AC446
Application Note
Optimization Techniques to Improve DDR Throughput for RTG4 Devices

Microsemi
a Microchip company

a MICROCHIP company

Microsemi Headquarters
One Enterprise, Aliso Viejo,
CA 92656 USA
Within the USA: +1 (800) 713-4113
Outside the USA: +1 (949) 380-6100
Sales: +1 (949) 380-6136
Fax: +1 (949) 215-4996
Email: sales.support@microsemi.com
www.microsemi.com

About Microsemi

Microsemi, a wholly owned subsidiary of Microchip Technology Inc. (Nasdaq: MCHP), offers a comprehensive portfolio of semiconductor and system solutions for aerospace & defense, communications, data center and industrial markets. Products include high-performance and radiation-hardened analog mixed-signal integrated circuits, FPGAs, SoCs and ASICs; power management products; timing and synchronization devices and precise time solutions, setting the world's standard for time; voice processing devices; RF solutions; discrete components; enterprise storage and communication solutions, security technologies and scalable anti-tamper products; Ethernet solutions; Power-over-Ethernet ICs and midspans; as well as custom design capabilities and services. Learn more at www.microsemi.com.

51900446. 6.0 9/21

Contents

Figures

Tables

1 Revision History

The revision history describes the changes that were implemented in the document. The changes are listed by revision, starting with the most current publication.

1.1 Revision 6.0

Updated Figure 6, page 8 and Figure 30, page 22.

1.2 Revision 5.0

The following is a summary of the changes made in this revision.

- Removed the Design Files Appendix section.
- Updated Figure 7, page 9
- Updated Figure 8, page 10

1.3 Revision 4.0

The following is a summary of the changes made in this revision.

- Added Appendix 1: Programming the Device Using FlashPro Express, page 26.
- Added Appendix 2: Running the TCL Script, page 29.
- Removed the references to Libero version numbers.

1.4 Revision 3.0

Updated the document for Libero SoC 11.9 SP1.

1.5 Revision 2.0

Updated the document for Libero SoC 11.8 SP2.

1.6 Revision 1.0

The first publication of this document.

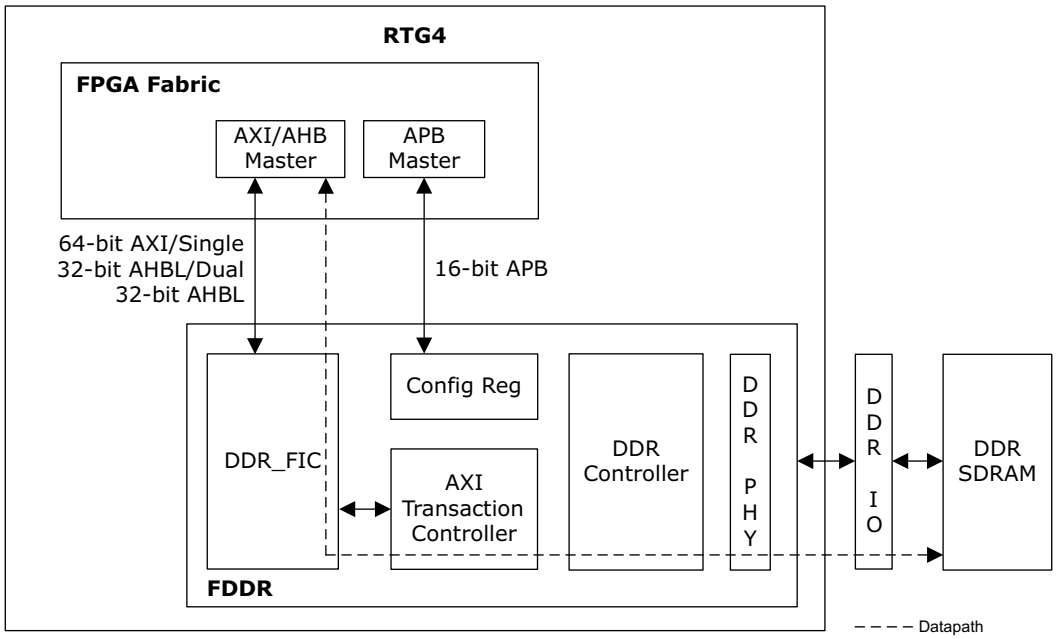
2 Optimization Techniques to Improve DDR Throughput for RTG4 Devices

This application note describes the optimization techniques for improving DDR3 SDRAM throughput using a reference design for the RTG4™ field programmable gate array (FPGA) Development Kit. It also describes simulating the design using the Micron DDR3 SDRAM simulation model.

The RTG4 devices have two fabric DDR (FDDR) blocks. Each FDDR subsystem is a hardened ASIC block used to interface with DDR2, DDR3, or low power DDR1 (LPDDR1) memories and access high-speed DDR memories for high-speed data transfer.

The DDR memory connected to the FDDR subsystem is accessed by the master logic implemented in the FPGA fabric. The FPGA fabric master communicates with the FDDR subsystem through AXI or AHB interfaces. The following figure shows the data path for DDR memory access by the FDDR subsystem.

Figure 1 • **FDDR Datapath for AXI/AHB Master**



The AXI master interface performs burst transfers that provide an efficient data access path with maximum throughput. The throughput of DDR depends on various parameters. This reference design describes optimization techniques that improve efficiency and provide better throughput.

An AXI master on the fabric logic performs AXI burst transfers of 2 KB, 4 KB, 8 KB, 16 KB, and 32 KB size. During a write operation, the AXI master writes incremental patterns to the DDR3 memory and returns the AXI clock count to the host PC (through the UART interface) for throughput calculation. During a read operation, the AXI master reads data from DDR3 memory and checks for data mismatch in the fabric logic. If there is a data mismatch then the error status is indicated through an on-board LED. A separate demo utility application calculates throughput using AXI clock count input.

2.1 Design Requirements

The following table lists the design requirements to run the design.

Table 1 • **Design Requirements**

Requirement	Version
Hardware	
RTG4 Development Kit • 12 V adapter • USB A to Mini-B cable	RTG4-DEV-KIT Rev C
Host PC or Laptop	64-bit Windows 7 and 10
Software	
Libero® System-on-Chip (SoC)	**Note:** Refer to the `readme.txt` file provided in the design files for the software versions used with this reference design.
FlashPro Express	
Host PC drivers	USB to UART drivers
Host PC demo utility application	–

Note: Libero SmartDesign and configuration screen shots shown in this guide are for illustration purpose only. Open the Libero design to see the latest updates.

2.2 Prerequisites

Before you start:

1. Download and install Libero SoC (as indicated in the website for this design) on the host PC from the following location: *https://www.microsemi.com/product-directory/design-resources/1750-libero-soc*
2. For demo design files download link:

   http://soc.microsemi.com/download/rsc/?f=rtg4_ac446_df

2.3 Optimization Techniques

Demand for performance continues to increase, and existing memories face limitations in throughput at higher data rates. The following optimization techniques help to achieve maximize throughput:

2.3.1 Frequency of Operation

The FDDR subsystem supports clock management dividers. The divider ratios can be selected from the Clock Configurator for DDR clocks (FDDR_CLK) and DDR_FIC clock. This reference design uses a 2:1 ratio between FDDR_CLK and DDR_FIC because the best throughput result is obtained for this ratio. The reference design uses a 64-bit AXI interface in the FPGA fabric, which operates at a maximum possible frequency of 166.66 MHz. The FDDR_CLK frequency is configured at 333 MHz.

2.3.2 Burst Length

The MDDR and FDDR subsystems support the DRAM burst lengths of 4, 8, or 16, depending on the configured bus width and the DDR type. The AXI transaction controller in the MDDR and FDDR subsystems support up to 16-beat burst read and write. The burst length (write and read) of AXI and DRAM affect the performance, but by setting the maximum supported burst length for DDR SDRAM and AXI interface optimal performance can be achieved. The reference design uses a DDR SDRAM burst length of 8 and an AXI write and read burst length of 16.

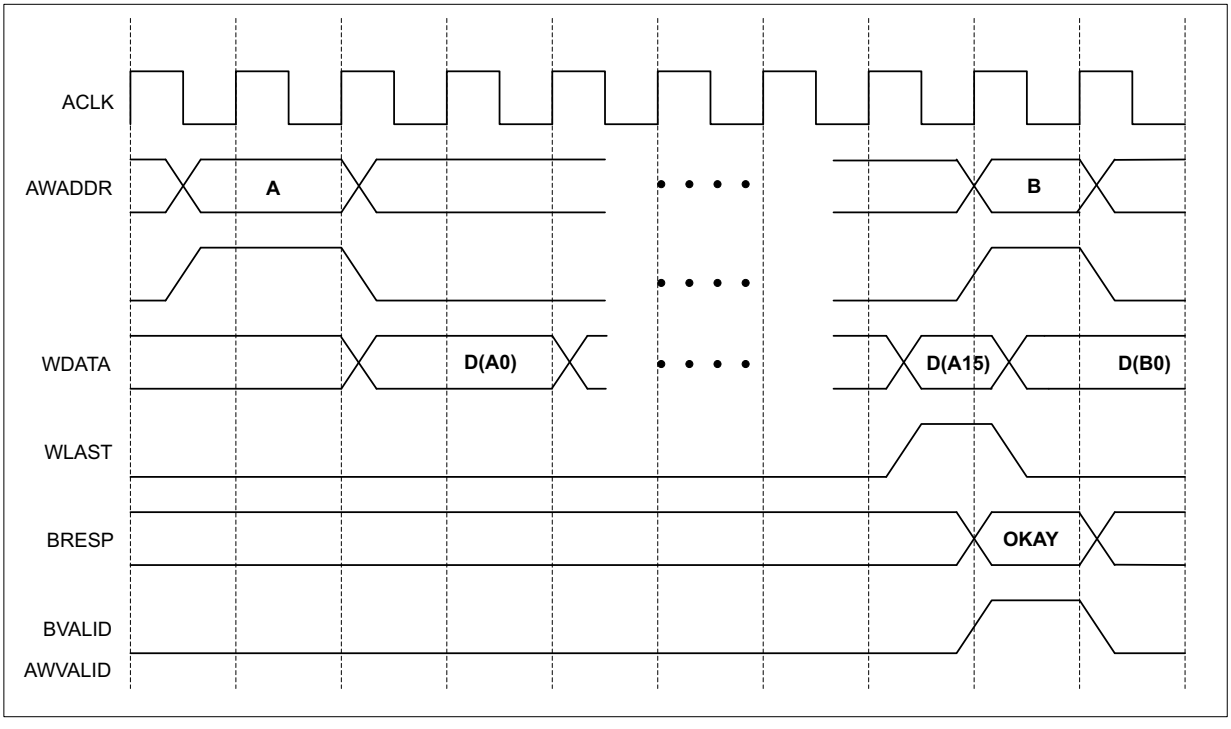
Note: The reference design is run on the RTG4 Development Kit board, which has the RT4G150 device and a DDR3 SDRAM from Micron with the part number MT41K256M8.

2.3.3 AXI Master without Write Response State

When the AXI master sends the last word (D[A15]), the WLAST signal is asserted, which indicates that the last word of the current write burst is transferred. When the AXI slave in the DDR subsystem accepts all the data items, it asserts the write response (BVALID) back to the master to indicate that the write transaction is complete, as shown in the following figure. When using the AXI protocol, the AXI master waits for the write response before initiating the next write transaction. However, waiting for write response increases latency and decreases overall throughput. The AXI master can send the second burst write address (B) without waiting for the write response of the first burst. This improves the write throughput by reducing the wait states.

This application note is focused on optimal throughput, and therefore the write response channel is not verified. It is recommended that when using this technique, the write response channel is used concurrently with starting the next transfer to ensure that the previous write data is accepted. The AXI protocol has a defined methodology for handling the termination of write burst transactions. This must be followed if the write response channel returns an incorrect value.

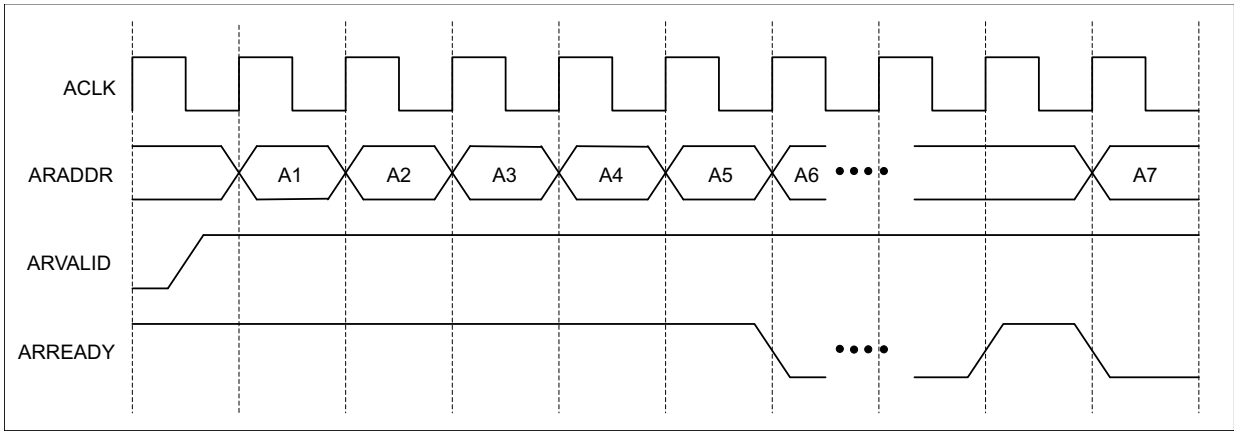
Figure 2 • **Write Transaction Timing Diagram without Write Response**



2.3.4 Read Address Queuing

The FDDR subsystem supports up to four outstanding read transactions. The following figure shows the burst read address queuing timing diagram. In 2:1 clock ratio, the FDDR controller starts the burst read transaction before the FIFO FULL command, which allows the AXI master to send five burst read addresses. The AXI master increments the burst read address as long as the AXI slave in the DDR subsystem asserts the ARREADY signal. The burst read address queuing significantly increases the read throughput compared to the normal AXI read sequence. Table 7, page 25 shows this significant increase. Read address queuing does not reduce the initial latency associated with a DDR memory read access. However, by issuing multiple reads in sequence, the read throughput is increased significantly when compared to the normal AXI read sequence.

