

SmartFusion2 - Optimizing DDR Controller for Improved Efficiency - Libero SoC v11.7

Table of Contents

Purpose
Introduction
References
Design Requirements
Optimization Techniques
Frequency of Operation
Burst Length
AXI Master without Write Response State
Read Address Queuing
Series of Writes or Reads
DDR Configuration Tuning
Implementation on the SmartFusion2 Device
Design Description
Hardware Implementation
Configuring the System Builder
Simulation Using Micron DDR3 SDRAM Model
Simulation using Microsemi DDR3 SDRAM VIP Model
Software Implementation
Running the Design
Board Jumper Settings
Host PC to Board Connections
USB Driver Installation
Steps to Run the Design
DDR3 SDRAM Bandwidth
Conclusion
Appendix: Design Files
List of Changes

Purpose

This application note describes the techniques for improving the efficiency of double data rate (DDR) controller using an example design for the SmartFusion[®]2 Advanced Development Kit board. It also provides details about implementing the DDR synchronous dynamic random access memory (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 application-specific integrated circuit (ASIC) memory controllers such as microcontroller subsystem (MSS) DDR (MDDR) and fabric DDR (FDDR) for interfacing with the DDR2, DDR3, and low power DDR1 (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 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.

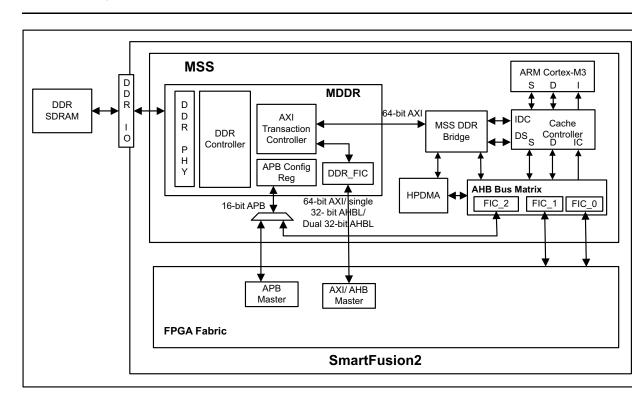


Figure 1 shows the MDDR data path for AXI/AHB interface.

Figure 1 • MDDR Data Path for AXI/AHB Interfaces

The AXI interface is 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. For more information on MDDR and FDDR subsystems, see the *UG0446: SmartFusion2 and IGLOO2 FPGA High Speed DDR Interfaces User Guide*.

The sample design consists 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 universal asynchronous receiver/transmitter (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. 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 the DDR2, DDR3, and LPDDR SDRAM memories and used 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. Ensure 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 are the references:

- UG0446: SmartFusion2 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 Application Note
- AC333: Connecting User Logic to the SmartFusion Microcontroller Subsystem Application Note
- DDR Controller and Serial High Speed Controller Initialization Methodology
- UG0557: SmartFusion2 SoC FPGA Advanced Development Kit User Guide

Design Requirements

Table 1 lists the design requirements.

Table 1 • Design Requirements

Design Requirements	Description
Hardware Requirements	
SmartFusion2 Advanced Development Kit	Rev B or later
Host PC or Laptop	Any 64-bit Windows Operating System
Software Requirements	
Libero SoC	v11.7
SoftConsole	v3.4 SP1*
Note: *For this application note, SoftConsole v3.4 S. SoftConsole v4.0 and Libero SoC v11.7 Tutorial.	P1 is used. For using SoftConsole v4.0, see the TU0546:

SmartFusion2 - Optimizing DDR Controller for Improved Efficiency - Libero SoC v11.7



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 inside the embedded block. The divider ratios can be selected 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 an FPGA fabric interface and is 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 subsystems 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, however, setting the maximum supported burst length for DDR SDRAM and AXI interface achieves 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 Advanced Development Kit board, which has the SmartFusion2 M2S150 device and a DDR3 SDRAM from Micron with the part number; MT41K256M8DA -125. Both the devices support the maximum burst length of 8.

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 must wait 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 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. Microsemi recommends 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 for handling the termination of write burst transaction. This must be followed if the write response channel returns an incorrect value.



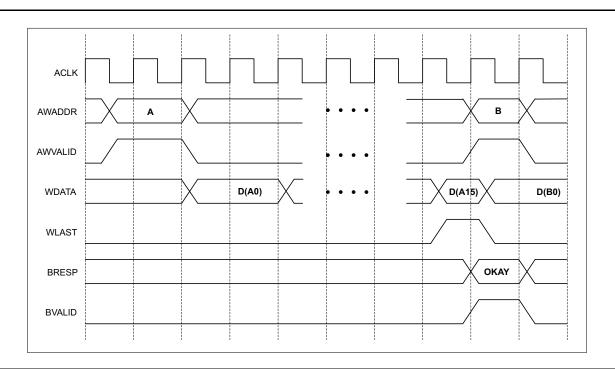


Figure 2 shows the write transaction timing diagram without the write response state.

Figure 2 • Write Transaction Timing Diagram without Write Response State

Read Address Queuing

The MDDR and FDDR subsystems support up to 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 five burst read addresses.

 ACLK
 ACLK

 ARADDR
 A1

 A2
 A3

 A4
 A5

 A6

 ARVALID

 ARREADY

 ARID

 0

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 7 on page 34 and Table 8 on page 35 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 reminder 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 depend 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 increase while performing a series of reads or writes from the same bank and row. Figure 4 shows the AXI to DDR3 address mapping for the DDR3 SDRAM on the SmartFusion2 Advanced Development Kit board.

