

SmartFusion2 - Optimizing DDR Controller for Improved Efficiency - Libero SoC v11.4

Table of Contents

| | |
|---|----|
| Purpose | 1 |
| Introduction | 2 |
| References | 3 |
| Design Requirements | 3 |
| Optimization Techniques | 4 |
| Frequency of Operation | 4 |
| Burst Length | 4 |
| AXI Master without Write Response State | 4 |
| Read Address Queuing | 5 |
| Series of Writes or Reads | 6 |
| DDR Configuration Tuning | 6 |
| Implementation on SmartFusion2 Device | 8 |
| Design Description | 8 |
| Hardware Implementation | 11 |
| Configuring the System Builder | 12 |
| Simulation using Micron DDR3 SDRAM Model | 18 |
| Simulation using Microsemi DDR3 SDRAM VIP Model | 23 |
| Software Implementation | 30 |
| Running the Design | 32 |
| Board Jumper Settings | 32 |
| Host PC to Board Connections | 32 |
| USB Driver Installation | 32 |
| Steps to Run the Design | 33 |
| DDR3 SDRAM Bandwidth | 37 |
| Conclusion | 39 |
| Appendix A – Design Files | 40 |
| List of Changes | 41 |

Purpose

This document describes the techniques for improving the efficiency of double-data-rate (DDR) Controller using an example design for the SmartFusion®2 Development Kit board. It also provides details about implementing the DDR SDRAM simulation flow using the Micron DDR3 SDRAM model and Microsemi DDR3 SDRAM verification ip (VIP) model.

Introduction

The SmartFusion2 device has two high-speed hardened ASIC memory controllers Microcontroller subsystem (MSS) 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 MSS 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 an FPGA fabric master. The FPGA fabric masters communicate with the MDDR and FDDR subsystems through the AXI or AHB interfaces.

Figure 1 illustrates the MDDR data path for AXI/AHB interfaces.

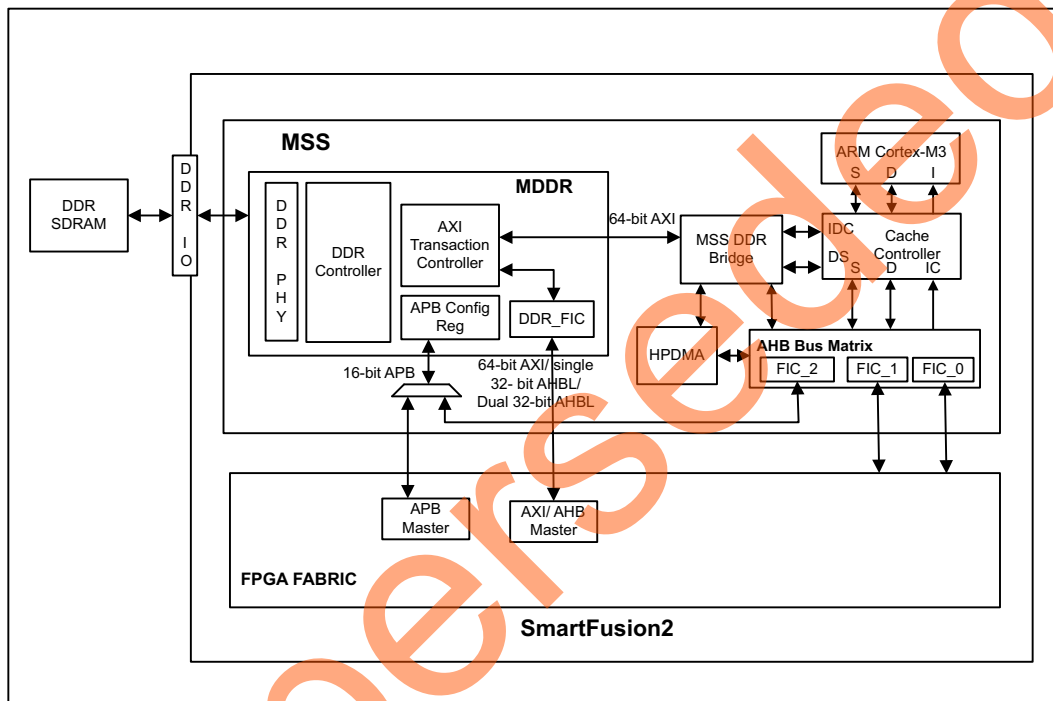


Figure 1 • MDDR Data Path for AXI/AHB Interfaces

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 SmartFusion2 Development Kit board. Refer to [SmartFusion2 SoC FPGA High Speed DDR Interfaces User Guide](#) for more information on MDDR and FDDR subsystems.

The sample design consists of an AXI master, LSRAM, and counters for throughput measurement. During the write operation, the AXI master reads the LSRAM and writes to the DDR3 memory and measures the throughput. During the read operation, the AXI master reads the DDR3 memory and writes to LSRAM and measures the throughput. The throughput values are displayed on the Host PC using the UART interface.

Following are the types of memory simulation models that can be used:

- **Microsemi provided generic DDR memory simulation model (VIP):** The Libero® System-on-Chip (SoC) includes a JEDEC compliant VIP model. This VIP model is attached to the pin side of the MDDR/FDDR subsystem and simulates the functionality of a DDR memory device. This VIP model can be configured for DDR2, DDR3, and LPDDR SDRAM memories. This VIP model 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. Make sure that the downloaded simulation model is JEDEC compliant.

This document also describes the DDR SDRAM simulation flow using the Micron DDR3 SDRAM and Microsemi DDR3 SDRAM VIP models.

References

The following list of references is used in this document. These references complement and help in understanding the relevant Microsemi® SmartFusion2 System-on-Chip (SoC) field programmable gate array (FPGA) device features and flows that are described in this document:

- [SmartFusion2 SoC FPGA High Speed DDR Interfaces User Guide](#)
- [Connecting User Logic to AXI Interfaces of High-Performance Communication Blocks-SmartFusion2](#)
- [Connecting User Logic to the SmartFusion Microcontroller Subsystem](#)
- [DDR Controller and Serial High Speed Controller Initialization Methodology](#)
- [SmartFusion2 Development Kit User Guide](#)

Design Requirements

Table 1 lists the design requirements.

Table 1 • Design Requirements

| Design Requirements | Description |
|--|-------------------------------------|
| Hardware Requirements | |
| SmartFusion2 Development Kit | Rev D or later |
| Host PC or Laptop | Any 64-bit Windows Operating System |
| Software Requirements | |
| Libero SoC | v11.4 |
| SoftConsole | v3.4 |
| One of the following serial terminal emulation programs: <ul style="list-style-type: none"> • HyperTerminal • TeraTerm • PuTTY | - |

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 divider ratios can be selected directly 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 configured to use 333.33 MHz as DDR clock frequency and 166.66 MHz as AXI clock. 166.66 MHz is the fastest clock frequency rate available to run the MDDR_CLK as this is the limit of the MSS CLK_BASE.

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 read and write. The AXI beat burst length (write and read) and burst length of DRAM affect the optimal performance, but by setting the maximum supported burst length for DDR SDRAM and AXI interface achieve the optimal performance. The design example uses a DDR SDRAM burst length of 8 and an AXI write and read beat burst length of 16.

Note: The design example is designed to run on the SmartFusion2 Development Kit board, which has the SmartFusion2 M2S050 device and a DDR3 SDRAM from Micron with the part number; MT41J256M8HX-15E. Both the devices support the maximum burst length of 8.

AXI Master without Write Response State

When the AXI master sends the last data (D (A15)), the WLAST signal goes HIGH, which indicates that the last transfer is in the first write burst. When the 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, the AXI master should wait for the write response before initiating the next write transaction. However, the time spent waiting for the write response wastes the clock cycles and reduce the 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 decreasing 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 has been fully accepted. The AXI protocol has a defined methodology for handling the termination of write burst transaction. This should be followed if the write response channel returns a non-OKAY value.

Figure 2 illustrates the write transaction timing diagram without the write response state.

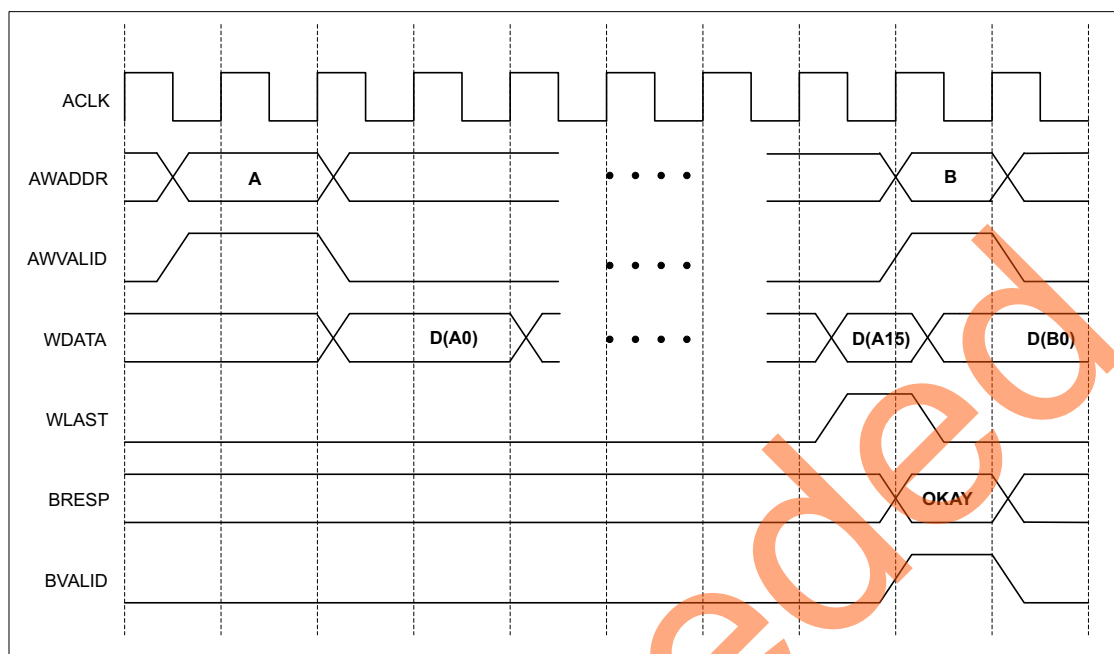


Figure 2 • Write Transaction Timing Diagram without Write Response State

Read Address Queuing

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

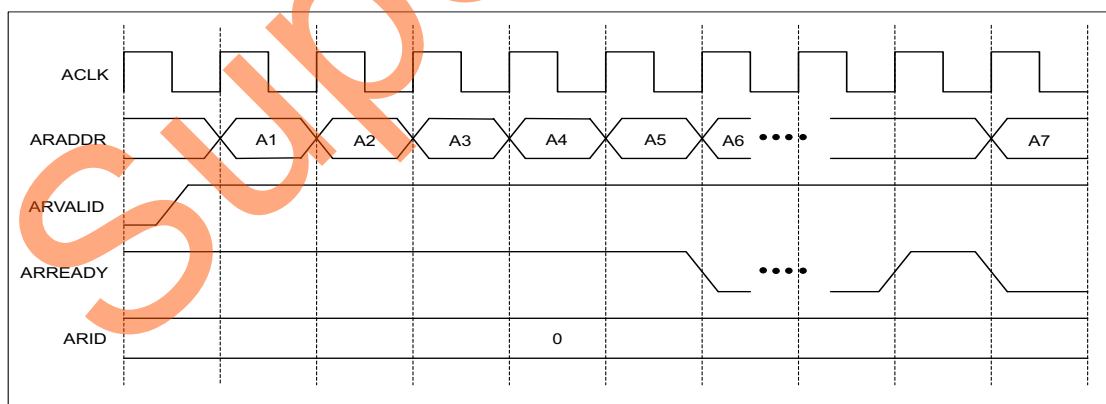


Figure 3 • Read Transaction Timing Diagram with Burst Read Address Queuing

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 on page 38](#) and [Table 8 on page 39](#) 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 to the remainder of the requested data, the requested data is returned in sequence without a large read access penalty associated with the first read.

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:

1. Single beat burst read and write operation
2. Random read and write operation
3. 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](#) illustrates the AXI to DDR3 address mapping for the DDR3 SDRAM on SmartFusion2 Development Kit board.

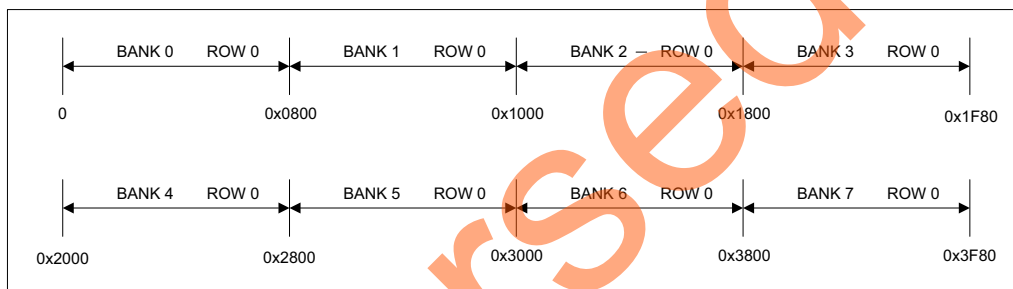


Figure 4 • AXI to DDR3 the Address Mapping

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 0x4000. 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 increase 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 should 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 SmartFusion2 Development Kit DDR3 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 parameter for better performance than that default configuration file.