Figure 3 • **FDDR Datapath for AXI/AHB Master**



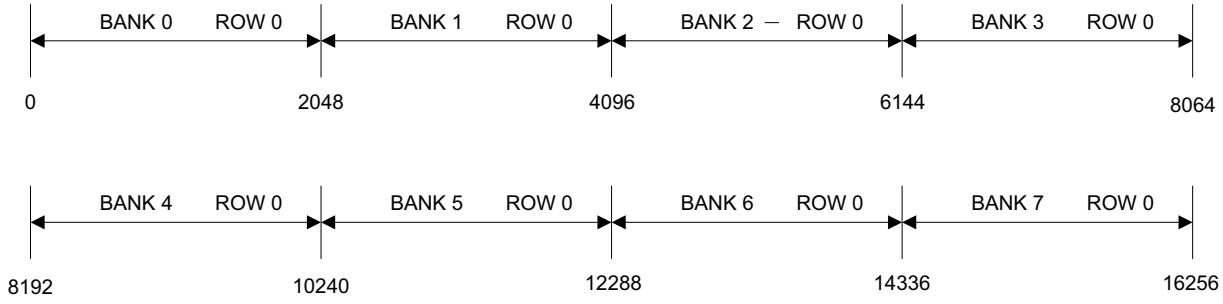
2.3.5 Series of Writes and Reads

The FDDR subsystem's performance depends on the method of data transfer between the DDR SDRAM and the AXI master. The following methods of data transfer reduce optimal performance:

- Single-beat burst read and write operation
- Random read and write operation
- Switching between read and write operation

The FDDR subsystem's performance increases while performing a series of reads or writes from the same bank and row. The following figure shows the AXI to DDR3 address mapping for the DDR3 SDRAM on the RTG4 Development Kit board.

Figure 4 • **AXI to DDR3 Address Mapping**



When the AXI address crosses 0x0800, the DDR subsystem activates Row 0 of Bank 1. Row 1 of Bank 0 is activated only when the AXI address crosses 0x4000. If a new row is accessed every time, it must be pre-charged first. This means that additional time is required before accessing a row and this reduces the overall throughput. Understanding the internal memory layout of the DDR subsystem and how it maps to the AXI address helps to minimize row changes and increase the overall throughput.

2.3.6 DDR Configuration Tuning

The DDR SDRAM datasheet provides the timing parameters required for the memory operation in terms of time units. These timings must match with the configuration registers in the FDDR controller. The timing parameters are entered in terms of several DDR clock cycles in the DDR Configurator. The selection of minimum write or read delay values may result in optimal performance. Implementing this approach depends on the vendor memory device that is used and its DDR controller and PHY blocks. It also requires extensive memory testing to ensure that the memory transfers are stable. For this reference design, the FDDR configuration register file is provided along with the design file.

The following table lists the key timing parameter values used in the reference design.

Table 2 • **Tuned DDR Timing Parameters**

Parameter	Values for Reference Design
CAS	5 (clks)
RAS min	12
RAS max	22528
RCD	5
RP	5
REFI	2592
RC	17
RFC	54
WR	5
FAW	10

For more information about improved throughput numbers after using optimization techniques, refer to Table 6, page 24 and Table 7, page 25.

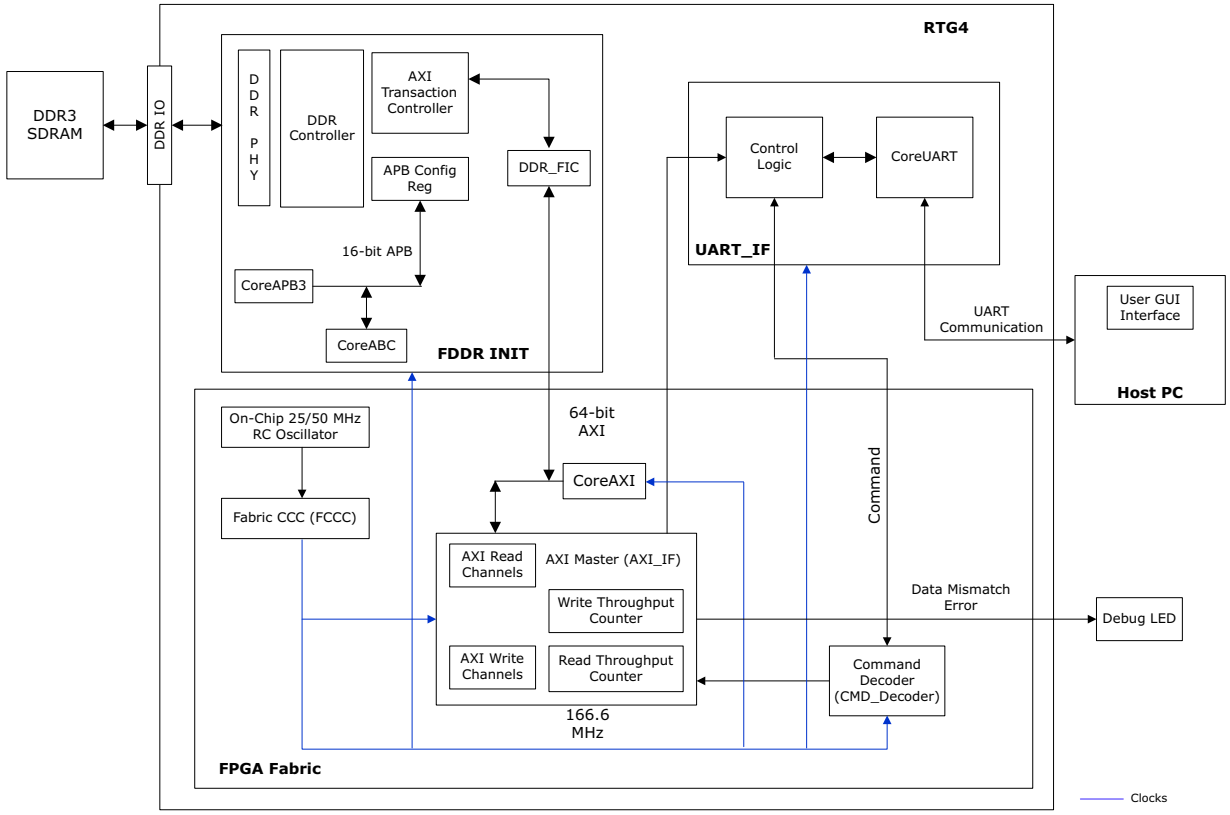
2.4 Design Description

The reference design consists of the following components:

- FDDR subsystem
- CoreABC with APB master
- CoreAXI
- CoreUART
- On-chip 25/50 MHz RC oscillator
- Fabric CCC (FCCC)
- AXI master (AXI_IF)
- Command decoder (CMD_Decoder)

The following figure shows the block diagram of the reference design.

Figure 5 • **Top-Level Block Diagram of the Reference Design**



The DDR_FIC interface is configured to use an AXI interface. CoreABC is used to initialize FDDR with required tuned configuration values through the APB interface. CoreUART and control_logic HDL modules are used as interfaces for writing to the host PC demo utility application. The demo utility application initiates AXI read and write operation using the CoreUART interface. The FDDR is configured to use the DDR3 memory interface through the AXI master interface in the fabric logic. Fabric CCC (FCCC) is configured to provide a 166.6 MHz reference clock to the fabric logic. The on-chip 25/50 MHz RC oscillator is the reference clock source for the FCCC.

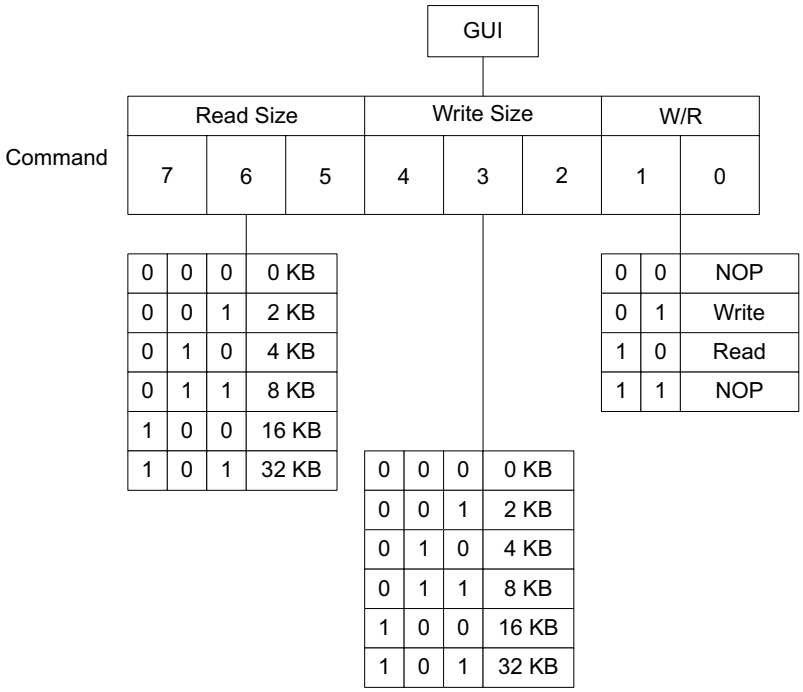
Table 3 • **FCCC Generated Clocks**

Clock Name	Frequency in MHz
FDDR_CLK	333
DDR_FIC	166.5
APB_CLK	41.667

The command decoder receives a command from the demo utility application through the UART interface, which is implemented using the CoreUART IP and generates read, write, write size, and read size signals.

The following figure shows command decoding.

Figure 6 • **Command Decoding**



The AXI master block consists of the following components:

- AXI read channel
- AXI write channel
- Write latency counter
- Read latency counter

It performs the write or read operation based on the input signals from the command decoder. During the write operation, the AXI master writes incremental pattern into the DDR3 memory and then measures write latency (number of AXI clocks) that is sent to the host PC demo utility application for throughput calculation. During the read operation, the AXI master reads the DDR3 memory and measures read latency (number of AXI clocks) that is sent to the host PC demo utility application for throughput calculation. During a read operation, if there is a data mismatch, an error status signal is shown using the on-board active LOW debug LED. The write latency counter counts the AXI clocks between AWVALID of the first data and WLAST of the last data.

Similarly, the read latency counter counts the AXI clocks between ARVALID of the first data and RLAST of the last data. During the write operation, the write address (AWADDR) starts from 0x00000000 and is incremented by 128 (16-beat burst). During the read operation, the read address (ARADDR) also starts from 0x00000000 and is incremented by 128. After each write or read operation, the AXI master sends the latency count value to the host PC for throughput calculation. The demo utility displays the number of AXI clocks and throughput value for write and read operations.

The following equation is applied to calculate the throughput:

Bandwidth (MB/s) = (16 ÷ [Total number of AXI clocks ÷ Total number of 16-beat bursts])×8×AXI Clock (MHz)

For more information about RTG4 FDDR, refer to *UG0573: RTG4 FPGA DDR Controller User Guide*.

For more information about accessing DDR3 memory using AXI interface, refer to *DG0625: Interfacing RTG4 with the External DDR3 Memory through the DDR Controller Demo Guide*.

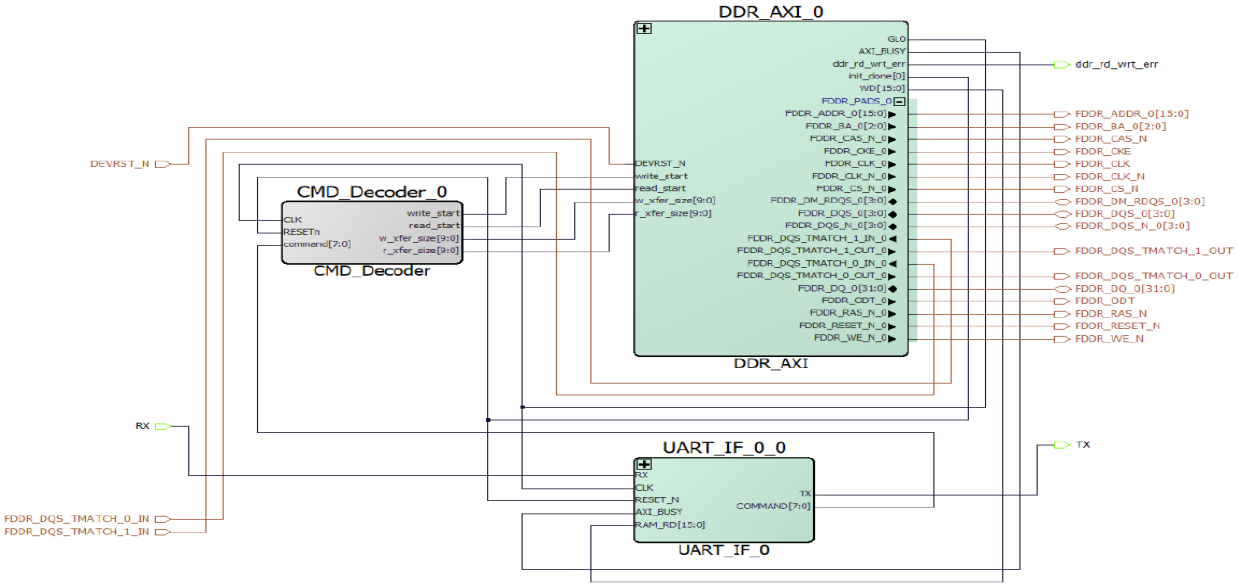
2.5 Hardware Implementation

This section describes:

- SmartDesign components
- FDDR configurations

The following figure shows the top-level SmartDesign component.

