Interfacing SmartFusion2 with DDR3 Memory through MDDR Controller

Libero SoC v11.7 System Builder Flow Tutorial TU0372

Microsemi
Power Matters.™

Contents

1 Interfacing SmartFusion2 with DDR3 Memory through MDDR Controller 5
- 1.1 Introduction 5
- 1.2 Design Requirements 5
  - 1.2.1 Project Files 5
- 1.3 Design Overview 6
- 1.4 Step 1: Creating a Libero SoC Project 8
  - 1.4.1 Opening System Builder Component as SmartDesign 17
- 1.5 Step 2: Generating the Testbench 19
- 1.6 Step 3: Modifying the BFM Scripts 27
- 1.7 Step 4: Simulating the Design 30
- 1.8 Step 5: Validating the Simulation Results 32
- 1.9 Conclusion 33

2 Appendix: VHDL Flow 34

3 Revision History 36

4 Product Support 37
- 4.1 Customer Service 37
- 4.2 Customer Technical Support Center 37
- 4.3 Technical Support 37
- 4.4 Website 37
- 4.5 Contacting the Customer Technical Support Center 37
  - 4.5.1 Email 37
  - 4.5.2 My Cases 37
  - 4.5.3 Outside the U.S. 38
- 4.6 ITAR Technical Support 38

Figures

Figure: 1 Top-Level Block Diagram 7
Figure: 2 New Project Details Window 8
Figure: 3 Device Selection Window 9
Figure: 4 Device Settings Window 10
Figure: 5 New Project Information Window 11
Figure: 6 Create New System Builder Dialog Box 11
Figure: 7 SmartFusion2 SoC FPGA System Builder Configurator 12
Figure: 8 System Builder Configurator - Memory Page 13
Figure: 9 System Builder Configurator - Select Peripherals Page 14
Figure: 10 System Builder Configurator - Clock Settings Page 15
Figure: 11 SmartFusion2 SoC FPGA System Builder Generated System 16
Figure: 12 SmartFusion2 SoC FPGA Generated Final System 16
Figure: 13 Open as SmartDesign Option 17
Figure: 14 Successful Conversion of System Builder to a SmartDesign Message 17
Figure: 15 System Builder Generated System Opened in the SmartDesign 18
Figure: 16 Simulation Cores in Libero IP Catalog 19
Figure: 17 Opening SmartDesign Testbench 19
Figure: 18 Create New SmartDesign Testbench Dialog Box 20
Figure: 19 SmartDesign Testbench Canvas 20
Figure: 20 RESET_GEN Configuration 20
Figure: 21 CLK_GEN Configuration 21
Figure: 22 Import the DDR3 Models as Stimulus Files 21
Figure: 23 Imported DDR3 in Stimulus Folder 22
Figure: 24 Stimulus Hierarchy Window 22
Figure: 25 System Testbench Canvas with DDR3 Models Instantiated 23
Figure: 26 Enabling the Connection Mode Option 23
Figure: 27 Creating a Bus Slice 24
Figure: 28 Edit Slices Dialog Box 24
Figure: 29 MDDR_ADDR Created Slices 25
Figure: 30 Fully Connected MDDR System 26
Figure: 31 Setting a Testbench as Active Testbench 26
Figure: 32 SmartDesign Generated user.bfm File 27
Figure: 33 Importing bfm Source File 28
Figure: 34 Replacing Existing user.bfm File 28
Figure: 35 Reloading and Updating user.bfm File 28
Figure: 36 user.bfm after Adding the Commands 29
Figure: 37 Project Setting - Do File Simulation Runtime Setting 30
Figure: 38 Project Setting - Specifying the MDDR_wave.do File Location 31
Figure: 39 Starting Pre-Synthesis Simulation 31
Figure: 40 ModelSim BFM Simulation Transcript Results 32
Figure: 41 Doc or Undock of ModelSim Wave Window 32
Figure: 42 Zoom Full Option 32
Figure: 43 Zoom In on Active Cursor Button 33
Figure: 44 Write/Read Data 33
Figure: 45 Project Settings - Specifying - novopt Simulation Option 34
Figure: 46 Project Settings - Specifying Precompiled Library Path 35

Tables

Table 1: Tutorial Requirements and Details 5
Table 2: DDR3 Pins Connections 25

1 Interfacing SmartFusion2 with DDR3 Memory through MDDR Controller

1.1 Introduction

This tutorial describes how to create a hardware design using the System Builder to access an external double data rate type 3 (DDR3) memory through the built-in hard ASIC microcontroller subsystem (MSS) DDR (MDDR) in SmartFusion®2 system-on-chip (SoC) field programmable gate array (FPGA) devices. This tutorial also shows how to functionally verify the design using bus functional model (BFM) simulation. The SmartFusion2 SoC FPGA has up to two DDR controllers. These controllers are the MDDR and fabric DDR (FDDR) controllers. The MDDR controller is a hard ASIC block in the SmartFusion2 SoC FPGA device. The FDDR controller is also a hard ASIC block, which can be used to simplify the interfacing of different DDR memory standards to the SmartFusion2 SoC FPGA fabric.

Note: The FDDR is not part of the MSS.

This design focuses on using the ARM® Cortex®-M3 processor as a master that communicates to an external DDR3 SDRAM memory through the MDDR controller. The MDDR controller interfaces with the Cortex-M3 processor through the 64-bit AXI bus interface.

This tutorial describes the following:

- Creating a Libero® System-on-Chip (SoC) v11.7 project using the SmartFusion2 SoC FPGA.
- Configuring and generating the various hardware blocks and clocking system using the System Builder.
- Creating and generating testbench using the SmartDesign testbench generator feature.
- Performing functional level verification of the design using AMBA® BFM simulation in MentorGraphics ModelSim® Simulator.
- Using the ModelSim GUI to see the various design signals in ModelSim Waveform window.

1.2 Design Requirements

This tutorial requires the following software and MSS core version installed in the PC:

Table 1 • **Tutorial Requirements and Details**

Design Requirements	Description
Software Requirements	
Libero SoC	v11.7
MSS	v1.1.400
Hardware Requirements	
Host PC or Laptop	Any 64-bit Windows Operating System

1.2.1 Project Files

The associated solution and source project files along with the `readme.txt` file for this tutorial can be downloaded from Microsemi website:
http://soc.microsemi.com/download/rsc/?f=m2s_tu0372_ddr3_mddr_liberov11p7_df

Note: Extract the design files to the root directory. The Source_files folder includes the `MDDR_wave.do`, `user.bfm`, and the DDR3 associated files.

1.3 Design Overview

The design demonstrates the read or write access to an external slave DDR3 memory using the SmartFusion2 device. Inside the SmartFusion2 SoC FPGA, the Cortex-M3 processor acts as the master and performs the read and write transactions on the external slave memory. These read and write transactions between the Cortex-M3 processor and the external DDR3 memory are executed through the DDR bridge and the MDDR memory controller, which are part of the MSS.

The DDR bridge block is basically responsible for managing the read and write requests from the various masters to the DDR controller in the MSS block. The DDR bridge also connects the AMBA high-performance bus (AHB) based masters such as the Cortex-M3 processor to AXI based MDDR controller.

The MDDR controller interfaces with the DDR bridge through a 64-bit AMBA AXI interface and with the external DDR3 memory through the SmartFusion2 SoC FPGA DDR I/Os. The MDDR controller takes care of converting the AXI transactions into the DDR3 memory read and write transactions with appropriate timing generation. It also handles the appropriate command generation for write/read/refresh/precharge operations required for DDR3 memory.

The MDDR contains two 64-bit AXI interfaces, one dedicated to the DDR interface and the other to the FPGA fabric. The MDDR can be used either to interface with the external DDR slave memory or to interface with the FPGA fabric through the DDR_FIC interface. The DDR_FIC interface provides either a single 64-bit AXI interface, one 32-bit AHB interface, or two 32-bit AHB interfaces to the FPGA fabric.

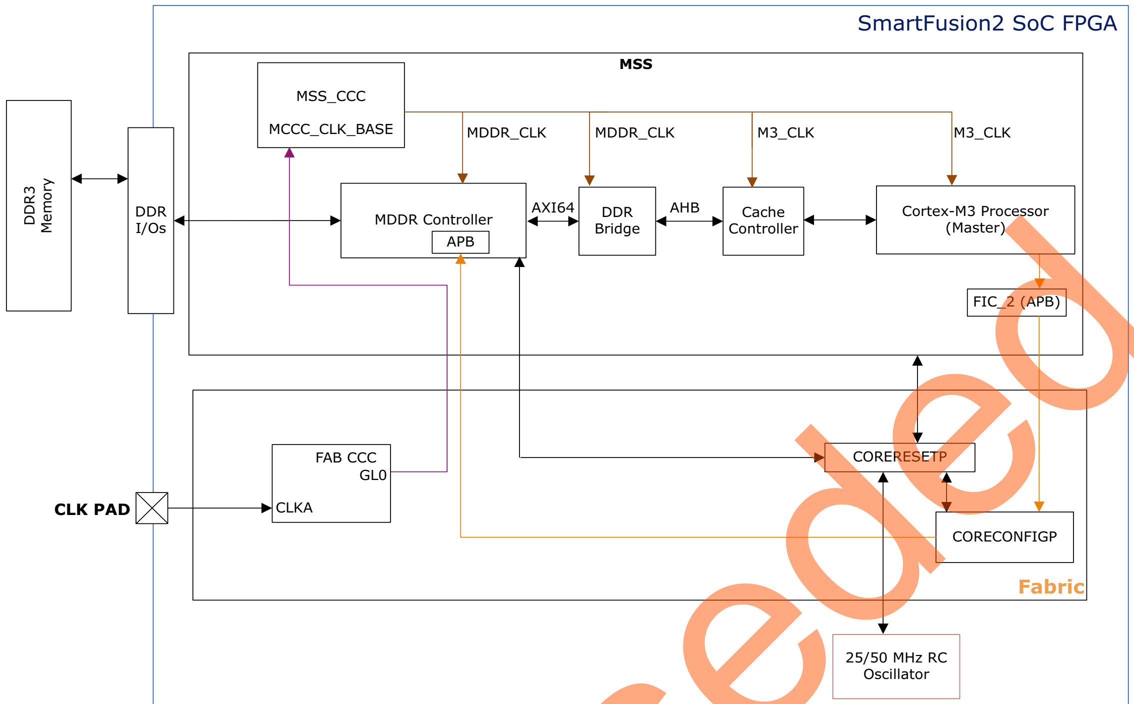
The MDDR controller must be configured to match the external DDR memory specifications. In this tutorial it is the DDR3 specifications. The configuration of the MDDR can be defined in a file and the file can be imported using the System Builder or using the DDR configurator. The configuration is done through the CoreConfigP soft IP core which is the master of the configuration data initialization process. Upon reset, the soft IP core CoreConfigP copies the data from embedded nonvolatile memory (eNVM) to the configuration registers of the DDR through the FIC_2 advanced peripheral bus (APB) interface based on user specific configurations. The RESET mechanism of the overall system is managed by the soft IP core CoreResetP. The CoreConfigP notifies the CoreRestP when the register configuration phase is complete. The MSS interfaces with the CoreConfigP IP core through the APB interface (FIC_2) to initialize the MDDR controller registers based on a user specified configuration file. See the CoreConfigP and CoreResetP handbooks in the IP Catalog of the Libero SoC software for more information.

The purpose of this tutorial is to demonstrate the interface of the MDDR with an external DDR slave memory through the MSS. In this design, the System Builder is used to configure the system clocks and the MDDR block to access the external DDR3 memory through the MSS through the DDR I/Os without going through the fabric.

In the SmartFusion2 SoC FPGA device, there are six clock conditioning circuits (Fabric CCCs) inside the fabric and one CCC (MSS CCC) block inside the MSS. Each of the CCC blocks has an associated PLL. These CCC blocks and their PLLs provide many clock conditioning capabilities such as clock frequency multiplication, clock division, phase shifting, and clock-to-output or clock-to-input delay canceling. The Fabric CCC blocks inside the fabric can directly drive the global routing buffers inside the fabric, which provides a very low skew clock routing network throughout the FPGA fabric. In this design, the MSS CCC and fabric CCC blocks are configured using the System Builder to generate the clocks for the various elements inside the MSS and the fabric respectively.

Figure 1 on page 7 shows the different blocks used in this design.

