

---

# **Interfacing SmartFusion2 SoC FPGA with DDR3 Memory through MDDR Controller**

**Libero SoC v11.5 System Builder Flow Tutorial  
TU0372**

Superseded

---

# Table of Contents

---

## Interfacing SmartFusion2 SoC FPGA with DDR3 Memory through MDDR Controller ..3

Introduction.....	3
Tutorial Requirements.....	3
Design Overview .....	4
Step 1: Creating a Libero SoC Project.....	5
Step 2: Generating the Testbench .....	15
Step 3: Modifying the BFM Scripts.....	23
Step 4: Simulating the Design.....	27
Step 5: Validating the Simulation Results .....	29
Conclusion.....	30
Appendix A: VHDL Flow.....	30
Appendix B: Abbreviation Used .....	32

## List of Changes .....34

## Product Support.....35

Customer Service.....	35
Customer Technical Support Center.....	35
Technical Support .....	35
Website .....	35
Contacting the Customer Technical Support Center.....	35
ITAR Technical Support.....	36

# Interfacing SmartFusion2 SoC FPGA with DDR3 Memory through MDDR Controller

## Introduction

This tutorial describes how to create a hardware design using the System Builder to access an external DDR3 memory through the built-in hard ASIC MDDR controller in SmartFusion<sup>®</sup>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. Those controllers are the microcontroller subsystem (MSS) DDR (MDDR) and fabric DDR (FDDR) controllers. The MDDR controller is a hard ASIC block in the SmartFusion2 SoC FPGA. 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<sup>®</sup> Cortex<sup>™</sup>-M3 processor as a master that talks 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<sup>®</sup> System-on-Chip (SoC) v11.5 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<sup>®</sup> BFM simulation in MentorGraphics ModelSim<sup>®</sup> Simulator
- Using the ModelSim GUI to see the various design signals in ModelSim Waveform window

## Tutorial Requirements

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

**Table 1.** Tutorial Requirements and Details

Design Requirements	Description
<b>Software Requirements</b>	
Libero SoC	v11.5
Microcontroller subsystem (MSS)	v1.1.300
<b>Hardware Requirements</b>	
Host PC or Laptop	Any 64-bit Windows Operating System

## Project Files

The associated solution and source project files along with the readme for this tutorial can be downloaded from Microsemi<sup>®</sup> website:

[http://soc.microsemi.com/download/rsc/?f=m2s\\_tu0372\\_ddr3\\_mddr\\_liberov11p5\\_df](http://soc.microsemi.com/download/rsc/?f=m2s_tu0372_ddr3_mddr_liberov11p5_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.

## Design Overview

The design demonstrates the read/write access to an external slave DDR3 memory using the SmartFusion2 SoC FPGA. Inside the SmartFusion2 SoC FPGA, the Cortex-M3 processor acts as the master and performs the read/write transactions on the external slave memory. These read/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/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/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 CoreResetP 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. Refer to 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 shows the different blocks used in this design.

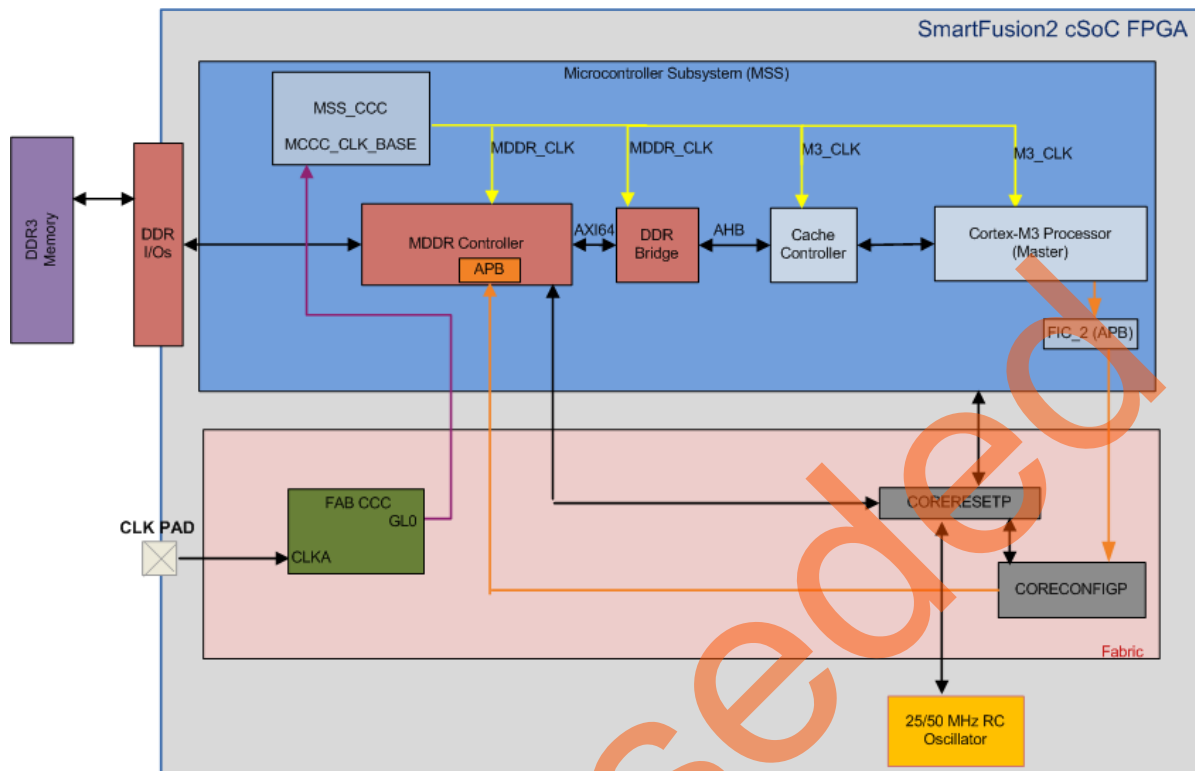
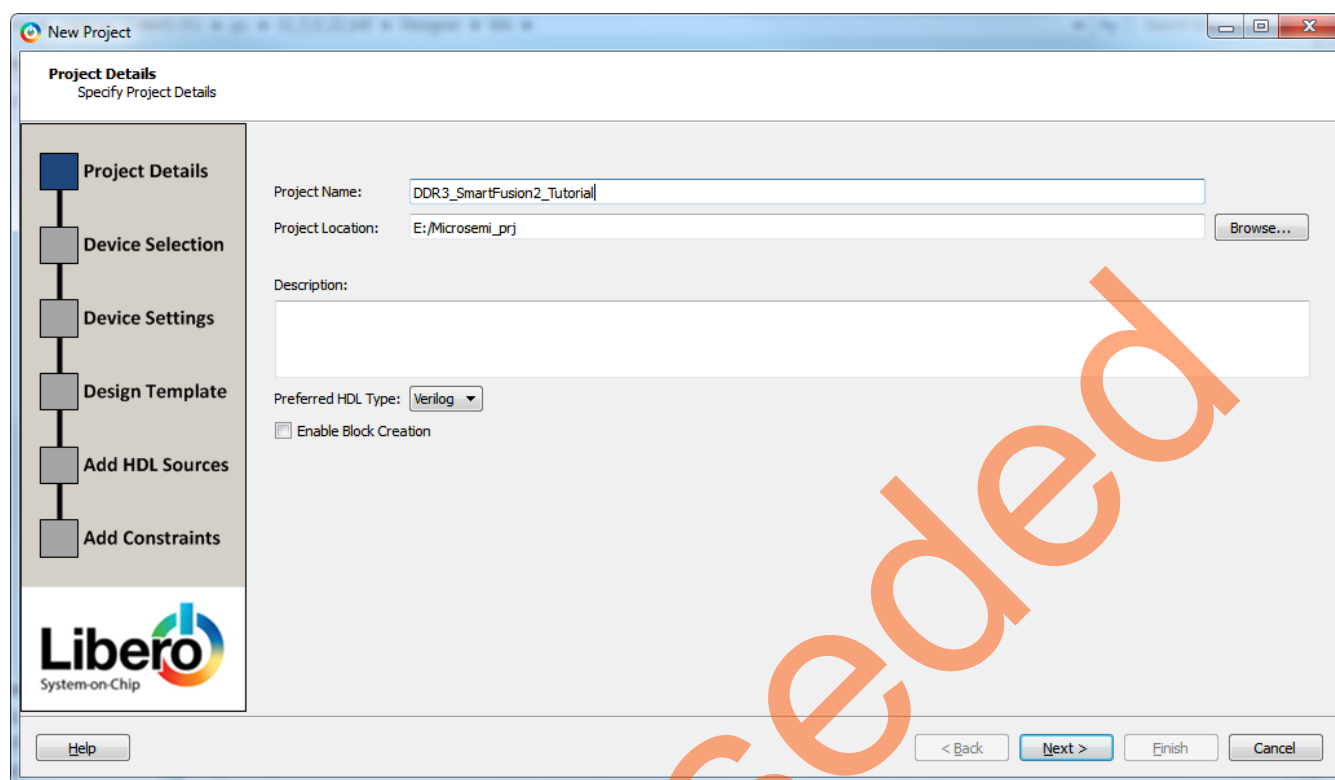


Figure 1. Top-Level Block Diagram

## Step I: Creating a Libero SoC Project

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

1. Launch Libero SoC v11.5.
2. From the **Project** menu, select **New Project**. In the **Project Details** window, enter the information as displayed in Figure 2.
  - 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

3. Click **Next**. In the **Device Selection** window, select the information displayed in [Figure 3](#).
  - Family: SmartFusion2
  - Die: M2S090T
  - Package: 484 FBGA
  - Part Number: M2S090T-1FG484

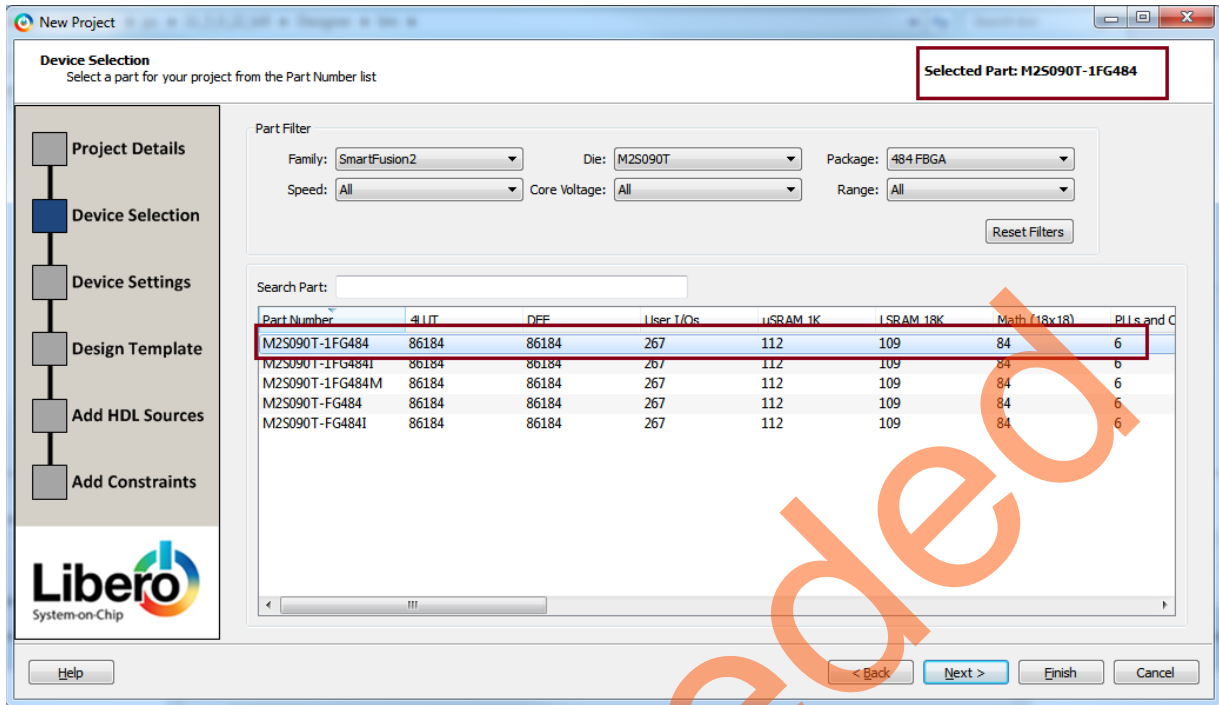


Figure 3. Device Selection Window

4. Click **Next**. In the **Device Settings** window, select the information displayed in Figure 4.
  - Default I/O Technology: LVCMOS 2.5 V
  - PLL Supply Voltage (V): 2.5
  - Maximum Core Voltage Rail Ramp Up Time: 100ms Minimum
  - System Controller Suspend Mode: Unchecked