Figure 7 • **Top-Level SmartDesign Component**



2.5.1 SmartDesign Components

DDR_AXI_0: handles the data transactions between the DDR_FIC interface and the DDR3 SDRAM memory.

UART_IF_0: handles the communication between the host PC and the RTG4 Development Kit.

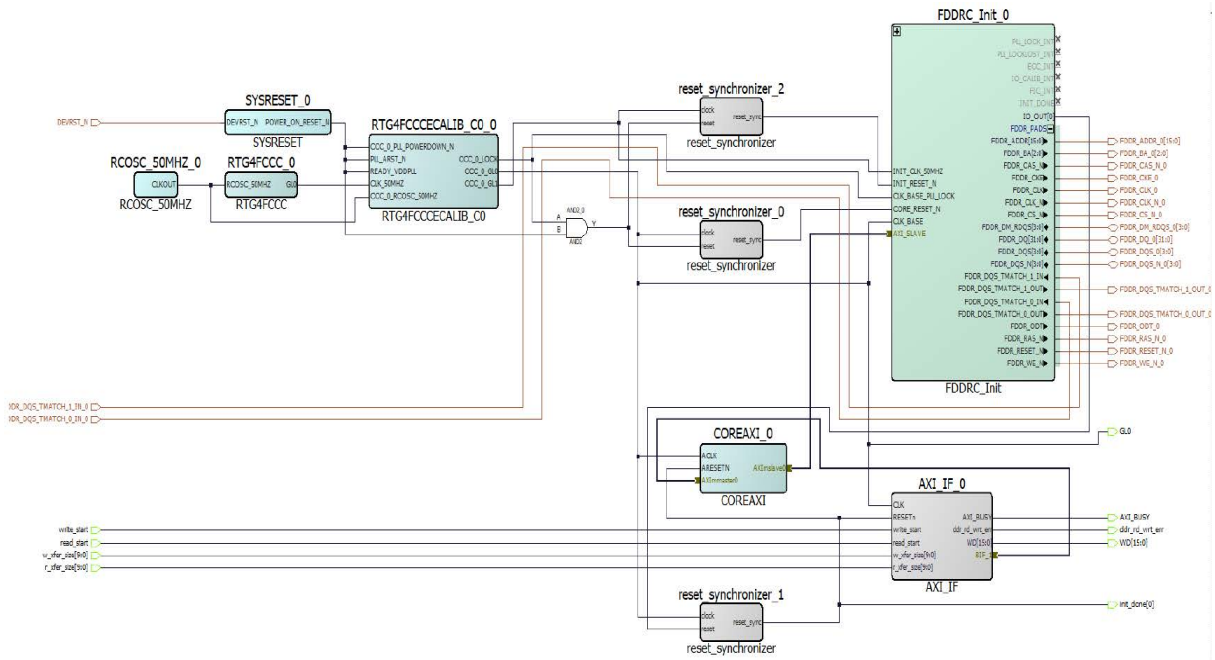
CMD_Decoder_0: used to receive a command from the host PC and generate required read, write, read size, and write size signals.

2.5.1.1 DDR_AXI_0

DDR_AXI_0 consists of the FDDR subsystem and the AXI_IF master logic. The AXI_IF_0 is an RTL code that implements the AXI read and write transactions. It receives the read/write commands, burst length (RLEN and WLEN), address, and data as inputs. Based on the inputs received, it communicates with the DDR3 memory through the FDDR subsystem.

The following figure shows the DDR_AXI_0 SmartDesign component.

Figure 8 • **DDR_AXI_0 SmartDesign Component**

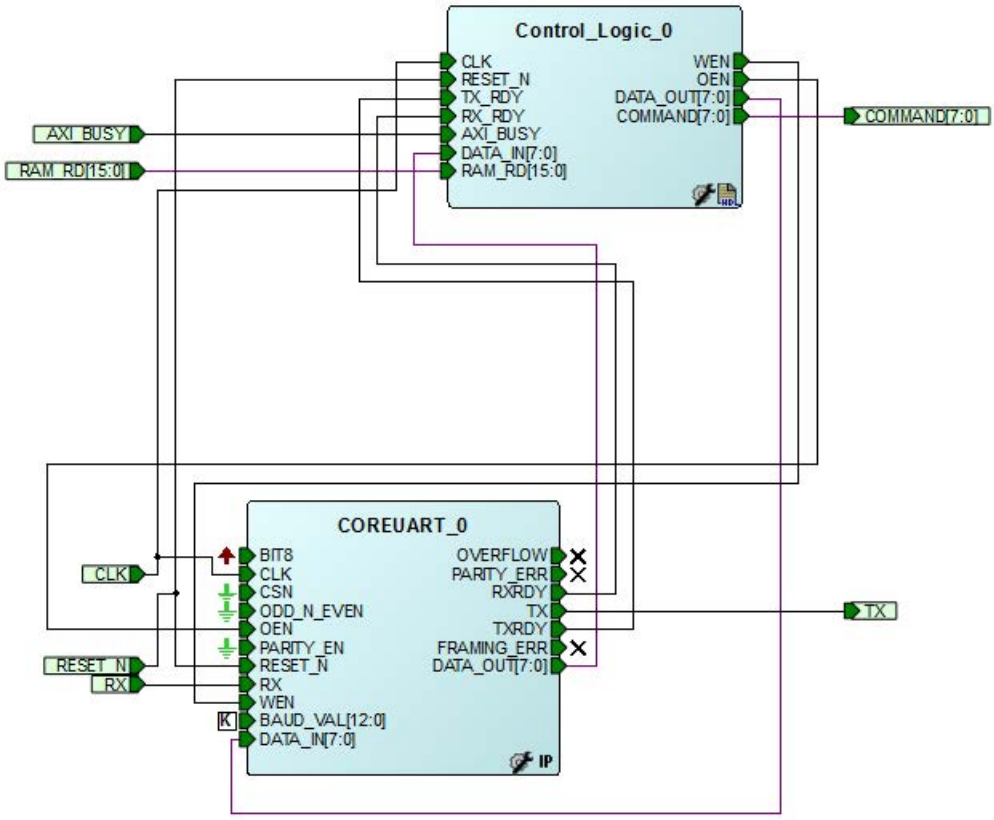


2.5.2 UART_IF_0

The UART_IF_0 SmartDesign component controls the UART communication between the host PC demo utility application and the AXI master logic. The COREUART_0 IP receives the UART signals from the host PC user interface. Control_logic_0 is an HDL wrapper for COREUART_0. It collects the data from COREUART_0 and sends a command to the command decoder. For different burst size write and read operations, the command is received from the demo utility and sent to the command decoder. The command decoder generates required read/write signals for the AXI master logic.

The following figure shows the UART_IF_0 SmartDesign component.

Figure 9 • **UART_IF_0 SmartDesign Component**



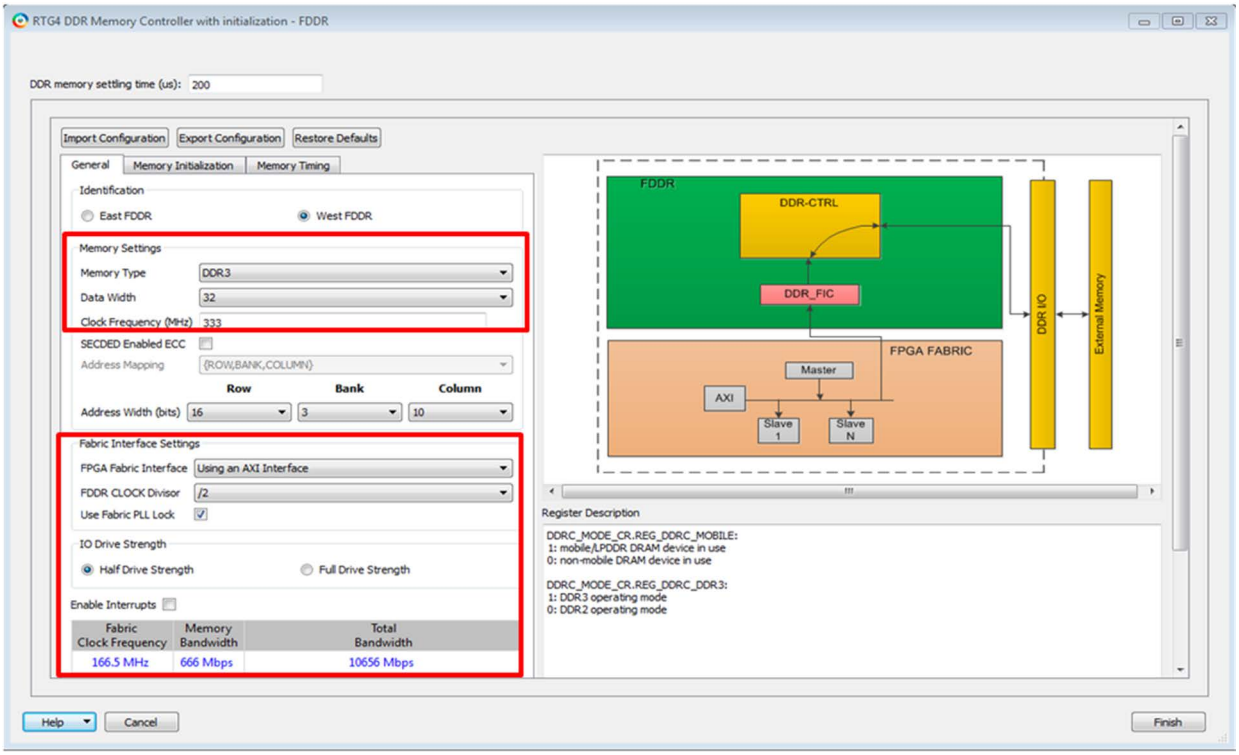
2.5.3 Configuring the FDDR Subsystem

This section describes how to configure the FDDR subsystem to perform data transactions with the external DDR3 memory device. For more information about the RTG4 DDR memory controller with initialization IP core, refer to *UG0573: RTG4 FPGA DDR Memory Controller User Guide*.

In the following procedure, the reference design uses **West FDDR** to access DDR3 with 32-bit data width and no ECC. The following steps describe how to configure FDDR and access it from the AXI master in the fabric logic:

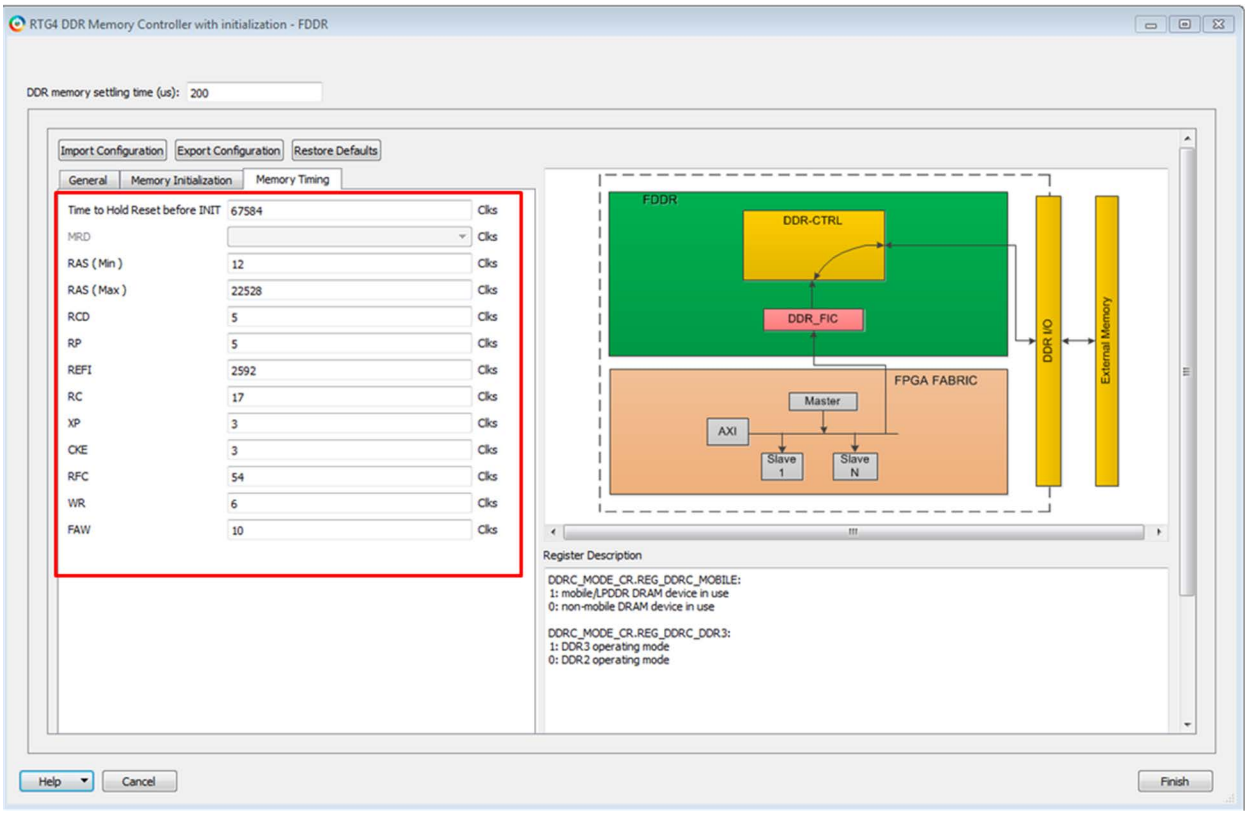
1. On the FDDR configurator **General** tab shown in the following figure, set the **DDR memory settling time** to 200 µs, and click **Import Configuration** to initialize the DDR memory.
   The FDDR subsystem registers must be initialized before accessing DDR memory through the FDDR subsystem. The FDDR configuration register file is provided along with the design file. Refer to Appendix 3: Design Files, page 30.

