IGLOO2 - Optimizing DDR Controller for Improved Efficiency - Libero SoC v11.6

Table of Contents

Purpose

This application note describes the techniques for improving the efficiency of double data rate (DDR) controller using an example design for the IGLOO®2 Evaluation Kit board. It also provides details about implementing the DDR synchronous dynamic random access memory (SDRAM) simulation flow using the Micron® low power DDR (LPDDR) SDRAM model and Microsemi® LPDDR SDRAM verification IP (VIP) model.

Introduction

The IGLOO2 device has two high-speed hardened application-specific integrated circuit (ASIC) memory controllers such as memory subsystem DDR (MDDR) and fabric DDR (FDDR) for interfacing with the DDR2, DDR3, and LPDDR1 SDRAM memories. The MDDR and FDDR subsystems are used to access high-speed DDR memories for high-speed data transfer and code execution.

The DDR memory connected to the MDDR subsystem can be accessed by the high-performance memory subsystem (HPMS) masters and the master logic implemented in the FPGA fabric (FPGA fabric master), whereas the DDR memory connected to the FDDR subsystem can be accessed only by a field programmable gate array (FPGA) fabric master.

The FPGA fabric masters communicate with the MDDR and FDDR subsystems through the AXI or AHB interfaces. Figure 1 shows the MDDR data path for advanced extensible interface (AXI)/advanced high-performance bus (AHB) interfaces.

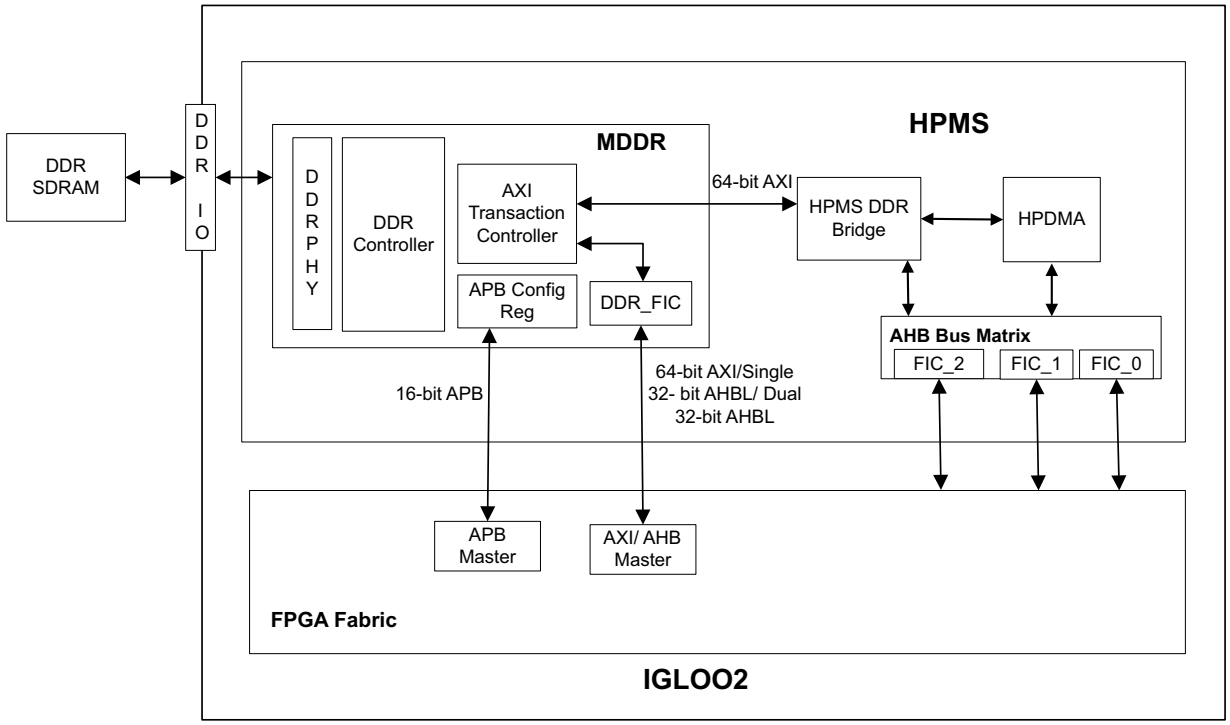

***Figure 1* • MDDR Data Path for AXI/AHB Interfaces**

The AXI interface is typically used for burst transfers that provide an efficient access path and high throughput. Though the throughput is dependent on many system level parameters, it can be improved by applying specific optimization techniques. This application note describes a few DDR SDRAM controller optimization techniques with an example design for IGLOO2 Evaluation Kit board. Refer to the *UG0446: SmartFusion and IGLOO2 FPGA High Speed DDR Interfaces User Guide,* for more information on MDDR and FDDR subsystems.

The sample design consists an AXI master, LSRAM, counters for throughput measurement, and CoreUART interface logic. During the write operation, AXI master reads the LSRAM and writes to the LPDDR memory and measures the throughput. During the read operation, AXI master reads the LPDDR memory and writes to LSRAM and measures the throughput. The throughput values are displayed on the host PC using the CoreUART interface.

There are two types of memory simulation models that can be used:

- **Microsemi provided Verification Intellectual Property (VIP):** The Libero® System-on-Chip (SoC) includes a JEDEC compliant VIP model. The VIP model is attached to the pin side of the MDDR/FDDR subsystem and simulates the functionality of a DDR memory device. It can be configured for DDR2, DDR3, and LPDDR SDRAM memories, and is intended to complement vendor models or to act as a substitute in case a vendor model is not available.
- **Vendor-specific memory model:** Memory vendors such as Micron®, Samsung, and Hynix provide downloadable simulation models for specific memory devices. The downloaded simulation model must be JEDEC compliant.

This application note also describes the DDR SDRAM simulation flow using the Micron LPDDR SDRAM model and Microsemi LPDDR SDRAM VIP model.

References

The following are the references:

- *UG0446: SmartFusion and IGLOO2 FPGA High Speed DDR Interfaces User Guide*
- *AC409: Connecting User Logic to AXI Interfaces of High-Performance Communication Blocks in the SmartFusion2 Devices*
- *AC333: Connecting User Logic to the SmartFusion Microcontroller Subsystem*
- *DDR Controller and Serial High Speed Controller Initialization Methodology*
- *UG0478: IGLOO2 FPGA Evaluation Kit User Guide*
- *IGLOO2 System Builder User Guide*

Design Requirements

Table 1 lists the design requirements.

Table 1 • **Design Requirements**

Hardware Requirements	**Description**
IGLOO2 Evaluation Kit. Refer the *UG0478: IGLOO2 FPGA Evaluation Kit User Guide* for more information.	Rev D or later
Desktop or Laptop	Any 64-bit Windows Operating System
Software Requirements	
Libero SoC	v11.6
Microsoft .NET Framework 4 Client Profile	

Optimization Techniques

This section describes the following optimization techniques:

- Frequency of Operation
- Burst Length
- AXI Master Without Write Response State
- Read Address Queuing
- Series of Writes or Reads
- DDR Configuration Tuning

Frequency of Operation

The MDDR and FDDR subsystems support clock management dividers directly inside the embedded block. The user can select the divider ratios from the Clock Configurator for DDR clocks (MDDR_CLK/FDDR_CLK) and DDR_FIC clock. The best overall throughput ratio is 2:1, that is, half the DDR clock frequency. Many other ratios are possible to provide flexibility to the FPGA design. To show the optimal data throughput, this application note shows all examples using the 2:1 ratio. The design example uses 64-bit AXI as a FPGA fabric interface and is configured to use 166 MHz as DDR clock frequency[1] and 83 MHz as AXI clock.

1. *IGLOO2 MDDR subsystem supports maximum of 200 MHz as DDR clock frequency for LPDDR1 memory type. But LPDDR memory on IGLOO2 Evaluation Kit board supports 166 MHz only.*

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 subsystem supports up to 16-beat burst reads and writes. The AXI beat burst length (write and read) and burst length of DRAM affect the optimal performance, but setting the maximum supported burst length for DDR SDRAM and AXI interface achieves optimal performance. The design example uses a DDR SDRAM burst length of 16 and an AXI write and read beat burst length of 16.

AXI Master Without Write Response State

When AXI master sends the last data (D (A15)), the WLAST signal goes HIGH indicating that it is the last transfer in the first write burst. When AXI slave in DDR subsystem accepts all the data items, it drives a write response (BVALID) back to the master to indicate that the write transaction is complete. By AXI protocol, AXI master waits for the write response before initiating the next write transaction. However, the time spent waiting for the write response reduces the overall throughput as the clock cycles are not used. AXI master can then send the second burst write address (B) without waiting for the write response of the first burst, which improves the write throughput.

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 fully accepted. The AXI protocol has a defined methodology on handling the termination of write burst transaction; this must be followed if the write response channel returns an incorrect value.
Figure 2 shows the write transaction timing diagram without the write response state.

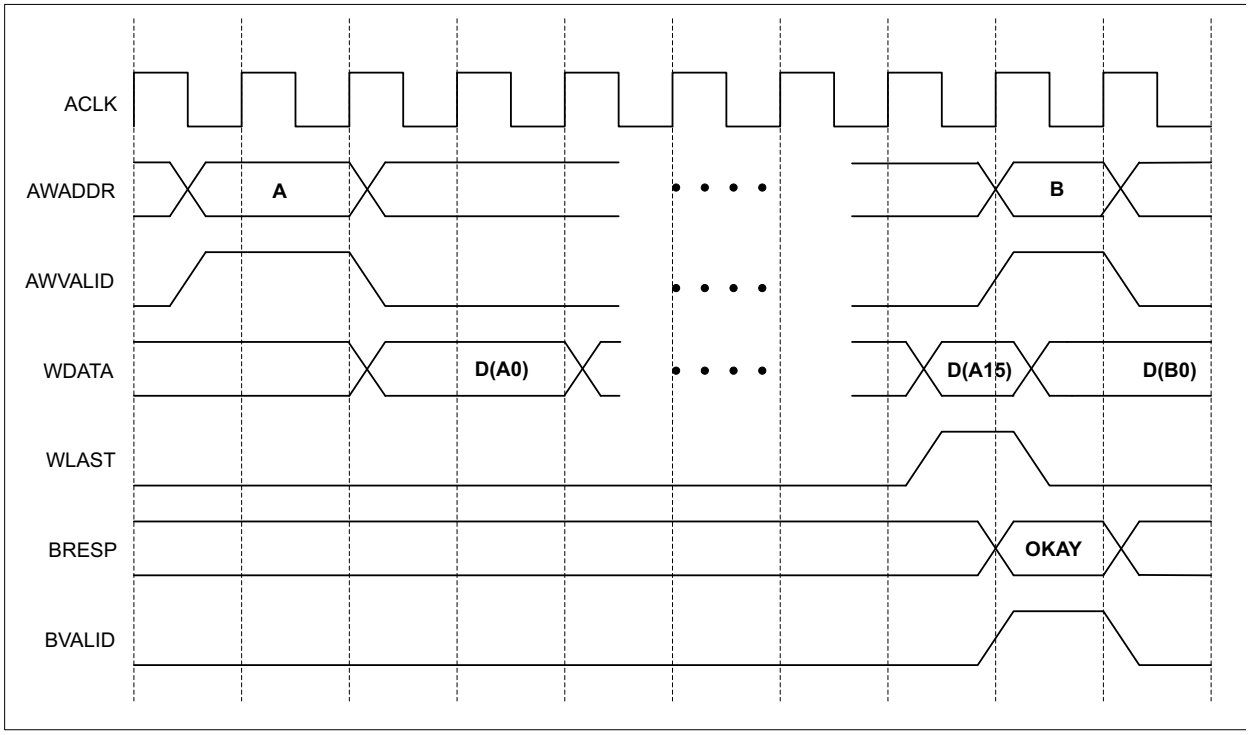

***Figure 2* • Write Transaction Timing Diagram Without Write Response State**

This technique is implemented in the example design. Comment or uncomment the following line of code in the AXI master interface (AXI_IF.v) to validate this technique.

```
define WITHOUT_WRITE_RESPONSE /* Comment this line to define With Write Response state */
```

Read Address Queuing

The MDDR and FDDR subsystems support upto four outstanding read transactions. In 2:1 clock ratio, the MDDR controller starts the burst read transaction before the command FIFO full, which allows AXI master to send 5 burst read address. Figure 3 shows the burst read address queuing timing diagram.

