Interfacing SmartFusion2 SoC FPGA with DDR3 Memory through MDDR Controller

Libero SoC System Builder Flow Tutorial

Microsemi

Table of Contents

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

Interfacing SmartFusion2 SoC FPGA with DDR3 Memory through MDDR Controller

Introduction

This tutorial demonstrates 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®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 the 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® Cortex™-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.

Upon completion of this tutorial, you will be familiar with the following:

1. Creating a Libero® System-on-Chip (SoC) v11.3 project using the SmartFusion2 SoC FPGA
2. Configuring and generating the various hardware blocks and clocking system using the System Builder
3. Creating and generating testbench using the SmartDesign testbench Generator feature
4. Performing functional level verification of the design using AMBA® BFM simulation in MentorGraphics ModelSim® Simulator
5. Using the ModelSim GUI to see the various design signals in ModelSim Waveform window

Tutorial Requirements

Software Requirements

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

- Libero SoC v11.3.
- MSS v1.1.100

Associated Project Files

You can download the associated solution and source project files along with the readme for this tutorial from Microsemi website:

http://soc.microsemi.com/download/rsc/?f=SmartFusion2_MDDR_DDR3_Tutorial_DF

Note: Extract the design files to 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(1) 32-bit AHB interface, or two(2) 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 will copy 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 will notify the CoreRestP when the register configuration phase is complete. The MSS interfaces with the CoreConfigP IP core through the APB interface (FIC_2) to initialize the MDDR controller registers based on a user specified configuration file. 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. The interface through the fabric is demonstrated in a different tutorial. In this design, you will use the System Builder 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, using the System Builder, you will configure both the MSS CCC and the Fabric CCC blocks to generate the clocks for the various elements inside the MSS and the fabric respectively.

Figure 1 shows an overall block diagram of the different blocks used in this design.