Figure 10 • **FDDR Memory Configuration**



The following figure shows the memory timing settings according to the tuned DDR configuration file.

Figure 11 • **FDDR Memory Timing**



2. Click **Finish**.
3. Instantiate the custom logic-AXI master, UART_IF, and a command decoder, and connect, as shown in Figure 8, page 10 and Figure 9, page 11.

2.6 Simulation Using Micron DDR3 SDRAM Model

The following steps describe how to set up and simulate the reference design:

1. Obtain the Micron DDR3 memory model files—the RTG4 Development Kit board has the DDR3 SDRAM from Micron with the part number MT41K256M8. The memory model used in the reference design supports this device.
2. Copy the `ddr3.v` and `ddr3_parameters.vh` simulation model files to the *\<Libero SoC project directory>\stimulus directory*.
3. Instantiate and connect the DDR3 memory model in the testbench, as shown in the following figure.

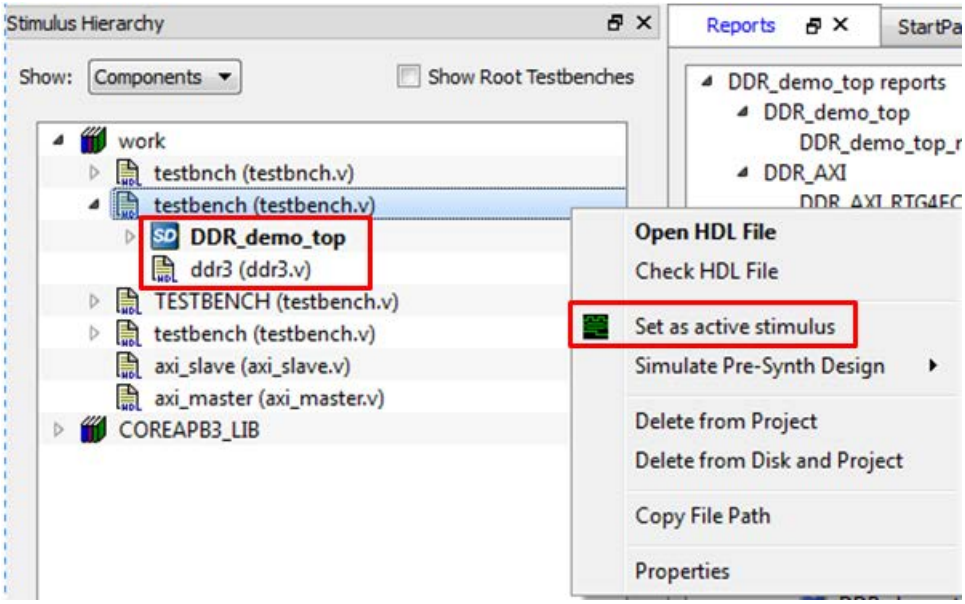
Figure 12 • **Instantiating Simulation Model**

```
ddr3 DDR3_0 (
    .rst_n(FDDR_RESET_N),
    .ck(FDDR_CLK),
    .ck_n(FDDR_CLK_N),

    .cke(FDDR_CKE),
    .cs_n(FDDR_CS_N),
    .ras_n(FDDR_RAS_N ),
    .cas_n(FDDR_CAS_N),
    .we_n(FDDR_WE_N),
    .addr(FDDR_ADDR[14:0]),
    .ba(FDDR_BA),
    .dm_tdqs(FDDR_DM_RDQS[0]),
    .dq(FDDR_DQ[7:0]),

    .dqs(FDDR_DQS[0]),
    .dqs_n(FDDR_DQS_N[0]),

    .odt(FDDR_ODT),
    .tdqs_n()
);

ddr3 DDR3_1 (
    .rst_n(FDDR_RESET_N),
    .ck(FDDR_CLK),
    .ck_n(FDDR_CLK_N),

    .cke(FDDR_CKE),
    .cs_n(FDDR_CS_N),
    .ras_n(FDDR_RAS_N ),
    .cas_n(FDDR_CAS_N),
    .we_n(FDDR_WE_N),
    .addr(FDDR_ADDR[14:0]),
    .ba(FDDR_BA),
    .dm_tdqs(FDDR_DM_RDQS[1]),
    .dq(FDDR_DQ[15:8]),

    .dqs(FDDR_DQS[1]),
    .dqs_n(FDDR_DQS_N[1]),

    .odt(FDDR_ODT),
    .tdqs_n()
);
```

```
ddr3 DDR3_2 (
    .rst_n(FDDR_RESET_N),
    .ck(FDDR_CLK),
    .ck_n(FDDR_CLK_N),

    .cke(FDDR_CKE),
    .cs_n(FDDR_CS_N),
    .ras_n(FDDR_RAS_N ),
    .cas_n(FDDR_CAS_N),
    .we_n(FDDR_WE_N),
    .addr(FDDR_ADDR[14:0]),
    .ba(FDDR_BA),
    .dm_tdqs(FDDR_DM_RDQS[2]),
    .dq(FDDR_DQ[23:16]),

    .dqs(FDDR_DQS[2]),
    .dqs_n(FDDR_DQS_N[2]),

    .odt(FDDR_ODT),
    .tdqs_n()
);

ddr3 DDR3_3 (
    .rst_n(FDDR_RESET_N),
    .ck(FDDR_CLK),
    .ck_n(FDDR_CLK_N),

    .cke(FDDR_CKE),
    .cs_n(FDDR_CS_N),
    .ras_n(FDDR_RAS_N ),
    .cas_n(FDDR_CAS_N),
    .we_n(FDDR_WE_N),
    .addr(FDDR_ADDR[14:0]),
    .ba(FDDR_BA),
    .dm_tdqs(FDDR_DM_RDQS[3]),
    .dq(FDDR_DQ[31:24]),

    .dqs(FDDR_DQS[3]),
    .dqs_n(FDDR_DQS_N[3]),

    .odt(FDDR_ODT),
    .tdqs_n()
);
```

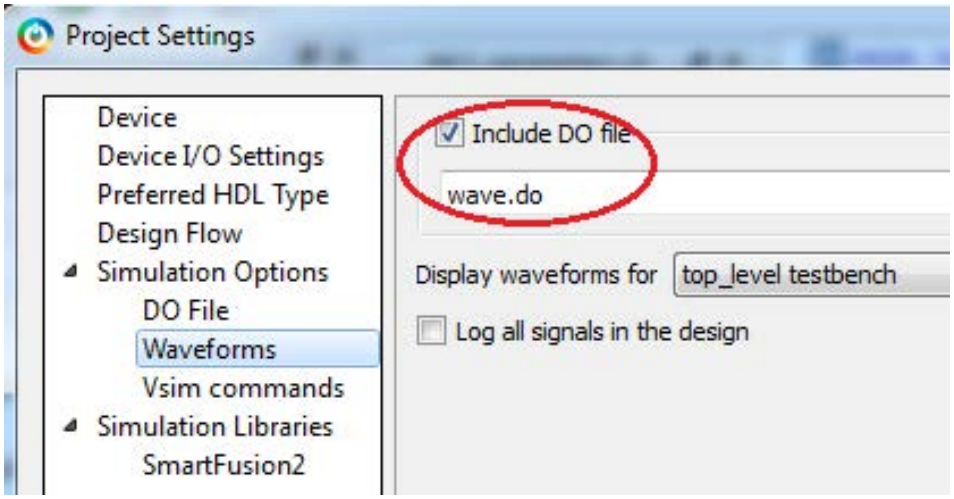
4. Ensure that the `ddr3.v` file is included at the top of the testbench file. The reference design uses four instances of DDR3 models with a data width of x8.
5. Set the testbench in which the DDR3 memory model is instantiated as an active stimulus. The following figure shows the settings under the stimulus hierarchy.

Figure 13 • **Stimulus Settings**



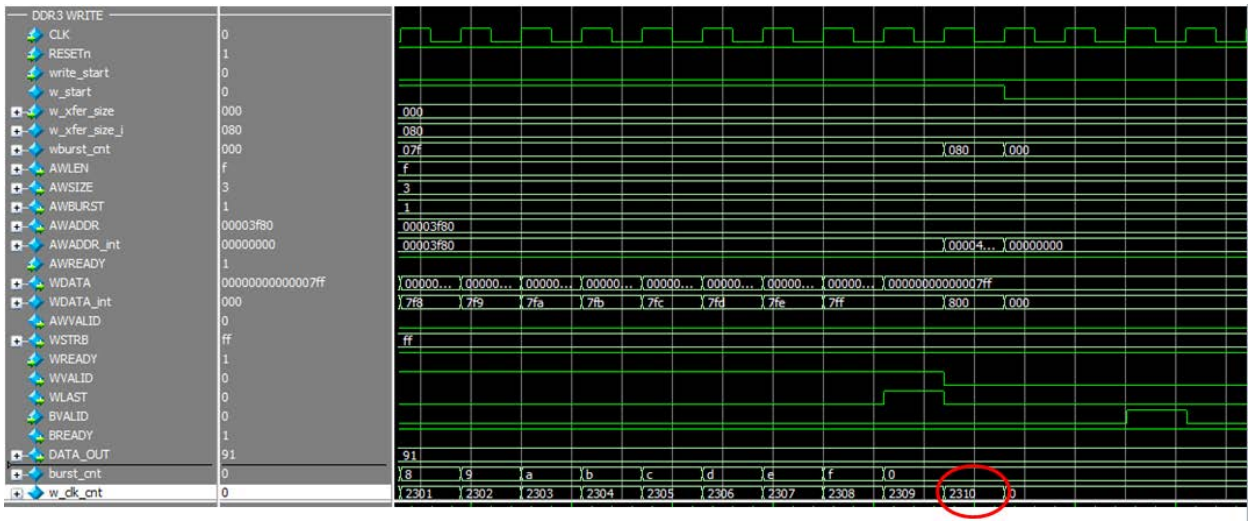
6. Click **Project** > **Project Settings** > **Simulation Options** > **Waveforms**. The following figure shows the waveforms settings on the right panel.

Figure 14 • **Waveform Settings**



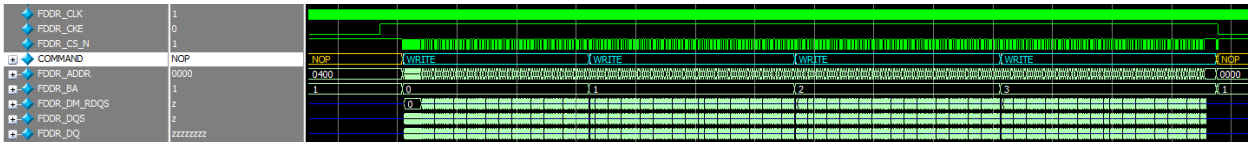
7. Select the **Include DO file** check box and enter **wave.do** in the box, as shown in the preceding figure.
   The following figure shows the AXI master signals, command from command decoder, and AXI clock count for a write operation, which calculates the write bandwidth.

Figure 15 • **AXI Master Signals During Write Operation**



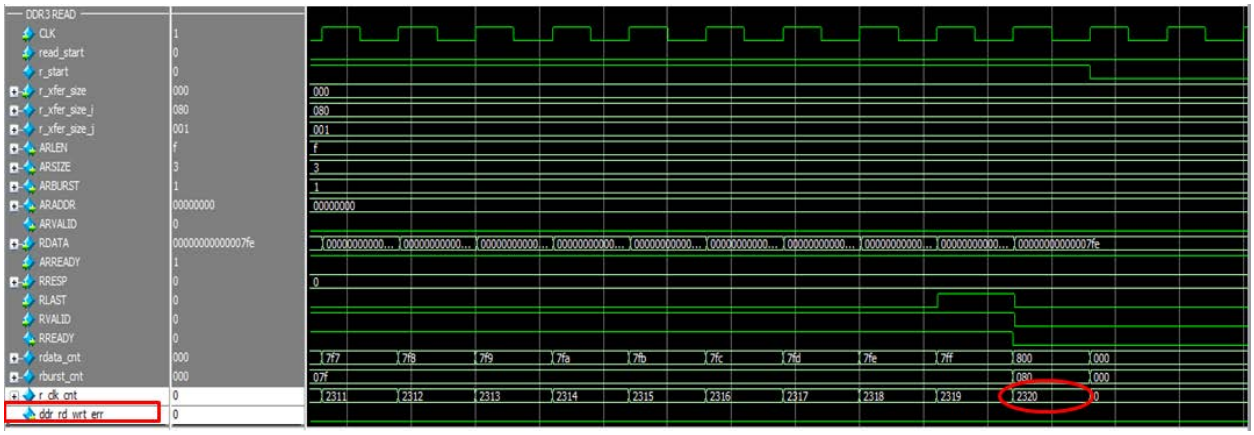
The following figure shows the FDDR signals. The AXI master writes 16 KB data into the DDR3 SDRAM. The data is written into Row 0 of banks (0-3).