Table 2 • Tuned DDR Timing Parameters

| Parameters | Default Values | Tuned Values |
|------------|----------------|--------------|
| CAS | 6 (clk) | 5 |
| RAS min | 15 | 12 |
| RAS max | 8192 | 22528 |
| RCD | 6 (clk) | 5 |
| RP | 7 (clk) | 5 |
| REFI | 3104 | 2592 |
| RC | 51 | 17 |
| RFC | 79 | 54 |
| WR | 6 | 5 |
| FAW | 32 | 10 |

Superseded

Implementation on SmartFusion2 Device

The optimization techniques that are mentioned in the above section have been implemented and validated using the SmartFusion2 Development Kit board. This section describes the following:

- Design Description
- Hardware Implementation
- Software Implementation
- Running the Design

Design Description

The design consists of MSS, CoreConfigP IP, CoreResetP IP, SYSRESET_POR Macro, on-chip 25/50 MHz RC oscillator, Fabric CCC (FCCC), AXI master (AXI_IF), AHB master (AHB_IF), and a command decoder (CMD_Decoder). Figure 5 shows the block diagram of the design.

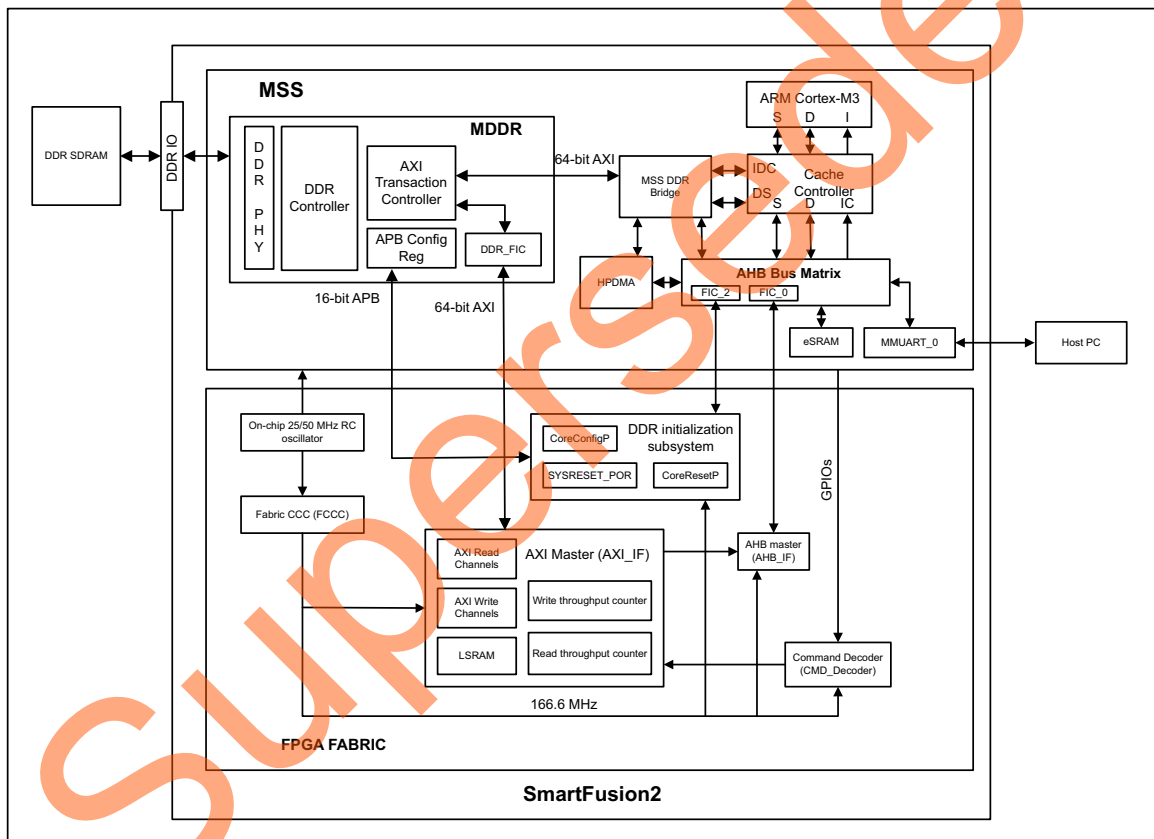


Figure 5 • Top-Level Block Diagram of the Design

MSS is configured to use one UART interface (MMUART_0), MSS clock conditioning circuit (MSS_CCC), RESET Controller, eight GPIOs, one instance of the fabric interface (FIC_0), FIC_2 (Peripheral Initialization), and MDDR.

The FIC_0 interface is configured to use a slave interface with the AHB-Lite (AHBL) interface type. The FIC_2 is configured to initialize the MSS DDR using the ARM® Cortex™-M3 processor along with the CoreConfigP, CoreResetP, and SYSRESET_POR macro. The MMUART_0 is used as an interface for writing to the HyperTerminal. Eight GPIOs are configured as output and routed to the FPGA fabric. The Cortex-M3 processor initiates the AXI write and read operation using these GPIOs. The MDDR is configured to use the DDR3 interface and routed the AXI interface to the FPGA fabric.

FCCC is configured to provide the 166.6 MHz reference clock to the MSS_CCC and the fabric logic. The on-chip 25 MHz/50 MHz RC oscillator is the reference clock source for the FCCC.

Table 3 lists the MSS_CCC generated clocks.

Table 3 • MSS_CCC Generated Clocks

| Clock Name | Frequency in MHz |
|-----------------|------------------|
| M3_CLK | 166.6 |
| MDDR_CLK | 333.2 |
| DDR_SMC_FIC_CLK | 166.6 |
| APB_0 | 83.3 |
| APB_1 | 83.3 |
| FIC_0_CLK | 166.6 |

The command decoder receives the AXI transaction control from the Cortex-M3 processor through GPIOs and generates write, read, write size, and read size signals. Figure 6 illustrates the command decoding.

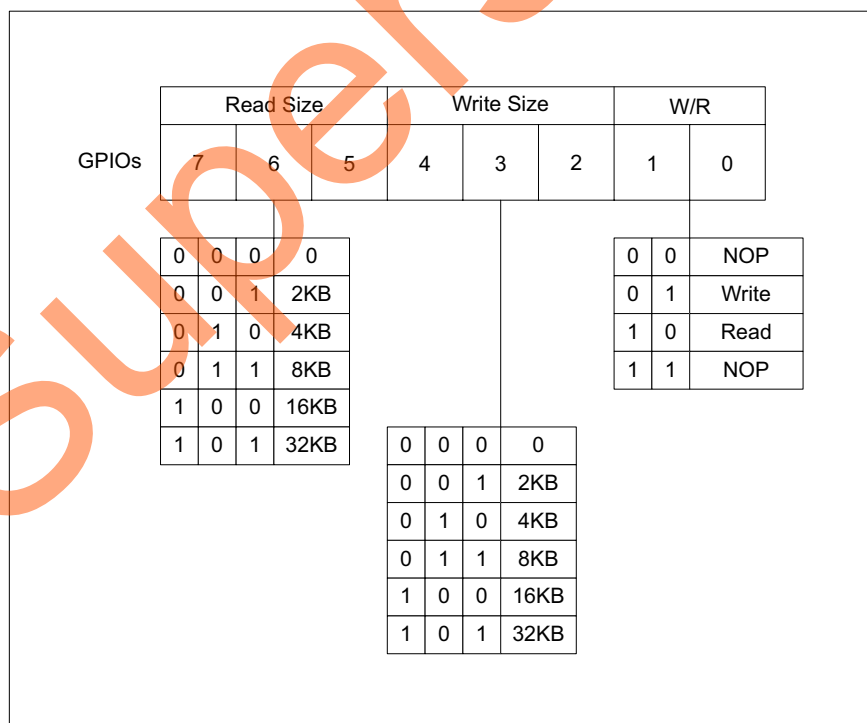


Figure 6 • Command Decoding

The AXI master block consists of AXI read channel, AXI write channel, write throughput counter, read throughput counter, and 512x64 LSRAM. It performs the write or read operation¹ based on the input signals from the command decoder. During the write operation, the AXI master reads the LSRAM and writes into the DDR3 memory, and then measures the write throughput. During the read operation, the AXI master reads the DDR3 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, the AXI master performs the write or read operation eight times to get the average throughput. 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 0x01000000, and is incremented by 128.

After each write or read operation, the AXI master sends the throughput count value and an eSRAM address starting from 0x20008104 to the AHBL master. Then, the AHBL master writes the throughput values into eSRAM. After that the Cortex-M3 processor reads the values and sends to the host PC using the UART interface.

Refer to [Connecting User Logic to AXI Interfaces of High-Performance Communication Blocks in the SmartFusion2 Devices](#) application note for information on creating a custom AXI interface on user logic.

Refer to [Connecting User Logic to the SmartFusion Microcontroller Subsystem](#) application note for information on creating a custom AHB interface on user logic.

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

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 the SmartFusion2 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 check the **MDDR** check box. Leave the rest of the check boxes unchecked. [Figure 8](#) shows the **System Builder - Device Features** tab.

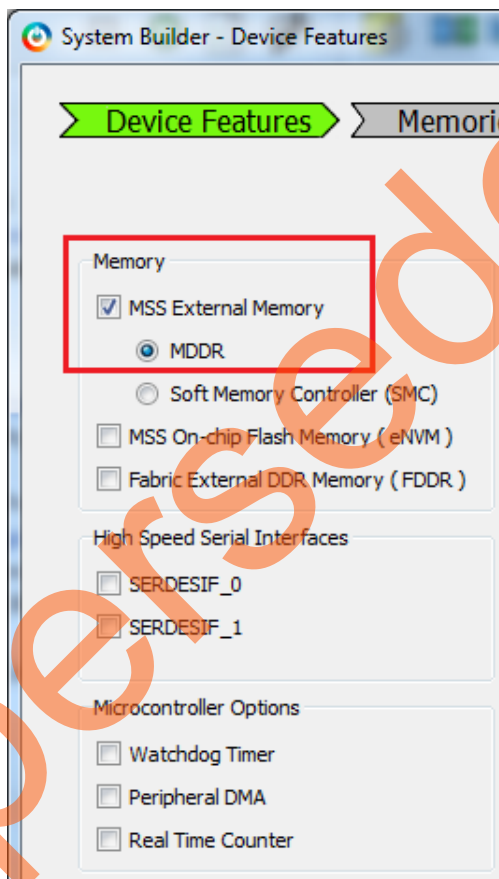


Figure 8 • System Builder - Device Features Tab

2. Configure the MDDR in **Memories** tab as shown in [Figure 9](#). In this example, the design is created to access the DDR3 memory with a 16-bit data width and no ECC.
3. Set the DDR memory settling time to 200 μ s and click **Import Configuration** file to initialize the DDR memory. The configuration file is stored in eNVM. The MDDR subsystem registers should be initialized before accessing DDR memory through the MDDR subsystem. The MDDR configuration register file is provided along with the design file (refer "[Appendix A – Design Files](#)" on page 40).

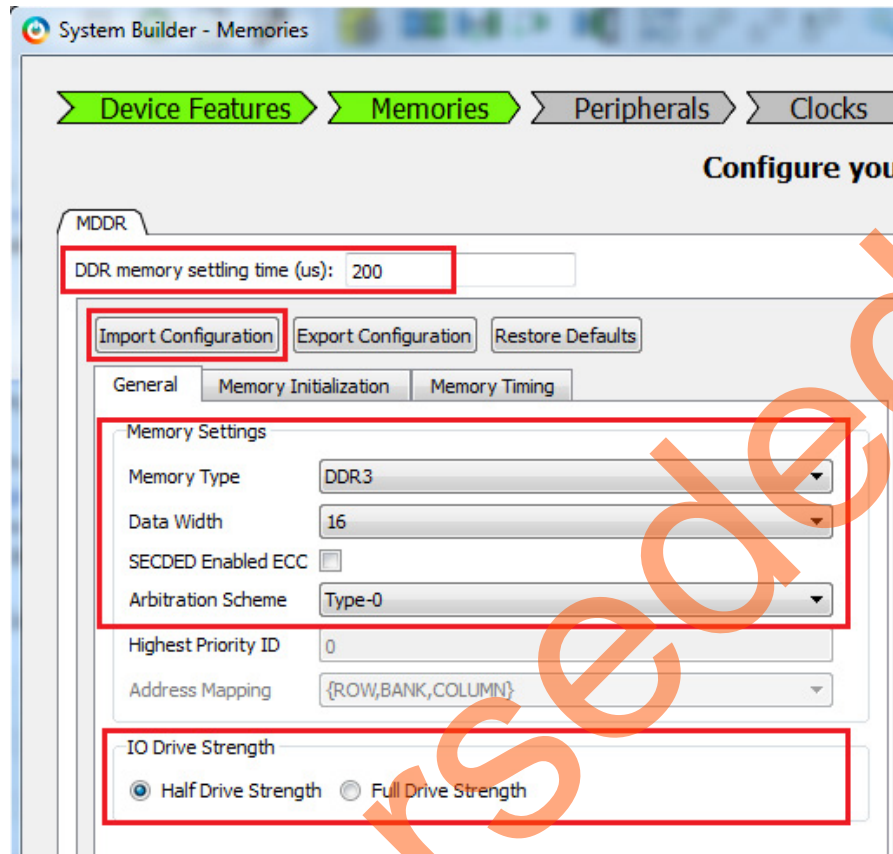


Figure 9 • Memory Configuration

4. Drag the **Fabric AMBA Master** in the peripherals tab and drop on to the MSS DDR FIC Subsystem. The AMBA_MASTER_0 is added to the subsystem. Configure the Interface Type as AXI. Figure 10 shows the Peripherals tab.