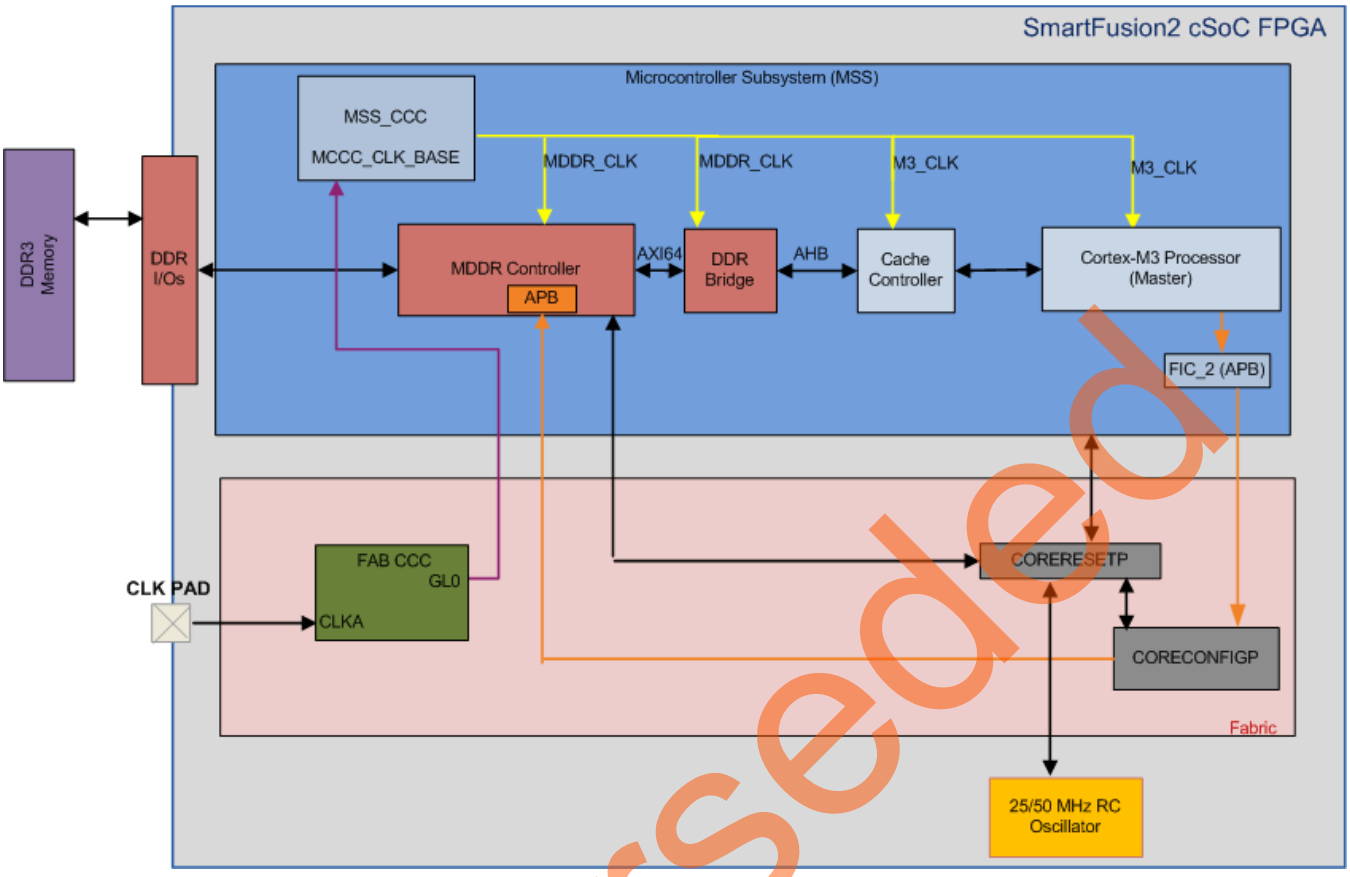


Figure 1. Top-Level Block Diagram

Design Steps

Step I: Creating a Libero SoC Project

1. Launch Libero SoC v11.3.
2. From the **Project** menu, select **New Project**. Enter the information as displayed in Figure 2.
   - Name: DDR3_SmartFusion2_Tutorial
   - Location: Select an appropriate location (For example, C:/Microsemi_prj)
   - Preferred HDL Type: Leave as Verilog
   - Family: SmartFusion2
   - Die: M2S050TS
   - Package: 896 FBGA
   - Speed: -1
   - Die Voltage: 1.2
   - Operating Conditions: IND
3. Select **Use System Builder** in the Design Templates and Creators of the New Project window.
4. Click **OK** on the New Project window.