Figure 16 • **FDDR Signals During Write Operation**



The following figure shows the AXI master signals, command from command decoder, and AXI clock count for a read operation, which calculates the read bandwidth.

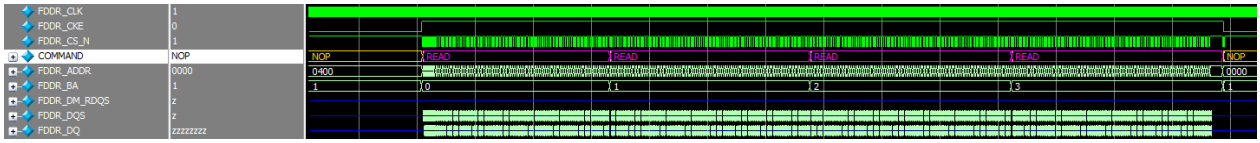
Figure 17 • **AXI Master Signals During Read Operation**



In the preceding figure, the read operation AXI clock count and the DDR_RD_WRT_ERR signal that is used to check for data mismatch between write and read operation are highlighted.

The following figure shows the FDDR signals. The AXI master reads 16 KB data from the DDR3 SDRAM. The data is read from Row 0 of banks (0-3).

Figure 18 • **FDDR Signals During Read Operation**



The following figures show the transcript window messages during write and read operations.

Figure 19 • **Transcript Window during Write Operation**

```
# testbench.DDR3_3.main: at time 636541220.0 ps INFO: Sync On Die Termination Rtt_NOM =          0 Ohm
# testbench.DDR3_0.data_task: at time 636542720.0 ps INFO: WRITE @ DQS= bank = 0 row = 0000 col = 00000338 data = 9c
# testbench.DDR3_1.data_task: at time 636542720.0 ps INFO: WRITE @ DQS= bank = 0 row = 0000 col = 00000338 data = 01
# testbench.DDR3_2.data_task: at time 636542720.0 ps INFO: WRITE @ DQS= bank = 0 row = 0000 col = 00000338 data = 00
# testbench.DDR3_3.data_task: at time 636542720.0 ps INFO: WRITE @ DQS= bank = 0 row = 0000 col = 00000338 data = 00
# testbench.DDR3_0.data_task: at time 636544220.0 ps INFO: WRITE @ DQS= bank = 0 row = 0000 col = 00000339 data = 00
# testbench.DDR3_1.data_task: at time 636544220.0 ps INFO: WRITE @ DQS= bank = 0 row = 0000 col = 00000339 data = 00
# testbench.DDR3_2.data_task: at time 636544220.0 ps INFO: WRITE @ DQS= bank = 0 row = 0000 col = 00000339 data = 00
# testbench.DDR3_3.data_task: at time 636544220.0 ps INFO: WRITE @ DQS= bank = 0 row = 0000 col = 00000339 data = 00
# testbench.DDR3_0.data_task: at time 636545720.0 ps INFO: WRITE @ DQS= bank = 0 row = 0000 col = 0000033a data = 9d
# testbench.DDR3_1.data_task: at time 636545720.0 ps INFO: WRITE @ DQS= bank = 0 row = 0000 col = 0000033a data = 01
# testbench.DDR3_2.data_task: at time 636545720.0 ps INFO: WRITE @ DQS= bank = 0 row = 0000 col = 0000033a data = 00
# testbench.DDR3_3.data_task: at time 636545720.0 ps INFO: WRITE @ DQS= bank = 0 row = 0000 col = 0000033a data = 00
# testbench.DDR3_0.data_task: at time 636547220.0 ps INFO: WRITE @ DQS= bank = 0 row = 0000 col = 0000033b data = 00
# testbench.DDR3_1.data_task: at time 636547220.0 ps INFO: WRITE @ DQS= bank = 0 row = 0000 col = 0000033b data = 00
# testbench.DDR3_2.data_task: at time 636547220.0 ps INFO: WRITE @ DQS= bank = 0 row = 0000 col = 0000033b data = 00
# testbench.DDR3_3.data_task: at time 636547220.0 ps INFO: WRITE @ DQS= bank = 0 row = 0000 col = 0000033b data = 00
# testbench.DDR3_0.data_task: at time 636548720.0 ps INFO: WRITE @ DQS= bank = 0 row = 0000 col = 0000033c data = 9e
# testbench.DDR3_1.data_task: at time 636548720.0 ps INFO: WRITE @ DQS= bank = 0 row = 0000 col = 0000033c data = 01
# testbench.DDR3_2.data_task: at time 636548720.0 ps INFO: WRITE @ DQS= bank = 0 row = 0000 col = 0000033c data = 00
# testbench.DDR3_3.data_task: at time 636548720.0 ps INFO: WRITE @ DQS= bank = 0 row = 0000 col = 0000033c data = 00
# testbench.DDR3_0.data_task: at time 636550220.0 ps INFO: WRITE @ DQS= bank = 0 row = 0000 col = 0000033d data = 00
# testbench.DDR3_1.data_task: at time 636550220.0 ps INFO: WRITE @ DQS= bank = 0 row = 0000 col = 0000033d data = 00
# testbench.DDR3_2.data_task: at time 636550220.0 ps INFO: WRITE @ DQS= bank = 0 row = 0000 col = 0000033d data = 00
# testbench.DDR3_3.data_task: at time 636550220.0 ps INFO: WRITE @ DQS= bank = 0 row = 0000 col = 0000033d data = 00
# testbench.DDR3_0.data_task: at time 636551720.0 ps INFO: WRITE @ DQS= bank = 0 row = 0000 col = 0000033e data = 9f
# testbench.DDR3_1.data_task: at time 636551720.0 ps INFO: WRITE @ DQS= bank = 0 row = 0000 col = 0000033e data = 01
# testbench.DDR3_2.data_task: at time 636551720.0 ps INFO: WRITE @ DQS= bank = 0 row = 0000 col = 0000033e data = 00
# testbench.DDR3_3.data_task: at time 636551720.0 ps INFO: WRITE @ DQS= bank = 0 row = 0000 col = 0000033e data = 00
# testbench.DDR3_0.data_task: at time 636553220.0 ps INFO: WRITE @ DQS= bank = 0 row = 0000 col = 0000033f data = 00
# testbench.DDR3_1.data_task: at time 636553220.0 ps INFO: WRITE @ DQS= bank = 0 row = 0000 col = 0000033f data = 00
# testbench.DDR3_2.data_task: at time 636553220.0 ps INFO: WRITE @ DQS= bank = 0 row = 0000 col = 0000033f data = 00
# testbench.DDR3_3.data_task: at time 636553220.0 ps INFO: WRITE @ DQS= bank = 0 row = 0000 col = 0000033f data = 00
# testbench.DDR3_0.cmd_task: at time 636562220.0 ps INFO: Write     bank 0 col 340, auto precharge 0
# testbench.DDR3_1.cmd_task: at time 636562220.0 ps INFO: Write     bank 0 col 340, auto precharge 0
# testbench.DDR3_2.cmd_task: at time 636562220.0 ps INFO: Write     bank 0 col 340, auto precharge 0
# testbench.DDR3_3.cmd_task: at time 636562220.0 ps INFO: Write     bank 0 col 340, auto precharge 0
# testbench.DDR3_0.main: at time 636568220.0 ps ERROR:  ODTH4 violation during ODT transition
# testbench.DDR3_1.main: at time 636568220.0 ps ERROR:  ODTH4 violation during ODT transition
# testbench.DDR3_2.main: at time 636568220.0 ps ERROR:  ODTH4 violation during ODT transition
# testbench.DDR3_3.main: at time 636568220.0 ps ERROR:  ODTH4 violation during ODT transition
# testbench.DDR3_0.main: at time 636574220.0 ps INFO: Sync On Die Termination Rtt_NOM =         40 Ohm
# testbench.DDR3_1.main: at time 636574220.0 ps INFO: Sync On Die Termination Rtt_NOM =         40 Ohm
# testbench.DDR3_2.main: at time 636574220.0 ps INFO: Sync On Die Termination Rtt_NOM =         40 Ohm
# testbench.DDR3_3.main: at time 636574220.0 ps INFO: Sync On Die Termination Rtt_NOM =         40 Ohm
# testbench.DDR3_0.main: at time 636577220.0 ps INFO: Sync On Die Termination Rtt_NOM =          0 Ohm
# testbench.DDR3_1.main: at time 636577220.0 ps INFO: Sync On Die Termination Rtt_NOM =          0 Ohm
# testbench.DDR3_2.main: at time 636577220.0 ps INFO: Sync On Die Termination Rtt_NOM =          0 Ohm
# testbench.DDR3_3.main: at time 636577220.0 ps INFO: Sync On Die Termination Rtt_NOM =          0 Ohm
# testbench.DDR3_0.data_task: at time 636578720.0 ps INFO: WRITE @ DQS= bank = 0 row = 0000 col = 00000340 data = a0
# testbench.DDR3_1.data_task: at time 636578720.0 ps INFO: WRITE @ DQS= bank = 0 row = 0000 col = 00000340 data = 01
# testbench.DDR3_2.data_task: at time 636578720.0 ps INFO: WRITE @ DQS= bank = 0 row = 0000 col = 00000340 data = 00
# testbench.DDR3_3.data_task: at time 636578720.0 ps INFO: WRITE @ DQS= bank = 0 row = 0000 col = 00000340 data = 00
# testbench.DDR3_0.data_task: at time 636580220.0 ps INFO: WRITE @ DQS= bank = 0 row = 0000 col = 00000341 data = 00
# testbench.DDR3_1.data_task: at time 636580220.0 ps INFO: WRITE @ DQS= bank = 0 row = 0000 col = 00000341 data = 00
# testbench.DDR3_2.data_task: at time 636580220.0 ps INFO: WRITE @ DQS= bank = 0 row = 0000 col = 00000341 data = 00
# testbench.DDR3_3.data_task: at time 636580220.0 ps INFO: WRITE @ DQS= bank = 0 row = 0000 col = 00000341 data = 00
# testbench.DDR3_0.data_task: at time 636581720.0 ps INFO: WRITE @ DQS= bank = 0 row = 0000 col = 00000342 data = a1
# testbench.DDR3_1.data_task: at time 636581720.0 ps INFO: WRITE @ DQS= bank = 0 row = 0000 col = 00000342 data = 01
```

Note: Memory vendors such as Micron, Samsung, and Hynix provide simulation models for specific memory devices, which can be downloaded. Ensure that the downloaded simulation model is JEDEC compliant.

2.7 Running the Design

The reference design is designed to run on the RTG4 Development Kit board. For more information about the RTG4 Development Kit board, refer to *http://www.microsemi.com/products/fpga-soc/design-resources/dev-kits/rtg4-development-kit*.

2.7.1 Board Jumper Settings

The following table lists the jumpers that must be connected on the RTG4 Development Kit board.

Table 4 • **RTG4 FPGA Development Kit Jumper Settings**

Jumper	Pin (From)	Pin (To)	Comments
J32, J27, J26, J23, J21, J19, J17, J11	1	2	Default
J16	2	3	Default
J33	1 3	2 4	Default

Note: Ensure that the power supply switch, SW6 is switched off while connecting the jumpers on the RTG4 Development Kit board.

2.7.2 Host PC to Board Connections

Connect the host PC to the J47 connector using the USB cable (mini USB to Type A).

2.7.3 USB Driver Installation

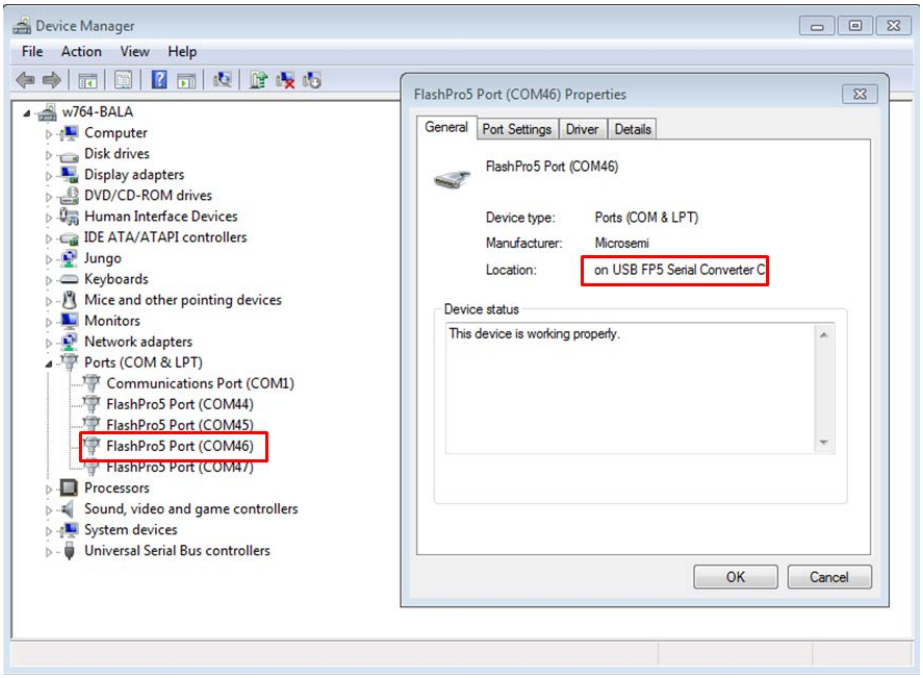
Install the FTDI D2XX driver for serial terminal communication through FTDI mini USB cable. The drivers and installation guide are available at:

www.microsemi.com/soc/documents/CDM_2.08.24_WHQL_Certified.zip.