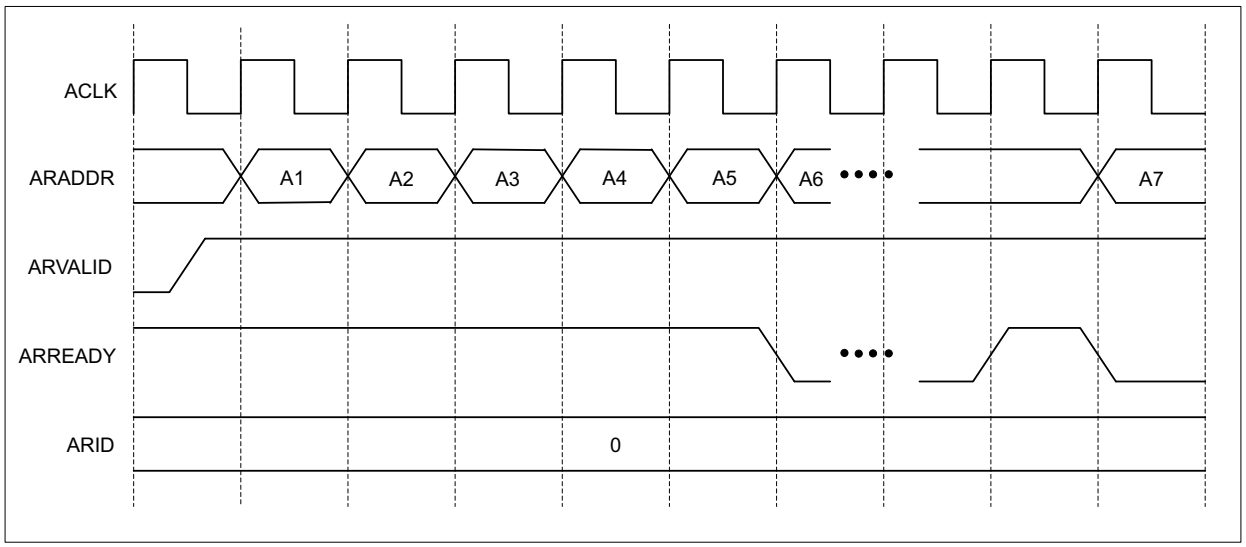

***Figure 3* • Read Transaction Timing Diagram with Burst Read Address Queuing**

AXI master increments the burst read address as long as 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 6 on page 31 and Table 7 on page 32 show this significant improvement. Read address queuing does not reduce the initial latency associated with a DDR memory read access. By issuing multiple reads in sequence the initial latency is only accounted for the first read. After the first read data is returned the remainder of the requested data is returned in sequence without a large read access penalty associated with the first read.

This technique is implemented in the example design. Comment or uncomment the following line of code in AXI master interface (AXI_IF.v) to validate this technique.

```
define READ_ADDRESS_QUEUING /* Comment this line to define Without Read Address Queuing */
```

Series of Writes or Reads

The MDDR and FDDR subsystems' performance depends on the method of data transfer between the DDR SDRAM and 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 MDDR and FDDR subsystems' performance increases while performing a series of reads or writes from the same bank and row. Figure 4 shows the AXI to LPDDR address mapping for the LPDDR SDRAM on the IGLOO2 Evaluation Kit board.

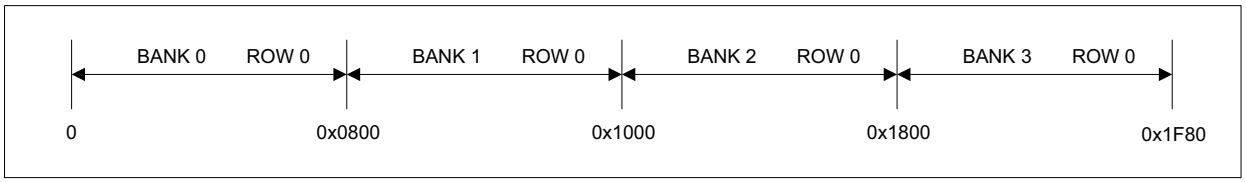

Figure 4 • **AXI to LPDDR the Address Mapping**

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 0x2000. If a new row is accessed every time, it must be pre-charged first. This means that additional time is needed before a row can be accessed and this reduces the overall throughput. Understanding the internal memory layout of the DDR and how it maps to the AXI address enables the accesses to minimize the row changes and increases the overall throughput.

DDR Configuration Tuning

The DDR SDRAM datasheet provides the timings parameters required for the proper operation in terms of time units. These timings must match with the configuration registers in the MDDR/FDDR controller. The timing parameters are required as number of DDR clock cycles and these are entered in the DDR configurator GUI. The selection of minimum write or read delay values can result in optimal performance. Implementing this approach requires extensive memory testing to ensure that the memory transfers are stable. The IGLOO2 Evaluation Kit LPDDR is supplied with a default configuration file to setup the MDDR controller, which is available on its documentation web page.

Table 2 lists the tuned parameters for better performance than the values in the default configuration file.

Table 2 • **Tuned DDR Timing Parameters**

Parameters	Default Values	Tuned Values
MRD	4	2
RAS min	8	7
RAS max	8192	11264
RCD	6	3
RP	7	3
REFI	3104	1280
RC	3	10
XP	3	1
CKE	3	1
RFC	79	25

Implementation on IGLOO2 Device

The optimization techniques mentioned in the preceding section are implemented and validated using the IGLOO2 Evaluation Kit board. This section describes the following:

- Design Description
- Hardware Implementation
- Running the Design

Design Description

The design consists HPMS, DDR initialization subsystem, AXI master (AXI_IF), Command decoder (CMD_Decoder), and a COM interface (COM_Interface) block. Figure 5 shows the block diagram of the design.

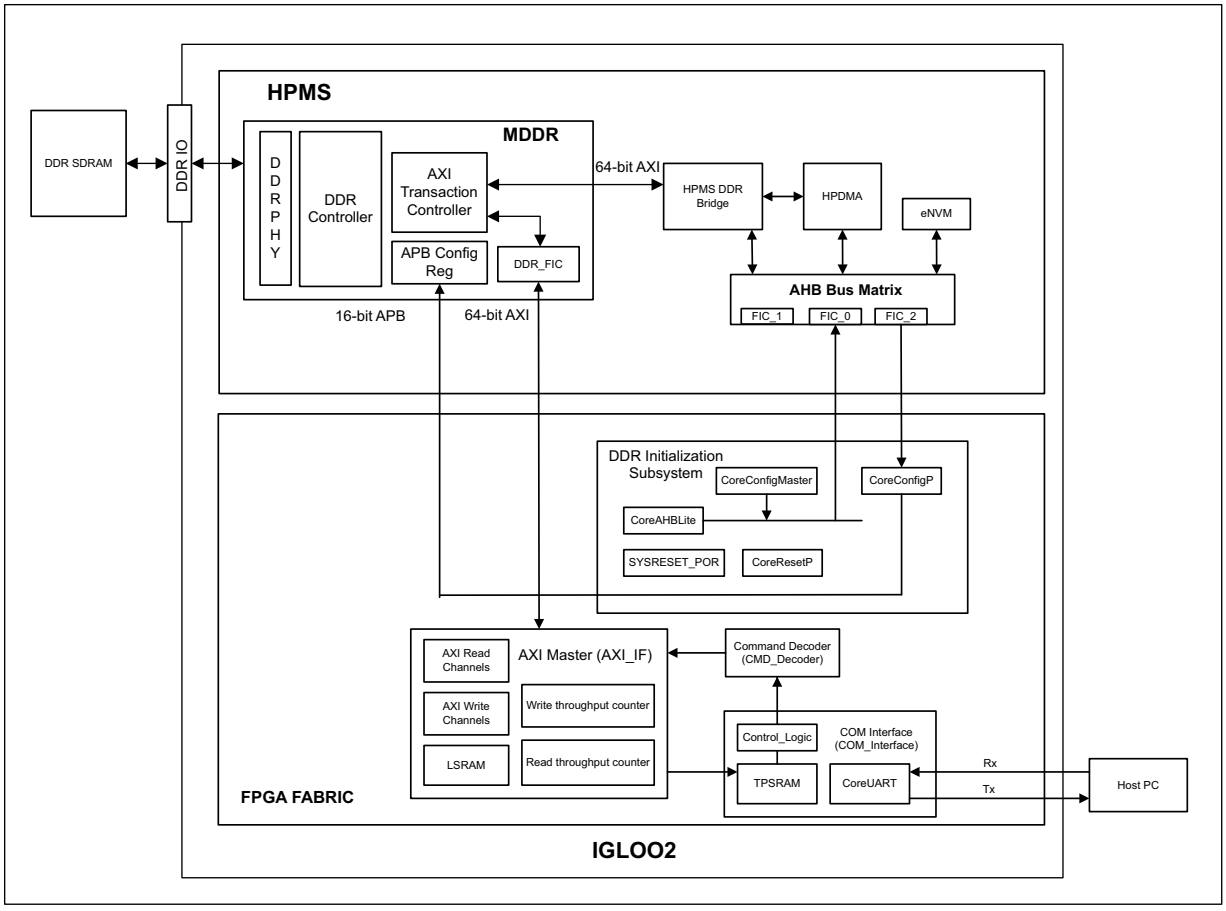

***Figure 5* • Top-Level Block Diagram of the Design**

MDDR in the HPMS is configured to use the LPDDR interface and route the AXI interface to the FPGA fabric. The DDR initialization subsystem consists CoreConfigMaster and CoreConfigP IPs that initialize the MDDR controller. The initialization process consists the following actions:

- CoreConfigMaster (AHBL Master) accesses the DDR configuration data stored in eNVM through FIC_0.
- The configuration data is sent to CoreConfigP through the FIC_2 master port.
- CoreConfigP sends the configuration data to advanced peripheral bus (APB) of the MDDR subsystem.

The command decoder receives the AXI transaction control from the COM interface block and generates write, read, write size, and read size signals. Figure 6 shows the command decoding.

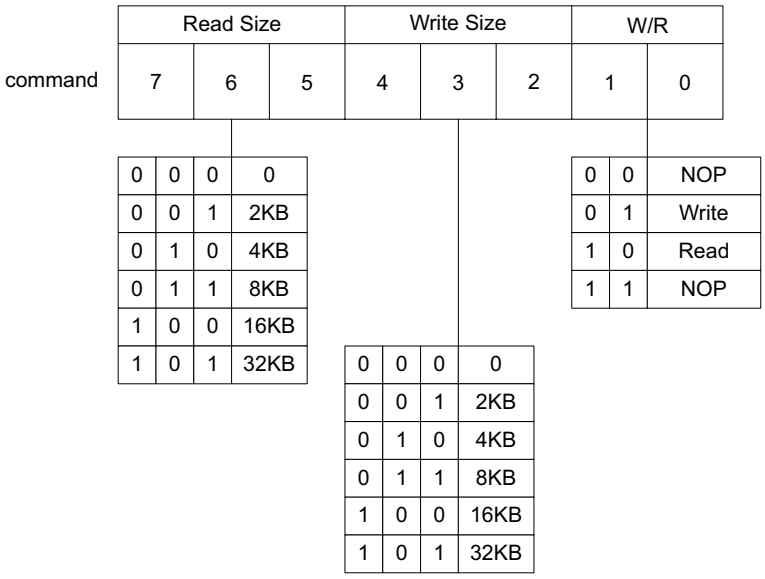

***Figure 6* • Command Decoding**

AXI master block consists AXI read channel, AXI write channel, write throughput counter, read throughput counter, and 512x64 LSRAM. It performs the write or read operation[2] based on the input signals from the command decoder. During the write operation, AXI master reads the LSRAM and writes into the LPDDR memory, and then measures the write throughput. During the read operation, AXI master reads the LPDDR memory and writes into LSRAM, and then measures the read throughput. The write throughput counter counts the AXI clocks between AWVALID of first data and WLAST of last data. Similarly, the read throughput counter counts the AXI clocks between ARVALID of first data and RLAST of last data.

After triggering the write or read operation, AXI master performs the write or read operation eight times to get the average throughput and to ACTIVATE all banks. 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) starts from 0x00000000, and is incremented by 128.

After each write or read operation, AXI master sends the throughput count value and an address starting from 0x0 to the COM interface block. Then, the COM interface block writes the throughput values into TPSRAM. The control logic in the COM interface block reads the values and sends to the host PC using the CoreUART interface.

For information on creating a custom AXI interface on user logic, refer to the *AC409: Connecting User Logic to AXI Interfaces of High-Performance Communication Blocks in the SmartFusion2 Devices Application Note*.