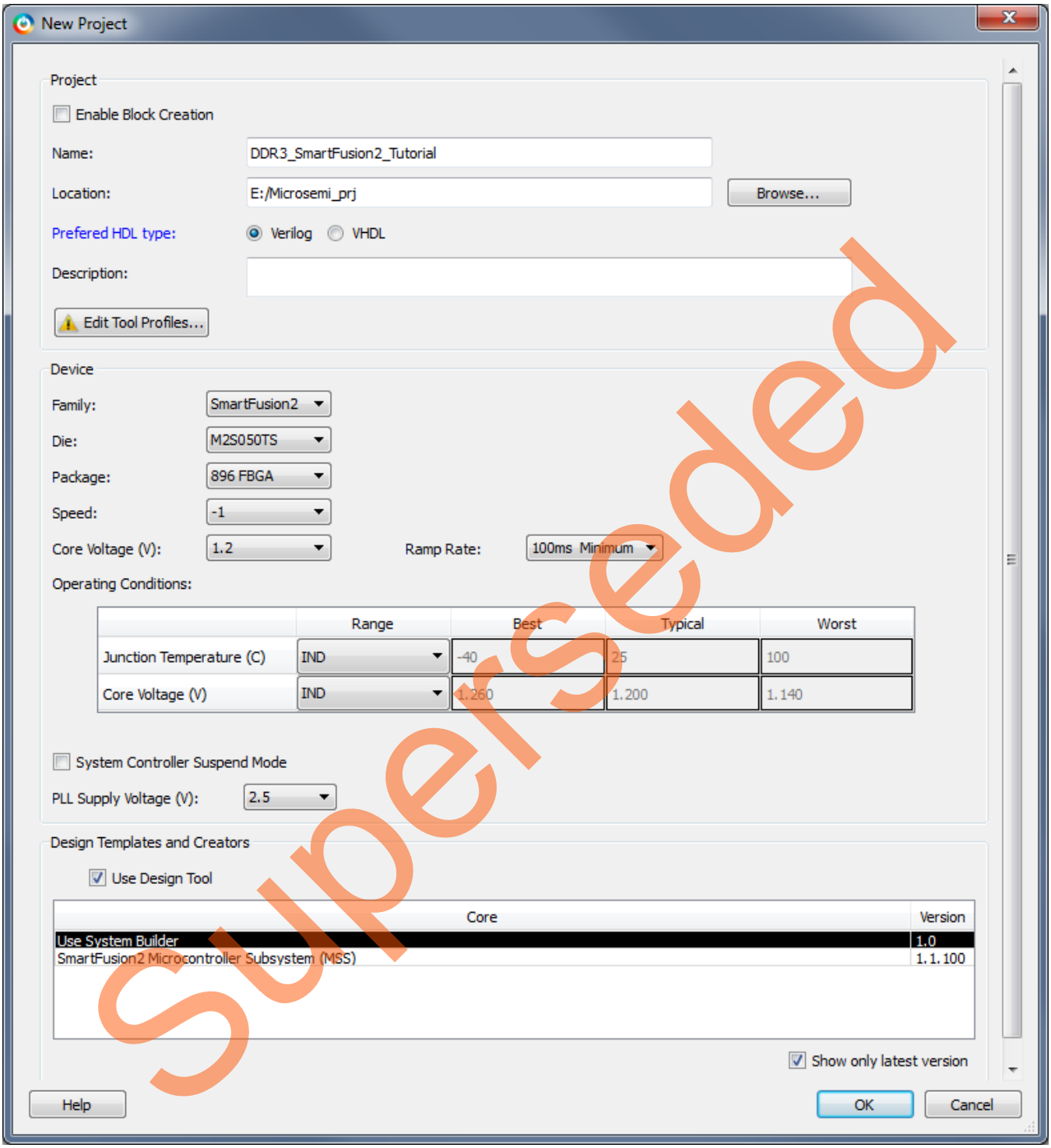


Figure 2. New Project Dialog Box

5. Since you selected **Use System Builder**, as shown in Figure 2, the **System Builder** window opens, to enter a name for your system, as shown in Figure 3.

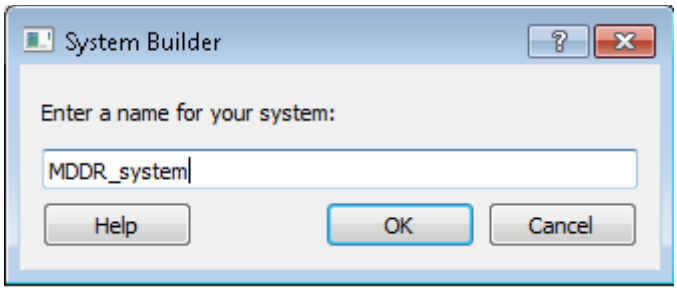


Figure 3. Create New System Builder Dialog Box

6. Enter **MDDR_system** as the name of the system and click **OK**. The System Builder dialog box is displayed with the **Device Features** page, as shown in Figure 4.

