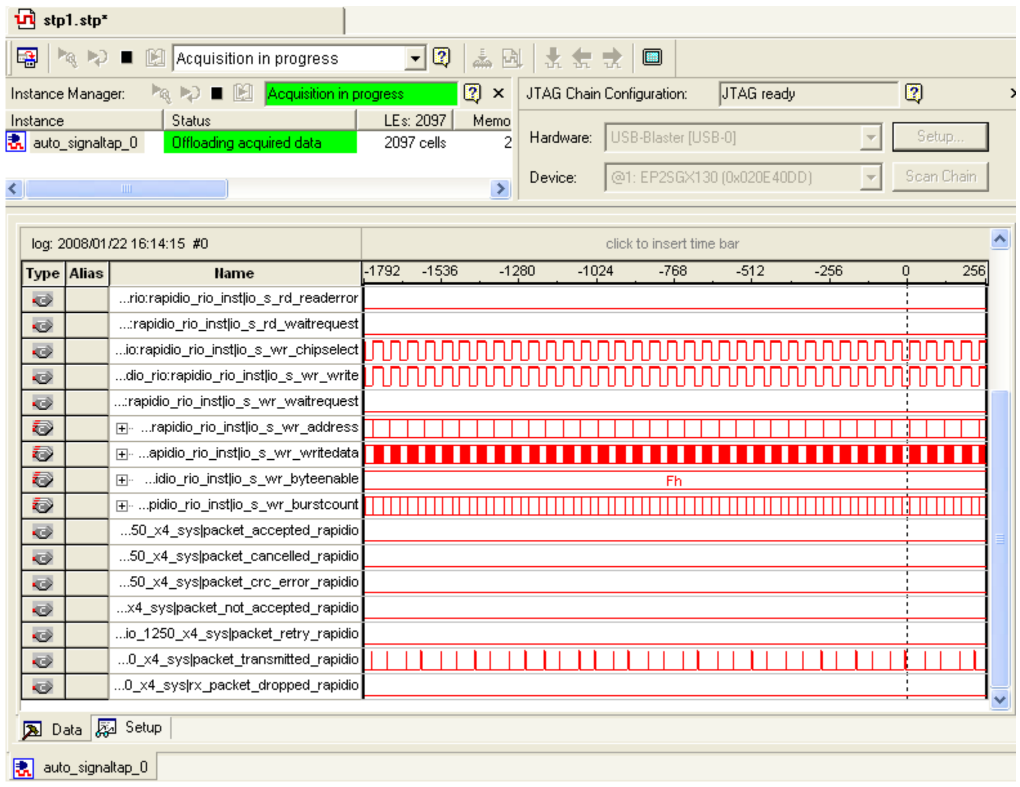


Figure 18 shows the SignalTap II Embedded Logic Analyzer. The window displays activity on the I/O slave write interface of the RapidIO MegaCore function.

**Figure 18.** SignalTap II Embedded Logic Analyzer Session


stp1.stp*
Acquisition in progress
Instance Manager:
Acquisition in progress
JTAG Chain Configuration:
JTAG ready
Instance
Status
LEs: 2097
Memo
auto_signaltap_0
Offloading acquired data
2097 cells
2
Hardware:
USB-Blaster [USB-0]
Setup...
Device:
@1: EP2SGX130 (0x020E40DD)
Scan Chain
log: 2008/01/22 16:14:15 #0
click to insert time bar
Type
Alias
Name
-1792
-1536
-1280
-1024
-768
-512
-256
0
256
...rio:rapidio_rio_inst|io_s_rd_readerror
...:rapidio_rio_inst|io_s_rd_waitrequest
...io:rapidio_rio_inst|io_s_wr_chipselect
...dio_rio:rapidio_rio_inst|io_s_wr_write
...:rapidio_rio_inst|io_s_wr_waitrequest
...rapidio_rio_inst|io_s_wr_address
...apidio_rio_inst|io_s_wr_writedata
...idio_rio_inst|io_s_wr_byteenable
Fh
...pidio_rio_inst|io_s_wr_burstcount
...50_x4_sys|packet_accepted_rapidio
...50_x4_sys|packet_cancelled_rapidio
...50_x4_sys|packet_crc_error_rapidio
...x4_sys|packet_not_accepted_rapidio
...io_1250_x4_sys|packet_retry_rapidio
...0_x4_sys|packet_transmitted_rapidio
...0_x4_sys|rx_packet_dropped_rapidio
Data
Setup
auto_signaltap_0

