

TU0372

Tutorial

**Interfacing SmartFusion2 SoC FPGA with DDR3
Memory Through MDDR Controller**



a  **MICROCHIP** company



a  MICROCHIP company

Microsemi 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

Email: sales.support@microsemi.com

www.microsemi.com

©2021 Microsemi, a wholly owned subsidiary of Microchip Technology Inc. All rights reserved. Microsemi and the Microsemi logo are registered trademarks of Microsemi Corporation. All other trademarks and service marks are the property of their respective owners.

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.

About Microsemi

Microsemi, a wholly owned subsidiary of Microchip Technology Inc. (Nasdaq: MCHP), offers a comprehensive portfolio of semiconductor and system solutions for aerospace & defense, communications, data center 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. Learn more at www.microsemi.com.

Contents

1	Revision History	1
1.1	Revision 14.0	1
1.2	Revision 13.0	1
1.3	Revision 12.0	1
1.4	Revision 11.0	1
1.5	Revision 10.0	1
1.6	Revision 9.0	1
1.7	Revision 8.0	1
1.8	Revision 7.0	1
1.9	Revision 6.0	1
1.10	Revision 5.0	1
1.11	Revision 4.0	1
1.12	Revision 3.0	1
1.13	Revision 2.0	2
1.14	Revision 1.0	2
2	Interfacing SmartFusion2 SoC FPGA with DDR3 Memory Through MDDR Controller	3
2.1	Design Requirements	3
2.2	Prerequisites	4
2.3	Design Overview	4
2.4	Simulating the Design	6
2.4.1	Opening the MDDR System in SmartDesign	15
2.4.2	Generating the Testbench	17
2.4.3	Modifying the BFM Script	26
2.4.4	Running the Simulation	29
2.4.5	Validating the Simulation Results	31
3	Appendix 1: VHDL Flow	33

Figures

Figure 1	Top-Level Block Diagram	5
Figure 2	New Project – Project Details Window	6
Figure 3	Device Selection Window	7
Figure 4	Device Settings Window	8
Figure 5	New Project Information Window	9
Figure 6	System Name Entry	9
Figure 7	SmartFusion2 SoC FPGA System Builder Configurator	10
Figure 8	System Builder Configurator – Memories Page	11
Figure 9	System Builder Configurator – Peripherals Page	12
Figure 10	System Builder Configurator – Clocks Page	13
Figure 11	SmartFusion2 SoC FPGA System Generated by System Builder	14
Figure 12	SmartFusion2 SoC FPGA Final System	14
Figure 13	Open as SmartDesign Option	15
Figure 14	Confirmation Message About System Builder Conversion to SmartDesign	15
Figure 15	System Builder-Generated System Opened in the SmartDesign	16
Figure 16	Simulation Cores in Libero IP Catalog	17
Figure 17	Libero Design Flow – Create SmartDesign Testbench Option	17
Figure 18	Create New SmartDesign Testbench Dialog Box	17
Figure 19	SmartDesign Testbench Canvas	18
Figure 20	RESET_GEN Configuration	18
Figure 21	CLK_GEN Configuration	19
Figure 22	Import Files Dialog Box (For Importing DDR3 Models to Stimulus)	19
Figure 23	Imported DDR3 in Stimulus Folder	20
Figure 24	Stimulus Hierarchy Window	21
Figure 25	System Testbench Canvas with DDR3 Models Instantiated	21
Figure 26	Connection Mode Icon	21
Figure 27	Edit Slice Option	22
Figure 28	Edit Slices Dialog Box	22
Figure 29	MDDR_ADDDR Slices	23
Figure 30	Fully-Connected MDDR System	24
Figure 31	Set as active stimulus Option	25
Figure 32	Files Tab with BFM File Opened	26
Figure 33	Files Tab - Import Files Option (For Importing BFM Source Files)	27
Figure 34	Warning Message for Replacing Existing BFM File	27
Figure 35	Warning Message when BFM File is Already Open	27
Figure 36	BFM File After Adding the Commands	28
Figure 37	Project Settings Window – Do File Simulation Runtime Setting	29
Figure 38	Project Settings Window – MDDR_wave.do File Location	30
Figure 39	Design Flow Tab – Simulate Option	30
Figure 40	ModelSim BFM Simulation Transcript Results	31
Figure 41	ModelSim Wave Window – Doc/Undock Icon	31
Figure 42	Zoom Full Icon	31
Figure 43	Zoom In on Active Cursor Icon	32
Figure 44	Write/Read Data	32
Figure 45	Project Settings Window – Entering the -novopt Vsim Command	33
Figure 46	Project Settings Window – Specifying Precompiled Library Path	33

Tables

Table 1	Design Requirements	3
Table 2	DDR3 Pin Connections	24

1 Revision History

The revision history describes the changes that were implemented in the document. The changes are listed by revision, starting with the most current publication.

1.1 Revision 14.0

The following is a summary of the changes made in this revision.

- Updated the document for Libero SoC v12.6.
- Removed the references to Libero version numbers.

1.2 Revision 13.0

The following is a summary of the changes made in revision 13.0 of this document.

- MSS was removed from the design requirements. For more information, refer to [Design Requirements](#), page 3.
- The simulation runtime value was updated. For more information, refer to [Running the Simulation](#), page 29.

1.3 Revision 12.0

Updated the document for Libero v11.8 software release.

1.4 Revision 11.0

Updated the document for Libero v11.7 software release (SAR 75916).

1.5 Revision 10.0

Updated the document for Libero v11.6 software release (SAR 68373).

1.6 Revision 9.0

Updated the document for Libero v11.5 software release (SAR 62935).

1.7 Revision 8.0

Updated the document for Libero v11.4 software release (SAR 59067).

1.8 Revision 7.0

Updated the document for Libero v11.3 software release (SAR 55761).

1.9 Revision 6.0

Updated the document for Libero v11.2 software release (SAR 53253).

1.10 Revision 5.0

Updated the document for 11.0 production SW release (SAR 46975).

1.11 Revision 4.0

Updated the document for Libero 11.0 Beta SP1 software release (SAR 44417).

1.12 Revision 3.0

Updated the document for Libero 11.0 Beta SPA software release (SAR 42888).

1.13 Revision 2.0

Updated the document for Libero 11.0 Beta launch (SAR 41898).

1.14 Revision 1.0

Updated the document for LCP2 software release (SAR 38956).

2 Interfacing SmartFusion2 SoC FPGA with DDR3 Memory Through MDDR Controller

This tutorial describes how to create a hardware design using System Builder to access an external DDR3 memory through the built-in hard ASIC Microcontroller subsystem DDR (MDDR) in SmartFusion[®]2 SoC FPGAs. This tutorial also shows how to functionally verify the design using Bus Functional Model (BFM) simulation. The SmartFusion2 SoC FPGA has two DDR controllers, MDDR and Fabric DDR (FDDR) controllers. The MDDR controller is a hard ASIC block in the SmartFusion2 SoC FPGA devices. 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 controller 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) v(x.x) project using the SmartFusion2 SoC FPGA.
- Configuring and generating the various hardware blocks and clocking system using 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 the ModelSim Waveform window.

2.1 Design Requirements

The following table lists and software and hardware requirements for this tutorial.

Table 1 • Design Requirements

Requirement	Version
Operating system	64 bit Windows 7 and 10
Host PC or laptop	
Software	
Libero SoC Design Suite	Note: Refer to the <code>readme.txt</code> file provided in the design files for the software versions used with this reference design.

Note: Libero SmartDesign and configuration screen shots shown in this guide are for illustration purpose only. Open the Libero design to see the latest updates.

2.2 Prerequisites

Before you begin:

1. Download and install Libero SoC (as indicated in the website for this design) on the host PC from the following location.

<https://www.microsemi.com/product-directory/design-resources/1750-libero-soc>

2. For demo design files download link:

http://soc.microsemi.com/download/rsc/?f=m2s_tu0372_df

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

2.3 Design Overview

The design demonstrates the read or write access to an external slave DDR3 memory using the SmartFusion2 device. In the SmartFusion2 SoC FPGA, the Cortex-M3 processor acts as the master and performs the read and write transactions on the external slave memory. The 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 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/pre-charge 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 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 the 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. For more information, refer to the CoreConfigP and CoreResetP handbooks in the IP Catalog of the Libero SoC software.

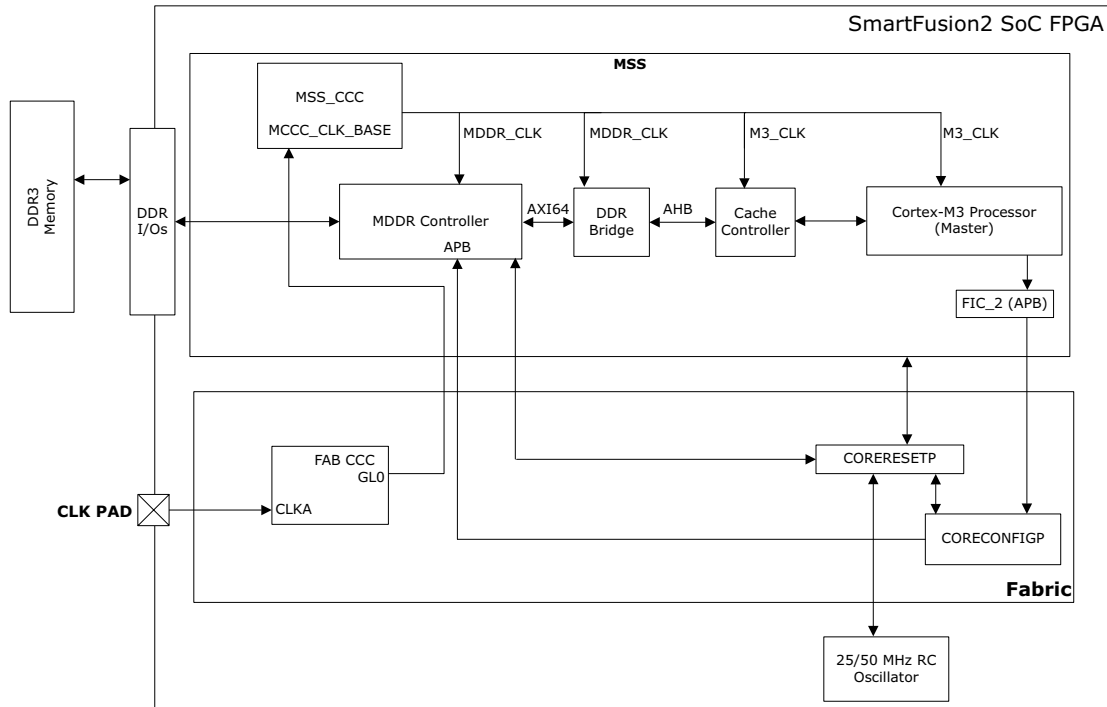
The purpose of this tutorial is to demonstrate the interface of the MDDR with an external DDR slave memory through the MSS. In this design, the System Builder is used to configure the system clocks and the MDDR block to access the external DDR3 memory through the MSS through the DDR I/Os without going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

In the SmartFusion2 SoC FPGA device, there are six Clock Conditioning Circuits (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 System Builder to generate clocks for the various elements in the MSS and the fabric respectively.