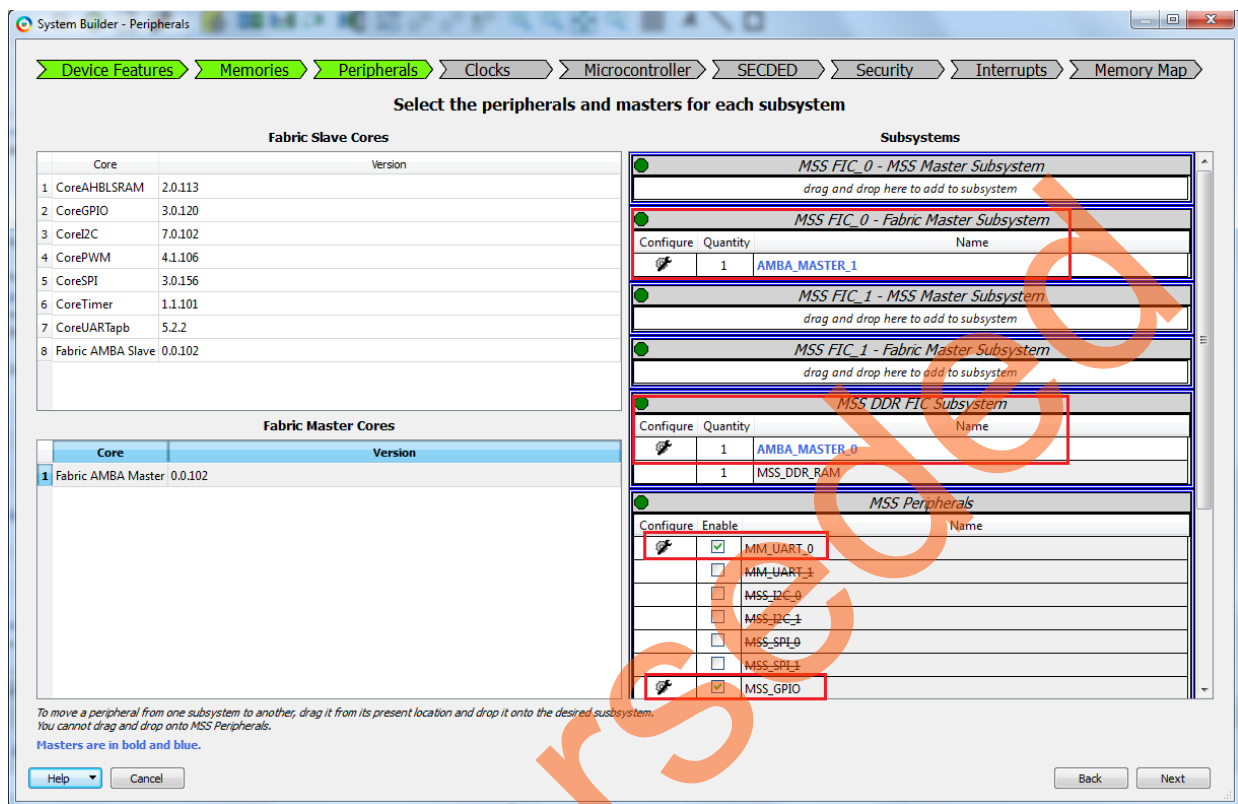


Figure 10 • Selecting MMUART_0 and MSS GPIO in Peripherals Tab

5. Drag the **Fabric AMBA Master** and drop on to the **MSS FIC_0 - Fabric Master Subsystem**. The **AMBA_MASTER_1** is added to the subsystem and configured with AHB Lite.
6. The design uses MMUART and GPIO MSS peripherals. Select **MM_UART_0**, **MSS_GPIO** and uncheck all other peripherals.

7. Select **Fabric** under **Connect To** option in **MM_UART_0 Configuration** dialog.

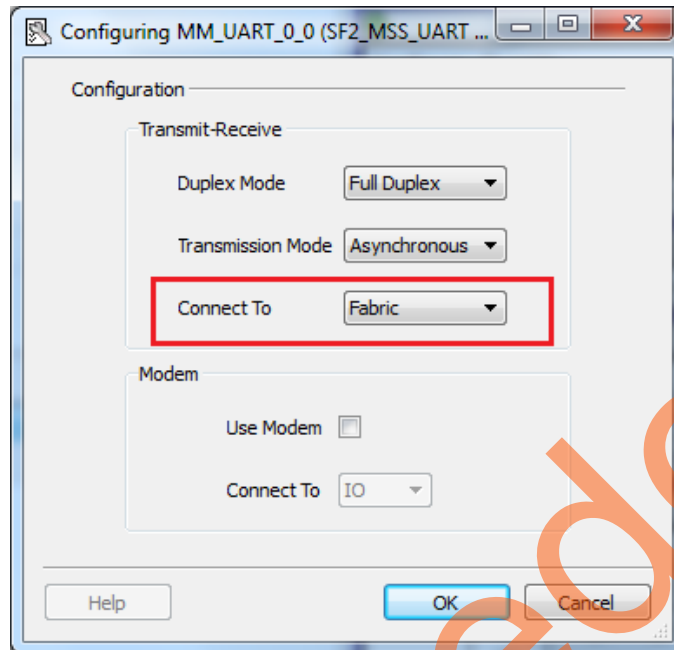


Figure 11 • MM_UART_0 Configuration Dialog

8. Use the settings in **MSS_GPIO Configurator tab** as shown in [Figure 12](#) and keep the rest at default states. Eight GPIOs are configured as output and routed to FPGA fabric.

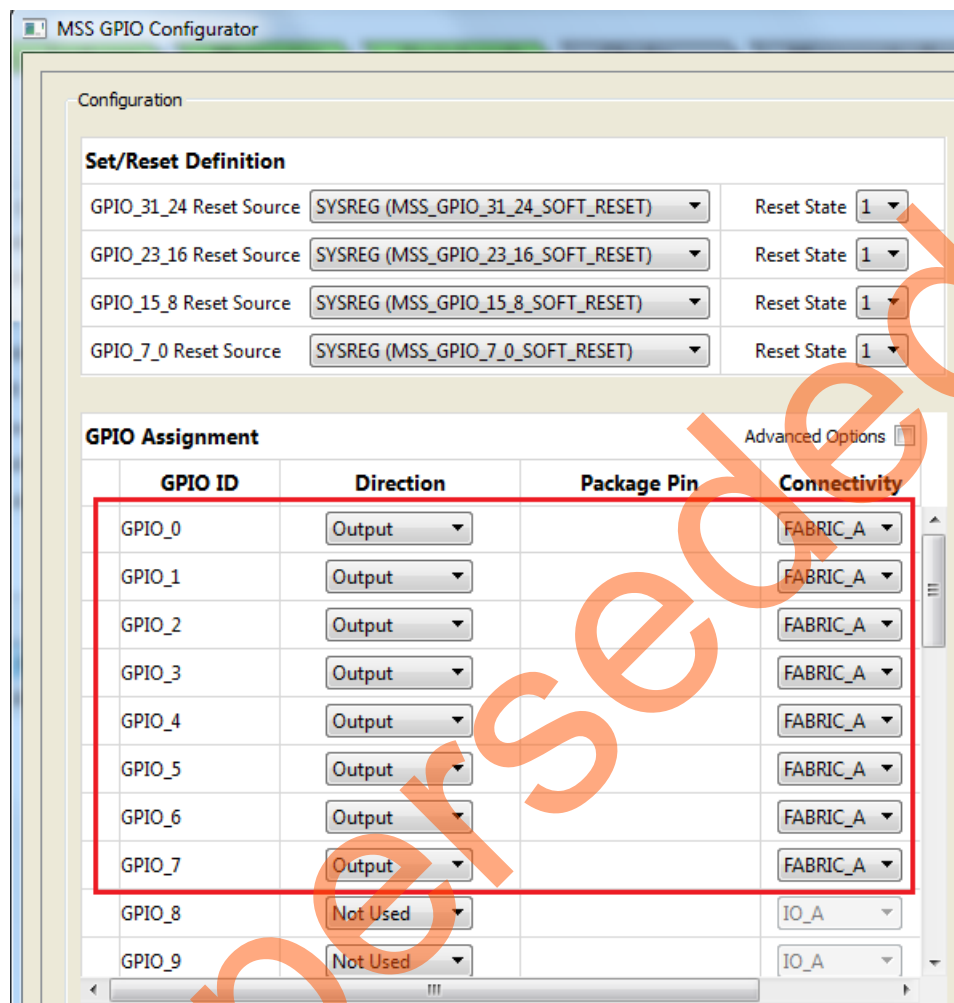


Figure 12 • MSS GPIO Configuration

9. Configure the System Clock and Subsystem clocks in Clocks tab as listed in [Table 4](#).

Table 4 • System and Subsystem Clocks

| Clock Name | Frequency in MHz |
|-----------------|-------------------------------------|
| System clock | On-chip 25 MHz/50 MHz RC oscillator |
| M3_CLK | 166.6 |
| MDDR_CLK | 333.2 |
| DDR/SMC_FIC_CLK | 166.6 |
| APB_0_CLK | 83.3 |
| APB_1_CLK | 83.3 |
| FIC_0_CLK | 166.6 |

Figure 13 shows the **clocks configuration dialog**.

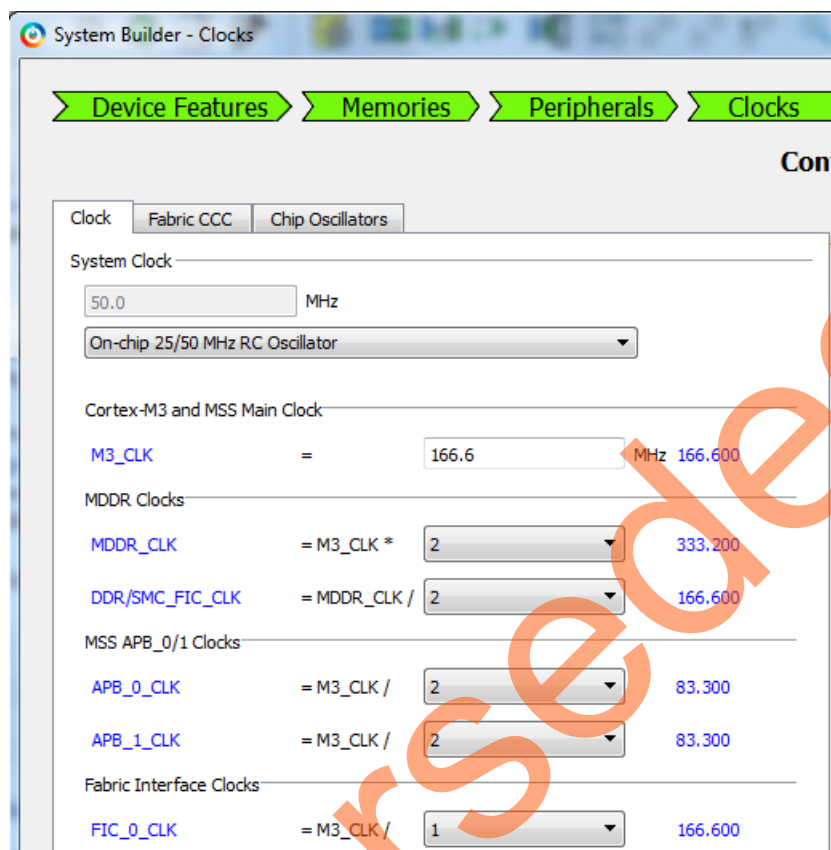


Figure 13 • System and Subsystem Clocks Configuration

10. Follow the rest of the steps with default settings and generate the design.
11. Instantiate the custom logic (AXI master, AHB master, and a command decoder) and make the connections as shown in Figure 7.