Figure 1 • **Top-Level Block Diagram**

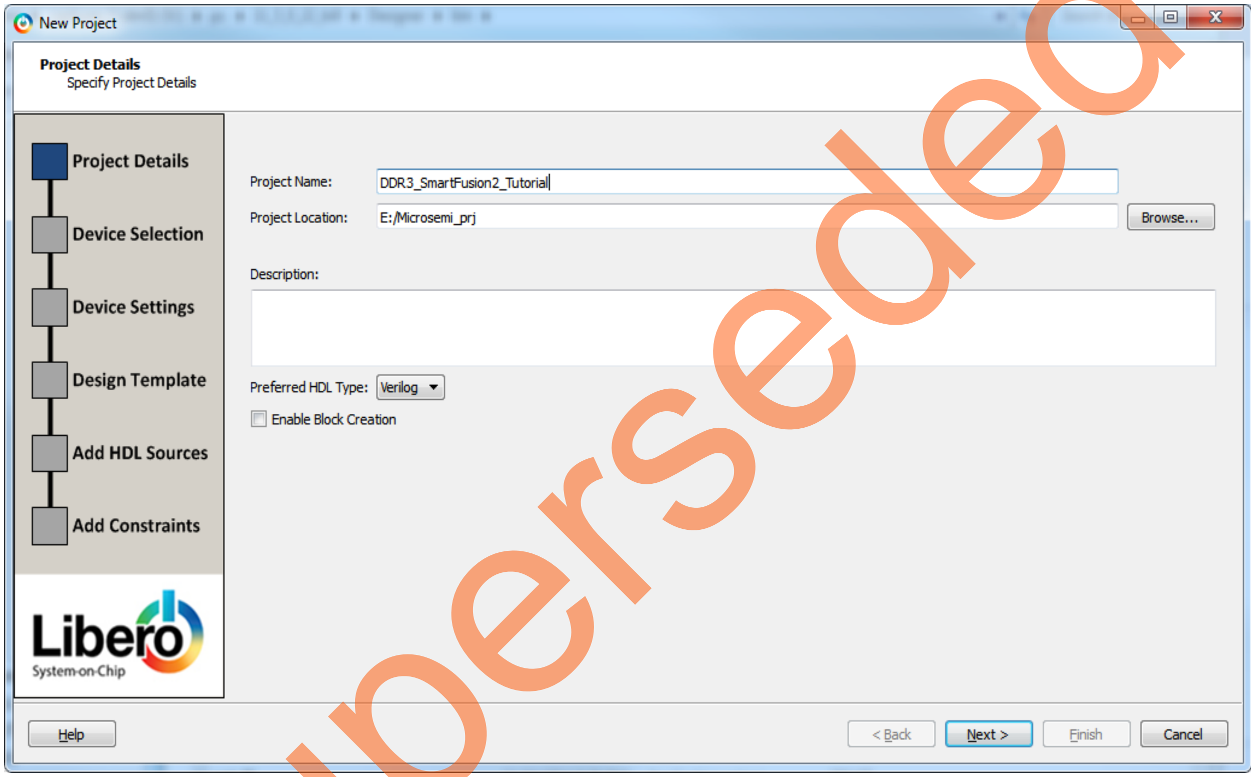


1.4 Step 1: Creating a Libero SoC Project

The following steps describe how to create a Libero SoC project:

1. Launch Libero SoC v11.7.
2. From the **Project** menu, select **New Project**. In the **Project Details** window, enter the information as displayed in Figure 2 on page 8.
   - Project Name: DDR3_SmartFusion2_Tutorial
   - Project Location: Select an appropriate location (For example, C:/Microsemi_prj)
   - Preferred HDL Type: Verilog
   - Enable Block Creation: Unchecked

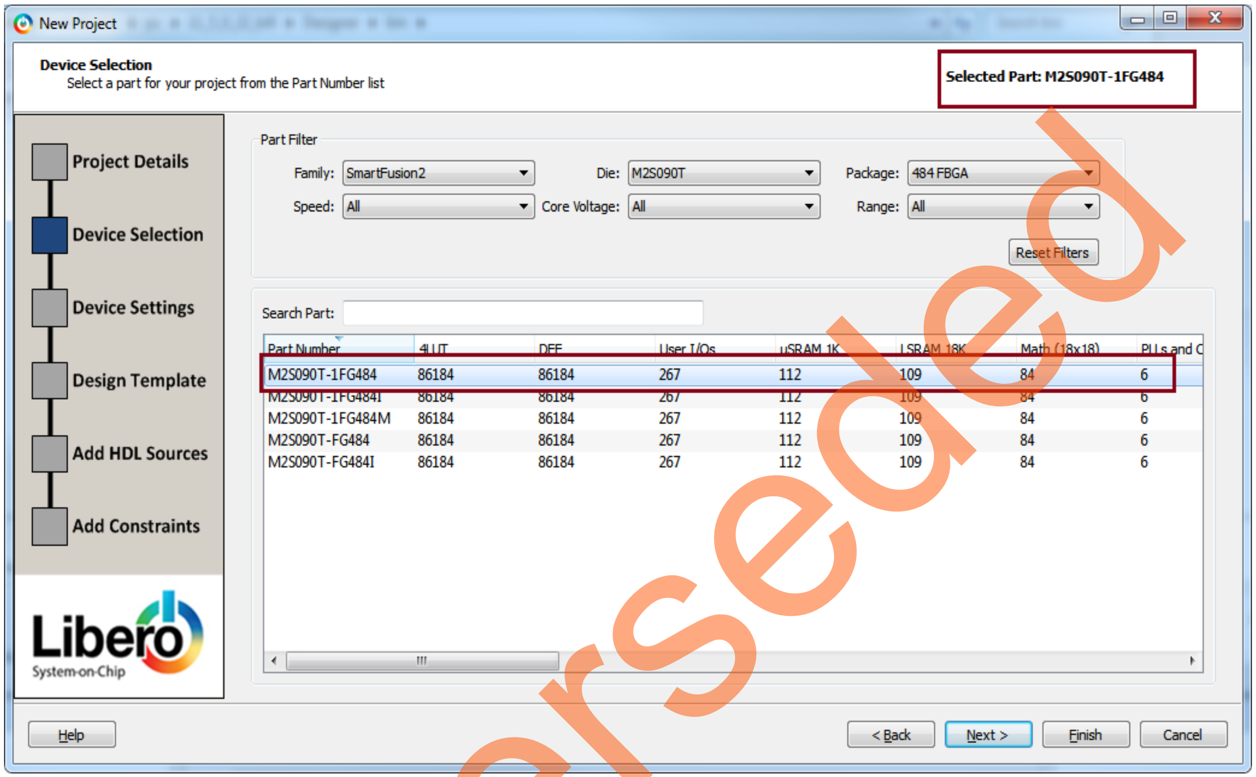
Figure 2 • **New Project Details Window**



Project Name: DDR3_SmartFusion2_Tutorial
Project Location: E:/Microsemi_prj
Description:
Preferred HDL Type: Verilog
Enable Block Creation: unchecked

3. Click **Next**. In the **Device Selection** window, select the information displayed in Figure 3 on page 9.
   - Family: SmartFusion2
   - Die: M2S090T
   - Package: 484 FBGA
   - Part Number: M2S090T-1FG484

Figure 3 • **Device Selection Window**



Device Selection
Select a part for your project from the Part Number list

Selected Part: M2S090T-1FG484

Part Filter

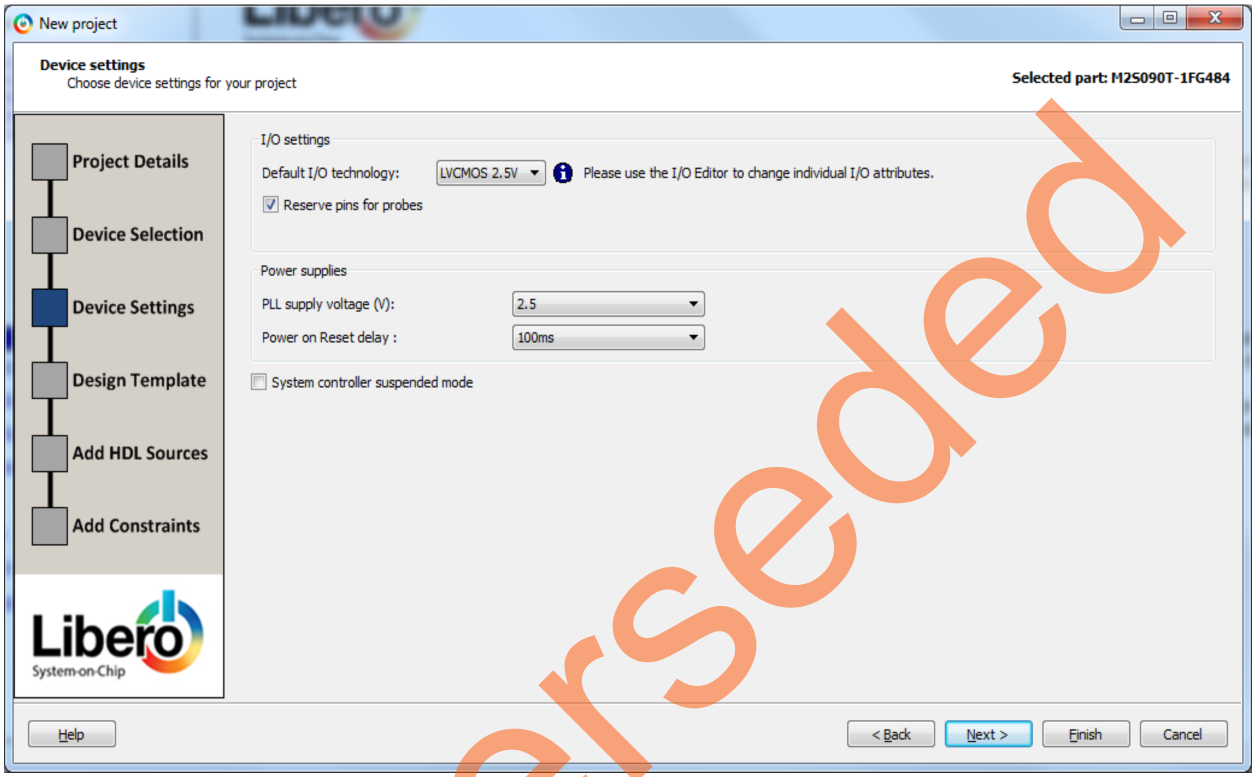
Family: SmartFusion2 Die: M2S090T Package: 484 FBGA
Speed: All Core Voltage: All Range: All

Search Part:

Part Number	4LUT	DFF	User I/Os	uSRAM 1K	LSRAM 18K	Math (18x18)	PLLs and C
M2S090T-1FG484	86184	86184	267	112	109	84	6
M2S090T-1FG484I	86184	86184	267	112	109	84	6
M2S090T-1FG484M	86184	86184	267	112	109	84	6
M2S090T-FG484	86184	86184	267	112	109	84	6
M2S090T-FG484I	86184	86184	267	112	109	84	6

4. Click **Next**. In the **Device Settings** window, select the information displayed in Figure 4 on page 10.
   - Default I/O Technology: LVCMOS 2.5V
   - PLL Supply Voltage (V): 2.5
   - Power on Reset delay: 100ms
   - System controller suspend mode: Unchecked

Figure 4 • **Device Settings Window**



New project

Device settings
Choose device settings for your project

Selected part: M2S090T-1FG484

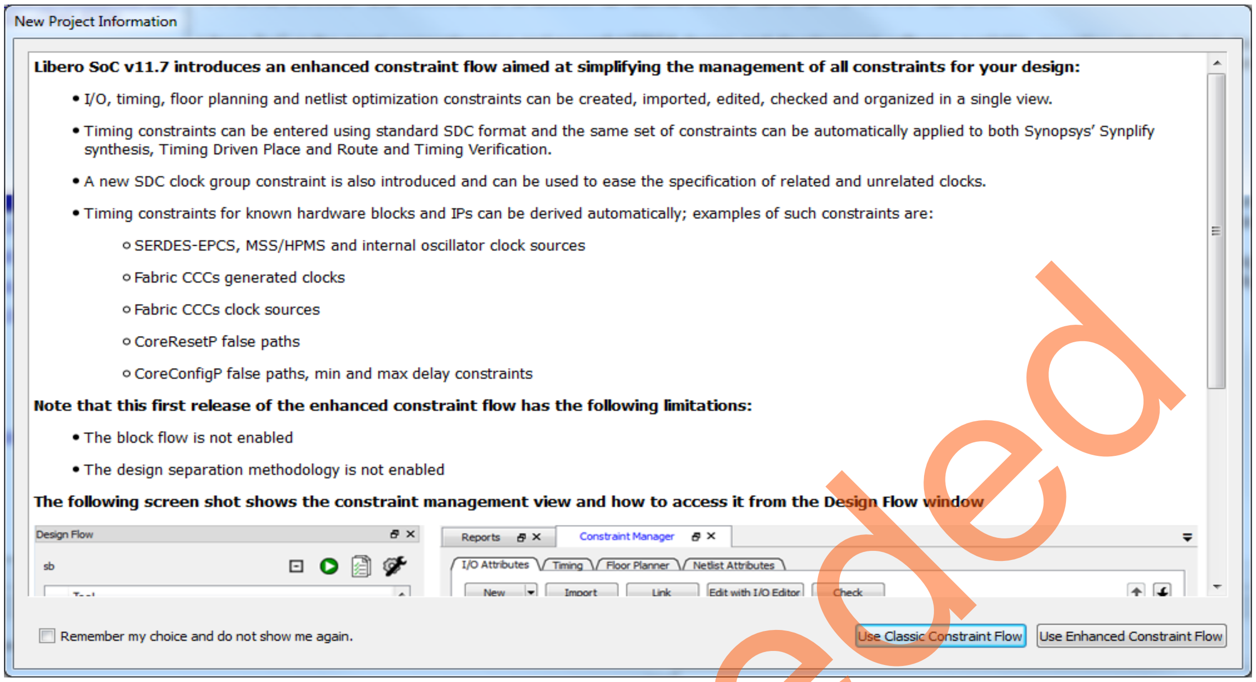
I/O settings
Default I/O technology: LVCMOS 2.5V
Please use the I/O Editor to change individual I/O attributes.
[✓] Reserve pins for probes

Power supplies
PLL supply voltage (V): 2.5
Power on Reset delay : 100ms

[ ] System controller suspended mode

5. Click **Next**. In the **Design Template** page, select the **Create a System Builder based design** under the **Design Templates and Creators**.
6. Click **Finish**.
7. A **New Project information** selection screen will appear, select **Use Enhanced Constraint Flow**.

Figure 5 • **New Project Information Window**



Libero SoC v11.7 introduces an enhanced constraint flow aimed at simplifying the management of all constraints for your design:

- I/O, timing, floor planning and netlist optimization constraints can be created, imported, edited, checked and organized in a single view.
- Timing constraints can be entered using standard SDC format and the same set of constraints can be automatically applied to both Synopsys' Synplify synthesis, Timing Driven Place and Route and Timing Verification.
- A new SDC clock group constraint is also introduced and can be used to ease the specification of related and unrelated clocks.
- Timing constraints for known hardware blocks and IPs can be derived automatically; examples of such constraints are:
  - SERDES-EPCS, MSS/HPMS and internal oscillator clock sources
  - Fabric CCCs generated clocks
  - Fabric CCCs clock sources
  - CoreResetP false paths
  - CoreConfigP false paths, min and max delay constraints

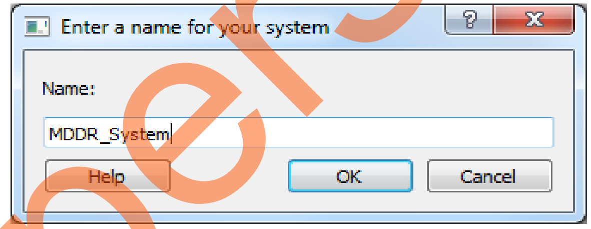
Note that this first release of the enhanced constraint flow has the following limitations:

- The block flow is not enabled
- The design separation methodology is not enabled

The following screen shot shows the constraint management view and how to access it from the Design Flow window

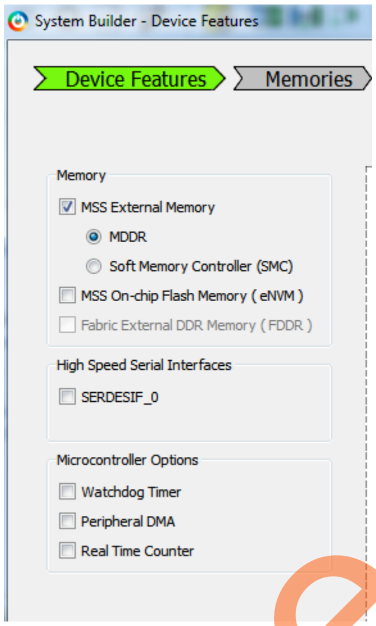
8. Enter **MDDR_system** as the name of the system and then click **OK**, as shown in Figure 6 on page 11.