The following figure illustrates the different blocks used in this design.

Figure 1 • Top-Level Block Diagram



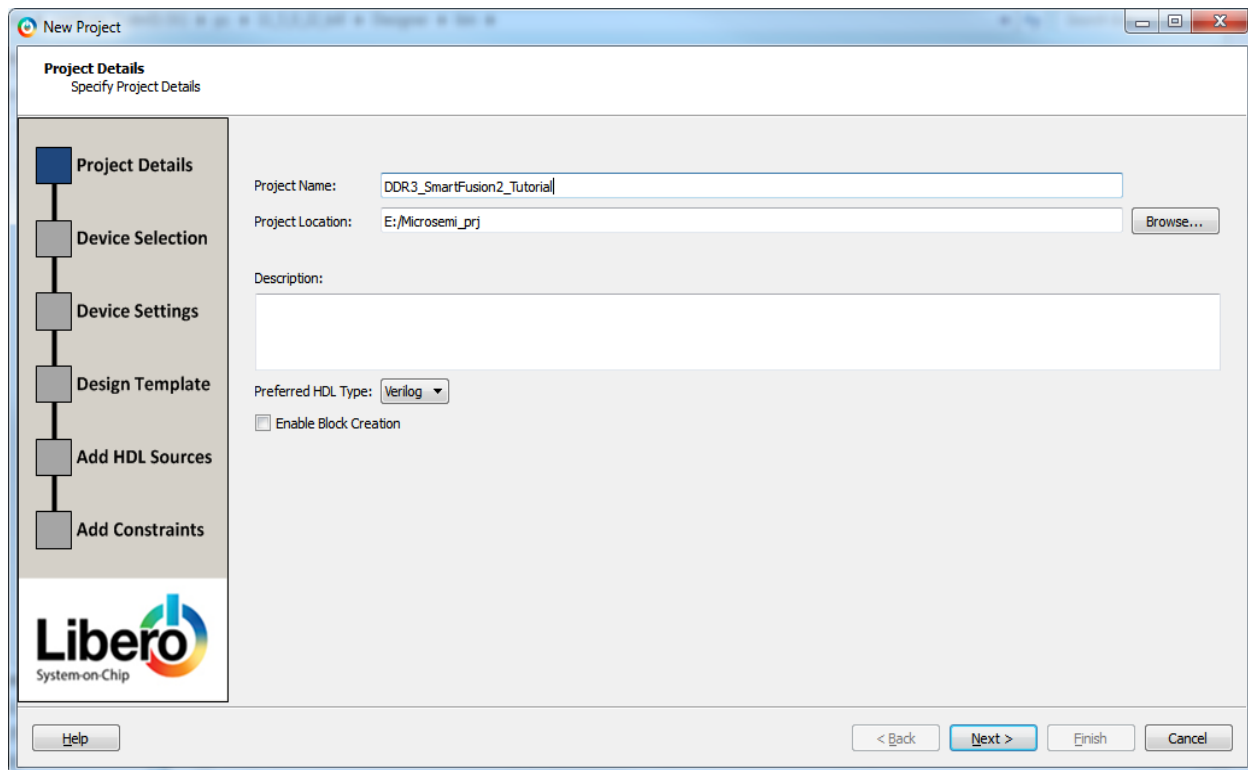
2.4 Simulating the Design

The following sections describe how to create the Libero SoC project and run the simulation for this tutorial. Creating a Libero SoC Project

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

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

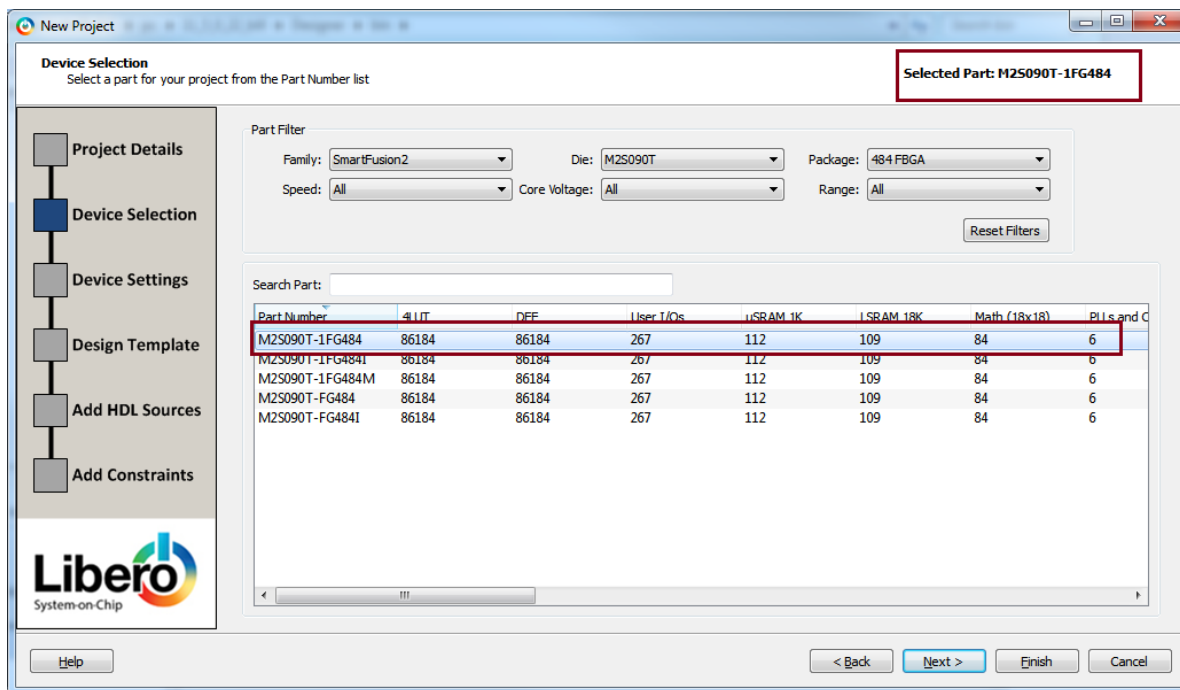
Figure 2 • New Project – Project Details Window



4. Click **Next**. The **Device Selection** window is displayed.

5. In the **Device Selection** window, select the information from the respective drop-down lists, as displayed in the following figure.
 - Family: SmartFusion2
 - Die: M2S090T
 - Package: 484 FBGA
 - Part Number: M2S090T-1FG484

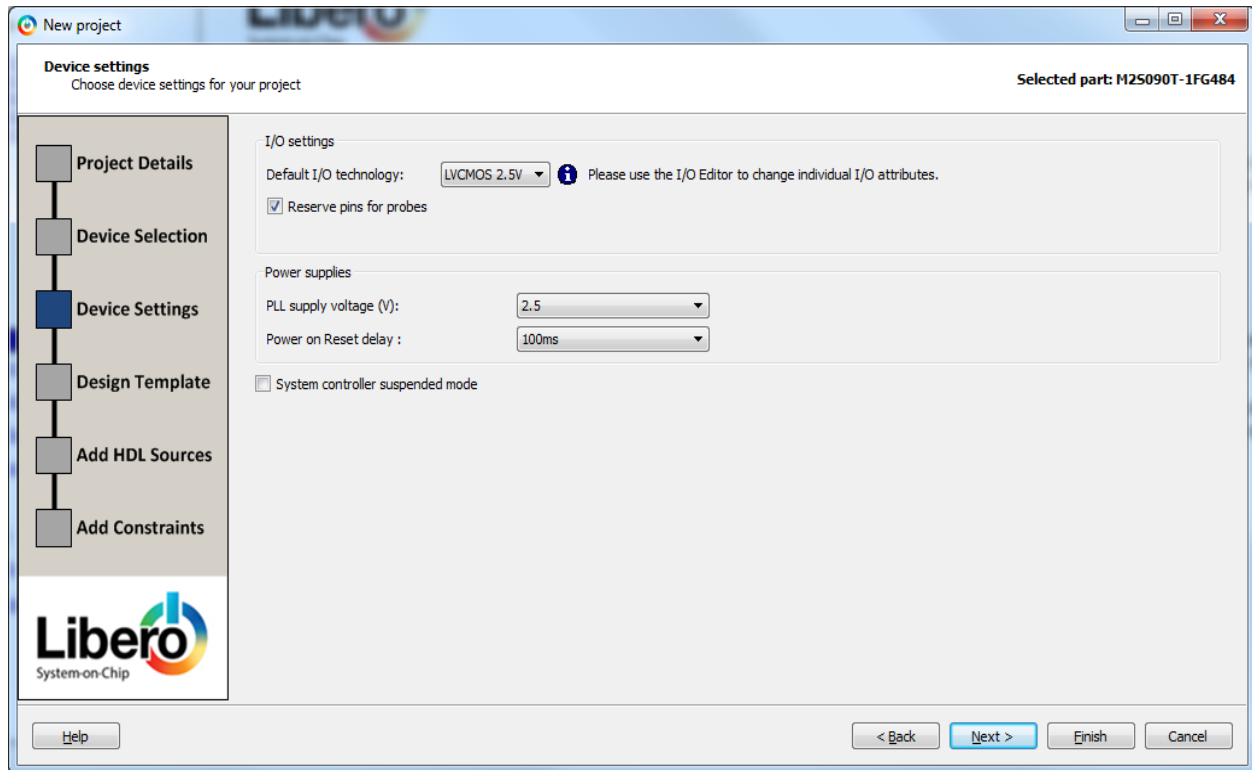
Figure 3 • Device Selection Window



6. Click **Next**. The **Device Settings** window is displayed.

7. In the **Device Settings** window, select the following information from the respective drop-down lists, as displayed in the following figure.
 - Default I/O Technology: LVCMOS 2.5V
 - PLL Supply Voltage (V): 2.5
 - Power on Reset delay: 100ms
 - System controller suspend mode: Unchecked