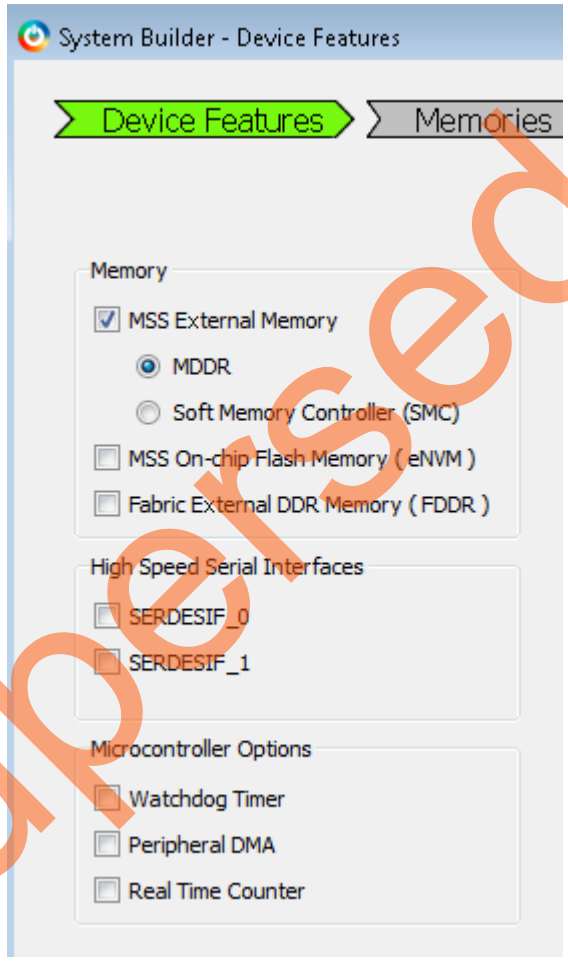


Figure 4. SmartFusion2 SoC FPGA System Builder Configurator

7. Under Memory, check **MSS External Memory** and select **MDDR**. Leave all other options unchecked.
8. Click **Next**, the System Builder – **Memory** page is displayed, as shown in Figure 5. You will be using the DDR3 external memory models for the purpose of this tutorial. Set the memory setting time to 200 us.
   - 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 will copy 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 5.
- The **Import File** window is displayed. Browse to the provided DDR3 configuration file **DDR3_PHY_16_NO_ECC_BL8_INTER.txt** and import it.