Figure 6 • **Create New System Builder Dialog Box**



9. The System Builder dialog box is displayed with the **Device Features** page, as shown in Figure 7 on page 12.

Figure 7 • **SmartFusion2 SoC FPGA System Builder Configurator**



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

Microcontroller Options
- [ ] Watchdog Timer
- [ ] Peripheral DMA
- [ ] Real Time Counter

10. Under Memory, check **MSS External Memory** and select **MDDR**. Leave all other options unchecked.
11. Click **Next**, the **System Builder - Memories** page is displayed, as shown in Figure 8 on page 13. The DDR3 external memory models are used in this tutorial.
    - DDR memory settling time (µs): 200

When you use an external memory model, you need to wait for the memory to initialize (settling time) before you try to access it. Since you are using the DDR3 memory model, you need to wait at least 200 µs.

The DDR controller must be configured to match the external DDR3 memory specifications. The configuration is done through the CoreConfigP soft IP core, which is the master of the configuration data initialization process. Upon reset, the soft CoreConfigP copies the data from eNVM to the configuration registers of the DDR controller through FIC_2 APB interface.

The System Builder enables you to import the register configuration file in which you defined the DDR controller configurations. For this design, a configuration file `DDR3_PHY_16_NO_ECC_BL8_INTER.txt`, is provided in the tutorial zip files. The configuration file is located under *<project directory>\m2s_tu0372_ddr3_mddr_liberov11p7_df\Source_files\DDR3* folder.

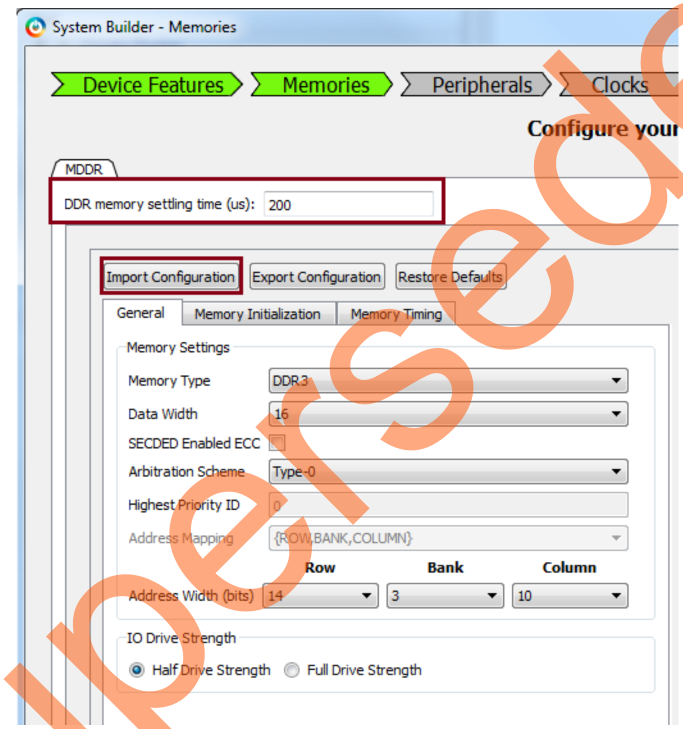
Import the register configuration file as follows:

- Click **Import Configuration**, as shown in Figure 8 on page 13.
- The Import File window is displayed. Browse to the provided DDR3 configuration file `DDR3_PHY_16_NO_ECC_BL8_INTER.txt` and import it.

After importing the register configuration file, confirm the settings as follows:

- Memory Type: DDR3
- Data Width: 16
- SECDED Enabled ECC: Unchecked

Figure 8 • **System Builder Configurator - Memory Page**

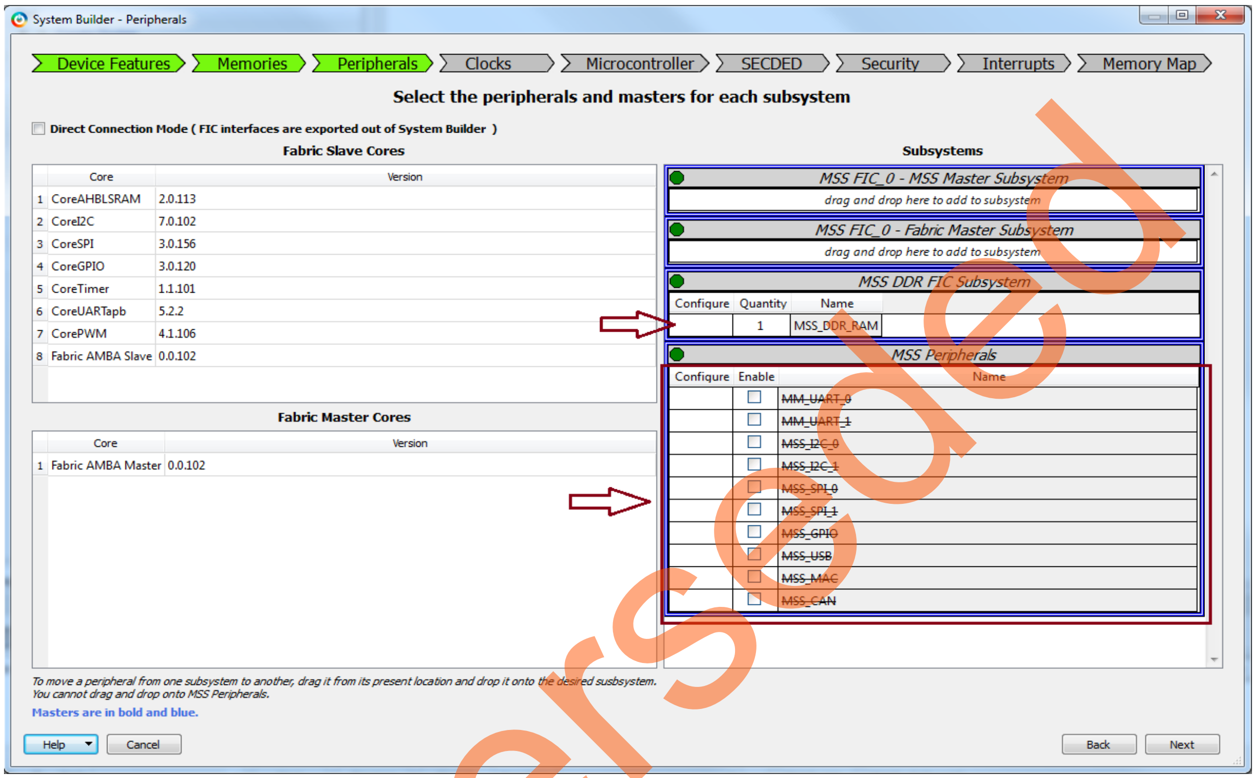


DDR memory settling time (us): 200

Memory Type: DDR3
Data Width: 16
SECDED Enabled ECC: ☐
Arbitration Scheme: Type-0
Highest Priority ID: 0
Address Mapping: {ROW,BANK,COLUMN}
Address Width (bits): Row 14, Bank 3, Column 10
IO Drive Strength: ● Half Drive Strength ○ Full Drive Strength

12. Click **Next**, **System Builder - Peripherals** page is displayed, as shown in Figure 9 on page 14.
    This tutorial does not use any of the MSS peripherals. Clear all the MSS Peripherals, as shown in Figure 9 on page 14.
    Since the system is using an MSS DDR (on the first page of the System Builder), the MSS_DDR_RAM is shown under the MSS DDR FIC Subsystem, as shown in Figure 9 on page 14.

Figure 9 • **System Builder Configurator - Select Peripherals Page**



Select the peripherals and masters for each subsystem

Direct Connection Mode (FIC interfaces are exported out of System Builder)

Fabric Slave Cores

	Core	Version
1	CoreAHBLSRAM	2.0.113
2	CoreI2C	7.0.102
3	CoreSPI	3.0.156
4	CoreGPIO	3.0.120
5	CoreTimer	1.1.101
6	CoreUARTapb	5.2.2
7	CorePWM	4.1.106
8	Fabric AMBA Slave	0.0.102

Fabric Master Cores

	Core	Version
1	Fabric AMBA Master	0.0.102

Subsystems

MSS FIC_0 - MSS Master Subsystem

drag and drop here to add to subsystem

MSS FIC_0 - Fabric Master Subsystem

drag and drop here to add to subsystem

MSS DDR FIC Subsystem

Configure	Quantity	Name
	1	MSS_DDR_RAM

MSS Peripherals

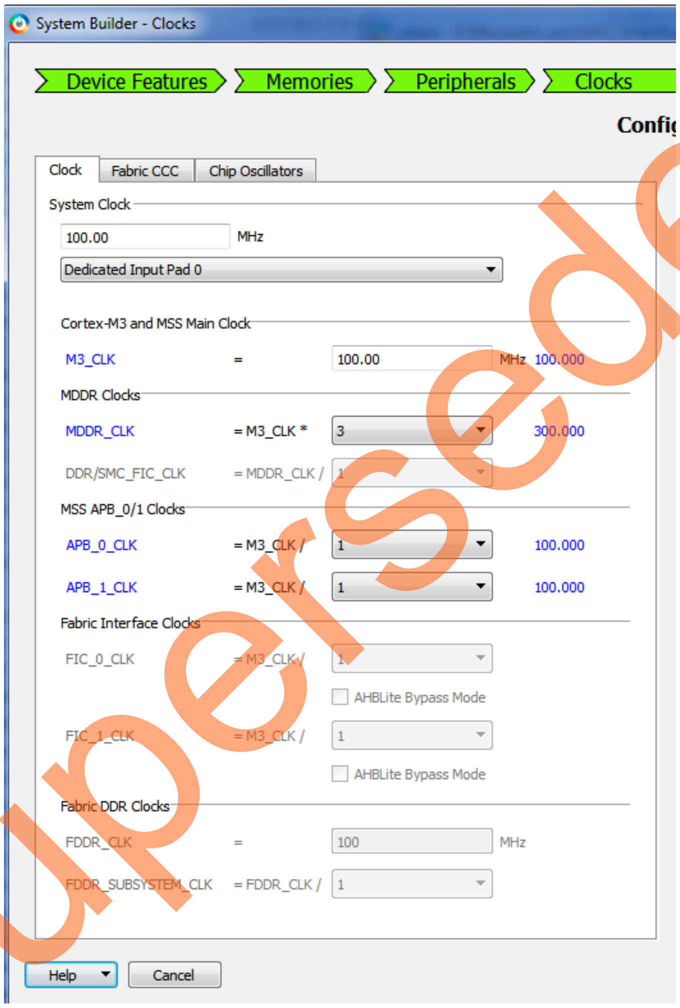
Configure	Enable	Name
	☐	~~MM_UART_0~~
	☐	~~MM_UART_1~~
	☐	~~MSS_I2C_0~~
	☐	~~MSS_I2C_1~~
	☐	~~MSS_SPI_0~~
	☐	~~MSS_SPI_1~~
	☐	~~MSS_GPIO~~
	☐	~~MSS_USB~~
	☐	~~MSS_MAC~~
	☐	~~MSS_CAN~~

To move a peripheral from one subsystem to another, drag it from its present location and drop it onto the desired susbsystem.
You cannot drag and drop onto MSS Peripherals.
Masters are in bold and blue.

13. Click **Next**. The System Builder- **Clocks** page is displayed, as shown in Figure 10 on page 15. Select the following options:
    - System Clock: Set it to 100 MHz (default) and select **Dedicated Input Pad0** from the drop-down list
    - M3_CLK: **100 MHz**
    - MDDR Clocks: Select 3 from the drop-down menu to get an MDDR_CLK of **300 MHz**.

Note: You can see the clock and the different blocks it drives by clicking the clock name shown in Blue color. For example, click the MDDR_CLK shown in Figure 10 on page 15 and the clock and blocks that the clock is driving are displayed on the right-side panel.

Figure 10 • **System Builder Configurator - Clock Settings Page**



System Clock
100.00 MHz
Dedicated Input Pad 0

Cortex-M3 and MSS Main Clock
M3_CLK = 100.00 MHz 100.000

MDDR Clocks
MDDR_CLK = M3_CLK * 3 300.000
DDR/SMC_FIC_CLK = MDDR_CLK / 1

MSS APB_0/1 Clocks
APB_0_CLK = M3_CLK / 1 100.000
APB_1_CLK = M3_CLK / 1 100.000

Fabric Interface Clocks
FIC_0_CLK = M3_CLK / 1
AHBLite Bypass Mode
FIC_1_CLK = M3_CLK / 1
AHBLite Bypass Mode

Fabric DDR Clocks
FDDR_CLK = 100 MHz
FDDR_SUBSYSTEM_CLK = FDDR_CLK / 1

14. Click **Next**, the System Builder - **Microcontroller Options** page is displayed.
    - Leave all the default selections
15. Click **Next**, the System Builder - **SECDED Options** page is displayed.
    - Leave all the default selections
16. Click **Next**, the System Builder - **Security Options** page is displayed.
    - Leave all the default selections
17. Click **Next**, the System Builder - **Interrupts Options** page is displayed.
    - Leave all the default selections
18. Click **Next**, the System Builder - **Memory Map Options** page is displayed.
    - Leave all the default selections
19. Click **Finish**.