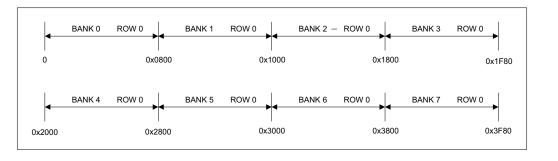


Figure 4 • AXI to DDR3 Address Mapping

When the AXI address crosses 0x0800, the DDR subsystem activates Row 0 of Bank 1. Row 1 of Bank 0 is activated only when the AXI address crosses 0x4000. If a new row is accessed every time, it must be pre-charged first. This means that additional time is 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 timing parameters required for the proper operation in terms of time units. These timings must match 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 Advanced 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 parameters for better performance than the values in the default configuration file.

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

Table 2 • Tuned DDR Timing Parameters

Implementation on the SmartFusion2 Device

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

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



Design Description

The design consists 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.

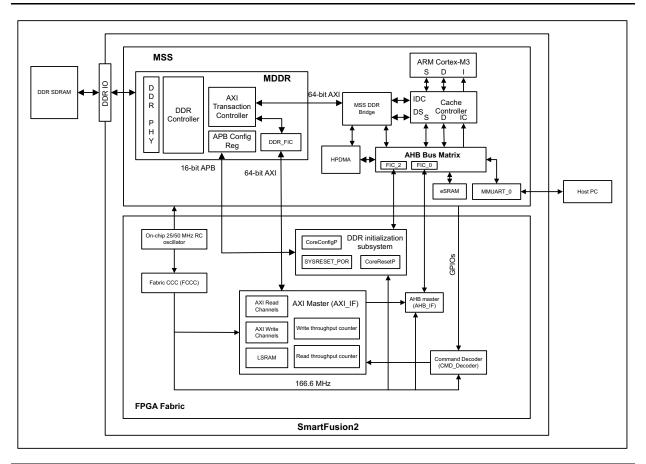


Figure 5 • Top-Level Block Diagram of the Design

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 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 routes 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 shows the command decoding.

		F	Read	l Siz	e		V	Vrite	Siz	е		W	/R	
GPIOs	-	7	6	6	5	2	1	3	3	2		I	0	
			1					1						
	0	0	0		0						0	0	NOP)
	0	0	1	2	KB						0	1	Write)
	0	1	0	4	KB						1	0	Read	ł
	0	1	1	8	KB						1	1	NOP)
	1	0	0	16	KB									
	1	0	1	32	KB	0	0	0		0				
						0	0	1	2	KB				
						0	1	0	4	KB				
						0	1	1	8	KB				
						1	0	0	16	KB				
						1	0	1	32	KB				

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 operation¹ based on the input signals from the command decoder. During the write operation, AXI master reads the LSRAM and writes into the DDR3 memory, and then measures the write throughput. During the read operation, 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 0x0000000, and is incremented by 128 (16-beat burst). During the read operation, the read address (ARADDR) starts from 0x0100000, and is incremented by 128.

After each write or read operation, AXI master sends the throughput count value and an eSRAM address starting from 0x20008104 to AHBL master. Then, 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.

For more information on creating a custom AXI interface on user logic,

see the AC409: Connecting User Logic to AXI Interfaces of High-Performance Communication Blocks in the SmartFusion2 Devices Application Note.

For more information on creating a custom AHB interface on user logic, see the AC333: Connecting User Logic to the SmartFusion Microcontroller Subsystem Application Note.

For more information on timing optimization performed in the AXI interface, see the UG0446: SmartFusion2 and IGLOO2 FPGA High Speed DDR Interfaces User Guide.

^{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).



Hardware Implementation

The hardware implementation involves:

- Configuring the System Builder
- Connecting with a user logic AXI master (AXI_IF), AHB master (AHB_IF), and a command decoder (CMD_Decoder)

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

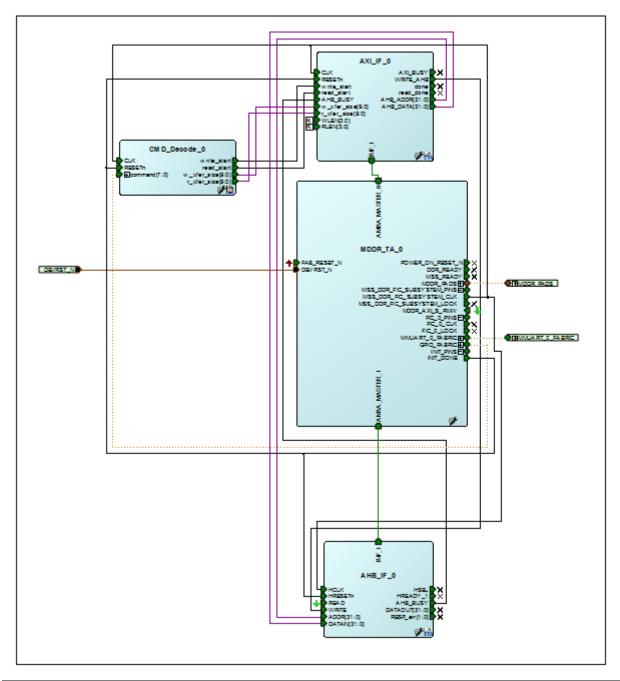


Figure 7 • Top-Level SmartDesign



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 more information on how to launch and use the System Builder wizard, see 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 select the **MDDR** check box. Leave the rest of the check boxes unchecked, as shown in Figure 8.