2. *The write or read operation depends on the size of the write or read data. For example, if the write size is selected as 2KB, then one AXI write operation equals to 16x16-beat burst (16x16x64).*

Hardware Implementation

The hardware implementation involves:

- Configuring the System Builder
- Connecting with custom logic (AXI master, Command decoder, and COM interface).

Figure 7 shows the top-level SmartDesign of the example design.

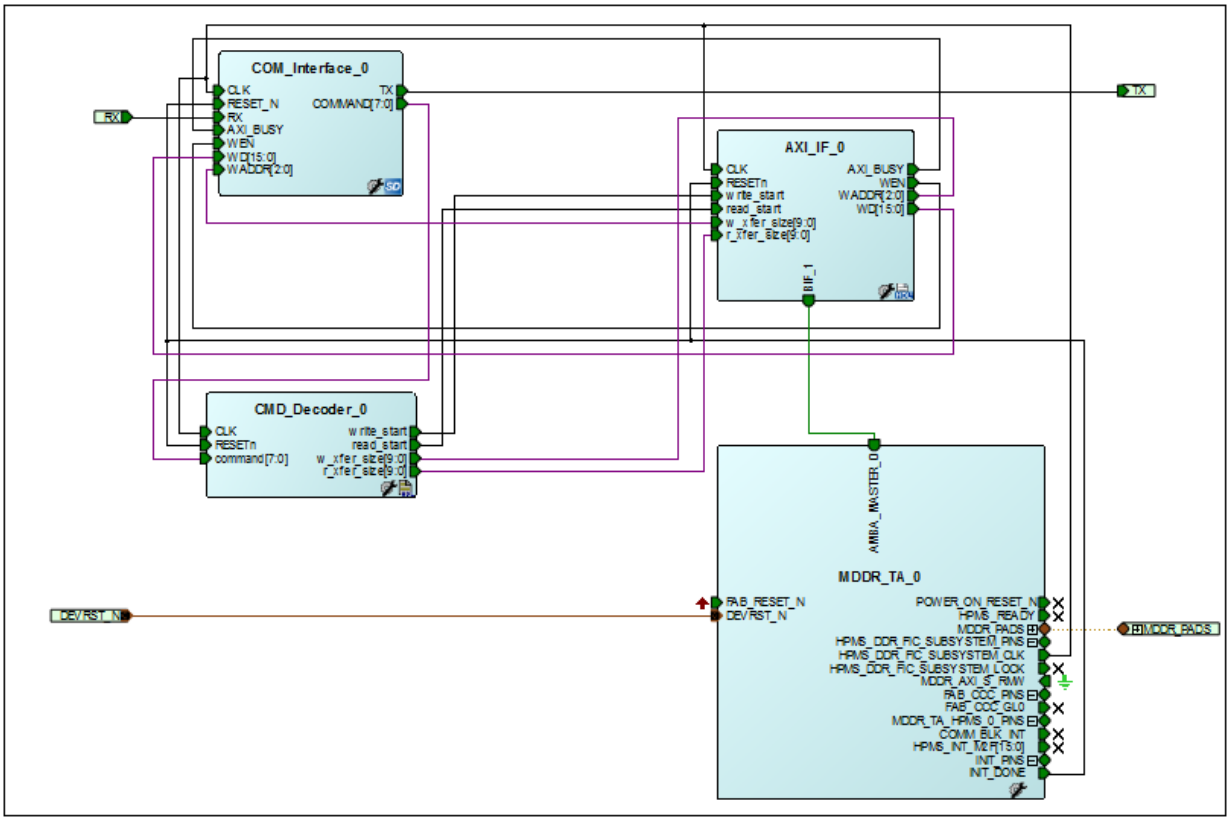

***Figure 7* • Top-Level SmartDesign**

Configuring the System Builder

This section describes how to configure the MDDR and other device features and then build a complete system using the System Builder graphical design wizard in the Libero SoC software. For details on how to launch the System Builder wizard and detailed information on how to use it, refer to the *IGLOO2 System Builder User Guide*.

The following steps describe how to configure the MDDR and access it from AXI master in the FPGA fabric:

1. Go to the **System Builder** - **Device Features** tab and select the **HPMS External Memory (MDDR)** check box and leave the rest of the check boxes unchecked. Figure 8 shows the **System Builder - Device Features** tab.

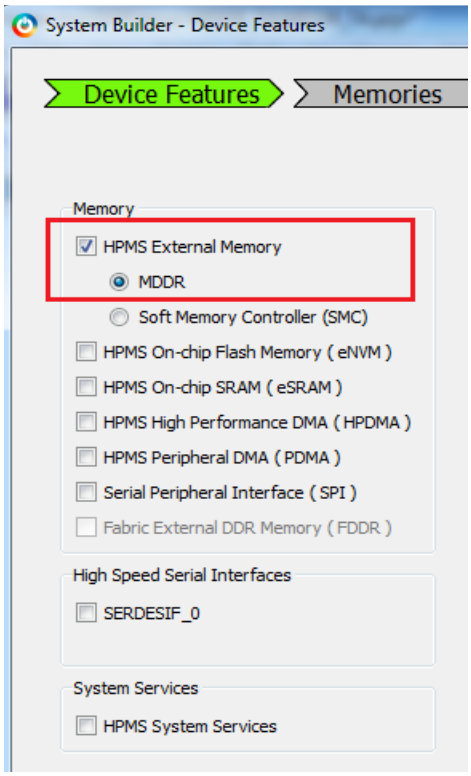

Figure 8 • **System Builder - Device Features Tab**

2. Configure the MDDR in **Memories** tab as shown in Figure 9. In this example, the design is created to access the LPDDR memory with a 16-bit data width and no ECC.

3. Set the DDR memory settling time to 200 µs and click **Import Configuration** file to initialize the DDR memory. The configuration file is stored in eNVM. The MDDR subsystem registers must be initialized before accessing DDR memory through the MDDR subsystem. The MDDR configuration register file is provided along with the design file (Refer to "Appendix: Design Files" section on page 33).

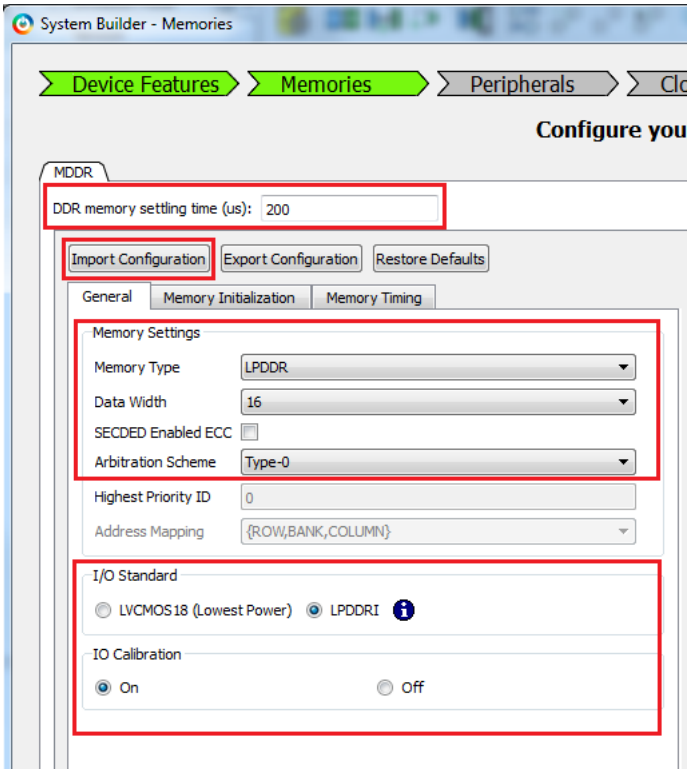

Figure 9 • **Memory Configuration**

4. In the **Peripherals** tab, drag the **Fabric AMBA Master** and drop on to the HPMS DDR FIC subsystem. The **AMBA_MASTER_0** is added to the subsystem and the Interface Type is configured as AXI. Figure 10 shows the **Peripherals** tab with the **AMBA_MASTER_0** added.

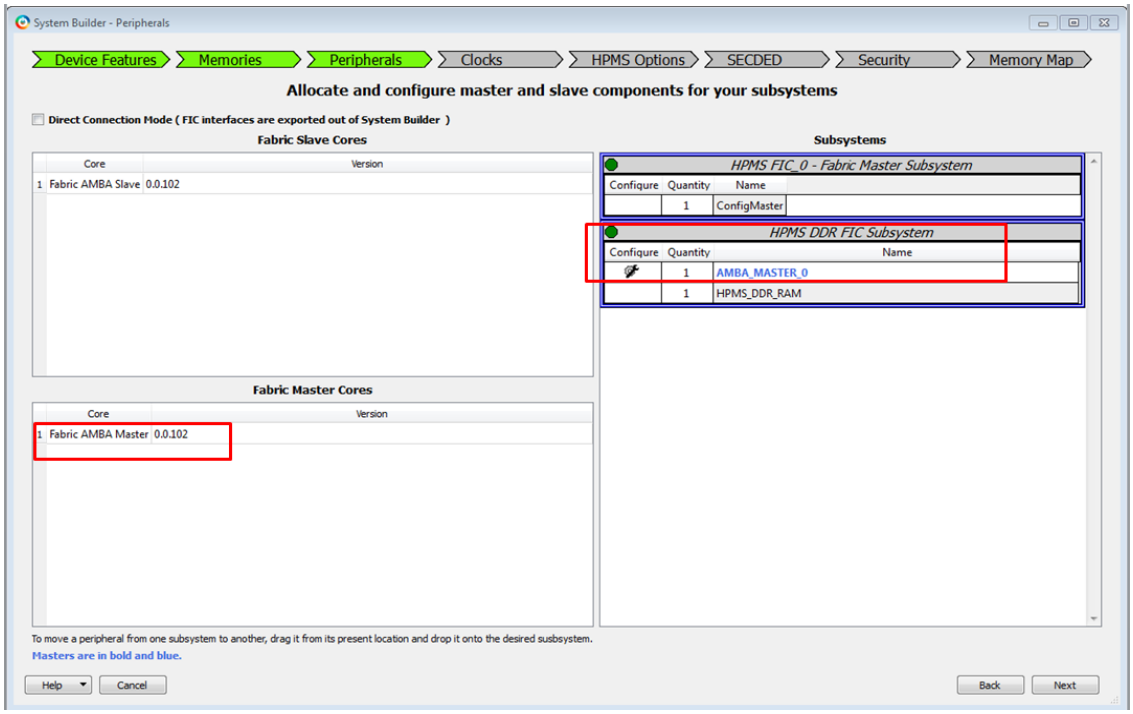

Figure 10 • **Peripherals Tab with the Fabric AMBA Master Added**

5. Configure the System clock and Subsystem clocks in **Clocks** tab as listed in Table 3.

Table 3 • **System and Subsystem Clocks**

Clock Name	Frequency in MHz
System Clock	On-chip 25/50 MHz RC oscillator
HPMS_CLK	83
MDDR_CLK	166
DDR/SMC_FIC_CLK	83
FIC_0_CLK	20.750

Figure 11 shows the **Clocks** configuration tab.

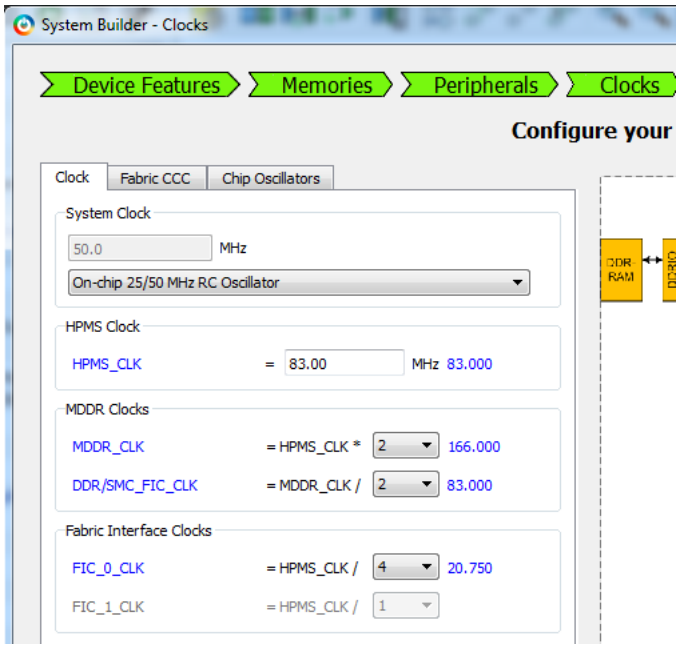

Figure 11 • **System and Subsystem Clocks Configuration**

6. Follow the rest of the steps with default settings and generate the design.
7. Instantiate the custom logic (AXI master, Command decoder, and COM interface) and connect as shown in Figure 7 on page 9.