The System Builder generates the system based on the selected options.

The System Builder block is created and added to Libero SoC project, as shown in Figure 11 on page 16. The CoreResetP and CoreConfigP cores are automatically instantiated and connected by the System Builder. See "Opening System Builder Component as SmartDesign" on page 17 for more information on how the blocks are connected in the System Builder component.

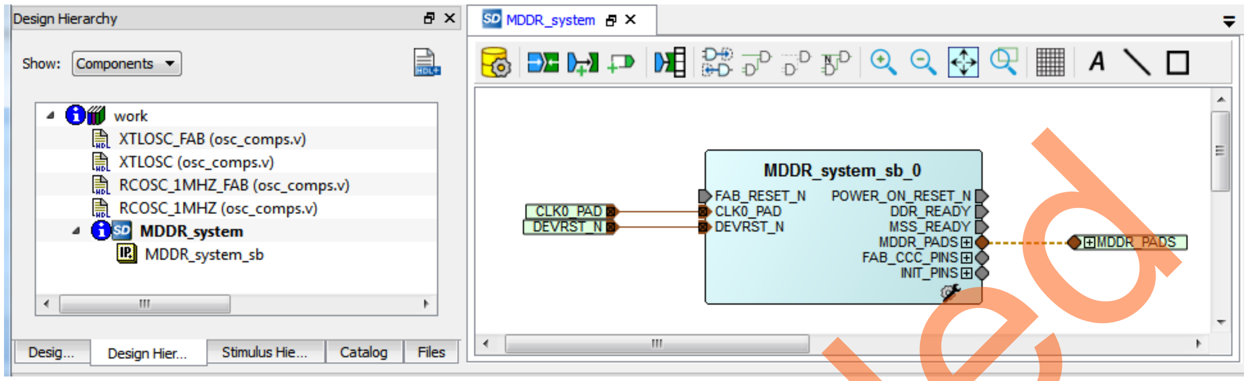


Figure 11 • SmartFusion2 SoC FPGA System Builder Generated System

20. Connect the pins as follows:
    - Tie the **FAB_RESET_ N** to high by right-clicking and selecting **Tie High**. This is an active low reset input that comes from the user logic in the fabric. In this tutorial, as you are not using this signal so you can tie it High.
    - Mark the output port **POWER_ON_RESET_N** as unused by right-clicking and selecting **Mark Unused**.
    - Mark the output port **MSS_READY** as unused by right-clicking and selecting **Mark Unused**.
    - Mark the output port **DDR_READY** as unused by right-clicking and selecting **Mark Unused**.
    - Mark the output port **FAB_CCC_GL0** (part of FAB_CCC_PINS group) as unused by right-clicking and selecting **Mark Unused**.
    - Mark the output port **FAB_CCC_LOCK** (part of FAB_CCC_PINS group) as unused by right-clicking and selecting **Mark Unused**.
    - Mark the output port **INIT_DONE** (part of INIT_PINS group) as unused by right-clicking and selecting **Mark Unused**.
21. Generate the final system by clicking **SmartDesign > Generate Component** or by clicking **Generate Component** icon on the SmartDesign toolbar.
22. Right-click **canvas** and select **Generate Component**, as shown in Figure 12 on page 16. After successful generation of the system, the message "Info: 'MDDR_system' was successfully generated.

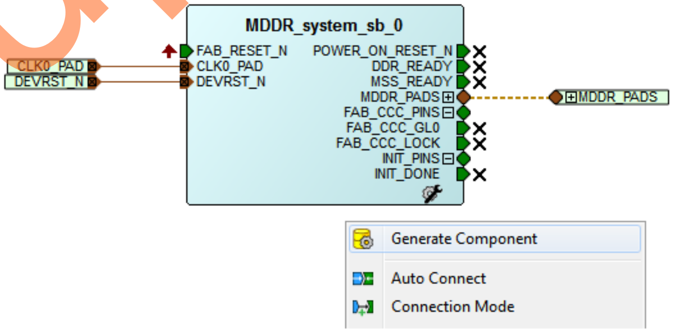


Figure 12 • SmartFusion2 SoC FPGA Generated Final System

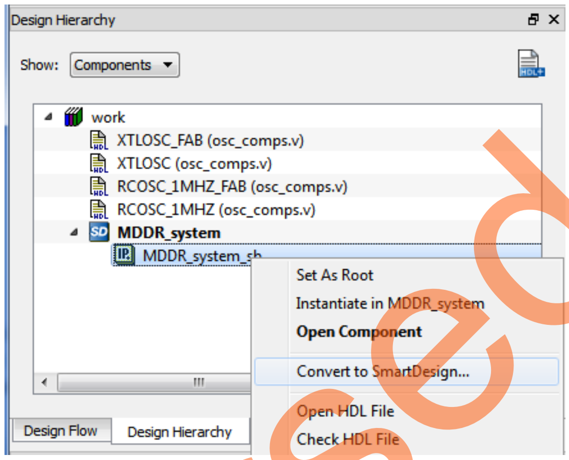
1.4.1 Opening System Builder Component as SmartDesign

Upon system generation, the System Builder configures, connects, and generates the entire MDDR system including all the required blocks such as the MSS, clocks, CoreConfigP, and CoreResetP.

The final System Builder generated system is shown in Figure 12 on page 16. You can dive (convert to SmartDesign) into that block to see the individual blocks that make-up the entire design. To do so, open the System Builder generated block using the SmartDesign. It enables you to check the internals of the overall design. To open the MDDR_system using the SmartDesign, use the following steps:

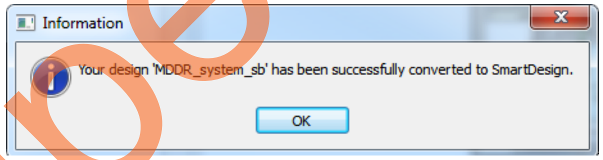
1. In the **Design Hierarchy**, expand **MDDR_system** component.
2. Right-click **MDDR_system** and select **Convert to SmartDesign**, as shown in Figure 13 on page 17.

Figure 13 • **Open as SmartDesign Option**



The system is converted to a SmartDesign component and the message is displayed, as shown in Figure 14 on page 17.

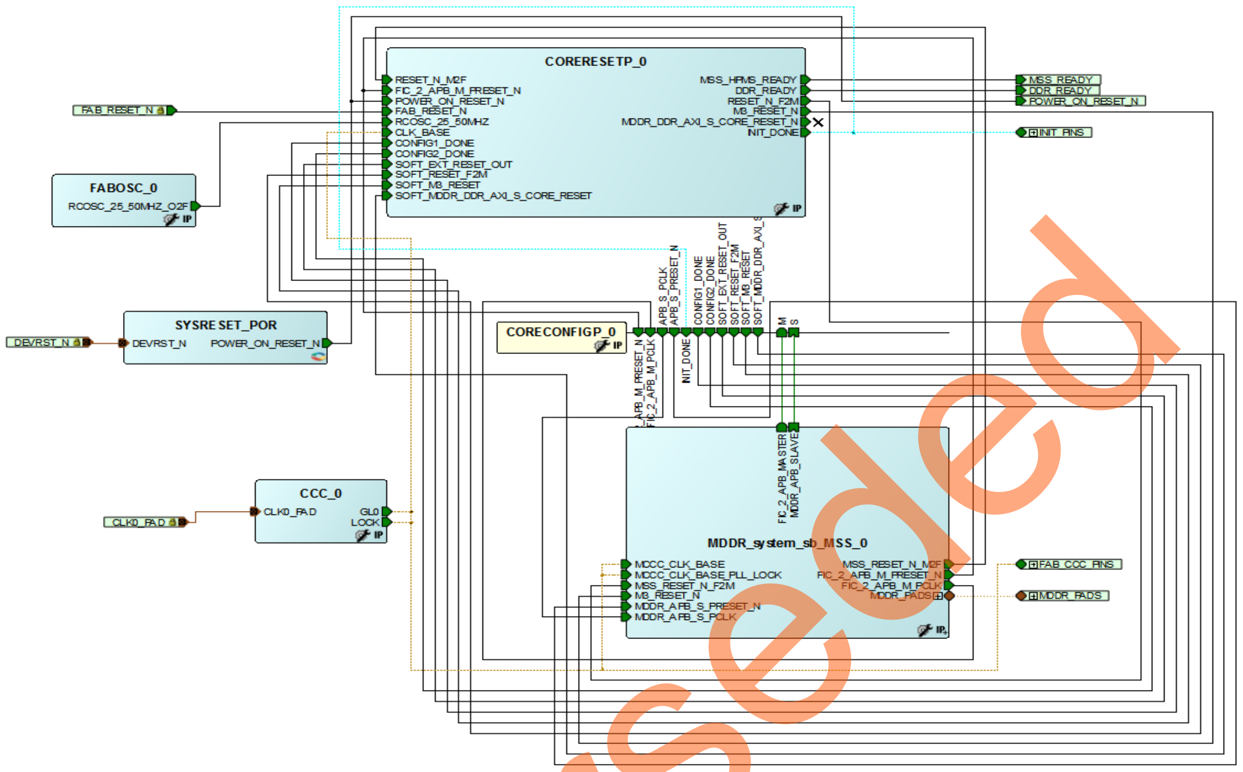
Figure 14 • **Successful Conversion of System Builder to a SmartDesign Message**



Your design 'MDDR_system_sb' has been successfully converted to SmartDesign.

3. Click **OK**. The system is shown in the SmartDesign canvas, as shown in Figure 15 on page 18.

Figure 15 • **System Builder Generated System Opened in the SmartDesign**



The System Builder is automatically instantiated and connected different blocks based on the different options that you have selected in the different pages of the System Builder.

- **SYSRESET_POR**: It generates the power-on reset signal for the CoreResetP block.
- **CORERESETP_0** (soft core): It is responsible for managing all the reset mechanism needed for the system.
- **FABOSC_0**: It generates the clock source for the CoreResetP block.
- **CCC_0**: It is used to generate the clock source for the MSS_CCC MCC_CLK_BASE reference. The MSS_CCC, which is part of the MSS, gets the reference clock from the Fabric CCC (CCC_0).
- **CORECONFIGP_0** (soft core): It manages the configuration aspect of the controller based on the specified configuration file.

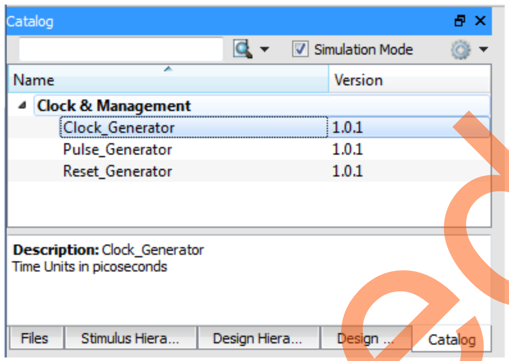
1.5 Step 2: Generating the Testbench

The following steps describe how to create a testbench for the design using the SmartDesign Testbench Generator:

1. Enable the SmartDesign simulation cores by selecting **Simulation Mode** check box in the Libero SoC IP catalog, as shown in Figure 16 on page 19. Under the **Clock & Management**, the IP catalog displays three simulation cores to drive the device under test (DUT):
   - Clock_Generator
   - Pulse_Generator
   - Reset_Generator

Note: If they appear in italic, double-click to download the cores to your local vault.

Figure 16 • **Simulation Cores in Libero IP Catalog**

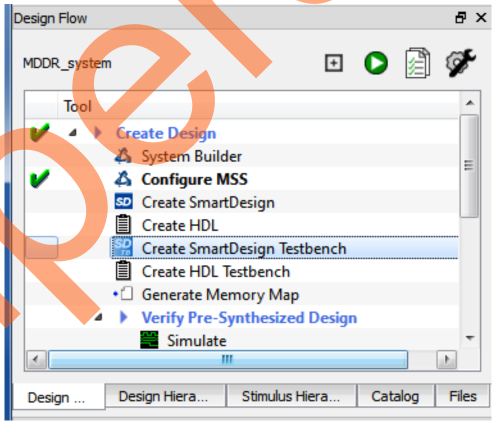


Name	Version
Clock & Management	
Clock_Generator	1.0.1
Pulse_Generator	1.0.1
Reset_Generator	1.0.1

Description: Clock_Generator
Time Units in picoseconds

2. Double-click **Create SmartDesign Testbench** in the **Libero Design Flow** window, as shown in Figure 17 on page 19.

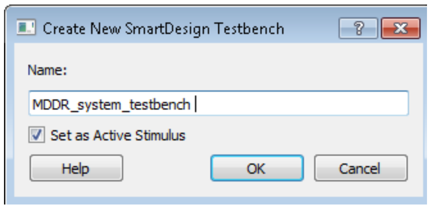
Figure 17 • **Opening SmartDesign Testbench**



The **Create New SmartDesign Testbench** dialog box is displayed, as shown in Figure 18 on page 20.

3. Enter **MDDR_system_testbench** in the **Create New SmartDesign Testbench** dialog box and then click **OK**.

Figure 18 • **Create New SmartDesign Testbench Dialog Box**

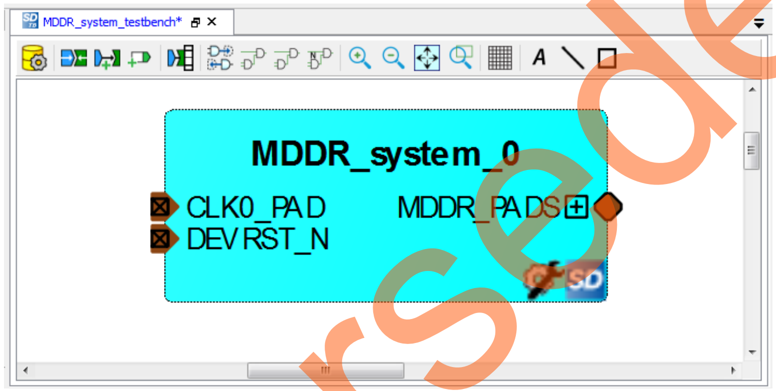


Name: MDDR_system_testbench

☑ Set as Active Stimulus

The SmartDesign canvas is displayed with the MDDR_system_0 component instantiated, as shown in Figure 19 on page 20.