Ensure that the USB to UART bridge drivers are automatically detected. This can be verified in the Device Manager of the host PC, as shown in the following figure. The FTDI USB to UART converter enumerates four COM ports. For USB 2.0, note down the USB serial converter C COM port number to use it in the demo utility.

The following figure shows the USB 2.0 serial port properties and how COM46 is connected to USB Serial Converter C. To find the correct COM port in USB 3.0, refer to Appendix 3: Finding Correct COM Port Number when using USB 3.0, page 30.

Figure 20 • **USB Serial 2.0 Port Properties**



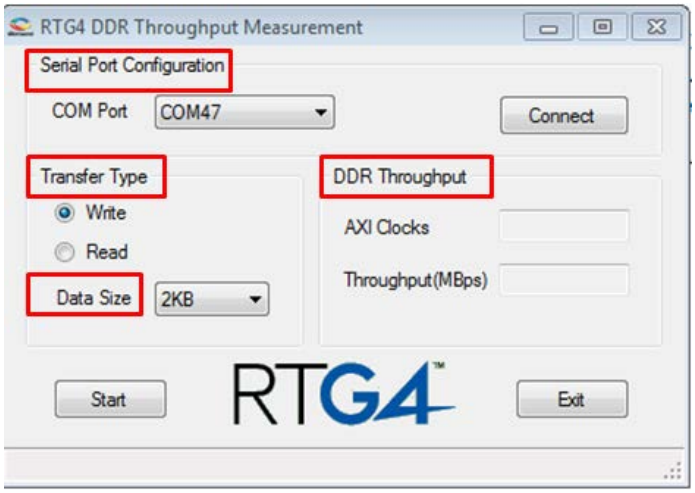
2.7.4 Programming the Device

Program the RTG4 Development Kit with the job file provided as part of the design files using FlashPro Express software, refer to Appendix 1: Programming the Device Using FlashPro Express, page 26.

2.7.4.1 Running the Hardware Demo

The RTG4 DDR bandwidth demo utility runs on the host PC to communicate with the RTG4 Development Kit. The UART protocol is used as the underlying communication protocol between the host PC and the RTG4 Development Kit. The following figure shows the initial screen of the RTG4_DDR_BW demo utility application.

Figure 21 • **RTG4 DDR Bandwidth Utility**



The RTG4_DDR_BW utility has the following sections:

- Serial port configuration: Displays the serial port. Baud rate is fixed at 115200
- Transfer type: Write or Read
- Data size: Option between 2 KB, 4 KB, 8 KB, 16 KB, and 32 KB burst size
- DDR throughput: Displays number of AXI clocks consumed and throughput in Mbps for the selected transfer type.

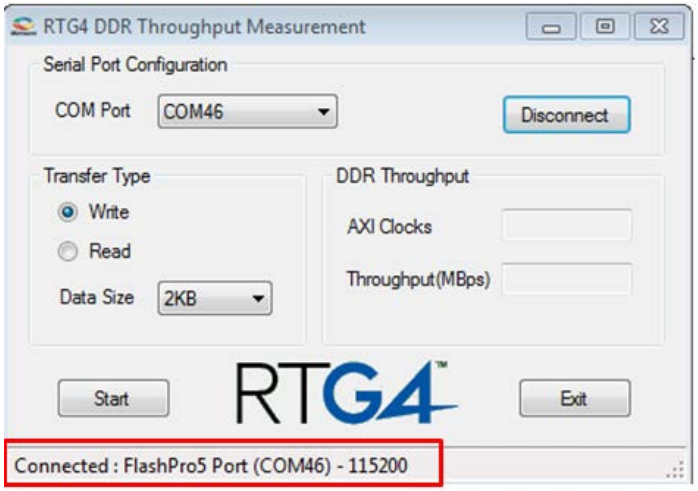
2.7.4.2 Steps to Run the Demo Utility

The following steps describe how to run the demo utility:

1. Launch the RTG4 DDR Bandwidth utility. The default location is: *<download_folder>\RTG4_DDRC_DF\Demo_Utility\RTG4_DDR_BW.exe*
2. Select the appropriate COM port from the drop-down menu. In this case, it is COM 46.
3. Click **Connect**. The connection status along with the COM port and Baud rate is shown at the bottom of the window.

The following figure shows the connection status of the utility.

Figure 22 • **RTG4 DDR Bandwidth Connection Status**



4. Select **DataSize** as 2 KB and perform write and read operations. The following figures show the respective results.

Figure 23 • **Throughput for 2 KB Write**

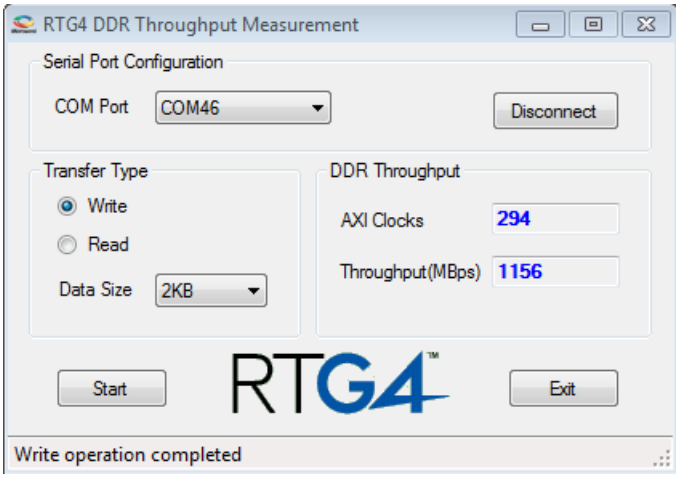
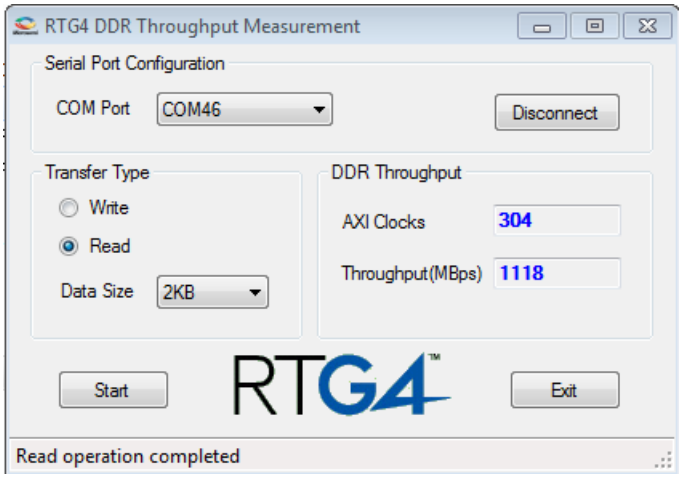
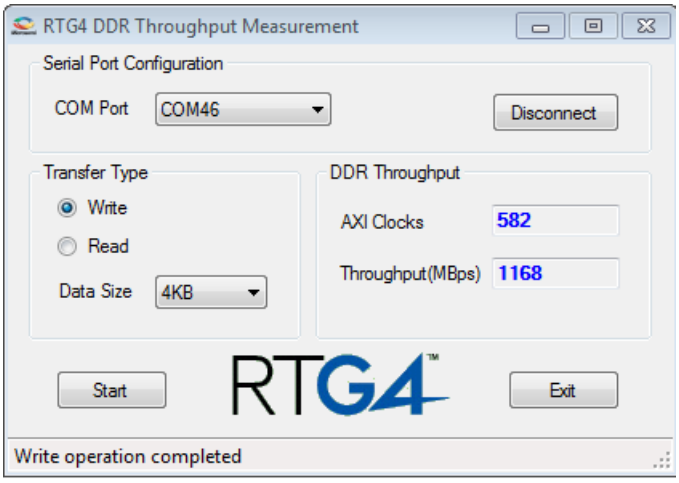


Figure 24 • **Throughput for 2 KB Read**



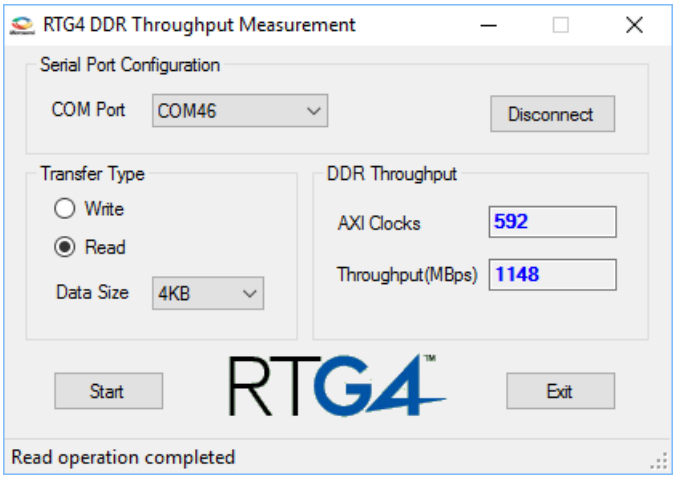
5. Select **DataSize** as 4 KB and perform write and read operations. The following figures show the respective results.

Figure 25 • **Throughput for 4 KB Write**



The following figure shows the throughput for 4 KB read.

Figure 26 • **Throughput for 4 KB Read**



6. Select **DataSize** as 8 KB and perform write and read operations. The following figures show the respective results.

Figure 27 • **Throughput for 8 KB Write**

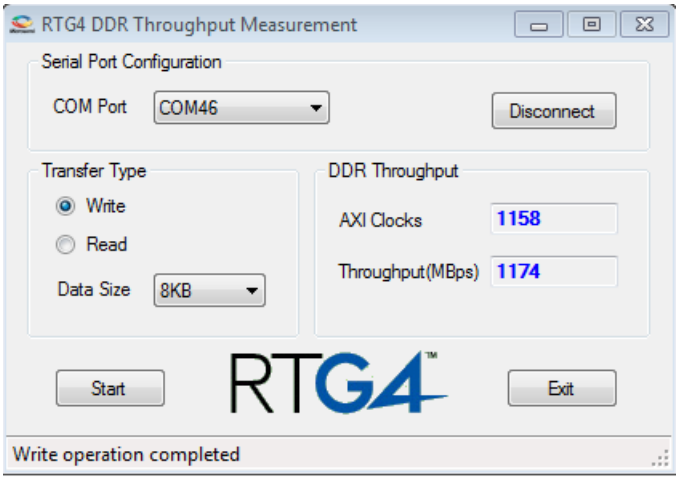
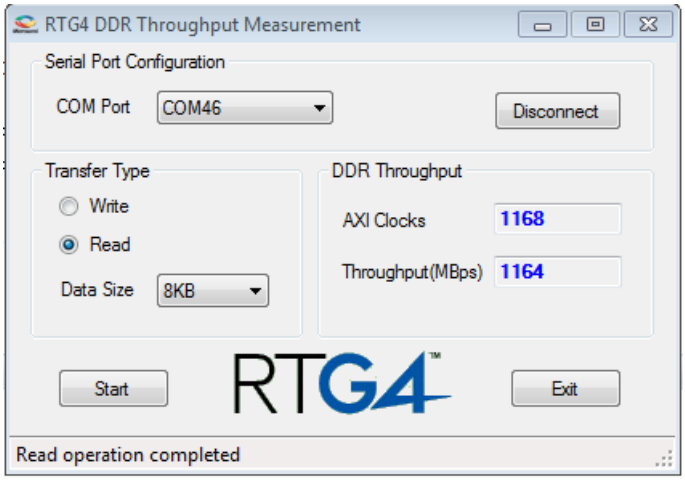
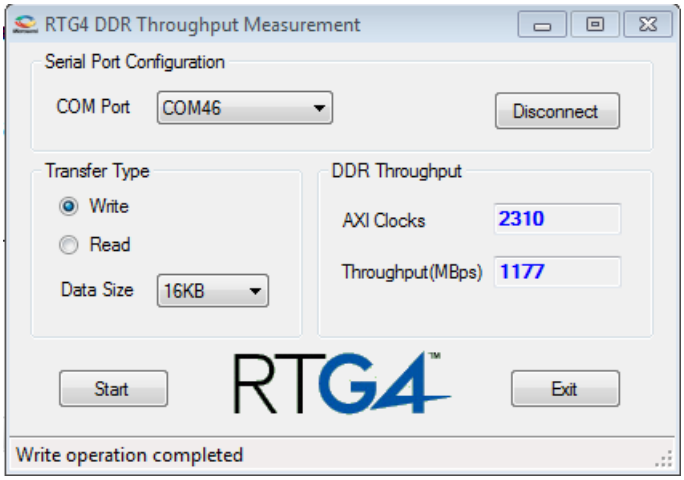


Figure 28 • **Throughput for 8 KB Read**



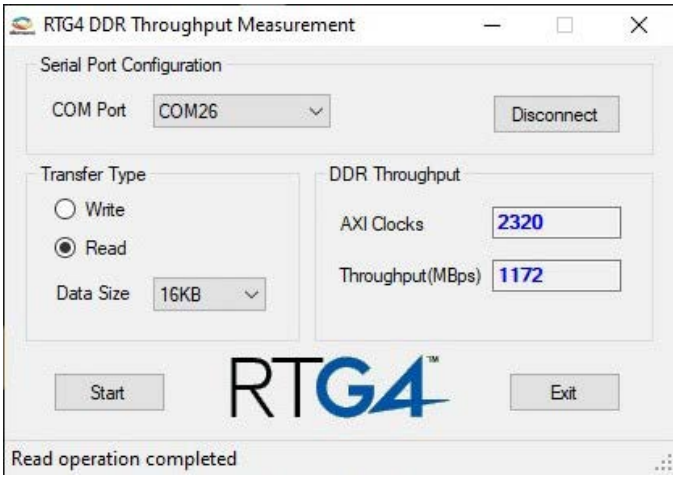
7. Select **Data Size** as 16 KB and perform write and read operations. The following figures show the respective results.