Figure 12 shows the SmartDesign of the COM interface block. The COM_interface SmartDesign component handles the UART communication between the host PC software utility and the AXI master logic.

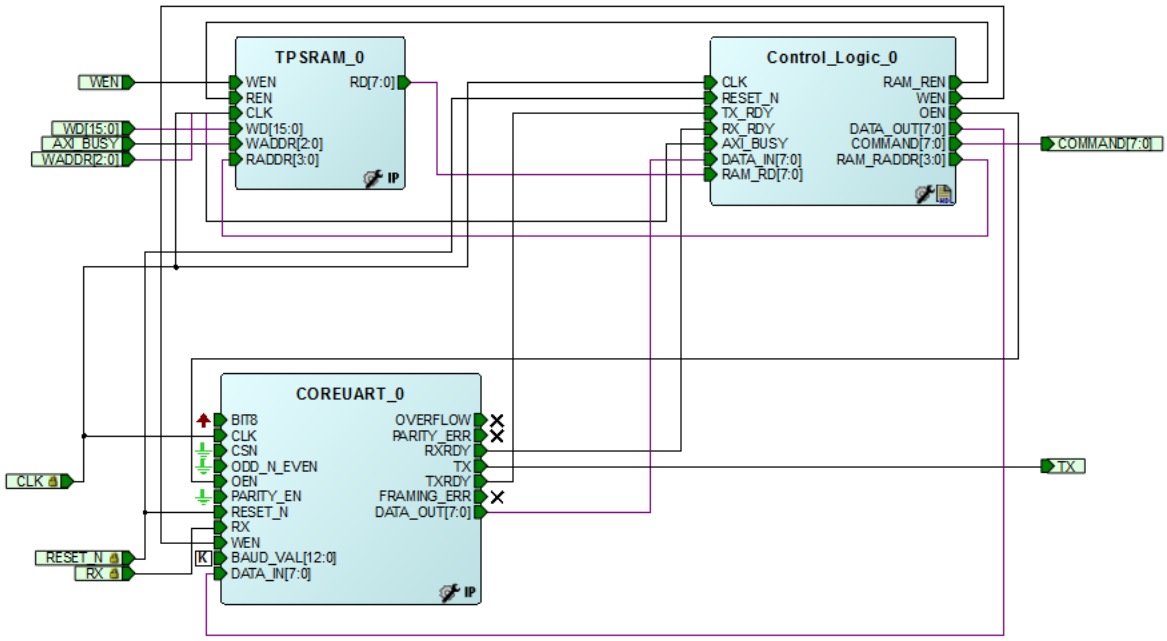

***Figure 12* • SmartDesign of the COM Interface Block**

The COREUART_0 IP receives the UART signals from the host PC user interface. The Control_Logic_0 collects the command from COREUART_0 and sends to AXI master through the Command decoder, which triggers the write/read operation. After the write/read operation, the Control_logic_0 reads the throughput count values from TPSRAM_0 and sends to the host PC through COREUART_0.

The configurations of CoreUART and TPSRAM are given below:

- CoreUART
  - Baud Rate: 115200
  - Data Bits: 8
  - Parity: None.
- TPSRAM
  - Write port depth: 8
  - Write port width: 16
  - Read port depth: 16
  - Read port width: 8

Simulation Using Micron LPDDR SDRAM Model

Setting up the Simulation Model

Setting up and running the simulation involve the following steps:

1. Obtain the Micron LPDDR SDRAM model files: The IGLOO2 Evaluation Kit board has the LPDDR DRAM from Micron with the part number MT46H32M16LFBF-6 IT:C TR. The memory model used in the example design supports this device (Refer to "Appendix: Design Files" section on page 33).
2. Copy the `dram.v` and `dram_parameters.vh` simulation model files to the *\<Libero SoC project directory\>\stimulus* directory.
3. Instantiate and connect the LPDDR SDRAM memory model in the testbench as shown in Figure 13.

```
dram dram_0 (
.CLK(MDDR_CLK),
.CLK_N(MDDR_CLK_N),
.CKE(MDDR_CKE),
.CS_N(MDDR_CS_N),
.RAS_N(MDDR_RAS_N ),
.CAS_N(MDDR_CAS_N),
.WE_N(MDDR_WE_N),
.DM(MDDR_DM_RDQS[1:0]),
.BA(MDDR_BA[1:0]),
.ADDR(MDDR_ADDR[13:0]),
.DQ(MDDR_DQ[15:0]),
.DQS(MDDR_DQS[1:0])
);
```

Figure 13 • **Instantiation of Simulation Model**

4. Ensure that `dram.v` file is included at the top of the testbench file. The example design uses one instance of LPDDR model with the device width 16.

5. Set the testbench in which LPDDR memory model is instantiated as active stimulus. Figure 14 shows the settings under Stimulus Hierarchy.

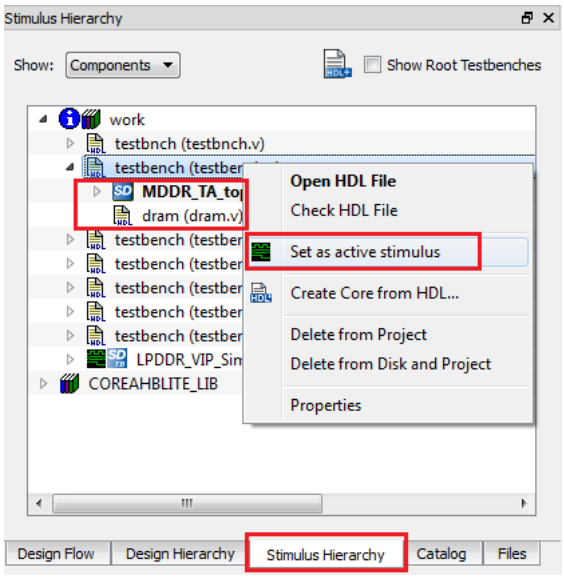

Figure 14 • **Stimulus Settings**

6. Click **Project > Project Settings > Simulation Options > Waveforms**. Figure 15 shows the Waveforms settings on the right.

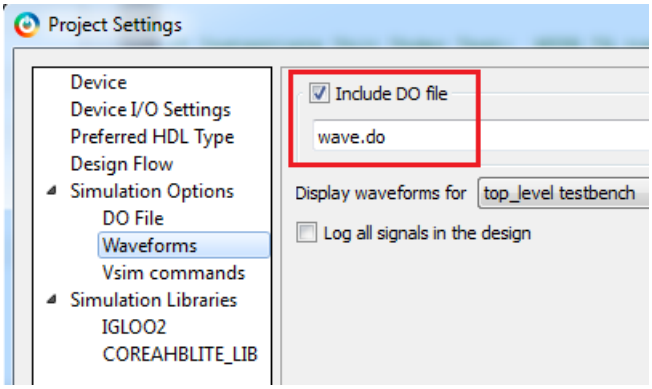

Figure 15 • **Waveforms Settings**

7. Select the **Include DO File** check box and enter **wave.do** in the box as shown in Figure 15.

Timing Diagrams

The timing diagrams from Figure 16 through Figure 18 on page 18 show the write operation. Figure 16 shows the control logic signals in the COM interface block.

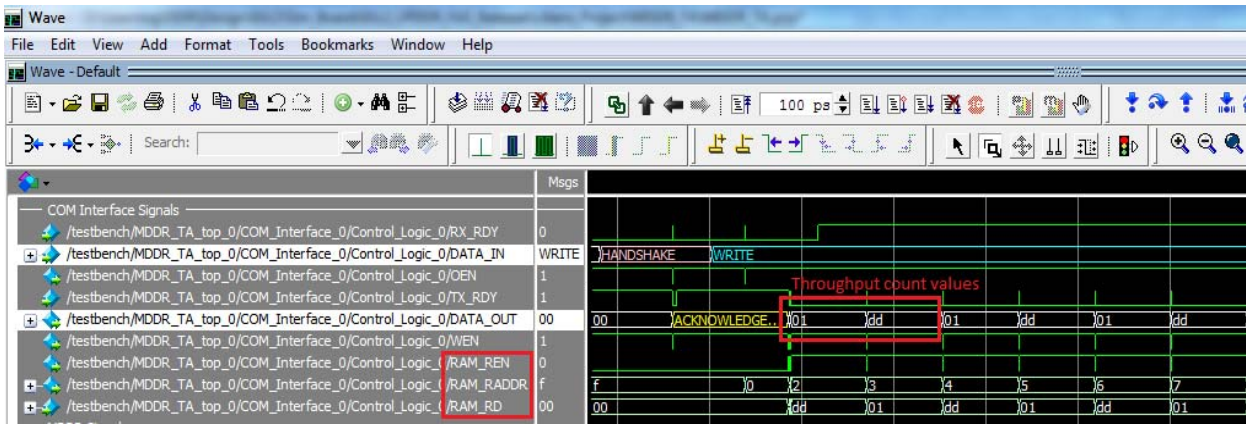

Figure 16 • **Control Logic Signals in the COM Interface Block for Write Operation**

After reset is de-asserted, the control logic receives the handshake (0x63) command through the CoreUART RX port. Then the control logic sends the acknowledgment (0x61) through the CoreUART TX port and waits for the write command. After the write command is received the control logic sends the write command to AXI master through the Command decoder, which triggers the write operation. After the write operation, the control logic reads the throughput count values from TPSRAM and sends to the CoreUART TX port.

Figure 17 shows the MDDR signals. AXI master reads 2 KB of data from LSRAM and writes to LPDDR SDRAM. The write operation is repeated eight times. The data is written into Row 0 and Row 1 of all banks (Bank 0 - Bank 3).

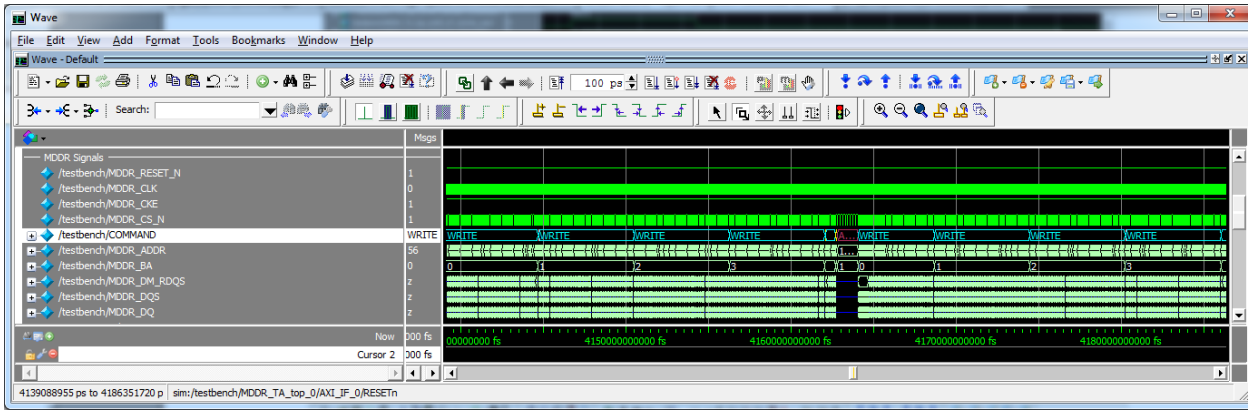

Figure 17 • **MDDR Signals for Write Operation**

Figure 18 shows the AXI master signals. AXI master sends the throughput count value and an address starting from 0x0 to the COM interface block.