## Performance Summary

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

Throughput is limited by the TI 6482 DSP. When the incoming traffic on the RapidIO serial link exceeds the capacity of the TI 6482 DSP, it begins issuing packet-retry control symbols.

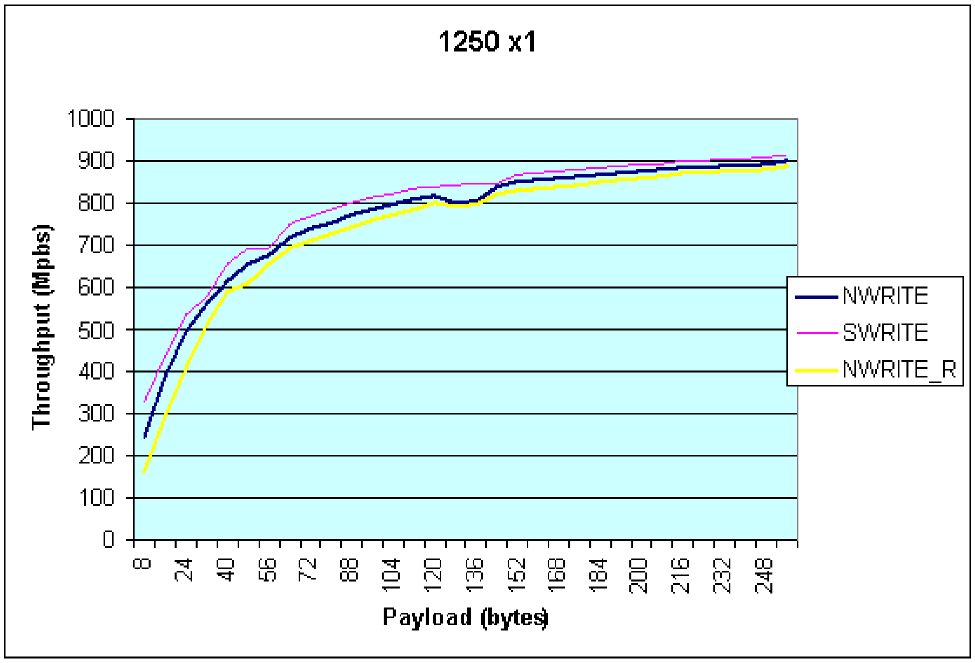
The maximum achievable throughput in this system is limited by the TI 6482 DSP.

For details about the bandwidth limitation of the TI 6482 DSP, refer to *Advisory 2.0.16 SRIO: Performance Issues Identified Prohibiting Full Utilization of Pin Bandwidth* in *TMS320TCI6482 Digital Signal Processor Silicon Revisions 2.0, 1.1 Silicon Errata*, available at the Texas Instruments website, www.ti.com.

The performance results reported in this section were obtained using the Quartus II 8.0 and CCS v3.2 software, and an assembly version 508550-2002 Rev C TI DSP card and a GXAM Rev 0.

The graphs in Figure 19 to Figure 24 show the performance numbers measured using this design. Each graph shows the performance for the three RapidIO WRITE transaction types.