Figure 29 • **Throughput for 16 KB Write**



The following figure shows the throughput for 16 KB Read.

Figure 30 • **Throughput for 16 KB Read**



8. Select **DataSize** as 32 KB and perform write and read operations. The following figures show the respective results.

Figure 31 • **Throughput for 32 KB Write**

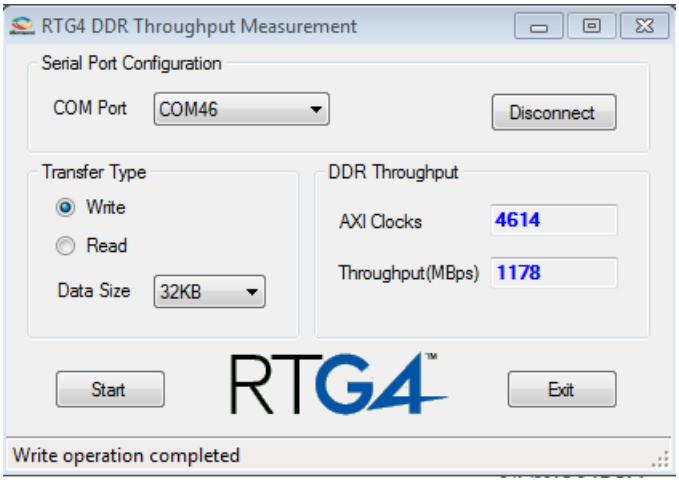
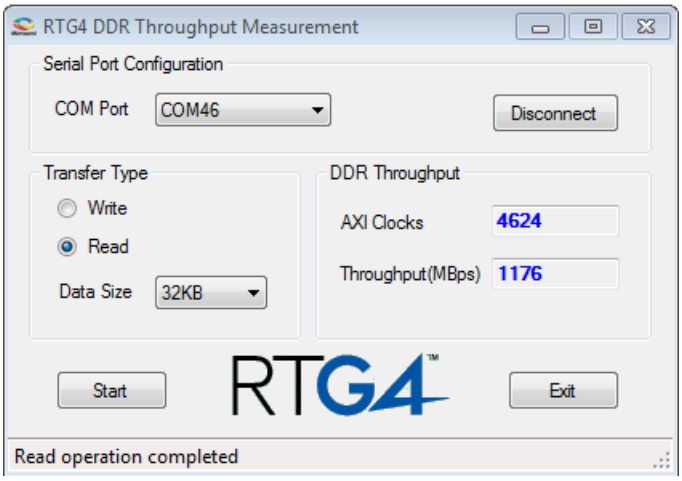


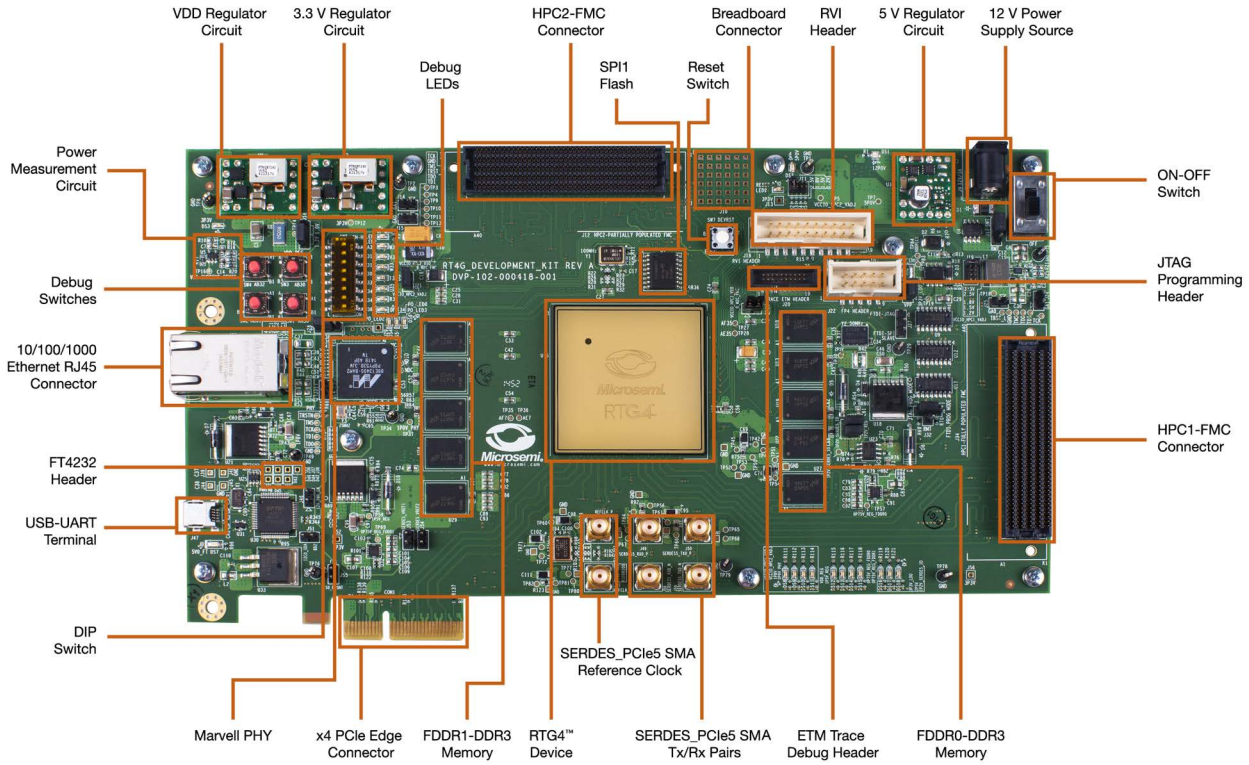
Figure 32 • **Throughput for 32 KB Read**



Note: While performing burst transactions for different transfer rates, perform the write operation first, and then read the data.

Observe the data mismatch error signal that is generated by the AXI master logic and given to the on-board active LOW debug LED (W34). The following figure shows the RTG4 Development Kit component placements on-board.

Figure 33 • **RTG4 Development Kit Board**



2.8 DDR3 SDRAM Bandwidth

The following table lists the total number of 16-beat bursts corresponding to the write or read size.

Table 5 • **Total Number of 16-Beat Bursts**

Write or Read Size	Total Number of 16-Beat Bursts
2 KB	16
4 KB	32
8 KB	64
16 KB	128
32 KB	256

Use the following equation to calculate the throughput:

Bandwidth (MB/s) = (16 ÷ [Total number of AXI clocks ÷ Total number of 16-beat bursts]) × 8 × AXI Clock (MHz)

2.8.1 Simulation Results

The following table lists the write and read bandwidth of the DDR3 SDRAM simulation. Data of incremental size (2 KB to 32 KB) is transferred from the fabric logic AXI master to the DDR3 SDRAM and the DDR3 SDRAM to the fabric logic AXI master. The throughput improvement percentage is shown for the baseline value.

In the reference design, the DDR3 SDRAM is set to a maximum supported burst length of 8, and the AXI interface is set to a maximum supported burst length of 16. All read and write transactions are performed on the same row to avoid precharge latency and improve overall throughput.

Table 6 • **DDR3 SDRAM Bandwidth - Simulation Results**

		Write		Read			
Optimization Technique	**Size**	**No of Cycles**	**Bandwidth (MB/s)**	**No of Cycles**	**Bandwidth (MB/s)**	**Average Write Throughput**	**Average Read Throughput**
Frequency of operation	2 KB	352	965	501	679	965 (Baseline)	682 (Baseline)
	4 KB	704	965	997	682		
	8 KB	1408	966	1992	683		
	16 KB	2816	966	3982	683		
	32 KB	5632	966	7962	683		
Frequency of operation AXI master without write response	2 KB	295	1152	501	679	1169 (Improvement of 21.1%)	682 (No Improvement)
	4 KB	583	1166	997	682		
	8 KB	1159	1173	1992	683		
	16 KB	2311	1177	3982	683		
	32 KB	4615	1179	7962	683		
Frequency of operation AXI master without write response and DDR configuration tuned	2 KB	294	1156	500	680	1172 (Improvement of 21.5%)	685 (Improvement of 0.43%)
	4 KB	582	1168	995	686		
	8 KB	1158	1179	1990	686		
	16 KB	2310	1177	3985	685		
	32 KB	4614	1178	7961	686		
Frequency of operation AXI master without write response DDR configuration tuned and read address queuing	2 KB	294	1161	304	1123	1175 (Improvement of 21.8%)	1161 (Improvement of 70.2%)
	4 KB	582	1173	592	1153		
	8 KB	1158	1179	1168	1169		
	16 KB	2310	1182	2320	1177		
	32 KB	4614	1178	4624	1181		

2.8.2 Board Results

The following table lists the write and read bandwidth of the DDR3 SDRAM on the RTG4 Development Kit board. Data of incremental size (2 KB to 32 KB) is transferred from the fabric logic AXI master to the DDR3 SDRAM and the DDR3 SDRAM to the fabric logic AXI master. The throughput improvement percentage is shown for the baseline value.

In the reference design, the DDR3 SDRAM is set to a maximum supported burst length of 8, and the AXI interface is set to a maximum supported burst length of 16. All read and write transactions are performed on the same row to avoid precharge latency and improve overall throughput.

Table 7 • **DDR3 SDRAM Bandwidth - Board Results**

		Write		Read			
Optimization technique	**Size**	**No of Cycles**	**Bandwidth (MB/s)**	**No of Cycles**	**Bandwidth (MB/s)**	**Average Write Throughput**	**Average Read Throughput**
Frequency of operation	2 KB	352	965	501	679	Avg: 965 (Baseline)	Avg: 682 (Baseline)
	4 KB	704	965	997	681		
	8 KB	1408	965	1992	682		
	16 KB	2816	965	3982	683		
	32 KB	5632	965	7960	683		
Frequency of operation and AXI master without write response	2 KB	295	1152	501	679	Avg: 1169 (Improvement of 21.1%)	Avg: 682 (No Improvement)
	4 KB	583	1166	997	681		
	8 KB	1159	1173	1992	682		
	16 KB	2311	1176	3991	683		
	32 KB	4615	1178	7975	683		
Frequency of operation and AXI master without write response and DDR configuration tuned	2 KB	294	1156	500	680	Avg: 1172 (Improvement of 0.25%)	Avg:685 (Improvement of 0.43%)
	4 KB	582	1168	996	686		
	8 KB	1158	1179	1990	686		
	16 KB	2310	1177	3985	685		
	32 KB	4614	1178	7959	686		
Frequency of operation and AXI master without write response and DDR configuration tuned and read address queuing	2 KB	294	1161	304	1123	Avg: 1175 (Improvement of 0.25%)	Avg: 1161 (Improvement of 69.5%)
	4 KB	582	1173	592	1153		
	8 KB	1158	1179	1168	1169		
	16 KB	2310	1182	2320	1177		
	32 KB	4614	1178	4622	1183		

2.9 Conclusion

This application note describes DDR3 SDRAM bandwidth optimization techniques using a reference design on the RTG4 Development Kit board. It also shows the DDR3 SDRAM simulation flow using the Micron DDR3 SDRAM model. Through optimization, 88% of the theoretical throughput value for both write and read is achieved.

3 Appendix 1: Programming the Device Using FlashPro Express

This section describes how to program the RTG4 device with the programming job file using FlashPro Express.