Simulation using Micron DDR3 SDRAM Model

Setting Up the Simulation Model

Setting up and running the simulation involve the following steps:

1. Obtain the Micron DDR3 memory model files - The SmartFusion2 Development Kit board has the DDR3 SDRAM from Micron with the part number; MT41J256M8HX-15E. The memory model used in the example design supports this device (Refer to "Appendix A – Design Files" on page 40).
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 Figure 14.

```

ddr3 DDR3_0 (
    .rst_n(MDDR_RESET_N),
    .ck(MDDR_CLK),
    .ck_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),
    .addr(MDDR_ADDR[14:0]),
    .ba(MDDR_BA),
    .dm_tdq(MDDR_DM_RDQS[0]),
    .dq(MDDR_DQ[7:0]),
    .dqs(MDDR_DQS[0]),
    .dqs_n(MDDR_DQS_N[0]),
    .odt(MDDR_ODT),
    .tdqs_n()
);

ddr3 DDR3_1 (
    .rst_n(MDDR_RESET_N),
    .ck(MDDR_CLK),
    .ck_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),
    .addr(MDDR_ADDR[14:0]),
    .ba(MDDR_BA),
    .dm_tdq(MDDR_DM_RDQS[1]),
    .dq(MDDR_DQ[15:8]),
    .dqs(MDDR_DQS[1]),
    .dqs_n(MDDR_DQS_N[1]),
    .odt(MDDR_ODT),
    .tdqs_n()
);

```

Figure 14 • Instantiating Simulation Model

4. Ensure that ddr3.v file is included at the top of the testbench file. The example design uses two instances of DDR3 models with the device width eight.

- Set the testbench in which DDR3 memory model is instantiated as active stimulus. Figure 15 shows the settings under Stimulus Hierarchy.

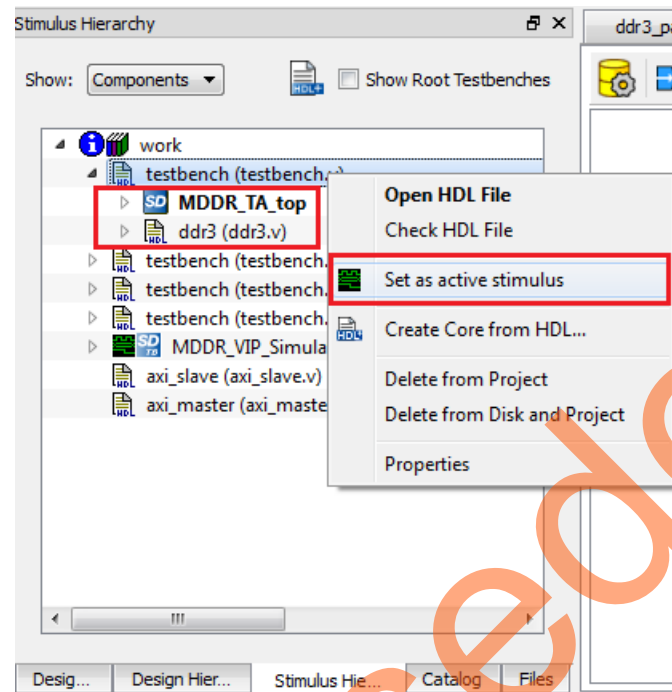


Figure 15 • Stimulus Settings

- Click **Project > Project Settings > Simulation Options > Waveforms**. Figure 16 shows the Waveforms settings on the right.

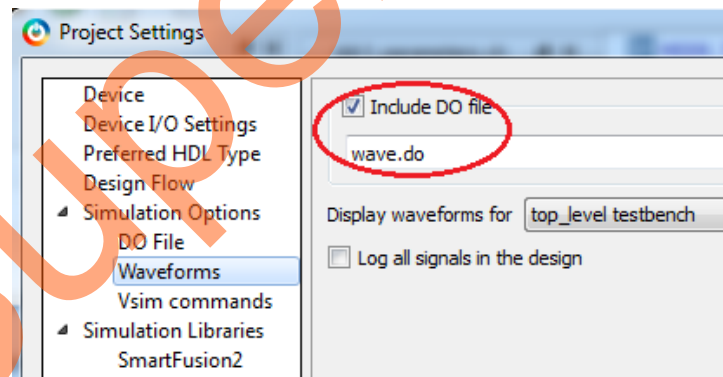


Figure 16 • Waveforms Settings

- Select the **Include DO file** check box and enter **wave.do** in the box as displayed in Figure 16.

The timing diagrams shown from Figure 17 through Figure 19 illustrate the write operation. Figure 17 illustrates the AXI master signals, command from CMD_Decoder, and address and data to the AHB master.

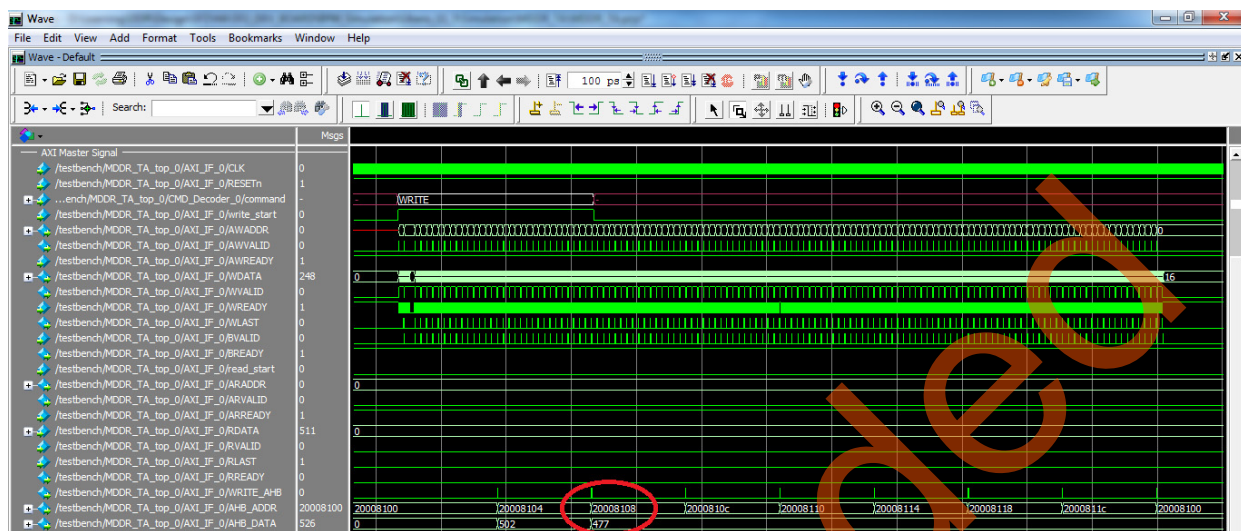


Figure 17 • AXI Master (AXI_IF) Signals for Write Operation

Figure 18 illustrates the MDDR signals. The AXI master reads 2 KB of data from LSRAM and writes to DDR3 SDRAM. The write operation is repeated eight times. The data is written into Row 0 of all banks (Bank 0 – Bank 7).

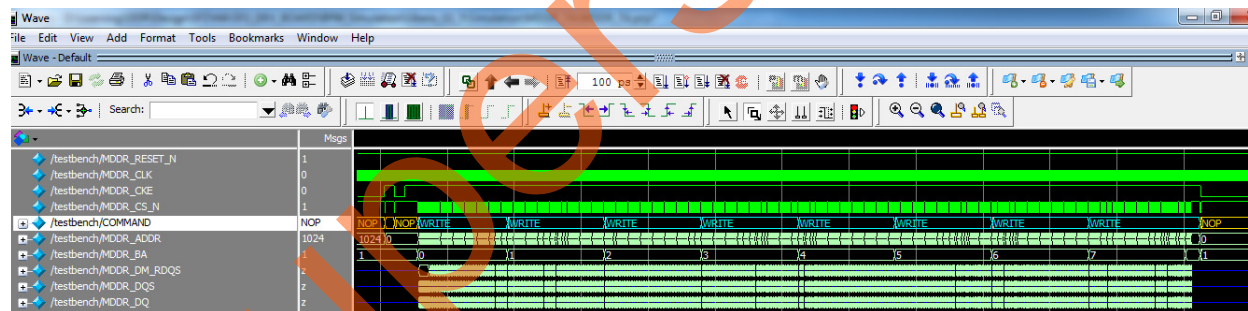


Figure 18 • MDDR Signals for Write Operation

Figure 21 illustrates the MDDR signals. The AXI master reads 2 KB of data from DDR3 SDRAM and writes to LSRAM. The read operation is repeated eight times. The data is read from Row 0 of all banks (Bank 0 – Bank 7).

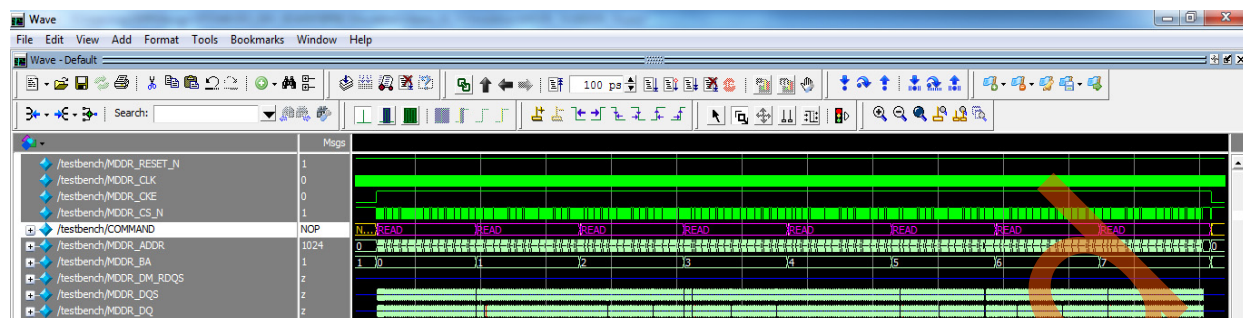


Figure 21 • MDDR Signals for Read Operation

Figure 22 illustrates the AHB master signals. The AHB master receives the address and data from AXI master and writes to eSRAM.

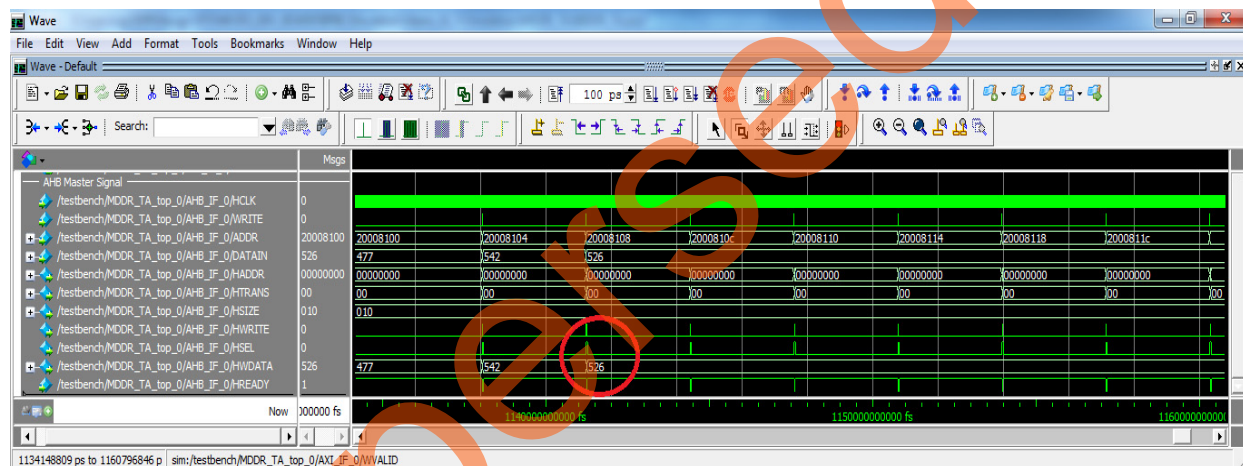


Figure 22 • AHB Master Signals

Simulation using Microsemi DDR3 SDRAM VIP Model

Libero SoC includes a generic DDR memory simulation model (VIP). This 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 Simulation Model

Setting up and running the simulation involves 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 to drag into the SmartDesign testbench canvas. Figure 23 shows the Simulation mode.

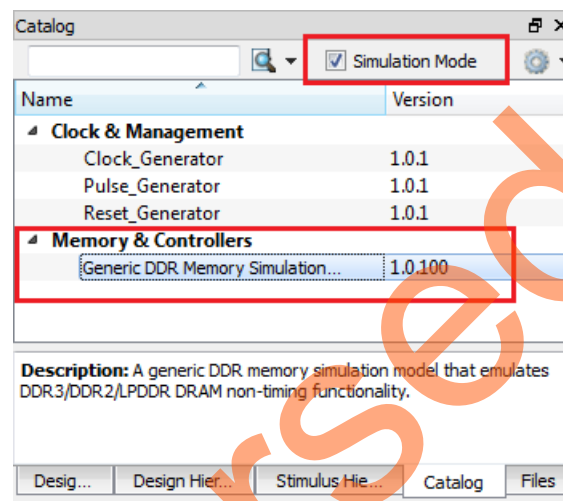


Figure 23 • Generic DDR Memory Simulation Model

4. Enter the Generic DDR Memory Simulation model configuration details as shown in [Figure 24](#). The example design uses two instances of SimDRAM (VIP model) with the device width size of eight.