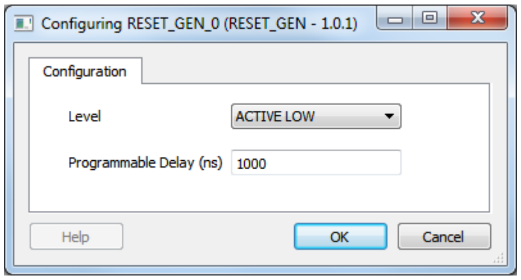
Figure 19 • **SmartDesign Testbench Canvas**



4. Drag the **Clock_Generator** and **Reset_Generator** simulation cores from the IP catalog to the MDDR_system_testbench SmartDesign canvas.
5. Open the **Reset_Generator** configurator by double-clicking **RESET_GEN_0** in the SmartDesign canvas. Ensure that the following information, as shown in Figure 20 on page 20, is set in the RESET_GEN_0 configurator and then click **OK**:
   - Level: ACTIVE LOW (default)
   - Programmable Delay (ns): 1000

The reset generator provides the reset pulse for the simulation.

Figure 20 • **RESET_GEN Configuration**



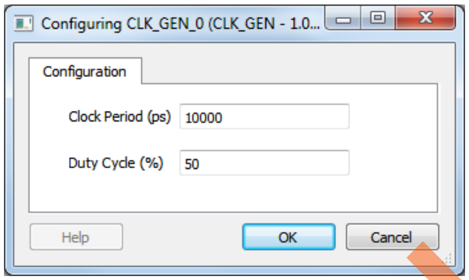
Level: ACTIVE LOW

Programmable Delay (ns): 1000

6. Open the **Clock_Generator** configurator by double-clicking the CLK_GEN_0 in the SmartDesign canvas. Ensure that the following information, as shown in Figure 21 on page 21, is set in the CLK_GEN_0 configurator and then click **OK**:
   - Clock Period (ps): 10000
   - Duty Cycle (%): 50

The clock generator period is set to 10000 ps to generate this 100 MHz clock, as shown in Figure 21 on page 21. The System Clock 10000 ps equal is to 100 MHz.

Figure 21 • **CLK_GEN Configuration**



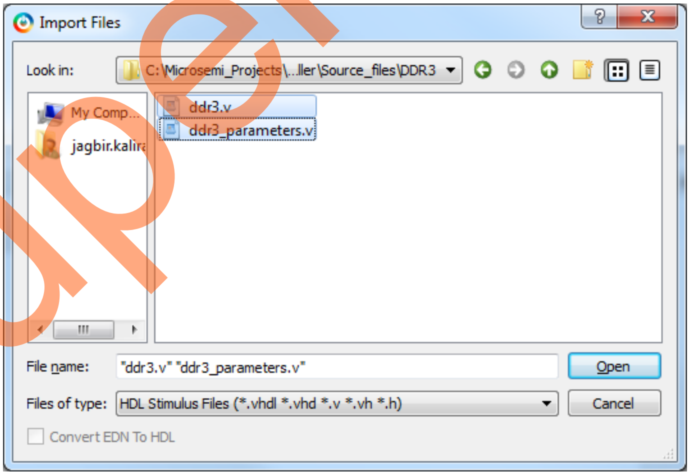
Clock Period (ps) 10000

Duty Cycle (%) 50

7. Import the provided DDR3 models into the Libero SoC project and instantiate those models into the testbench that you created in the previous steps. The DDR3 model must be imported as **Stimulus** files as follows:
   - **File > Import Files > HDL Stimulus Files**. This opens the **Import Files** dialog box
   - Select **HDL Stimulus Files** (*.vhd *.v) option from the **Files of type**, as shown in Figure 22 on page 21.
   - Select the provided **ddr3.v** and the **ddr3_parameters.v** files and then click **Open**, as shown in Figure 22 on page 21.

The files are located under the *<project directory>\m2s_tu0372_ddr3_mddr_liberov11p7_df\Source_files\DDR3_folder.*

Figure 22 • **Import the DDR3 Models as Stimulus Files**



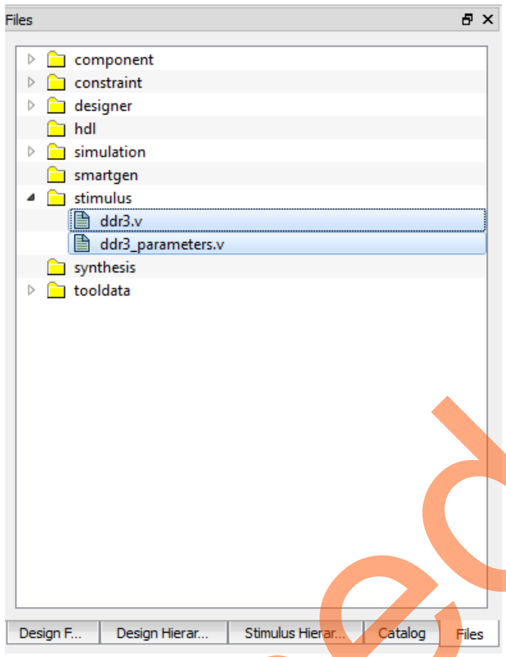
Look in: C:\Microsemi_Projects\...ller\Source_files\DDR3

File name: "ddr3.v" "ddr3_parameters.v"

Files of type: HDL Stimulus Files (*.vhdl *.vhd *.v *.vh *.h)

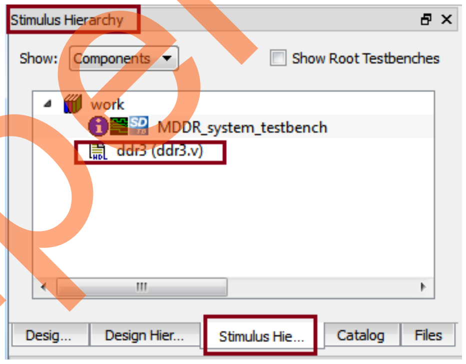
Verify that the files are imported correctly as stimulus files by checking under the Stimulus folder in the Files tab, as shown in Figure 23 on page 22.

Figure 23 • **Imported DDR3 in Stimulus Folder**



When the file is imported as a Stimulus, the file is displayed in the **Stimulus Hierarchy** window, as shown Figure 24 on page 22.

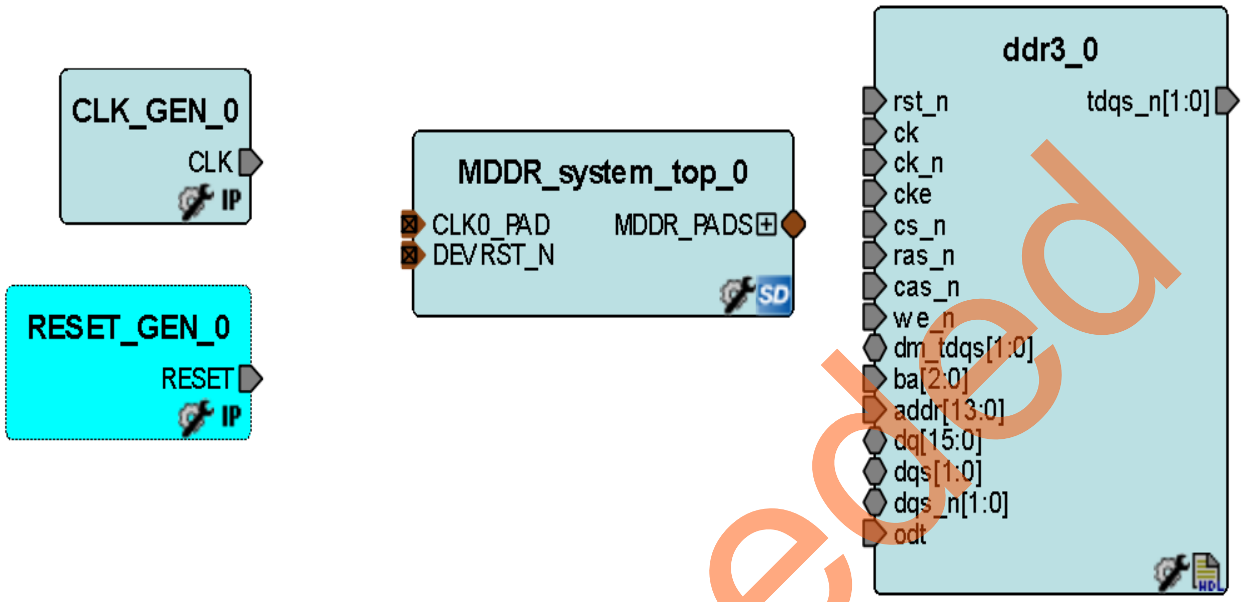
Figure 24 • **Stimulus Hierarchy Window**



8. From the **Stimulus Hierarchy** window, drag the **ddr3** file into the **MDDR_system_testbench** canvas.

You are basically instantiating the DDR3 models into the testbench to emulate an external DDR3 memory. You are going to simulate the write and read from the DDR3 using the Cortex-M3 processor as the master through the MDDR controller in the MSS. After you instantiate the DDR3, the canvas is displayed, as shown in Figure 25 on page 23.

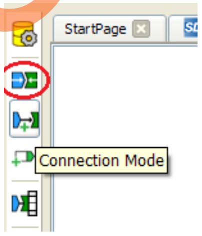
Figure 25 • **System Testbench Canvas with DDR3 Models Instantiated**



The next step is to connect all the blocks on the testbench canvas. There are two different ways to make the connections. The first method is by using the **Connection Mode** option.

To use the Connection Mode method, change the SmartDesign to connection mode by clicking the **Connection Mode** on the SmartDesign toolbar, as shown in Figure 26 on page 23. The cursor changes from the normal arrow shape to the connection mode icon shape. To make a connection in this mode, click the first pin and drag-drop to the second pin that you want to connect.

Figure 26 • **Enabling the Connection Mode Option**



The second method to connect is, by selecting the pins to be connected together, right-click and select **Connect**.

To select multiple pins to be connected, select a pin, hold down the CTRL key while selecting the other pins, right-click the input source pin, and select **Connect** to connect all the pins together. In the same way, select the input source pin, right-click, and select **Disconnect** to disconnect the signals already connected.

9. Using whichever connection method described above, make the following connections in the SmartDesign canvas between the **RESET_GEN_0**, **CLK_GN_0** and the **MDDR_system_0**:
   - From **RESET_GEN_0**: **RESET** to **MDDR_system_0**: **DEVRST_N**
   - From **CLK_GEN_0:CLK** to **MDDR_system_0:CLK0_PAD**

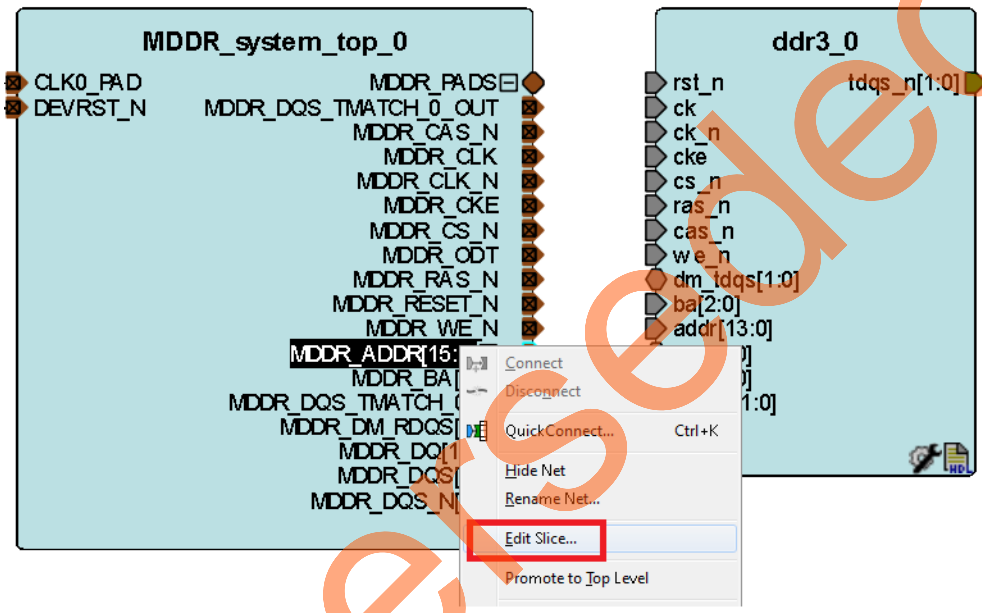
There are buses on the MDDR_system_0 and the ddr3_0 that do not match in width. To connect those buses, you need to slice them first to create an equivalent bus width that matches between the MDDR_system_0 and the ddr3_0.

For example, the MDDR_ADDR [15:0] is a 16 bits bus while the addr[13:0] on ddr3_0 is a 14 bits bus. To connect these two, MDDR_ADDR[15:0] needs to be sliced into two slices. The first slice is MDDR_ADDR [13:0] and the second slice is MDDR_ADDR [15:14]. After doing the slicing, connect MDDR_ADDR [13:0] to addr[13:0] and mark the MDDR_ADDR[15:14] as **Unused**.

To slice a bus, use the following steps:

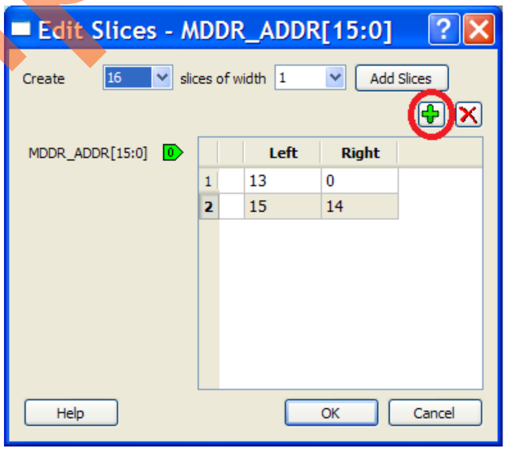
1. Right-click bus and select **Edit Slice**, as shown in Figure 27 on page 24. The dialog box is displayed, as shown in Figure 28 on page 24.