	Memory
	MSS External Memory
_	MDDR
	Soft Memory Controller (SMC)
	MSS On-chip Flash Memory (eNVM)
	Fabric External DDR Memory (FDDR)
	High Speed Serial Interfaces
	SERDESIF_0
	SERDESIF_1
	SERDESIF_2
	SERDESIF_3
	Microcontroller Options
	Watchdog Timer
	Peripheral DMA
	Real Time Counter

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 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 (See "Appendix: Design Files" on page 36).

		Config	jur
1DDR	(ENVM)	-	
		200	
ſ	Import Configuration Ex	xport Configuration Restore Defaults	
	General Memory Init	itialization Memory Timing	
	Memory Settings		
	Memory Settings Memory Type	DDR3	
		DDR3 16	
	Memory Type	[16	
	Memory Type Data Width	[16	•
	Memory Type Data Width SECDED Enabled ECC Arbitration Scheme	16	•
	Memory Type Data Width SECDED Enabled ECC Arbitration Scheme Highest Priority ID	16 Type-0 0	•
	Memory Type Data Width SECDED Enabled ECC Arbitration Scheme	16	•

Figure 9 • Memory Configuration



 In the Peripherals tab, drag Fabric AMBA Master to MSS DDR FIC Subsystem, as shown in Figure 10. AMBA_MASTER_0 is added to the subsystem. Configure the Interface Type as AXI. Figure 10 shows the Peripherals tab.

Direct Connection	Mode (FIC interfaces are exported out of System Builder)	
	Fabric Slave Cores	Subsystems
Core	Version	MSS FIC_0 - Fabric Master Subsystem
CoreAHBLSRAM	2.0.113	Configure Quantity Name
CoreI2C	7.0.102	1 AMBA_MASTER_1
CoreSPI	3.0.156	MSS FIC_1 - MSS Master Subsystem
CoreGPIO	3.0.120	drag and drop here to add to subsystem
CoreTimer	1.1.101	MSS FIC_1 - Fabric Master Subsystem
CoreUARTapb	5.2.2	drag and drop here to add to subsystem
CorePWM	4.1.106	
Fabric AMBA Slave	0.0.102	MSS DDR FIC Subsystem
		Configure Quantity Name
	Fabric Master Cores	
Core	Version	MSS Peripherals
Fabric AMBA Mast	er 0.0.102	Configure Enable Name
		MM UART 0
		MM-UART_1
		MSS_12C_0
		MSS_J2C_1
		MSS_SPL0
		MSS_SPL1
		MSS_GPIO
		MSS_USB

Figure 10 • Selecting MMUART_0 and MSS GPIO in Peripherals Tab

- 5. Drag Fabric AMBA Master to MSS FIC_0 Fabric Master Subsystem. AMBA_MASTER_1 is added to the subsystem and configured with AHBLite.
- 6. Under MSS Peripherals, select MM_UART_0 and MSS_GPIO.



7. Select **IO** under **Connect To** option in the **MM_UART_0 Configuration** window, as shown in Figure 11.

🕄 Configuring MM_UART_0_0 (SF2_MSS_UART 👝 💷 💌
Configuration
Transmit-Receive
Duplex Mode Full Duplex 🔹
Transmission Mode Asynchronous 💌
Connect To IO 🔹
Modem
Use Modem
Connect To IO 💌
Help OK Cancel

Figure 11 • MM_UART_0 Configuration Window



8. Use the settings in the **MSS_GPIO Configurator** tab, as shown in Figure 12 and keep the remaining at default state. Eight GPIOs are configured as output and routed to the FPGA fabric.

Configuration			
Set/Reset Definitio	n		
GPIO_31_24 Reset Sou	urce SYSREG (MSS_GPIO_31	_24_SOFT_RESET) 🔻	Reset State 1
GPIO_23_16 Reset Sou	urce SYSREG (MSS_GPIO_23	_16_SOFT_RESET) 🔻	Reset State 1
GPIO_15_8 Reset Sour	system System (MSS_GPIO_15	8_SOFT_RESET)	Reset State 1
GPIO_7_0 Reset Sourc	e SYSREG (MSS_GPIO_7_0	_SOFT_RESET) ▼	Reset State 1
GPIO ID	Direction	Package Pin	
GPIO ID	Direction	Package Pin	Connectiv
CDIO 0			EL DOTO A
GPIO_0	Output 🔹		FABRIC_A
GPIO_1	Output		_
-			FABRIC_A
GPIO_1	Output -		FABRIC_A
GPIO_1 GPIO_2	Output		FABRIC_A FABRIC_A FABRIC_A
GPIO_1 GPIO_2 GPIO_3	Output Output Output		FABRIC_A FABRIC_A FABRIC_A FABRIC_A
GPIO_1 GPIO_2 GPIO_3 GPIO_4	Output Output Output Output Output		FABRIC_A FABRIC_A
GPIO_1 GPIO_2 GPIO_3 GPIO_4 GPIO_5	Output Output Output Output Output Output Output		FABRIC_A FABRIC_A FABRIC_A FABRIC_A FABRIC_A

Figure 12 • MSS GPIO Configuration

9. Configure the System clock and Subsystem clocks in the **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 window.