**Figure 19.** Throughput with Increasing Payload in ×1 RapidIO MegaCore function at 1.25 Gbaud *(Note 1)(2)(3)*



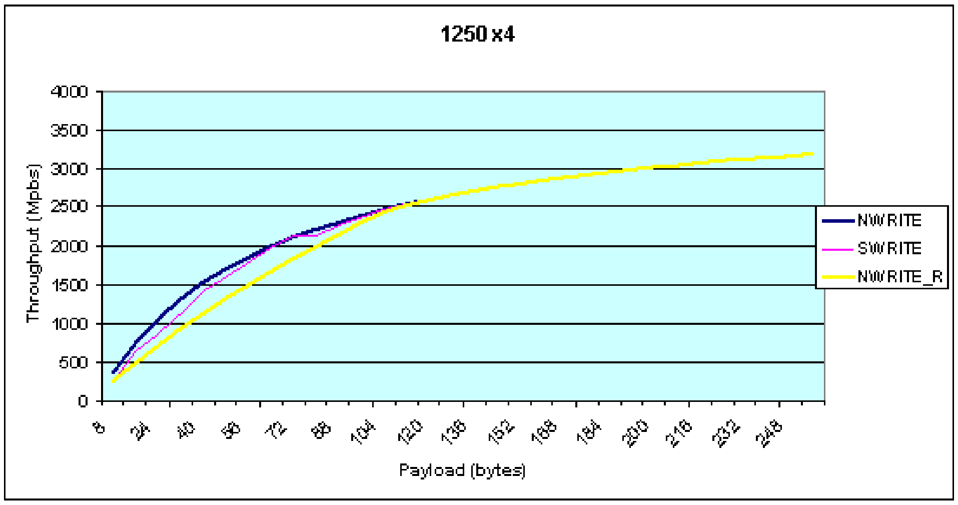
**Notes to Figure 19:**

(1) NWRITE transactions: The TI 6482 DSP was able to sink all traffic received, and issued no packet retries.

(2) NWRITE_R transactions: The TI 6482 DSP issued minimal packet retries.

(3) SWRITE transactions: The TI 6482 DSP issued no packet retries.

**Figure 20.** Throughput with Increasing Payload in ×4 RapidIO MegaCore function at 1.25 Gbaud *(Note 1)(2)(3)*



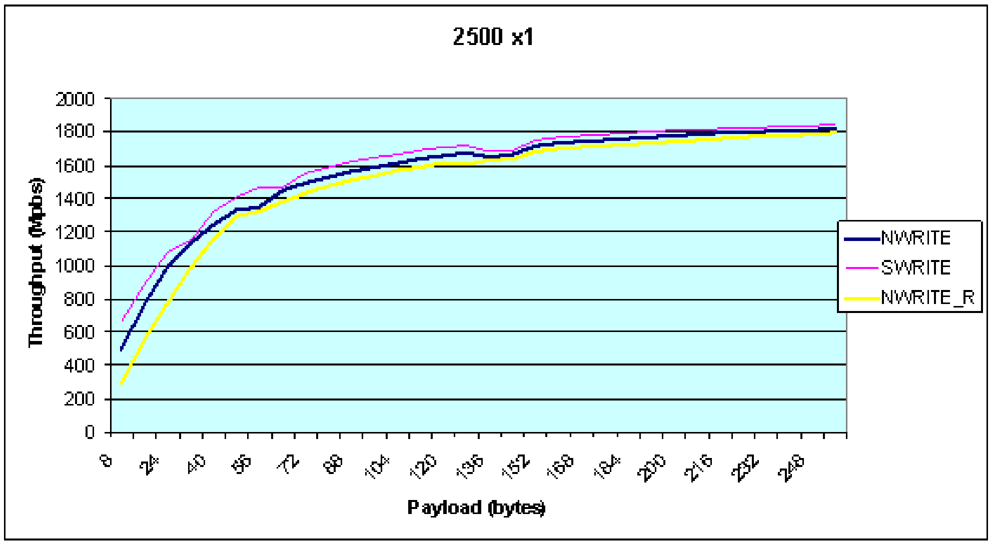
**Notes to Figure 20:**

(1) NWRITE transactions: The TI 6482 DSP issued significant packet retries.