Figure 27 • **Creating a Bus Slice**



2. Click icon (highlighted in Figure 28 on page 24) to add a slice. Double-click icon to add two slices. Then add the slices, as shown in Figure 28 on page 24.

Figure 28 • **Edit Slices Dialog Box**


Edit Slices - MDDR_ADDR[15:0]

Create 16 slices of width 1 Add Slices

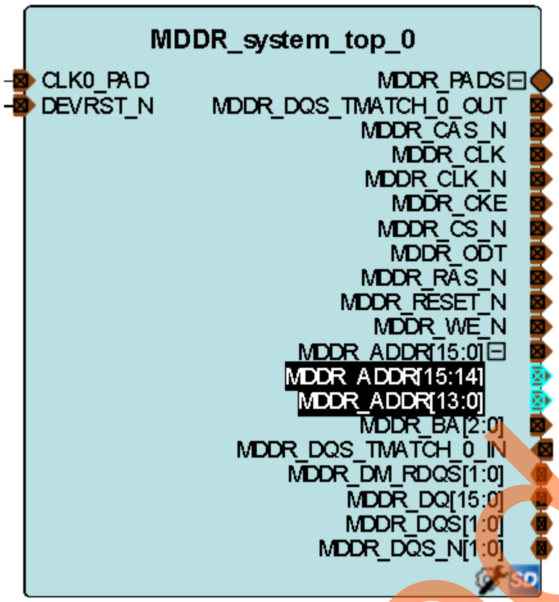
MDDR_ADDR[15:0]

	Left	Right
1	13	0
2	15	14

Help OK Cancel

3. Click **OK**. This creates two slices of the MDDR_ADDR bus, as shown in Figure 29 on page 25.

Figure 29 • **MDDR_ADDR Created Slices**



4. Expand the **MDDR_system_0: MDDR_PADS**. Using whichever connection method described, make the following connections in the SmartDesign canvas between the **MDDR_system_0** and the **ddr3_0**:
   - Connect **MDDR_DQS_TMATCH_0_IN** to **MDDR_DQS_TMATCH_0_OUT** of the **MDDR_system_0** block.
   - Connect the rest of the pins as shown in Table 2 on page 25.

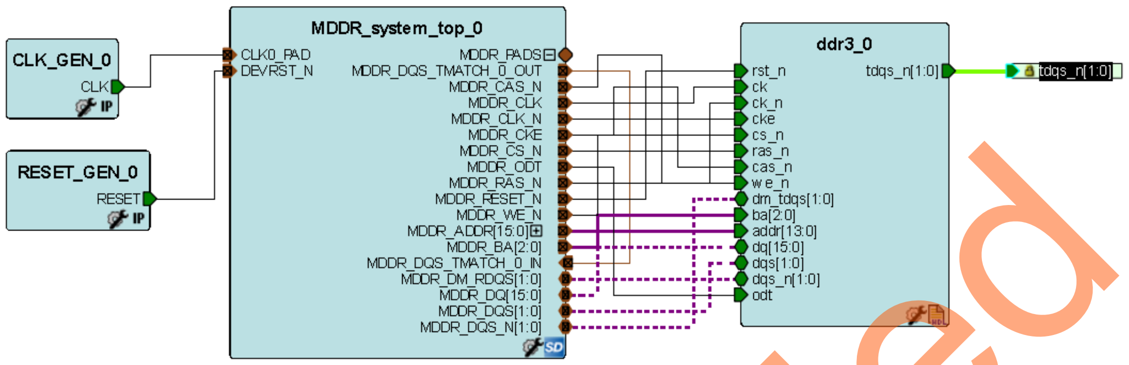
Table 2 • **DDR3 Pins Connections**

MDDR_System_0_Pins	**DDR3_0_Pins**
MDDR_CAS_N	cas_n
MDDR_CKE	cke
MDDR_CLK	ck
MDDR_CLK_N	ck_n
MDDR_CS_N	cs_n
MDDR_ODT	odt
MDDR_RAS_N	ras_n
MDDR_RESET_N	rst_n
MDDR_WE_N	we_n
MDDR_BA[2:0]	ba[2:0]
MDDR_DM_RDQS[1:0]	dm_tdqs[1:0]
MDDR_DQ[15:0]	dq[15:0]
MDDR_DQS[1:0]	dqs[1:0]
MDDR_DQS_N[1:0]	dqs_n[1:0]
MDDR_ADDR[13:0]	Addr[13:0]
MDDR_ADDR[15:14]	Mark Unused

5. Promote **tdqs_n[1:0] of ddr3_0** instance to top by right-clicking on the pin and selecting **Promote to Top Level**.

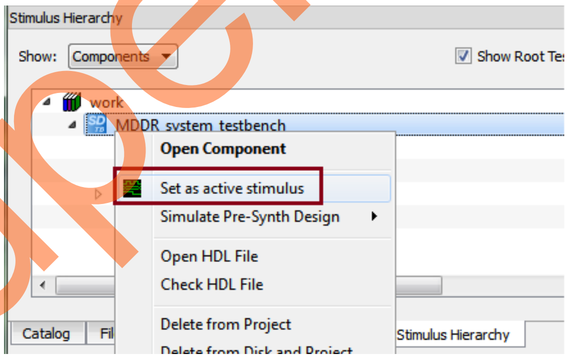
After making all the connections, the canvas is displayed, as shown in Figure 30 on page 26.

Figure 30 • **Fully Connected MDDR System**



6. Generate the final system testbench by clicking **SmartDesign > Generate Component** or by clicking **Generate Component** icon on the SmartDesign toolbar. You can also right-click on the canvas and select **Generate Component**.
On successful generation, the message "Info: 'MDDR_system_testbench' was successfully generated" is displayed on the log window:
7. After generating the testbench, you need to make it Active testbench (if it is not already set). By doing so, you are specifying the testbench that should be used for simulation. To set the testbench as the active testbench, use the following steps:
   a. Go to the **Stimulus Hierarchy** tab
   b. If not already set, right-click **MDDR_system_testbench** and select **Set as active stimulus**, as shown in Figure 31 on page 26. The **MDDR_system_testbench** is already the active stimulus as highlighted in Figure 31 on page 26, If the **Set as active stimulus** is not displayed.

Figure 31 • **Setting a Testbench as Active Testbench**

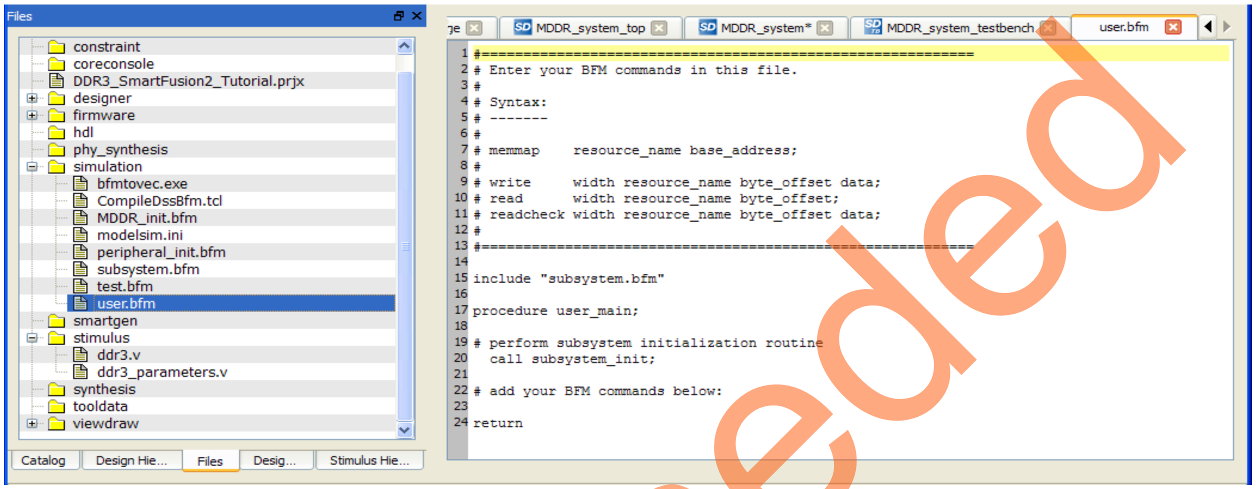


1.6 Step 3: Modifying the BFM Scripts

The following steps describe how to modify the BFM script (user.bfm) file that is generated by the SmartDesign. The BFM script file simulates Cortex-M3 processor writing to /reading from the DDR3 model through the MDDR.

1. Open the user.bfm file. To open the user.bfm, go to the **Files tab > Simulation folder**, double-click **user.bfm**. The user.bfm file is shown, as shown in Figure 32 on page 27.

Figure 32 • **SmartDesign Generated user.bfm File**



```
#============================================================
# Enter your BFM commands in this file.
#
# Syntax:
# -------
#
# memmap    resource_name base_address;
#
# write     width resource_name byte_offset data;
# read      width resource_name byte_offset;
# readcheck width resource_name byte_offset data;
#
#============================================================

include "subsystem.bfm"

procedure user_main;

# perform subsystem initialization routine
  call subsystem_init;

# add your BFM commands below:

return
```

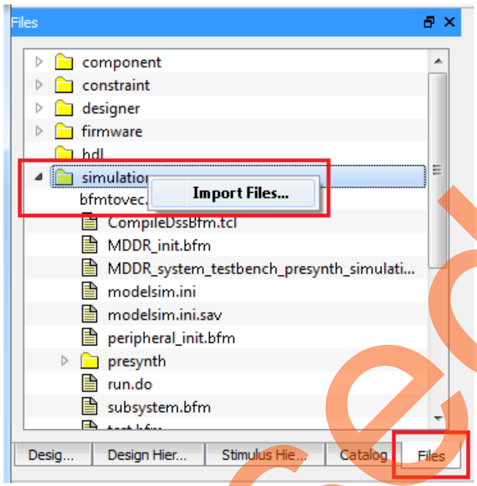
2. Modify the user.bfm to add the following bfm commands of writing and reading and click **Save**.

```
# add your BFM commands below:


# DDR memory map

a. memmap M_MDDR0_SPACE_0 0xA0000000;


print "TEST STARTS";

b.#write different values to different location

write w M_MDDR0_SPACE_0 0x0000 0xA1B2C3D4;

write w M_MDDR0_SPACE_0 0x0004 0x10100101;

write w M_MDDR0_SPACE_0 0x0008 0xD7D7E1E1;

write w M_MDDR0_SPACE_0 0x000C 0xA5DEF6E7;

write w M_MDDR0_SPACE_0 0x0010 0xABCDEF01;

write w M_MDDR0_SPACE_0 0x0014 0xCCBBAADD;


c. #read check what you wrote in step#b above

readcheck w M_MDDR0_SPACE_0 0x0000 0xA1B2C3D4;

readcheck w M_MDDR0_SPACE_0 0x0004 0x10100101;

readcheck w M_MDDR0_SPACE_0 0x0008 0xD7D7E1E1;

readcheck w M_MDDR0_SPACE_0 0x000C 0xA5DEF6E7;
readcheck w M_MDDR0_SPACE_0 0x0010 0xABCDEF01;

readcheck w M_MDDR0_SPACE_0 0x0014 0xCCBBAADD;

print "TEST ENDS";
```

Note: An updated user.bfm file is included in the source files folder (*<project directory>\DDR3_SmartFusion2_Tutorial\Source_files*). You can import this file instead of manually modifying the user.bfm file as follows:

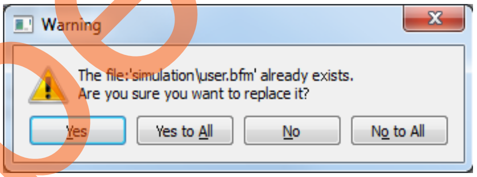
3. Go to **Files** tab and right-click **simulation** Folder as shown in Figure 33 on page 28.

Figure 33 • **Importing bfm Source File**



4. Browse to *<project directory>\m2s_tu0372_ddr3_mddr_liberov11p7_df\Source_files* and select **user.bfm** and select **Open**.
5. A warning message is displayed, as shown in Figure 34 on page 28. Click **Yes**.

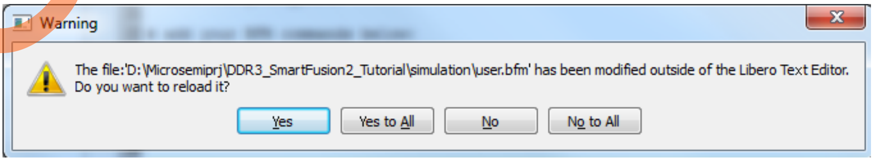
Figure 34 • **Replacing Existing user.bfm File**


Warning

The file:'simulation\user.bfm' already exists.
Are you sure you want to replace it?

6. A warning message is displayed as shown in Figure 35 on page 28, if the `user.bfm` file is already opened in your Libero window. Click **Yes**. If the `user.bfm` is not opened already in your Libero window, the message does not show up.

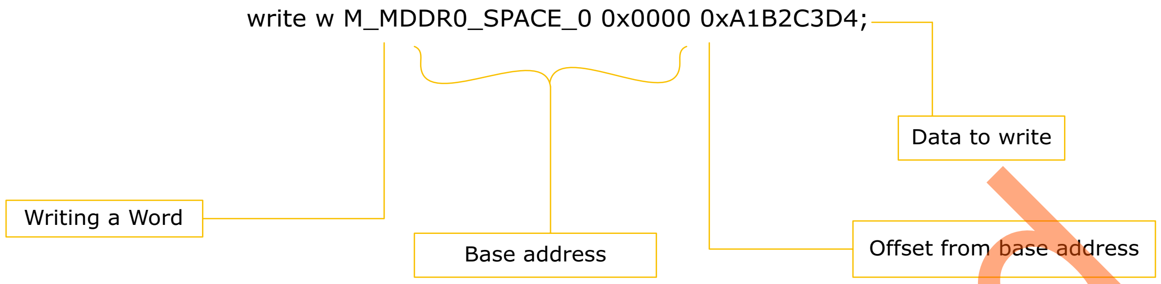
Figure 35 • **Reloading and Updating user.bfm File**


Warning

The file:'D:\Microsemiprj\DDR3_SmartFusion2_Tutorial\simulation\user.bfm' has been modified outside of the Libero Text Editor.
Do you want to reload it?