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

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

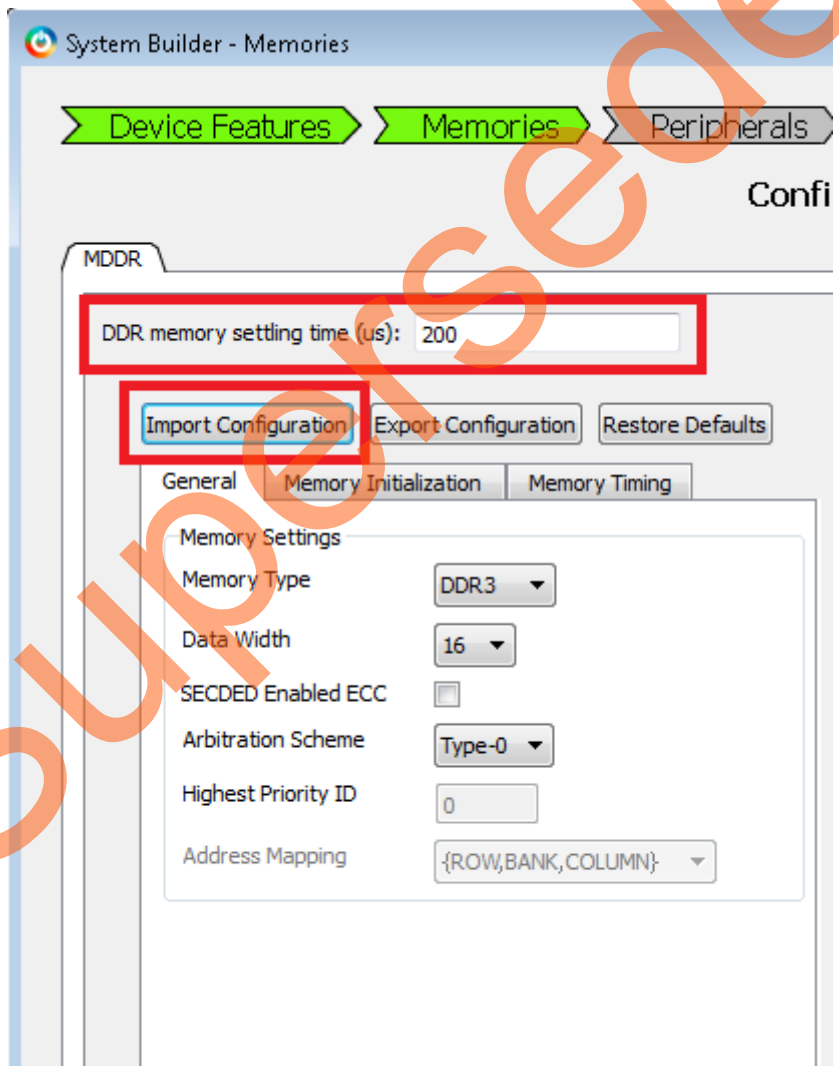


Figure 5. System Builder Configurator – Memory Page

9. Select **Next**, System Builder- **Select Peripherals** page is displayed, as shown in Figure 6.

   This tutorial will not use any MSS peripherals, therefore clear all the **MSS Peripherals** (marked on Figure 6 below).

   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 6.