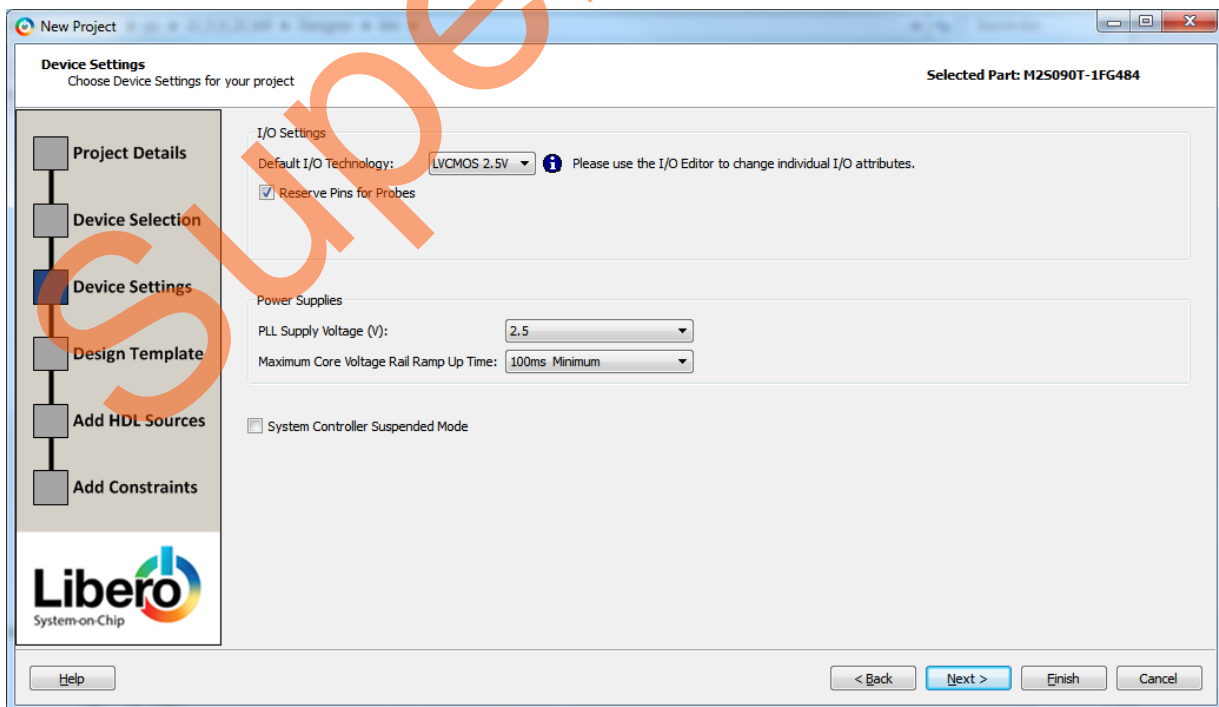
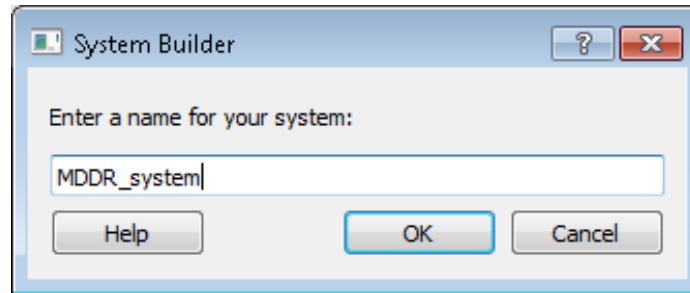


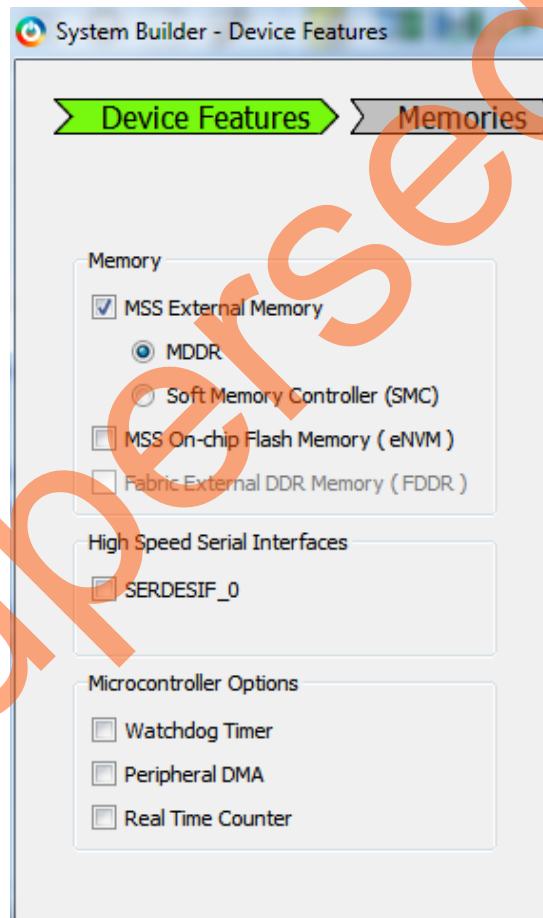
Figure 4. Device Settings Window

5. Click **Next**. In the **Design Template** page, select the **Create a System Builder base design** under the **Design Templates and Creators**.
6. Click **Finish**.
7. Enter **MDDR\_system** as the name of the system and click **OK** , as shown in [Figure 5](#).



**Figure 5.** Create New System Builder Dialog Box

8. The System Builder dialog box is displayed with the **Device Features** page, as shown in [Figure 6](#).



**Figure 6.** SmartFusion2 SoC FPGA System Builder Configurator

9. Under Memory, check **MSS External Memory** and select **MDDR**. Leave all other options unchecked.
10. Click **Next**, the **System Builder – Memories** page is displayed, as shown in [Figure 7](#). The DDR3 external memory models are used in this tutorial.
  - DDR memory settling time (us): **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 200us.

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.

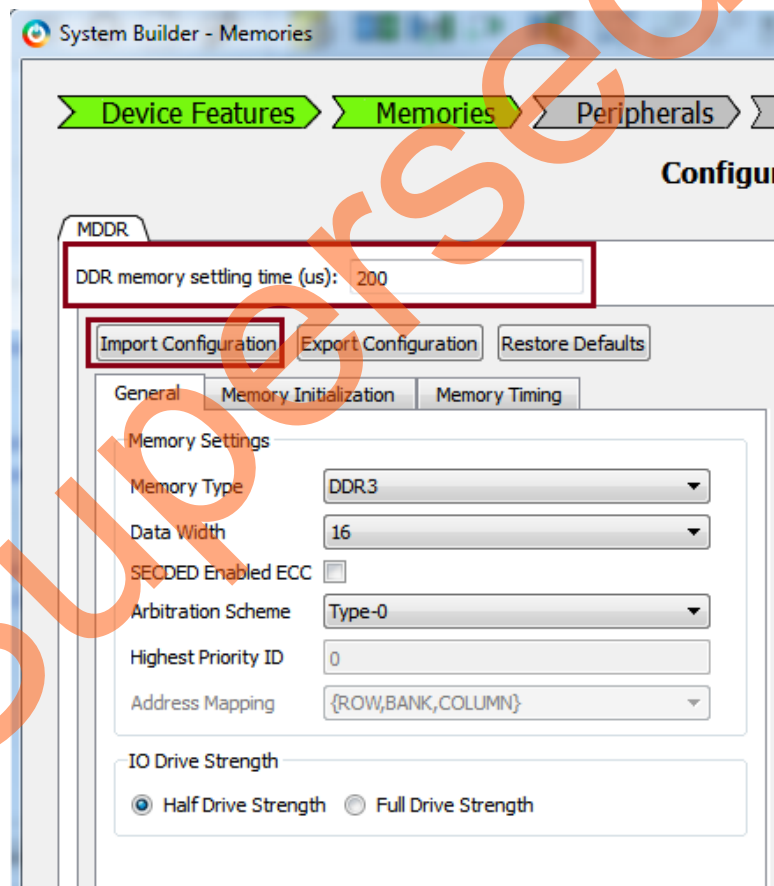
The System Builder enables you to import the register configuration file in which you defined the DDR controller configurations. For this design a configuration file, **DDR3\_PHY\_16\_NO\_ECC\_BL8\_INTER.txt**, is provided in the tutorial zip files. The configuration file is located under <project directory>\DDR3\_SmartFusion2\_Tutorial\Source\_files\DDR3 folder.

Import the register configuration file as follows:

- Click **Import Configuration**, as shown in [Figure 7](#).
- 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
- SECEDED Enabled ECC: Unchecked

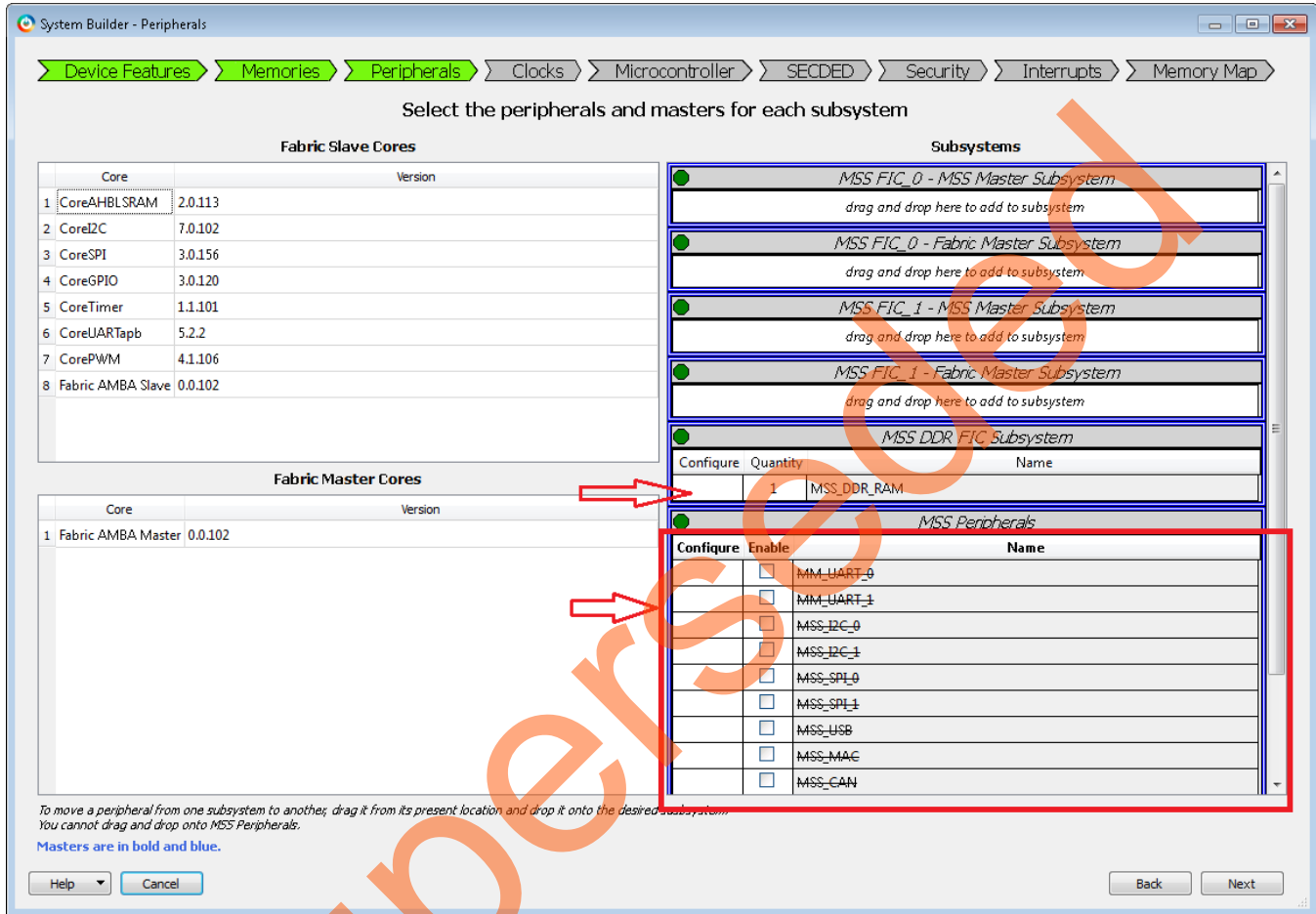


**Figure 7.** System Builder Configurator – Memory Page

11. Select **Next**, System Builder- **Select Peripherals** page is displayed, as shown in [Figure 8](#).

This tutorial does not use any of the MSS peripherals. Clear all the **MSS Peripherals**, as shown in [Figure 8](#).