The following is an explanation for the different steps that you added into the bfm above.

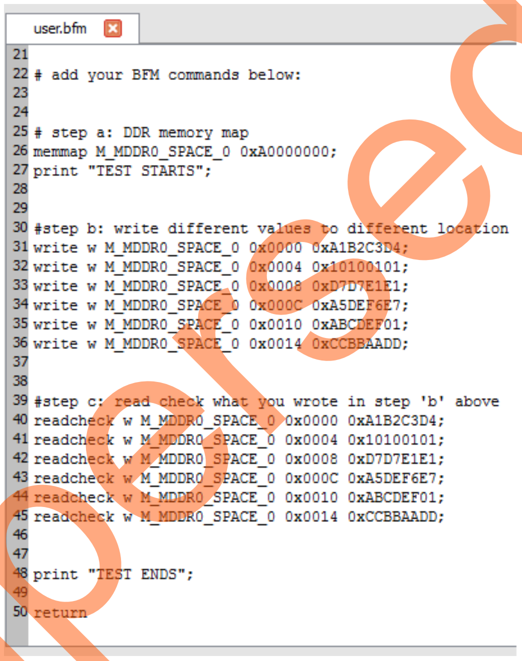
- **Step a**: In this step you are specifying the base address at which the MDDR is located. In this case it is 0xA0000000
- **Step b**: In this step you are writing different values to different locations. For example,



```
write w M_MDDR0_SPACE_0 0x0000 0xA1B2C3D4;
```

- **Step c**: In this step you are checking what you wrote. The final user.bfm is displayed, as shown in Figure 36 on page 29.

Figure 36 • **user.bfm after Adding the Commands**



```

# add your BFM commands below:


# step a: DDR memory map
memmap M_MDDR0_SPACE_0 0xA0000000;
print "TEST STARTS";


#step b: write different values to different location
write w M_MDDR0_SPACE_0 0x0000 0xA1B2C3D4;
write w M_MDDR0_SPACE_0 0x0004 0x10100101;
write w M_MDDR0_SPACE_0 0x0008 0xD7D7E1E1;
write w M_MDDR0_SPACE_0 0x000C 0xA5DEF6E7;
write w M_MDDR0_SPACE_0 0x0010 0xABCDEF01;
write w M_MDDR0_SPACE_0 0x0014 0xCCBBAADD;


#step c: read check what you wrote in step 'b' above
readcheck w M_MDDR0_SPACE_0 0x0000 0xA1B2C3D4;
readcheck w M_MDDR0_SPACE_0 0x0004 0x10100101;
readcheck w M_MDDR0_SPACE_0 0x0008 0xD7D7E1E1;
readcheck w M_MDDR0_SPACE_0 0x000C 0xA5DEF6E7;
readcheck w M_MDDR0_SPACE_0 0x0010 0xABCDEF01;
readcheck w M_MDDR0_SPACE_0 0x0014 0xCCBBAADD;


print "TEST ENDS";

return
```

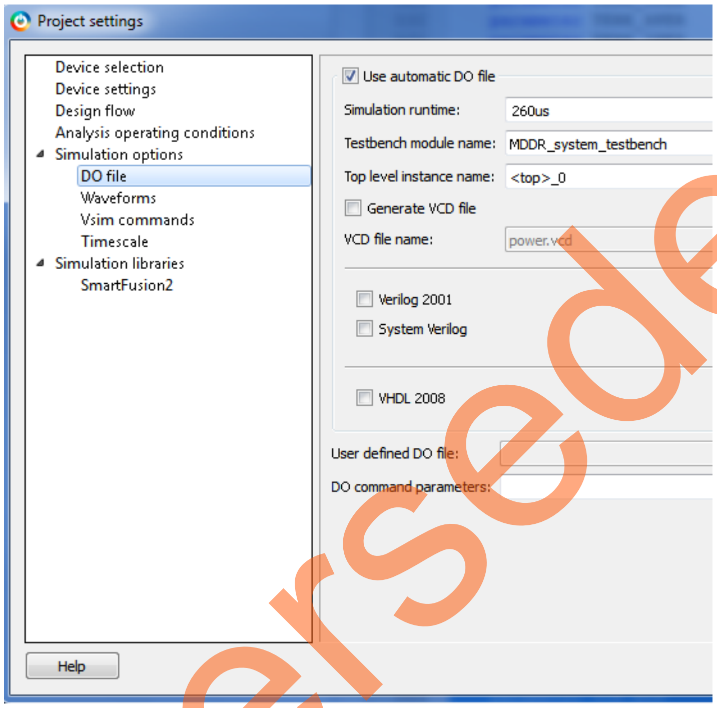
See the *DirectCore Advanced Microcontroller Bus Architecture - Bus Functional Model User Guide* for more details.

1.7 Step 4: Simulating the Design

The following steps describe how to simulate the design using the SmartDesign testbench and BFM script files:

1. Navigate to **Project > Project Settings** to open the Libero SoC project settings.
2. Select **Do File** under **Simulation Options** in the **Project Settings** window. Change the **Simulation runtime** to 260 µs, as shown in Figure 37 on page 30.

Figure 37 • **Project Setting - Do File Simulation Runtime Setting**

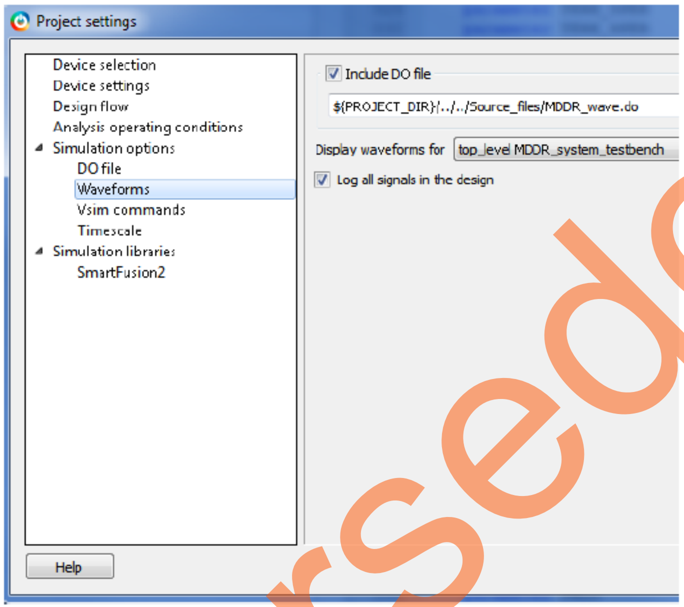


[✓] Use automatic DO file
Simulation runtime: 260us
Testbench module name: MDDR_system_testbench
Top level instance name: <top>_0
[ ] Generate VCD file
VCD file name: power.vcd
[ ] Verilog 2001
[ ] System Verilog
[ ] VHDL 2008
User defined DO file:
DO command parameters:

3. Select **Waveforms** under Simulation Options:
   - Select **Include DO file**, browse to where you extracted the provided source files, and select **MDDR_wave.do** file, as shown in Figure 38 on page 31. In this file, the list of signals that are required is already selected so you can check for the expected results.
   - Select **Log all signals in the design**.
   - Click **Close** to close the Project settings dialog box.
   - Select **Save** when prompted to save the changes.

Note: The path is modified to use ${PROJECT_DIR} so that the path is always relative to the project directory.

Figure 38 • **Project Setting - Specifying the MDDR_wave.do File Location**



☑ Include DO file

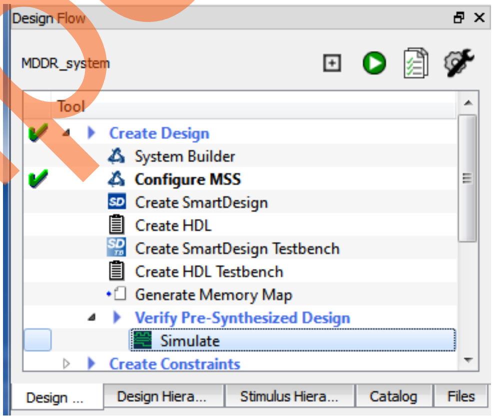
${PROJECT_DIR}/../../Source_files/MDDR_wave.do

Display waveforms for top_level MDDR_system_testbench

☑ Log all signals in the design

4. Expand **Verify Pre-Synthesized Design** in the Design Flow window, as shown in Figure 39 on page 31. Double-click **Simulate** to launch ModelSim in GUI mode.

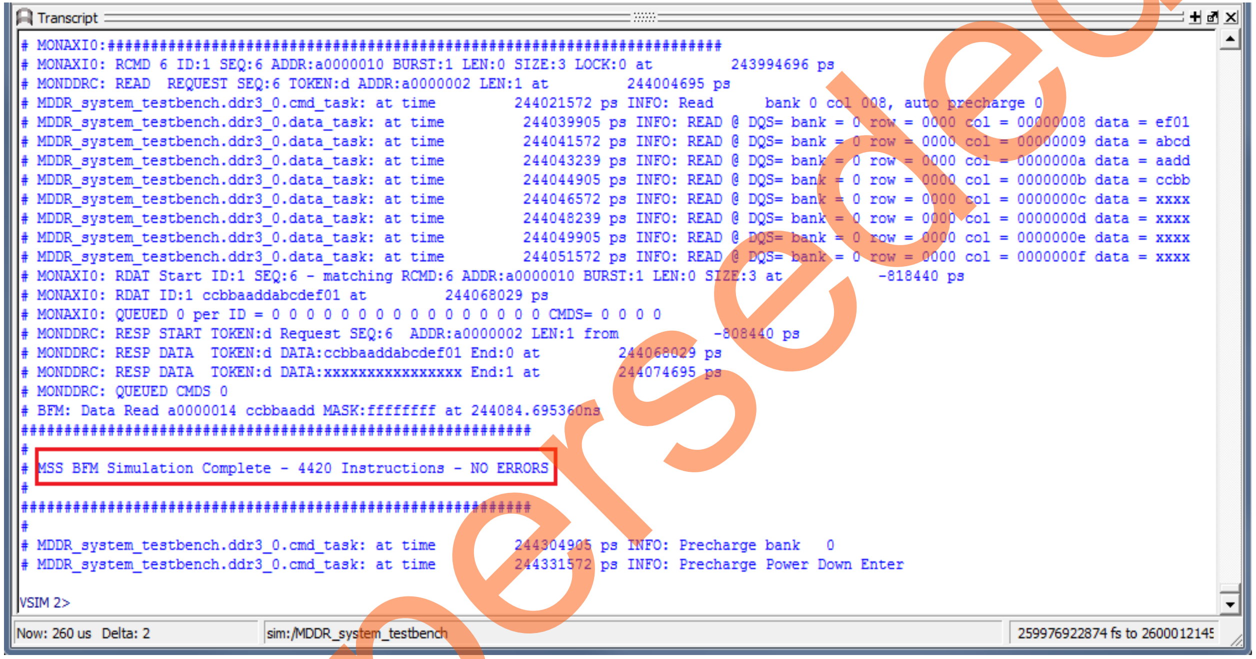
Figure 39 • **Starting Pre-Synthesis Simulation**



1.8 Step 5: Validating the Simulation Results

1. ModelSim runs the design for about 260 µs, as specified in the Project Settings window.
   It initializes the MDDR by writing a specific set of configuration options to the configurations registers. Once the configurations are written to the registers, then you can write to the DDR3 memory. The results are checked by the readcheck command. The external DDR3 memory must initialize before it can be used. This is done by adding the 200 us as specified in the System Builder-Memory page, as shown in Figure 8 on page 13. You can write and read from the external DDR.
2. The ModelSim transcript window displays the BFM commands and the BFM simulation completed with no errors, as shown in Figure 40 on page 32. Scroll in the window to see the different commands. In the BFM script provided in the user.bfm earlier, the readcheck command reads the data and verifies if the data read matches with the value provided along with the readcheck command. If the value read does not match, the simulation shows an error.

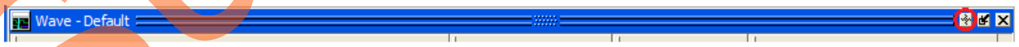
Figure 40 • **ModelSim BFM Simulation Transcript Results**



```
# MONAXI0:##########################################################################
# MONAXI0: RCMD 6 ID:1 SEQ:6 ADDR:a0000010 BURST:1 LEN:0 SIZE:3 LOCK:0 at         243994696 ps
# MONDDRC: READ  REQUEST SEQ:6 TOKEN:d ADDR:a0000002 LEN:1 at         244004695 ps
# MDDR_system_testbench.ddr3_0.cmd_task: at time         244021572 ps INFO: Read      bank 0 col 008, auto precharge 0
# MDDR_system_testbench.ddr3_0.data_task: at time         244039905 ps INFO: READ @ DQS= bank = 0 row = 0000 col = 00000008 data = ef01
# MDDR_system_testbench.ddr3_0.data_task: at time         244041572 ps INFO: READ @ DQS= bank = 0 row = 0000 col = 00000009 data = abcd
# MDDR_system_testbench.ddr3_0.data_task: at time         244043239 ps INFO: READ @ DQS= bank = 0 row = 0000 col = 0000000a data = aadd
# MDDR_system_testbench.ddr3_0.data_task: at time         244044905 ps INFO: READ @ DQS= bank = 0 row = 0000 col = 0000000b data = ccbb
# MDDR_system_testbench.ddr3_0.data_task: at time         244046572 ps INFO: READ @ DQS= bank = 0 row = 0000 col = 0000000c data = xxxx
# MDDR_system_testbench.ddr3_0.data_task: at time         244048239 ps INFO: READ @ DQS= bank = 0 row = 0000 col = 0000000d data = xxxx
# MDDR_system_testbench.ddr3_0.data_task: at time         244049905 ps INFO: READ @ DQS= bank = 0 row = 0000 col = 0000000e data = xxxx
# MDDR_system_testbench.ddr3_0.data_task: at time         244051572 ps INFO: READ @ DQS= bank = 0 row = 0000 col = 0000000f data = xxxx
# MONAXI0: RDAT Start ID:1 SEQ:6 - matching RCMD:6 ADDR:a0000010 BURST:1 LEN:0 SIZE:3 at         -818440 ps
# MONAXI0: RDAT ID:1 ccbbaaddabcdef01 at         244068029 ps
# MONAXI0: QUEUED 0 per ID = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 CMDS= 0 0 0 0
# MONDDRC: RESP START TOKEN:d Request SEQ:6  ADDR:a0000002 LEN:1 from         -808440 ps
# MONDDRC: RESP DATA  TOKEN:d DATA:ccbbaaddabcdef01 End:0 at         244068029 ps
# MONDDRC: RESP DATA  TOKEN:d DATA:xxxxxxxxxxxxxxxx End:1 at         244074695 ps
# MONDDRC: QUEUED CMDS 0
# BFM: Data Read a0000014 ccbbaadd MASK:ffffffff at 244084.695360ns
#############################################################
#
# MSS BFM Simulation Complete - 4420 Instructions - NO ERRORS
#
#############################################################
#
# MDDR_system_testbench.ddr3_0.cmd_task: at time         244304905 ps INFO: Precharge bank   0
# MDDR_system_testbench.ddr3_0.cmd_task: at time         244331572 ps INFO: Precharge Power Down Enter

VSIM 2>
Now: 260 us  Delta: 2     sim:/MDDR_system_testbench     259976922874 fs to 260012145
```

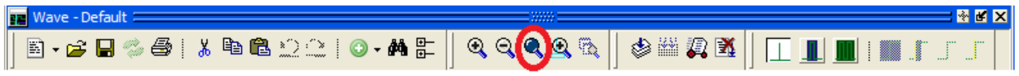
Once the simulation is run completely, undock the Wave window. The Wave window can be undocked by clicking the Dock/Undock icon on the Wave window, as shown in Figure 41 on page 32.