(2) NWRITE_R transactions: The TI 6482 DSP issued no packet retries.

(3) SWRITE transactions: The TI 6482 DSP issued packet retries.

**Figure 21.** Throughput with Increasing Payload in ×1 RapidIO MegaCore function at 2.5 Gbaud *(Note 1)(2)(3)*



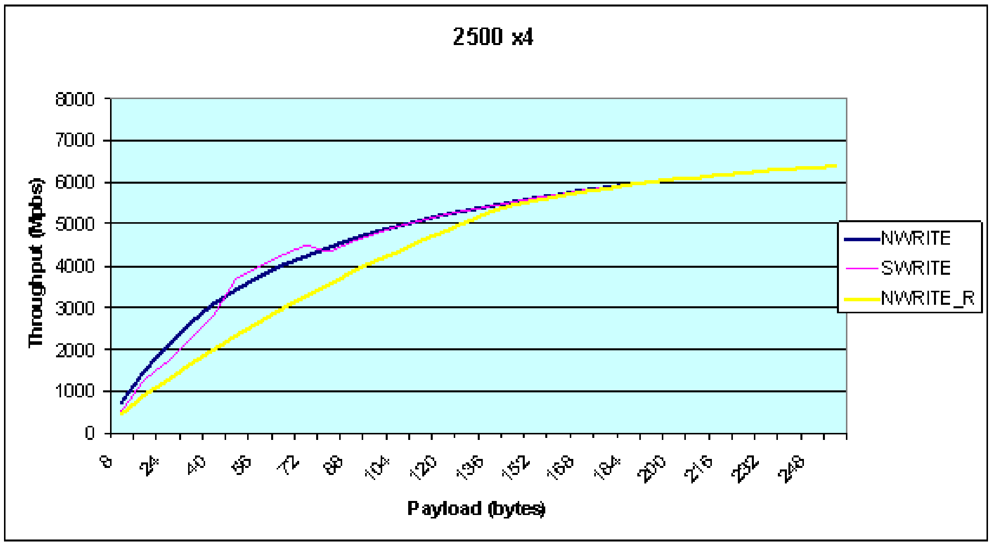
**Notes to Figure 21:**

(1) NWRITE transactions: The TI 6482 DSP issued no packet retries.

(2) NWRITE_R transactions: The TI 6482 DSP issued no packet retries.

(3) SWRITE transactions: The TI 6482 DSP issued no packet retries.

**Figure 22.** Throughput with Increasing Payload in ×4 RapidIO MegaCore function at 2.5 Gbaud *(Note 1)(2)(3)*



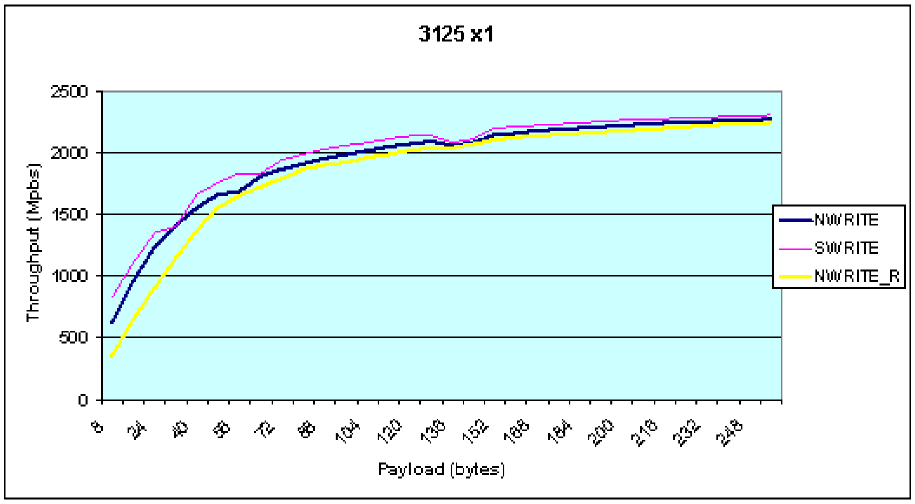
**Notes to Figure 22:**

(1) NWRITE transactions: The TI 6482 DSP issued packet retries.