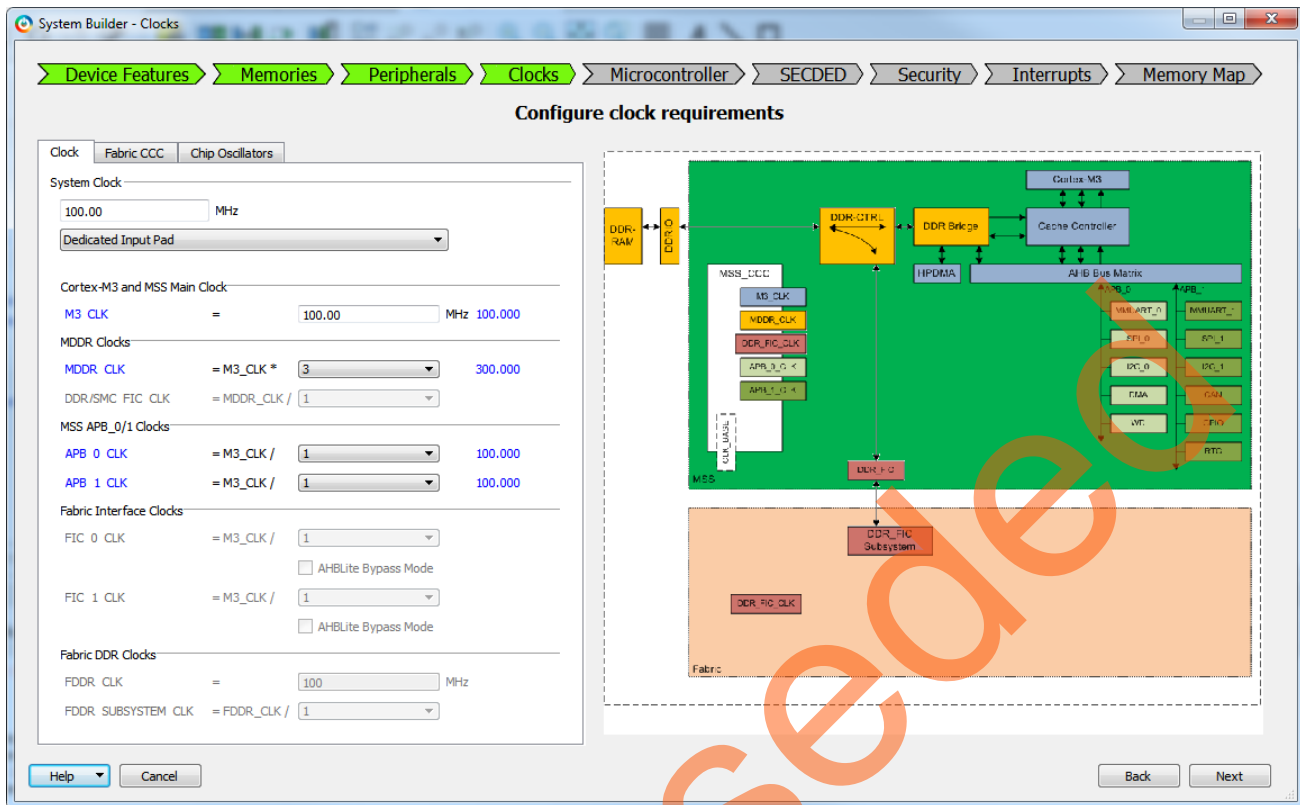
Since the system is using an MSS DDR (on the first page of the System Builder), the MSS\_DDR\_RAM is shown under the MSS DDR FIC Subsystem, as shown in [Figure 8](#).



12. Click **Next**. The System Builder- **Clock Settings** page is displayed, as shown in [Figure 9](#). Select the following options:

- **System Clock:** Set it to 100 MHz (default) and select **Dedicated Input Pad** 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 9](#) and the clock and the blocks that the clock is driving are displayed on the right-side panel.

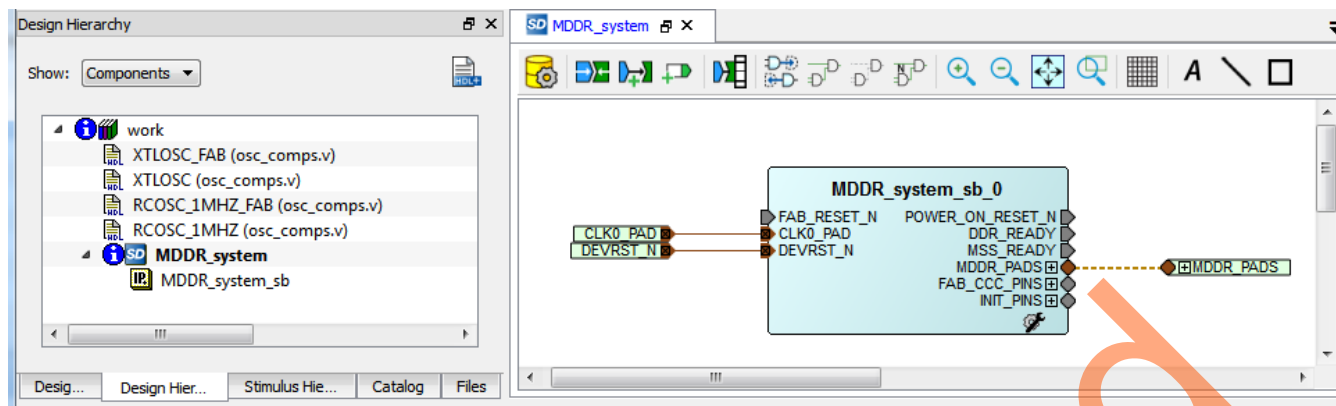


**Figure 9.** System Builder Configurator – Clock Settings Page

13. Click **Next**, the System Builder - **Microcontroller Options** page is displayed.
  - Leave all the default selections
14. Click **Next**, the System Builder - **SECEDED Options** page is displayed.
  - Leave all the default selections
15. Click **Next**, the System Builder - **Security Options** page is displayed.
  - Leave all the default selections
16. Click **Next**, the System Builder - **Interrupts Options** page is displayed.
  - Leave all the default selections
17. Click **Next**, the System Builder - **Memory Map Options** page is displayed.
  - Leave all the default selections
18. 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 10](#). The CoreResetP and CoreConfigP cores are automatically instantiated and connected by the System Builder. Refer to [Opening System Builder Component as SmartDesign](#) section on page 13 for more information on how the blocks are connected in the System Builder component.



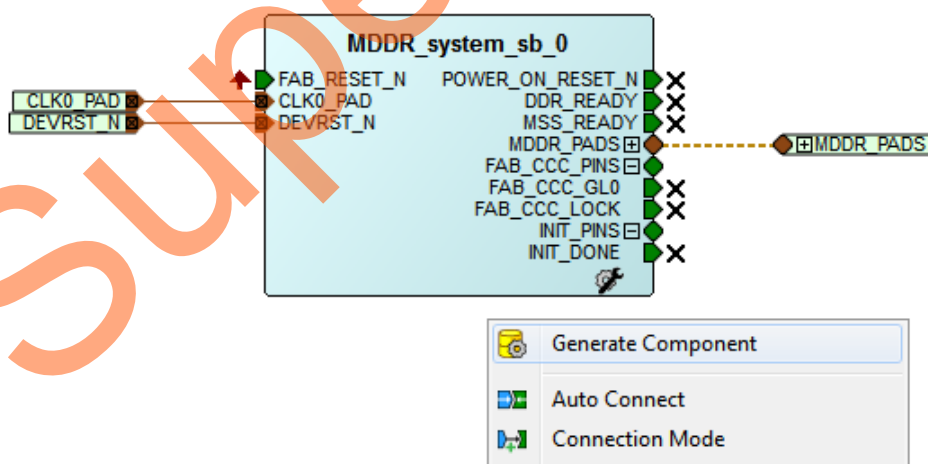
**Figure 10.** SmartFusion2 SoC FPGA System Builder Generated System

19. 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 **FAB\_CCC\_GLO** as unused by right-clicking and selecting **Mark Unused**.
- Mark the output port **FAB\_CCC\_LOCK** as unused by right-clicking and selecting **Mark Unused**.
- Mark the output port **INIT\_DONE** as unused by right-clicking and selecting **Mark Unused**.

20. Generate the final system 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**, as shown in [Figure 11](#).



**Figure 11.** SmartFusion2 SoC FPGA Generated Final System

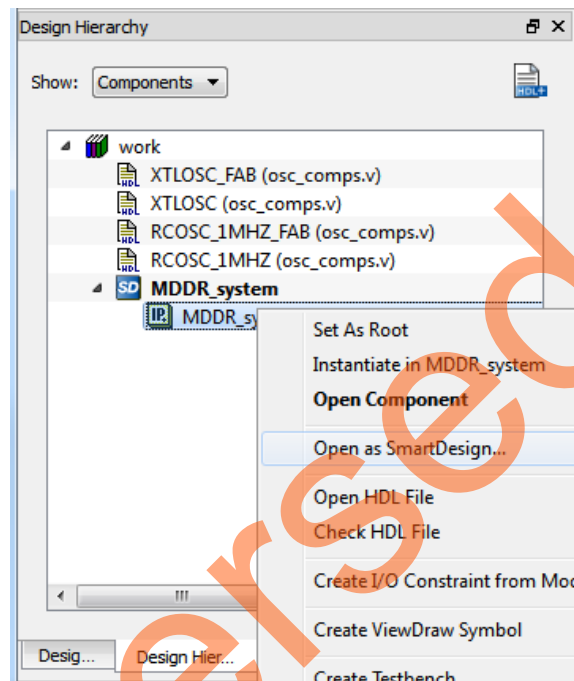
After successful generation of the system, the message "Info: 'MDDR\_system\_top' was successfully generated. Open datasheet for details" is displayed on the log window.

## Opening System Builder Component as SmartDesign

Upon 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.

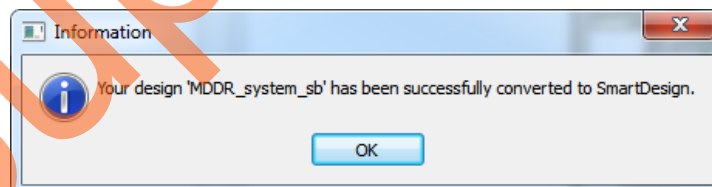
The final System Builder generated system is shown in [Figure 11](#). You can dive into that block to see the individual blocks that make-up the entire design. To do so, open the System Builder generated block using the SmartDesign. It enables you to check the internals of the overall design. To open the MDDR\_system using the SmartDesign, use the following steps:

1. In the **Design Hierarchy**, expand **MDDR\_system\_top** component.
2. Right-click the **MDDR\_system** and select **Open as SmartDesign**, as shown in [Figure 12](#).