Clock Fabric CCC	Chip Oscillators		
System Clock			
50.0	MHz		
On-chip 25/50 MHz R	C Oscillator		•
M3_CLK	=	166.6	
MDDR Clocks			MHz 166.600
MDDR Clocks	= M3_CLK *	2	MHz 166.600
	_	2	
MDDR_CLK	_	2	333.200
MDDR_CLK DDR/SMC_FIC_CLK	_	2	333.200

Figure 13 • System and Subsystem Clocks Configuration

- 10. Follow the remaining steps with default settings and generate the design.
- 11. Instantiate the custom logic (AXI master, AHB master, and a command decoder) and connect, as shown in Figure 7 on page 11.



Simulation Using Micron DDR3 SDRAM Model

Setting Up the Simulation Model

Setting up and running the simulation involve the following steps:

- Obtain the Micron DDR3 memory model files the SmartFusion2 Advanced Development Kit board has the DDR3 SDRAM from Micron with the part number; MT41K256M8DA -125. The memory model used in the example design supports this device (See "Appendix: Design Files" on page 36).
- 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 tdqs(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_tdqs(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 the ddr3.v file is included at the top of the testbench file. The example design uses two instances of DDR3 models with a device width of eight.



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

Stimulus Hierarchy Show: Components	B × ddr3_pa Show Root Testbenches 5
 work testbench (testbench, MDDR_TA_top ddr3 (ddr3.v) testbench (testbench, testbench (testbench, testbench (testbench, testbench (testbench, axi_slave (axi_slave.v) axi_master (axi_maste) 	Open HDL File Check HDL File Set as active stimulus Create Core from HDL Delete from Project Delete from Disk and Project Properties
III Desig Design Hier Stimulus H	ie Catalog Files

Figure 15 • Stimulus Settings

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

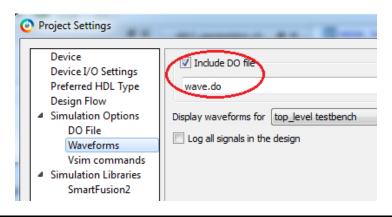


Figure 16 • Waveforms Settings

7. 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 shows the AXI master signals, command from CMD_Decoder, and address and data to AHB master.

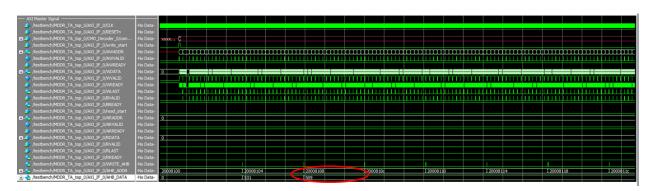


Figure 17 • AXI Master (AXI_IF) Signals for Write Operation

Figure 18 shows the MDDR signals. 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).

/testbench/MDDR_RESET_N	-No Data-										
/testbench/MDDR_CLK	-No Data-										
/testbench/MDDR_CKE	-No Data-	л									
/testbench/MDDR_CS_N	-No Data-	π				l l					
/testbench/COMMAND	-No Data-	<u> NOP</u>	WRITE	WRITE	WRITE	(WRITE	WRITE	(WRITE	WRITE	WRITE	NOP
	-No Data-	<u>) o</u>		-+++++	{{			-{{			
	-No Data-	1	<u>(0</u>	1	2	(3	(4	(5	<u>(6</u>	(7	((1
/testbench/MDDR_DM_RDQS	-No Data-		0								
- /testbench/MDDR_DQS	-No Data-										
/testbench/MDDR_DQ	-No Data-				*****						

Figure 18 • MDDR Signals for Write Operation

Figure 19 shows the AHB master signals. AHB master receives the address and data from AXI master and writes into eSRAM.



Figure 19 • AHB Master Signals



The timing diagrams shown from Figure 20 through Figure 22 shows the read operation. Figure 20 shows the AXI master signals, command from CMD_Decoder, and address and data to AHB master.



Figure 20 • AXI Master (AXI_IF) Signals for Read Operation

Figure 21 shows the MDDR signals. 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).

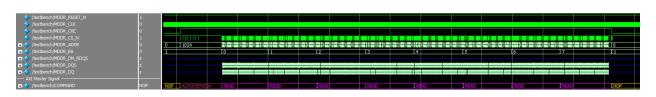


Figure 21 • MDDR Signals for Read Operation

Figure 22 shows the AHB master signals. 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). 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 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 23 shows the Simulation mode.

	🔍 -	V :	Simulation Mode	0
Name	×		Version	
Clock & Manageme	ent			
4 Memory & Control	lers			
Generic DDR Me	mory Simulation	Model	1.0.102	

Figure 23 • Generic DDR Memory Simulation Model

 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 a device width size of eight.

Memory Conf	figuration	
	DRAM Memory Type DDR3	•
	Data Width	•
	Self Refresh Enabled No	•
Burst Configu	uration	
	Burst Length (Bits) 8 💌	
Timing Config	uration	
CAS Late	ency (Clks)	5 🔹
CAS Writ	te Latency	5 🔹
Additive	CAS Latency (Clks)	0 🔹
Write Re	covery (Clks)	5
RFC - Mit	nimum time from refresh to refresh or activate	54
Terminat	ion Resistance (Ohm)	240

Figure 24 • Configuring SimDRAM



 Connect as described in "Simulation using Microsemi DDR3 SDRAM VIP Model" section on page 22. The connections are same as the Micron model. Figure 25 shows the SmartDesign testbench for the example design with Microsemi DDR3 SDRAM VIP model.