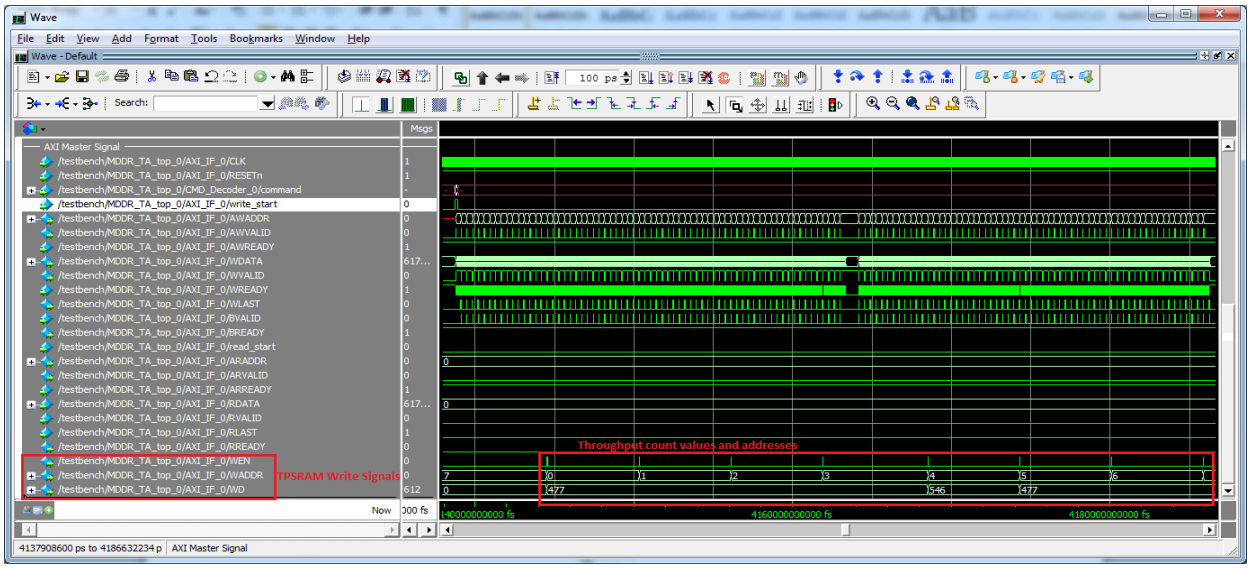

Figure 18 • **AXI Master Signals for Write Operation**

The timing diagrams from Figure 19 through Figure 21 on page 19 show the read operation. Figure 19 shows the control logic signals in the COM interface block.

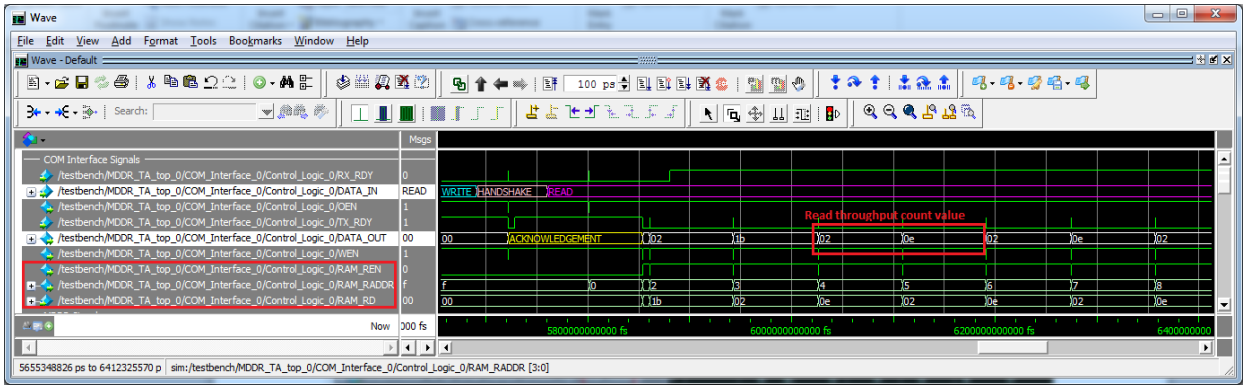

Figure 19 • **Control Logic Signals in the COM Interface Block for Read Operation**

After the write operation, the control logic receives the handshake (0x63) command through the CoreUART RX port. Then the control logic sends the acknowledgment (0x61) through the CoreUART TX port and waits for the read command. After the read command is received, the control logic sends the read command to AXI master through the Command decoder, which triggers the read operation. After the read operation, the control logic reads the throughput count values from TPSRAM and sends to the CoreUART TX port.

Figure 20 shows the MDDR signals. AXI master reads 2 KB of data from LPDDR SDRAM and writes to LSRAM. The read operation is repeated eight times. The data is read from Row 0 and Row 1 of all banks (Bank 0 - Bank 3).

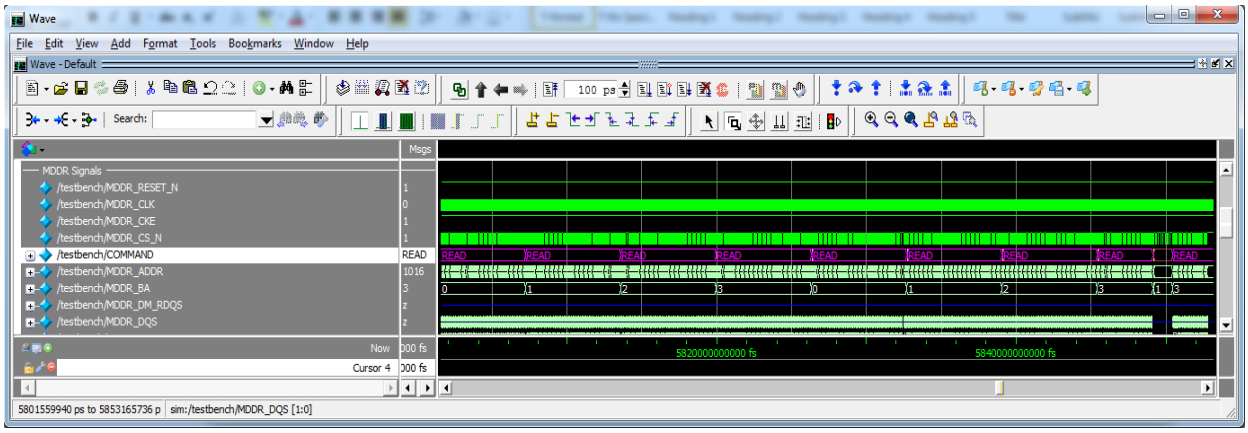

***Figure 20* • MDDR Signals for Read Operation**

Figure 21 shows the AXI master signals. AXI master sends the throughput count value and an address starting from 0x0 to the COM interface block.

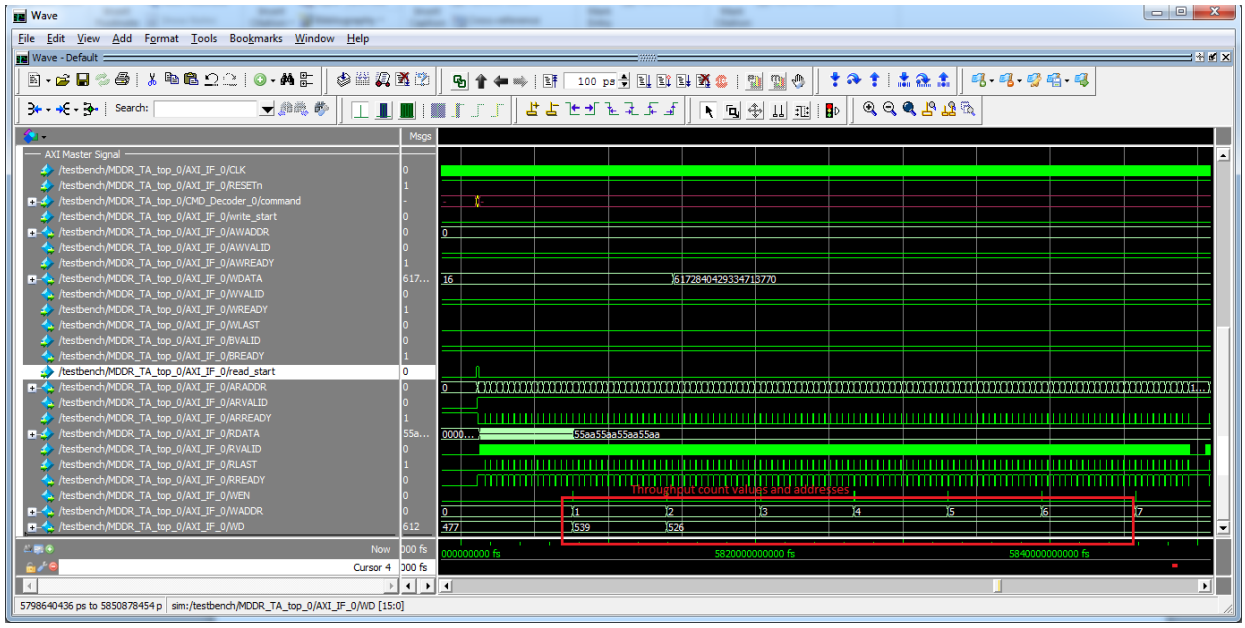

***Figure 21* • AXI Master Signals for Read Operation**

Simulation Using Microsemi LPDDR SDRAM VIP Model

Libero SoC includes a generic DDR memory simulation model, also called verification intellectual property (VIP).The VIP is attached to the pin side of the MDDR or FDDR subsystem, and simulates the functionality of a DDR memory device. It can be configured for DDR2, DDR3, and LPDDR SDRAM memories as well.

Setting Up the Simulation Model

Setting up and running the simulation involve the followings steps:

1. Click **Catalog** tab in the Libero SoC.
2. Select the **Simulation Mode** check box.
3. Under **Memory and Controller**, select **Generic DDR Memory Simulation** model and drag into the SmartDesign testbench canvas. Figure 22 shows the Simulation model.

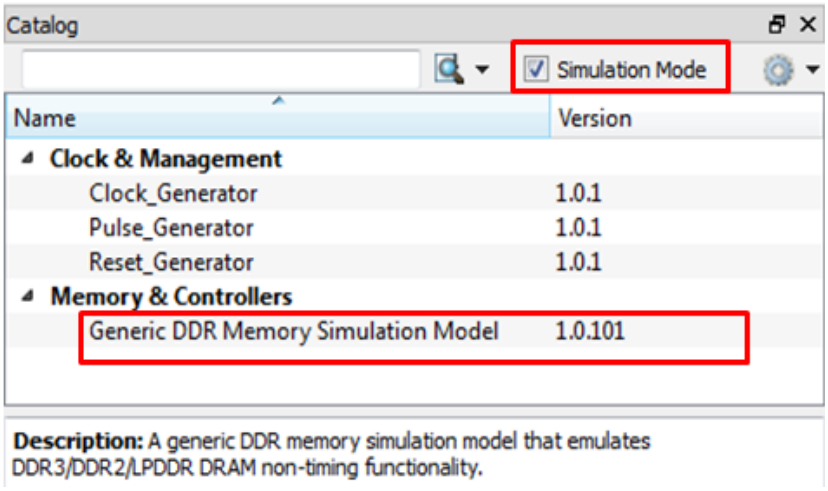

***Figure 22* • Generic DDR Memory Simulation Model**

4. Enter the Generic DDR Memory Simulation model configuration details as shown in Figure 23. The example design uses one instance of SimDRAM (VIP model) with the device width size of 16.

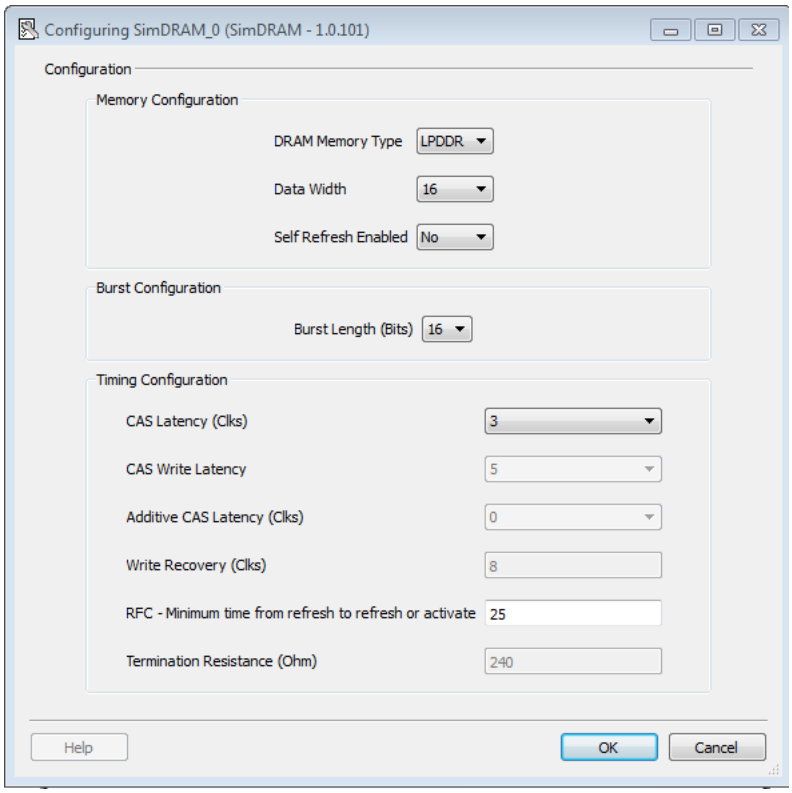

Figure 23 • **Configuring SimDRAM**

5. Connect as described in "Simulation Using Micron LPDDR SDRAM Model" section on page 15. The connections are same as the Micron model. Figure 24 shows the SmartDesign testbench for the example design with Microsemi LPDDR SDRAM VIP model.