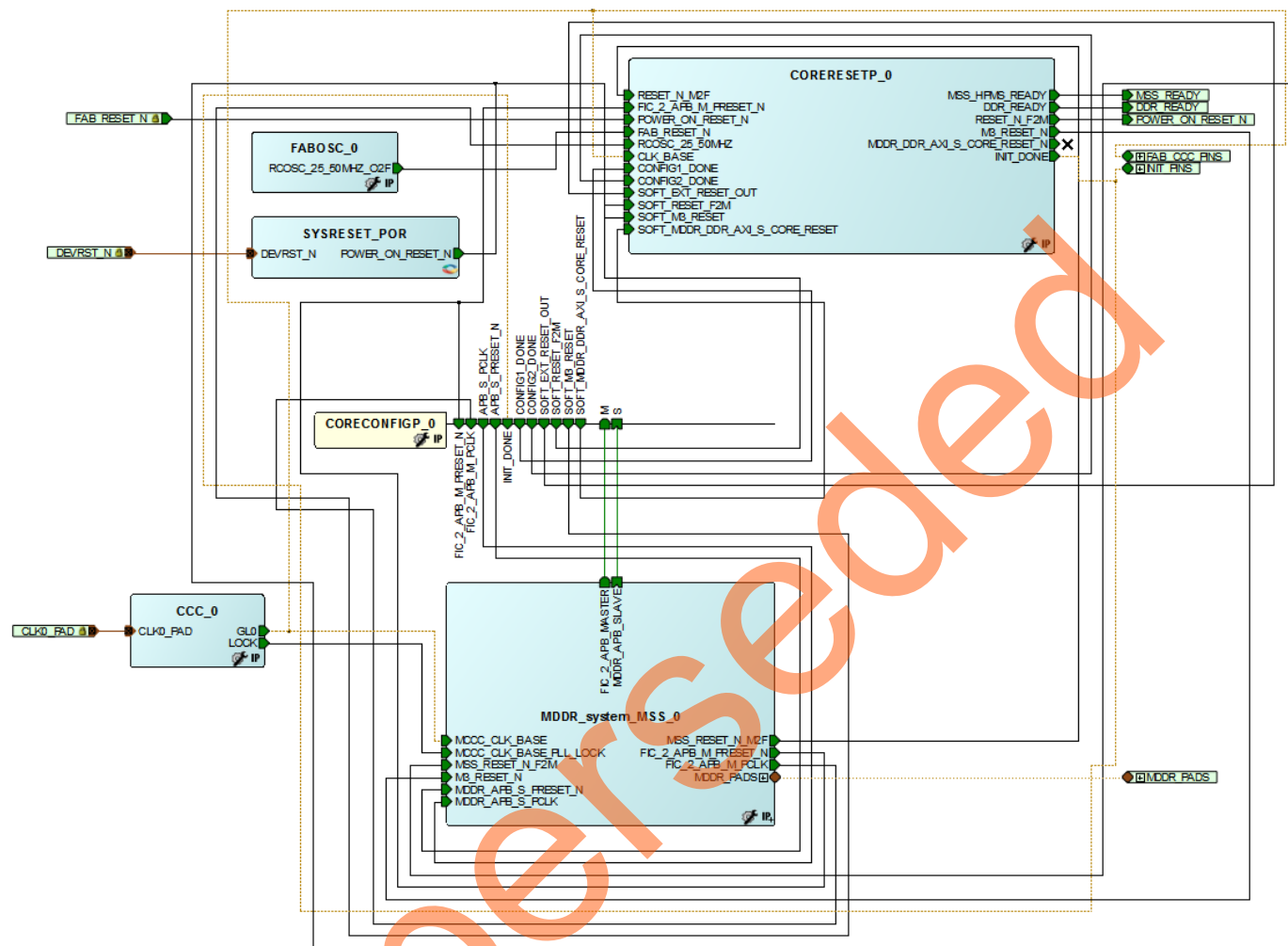
**Figure 12.** Open as SmartDesign Option

The system is converted to a SmartDesign component and the message is displayed, as shown in [Figure 13](#).



**Figure 13.** Successful Conversion of System Builder to a SmartDesign Message

- Click **OK**. The system is shown in the SmartDesign canvas, as shown in Figure 14.



**Figure 14.** 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.

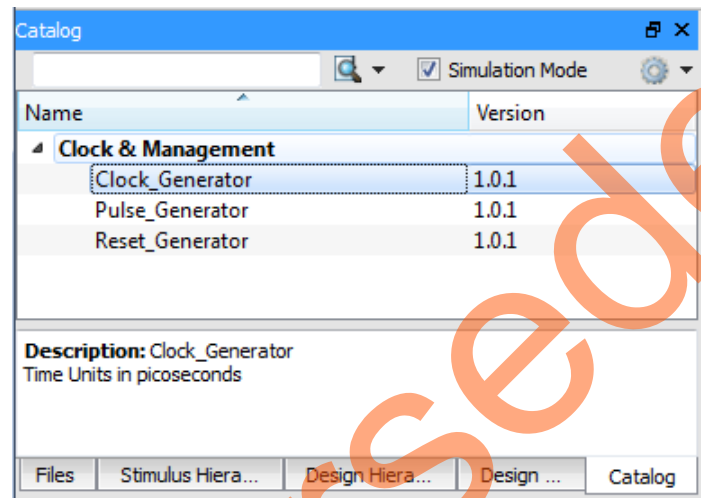
- **SYSRESET\_POR**: It generates the power-on reset signal for the CoreResetP block.
- **CORERESETP\_0** (soft core): It is responsible for managing all the reset mechanism needed for the system.
- **FABOSC\_0**: It generates the clock source for the CoreResetP block.
- **CCC\_0**: It is used to generate 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).
- **CORECONFIG\_0** (soft core): It manages the configuration aspect of the controller based on the specified configuration file.

## Step 2: Generating the Testbench

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

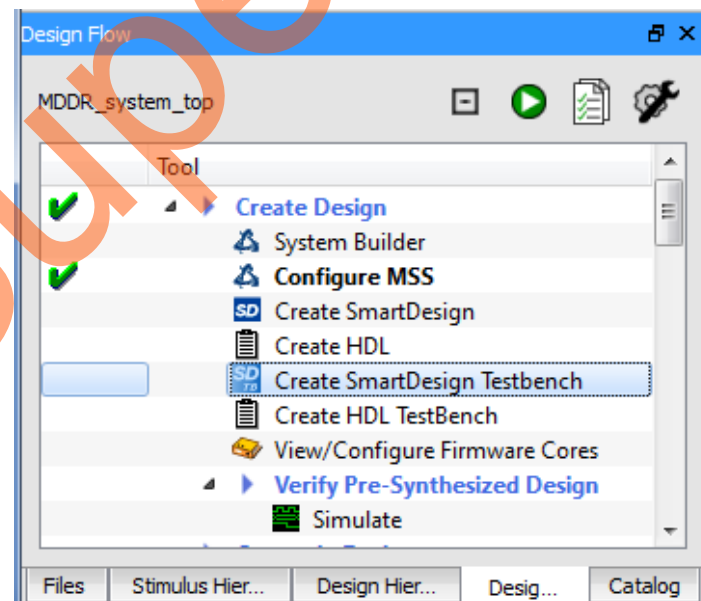
1. Enable the SmartDesign simulation cores by selecting **Simulation Mode** check box in the Libero SoC IP catalog, as shown in Figure 15. The IP catalog displays three simulation cores to drive the device under test (DUT):
  - Clock\_Generator
  - Pulse\_Generator
  - Reset\_Generator

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



**Figure 15.** Simulation Cores in Libero IP Catalog

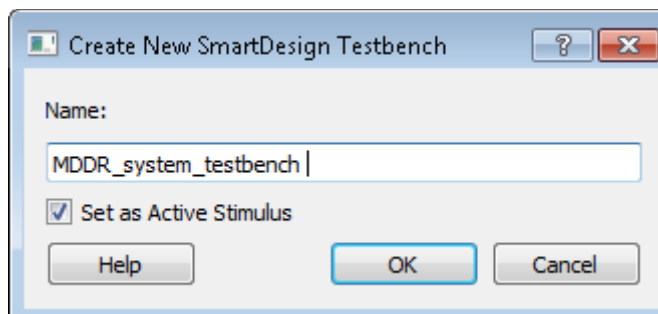
2. Double-click the **Create SmartDesign Testbench** in the **Libero Design Flow** window, as shown in Figure 16.



**Figure 16.** Opening SmartDesign Testbench

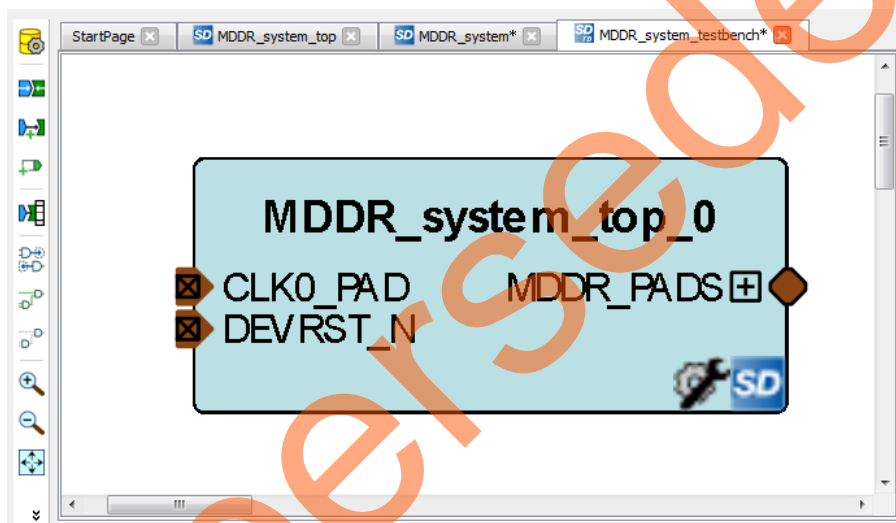
3. The **Create New SmartDesign Testbench** dialog box is displayed, as shown in Figure 17.

4. Enter **MDDR\_system\_testbench** in the **Create New SmartDesign Testbench** dialog box and click **OK**.



**Figure 17.** Create New SmartDesign Testbench Dialog Box

5. The SmartDesign canvas is displayed with the **MDDR\_system\_top\_0** component instantiated, as shown in [Figure 18](#).

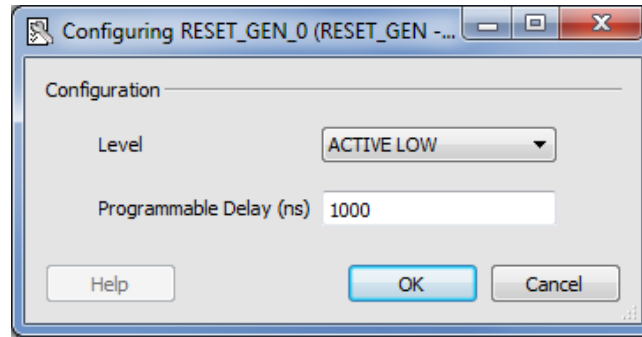


**Figure 18.** SmartDesign Testbench Canvas

6. Drag the **Clock\_Generator** and **Reset\_Generator** simulation cores from the IP catalog to the **MDDR\_system\_testbench** SmartDesign canvas.
7. Open the **Reset\_Generator** configurator by double-clicking **RESET\_GEN\_0** in the SmartDesign canvas. Ensure that the following information, as shown in [Figure 19](#), is set in the **RESET\_GEN\_0** configurator and click **OK**:
  - Level: ACTIVE LOW (default)
  - Programmable Delay (ns): 1000

The reset generator provides the reset pulse for the simulation.