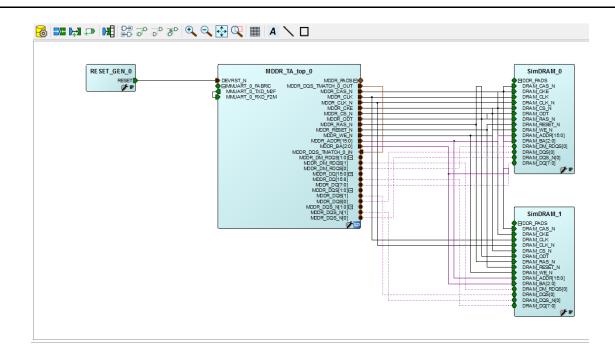


Figure 25 • SmartDesign Testbench for Example Design with Microsemi DDR3 SDRAM VIP

- Generate the design by clicking SmartDesign > Generate Component or by clicking Generate Component on the SmartDesign toolbar.
- 7. 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 ("Loading LSRAM from lsram.mem file");
$display ("");
$readmemh("lsram 512x64.mem",MDDR VIP Simulation.MDDR TA top 0.AXI IF 0.Rdata me
m);
$display (" Completed Loading LSRAM");
@(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;
/* 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;
```

SmartFusion2 - Optimizing DDR Controller for Improved Efficiency - Libero SoC v11.7