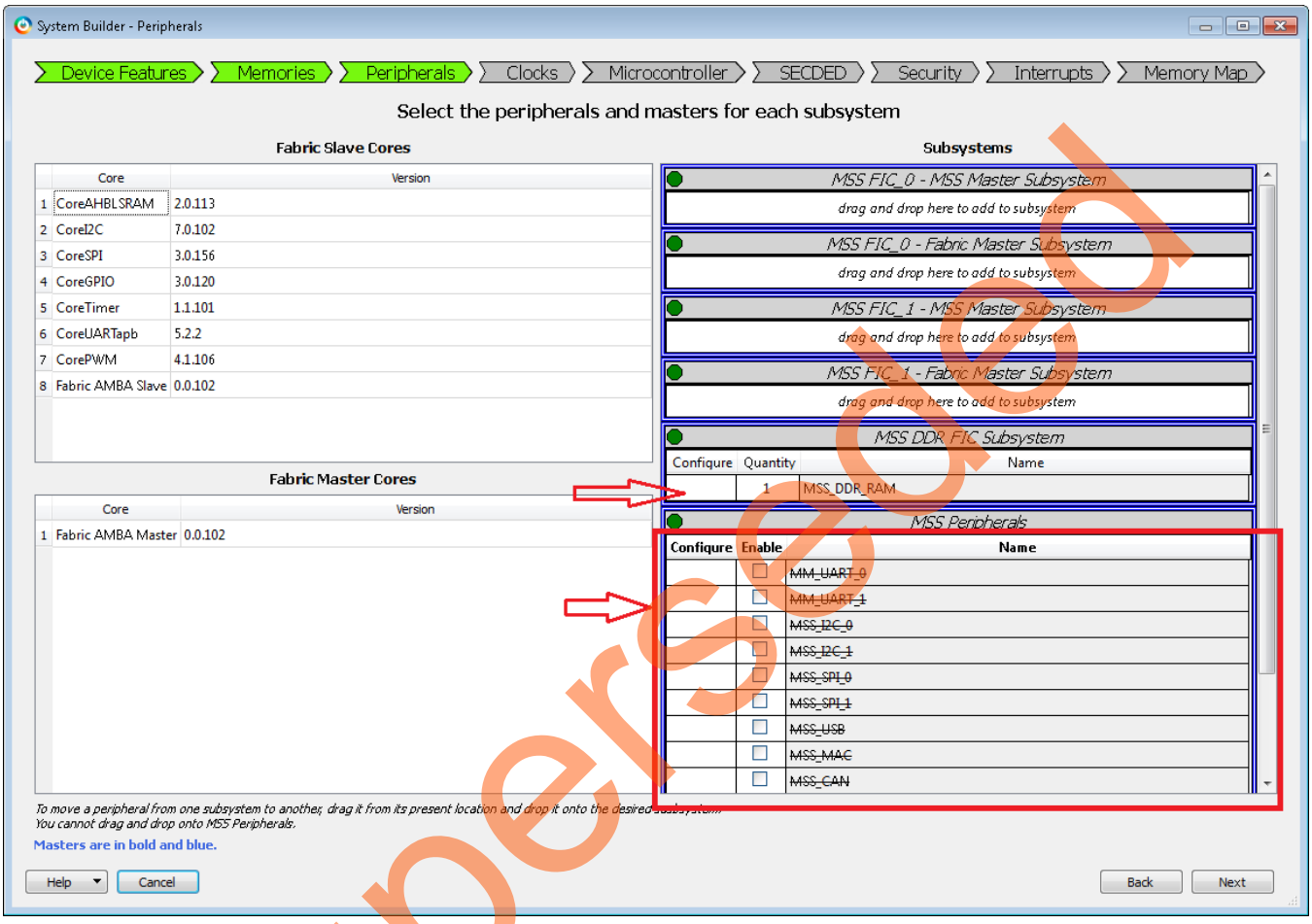


Figure 6. System Builder Configurator – Select Peripherals Page

10. Select **Next**, System Builder- **Clock Settings** page is displayed, as shown in Figure 7. 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 7 and the clock and the blocks that the clock is driving are displayed on the right-side panel.