Figure 41 • **Doc or Undock of ModelSim Wave Window**



3. Once the Wave window is undocked, click the Zoom Full icon as shown in Figure 42 on page 32 to fit all the waveforms in the single view.

Figure 42 • **Zoom Full Option**



4. Place the cursor around 233 ps on the Wave window and click the Zoom In on the Active Cursor icon as shown in Figure 43 on page 33, to zoom in at that location. That shows the time at which the data was written/read-back to/from the DDR3 external modules, as shown in Figure 44 on page 33.

Figure 43 • **Zoom In on Active Cursor Button**

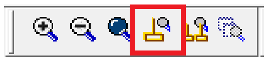
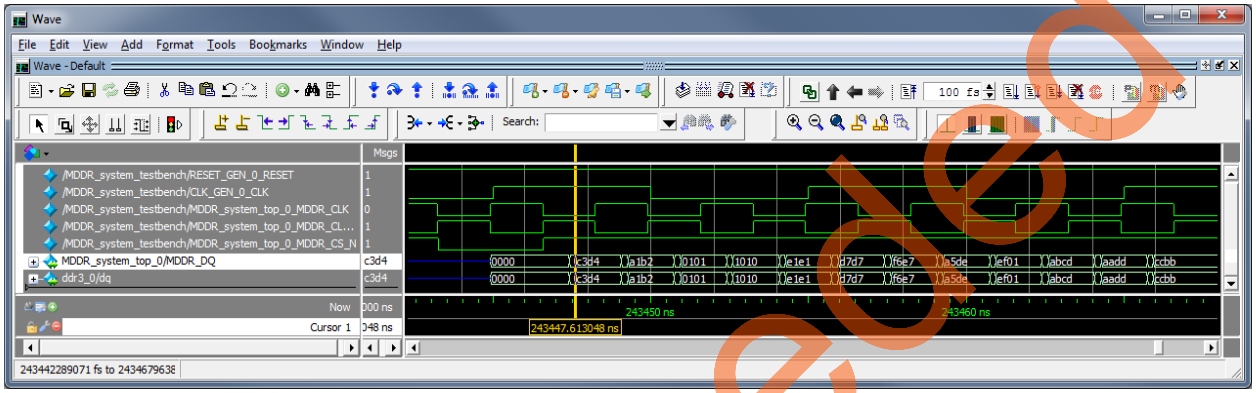


Figure 44 on page 33 shows the time at which the data was written/read-back to/from the DDR3 external modules. That concludes this tutorial.

Figure 44 • **Write/Read Data**



	Msgs
/MDDR_system_testbench/RESET_GEN_0_RESET	1
/MDDR_system_testbench/CLK_GEN_0_CLK	1
/MDDR_system_testbench/MDDR_system_top_0_MDDR_CLK	0
/MDDR_system_testbench/MDDR_system_top_0_MDDR_CL...	1
/MDDR_system_testbench/MDDR_system_top_0_MDDR_CS_N	1
MDDR_system_top_0/MDDR_DQ	c3d4
ddr3_0/dq	c3d4
Now	000 ns
Cursor 1	048 ns

```
MDDR_DQ: 0000 c3d4 a1b2 0101 1010 e1e1 d7d7 f6e7 a5de ef01 abcd aadd ccbb
dq:      0000 c3d4 a1b2 0101 1010 e1e1 d7d7 f6e7 a5de ef01 abcd aadd ccbb
243450 ns    243460 ns
243447.613048 ns
243442289071 fs to 243467963
```

5. Quit the ModelSim simulator by selecting **File > Quit**.

1.9 Conclusion

In this tutorial, you created a new project in Libero SoC, configured the MDDR system using the System Builder to access an external DDR3 SDRAM memory through MDDR controller with the Cortex-M3 processor as master, created a testbench using the SmartDesign testbench generator, and connected the different blocks using the SmartDesign tools.

The design in Model*Sim* using AMBA BFM simulation is verified.

2 Appendix: VHDL Flow

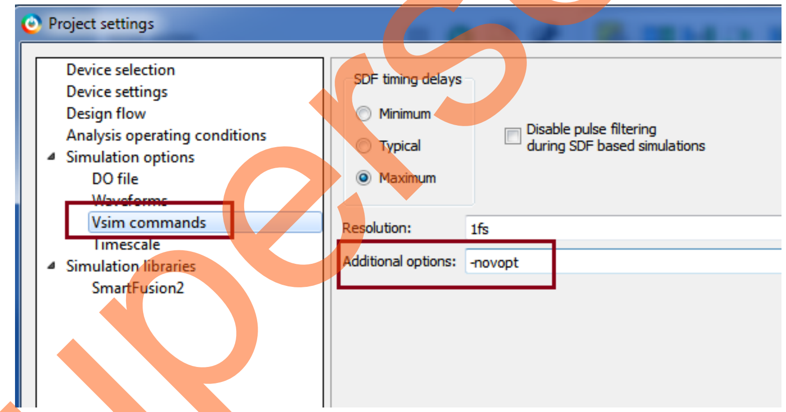
If you are designing with VHDL, since the DDR3 memory models used here are in Verilog, you need to use the ModelSim full version (for example, ModelSim SE) instead of ModelSim AE. ModelSim AE does not support mixed-language flows. Use the following steps to simulate VHDL design:

1. Copy the precompiled VHDL simulation library folder **SmartFusion2** from the Libero SoC install area (<Libero install> \Designer\lib\modelsim\precompiled\vhdl\) to a different folder on your disk (for example, E:\Microsemi_prj\)
2. Remove the **Read-Only** attribute from the **SmartFusion2** folder at the new location.

Note: The reason for steps 1 and 2 is that ModelSim full version needs to refresh the precompiled library. Those steps are to enable ModelSim full version to refresh the precompiled library and to ensure that the original precompiled library, which is installed with the Libero SoC, is unchanged.

3. Simulate with automatic design optimization option disabled (-novopt) and point to the new precompiled library location (for example, E:\Microsemi_prj\smartfusion2) in the **Project Settings** window as shown below:
   - Select **Project Settings** from the Project menu
   - Select **Vsim commands** under **Simulation Options**. Add the -novopt option into the **Additional options** field, as shown in Figure 45 on page 34. The -novopt option disables the automatic design optimization run.

Figure 45 • **Project Settings - Specifying - novopt Simulation Option**



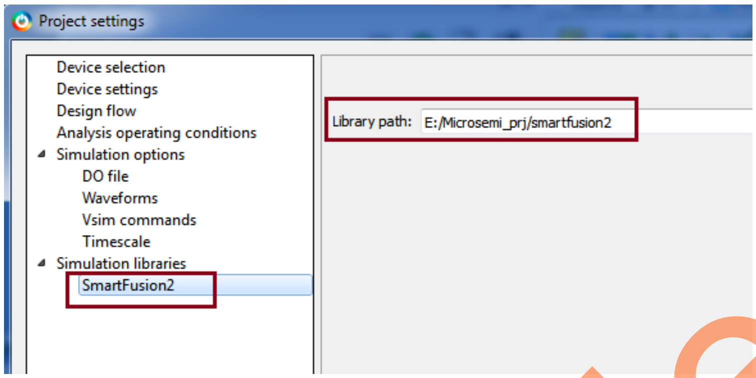
SDF timing delays: Minimum Typical [✓ Maximum]

Resolution: 1fs

Additional options: -novopt

- Select **SmartFusion2** under the **Simulation Libraries**
  - In the **Library path**, enter the new location where you copied the precompiled library (for example, E:/Microsemi_prj/smartfusion2), as shown in Figure 46 on page 35.

Figure 46 • **Project Settings - Specifying Precompiled Library Path**



Library path: E:/Microsemi_prj/smartfusion2

- Click **Save** to save the project settings and click **Close** to close the **Project Settings** window.

3 Revision History

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

Revision	Changes	Page
Revision 11 (February 2016)	Updated the document for Libero v11.7 software release (SAR 75916).	NA
Revision 10 (October 2015)	Updated the document for Libero v11.6 software release (SAR 68373).	NA
Revision 9 (January 2015)	Updated the document for Libero v11.5 software release (SAR 62935).	NA
Revision 8 (August 2014)	Updated the document for Libero v11.4 software release (SAR 59067).	NA
Revision 7 (March 2014)	Updated the document for Libero v11.3 software release (SAR 55761).	NA
Revision 6 (January 2014)	Updated the document for Libero v11.2 software release (SAR 53253).	NA
Revision 5 (April 2013)	Updated the document for 11.0 production SW release (SAR 46975).	NA
Revision 4 (February 2013)	Updated the document for Libero 11.0 Beta SP1 software release (SAR 44417).	NA
Revision 3 (November 2012)	Updated the document for Libero 11.0 Beta SPA software release (SAR 42888).	NA
Revision 2 (October 2012)	Updated the document for Libero 11.0 Beta launch (SAR 41898).	NA
Revision 1 (May 2012)	Updated the document for LCP2 software release (SAR 38956).	NA

4 Product Support

Microsemi SoC Products Group backs its products with various support services, including Customer Service, Customer Technical Support Center, a website, electronic mail, and worldwide sales offices. This appendix contains information about contacting Microsemi SoC Products Group and using these support services.

4.1 Customer Service

Contact Customer Service for non-technical product support, such as product pricing, product upgrades, update information, order status, and authorization.

From North America, call 800.262.1060
From the rest of the world, call 650.318.4460
Fax, from anywhere in the world, 408.643.6913

4.2 Customer Technical Support Center

Microsemi SoC Products Group staffs its Customer Technical Support Center with highly skilled engineers who can help answer your hardware, software, and design questions about Microsemi SoC Products. The Customer Technical Support Center spends a great deal of time creating application notes, answers to common design cycle questions, documentation of known issues, and various FAQs. So, before you contact us, please visit our online resources. It is very likely we have already answered your questions.

4.3 Technical Support

For Microsemi SoC Products Support, visit
http://www.microsemi.com/products/fpga-soc/design-support/fpga-soc-support.

4.4 Website

You can browse a variety of technical and non-technical information on the Microsemi SoC Products Group home page, at http://www.microsemi.com/products/fpga-soc/fpga-and-soc.

4.5 Contacting the Customer Technical Support Center

Highly skilled engineers staff the Technical Support Center. The Technical Support Center can be contacted by email or through the Microsemi SoC Products Group website.

4.5.1 Email

You can communicate your technical questions to our email address and receive answers back by email, fax, or phone. Also, if you have design problems, you can email your design files to receive assistance. We constantly monitor the email account throughout the day. When sending your request to us, please be sure to include your full name, company name, and your contact information for efficient processing of your request.

The technical support email address is soc_tech@microsemi.com.

4.5.2 My Cases

Microsemi SoC Products Group customers may submit and track technical cases online by going to My Cases.

4.5.3 Outside the U.S.

Customers needing assistance outside the US time zones can either contact technical support via email (soc_tech@microsemi.com) or contact a local sales office. Visit About Us for sales office listings and corporate contacts.

4.6 ITAR Technical Support

For technical support on RH and RT FPGAs that are regulated by International Traffic in Arms Regulations (ITAR), contact us via soc_tech@microsemi.com. Alternatively, within My Cases, select **Yes** in the ITAR drop-down list. For a complete list of ITAR-regulated Microsemi FPGAs, visit the ITAR web page.

Microsemi Corporate Headquarters
One Enterprise, Aliso Viejo,
CA 92656 USA

Within the USA: +1 (800) 713-4113
Outside the USA: +1 (949) 380-6100
Sales: +1 (949) 380-6136
Fax: +1 (949) 215-4996
E-mail: sales.support@microsemi.com

Microsemi Corporation (Nasdaq: MSCC) offers a comprehensive portfolio of semiconductor and system solutions for communications, defense & security, aerospace and industrial markets. Products include high-performance and radiation-hardened analog mixed-signal integrated circuits, FPGAs, SoCs and ASICs; power management products; timing and synchronization devices and precise time solutions, setting the world's standard for time; voice processing devices; RF solutions; discrete components; Enterprise Storage and Communication solutions; security technologies and scalable anti-tamper products; Ethernet solutions; Power-over-Ethernet ICs and midspans; as well as custom design capabilities and services. Microsemi is headquartered in Aliso Viejo, Calif, and has approximately 4,800 employees globally. Learn more at **www.microsemi.com**.

50200372-11/02.16