```
Wicrosemi
Power Matters.
```

```
/* 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 the Files tab.
```

es		8	>
4.0] component		-
1	🔁 Actel		
3	📋 User		
	🗀 work		
	DESIGN_FIRMWARE		
	MDDR_TA		
	MDDR_TA_MSS		
	MDDR_TA_top		
	MDDR_VIP_Simulation		
	MDDR_VIP_Simulation.v		
Þ 🗋	constraint		
Þ	designer		
Þ 🗋	firmware		
Þ [hdl		
Þ 🗋	simulation		
C	smartgen		
Þ	SoftConsole		
Þ	stimulus		
Þ 🗋	synthesis		
D C	tooldata		

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.

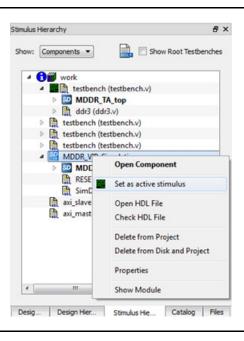


Figure 27 • Stimulus Settings



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

Simulation Libraries

Figure 28 • Waveforms Settings

The timing diagrams from Figure 29 through Figure 31 on page 26 shows the write operation. Figure 29 shows the AXI master signals, command from CMD_Decoder, and address and data to AHB master.

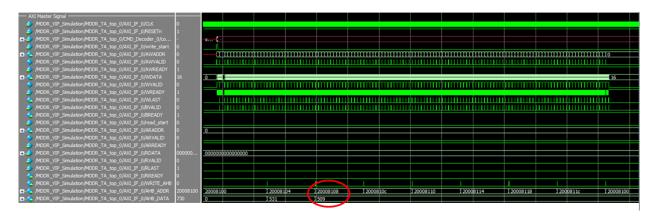


Figure 29 • AXI Master (AXI_IF) Signals for Write Operation

Figure 30 shows the MDDR subsystem signals. 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 31 shows the AHB master signals. AHB master receives address and data from AXI master and writes into eSRAM.

- AHB Master Signal										
/MDDR_VIP_Simulation/MDDR_TA_top_0/AHB_IF_0/HCLK	0									
/MDDR_VIP_Simulation/MDDR_TA_top_0/AHB_IF_0/WRITE	0									
MDDR_VIP_Simulation/MDDR_TA_top_0/AH8_IF_0/ADDR	20008100	20008100	20008104	20008108	2000810c	20008110	20008114	20008118	(2000811c	20008100
🛃 /MDDR_VIP_Simulation/MDDR_TA_top_0/AHB_IF_0/DATAIN	730	0	(531	509						
MDDR_VIP_Simulation/MDDR_TA_top_0/AHB_IF_0/HADDR	00000000	0000000	(00000000	(0000000	00000000	(00000000)	(00000000	(00000000	0000000	(0000000
MDDR_VIP_Simulation/MDDR_TA_top_0/AHB_IF_0/HTRANS	00	00	(00	(00	(00	(00	(00	(00	(00	(00
MDDR_VIP_Simulation/MDDR_TA_top_0/AH8_IF_0/HSIZE	010	010								
4 MDDR_VIP_Simulation/MDDR_TA_top_0/AH8_IF_0/HWRITE	0									
MDDR_VIP_Simulation/MDDR_TA_top_0/AH8_IF_0/HSEL	0							1		
4 /MDDR_VIP_Simulation/MDDR_TA_top_0/AHB_IF_0/HWDATA	730	0	(531	(509						
/MDDR_VIP_Simulation/MDDR_TA_top_0/AHB_IF_0/HREADY	1									

Figure 31 • AHB Master Signals

The timing diagrams from Figure 32 through Figure 34 on page 27 shows the read operation. Figure 32 shows the AXI master signals, command from CMD_Decoder, and address and data to AHB master.

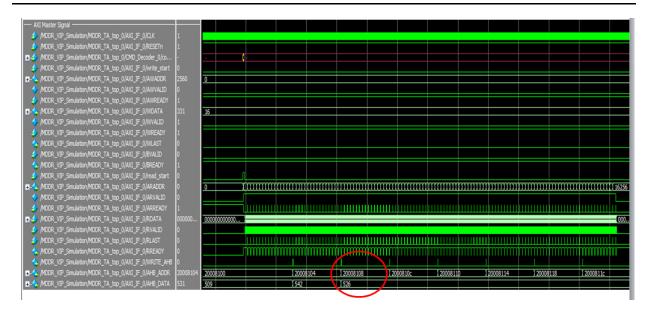


Figure 32 • AXI Master (AXI_IF) Signals for Read Operation

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

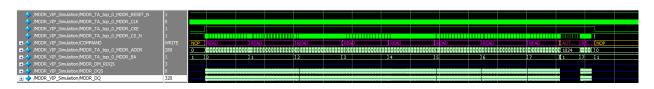


Figure 33 • MDDR Signals for Read Operation



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

AHB Master Signal										
/// MDDR_VIP_Simulation/MDDR_TA_top_0/AH8_JF_0/HCLK	1									
MDDR_VIP_Simulation/MDDR_TA_top_0/AHB_IF_0/WRITE MDDR_VIP_Simulation/MDDR_TA_top_0/AHB_IF_0/ADDR	0 20008104	20008100	20008104	20008108	(2000810c	20008110	20008114	20008118	2000811c	20008100
MDDR_VIP_Simulation/MDDR_TA_top_0/AH8_IF_0/DATAIN		509	(542	(526	, to to the					(730
MDDR_VIP_Simulation/MDDR_TA_top_0/AHB_IF_0/HADDR		00000000	(00000000	(0000000	(00000000	(00000000	(00000000	(00000000	(00000000	(00000000
MDDR_VIP_Simulation/MDDR_TA_top_0/AHB_IF_0/HTRANS		00	100	100	100	(00	100	<u>(00</u>	100	(00
/MDDR_VIP_Simulation/MDDR_TA_top_0/AHB_IF_0/HSIZE /MDDR_VIP_Simulation/MDDR_TA_top_0/AHB_IF_0/HWRITE	010	010			1					
MDDR_VIP_Simulation/MDDR_TA_top_0/AHB_IF_0/HSEL	0									
+ MDDR_VIP_Simulation/MDDR_TA_top_0/AHB_IF_0/HWDATA	531	509	(542	(526						(730
MDDR_VIP_Simulation/MDDR_TA_top_0/AHB_IF_0/HREADY	1									

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.

type filter text	Settings	SC Enter Value
Resource		Defined symbols (-D)
Builders		
C/C++ Build	Configuration: Debug [MICROSEMI_STDIO_THRU_MMUART0
Build Variables		
Discovery Options		
Environment	🛞 Tool Settings 🎤 Bu	di la constante de la constante
Settings		OK Cancel
Tool Chain Editor	🛞 GNU C Compiler	
Tool Chain Editor C/C++ General	Preprocessor	
C/C++ General	Preprocessor	
C/C++ General Project References	🖄 Preprocessor 🖄 Symbols	
C/C++ General Project References Refactoring History	Preprocessor Preprocessor Symbols Preprocessor	

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 AXI master for reading operation through GPIOs.
- Sending a command to 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 the 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. 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. 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 0xA0000000-0xDFFFFFF.
- 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
* Performing the read operation. Uncomment the any of the following lines
\ast based on the size of the data to be read from DDR3 SDRAM. The default size is 2KB*/
//MSS_GPIO_set_outputs(0x26); // 2KB
//delav(50);
                              // 2KB
//MSS GPIO set outputs(0x24); // 2KB
//MSS_GPIO_set_outputs(0x4A); // 4KB
//delay(50);
                              // 4KB
//MSS_GPIO_set_outputs(0x48); // 4KB
//MSS_GPIO_set_outputs(0x6E); // 8KB
//delay(50);
                             // 8KB
//MSS GPIO set outputs(0x6C); // 8KB
//MSS GPIO set outputs(0x92); // 16KB
//delay(50);
                             // 16KB
//MSS GPIO set outputs(0x90); // 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
 * Performing the write operation. Uncomment the any of the following lines based on
* the size of the data to be written into DDR3 SDRAM. The default size is 2KB */
//MSS GPIO set outputs(0x25); // 2KB
//delay(50);
                             // 2KB
//MSS GPIO set outputs(0x24); // 2KB
//MSS_GPIO_set_outputs(0x49); // 4KB
//delay(50);
                            // 4KB
//MSS_GPIO_set_outputs(0x48); // 4KB
//MSS_GPIO_set_outputs(0x6D); // 8KB
//delay(50);
                            // 8KB
//MSS GPIO set outputs(0x6C); // 8KB
//MSS GPIO set outputs(0x91); // 16KB
//delav(50);
                            // 16KB
//MSS_GPIO_set_outputs(0x90); // 16KB
```

6. Printing the write throughput values on HyperTerminal.

Running the Design

The design example is designed to run on the SmartFusion2 Advanced Development Kit board. For more information about the kit board, see *http://www.microsemi.com/products/fpga-soc/design-resources/dev-kits/smartfusion2/smartfusion2-advanced-development-kit*.