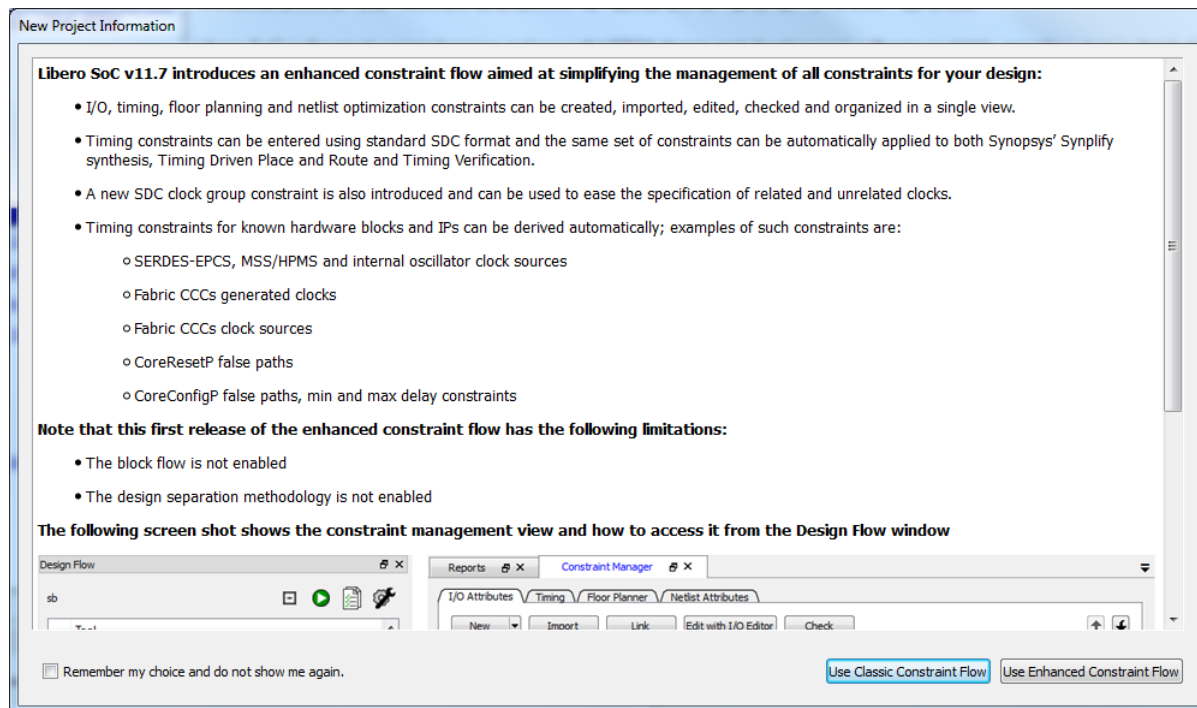
Figure 4 • Device Settings Window



8. Click **Next**. The **Design Template** window is displayed.
9. On the **Design Template** page, select the **Create a System Builder based design** under the **Design Templates and Creators**.
10. Click **Finish**. A **New Project information** selection screen is displayed.

11. Click **Use Enhanced Constraint Flow**. The **Create New System Builder** dialog box is displayed.

Figure 5 • New Project Information Window



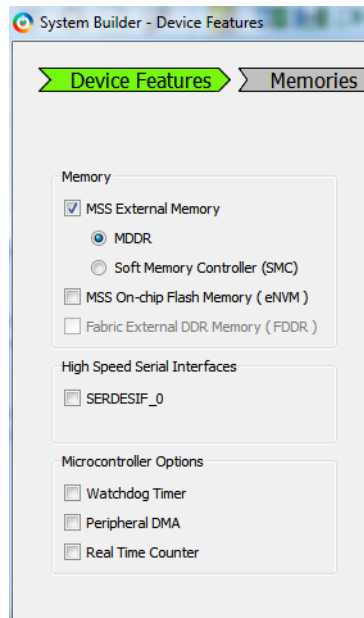
12. Enter **Top** as the name of the system, and then click **OK**, as shown in the following figure.

Figure 6 • System Name Entry



The System Builder dialog box is displayed with the **Device Features** page, as shown in the following figure.

Figure 7 • SmartFusion2 SoC FPGA System Builder Configurator



13. Under Memory, select the **MSS External Memory** check box and click **MDDR**. Leave all other options unchecked.
14. Click **Next**. The **System Builder - Memories** page is displayed, as shown in Figure 8, page 11. The DDR3 external memory models are used in this tutorial.

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

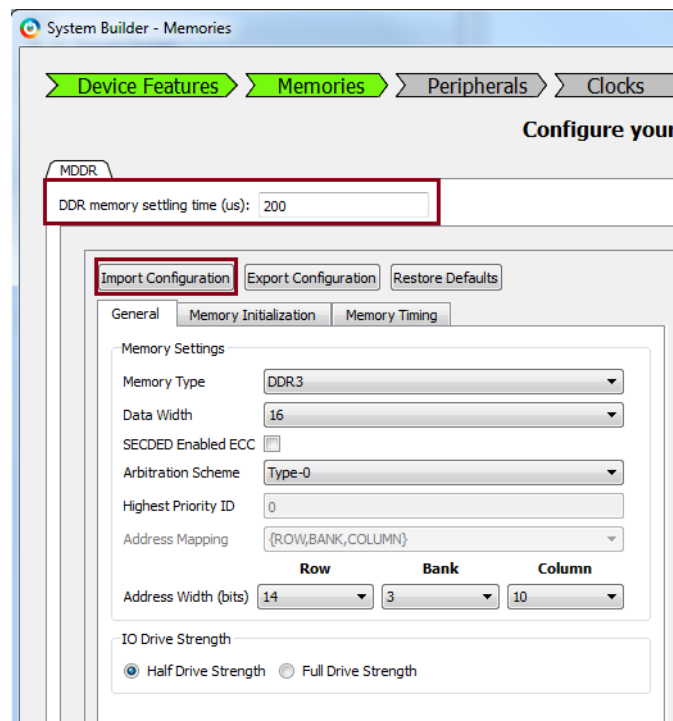
The DDR controller must be configured to match the external DDR3 memory specifications. The configuration is done through the CoreConfigP soft IP core, which is the master of the configuration data initialization process. Upon reset, the soft CoreConfigP copies the data from eNVM to the configuration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

Import the register configuration file as follows:

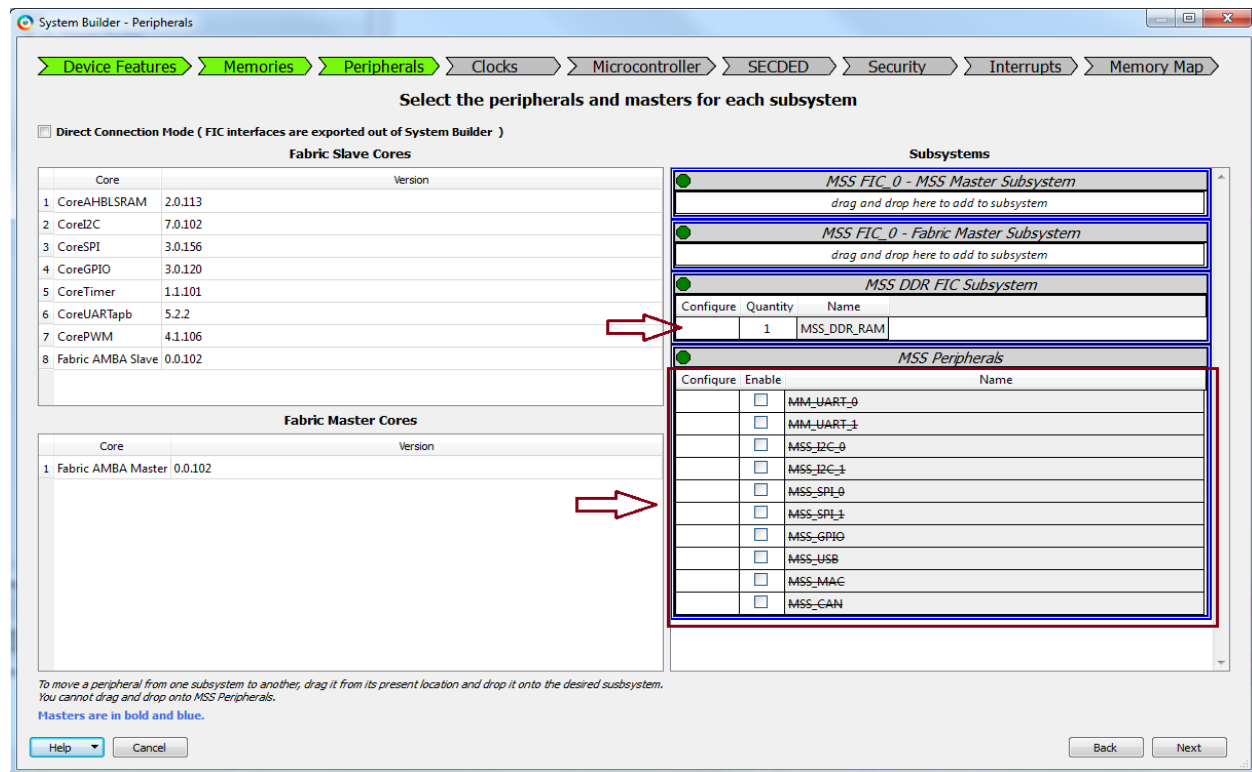
1. Click **Import Configuration**, as shown in Figure 8, page 11. The **Import File** window is displayed.
2. Browse to the provided DDR3 configuration file `DDR3_PHY_16_NO_ECC_BL8_INTER.txt` and import it. The register configuration file is imported.
3. Select the following settings from the respective list:
 - Memory Type: DDR3
 - Data Width: 16
 - SECEDED Enabled ECC: Unchecked

Figure 8 • System Builder Configurator – Memories Page



4. Click **Next**, **System Builder - Peripherals** page is displayed, as shown in Figure 9, page 12. This tutorial does not use any of the MSS peripherals. Clear all the MSS Peripherals. As this system uses 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, page 12.