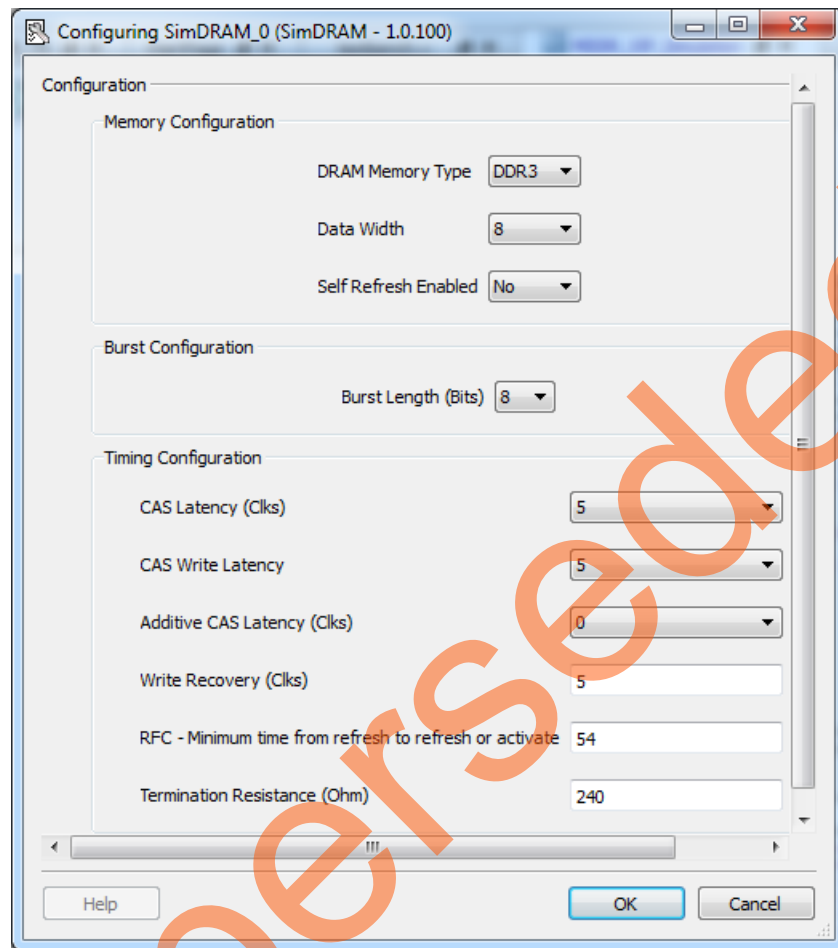


Figure 24 • Configuring SimDRAM

- Make the connections as described in "Simulation using Microsemi DDR3 SDRAM VIP Model" section on page 23. The connections are same as the Micron model. Figure 25 shows the SmartDesign testbench for the example design with Microsemi DDR3 SDRAM VIP model.

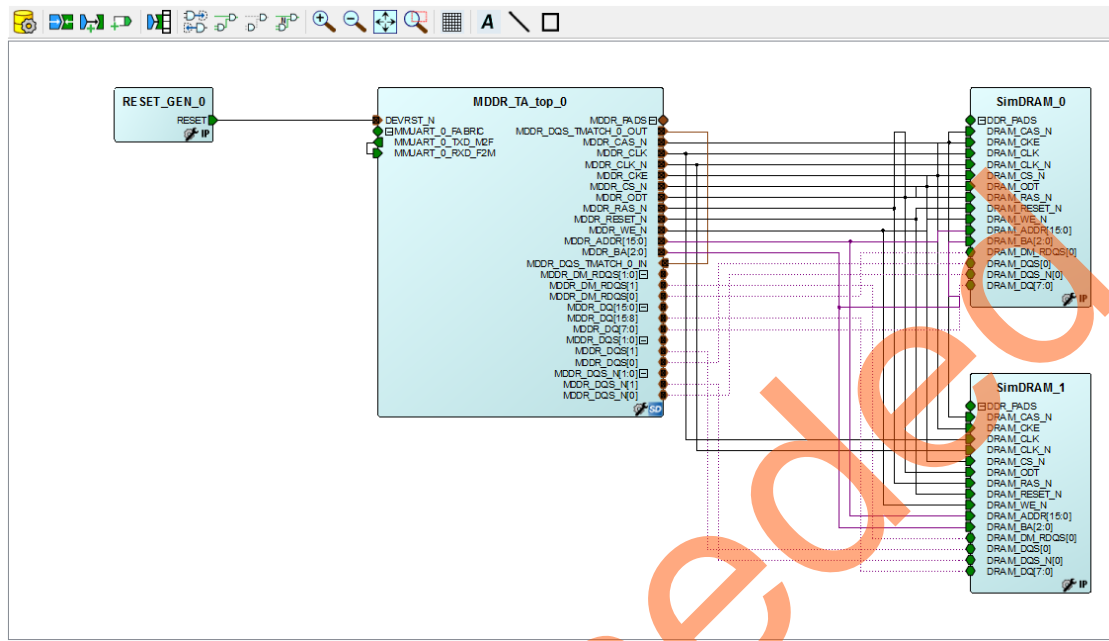


Figure 25 • SmartDesign Testbench for Example Design with Microsemi DDR3 SDRAM VIP

- Generate the design by clicking **SmartDesign > Generate Component** or by clicking **Generate Component** on the SmartDesign toolbar.
- Add the following code above **endmodule** in the generated SmartDesign testbench file, MDDR_VIP_Simulation.v.

```

wire [1:0] MDDR_DM_RDQS;
wire [15:0] MDDR_DQ;
wire [1:0] MDDR_DQS;
wire [2:0] COMMAND;
assign COMMAND =
{MDDR_TA_top_0 MDDR_RAS_N, MDDR_TA_top_0 MDDR_CAS_N, MDDR_TA_top_0 MDDR_WE_N};
assign MDDR_DM_RDQS = MDDR_DM_RDQS_net_0;
assign MDDR_DQ = MDDR_DQ_net_0;
assign MDDR_DQS = MDDR_DQS_net_0;
initial
begin
$display ("+++++");
$display ("Loading LSRAM from lsram.mem file");
$display ("");
$readmemh("lsram_512x64.mem", MDDR_VIP_Simulation.MDDR_TA_top_0.AXI_IF_0.Rdata_mem);
$display (" Completed Loading LSRAM");
$display ("+++++");
@(posedge MDDR_VIP_Simulation.MDDR_TA_top_0.AXI_IF_0.RESETn);
/* 2KB write */
repeat(9500) @(posedge MDDR_VIP_Simulation.MDDR_TA_top_0.AXI_IF_0.CLK);
force MDDR_VIP_Simulation.MDDR_TA_top_0.CMD_Decoder_0.command = 8'b001_001_01;
/* Disable Write */
repeat(15) @(posedge MDDR_VIP_Simulation.MDDR_TA_top_0.AXI_IF_0.CLK);
force MDDR_VIP_Simulation.MDDR_TA_top_0.CMD_Decoder_0.command = 8'b000_000_00;
/* 2KB Read */
repeat(5000) @(posedge MDDR_VIP_Simulation.MDDR_TA_top_0.AXI_IF_0.CLK);
force MDDR_VIP_Simulation.MDDR_TA_top_0.CMD_Decoder_0.command = 8'b001_001_10;

```

```
/* Disable Read */
repeat(15) @(posedge MDDR_VIP_Simulation.MDDR_TA_top_0.AXI_IF_0.CLK);
force MDDR_VIP_Simulation.MDDR_TA_top_0.CMD_Decoder_0.command = 8'b000_000_00;
end
```

Figure 26 shows the SmartDesign generated testbench file under **Files** tab.

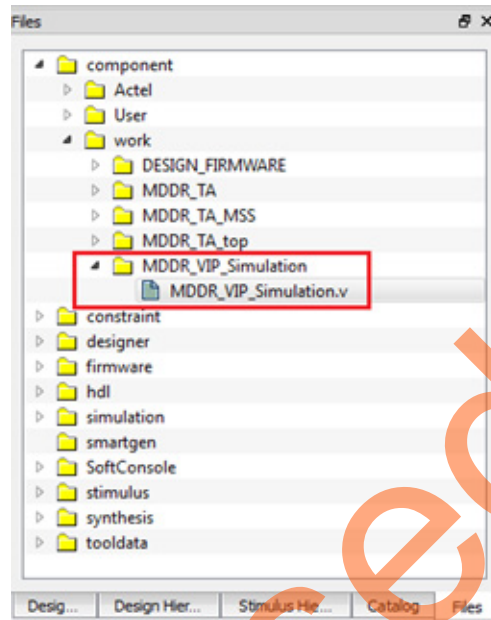


Figure 26 • SmartDesign Generated Testbench File

8. Under the **Stimulus Hierarchy** tab, set the SmartDesign testbench as **Set as active stimulus**.
Figure 27 shows the Stimulus Hierarchy settings.

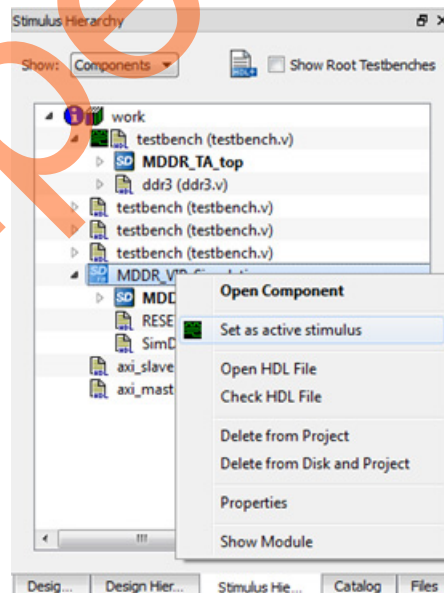


Figure 27 • Stimulus Settings

- Change the default DO file name to **wave_vip.do** file in **Project > Project Settings > Simulation Options > Waveforms**. Figure 28 shows the Waveforms settings.

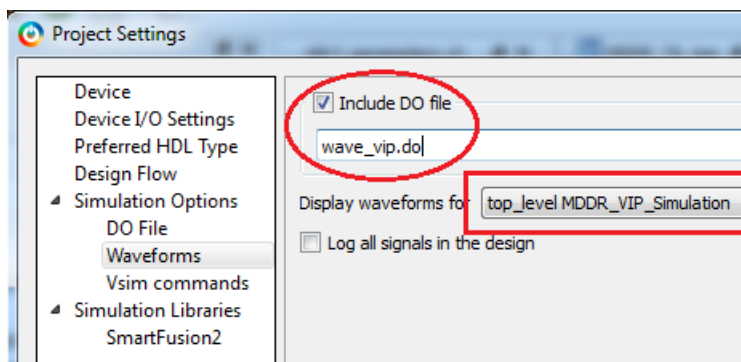


Figure 28 • Waveforms Settings

The timing diagrams shown from Figure 29 through Figure 31 on page 28 illustrate the write operation. Figure 29 illustrates the AXI master signals, command from CMD_Decoder, and address and data to AHB master.

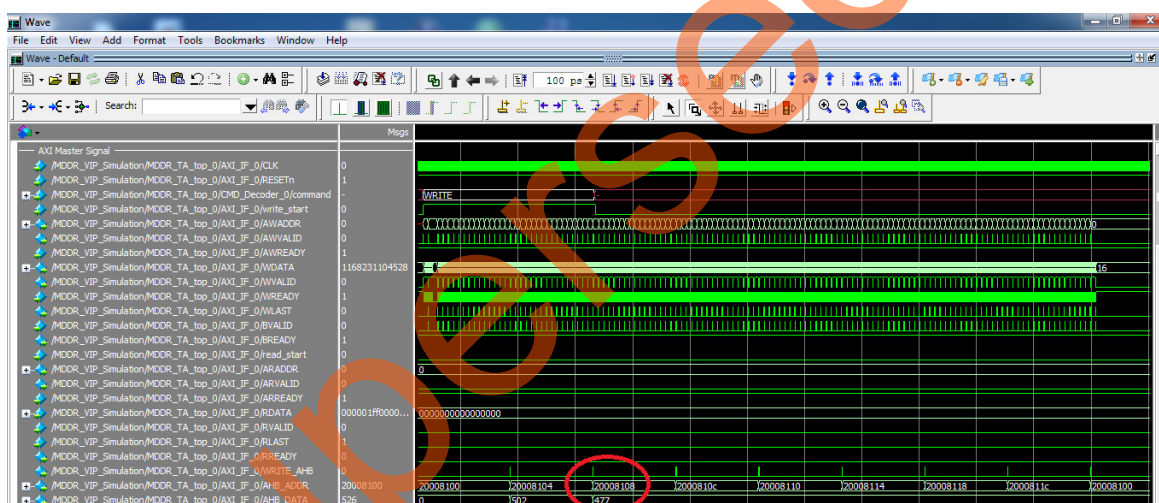


Figure 29 • AXI Master (AXI_IF) Signals for Write Operation

Figure 30 illustrates the MDDR subsystem signals. The AXI master reads 2 KB of data from LSRAM and writes into DDR3 SDRAM. The write operation is repeated eight times. The data is written into Row 0 of all banks (Bank 0 – Bank 7).