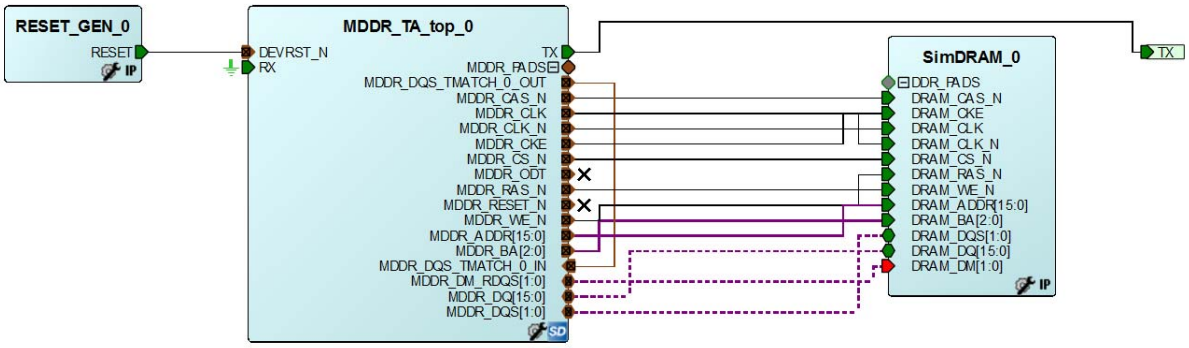

Figure 24 • **SmartDesign Testbench for Example Design with Microsemi LPDDR SDRAM VIP**

6. Generate the design by clicking **SmartDesign > Generate Component** or by clicking **Generate Component** on the SmartDesign tool bar.

7. Open the generated SmartDesign testbench file, **LPDDR_VIP_Simulation.v**. Figure 25 shows the SmartDesign generated testbench file under **Files** tab.

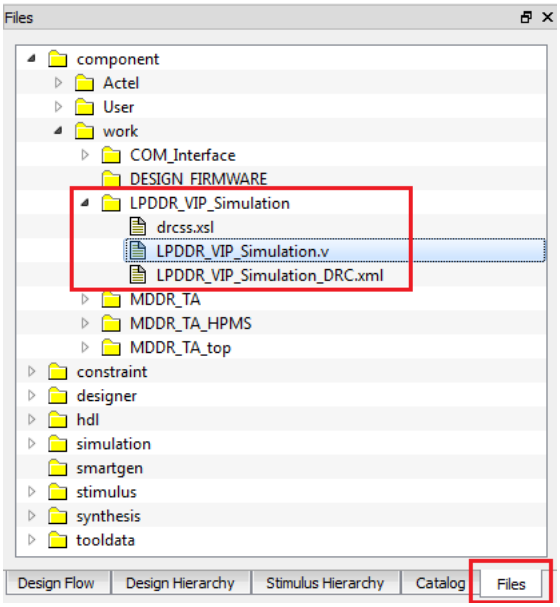

***Figure 25* • SmartDesign Generated Testbench File**

8. Replace timescale 1 ns/100 ps with timescale 1ps/1fs.
9. Add the following code above **endmodule**.

```
wire     MDDR_CLK;
wire     MDDR_CKE;
wire     MDDR_CS_N;
wire [15:0] MDDR_ADDR;
wire [2:0]  MDDR_BA;

wire [3:0]  fsm;
wire [1:0]  MDDR_DM_RDQS;
wire [15:0] MDDR_DQ;
wire [1:0]  MDDR_DQS;
wire [2:0]  COMMAND;
reg fsm_en;

reg BRCLK;
parameter BRCLK_PERIOD  = 8680500;  // 115200Hz

assign MDDR_DM_RDQS = net_2;
assign MDDR_DQ      = net_1;
assign MDDR_DQS     = net_0;
assign MDDR_CLK     = MDDR_TA_top_0_MDDR_CLK;
assign MDDR_CKE     = MDDR_TA_top_0_MDDR_CKE;
assign MDDR_CS_N    = MDDR_TA_top_0_MDDR_CS_N;
assign MDDR_ADDR    = MDDR_TA_top_0_MDDR_ADDR;
assign MDDR_BA      = MDDR_TA_top_0_MDDR_BA;

assign                                  COMMAND                                 =
{MDDR_TA_top_0_MDDR_RAS_N,MDDR_TA_top_0_MDDR_CAS_N,MDDR_TA_top_0_MDDR_WE_N};
```

```
assign            fsm                                                    =
LPDDR_VIP_Simulation.MDDR_TA_top_0.COM_Interface_0.Control_Logic_0.fsm;

initial
begin
    BRCLK     = 1'b0;
    @(posedge LPDDR_VIP_Simulation.MDDR_TA_top_0.AXI_IF_0.CLK);
    repeat(2000)
      begin
         #(BRCLK_PERIOD / 2.0) BRCLK <= !BRCLK;
      end
end

initial
begin
    $display ("++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++");
    $display ("Loading LSRAM from lsram.mem file");
    $display ("");

$readmemh("lsram_512x64.mem",LPDDR_VIP_Simulation.MDDR_TA_top_0.AXI_IF_0.Rdata_m
em);
    $display (" Completed Loading LSRAM");
    $display ("++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++");

    @(posedge LPDDR_VIP_Simulation.MDDR_TA_top_0.AXI_IF_0.RESETn);
    force LPDDR_VIP_Simulation.MDDR_TA_top_0.COM_Interface_0.COREUART_0.DATA_OUT
= 8'b1100011;   /* Handshaking Commmand 'c'  */

                                                                    @(posedge
LPDDR_VIP_Simulation.MDDR_TA_top_0.COM_Interface_0.Control_Logic_0.RX_RDY);
    repeat(5) @(posedge BRCLK);
    force LPDDR_VIP_Simulation.MDDR_TA_top_0.COM_Interface_0.COREUART_0.DATA_OUT
= 8'b00100101;  /* 2KB Write */

@(posedge fsm_en);
    repeat(40) @(posedge BRCLK);
    force LPDDR_VIP_Simulation.MDDR_TA_top_0.COM_Interface_0.COREUART_0.DATA_OUT
= 8'b1100011;  /* Handshaking Commmand 'c'  */

                                                                    @(posedge
LPDDR_VIP_Simulation.MDDR_TA_top_0.COM_Interface_0.Control_Logic_0.RX_RDY);
    repeat(5) @(posedge BRCLK);
    force LPDDR_VIP_Simulation.MDDR_TA_top_0.COM_Interface_0.COREUART_0.DATA_OUT
= 8'b00100110; /* 2KB Read */

end

always @(posedge LPDDR_VIP_Simulation.MDDR_TA_top_0.AXI_IF_0.CLK)
begin
 if(fsm == 4'b1001)
   begin
     fsm_en  <= 1'b1;
   end
 else
   begin
```

```
        fsm_en  <= 1'b0;
    end
end
```

10. Under the **Stimulus Hierarchy** tab, set the SmartDesign testbench as **Set as active stimulus**. Figure 26 shows the **Stimulus Hierarchy** settings.

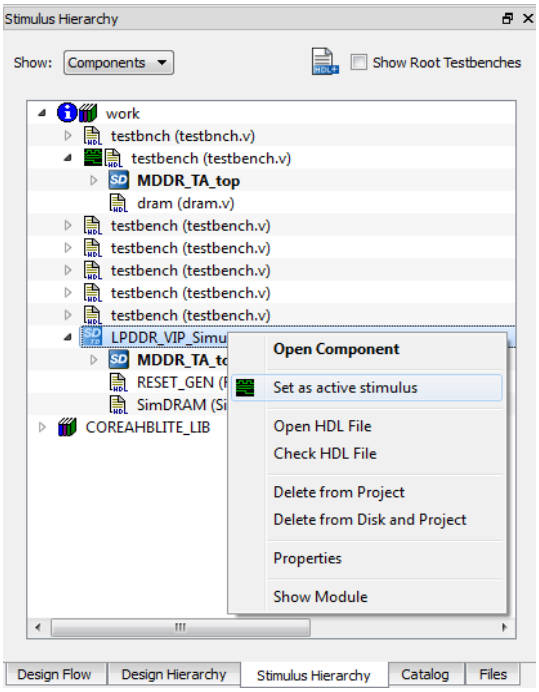

Figure 26 • **Stimulus Settings**

11. Change the default DO file name to **wave_vip.do** file in **Project > Project Settings > Simulation Options > Waveforms**. Figure 27 shows the **Waveforms** settings.

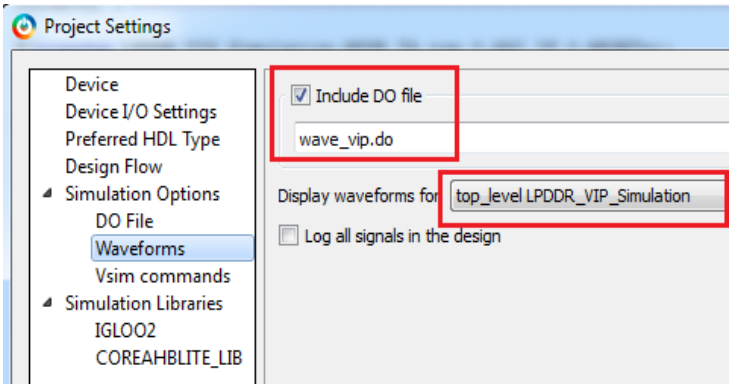

Figure 27 • **Waveforms Settings**

Timing Diagrams

The timing diagrams from Figure 28 through Figure 30 on page 26 show the write operation. Figure 28 shows the control logic signals in the COM interface block.

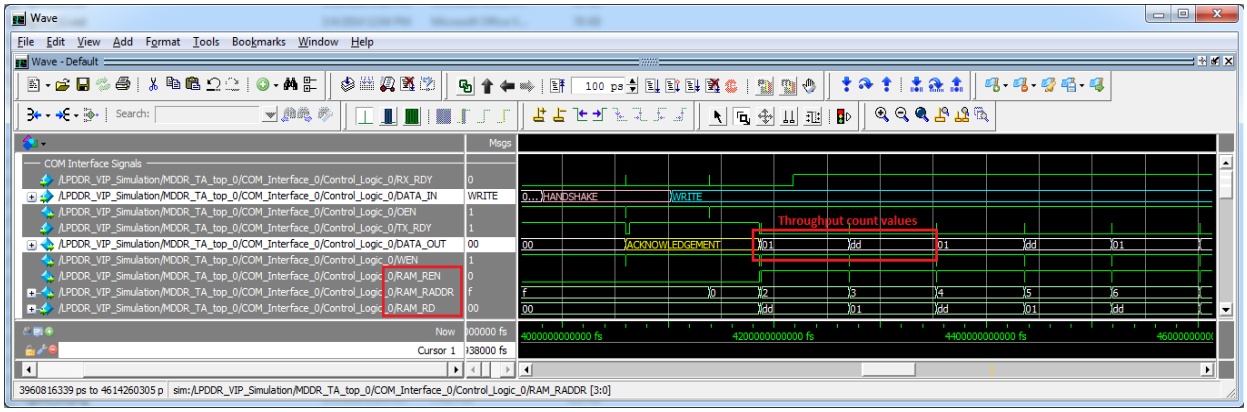

***Figure 28* • Control Logic Signals in the COM Interface Block for Write Operation**

After reset is de-asserted, the control logic receives the handshake (0x63) command through the CoreUART RX port. Then the control logic sends the acknowledgment (0x61) through the CoreUART TX port and waits for the write command. After the write command is received, the control logic sends the write command to AXI master through the Command decoder, which triggers the write operation. After the write operation, the control logic reads the throughput count values from TPSRAM and sends to the CoreUART TX port.

Figure 29 shows the MDDR signals. AXI master reads 2 KB of data from LSRAM and writes to LPDDR SDRAM. The write operation is repeated eight times. The data is written into Row 0 and Row 1 of all banks (Bank 0 - Bank 3).