Board Jumper Settings

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

Jumper	Pin (From)	Pin (To)	Comments
J116, J353, J354, J54	1	2	These are the default jumper settings of the Advanced
J123	2	3	Development Kit Board. Ensure that these jumpers are set accordingly.
J124, J121, J32	1	2	JTAG programming via FTDI

Table 5 • SmartFusion2 Advanced Development Kit Jumper Settings

Note: Switch OFF the power switch, SW7, while connecting the jumpers.

Host PC to Board Connections

- 1. Connect the FlashPro4 programmer to the FP4 HEADER J37 connector of the SmartFusion2 Advanced Development Kit board.
- 2. Connect the J33 connector on the SmartFusion2 Advanced Development Kit board to the host PC using the USB mini-B (FTDI interface) cable.



USB Driver Installation

Install the FTDI D2XX driver for serial terminal communication through the 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 by verifying the Device Manager, as shown in Figure 36 on page 30.

Note: Copy the COM port number for serial port configuration. Ensure that the COM port Location is specified as on USB Serial Converter C, as shown in Figure 36.

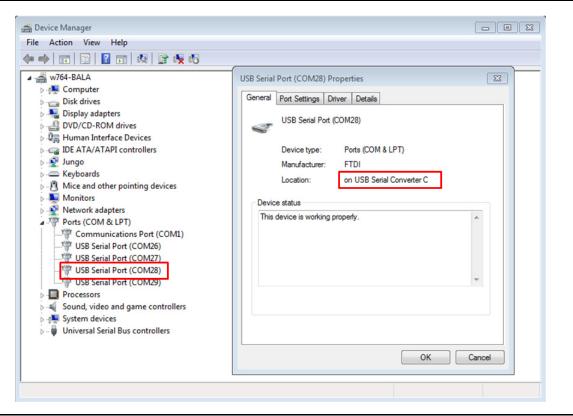


Figure 36 • USB to UART Bridge Drivers

Steps to Run the Design

- 1. Connect the power supply to the J42 connector and FlashPro programmer.
- 2. Switch ON the power supply switch, SW7.
- 3. Program the SmartFusion2 Advanced Development Kit board with the generated or provided *.stp file (See "Appendix: Design Files" on page 36) using FlashPro.
- 4. Invoke the SoftConsole v3.4 integrated design environment (IDE) and launch the debugger.
- 5. Start the HyperTerminal program with the baud rate set to 115200, 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 TeraTerm. See the *Configuring Serial Terminal Emulation Programs Tutorial*, for configuring HyperTerminal, TeraTerm, and PuTTY.



When the debugger runs in SoftConsole, the 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 used for 2 KB of data transferred from LSRAM to DDR3 SDRAM and DDR3 SDRAM to LSRAM.

🚇 COM54:115200baud - Tera Term VT	
File Edit Setup Control Window Help	
Øxa1007ffc : 1023 Øxa0007ffc : 1023	A
Integrity Test Completed	
Status : PASSED !	
DDR SDRAM Bandwidth	
Read Throughput 	
AXI Clocks (1) : 543 AXI Clocks (2) : 730 AXI Clocks (3) : 526 AXI Clocks (4) : 526 AXI Clocks (5) : 526 AXI Clocks (5) : 526 AXI Clocks (6) : 526 AXI Clocks (8) : 526 AXI Clocks (8) : 526	
Write Throughput	
AXI Clocks (1) : 529 AXI Clocks (2) : 509 AXI Clocks (3) : 509 AXI Clocks (4) : 509 AXI Clocks (5) : 509 AXI Clocks (5) : 509 AXI Clocks (6) : 509 AXI Clocks (8) : 695	
Completed !. Re-run for next size	*

Figure 37 • Throughput for 2 KB Data

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

0xa1007ffc : 1023	0xa0007ffc	: 1023	
Integrity Test Completed			
Status : PASSED !			
DDR SDRAM Bandwidth The second s			
AXI Clocks (1) : 1071 AXI Clocks (2) : 1054 AXI Clocks (3) : 1054 AXI Clocks (3) : 1054 AXI Clocks (5) : 1263 AXI Clocks (6) : 1054 AXI Clocks (7) : 1054 AXI Clocks (7) : 1054			
Write Throughput			
AXI Clocks (1) : 1041 AXI Clocks (2) : 1021 AXI Clocks (3) : 1021 AXI Clocks (4) : 1021 AXI Clocks (5) : 1021 AXI Clocks (6) : 1021 AXI Clocks (6) : 1021 AXI Clocks (8) : 1206			

Figure 38 • Throughput for 4 KB Data



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

ile Edit Setup Control Window He	lp		
Øxa1007ffc : 1023	0xa0007ffc	: 1023	
tegrity Test Completed			
atus : PASSED !			

OR SDRAM Bandwidth			

lead Throughput			
I Clocks (1) : 2127 I Clocks (2) : 2110			
(1) Clocks (2) = 2110			
I Clocks (4) : 2110			
I Clocks (5) : 2110			
I Clocks (6) : 2110			
I Clocks (7) : 2110			
I Clocks (8) : 2315			
hrite Throughput			
I Clocks (1) : 2065			
(1) = 2065 (1) Clocks (2) = 2045			
(1) Clocks (2) + 2043			
I Clocks (4) : 2045			
I Clocks (5) : 2045			
I Clocks (6) : 2045 I Clocks (7) : 2045			

Figure 39 • Throughput for 8 KB Data

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

I	ontrol Window Help			
0xa1007ffc	: 1023	0xa0007ffc	: 1023	
Integrity Test C	ompleted			
Status : PASSED	•			
DR SDRAM Bandwi Read Throughput	.dth ++++			
NXI Clocks (1) : IXI Clocks (2) : IXI Clocks (3) : IXI Clocks (4) : IXI Clocks (5) : IXI Clocks (6) : IXI Clocks (7) : IXI Clocks (8) :	4222 4426 4222 4434 4222 4222 4222			
Write Throughpu	t			
NXI Clocks (1) : NXI Clocks (2) :				

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 КВ	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)



Simulation Result

Table 7 lists the write and read bandwidths 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 DDR3 SDRAM to LSRAM.