Figure 9 • System Builder Configurator – Peripherals Page

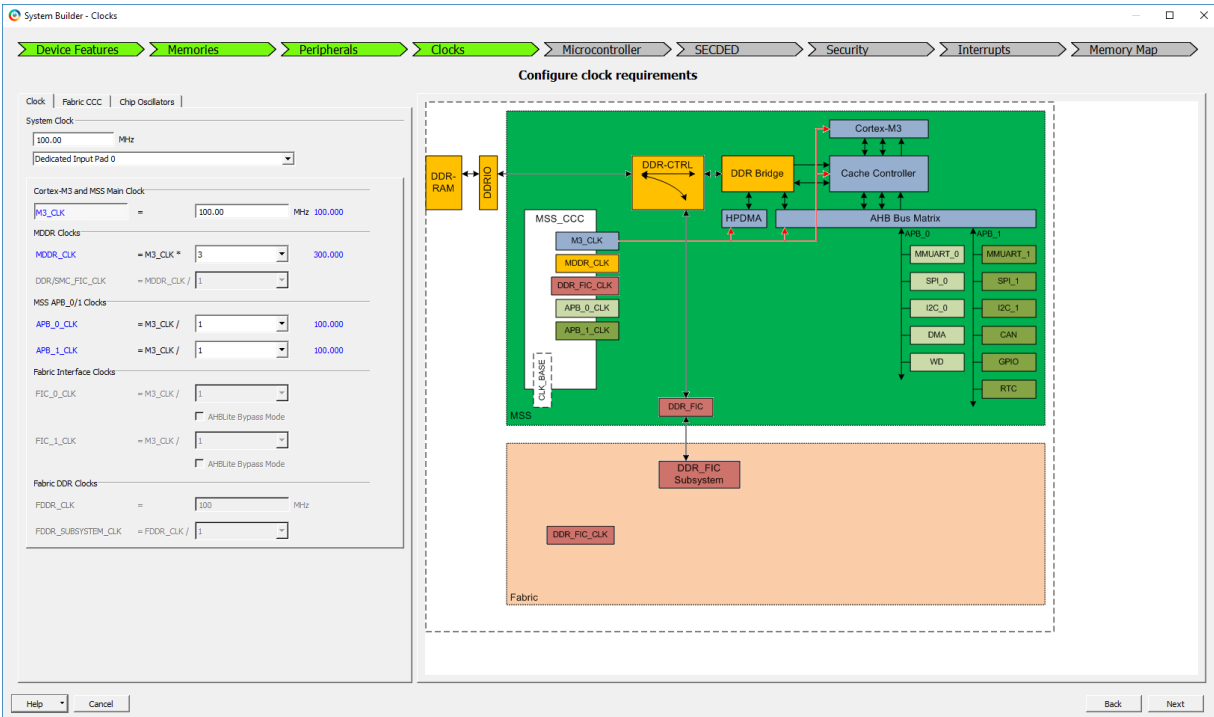


5. Click **Next**. The System Builder- **Clocks** page is displayed, as shown in Figure 10, page 13.

6. 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 list 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 **MDDR_CLK** shown in Figure 10, page 13 and the clock and blocks that the clock is driving are displayed on the right-side panel.

Figure 10 • System Builder Configurator – Clocks Page

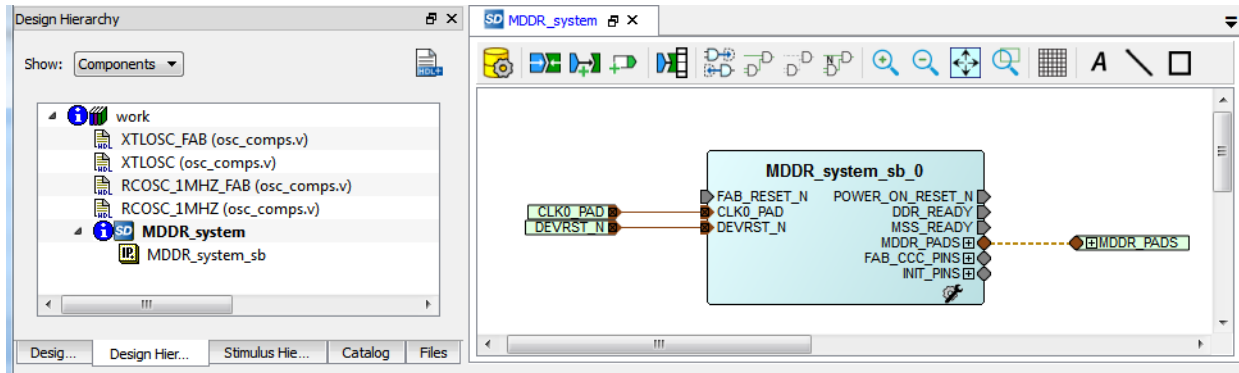


7. Click **Next**, the System Builder - **Microcontroller Options** page is displayed. Leave all the default selections.
8. Click **Next**, the System Builder - **SECEDED Options** page is displayed. Leave all the default selections.
9. Click **Next**, the System Builder - **Security Options** page is displayed. Leave all the default selections.
10. Click **Next**, the System Builder - **Interrupts Options** page is displayed. Leave all the default selections.

11. Click **Next**, the System Builder - **Memory Map Options** page is displayed. Leave all the default selections.
12. Click **Finish**. The System Builder generates the system based on the selected options.

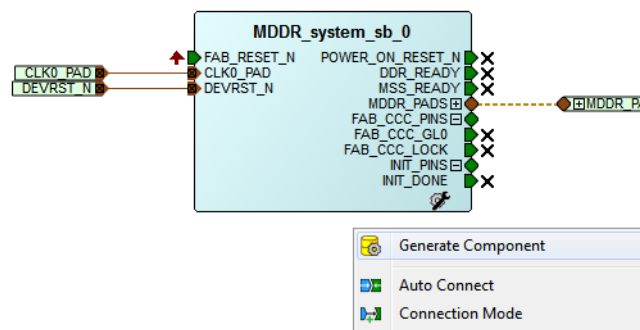
The System Builder block is created and added to the Libero SoC project, as shown in the following figure. The CoreResetP and CoreConfigP cores are automatically instantiated and connected by the System Builder. For more information about how the blocks are connected in the System Builder component, refer to [Opening the MDDR System in SmartDesign](#), page 15.

Figure 11 • SmartFusion2 SoC FPGA System Generated by System Builder



13. 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_GLO** (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**.
14. Generate the final system by clicking **SmartDesign > Generate Component** or by clicking **Generate Component** icon on the SmartDesign toolbar.
15. Right-click **canvas** and select **Generate Component**, as shown in the following figure. After the successful generation of the system, the message "Info: 'Top' was successfully generated".

Figure 12 • SmartFusion2 SoC FPGA Final System



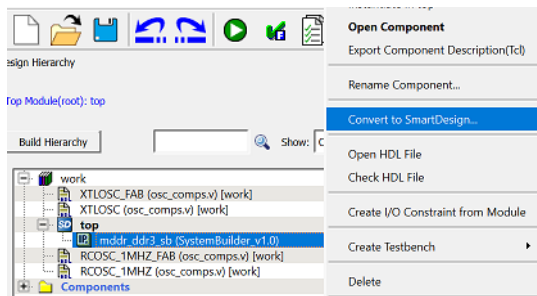
2.4.1 Opening the MDDR System in SmartDesign

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, page 14. 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 SmartDesign. This enables you to check the internals of the overall design. To open the Top using SmartDesign, use the following steps:

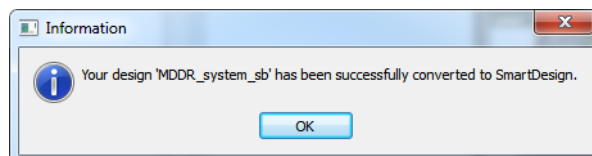
1. In the **Design Hierarchy**, expand **Top** component.
2. Right-click **mddr_dds3_sb** and select **Convert to SmartDesign**, as shown in the following figure.

Figure 13 • Open as SmartDesign Option



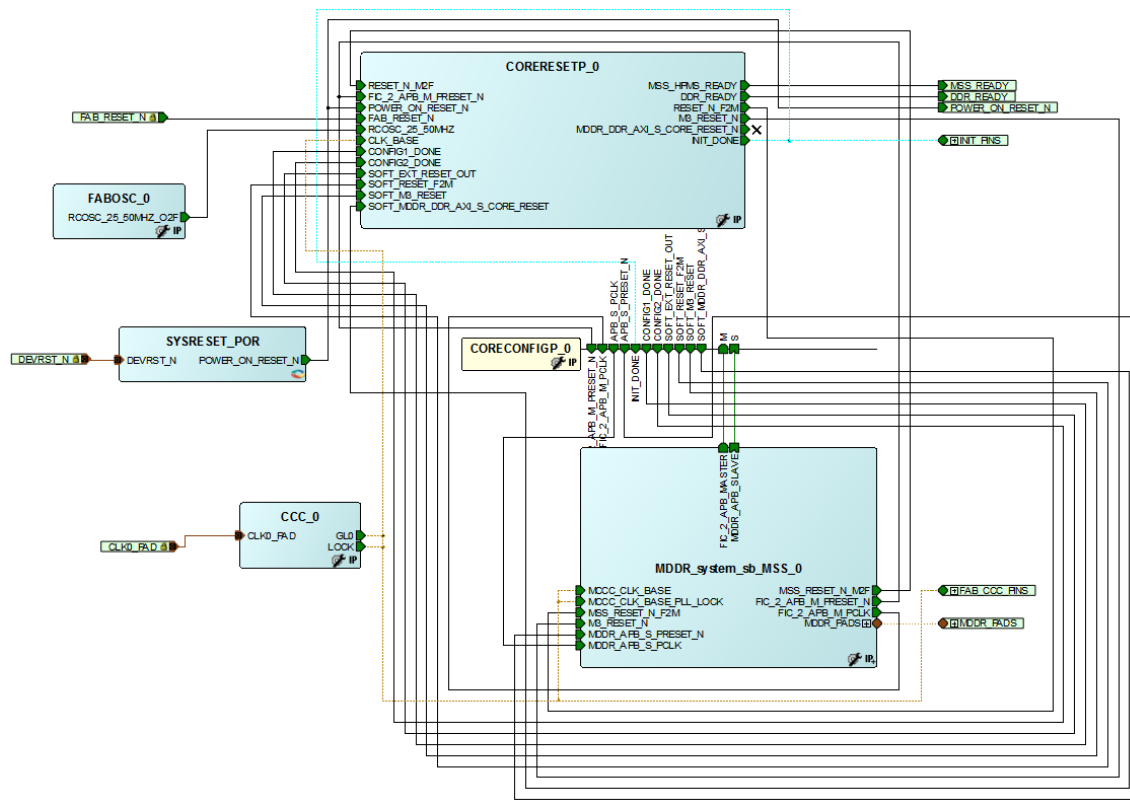
The system is converted to a SmartDesign component, and the message is displayed, as shown in the following figure.