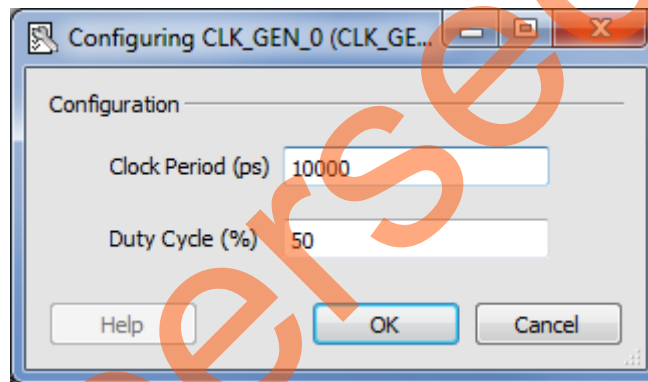




**Figure 19.** RESET\_GEN Configuration

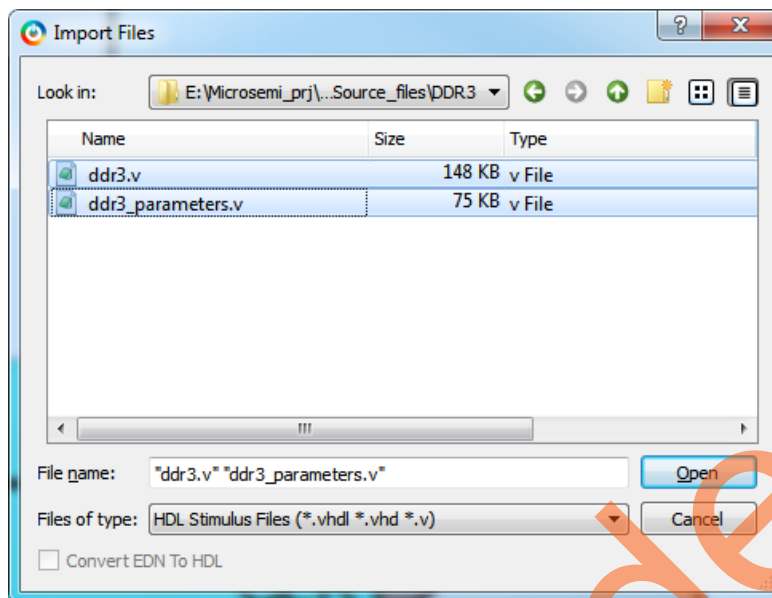
8. Open the **Clock\_Generator** configurator by double-clicking the CLK\_GEN\_0 in the SmartDesign canvas. Ensure that the following information, as shown in Figure 20, is set in the CLK\_GEN\_0 configurator and click **OK**:
  - Clock Period (ps): 10000
  - Duty Cycle (%): 50

The clock generator period is set to 10000 ps to generate this 100 MHz clock, as shown in Figure 9. The System Clock 10000 ps equal is to 100 MHz.



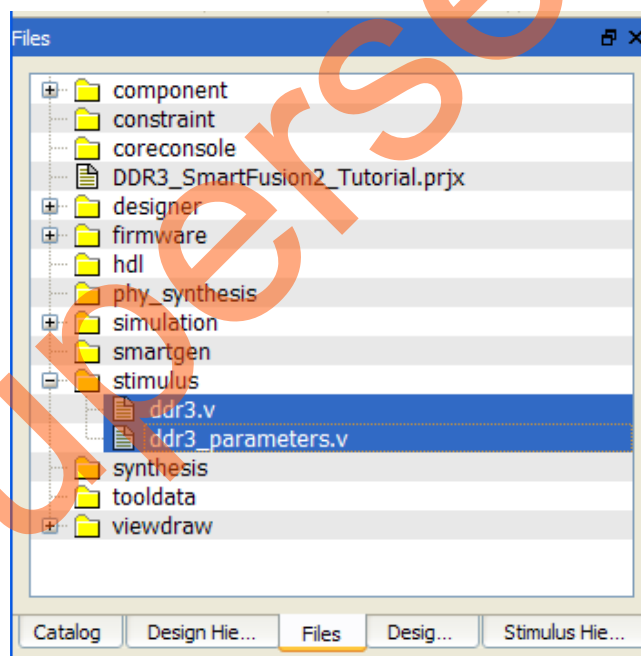
**Figure 20.** CLK\_GEN Configuration

9. 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 21.
  - Select the provided **ddr3.v** and the **ddr3\_parameters.v** files and click **Open**, as shown in Figure 21. The files are located under the <project directory>\DDR3\_SmartFusion2\_Tutorial\Source\_files\DDR3\_folder.



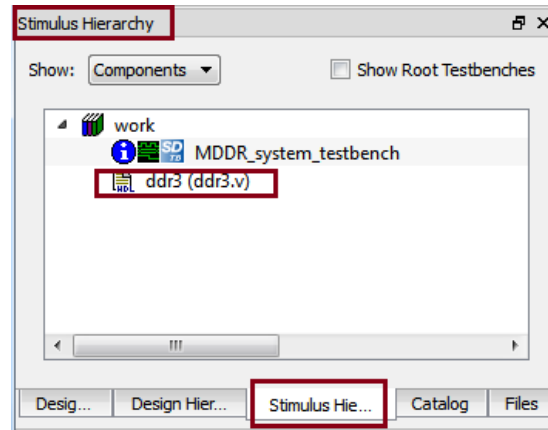
**Figure 21.** Import the DDR3 Models as Stimulus Files

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



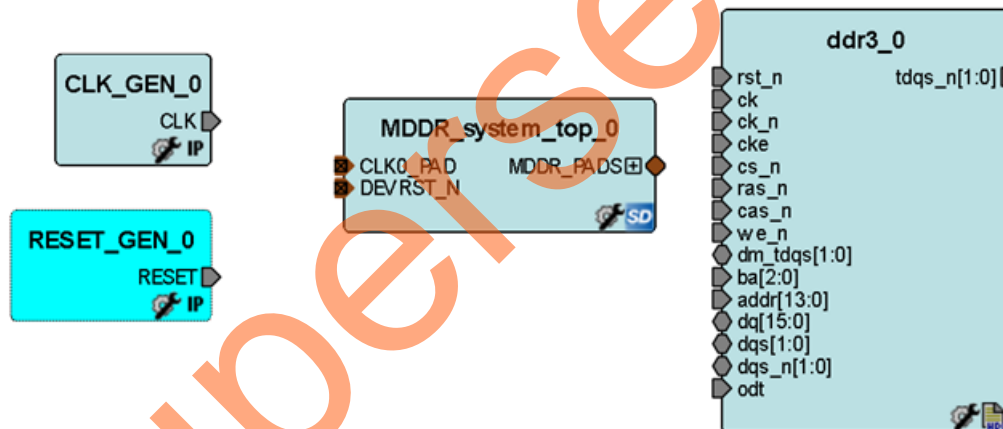
**Figure 22.** Imported DDR3 in Stimulus Folder

When the file is imported as a Stimulus, the file is displayed in the **Stimulus Hierarchy** window, as shown Figure 23.



**Figure 23.** Stimulus Hierarchy Window

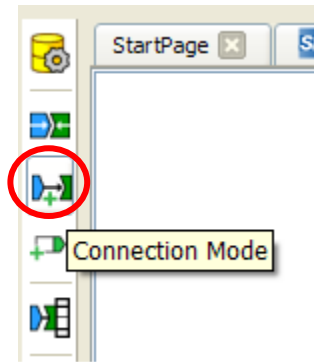
10. From the Stimulus Hierarchy window, drag the **ddr3** file into the **MDDR\_system\_testbench** canvas. You are basically instantiating the DDR3 models into the testbench to emulate an external DDR3 memory. You are going to simulate the write and read from the DDR3 using the Cortex-M3 processor as the master through the MDDR controller in the MSS. After you instantiate the DDR3, the canvas is displayed, as shown in Figure 24.



**Figure 24.** 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 25. 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 25.** 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.

11. Using whichever connection method described above, make the following connections in the SmartDesign canvas between the **RESET\_GEN\_0**, **CLK\_GEN\_0** and the **MDDR\_system\_top\_0**:
  - From **RESET\_GEN\_0: RESET** to **MDDR\_system\_top\_0: DEVRST\_N**
  - From **CLK\_GEN\_0:CLK** to **MDDR\_system\_top\_0:CLK0\_PAD**
12. Expand the **MDDR\_system\_top\_0: MDDR\_PADS**. Using whichever connection method described above, make the following connections in the SmartDesign canvas between the **MDDR\_system\_top\_0** and the **ddr3\_0**:
  - Connect **MDDR\_DQS\_TMATCH\_0\_IN** to **MDDR\_DQS\_TMATCH\_0\_OUT** of the **MDDR\_system\_top\_0** block.
  - Connect the rest of the pins as shown in [Table 2](#).

**Table 2.** DDR3 Pins Connections

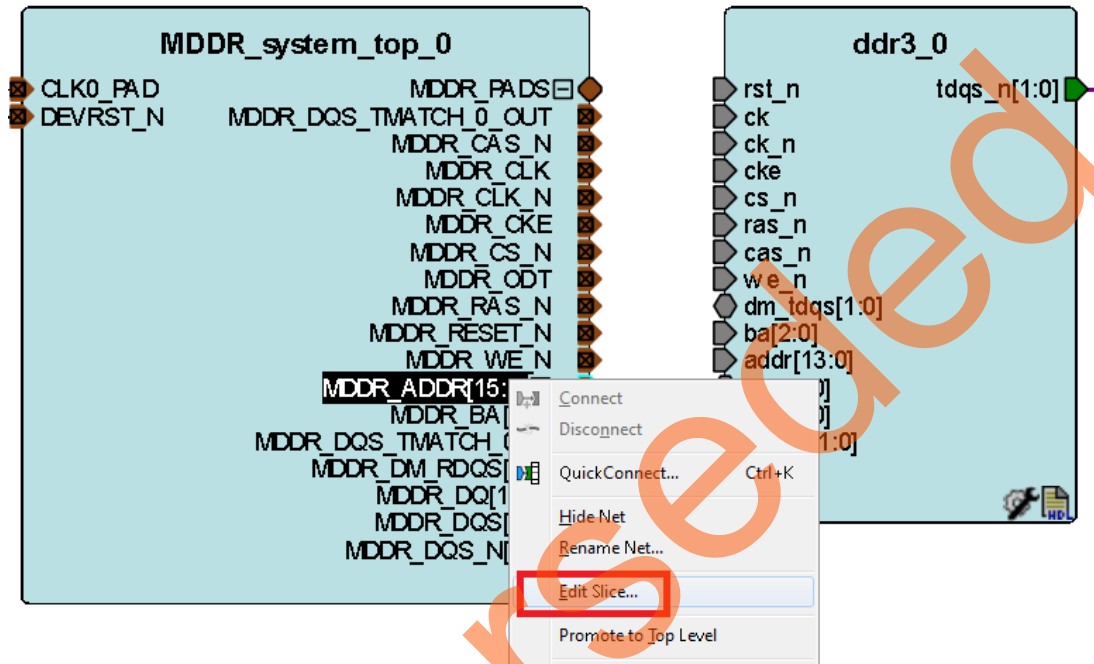
MDDR_System_Top_0_Pins	DDR3_0_Pins
MDDR_CAS_N	cas_n
MDDR_CKE	cke
MDDR_CLK	ck
MDDR_CLK_N	ck_n
MDDR_CS_N	cs_n
MDDR_ODT	odt
MDDR_RAS_N	ras_n
MDDR_RESET_N	rst_n
MDDR_WE_N	we_n
MDDR_BA[2:0]	ba[2:0]
MDDR_DM_RDQS[1:0]	dm_tdqs[1:0]
MDDR_DQ[15:0]	dq[15:0]
MDDR_DQS[1:0]	dqs[1:0]
MDDR_DQS_N[1:0]	dqs_n[1:0]
MDDR_ADDR[13:0]	Addr[13:0]
MDDR_ADDR[15:14]	Mark Unused

There are buses on the **MDDR\_system\_top\_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\_top\_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**.