(2) NWRITE_R transactions: The TI 6482 DSP issued no packet retries.

(3) SWRITE transactions: The TI 6482 DSP issued packet retries.

**Figure 23.** Throughput with Increasing Payload in ×1 RapidIO MegaCore function at 3.125 Gbaud *(Note 1)(2)(3)*



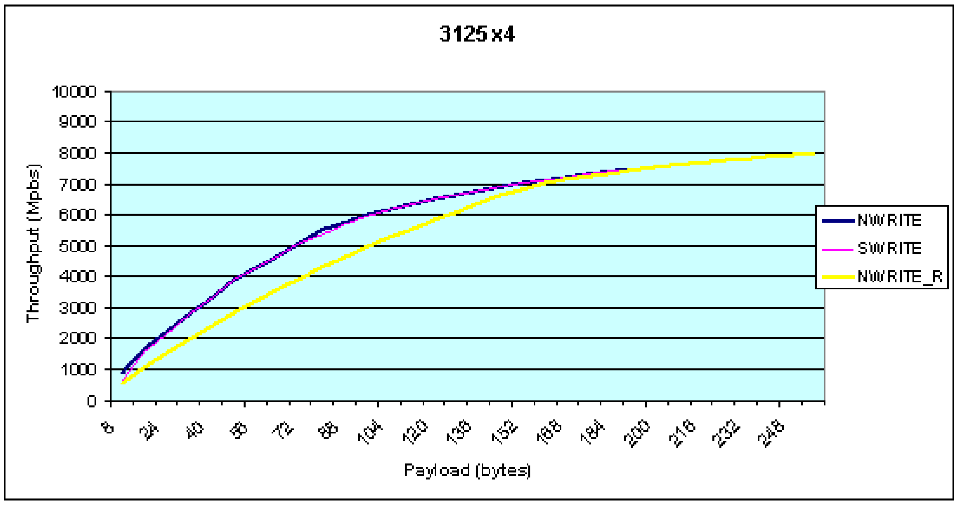
**Notes to Figure 23:**

(1) NWRITE transactions: The TI 6482 DSP issued no packet retries.

(2) NWRITE_R transactions: The TI 6482 DSP issued no packet retries.

(3) SWRITE transactions: The TI 6482 DSP issued no packet retries.

**Figure 24.** Throughput with Increasing Payload in ×4 RapidIO MegaCore function at 3.125 Gbaud *(Note 1)(2)(3)*



**Notes to Figure 24:**

(1) NWRITE transactions: The TI 6482 DSP issued packet retries.

(2) NWRITE_R transactions: The TI 6482 DSP issued no packet retries.

(3) SWRITE transactions: The TI 6482 DSP issued packet retries.

## 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
- Doorbell Messages

The design can saturate the link with line rate traffic using the DMA approach only in the 1× 1.25 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.

The maximum achievable throughput in this system is limited by the TI 6482 DSP. For additional details, refer to "Performance Summary" on page 34.

## Referenced Documents

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

- *Advisory 2.0.16 SRIO: Performance Issues Identified Prohibiting Full Utilization of Pin Bandwidth* in *TMS320TCI6482 Digital Signal Processor Silicon Revisions 2.0, 1.1 Silicon Errata*, available at the Texas Instruments website, www.ti.com
- *RapidIO MegaCore Function User Guide*
- *TMS320TCI6482 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 |
|---|---|---|
| November 2008 v2.0 | Added the following new reference design features: <br>■ NREAD transactions <br>■ Simultaneous bidirectional operation <br>■ Support for automatic data integrity checking | Updated document for new reference design features. |
| January 2008 v1.1 | Updated instructions for reference design file. | — |
| January 2008 v1.0 | Initial release. | — |

.