Figure 14 • Confirmation Message About System Builder Conversion to SmartDesign



3. Click **OK**. The system is shown in the SmartDesign canvas, as shown in the following figure.

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



The System Builder automatically instantiates and connects different blocks based on the different options that you have selected in the different pages of the System Builder.

- **SYSRESET_POR**: Generates the power-on reset signal for the CoreResetP block.
- **CORERESETP_0** (soft core): Manages all the reset mechanisms needed for the system.
- **FABOSC_0**: Generates the clock source for the CoreResetP block.
- **CCC_0**: Generates 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): Manages the configuration aspect of the controller based on the specified configuration file.

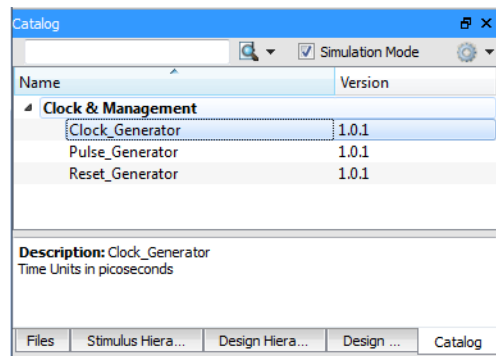
2.4.2 Generating the Testbench

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 the **Simulation Mode** check box in the Libero SoC IP catalog, as shown in the following figure.
2. Under the **Clock & Management**, the IP catalog displays three simulation cores to drive the device under test (DUT):
 - Clock_Generator
 - Pulse_Generator
 - Reset_Generator

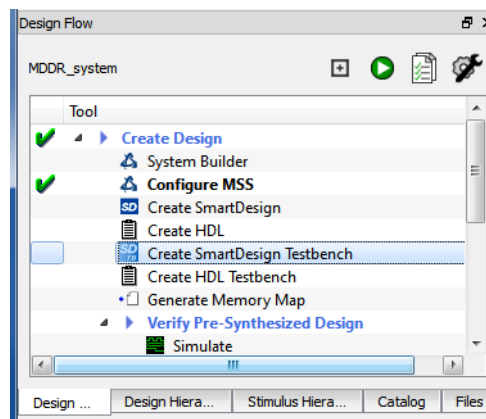
Note: If they appear in italic, double-click to download the cores to your local vault.

Figure 16 • Simulation Cores in Libero IP Catalog



3. Double-click **Create SmartDesign Testbench** in the **Libero Design Flow** window, as shown in the following figure.

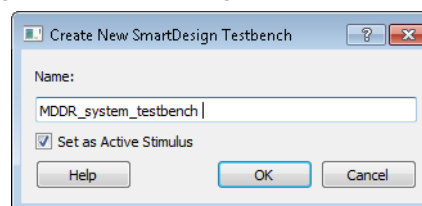
Figure 17 • Libero Design Flow – Create SmartDesign Testbench Option



The **Create New SmartDesign Testbench** dialog box is displayed, as shown in the following figure.

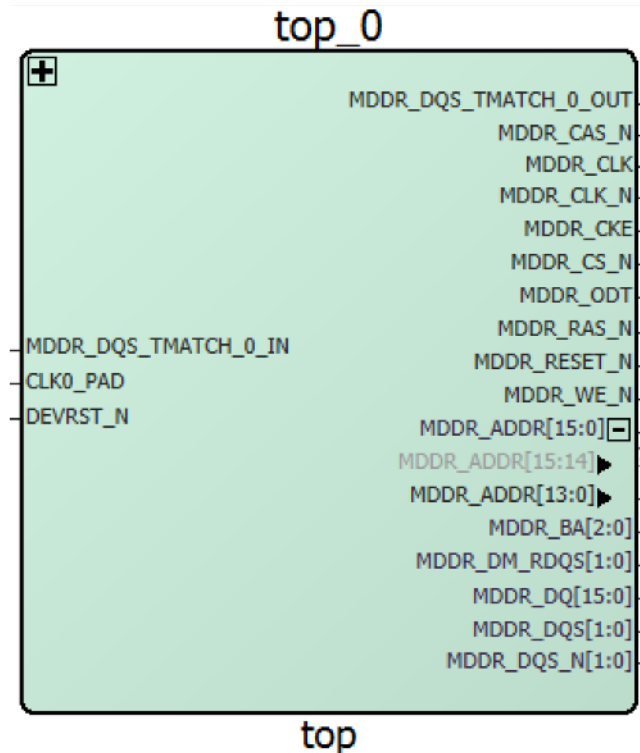
4. Enter **MDDR_system_testbench** in the **Create New SmartDesign Testbench** dialog box and then click **OK**.

Figure 18 • Create New SmartDesign Testbench Dialog Box



The SmartDesign canvas is displayed with the top_0 component instantiated, as shown in the following figure.

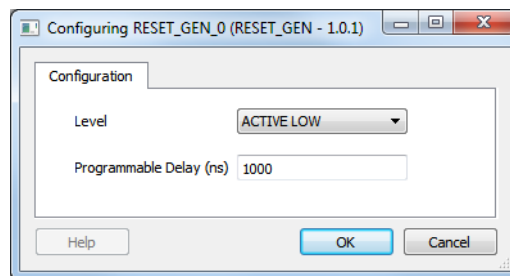
Figure 19 • SmartDesign Testbench Canvas



5. Drag the **Clock_Generator** and **Reset_Generator** simulation cores from the IP catalog to the MDDR_system_testbench SmartDesign canvas.
6. Open the **Reset_Generator** configurator by double-clicking **RESET_GEN_0** in the SmartDesign canvas.
7. Ensure that the following options, as shown in the following figure, are set in the RESET_GEN_0 configurator and then click **OK**:
 - Level: ACTIVE LOW (default)
 - Programmable Delay (ns): 1000

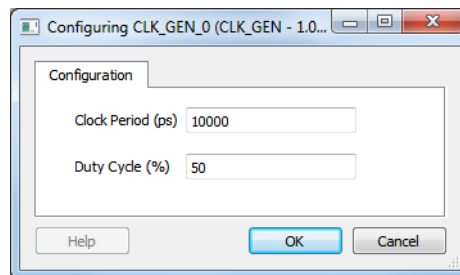
The reset generator provides the reset pulse for the simulation.

Figure 20 • RESET_GEN Configuration



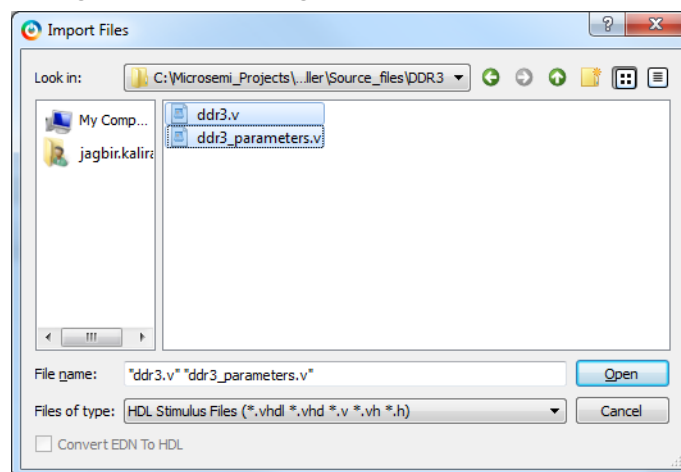
8. Open the **Clock_Generator** configurator by double-clicking the CLK_GEN_0 in the SmartDesign canvas.
9. Ensure that the following information, as shown in the following figure, is set in the CLK_GEN_0 configurator and then click **OK**:
 - Clock Period (ps): 10000
 - Duty Cycle (%): 50

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

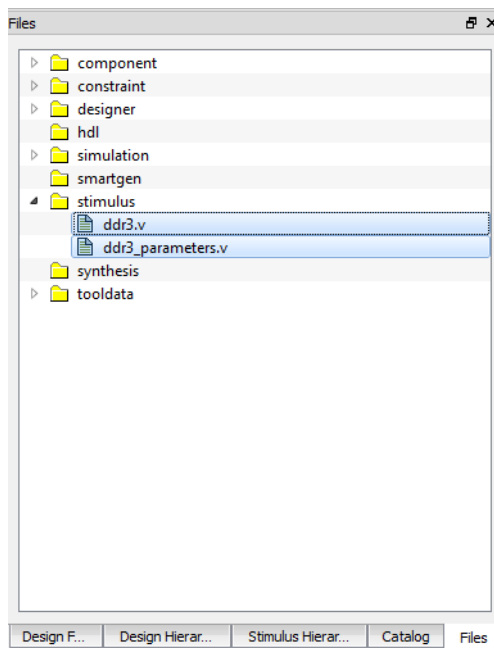
Figure 21 • CLK_GEN Configuration

10. Import the provided DDR3 model into the Libero SoC project and instantiates those model 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.** The **Import Files** dialog box is displayed.
 - Select the **HDL Stimulus Files (*.vhd and *.v)** option from the **Files of type**, as shown in the following figure.
 - Select the provided **ddr3.v** and the **ddr3_parameters.v** files and then click **Open**, as shown in the following figure.

The files are located under the
 <project directory>\m2s_tu0372_dflSource_files\DDR3 folder.

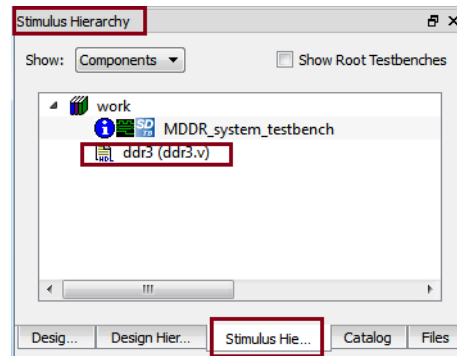
Figure 22 • Import Files Dialog Box (For Importing DDR3 Models to Stimulus)

11. Verify that the files are imported correctly as stimulus files by checking under the Stimulus folder in the **Files** tab, as shown in the following figure.