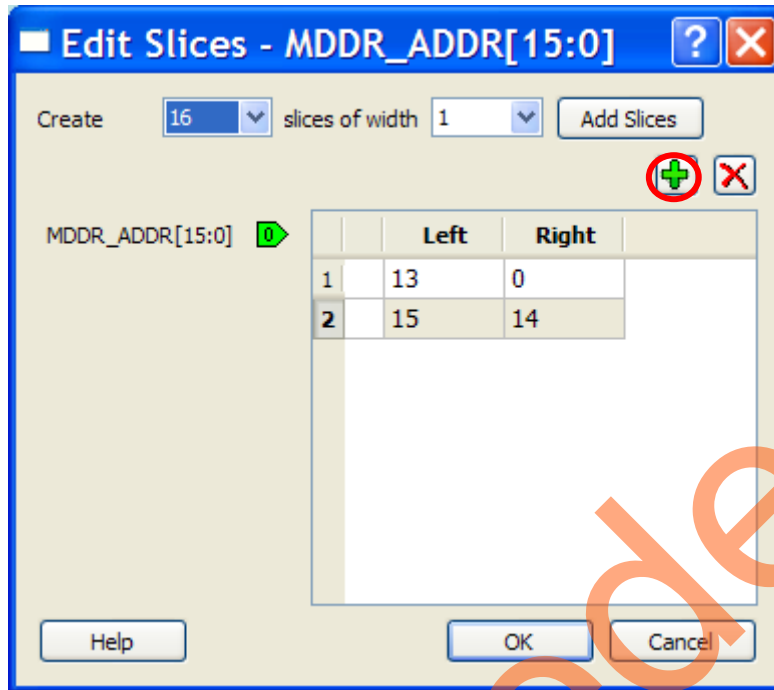
**To slice a bus, use the following steps:**

- Right-click the bus and select **Edit Slice**, as shown in Figure 26. The dialog box is displayed, as shown in Figure 27.



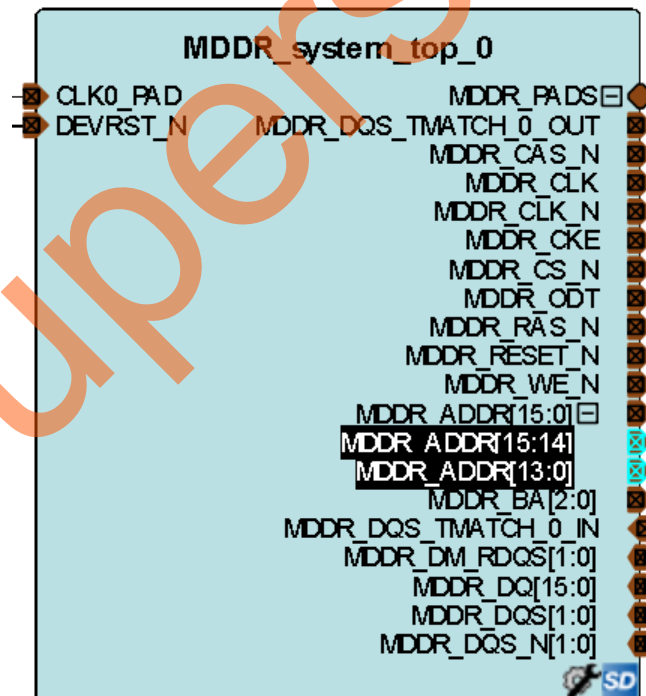
**Figure 26.** Creating a Bus Slice

- Click  icon (circled in Figure 27) to add a slice. Since you need to add two slices, click  icon twice. Then add the slices, as shown in Figure 27.



**Figure 27.** Edit Slices Dialog Box

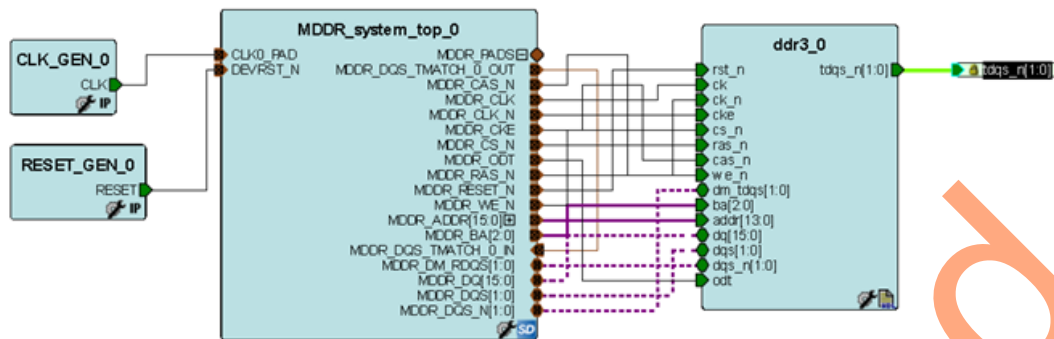
- Click **OK**. This creates two slices of the MDDR\_ADDR bus, as shown in Figure 28.



**Figure 28.** MDDR\_ADDR Created Slices

13. 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 29.

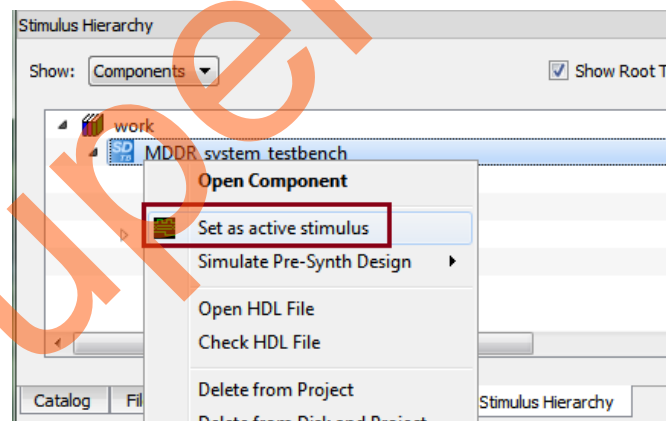


**Figure 29.** Fully Connected MDDR System

14. 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:

15. After generating the testbench, you need to make it **Active testbench**. 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:
  1. Go to the **Stimulus Hierarchy** tab
  2. If not already set, right-click the **MDDR\_system\_testbench** and select **Set as active stimulus**, as shown in Figure 30.

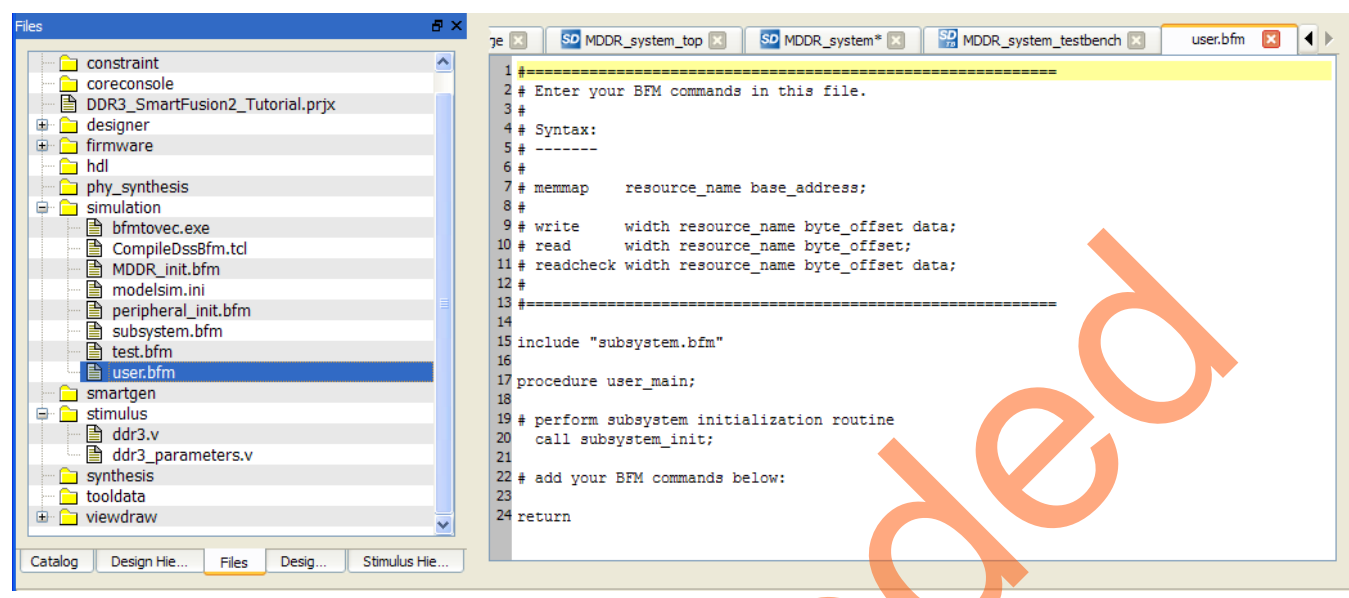


**Figure 30.** Setting a testbench as Active testbench

## 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 the user.bfm. The user.bfm file is shown, as shown in Figure 31.



**Figure 31.** SmartDesign Generated user.bfm File

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";

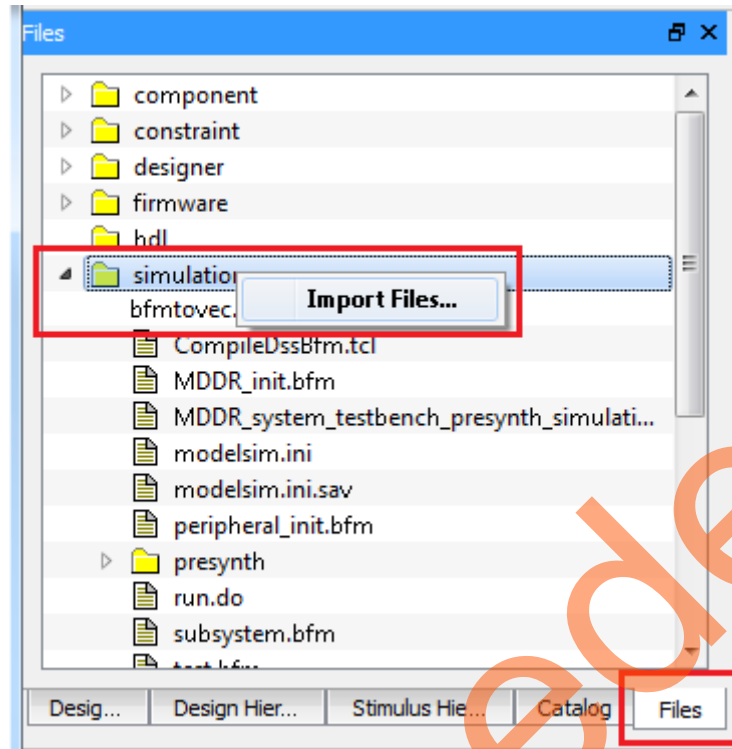
b. #write different values to different location
write w M_MDDR0_SPACE_0 0x0000 0xA1B2C3D4;
write w M_MDDR0_SPACE_0 0x0004 0x10100101;
write w M_MDDR0_SPACE_0 0x0008 0xD7D7E1E1;
write w M_MDDR0_SPACE_0 0x000C 0xA5DEF6E7;
write w M_MDDR0_SPACE_0 0x0010 0xABCDEF01;
write w M_MDDR0_SPACE_0 0x0014 0xCCBBAADD;

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:

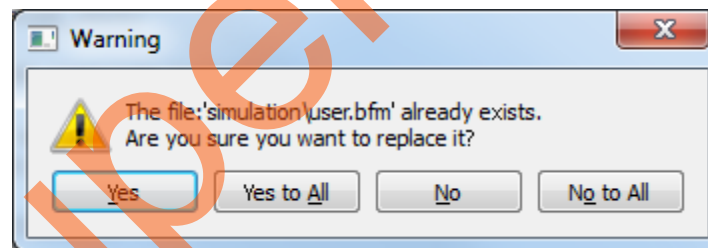
- Go to Files tab and right-click on the simulation Folder as shown in Figure 32.