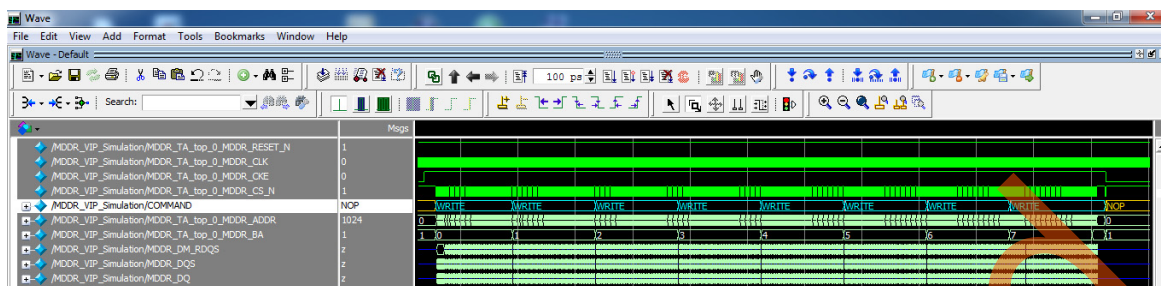


Figure 30 • MDDR Signals for Write Operation

Figure 31 illustrates the AHB master signals. The AHB master receives address and data from AXI master and writes into eSRAM.

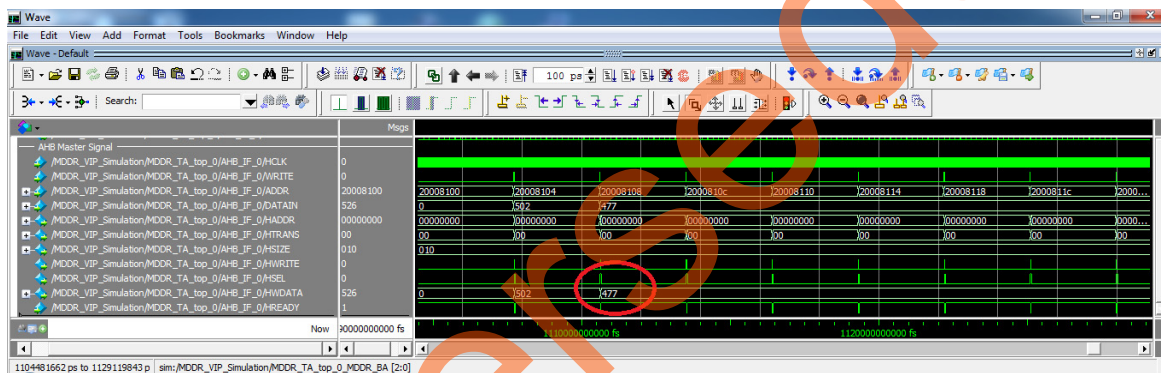


Figure 31 • AHB Master Signals

The timing diagrams shown from Figure 32 through Figure 34 on page 30 illustrate the read operation. Figure 32 illustrates the AXI master signals, command from CMD_Decoder, and address and data to AHB master.

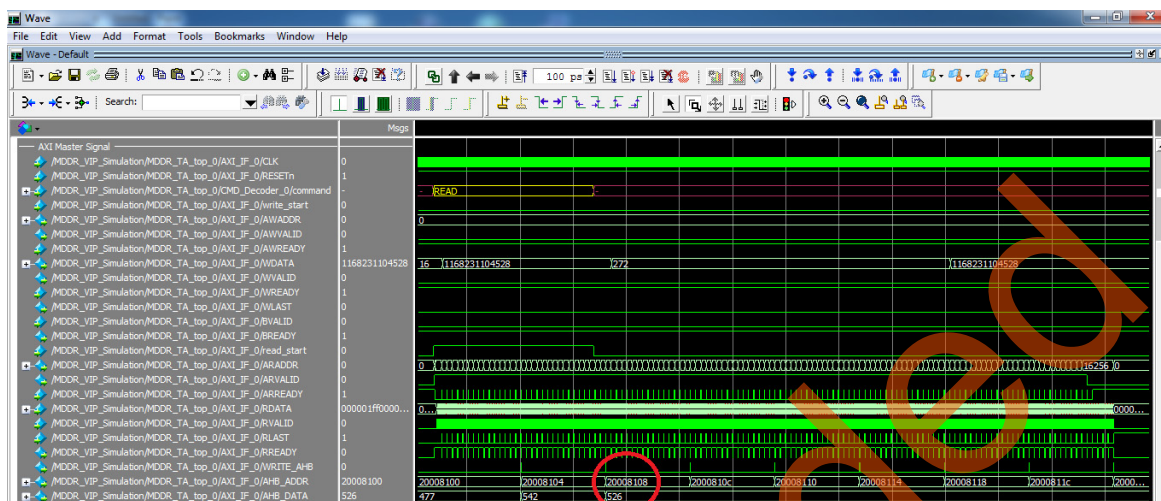


Figure 32 • AXI Master (AXI_IF) Signals for Read Operation

Figure 33 illustrates the MDDR signals. The AXI master reads 2 KB of data from DDR3 SDRAM and writes into LSRAM. The read operation is repeated eight times. The data is red from Row 0 of all banks (Bank 0 – Bank 7).

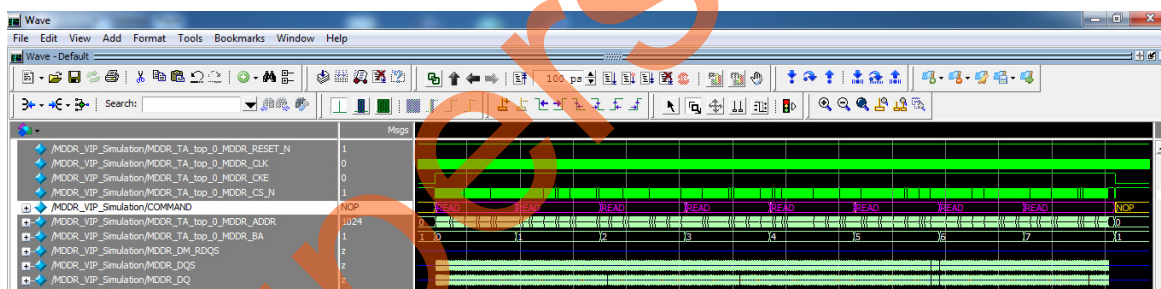


Figure 33 • MDDR Signals for Read Operation

Figure 34 illustrates the AHB master signals. The AHB master receives address and data from AXI master and writes into eSRAM.

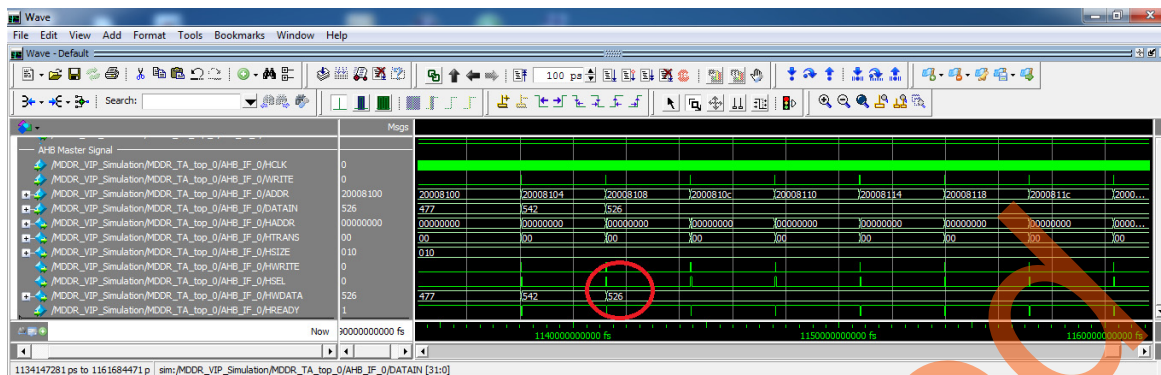


Figure 34 • AHB Master Signal

Software Implementation

The software design example performs the following operations:

- Initializing and configuring the MMUART_0 with 115200 baud rate, 8 data bits, 1 stop bit, no parity, and no flow control. This is done by adding MICROSEMI_STDIO_THRU_MMUART0 symbol in the project settings as shown in Figure 35.

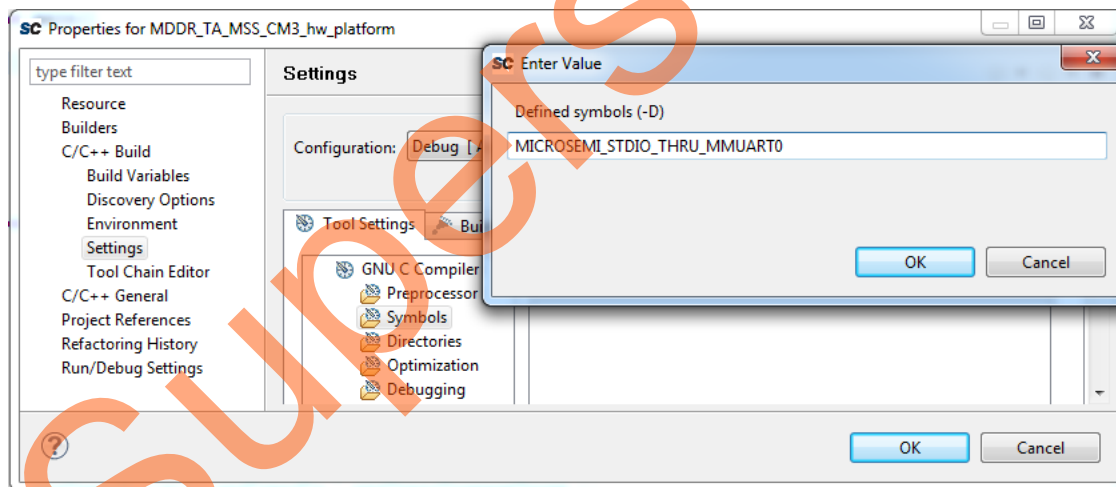


Figure 35 • MICROSEMI_STDIO_THRU_MMUART0 Symbol Settings

- Initializing and configuring the GPIOs (MSS_GPIO_0 to MSS_GPIO_7 are configured in the output mode).
- Initializing the DDR3 SDRAM:
 - 16777216x4 locations, starting from address 0xA0000000, are filled with zeros.
 - 8x1024x4 locations, starting from address 0xA1000000, are filled with incremental patterns.
- Initializing the eSRAM:
 - 8x4 locations, starting from address 0x20008104, are filled with zeros.
- Performing the data integrity checks.
- Sending a command to the AXI master for reading operation through GPIOs.

- Sending a command to the AXI master for writing operation through GPIOs.

List of firmware drivers used in this application:

- SmartFusion2 MSS GPIO driver
- SmartFusion2 MSS MMUART driver:
 - To communicate with the serial terminal program running on host PC

In this design example, the application software performs the following steps:

1. Performing the following data integrity checks:
 - a. The Cortex-M3 processor initializes the 8x1024x4 (8 repetitions x 1024 locations x 4 bytes) locations of DDR3 SDRAM, starting from address 0xA1000000, with incremental patterns. The pattern increments from 0 to 1023, and is repeated eight times.
 - b. The AXI master reads 4 KB of data from DDR3 SDRAM, starting from the address 0x01000000, that is 0xA1000000¹, and writes into LSRAM. The read operation is repeated eight times. The last 4 KB of data is fetched from the address 0x01007000, that is 0xA1007000¹.
 - c. The AXI master reads 4 KB of data from LSRAM (512x64) and writes into DDR3 SDRAM, starting from address 0x00000000, that is, 0xA0000000¹. The write operation is repeated eight times. The last 4 KB of data is written at the address 0x00007000, that is 0xA0007000.
 - d. The Cortex-M3 processor compares the 4 KB data at address 0xA0007000 and 0xA1007000. The status is printed on HyperTerminal with error count, if any.

Note: The address map to access the DDR memory from MSS masters through MDDR is 0xA0000000-0xDFFFFFFF.