SI No	Optimization Techniques		Write		Read			
		Size (KB)	Number of Cycles	Bandwidth (MB/Sec)	Number of Cycles	Bandwidth (MB/Sec)	Write Improvement (Average)	Read Improvement (Average)
Base	160 MHz	2	539	607	737	445	605	448
		4	1083	605	1476	444		
		8	2171	604	2954	443		
		16	4347	603	5921	442		1
1	166 MHz	2	539	631	737	461	627.5	461 2.8%
		4	1083	628	1476	461	3.7%	
		8	2171	626	2954	460		
		16	4347	625	5913	460	-	
2	166 MHz Without Write	2	509	668	737	461	666 6.13%	461 2.8%
		4	1021	666	1476	461		
	Response State	8	2045	665	2954	460		
		16	4093	664	5913	460		
3	166 MHz	2	509	668	736	462	666	461
	Without Write	4	1021	666	1474	461	6.13%	2.8%
	Response State Tuned DDR	8	2045	665	2950	461	-	
	Configuration		460					
4	166 MHz	2	509	668		666	645	
	Without Write Response State Tuned DDR Configuration	4	1021	666	1054	645	6.13%	39.9%
		8	2045	665	2110	644		
		16	4093	664	4222	644		
	Read Command Queuing							

 Table 7 • DDR3 SDRAM Bandwidth - Simulation Result



Board Test Result

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

			Write		Read		Write	Read	
SI No	Optimization Techniques	Size (KB)	Number of Cycles	Bandwidth (MB/Sec)	Number of Cycles	Bandwidth (MB/Sec)	Improvement (Average)	Improvement (Average)	
Base	160 Mhz	2	539	607	737	445	605	448	
		4	1083	605	1476	444			
		8 2171 604 2954 443							
		16	4347	603	5913	460			
1	166.6 Mhz	2	539	631	736	461	627.5 3.7%	461 2.8%	
		4	1083	628	1474	461			
		8	2171	626	2950	461			
		16	4347	625	5924	460			
2	166.6 Mhz	2	509	668	736	461	666 6.13%	461 2.8%	
	Without Write Response State	4	1021	666	1477	461			
		8	2045	665	2959	461	-		
		16	4093	664	5923	459			
3	166.6 Mhz	2	509	668	736	462	666 6.13%	461 2.8%	
	Without Write Response State Tuned DDR	4	1021	666	1474	461			
		8	2045	665	2953	461			
	Configuration	16	4093	664	5918	460			
4	166.6 MHz Without Write Response State Tuned DDR	2	509	668	526	646	666 6.13%	645 39.9%	
		4	1021	666	1054	645			
		8	2045	665	2110	644			
	Configuration Read Command Queuing	16	4093	664	4222	644			

Conclusion

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



Appendix: Design Files

The design files can be downloaded from the Microsemi website: http://soc.microsemi.com/download/rsc/?f=m2s_ac422_liberov11p7_df

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



List of Changes

Revision	Changes	Page
Revision 6 (April 2016)	Updated the document for Libero SoC v11.7 software release (SAR 78130).	NA
Revision 5 (October 2015)	Updated the document for Libero SoC v11.6 software release (SAR 72584).	NA
Revision 4 (May, 2015)	Updated the document for Libero SoC v11.5 software release (SAR 67501).	NA
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

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



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

© 2016 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.

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; enterprise storage and communication solutions, security technologies and scalable anti-tamper products; Ethernet solutions; Powerover-Ethernet ICs and midspans; as well as custom design capabilities and services. Microsemi is headquartered in Aliso Viejo, Calif., and has approximately 4,800 employees globally. Learn more at **www.microsemi.com**.

Microsemi makes no warranty, representation, or guarantee regarding the information contained herein or the suitability of its products and services for any particular purpose, nor does Microsemi assume any liability whatsoever arising out of the application or use of any product or circuit. The products sold hereunder and any other products sold by Microsemi have been subject to limited testing and should not be used in conjunction with mission-critical equipment or applications. Any performance specifications are believed to be reliable but are not verified, and Buyer must conduct and complete all performance and other testing of the products, alone and together with, or installed in, any end-products. Buyer shall not rely on any data and performance specifications or parameters provided by Microsemi. It is the Buyer's responsibility to independently determine suitability of any products and to test and verify the same. The information provided by Microsemi hereunder is provided "as is, where is" and with all faults, and the entire risk associated with such information is entirely with the Buyer. Microsemi does not grant, explicitly or implicitly, to any party any patent rights, licenses, or any other IP rights, whether with regard to such information itself or anything described by such information. Information provided in this document is proprietary to Microsemi, and Microsemi reserves the right to make any changes to the information in this document or to any products and services at any time without notice.