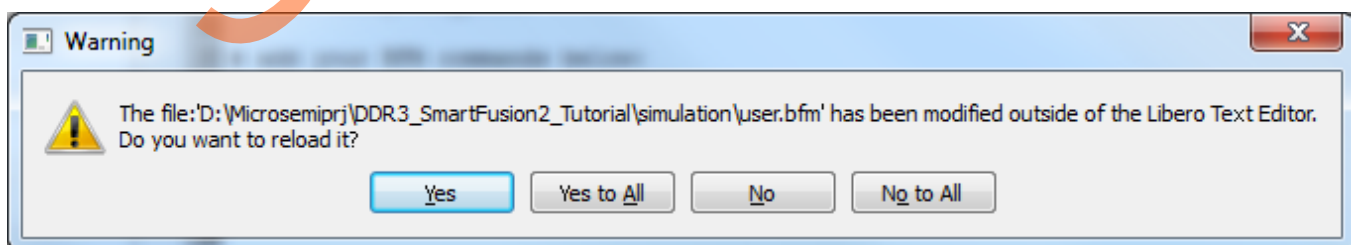
**Figure 32.** Importing bfm Source File

- Browse to <project directory>\DDR3\_SmartFusion2\_Tutorial\Source\_files and select **user.bfm** and select **Open**.
- A warning message is displayed, as shown in Figure 33. Click **Yes**.



**Figure 33.** Replacing Existing user.bfm File

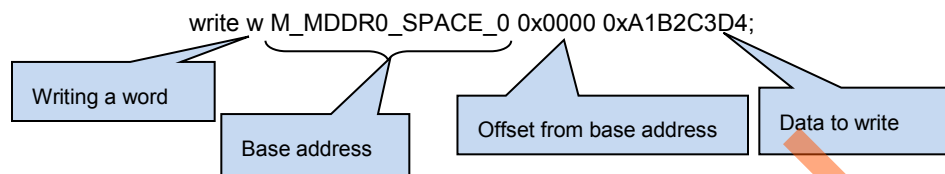
- A warning message is displayed as shown in Figure 34 if the user.bfm file is already opened in your Libero window. Click **Yes**. If the user.bfm is not opened already in your Libero window, the message does not show up.



**Figure 34.** Reloading and Updating user.bfm File

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,



- **Step c:** In this step you are checking what you wrote. The final user.bfm is displayed, as shown in Figure 35.

```

user.bfm
21
22 # add your BFM commands below:
23
24
25 # step a: DDR memory map
26 memmap M_MDDR0_SPACE_0 0xA0000000;
27 print "TEST STARTS";
28
29
30 #step b: write different values to different location
31 write w M_MDDR0_SPACE_0 0x0000 0xA1B2C3D4;
32 write w M_MDDR0_SPACE_0 0x0004 0x10100101;
33 write w M_MDDR0_SPACE_0 0x0008 0xD7D7E1E1;
34 write w M_MDDR0_SPACE_0 0x000C 0xA5DEF6E7;
35 write w M_MDDR0_SPACE_0 0x0010 0xABCDEF01;
36 write w M_MDDR0_SPACE_0 0x0014 0xCCBBAADD;
37
38
39 #step c: read check what you wrote in step 'b' above
40 readcheck w M_MDDR0_SPACE_0 0x0000 0xA1B2C3D4;
41 readcheck w M_MDDR0_SPACE_0 0x0004 0x10100101;
42 readcheck w M_MDDR0_SPACE_0 0x0008 0xD7D7E1E1;
43 readcheck w M_MDDR0_SPACE_0 0x000C 0xA5DEF6E7;
44 readcheck w M_MDDR0_SPACE_0 0x0010 0xABCDEF01;
45 readcheck w M_MDDR0_SPACE_0 0x0014 0xCCBBAADD;
46
47
48 print "TEST ENDS";
49
50 return

```

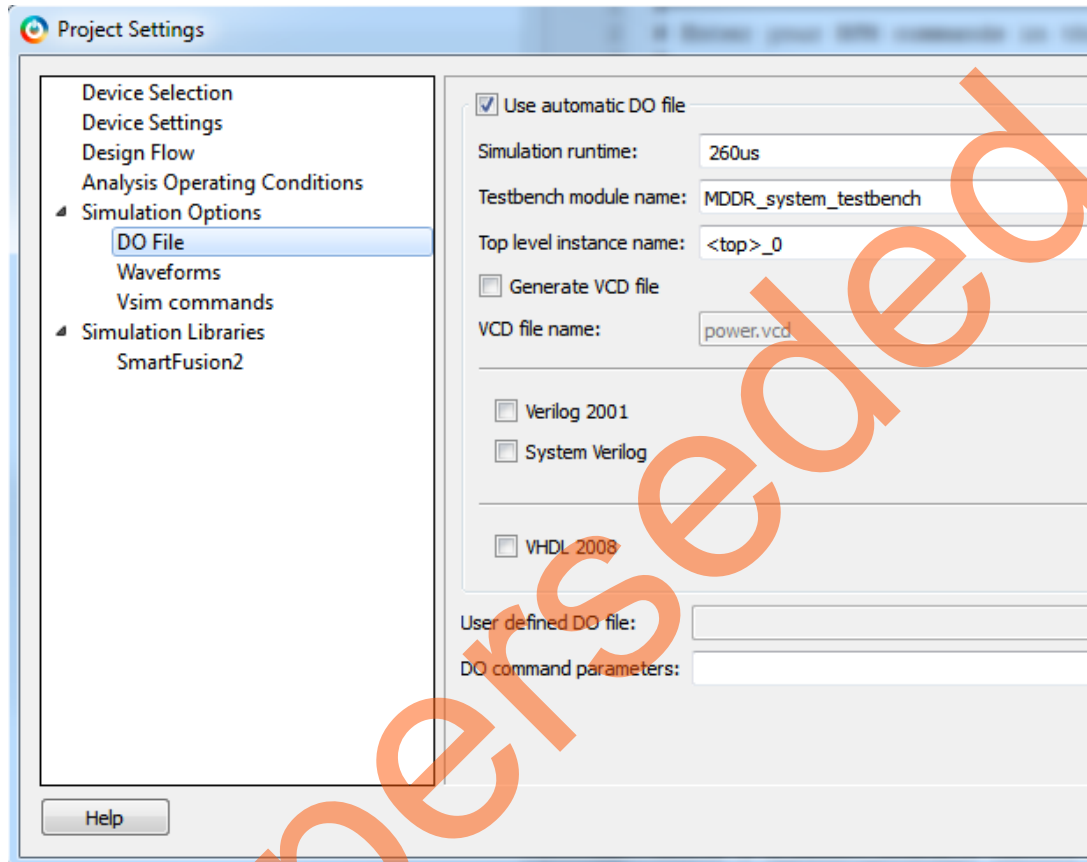
**Figure 35.** user.bfm after Adding the Commands

Refer to [DirectCore Advanced Microcontroller Bus Architecture – Bus Functional Model User Guide](#) for more details.

## Step 4: Simulating the Design

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

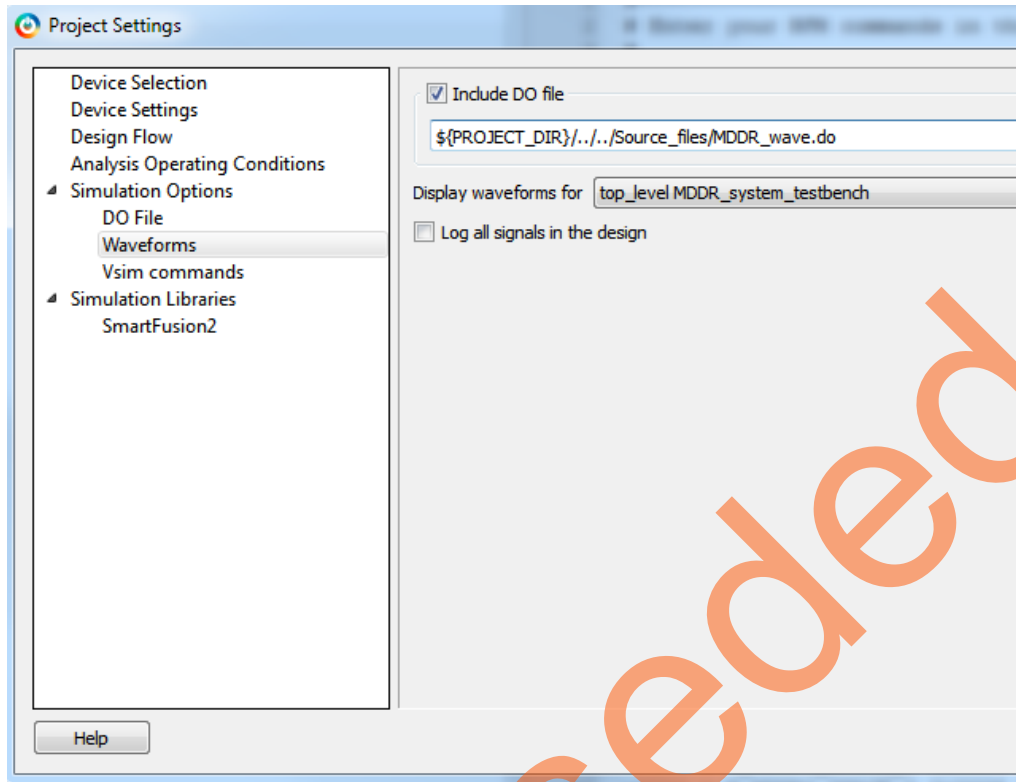
1. Navigate to **Project > Project Settings** to open the Libero SoC project settings .
2. Select **Do File** under Simulation Options in the Project Settings window. Change the **Simulation runtime** to 260us, as shown in Figure 36.



**Figure 36.** Project Setting – Do File Simulation Runtime Setting

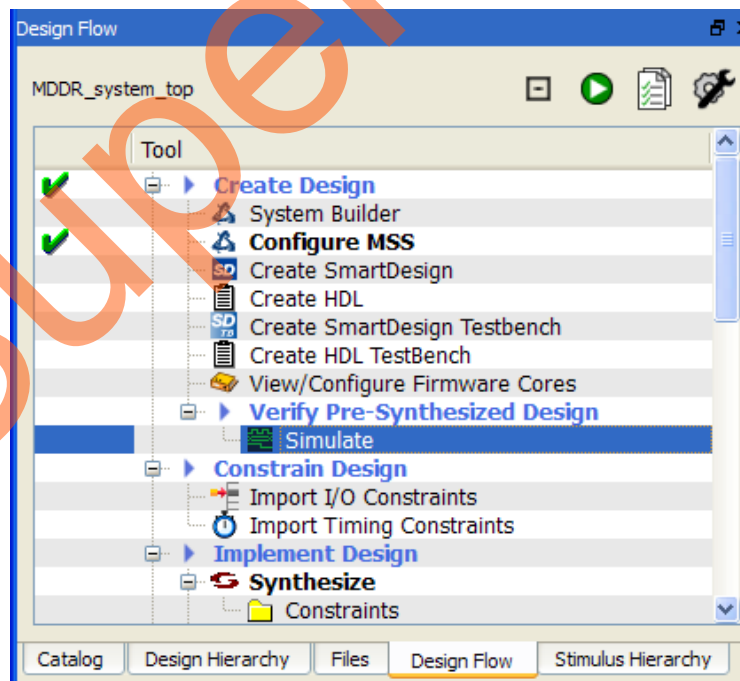
3. Select **Waveforms** under Simulation Options:
  - Select **Include DO file**, browse to where you extracted the provided source files, and select **MDDR\_wave.do** file, as shown in Figure 37. In this file, the list of signals that are required is already selected so you can check for the expected 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 37.** Project Setting – Specifying the MDDR\_wave.do File Location

4. Expand **Verify Pre-Synthesized Design** in the Design Flow window, as shown in Figure 38. Double-click **Simulate** to launch ModelSim in GUI mode.