2. Initializing the DDR3 SDRAM again.
3. Perform the read operation. Uncomment any of the following lines based on the size of data to be read from DDR3 SDRAM. The default size is 2 KB.

```
/* DDR3 SDRAM READ OPERATION */

//MSS_GPIO_set_outputs (0x26); // 2KB
//MSS_GPIO_set_outputs (0x4A); // 4KB
//MSS_GPIO_set_outputs (0x6E); // 8KB
//MSS_GPIO_set_outputs (0x92); // 16KB
```

4. Printing the read throughput values on HyperTerminal.
5. Performing the write operation. Uncomment any of the following lines based on the size of data to be written into DDR3 SDRAM. The default size is 2 KB.

```
/* DDR3 SDRAM WRITE OPERATION */

//MSS_GPIO_set_outputs (0x25); // 2KB
//MSS_GPIO_set_outputs (0x49); // 4KB
//MSS_GPIO_set_outputs (0x6D); // 8KB
//MSS_GPIO_set_outputs (0x91); // 16KB
```

6. Printing the write throughput values on HyperTerminal.

Running the Design

The design example is designed to run on the SmartFusion2 Development Kit board. Refer to <http://www.microsemi.com/products/fpga-soc/design-resources/dev-kits/smartfusion2/smartfusion2-development-kit> for more detailed board information.

Board Jumper Settings

Table 5 lists the jumpers that need to be connected on SmartFusion2 Development Kit board.

Table 5 • SmartFusion2 SoC FPGA Development Kit Jumper Settings

| Jumper | Pin (From) | Pin (To) | Comments |
|---|------------|----------|----------|
| J70, J93, J94, J117, J123, J142, J157, J160, J167, J225, J226, J227 | 1 | 2 | Default |
| J23 | 2 | 3 | Default |
| J129, J133 | 2 | 3 | Default |

Note: Press **SW7** power switch on the board to OFF position while providing the jumper connections.

Host PC to Board Connections

1. Connect the FlashPro4 programmer to the FP4 HEADER J59 connector of the SmartFusion2 Development Kit board.
2. Connect one end of the USB mini-B (FTDI interface) cable to the J24 connector provided on the SmartFusion2 Development Kit board and connect the other end of the USB cable to the host PC.

USB Driver Installation

Install the FTDI D2XX driver for serial terminal communication through FTDI mini USB cable. The drivers and installation guide can be downloaded from

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

Ensure that the USB to UART bridge drivers are detected (can be verified in Device Manager in the system), as shown in Figure 36 on page 33.

Note: Copy the COM port number for serial port configuration. Ensure that the COM port **Location** is specified as **on USB Serial Converter D**, as shown in [Figure 36](#).

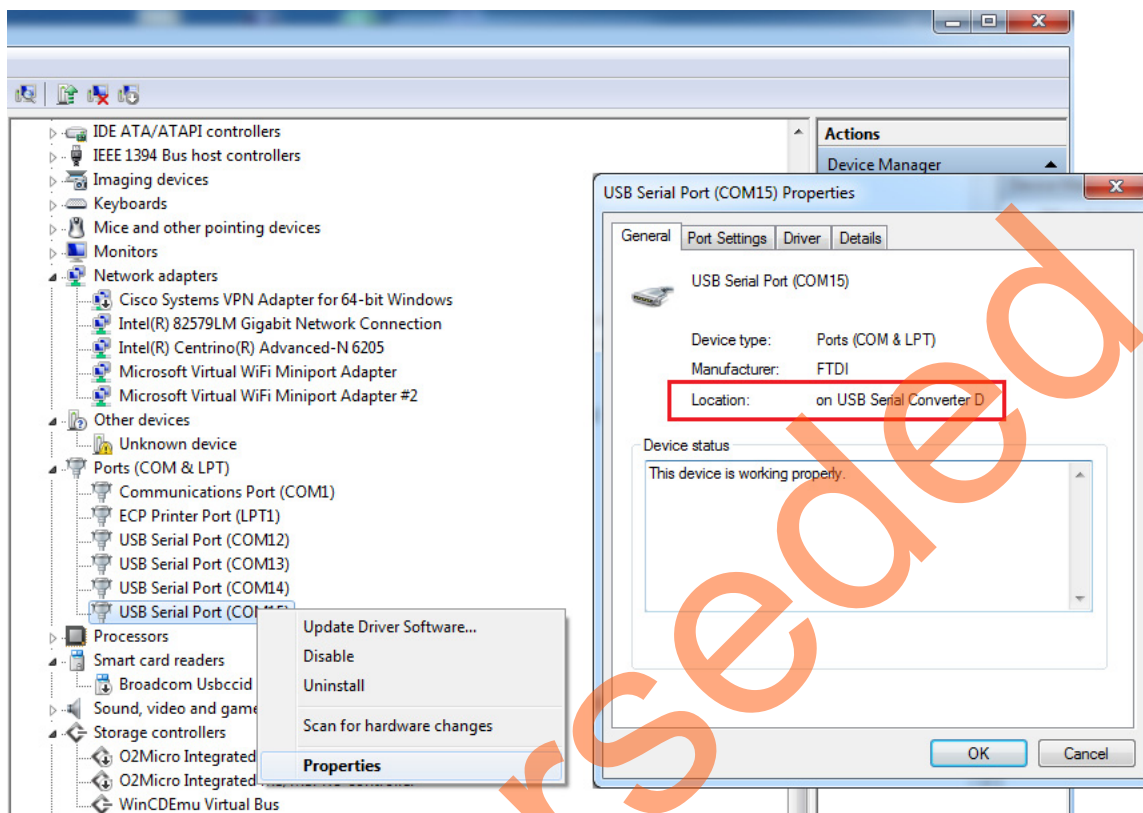
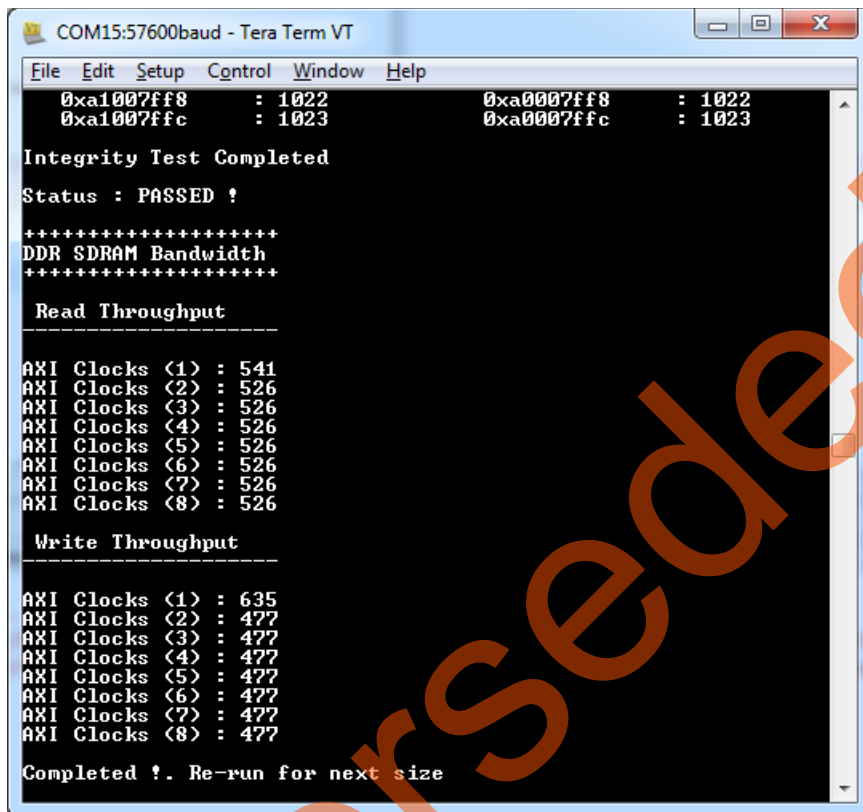


Figure 36 • USB to UART Bridge Drivers

Steps to Run the Design

1. Connect the power supply to the J18 connector and FlashPro programmer.
2. Press the **SW7** power supply switch to ON.
3. Program the SmartFusion2 Development Kit board with the generated or provided *.stp file (Refer to "Appendix A – Design Files" on page 40) using FlashPro.
4. Invoke the SoftConsole3.4 integrated design environment (IDE) and launch the debugger.
5. Start the HyperTerminal program with the baud rate set to 57600, 8 data bits, 1 stop bit, no parity, and no flow control. If the PC does not have HyperTerminal, use any free serial terminal emulation program, such as PuTTY or Tera Term. Refer to the [Configuring Serial Terminal Emulation Programs Tutorial](#) for configuring HyperTerminal, Tera Term, and PuTTY.

When the debugger runs in SoftConsole, HyperTerminal window is displayed with the data integrity check-status followed by the read and write throughputs. [Figure 37](#) shows the total number of AXI clocks taken for 2 KB of data transferred from LSRAM to DDR3 SDRAM and DDR3 SDRAM to LSRAM.



```

COM15:57600baud - Tera Term VT
File Edit Setup Control Window Help
0xa1007ff8 : 1022      0xa0007ff8 : 1022
0xa1007ffc : 1023      0xa0007ffc : 1023

Integrity Test Completed
Status : PASSED ?

+++++
DDR SDRAM Bandwidth
+++++

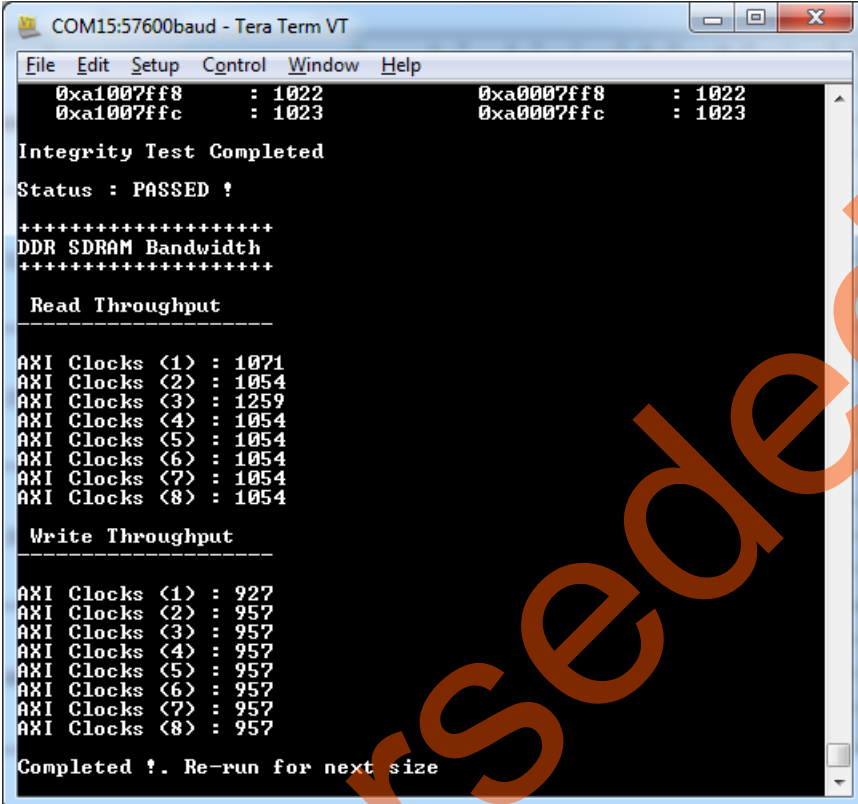
Read Throughput
-----
AXI Clocks <1> : 541
AXI Clocks <2> : 526
AXI Clocks <3> : 526
AXI Clocks <4> : 526
AXI Clocks <5> : 526
AXI Clocks <6> : 526
AXI Clocks <7> : 526
AXI Clocks <8> : 526

Write Throughput
-----
AXI Clocks <1> : 635
AXI Clocks <2> : 477
AXI Clocks <3> : 477
AXI Clocks <4> : 477
AXI Clocks <5> : 477
AXI Clocks <6> : 477
AXI Clocks <7> : 477
AXI Clocks <8> : 477

Completed !. Re-run for next size
  