Figure 23 • Imported DDR3 in Stimulus Folder

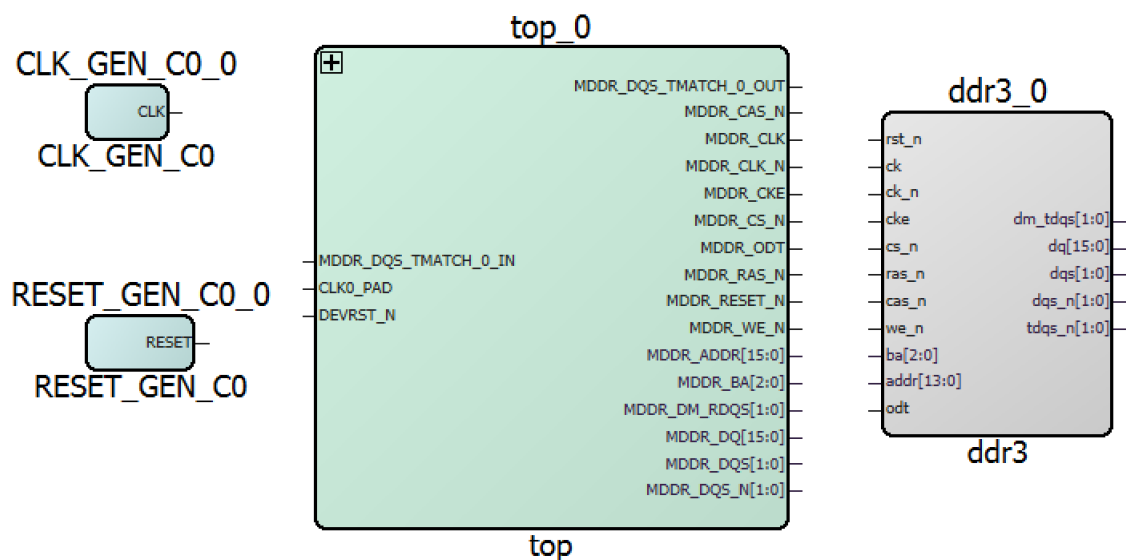
When the file is imported as a Stimulus, the file is displayed in the **Stimulus Hierarchy** window, as shown in the following figure.

Figure 24 • Stimulus Hierarchy Window



12. From the **Stimulus Hierarchy** window, drag the **ddr3** file onto the **MDDR_system_testbench** canvas.
You are 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 instantiating the DDR3, the canvas is displayed, as shown in the following figure.

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 the following figure. 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-and-drop to the second pin that you want to connect.

Figure 26 • Connection Mode Icon



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.

13. After connecting the pins by one of the connection methods, make the following connections in the SmartDesign canvas among **RESET_GEN_0**, **CLK_GEN_0**, and **top_0**:
 - From **RESET_GEN_0: RESET** to **top_0: DEVRST_N**
 - From **CLK_GEN_0:CLK** to **top_0:CLK0_PAD**

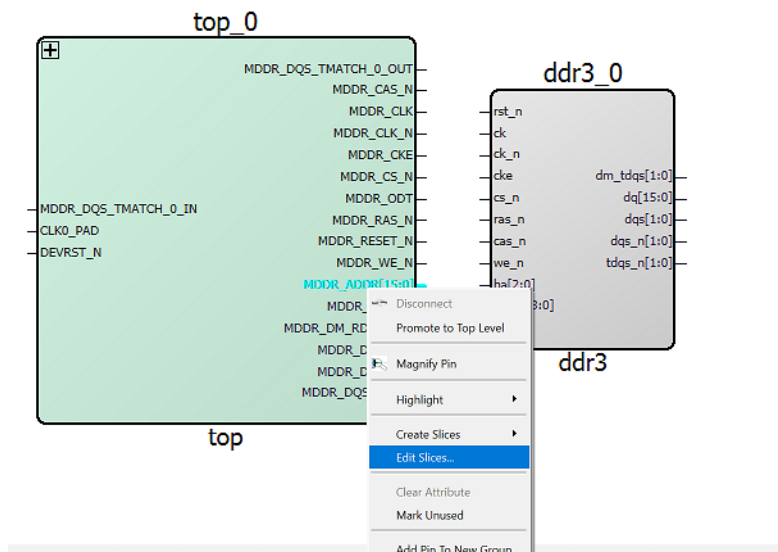
There are buses on the **top_0** and the **ddr3_0** that do not match in width. To connect those buses, you need to slice them first to create an equivalent bus width that matches between the **top_0** and the **ddr3_0**.

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:

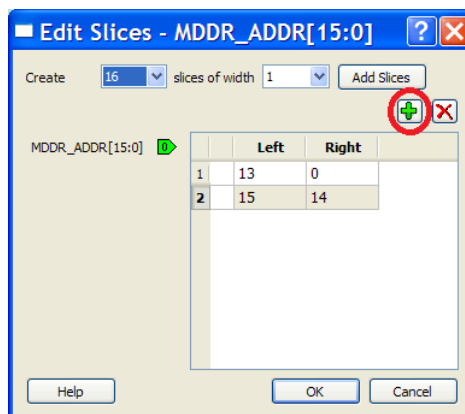
1. Right-click bus and select **Edit Slice**, as shown in the following figure. The dialog box is displayed, as shown in [Figure 28](#), page 22.

Figure 27 • Edit Slice Option



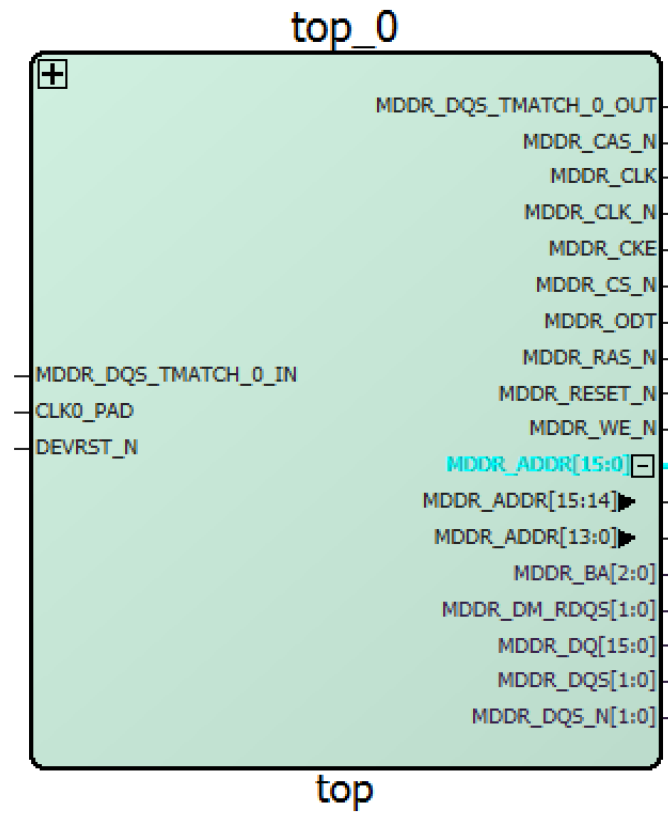
2. Click  icon, as highlighted in the following figure to add a slice.
3. Double-click  icon to add two slices and enter the slice details, as shown in the following figure.

Figure 28 • Edit Slices Dialog Box



4. Click **OK**. The MDDR_ADDR bus is divided into two slices, as shown in the following figure.

Figure 29 • MDDR_ADDR Slices



5. Expand the **top_0: MDDR_PADS**.

6. After connecting the pins by following one of the methods described, make the following connections in the SmartDesign canvas between the **top_0** and the **ddr3_0**:
- Connect **MDDR_DQS_TMATCH_0_IN** to **MDDR_DQS_TMATCH_0_OUT** of the **top_0** block.
 - Connect the rest of the pins, as listed in the following table.

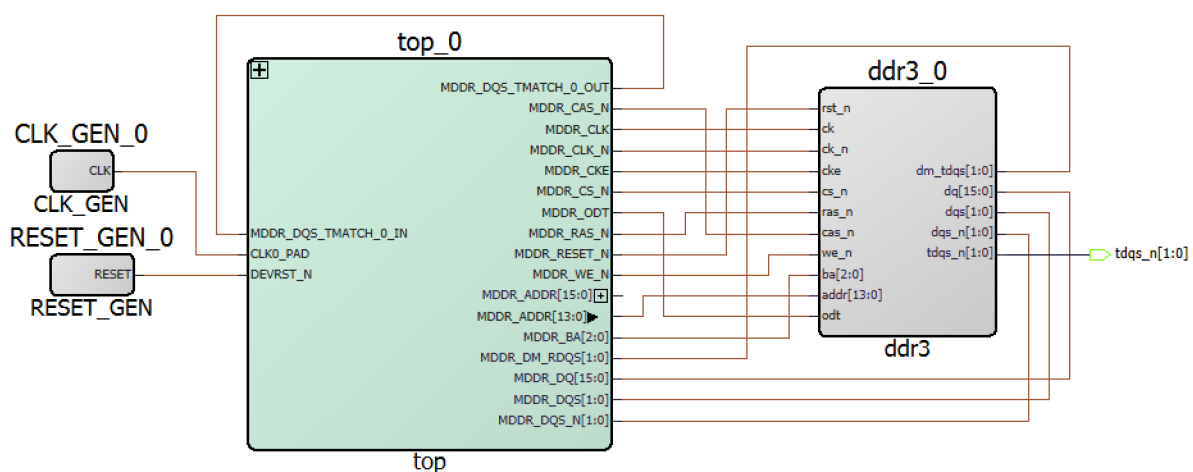
Table 2 • DDR3 Pin Connections

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_tdq[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

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

After connecting all the pins, the canvas is displayed, as shown in the following figure.

Figure 30 • Fully-Connected MDDR System

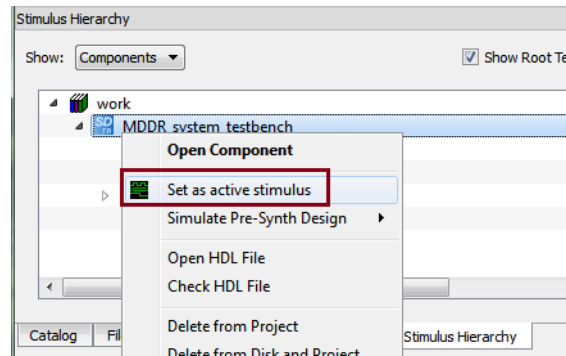


8. Generate the final system testbench by clicking **SmartDesign > Generate Component** or by clicking **Generate Component** icon on the SmartDesign toolbar. You can also right-click on the canvas and select **Generate Component**.