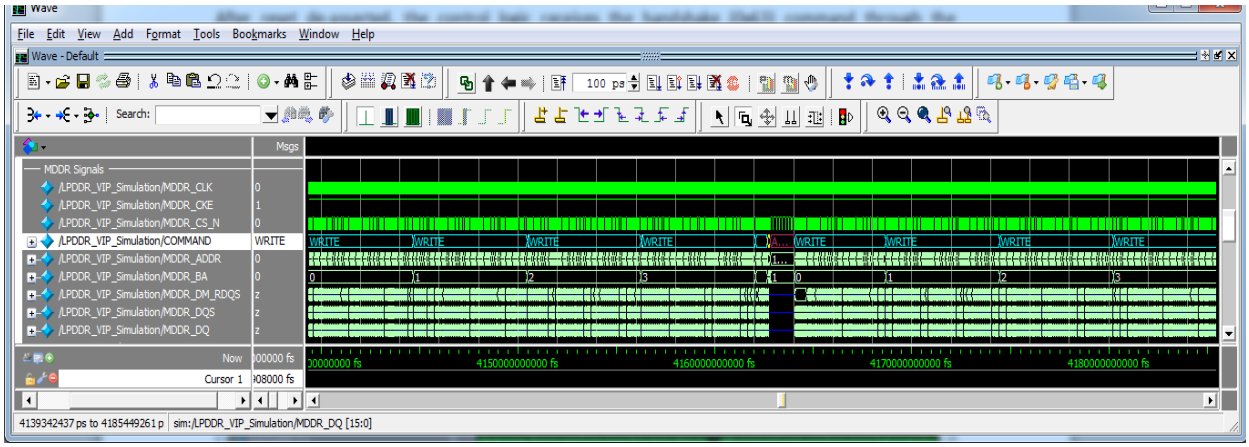

***Figure 29* • MDDR Signals for Write Operation**

Figure 30 shows the AXI master signals. AXI master sends the throughput count value and an address starting from 0x0 to the COM interface block.

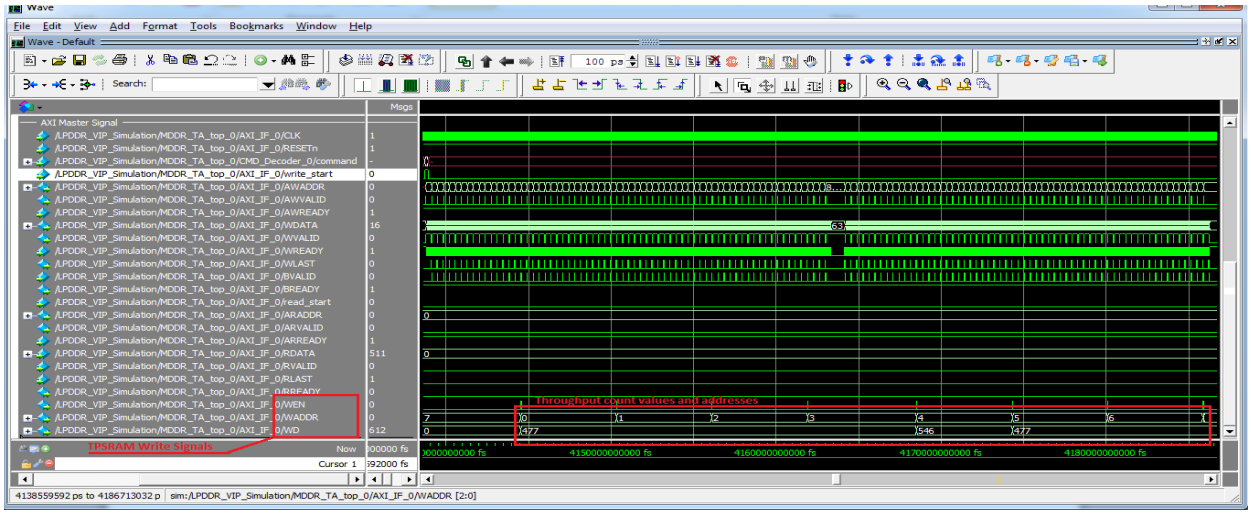

***Figure 30* • AXI Master Signals for Write Operation**

The timing diagrams from Figure 31 through Figure 33 on page 27 show the read operation. Figure 31 shows the control logic signals in the COM interface block.

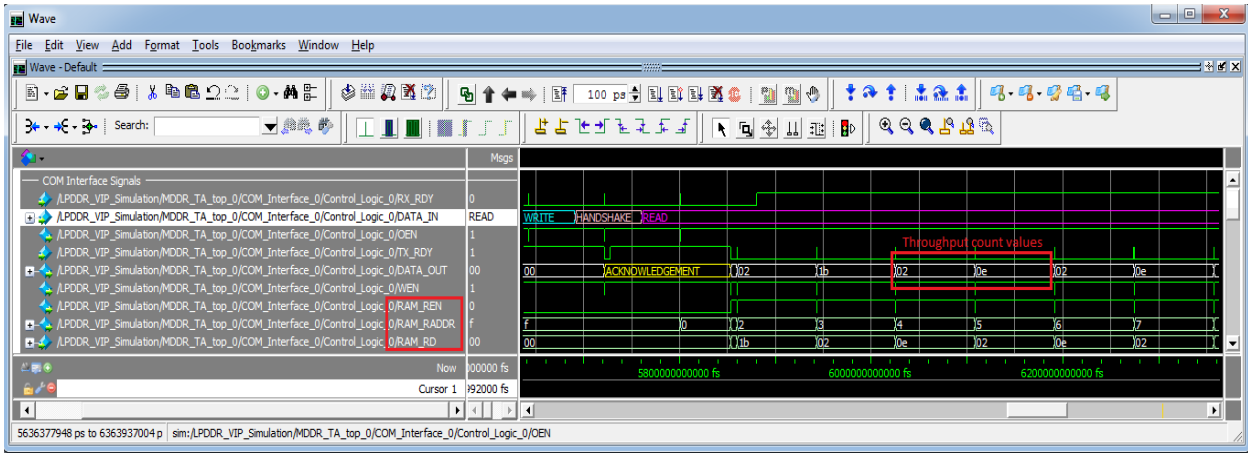

***Figure 31* • Control Logic Signals in the COM Interface Block for Read Operation**

After the write operation, the control logic receives the handshake (0x63) command through the CoreUART RX port. Then the control logic sends the acknowledgment (0x61) through the CoreUART TX port and waits for the read command. After the read command is received, the control logic sends the read command to AXI master through the Command decoder, which triggers the read operation. After the read operation, the control logic reads the throughput count values from TPSRAM and sends to the CoreUART TX port.

Figure 32 shows the MDDR signals. AXI master reads 2 KB of data from LPDDR SDRAM and writes to LSRAM. The read operation is repeated eight times. The data is read from Row 0 and Row 1 of all banks (Bank 0 - Bank 3).

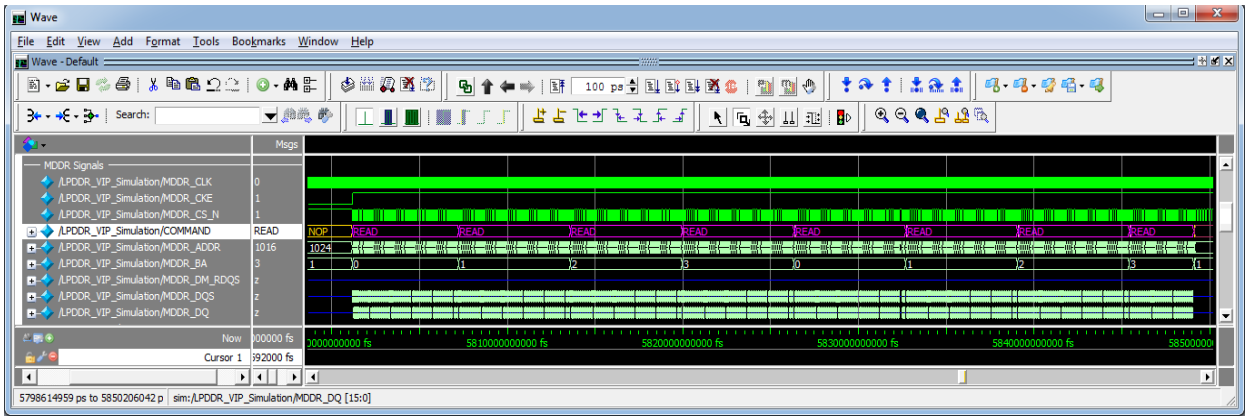

***Figure 32* • MDDR Signals for Read Operation**

Figure 33 shows the AXI master signals. AXI master sends the throughput count value and an address starting from 0x0 to the COM interface block.

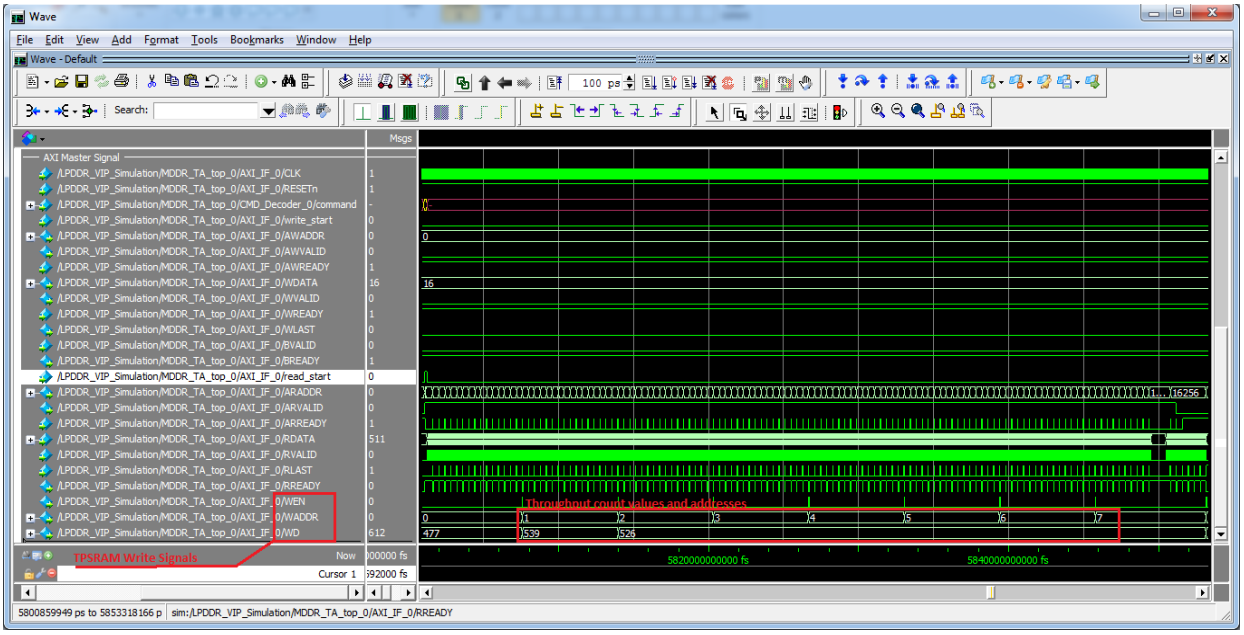

***Figure 33* • AXI Master Signals for Read Operation**

Running the Design

The design example is designed to run on the IGLOO2 Evaluation Kit board. For more detailed board information, refer to *www.microsemi.com/products/fpga-soc/design-resources/dev-kits/igloo2/igloo2-evaluation-kit*.

Setting Up the Hardware

Use the following steps to setup the hardware:

1. Connect the jumpers on the IGLOO2 Evaluation Kit board as listed in Table 4.

Table 4 • **IGLOO2 FPGA Evaluation Kit Jumper Settings**

Jumper	Pin (from)	Pin (to)	Comments
J22	1	2	Default
J23	1	2	Default
J24	1	2	Default
J8	1	2	Default
J3	1	2	Default

CAUTION: While making the jumper connections, the power supply switch **SW7** must be switched OFF.

2. Connect the power supply to the J6 connector; switch ON the power supply switch, **SW7**.
3. Connect the FlashPro4 programmer to the PROG HEADER J5 connector of the IGLOO2 Evaluation Kit board.
4. Connect the host PC USB port to the IGLOO2 Evaluation Kit board's J18 (FTDI) USB connector using the USB mini-B cable.
5. Ensure that the USB to UART bridge drivers are automatically detected by verifying the Device Manager of the host PC.