```

Figure 37 • Throughput for 2 KB Data

Figure 38 shows the total number of AXI clocks taken for 4 KB of data transferred from LSRAM to DDR3 SDRAM and DDR3 to LSRAM.



```

COM15:57600baud - Tera Term VT
File Edit Setup Control Window Help
0xa1007ff8 : 1022      0xa0007ff8 : 1022
0xa1007ffc : 1023      0xa0007ffc : 1023

Integrity Test Completed
Status : PASSED !

*****
DDR SDRAM Bandwidth
*****

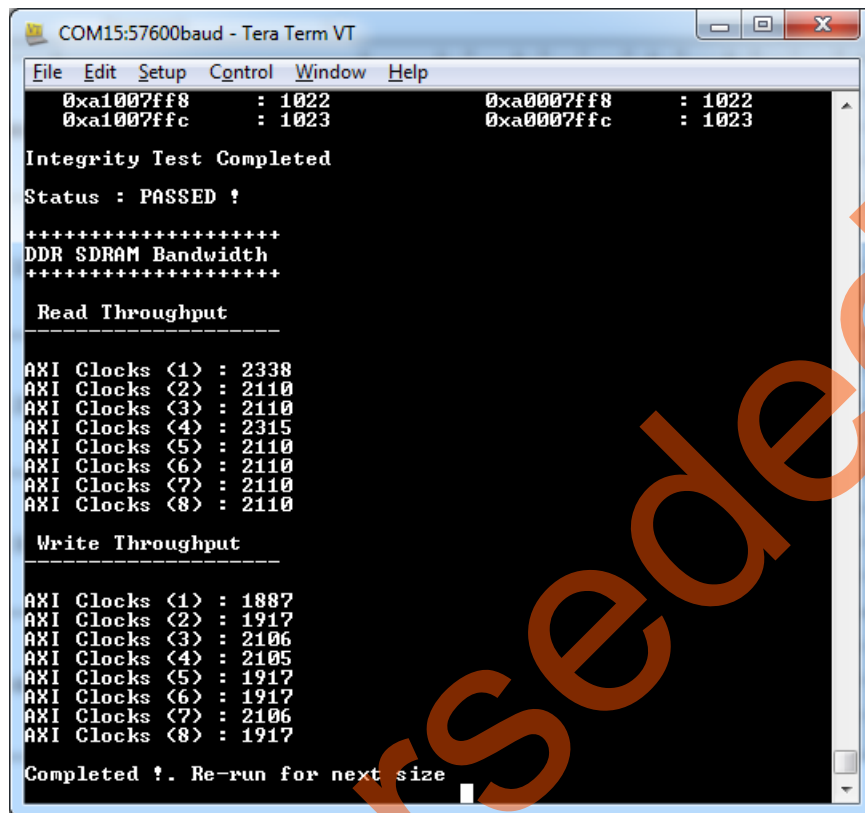
Read Throughput
-----
AXI Clocks <1> : 1071
AXI Clocks <2> : 1054
AXI Clocks <3> : 1259
AXI Clocks <4> : 1054
AXI Clocks <5> : 1054
AXI Clocks <6> : 1054
AXI Clocks <7> : 1054
AXI Clocks <8> : 1054

Write Throughput
-----
AXI Clocks <1> : 927
AXI Clocks <2> : 957
AXI Clocks <3> : 957
AXI Clocks <4> : 957
AXI Clocks <5> : 957
AXI Clocks <6> : 957
AXI Clocks <7> : 957
AXI Clocks <8> : 957

Completed !. Re-run for next size
  
```

Figure 38 • Throughput for 4 KB Data

Figure 39 shows the total number of AXI clocks taken for 8 KB of data transferred from LSRAM to DDR3 SDRAM and vice-versa.



```

COM15:57600baud - Tera Term VT
File Edit Setup Control Window Help
0xa1007ff8 : 1022 0xa0007ff8 : 1022
0xa1007ffc : 1023 0xa0007ffc : 1023

Integrity Test Completed
Status : PASSED !

*****
DDR SDRAM Bandwidth
*****

Read Throughput
-----
AXI Clocks <1> : 2338
AXI Clocks <2> : 2110
AXI Clocks <3> : 2110
AXI Clocks <4> : 2315
AXI Clocks <5> : 2110
AXI Clocks <6> : 2110
AXI Clocks <7> : 2110
AXI Clocks <8> : 2110

Write Throughput
-----
AXI Clocks <1> : 1887
AXI Clocks <2> : 1917
AXI Clocks <3> : 2106
AXI Clocks <4> : 2105
AXI Clocks <5> : 1917
AXI Clocks <6> : 1917
AXI Clocks <7> : 2106
AXI Clocks <8> : 1917

Completed !. Re-run for next size
  
```

Figure 39 • Throughput for 8 KB Data

Figure 40 shows the total number of AXI clocks taken for 16 KB of data transferred from LSRAM to DDR3 SDRAM and vice-versa.

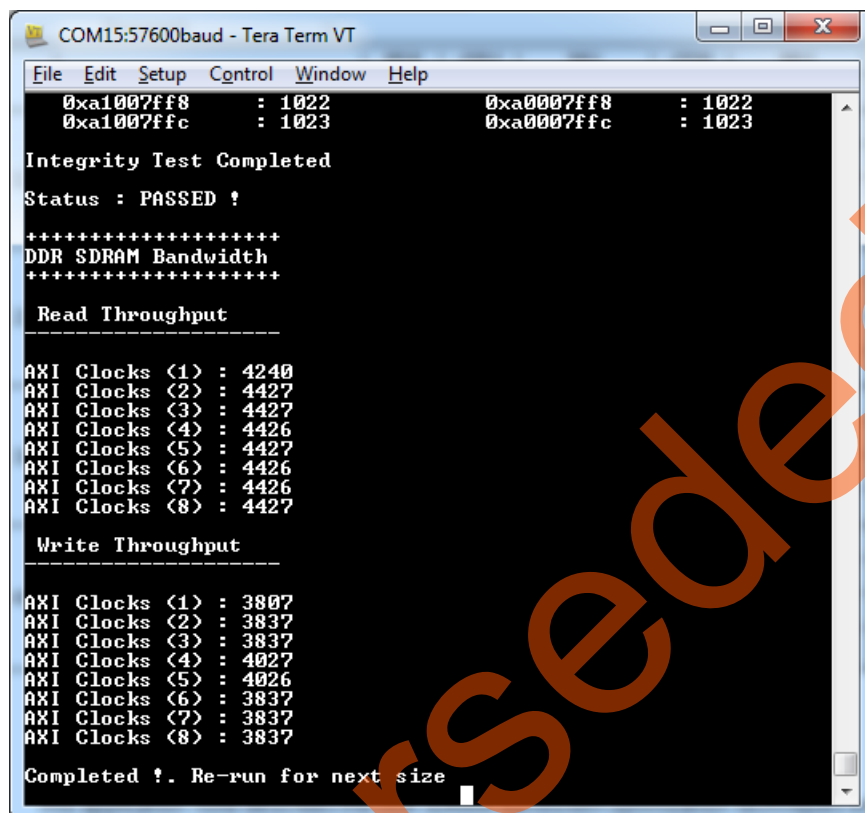


Figure 40 • Throughput for 16 KB Data

DDR3 SDRAM Bandwidth

Table 6 provides the total number of 16 beat bursts corresponding to the write or read size.

Table 6 • 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:

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

Simulation Result

Table 7 lists the write and read bandwidth of DDR3 SDRAM simulation. The incremental pattern of size varies from 2 KB to 16 KB, which is transferred from LSRAM to DDR3 SDRAM and vice-versa.

Table 7 • DDR3 SDRAM Bandwidth

| SI No | Optimization Techniques | Size | Write | | Read | | Write Improvement | Read Improvement |
|-------|--|-------|-------------|--------------------|-------------|--------------------|-------------------|-------------------|
| | | | No of cycle | Bandwidth (MB/Sec) | No of cycle | Bandwidth (MB/Sec) | | |
| Base | 160 MHz | 2 KB | 529 | 619 | 737 | 444 | avg: 630 | avg:440 |
| | | 4 KB | 1041 | 629 | 1476 | 444 | | |
| | | 8 KB | 2065 | 634 | 2954 | 443 | | |
| | | 16 KB | 4113 | 637 | 6233 | 420 | | |
| 1 | 1. 166 MHz | 2 KB | 529 | 642 | 737 | 461 | avg: 650 4.7% | avg: 460 4.5% |
| | | 4 KB | 1041 | 653 | 1476 | 460 | | |
| | | 8 KB | 2065 | 658 | 2954 | 460 | | |
| | | 16 KB | 4401 | 617 | 6236 | 436 | | |
| 2 | 1. 166 MHz 2. Without Write Response State | 2 KB | 501 | 678 | 737 | 461 | avg: 680 7.9% | avg: 460 4.5% |
| | | 4 KB | 981 | 693 | 1476 | 460 | | |
| | | 8 KB | 1941 | 700 | 2954 | 460 | | |
| | | 16 KB | 4151 | 655 | 6236 | 436 | | |
| 3 | 1. 166 MHz 2. Without Write Response State 3. Tuned DDR Configuration | 2 KB | 502 | 677 | 721 | 471 | avg: 700 11% | avg: 470 6.8% |
| | | 4 KB | 982 | 692 | 1444 | 470 | | |
| | | 8 KB | 1942 | 700 | 2890 | 470 | | |
| | | 16 KB | 3862 | 704 | 5788 | 470 | | |
| 4 | 1. 166 MHz 2. Without Write Response State 3. Tuned DDR Configuration 4. Read Command Queuing | 2 KB | 477 | 712 | 526 | 646 | avg: 710 12% | avg: 645 46.6% |
| | | 4 KB | 957 | 710 | 1054 | 645 | | |
| | | 8 KB | 1917 | 709 | 2110 | 644 | | |
| | | 16 KB | 3837 | 708 | 4222 | 644 | | |

Board Test Result

Table 8 lists the write and read bandwidth of DDR3 SDRAM on SmartFusion2 Development Kit board. The incremental pattern of size varies from 2 KB to 16 KB, which is transferred from LSRAM to DDR3 SDRAM and vice-versa.

Table 8 • DDR3 SDRAM Bandwidth

| SI No | Optimization Techniques | Size | Write | | Read | | Write Improvement | Read Improvement |
|-------|--|-------|-------------|--------------------|-------------|--------------------|-------------------|------------------|
| | | | No of Cycle | Bandwidth (MB/Sec) | No of Cycle | Bandwidth (MB/Sec) | | |
| Base | 160 MHz | 2 KB | 507 | 646 | 736 | 445 | avg: 640 | avg: 440 |
| | | 4 KB | 1019 | 643 | 1475 | 444 | | |
| | | 8 KB | 2043 | 641 | 2956 | 443 | | |
| | | 16 KB | 4091 | 640 | 5915 | 443 | | |
| 1 | 1. 166.6 MHz | 2 KB | 507 | 672 | 736 | 463 | avg: 670 4.7% | avg: 460 4.5% |
| | | 4 KB | 1019 | 669 | 1475 | 462 | | |
| | | 8 KB | 2043 | 668 | 2956 | 461 | | |
| | | 16 KB | 4091 | 667 | 5912 | 461 | | |
| 2 | 1. 166.6 MHz 2. Without Write Response State | 2 KB | 492 | 693 | 736 | 463 | avg: 690 7.8% | avg: 460 4.5% |
| | | 4 KB | 957 | 713 | 1475 | 462 | | |
| | | 8 KB | 1980 | 689 | 2953 | 462 | | |
| | | 16 KB | 3963 | 688 | 5912 | 461 | | |
| 3 | 1. 166.6 MHz 2. Without Write Response State 3. Tuned DDR Configuration | 2 KB | 477 | 715 | 720 | 473 | avg: 710 11% | avg: 470 6.8% |
| | | 4 KB | 957 | 713 | 1443 | 472 | | |
| | | 8 KB | 1980 | 689 | 2889 | 472 | | |
| | | 16 KB | 3837 | 711 | 5799 | 470 | | |
| 4 | 1. 166.6 MHz 2. Without Write Response State 3. Tuned DDR Configuration 4. Read Command Queuing | 2 KB | 477 | 715 | 526 | 648 | avg: 710 11% | avg: 640 45% |
| | | 4 KB | 957 | 713 | 1054 | 647 | | |
| | | 8 KB | 1917 | 711 | 2110 | 646 | | |
| | | 16 KB | 3837 | 711 | 4427 | 616 | | |

Conclusion

This application note describes the DDR SDRAM bandwidth optimization techniques with an example design on SmartFusion2 Development Kit board. It also shows the DDR SDRAM simulation flow using the Micron DDR3 SDRAM model and Microsemi DDR3 SDRAM VIP model.

Appendix A – Design Files

The design files can be downloaded from the Microsemi SoC Products Group website:

http://soc.microsemi.com/download/rsc/?f=M2S_AC422_11p4_DF

The design file consists of Libero SoC Verilog project, SoftConsole software project, MDDR Configuration files, Simulation model files and programming files (*.stp) for SmartFusion2 Development Kit board. Refer to the Readme.txt file included in the design file for the directory structure and description.

Superseded

List of Changes

The following table lists the critical changes that are made in the current version:

| Date | Changes | Page |
|------------------------------|---|------|
| Revision 3 (August, 2014) | Rearranged a few sections. No change in content. | NA |
| Revision 2 (August, 2014) | Updated the document for Libero SoC v11.4 software release (SAR 59944). | NA |
| Revision 1 (June, 2014) | Initial release. | NA |

Superseded

Superseded



Microsemi®

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 and 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; Power-over-Ethernet ICs and midspans; as well as custom design capabilities and services. Microsemi is headquartered in Aliso Viejo, Calif. and has approximately 3,400 employees globally. Learn more at www.microsemi.com.

© 2014 Microsemi Corporation. All rights reserved. Microsemi and the Microsemi logo are trademarks of Microsemi Corporation. All other trademarks and service marks are the property of their respective owners.