On successful generation, the message "Info: 'MDDR_system_testbench' was successfully generated" is displayed on the log window:

9. After generating the testbench, you need to make it an 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. Right-click **MDDR_system_testbench** and select **Set as active stimulus**, as shown in the following figure. If the **Set as active stimulus** option is not displayed, the **MDDR_system_testbench** is already in active stimulus mode.

Figure 31 • Set as active stimulus Option

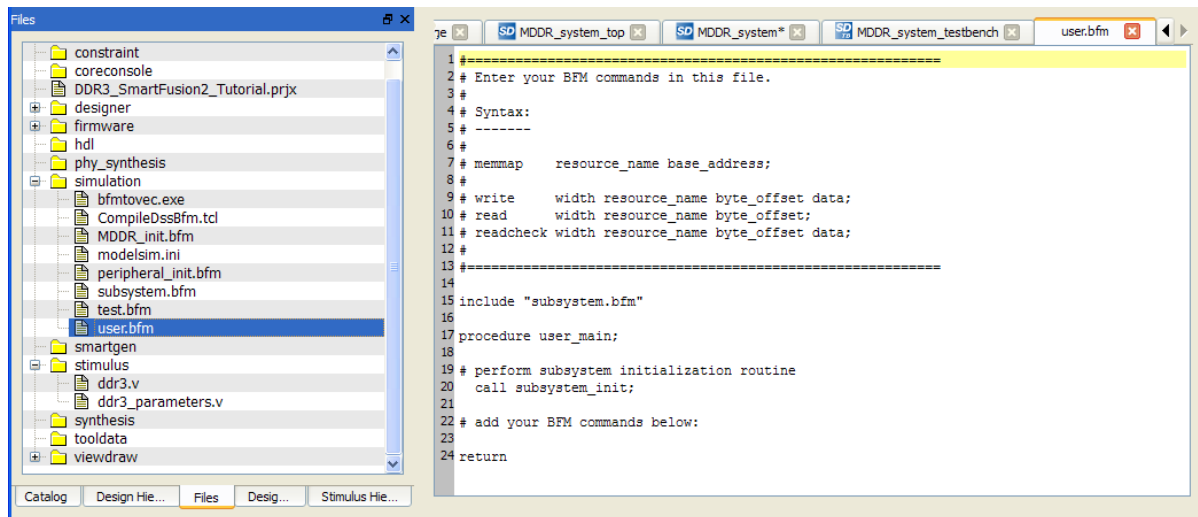


2.4.3 Modifying the BFM Script

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

1. Go to the **Files** tab > **Simulation** folder, and double-click **user.bfm** file. The BFM file is displayed, as shown in the following figure.

Figure 32 • Files Tab with BFM File Opened



2. Modify the `user.bfm` to add the following BFM commands associated with 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;`

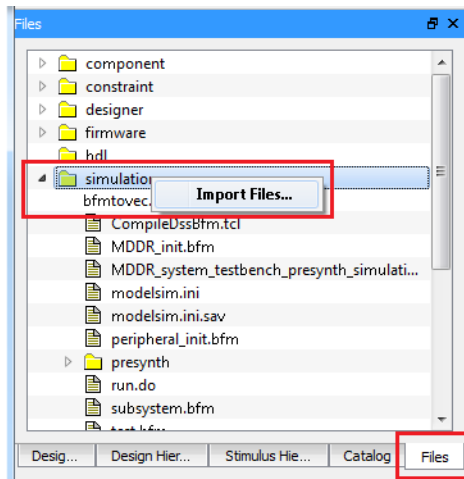
```
readcheck w M_MDDR0_SPACE_0 0x0010 0xABCDEF01;
readcheck w M_MDDR0_SPACE_0 0x0014 0xCCBBAADD;

print "TEST ENDS";
```

Note: The 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:

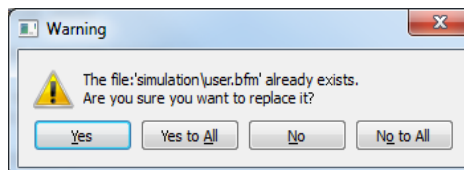
3. Go to **Files** tab and right-click **simulation** folder, as shown in the following figure.

Figure 33 • Files Tab - Import Files Option (For Importing BFM Source Files)



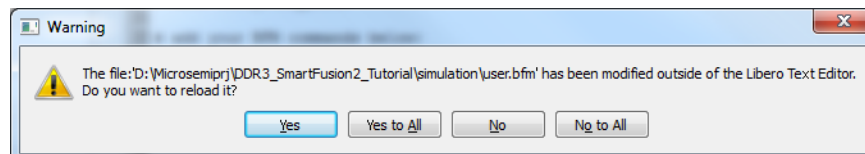
4. Browse to `<project directory>\m2s_tu0372_df\Source_files` and select `user.bfm` and select **Open**. A warning message is displayed, as shown in the following figure.
5. Click **Yes**.

Figure 34 • Warning Message for Replacing Existing BFM File



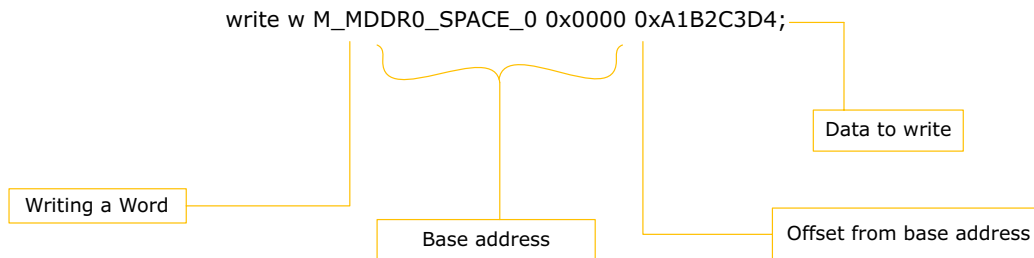
6. A warning message is displayed as shown in the following figure, if the `user.bfm` file is already opened in your Libero window. Click **Yes**. If the `user.bfm` is not opened already in your Libero window, the warning message is not displayed.

Figure 35 • Warning Message when BFM File is Already Open



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

- **Step a:** You specify the base address at which the MDDR is located. For example, the base address is 0xA0000000.
- **Step b:** You write different values to different locations. For example,



- **Step c:** You check what is written. The final `user.bfm` is displayed, as shown in the following figure.

Figure 36 • BFM File After Adding the Commands

```

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

```

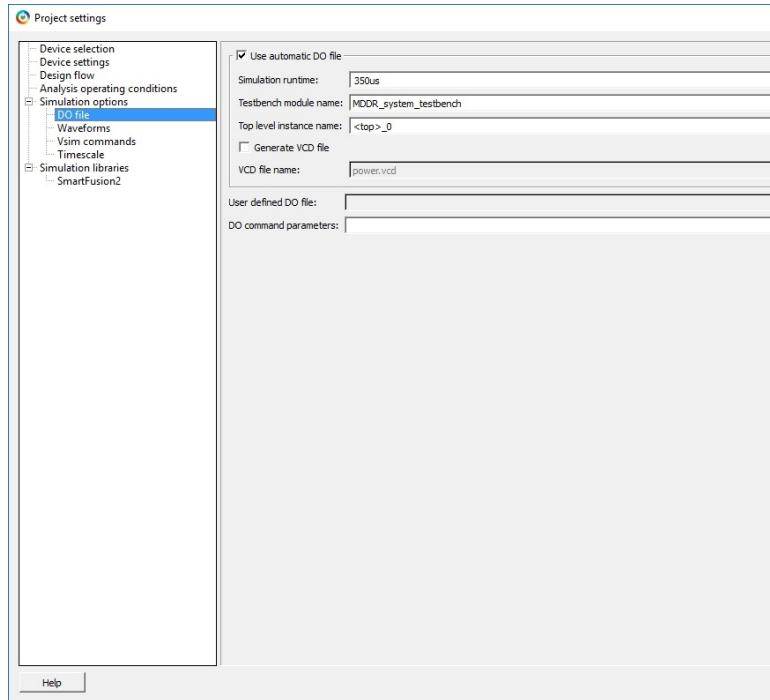
For more information, refer to *DirectCore Advanced Microcontroller Bus Architecture - Bus Functional Model User Guide*.

2.4.4 Running the Simulation

The following steps describe how to simulate the design using the SmartDesign testbench and the user BFM 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.
3. Enter the **Simulation runtime** as 350 μ s, as shown in the following figure.

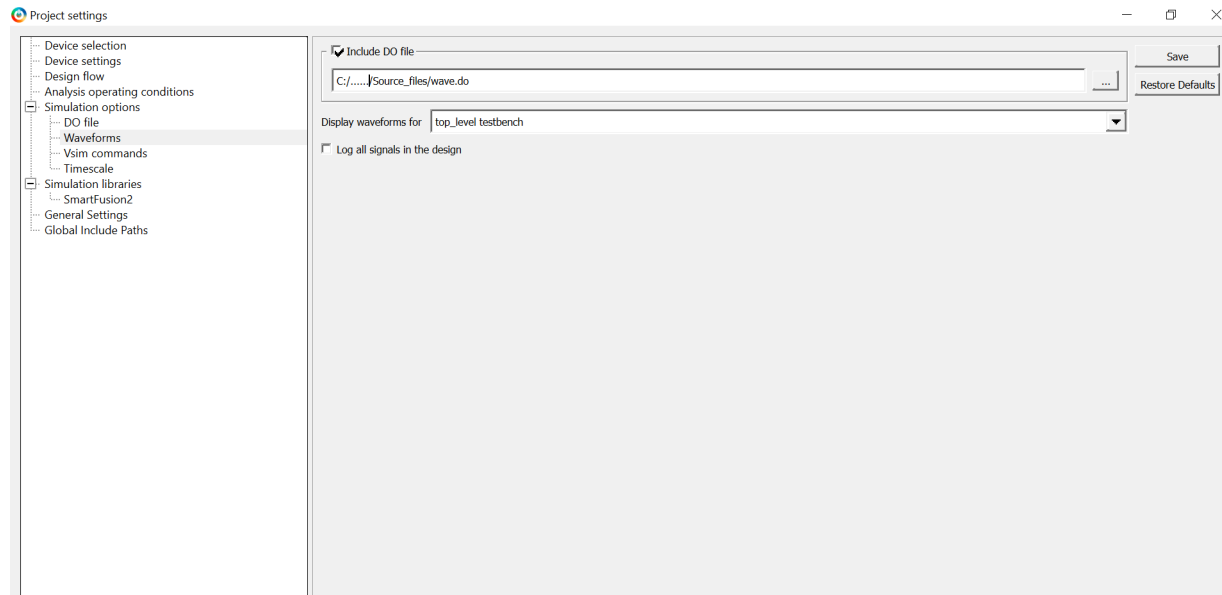
Figure 37 • Project Settings Window – Do File Simulation Runtime Setting



4. 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, page 30. 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.