**Figure 38.** Starting Pre-Synthesis Simulation

## Step 5: Validating the Simulation Results

1. ModelSim runs the design for about 260 us, 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 200us as specified in the System Builder- Memory page, as shown in [Figure 7](#). 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 39](#). 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.

```

# BFM: Data Read a0000010 adcbbaadd MASK:ffffff at 233704.646912ns
# MONAXIO:*****
# MONAXIO: RCMD 6 ID:1 SEQ:6 ADDR:a0000010 BURST:1 LEN:0 SIZE:3 LOCK:0 at 233714647 ps
# MONDDRC: READ REQUEST SEQ:6 TOKEN:d ADDR:a0000002 LEN:1 at 233724647 ps
# MDDR_system_testbench. ddr3_0.cmd_task: at time 233741524 ps INFO: Read bank 0 col 008, auto precharge 0
# MDDR_system_testbench. ddr3_0.data_task: at time 233759857 ps INFO: READ @ DQS= bank = 0 row = 0000 col = 00000008 data = ef01
# MDDR_system_testbench. ddr3_0.data_task: at time 233761524 ps INFO: READ @ DQS= bank = 0 row = 0000 col = 00000009 data = abcd
# MDDR_system_testbench. ddr3_0.data_task: at time 233763190 ps INFO: READ @ DQS= bank = 0 row = 0000 col = 0000000a data = aadd
# MDDR_system_testbench. ddr3_0.data_task: at time 233764857 ps INFO: READ @ DQS= bank = 0 row = 0000 col = 0000000b data = cbbb
# MDDR_system_testbench. ddr3_0.data_task: at time 233766524 ps INFO: READ @ DQS= bank = 0 row = 0000 col = 0000000c data = xxxx
# MDDR_system_testbench. ddr3_0.data_task: at time 233768190 ps INFO: READ @ DQS= bank = 0 row = 0000 col = 0000000d data = xxxx
# MDDR_system_testbench. ddr3_0.data_task: at time 233769857 ps INFO: READ @ DQS= bank = 0 row = 0000 col = 0000000e data = xxxx
# MDDR_system_testbench. ddr3_0.data_task: at time 233771524 ps INFO: READ @ DQS= bank = 0 row = 0000 col = 0000000f data = xxxx
# MONAXIO: RDAT Start ID:1 SEQ:6 - matching RCMD:6 ADDR:a0000010 BURST:1 LEN:0 SIZE:3 at 1786418 ps
# MONAXIO: RDAT ID:1 ccbbaaddabcbdef01 at 233787980 ps
# MONAXIO: 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 1796418 ps
# MONDDRC: RESP DATA TOKEN:d DATA:ccbbaaddabcbdef01 End:0 at 233787980 ps
# MONDDRC: RESP DATA TOKEN:d DATA:xxxxxxxxxxxxxxxx End:1 at 233794647 ps
# MONDDRC: QUEUED CMDS 0
# BFM: Data Read a0000014 ccbbaadd MASK:ffffff at 233804.646912ns
*****
# BFM Simulation Complete - 5202 Instructions - NO ERRORS
*****
# MDDR_system_testbench. ddr3_0.cmd_task: at time 234038190 ps INFO: Precharge bank 0
# MDDR_system_testbench. ddr3_0.cmd_task: at time 234064856 ps INFO: Precharge Power Down Enter
Now: 260 us Delta: 2 tdqs_n 259976922874 fs to 260001214E

```

**Figure 39.** ModelSim BFM Simulation Transcript Results

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 40](#).



**Figure 40.** Doc or Undock of ModelSim Wave Window

3. Once the Wave window is undocked, click the **Zoom Full** icon as shown in [Figure 41](#) to fit all the waveforms in the single view.



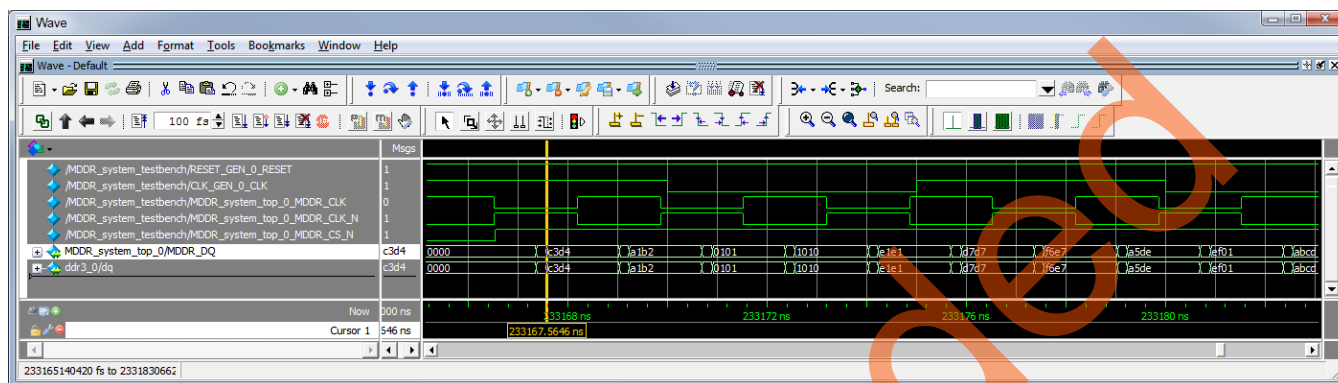
**Figure 41.** Zoom Full Option

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 42](#), 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 43](#).



**Figure 42.** Zoom In on Active Cursor Button

Figure 43 shows the time at which the data was written/read-back to/from the DDR3 external modules. That concludes this tutorial.



**Figure 43.** Write/Read Data

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

## 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 ModelSim using AMBA BFM simulation is verified.

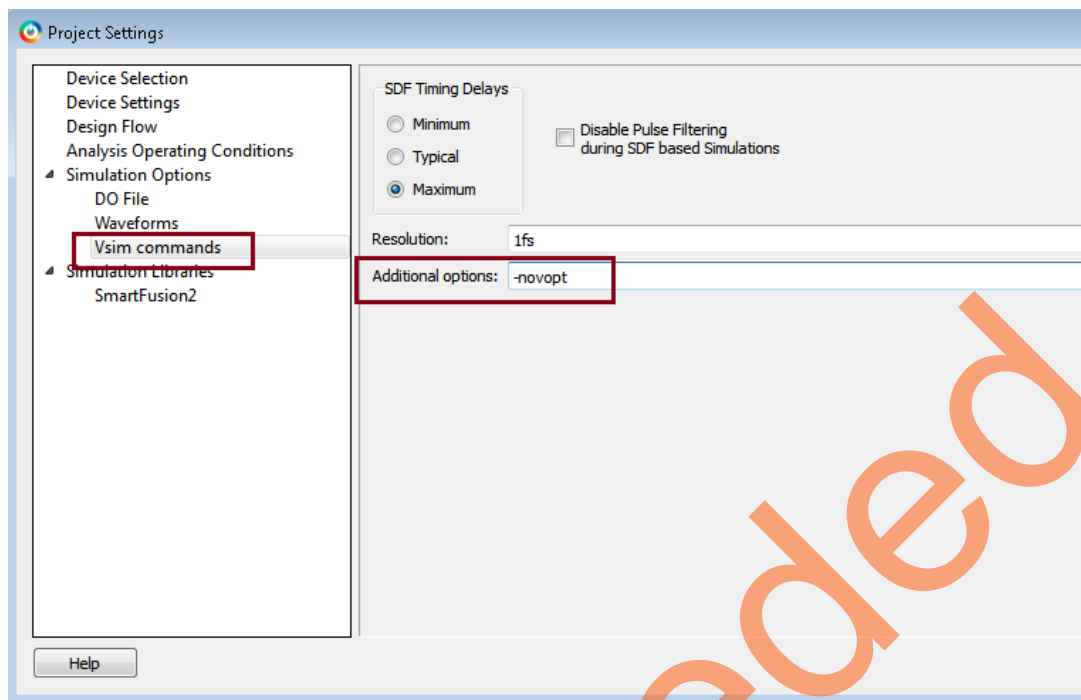
## Appendix A: VHDL Flow

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:

- 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\)
- 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.

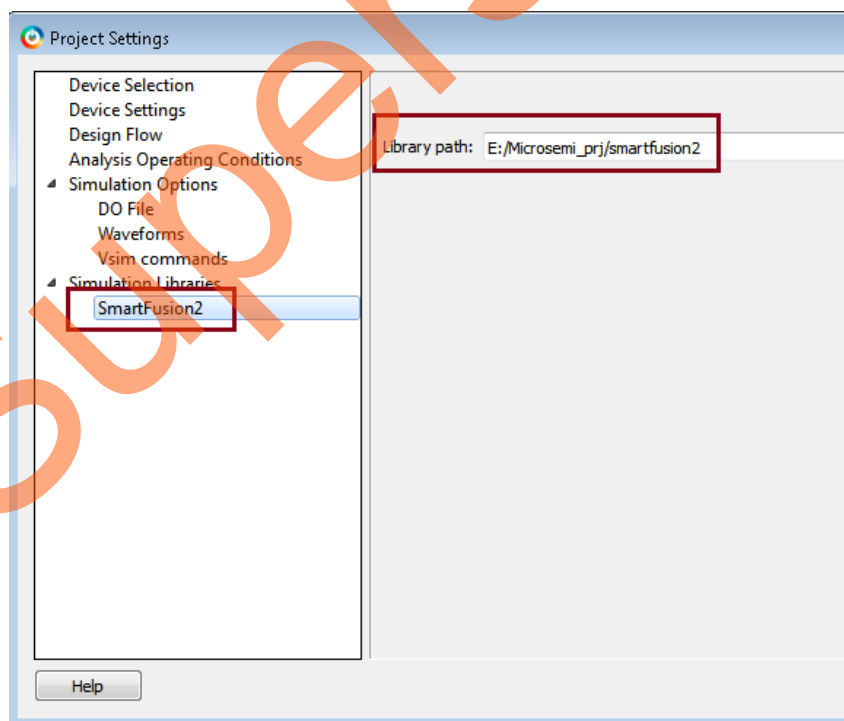
- Simulate with automatic design optimization option disabled (-novopt) and point to the new precompiled library location (for example, E:\Microsemi\_prj\smartfusion2) in the **Project Settings** window as shown below:
  - Select **Project Settings** from the **Project** menu
  - Select **Vsim commands** under **Simulation Options**. Add the -novopt option into the **Additional options** field, as shown in Figure 44. The -novopt option disables the automatic design optimization run.



**Figure 44.** Project Settings – Specifying **-novopt** Simulation Option

(c) Select **SmartFusion2** under the **Simulation Libraries**

- I. In the **Library path**, enter the new location where you copied the precompiled library (for example, E:/Microsemi\_prj/smartfusion2), as shown in Figure 45.



**Figure 45.** Project Settings – Specifying Precompiled Library Path

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

## Appendix B: Abbreviation Used

- cSoC – customizable system-on-chip
- MSS – Microcontroller subsystem
- DDR3 SDRAM – Double data rate synchronous dynamic Random Access Memory
- CCC – Clock conditioning circuitry
- MSS CCC – CCC block inside the MSS component
- Fabric CCC - CCC block instantiated inside the FPGA fabric
- DDR – Dual data rate memory controller
- MDDR – DDR controller inside the MSS component.
- BFM – Bus functional model
- FIC – MSS fabric interface

Superseded



## List of Changes

Revision	Changes	Page
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

**Note:** The revision number is located in the part number after the hyphen. The part number is displayed at the bottom of the last page of the document. The digits following the slash indicate the month and year of publication.

---

# 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.

## 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**

## 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.

## Technical Support

For Microsemi SoC Products Support, visit

<http://www.microsemi.com/products/fpga-soc/designsupport/fpga-soc-support>

## 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/soc/>.

## 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.

### 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](mailto:soc_tech@microsemi.com).

### My Cases

Microsemi SoC Products Group customers may submit and track technical cases online by going to [My Cases](#).

### Outside the U.S.

Customers needing assistance outside the US time zones can either contact technical support via email ([soc\\_tech@microsemi.com](mailto:soc_tech@microsemi.com)) or contact a local sales office. [Sales office listings](#) can be found at [www.microsemi.com/soc/company/contact/default.aspx](http://www.microsemi.com/soc/company/contact/default.aspx).

## 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\\_itar@microsemi.com](mailto:soc_tech_itar@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.

Superseded



**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](mailto:sales.support@microsemi.com)

© 2015 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; security technologies and scalable anti-tamper products; Power-over-Ethernet ICs and midspans; as well as custom design capabilities and services. Microsemi is headquartered in Aliso Viejo, Calif., and has approximately 3,400 employees globally. Learn more at [www.microsemi.com](http://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.