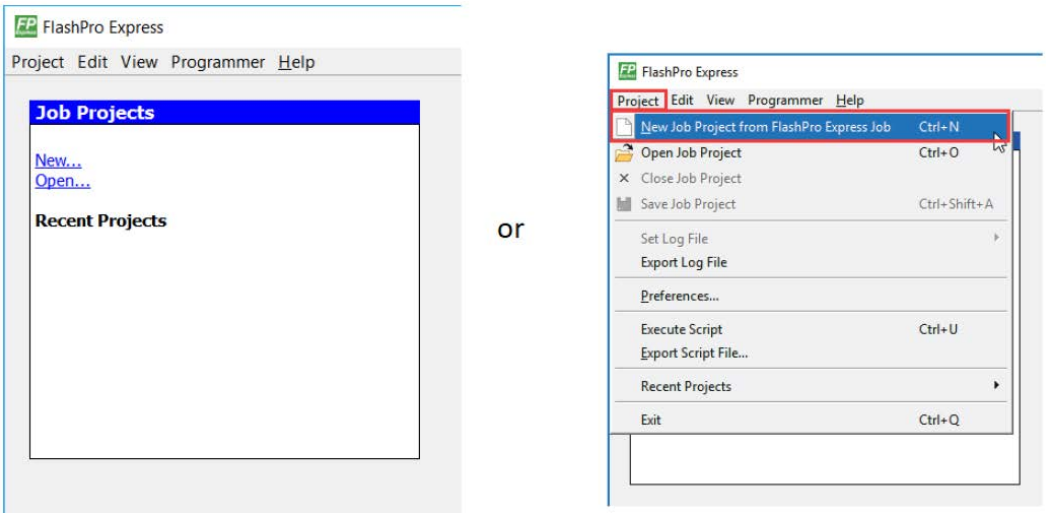
To program the device, perform the following steps:

1. Ensure that the jumper settings on the board are the same as those listed in *Table 3 of UG0617: RTG4 Development Kit User Guide*
2. Optionally, jumper **J32** can be set to connect pins 2-3 when using an external FlashPro4, FlashPro5, or FlashPro6 programmer instead of the default jumper setting to use the embedded FlashPro5.

Note: The power supply switch, **SW6** must be switched **OFF** while making the jumper connections.

3. Connect the power supply cable to the **J9** connector on the board.
4. Power **ON** the power supply switch **SW6**.
5. If using the embedded FlashPro5, connect the USB cable to connector **J47** and the host PC. Alternatively, if using an external programmer, connect the ribbon cable to the JTAG header **J22** and connect the programmer to the host PC.
6. On the host PC, launch the **FlashPro Express** software.
7. Click **New** or select **New Job Project from FlashPro Express Job** from **Project** menu to create a new job project, as shown in the following figure.

Figure 34 • **FlashPro Express Job Project**



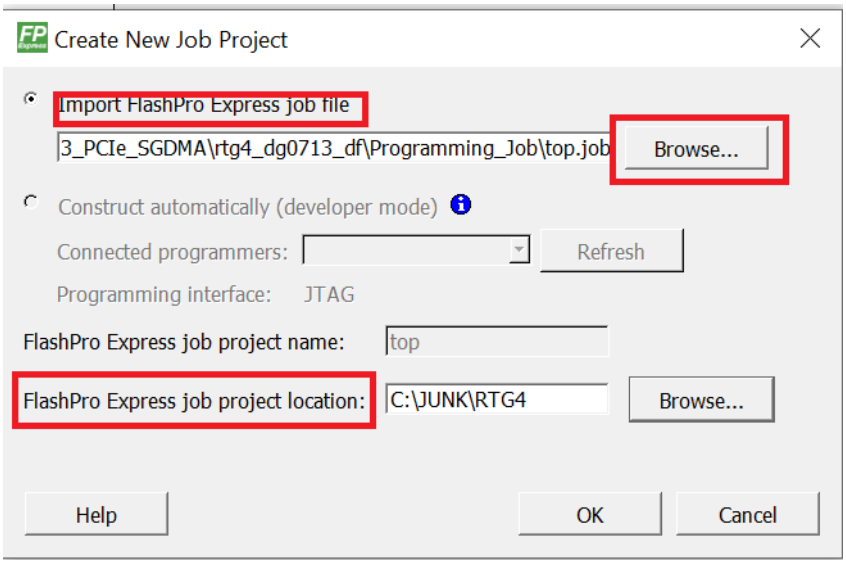
8. Enter the following in the **New Job Project from FlashPro Express Job** dialog box:

- **Programming job file:** Click **Browse**, and navigate to the location where the .job file is located and select the file. The default location is:

  `<download_folder>\rtg4_ac446_df\Programming_Job`

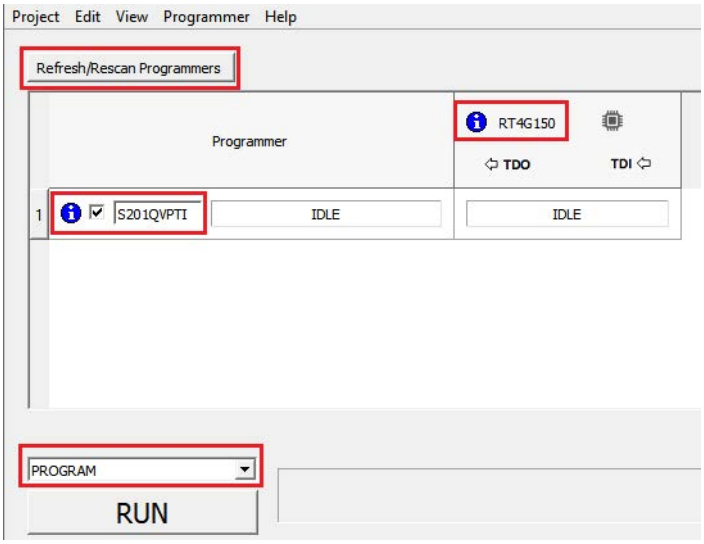
- **FlashPro Express job project location:** Click **Browse** and navigate to the desired FlashPro Express project location.

Figure 35 • **New Job Project from FlashPro Express Job**



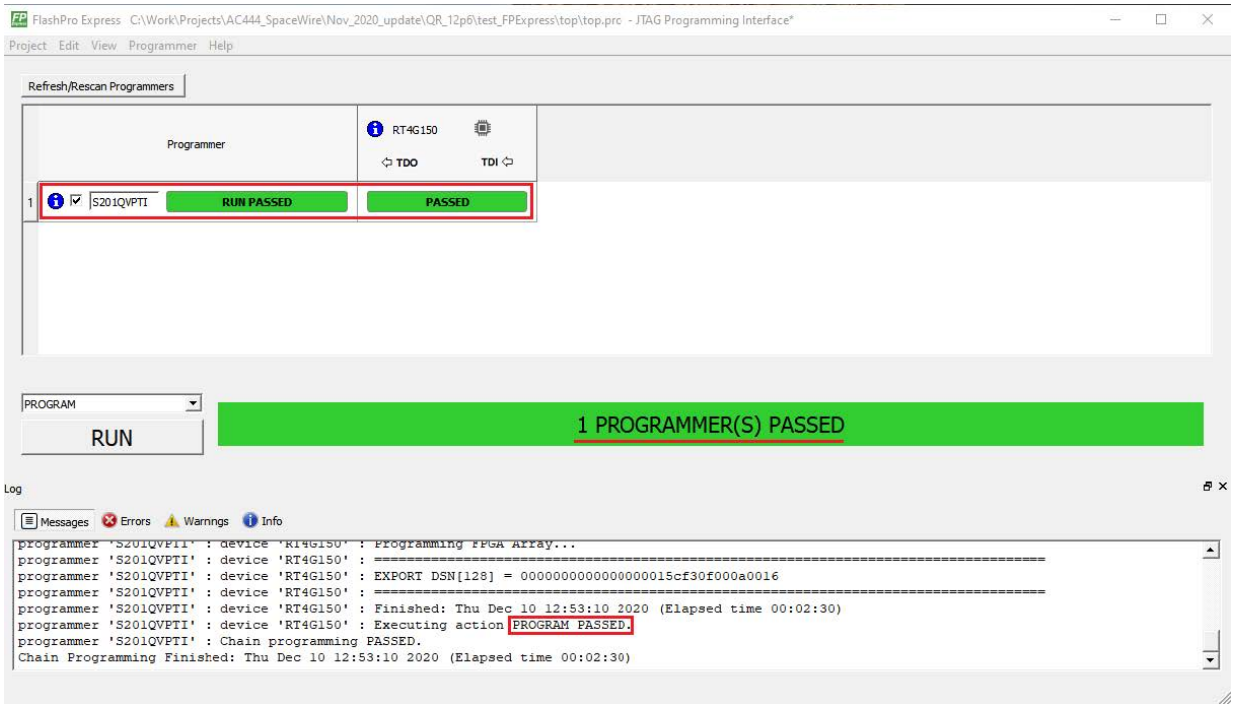
9. Click **OK**. The required programming file is selected and ready to be programmed in the device.
10. The FlashPro Express window appears as shown in the following figure. Confirm that a programmer number appears in the Programmer field. If it does not, confirm the board connections and click **Refresh/Rescan** Programmers.

Figure 36 • **Programming the Device**



11. Click **RUN**. When the device is programmed successfully, a **RUN PASSED** status is displayed as shown in the following figure.

Figure 37 • **FlashPro Express—RUN PASSED**



12. Close **FlashPro Express** or click **Exit** in the Project tab.

4 Appendix 2: Running the TCL Script

TCL scripts are provided in the design files folder under directory TCL_Scripts. If required, the design flow can be reproduced from Design Implementation till generation of job file.

To run the TCL, follow the steps below:

1. Launch the Libero software
2. Select **Project > Execute Script....**
3. Click Browse and select `script.tcl` from the downloaded TCL_Scripts directory.
4. Click **Run**.

After successful execution of TCL script, Libero project is created within TCL_Scripts directory.

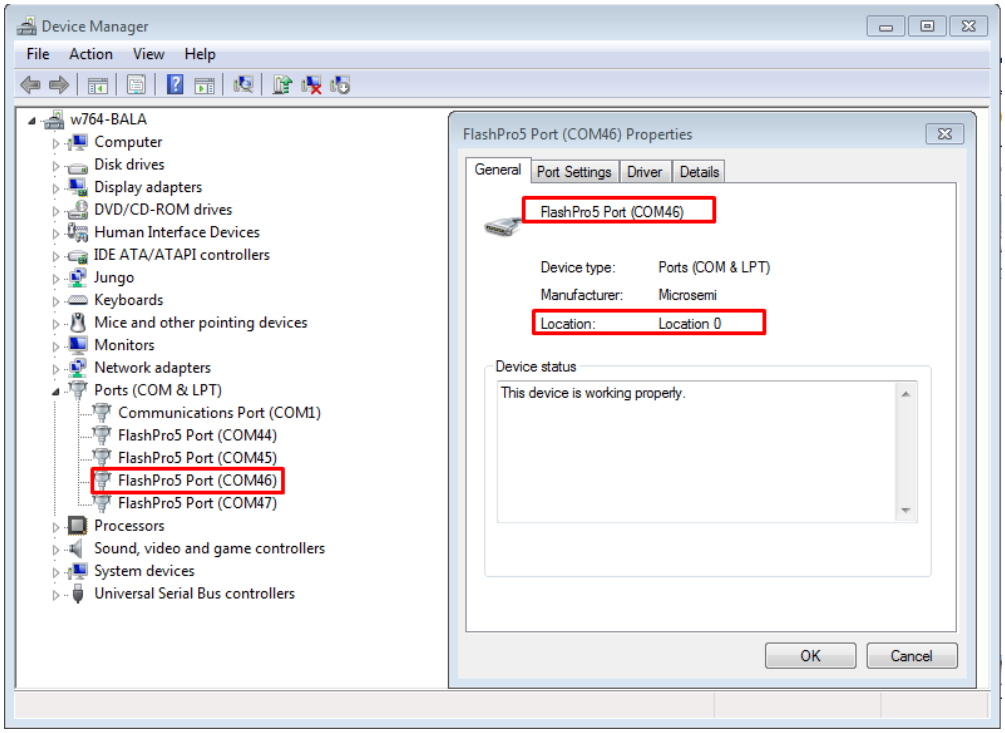
For more information about TCL scripts, refer to **rtg4_ac446_df/TCL_Scripts/readme.txt.**

Refer to *Libero® SoC TCL Command Reference Guide* for more details on TCL commands. Contact Technical Support for any queries encountered when running the TCL script.

5 Appendix 3: Finding Correct COM Port Number when using USB 3.0

FTDI USB to UART converter enumerates the four COM ports. In USB 3.0, the four available COM ports are in Location 0. The following figure shows the USB 3.0 serial port properties.

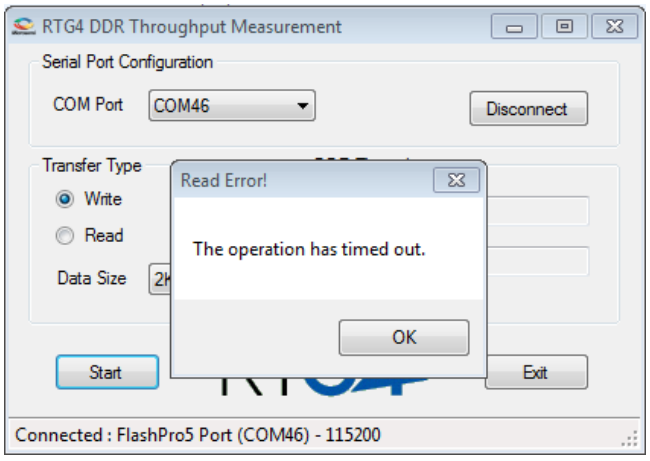
Figure 1 • **USB 3.0 Serial Port Properties**



To find the correct COM port:

1. Program the RTG4 Development Kit board with the provided programming file.
2. Select a COM port from the drop-down list, and click **Start**. If the wrong COM port is selected, the demo utility displays a read error. The following figure shows the read error message.

Figure 2 • **Read Error**



3. Repeat step 2 until the correct COM port is connected.