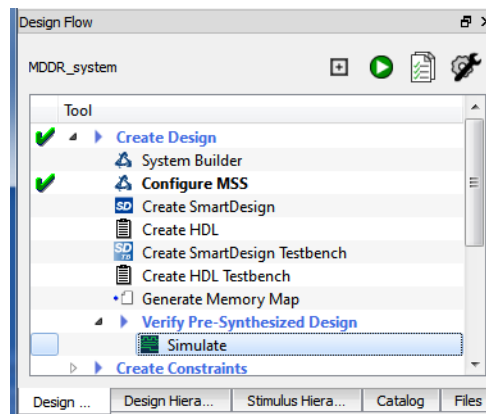
Note: The path is modified to use \${PROJECT_DIR} so that the path is always relative to the project directory.

Figure 38 • Project Settings Window – MDDR_wave.do File Location



5. Expand **Verify Pre-Synthesized Design** in the **Design Flow** window, as shown in the following figure.
6. Double-click **Simulate** to launch ModelSim in GUI mode.

Figure 39 • Design Flow Tab – Simulate Option



2.4.5 Validating the Simulation Results

1. ModelSim runs the design for about 260 μ s, as specified in the **Project Settings** window. It initializes the MDDR by writing a specific set of configuration options to the configuration registers. After recording the configuration details in the registers, 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 the following figure. 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 the following figure. Scroll down to see different commands. In the BFM script provided in the `user.bfm` file, 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

```

# BFM: Data Read a0000010 abcdef01 MASK:ffffffff at 308865.020000ns
# MDDR_system_testbench.ddr3_0.cmd_task: at time 308801897 ps INFO: Read bank 0 col 008, auto precharge 0
# MDDR_system_testbench.ddr3_0.data_task: at time 308820230 ps INFO: READ @ DQS= bank = 0 row = 0000 col = 00000008 data = ef01
# MDDR_system_testbench.ddr3_0.data_task: at time 308821897 ps INFO: READ @ DQS= bank = 0 row = 0000 col = 00000009 data = abcd
# MDDR_system_testbench.ddr3_0.data_task: at time 308823563 ps INFO: READ @ DQS= bank = 0 row = 0000 col = 0000000a data = aadd
# MDDR_system_testbench.ddr3_0.data_task: at time 308825230 ps INFO: READ @ DQS= bank = 0 row = 0000 col = 0000000b data = cccb
# MDDR_system_testbench.ddr3_0.data_task: at time 308826897 ps INFO: READ @ DQS= bank = 0 row = 0000 col = 0000000c data = xxxx
# MDDR_system_testbench.ddr3_0.data_task: at time 308828563 ps INFO: READ @ DQS= bank = 0 row = 0000 col = 0000000d data = xxxx
# MDDR_system_testbench.ddr3_0.data_task: at time 308830230 ps INFO: READ @ DQS= bank = 0 row = 0000 col = 0000000e data = xxxx
# MDDR_system_testbench.ddr3_0.data_task: at time 308831897 ps INFO: READ @ DQS= bank = 0 row = 0000 col = 0000000f data = xxxx
# BFM: Data Read a0000014 ccbbaadd MASK:ffffffff at 308965.020000ns
# MDDR_system_testbench.ddr3_0.cmd_task: at time 308901897 ps INFO: Read bank 0 col 008, auto precharge 0
# MDDR_system_testbench.ddr3_0.data_task: at time 308920230 ps INFO: READ @ DQS= bank = 0 row = 0000 col = 00000008 data = ef01
# MDDR_system_testbench.ddr3_0.data_task: at time 308921897 ps INFO: READ @ DQS= bank = 0 row = 0000 col = 00000009 data = abcd
# MDDR_system_testbench.ddr3_0.data_task: at time 308923563 ps INFO: READ @ DQS= bank = 0 row = 0000 col = 0000000a data = aadd
# MDDR_system_testbench.ddr3_0.data_task: at time 308925230 ps INFO: READ @ DQS= bank = 0 row = 0000 col = 0000000b data = cccb
# MDDR_system_testbench.ddr3_0.data_task: at time 308926897 ps INFO: READ @ DQS= bank = 0 row = 0000 col = 0000000c data = xxxx
# MDDR_system_testbench.ddr3_0.data_task: at time 308928563 ps INFO: READ @ DQS= bank = 0 row = 0000 col = 0000000d data = xxxx
# MDDR_system_testbench.ddr3_0.data_task: at time 308930230 ps INFO: READ @ DQS= bank = 0 row = 0000 col = 0000000e data = xxxx
# MDDR_system_testbench.ddr3_0.data_task: at time 308931897 ps INFO: READ @ DQS= bank = 0 row = 0000 col = 0000000f data = xxxx
# BFM Simulation Complete - 5216 Instructions - NO ERRORS
*****
# MDDR_system_testbench.ddr3_0.cmd_task: at time 309171897 ps INFO: Precharge bank 0
# MDDR_system_testbench.ddr3_0.cmd_task: at time 309198563 ps INFO: Precharge Power Down Enter
V$IM 2>

```

After running the simulation successfully, undock the **Wave** window. To undock the window, click Dock/Undock icon, as shown in the following figure.

Figure 41 • ModelSim Wave Window – Doc/Undock Icon



3. After undocking the Wave window, click **Zoom Full** icon, as shown in the following figure to fit all the waveforms in the single view.

Figure 42 • Zoom Full Icon



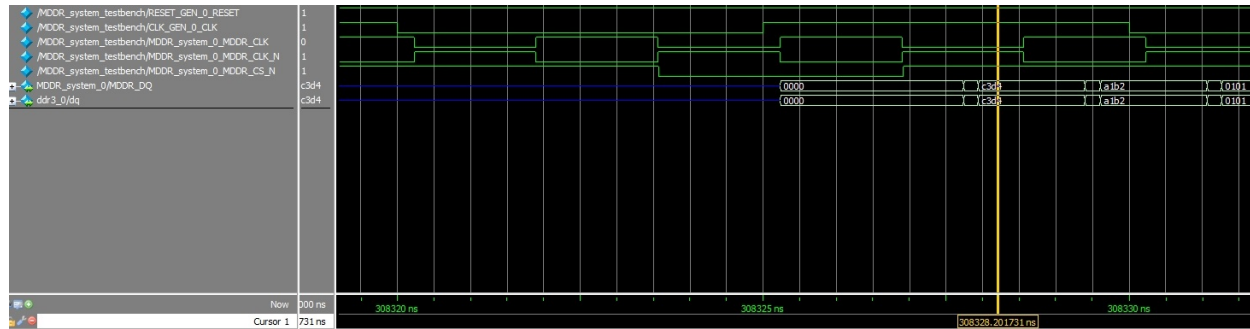
4. Place the cursor around 238 ps on the **Wave** window, and click **Zoom In on the Active Cursor** icon (shown in the following figure) to zoom in on that location.

Figure 43 • Zoom In on Active Cursor Icon



The time at which the data was written/read-back to/from the DDR3 external modules is displayed, as shown in the following figure.

Figure 44 • Write/Read Data



5. Go to **File > Quit** to exit the ModelSim simulator.

3 Appendix 1: VHDL Flow

If you plan design with VHDL, the DDR3 memory models use 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.

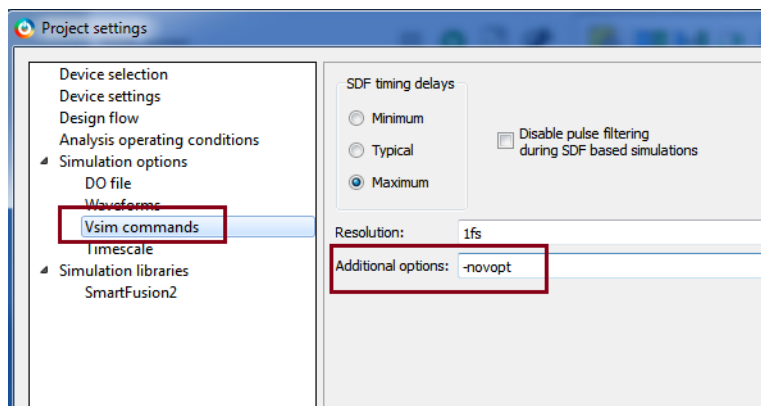
The following steps describe how to simulate VHDL design:

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.

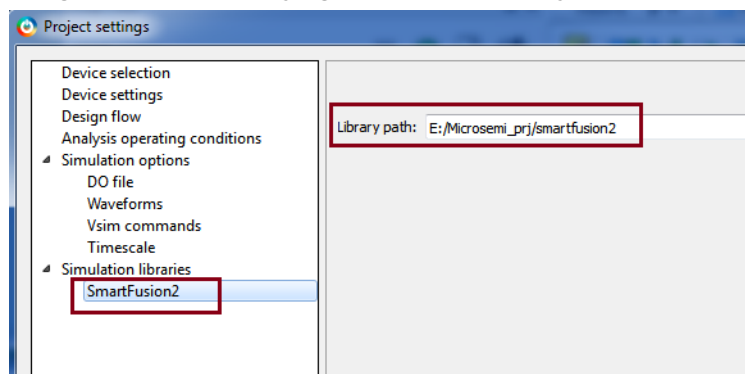
3. Simulate with automatic design optimization option disabled (`-novopt`), and point to the new precompiled library location (for example, `E:\Microsemi_prj\smartfusion2`) in the **Project Settings** window, as shown in the following figure:
 - Select **Project Settings** from the Project menu.
 - Select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

Figure 45 • Project Settings Window – Entering the -novopt Vsim Command



- 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 the following figure.

Figure 46 • Project Settings Window – Specifying Precompiled Library Path



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