   If the USB to UART bridge drivers are not installed, download and install the drivers from *www.microsemi.com/soc/documents/CDM_2.08.24_WHQL_Certified.zip*.
6. Program the IGLOO2 Evaluation Kit board with the generated or provided *.stp file (Refer to "Appendix: Design Files" section on page 33) using FlashPro.

Running the Performance Measurement Utility

The example design provides the performance measurement utility, IGL2_LPDDR_BW that runs on the host PC to communicate with the IGLOO2 Evaluation Kit board. The UART protocol is used as the underlying communication protocol between the host PC and the IGLOO2 Evaluation Kit board. Figure 34 shows the initial IGL2_LPDDR_BW utility window.

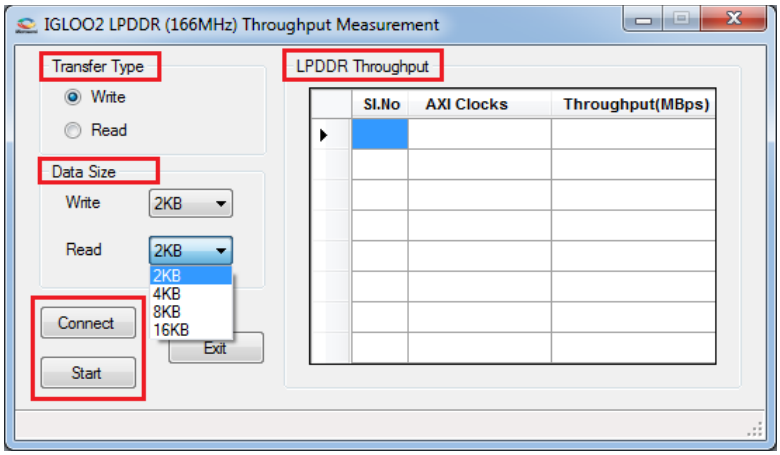

Figure 34 • **IGL2_LPDDR_BW Utility**

The IGL2_LPDDR_BW utility consists the following sections:

- **Transfer Type**: Write or Read
- **Data Size**: Write data size or Read data size can be selected from the drop-down list. The data size varies from 2 KB to 16 KB.
- **LPDDR Throughput**: It displays the number of AXI clocks and corresponding throughput values in MB/s.
- **Buttons**:
  - **Connect** button to connect or disconnect the serial port communication between the host PC and the IGLOO2 Evaluation Kit board.
  - **Start** button to start the performance measurement.
  - **Exit** button to exit the application.

Steps to Run the Utility

1. Launch the utility. The default location is:
   <download_folder>\M2GL_AC424_DF\Windows_Utility\IGL2_LPDDR_BW.exe
2. Click **Connect** and wait for few seconds to connect the proper FDTI COM port. The connection status along with the COM port and Baud rate is shown in the left bottom corner of the window. Figure 35 shows the connection status of the utility.

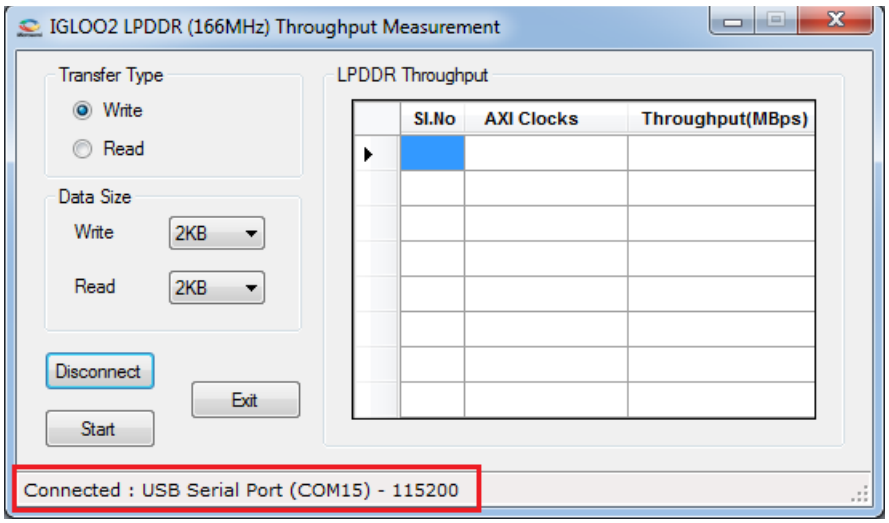

Figure 35 • **IGL2_LPDDR_BW Connection Status**

3. Select Write or Read as **Transfer Type**.
4. Select Write data size or Read data size from the drop-down box and click **Start**. Figure 36 shows the write throughput measurement for 2 KB data transfer.

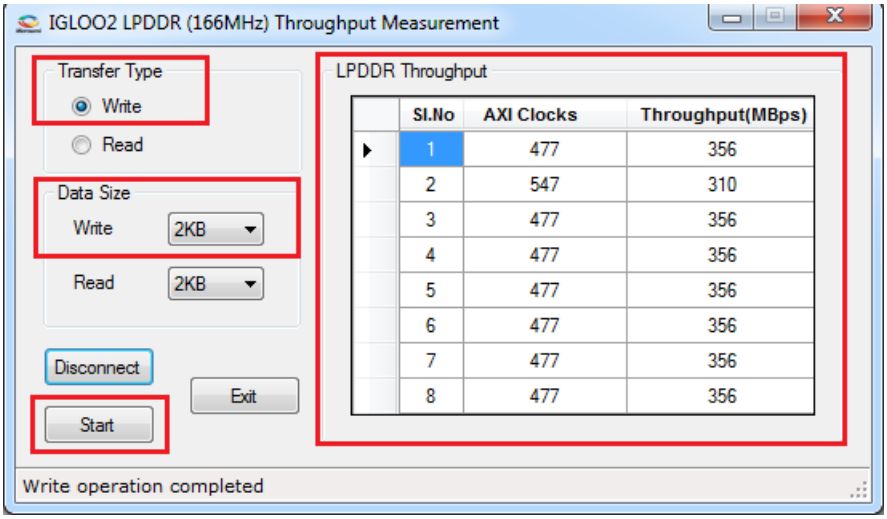

Figure 36 • **Write Throughput Measurement**

The number of AXI clocks may differ for different run. It is due to PRE-CHARGE, ACTIAVTE or REFRESH cycle that runs between the memory transactions.
Table 7 on page 32 lists the write and read bandwidth for data size varies from 2 KB to 16 KB.

LPDDR SDRAM Bandwidth

Table 5 provides 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 Data Size	Total Number of 16 Beat Bursts
2 KB	16
4 KB	32
8 KB	64
16 KB	128

The following equation is applied to calculate the throughput:

$$Bandwidth\ (MB/s) = (16 \div (Total\ number\ of\ AXI\ clocks \div Total\ number\ of\ 16\ beat\ bursts)) \times 8 \times AXI\ clock\ (MHz)$$

EQ 1

Simulation Result

Table 6 lists the write and read bandwidth of LPDDR SDRAM simulation. The incremental pattern of size varies from 2 KB to 16 KB, which is transferred from LSRAM to LPDDR SDRAM and vice-versa.

Table 6 • **LPDDR SDRAM Bandwidth**

Sl No	Optimization Techniques	Size (KB)	Write		Read		Write Improvement	Read Improvement
			Number of Cycles	Bandwidth (MB/Sec)	Number of Cycles	Bandwidth (MB/Sec)		
Base	AXI CLK 80 MHz	2	507	323	721	227	avg:320	avg:225
		4	1019	321	1450	225		
		8	2043	320	2904	225		
		16	4091	320	5809	225		
1	AXI CLK 83 MHz	2	507	335	721	235	avg:332 3.75%	avg:234 4%
		4	1019	333	1450	234		
		8	2043	332	2905	234		
		16	4091	332	5800	234		
2	AXI CLK 83 MHz Without Write Response State	2	477	356	721	235	avg:354 10.6%	avg:234 4%
		4	957	355	1450	234		
		8	1917	354	2905	234		
		16	3837	354	5809	234		
3	AXI CLK 83 MHz Without Write Response State Tuned DDR Configuration	2	477	356	719	236	avg:354 10.6%	avg:235 4.4%
		4	957	355	1440	236		
		8	1917	354	2886	235		
		16	3906	348	5883	231		

Table 6 • **LPDDR SDRAM Bandwidth (continued)**

Sl No	Optimization Techniques	Size (KB)	Write		Read		Write Improvement	Read Improvement
			Number of Cycles	Bandwidth (MB/Sec)	Number of Cycles	Bandwidth (MB/Sec)		
4	AXI CLK 83 MHz Without Write Response State Tuned DDR Configuration Read Command Queuing	2	477	356	526	323	avg:354 10.6%	avg:322 43%
		4	957	355	1054	322		
		8	1917	354	2110	322		
		16	3907	348	4317	315		
5	AXI CLK 100 MHz (MDDR CLK 200 MHZ) Without Write Response State Tuned DDR Configuration Read Command Queuing	2	477	429	526	389	avg:428 33.75%	avg:388 72%
		4	957	428	1054	388		
		8	1917	427	2110	388		
		16	3907	419	4317	379		

Board Test Result

Table 7 lists the write and read bandwidth of LPDDR SDRAM on the IGLOO2 Evaluation kit board. The incremental pattern of size varies from 2 KB to 16 KB, which is transferred from LSRAM to LPDDR SDRAM and vice-versa.

Table 7 • **LPDDR SDRAM Bandwidth**

Sl No	Optimization Techniques	Size (KB)	Write		Read		Write Improvement	Read Improvement
			Number of Cycles	Bandwidth (MB/Sec)	Number of Cycles	Bandwidth (MB/Sec)		
Base	AXI CLK 80 MHz	2	507	323	721	227	avg:320	avg:225
		4	1019	321	1450	225		
		8	2043	320	2905	225		
		16	4091	320	5812	225		
1	AXI CLK 83 MHz	2	507	335	721	235	avg:332 3.75%	avg:234 4%
		4	1019	333	1450	234		
		8	2043	332	2905	234		
		16	4091	332	5809	234		
2	AXI CLK 83 MHz Without Write Response State	2	477	356	721	235	avg:354 10.6%	avg:234 4%
		4	957	355	1450	234		
		8	1917	354	2905	234		
		16	3837	354	5809	234		

Table 7 • **LPDDR SDRAM Bandwidth (continued)**

Sl No	Optimization Techniques	Size (KB)	Write		Read		Write Improvement	Read Improvement
			Number of Cycles	Bandwidth (MB/Sec)	Number of Cycles	Bandwidth (MB/Sec)		
3	AXI CLK 83 MHz Without Write Response State Tuned DDR Configuration	2	477	356	719	236	avg:354 10.6%	avg:235 4.4%
		4	957	355	1444	235		
		8	1917	354	2886	235		
		16	3907	348	5883	231		
4	AXI CLK 83 MHz Without Write Response State Tuned DDR Configuration Read Command Queuing	2	477	356	526	323	avg:354 10.6%	avg:322 43%
		4	957	355	1054	322		
		8	1917	354	2110	322		
		16	3907	348	4313	315		

Conclusion

This application note describes the DDR SDRAM bandwidth optimization techniques with an example design on the IGLOO2 Evaluation Kit board. It also shows the LPDDR SDRAM simulation flow using the Micron LPDDR SDRAM model and Microsemi LPDDR SDRAM VIP model.

Appendix: Design Files

The design files can be downloaded from the Microsemi SoC Products Group website: *http://soc.microsemi.com/download/rsc/?f=m2gl_ac424_liberov11p6_df*

The design file consists Libero SoC Verilog project, MDDR Configuration files, Simulation model files and programming files (*.stp) for the IGLOO2 Evaluation Kit board. Refer to the `Readme.txt` file included in the design file for the directory structure and description.

List of Changes

The following table shows the important changes made in this document for each revision:

Date	Changes	Page
Revision 4 (October 2015)	Updated the document for Libero v11.6 software release (SAR 71831).	NA
Revision 3 (May 2015)	Updated the document for Libero v11.5 software release (SAR 67502).	NA
Revision 2 (August 2014)	Updated the document for Libero v11.4 software release (SAR 59677).	NA

Microsemi Corporate 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

E-mail: sales.support@microsemi.com

Microsemi Corporation (Nasdaq: MSCC) offers a comprehensive portfolio of semiconductor and system solutions for communications, defense & security, aerospace 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; security technologies and scalable anti-tamper products; Ethernet solutions; Power-over-Ethernet ICs and midspans; as well as custom design capabilities and services. Microsemi is headquartered in Aliso Viejo, Calif., and has approximately 3,600 employees globally. Learn more at **www.microsemi.com**.

51900292-4/10-15