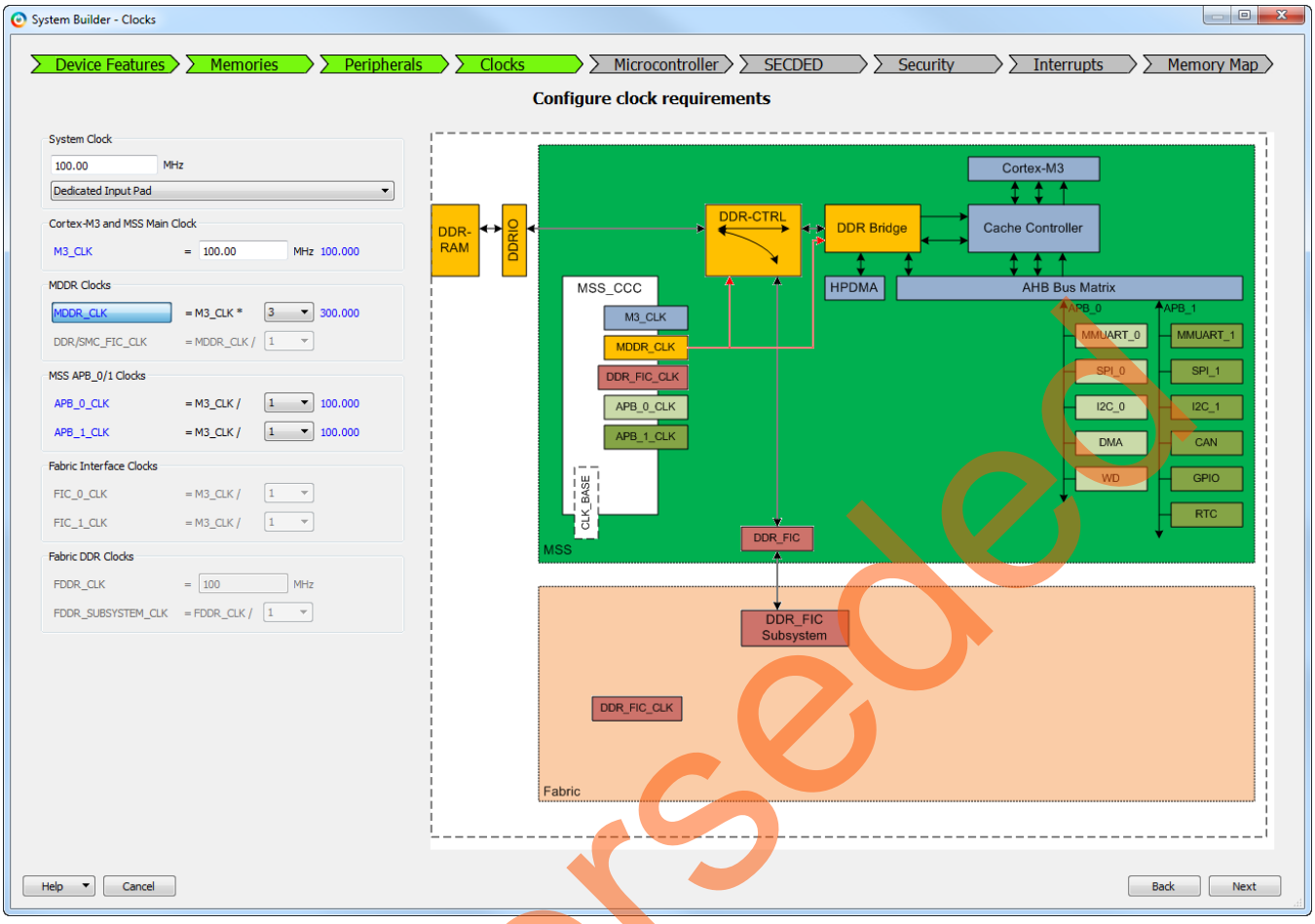


Figure 7. System Builder Configurator – Clock Settings Page

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

    The System Builder will generate the system based on the selected options.

    The System Builder block is created and added to Libero SoC project, as shown in Figure 8. The two soft cores CoreResetP and CoreConfigP will automatically be instantiated and connected by the System Builder. How these blocks are connected can be seen by opening the System Builder component in the SmartDesign canvas and this is explained in the later section of this tutorial. Refer to Opening the System Builder Component as SmartDesign section on 14.

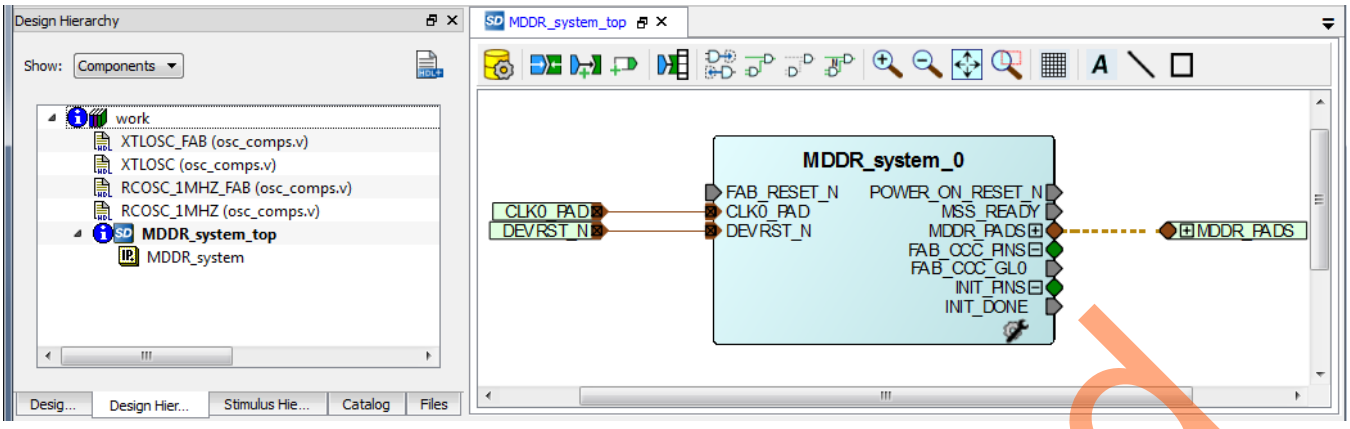


Figure 8. SmartFusion2 SoC FPGA System Builder Generated System

17. 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_GL0** as unused by right-clicking and selecting **Mark Unused**.
    - Mark the output port **INIT_DONE** as unused by right-clicking and selecting **Mark Unused**.
18. 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 9.

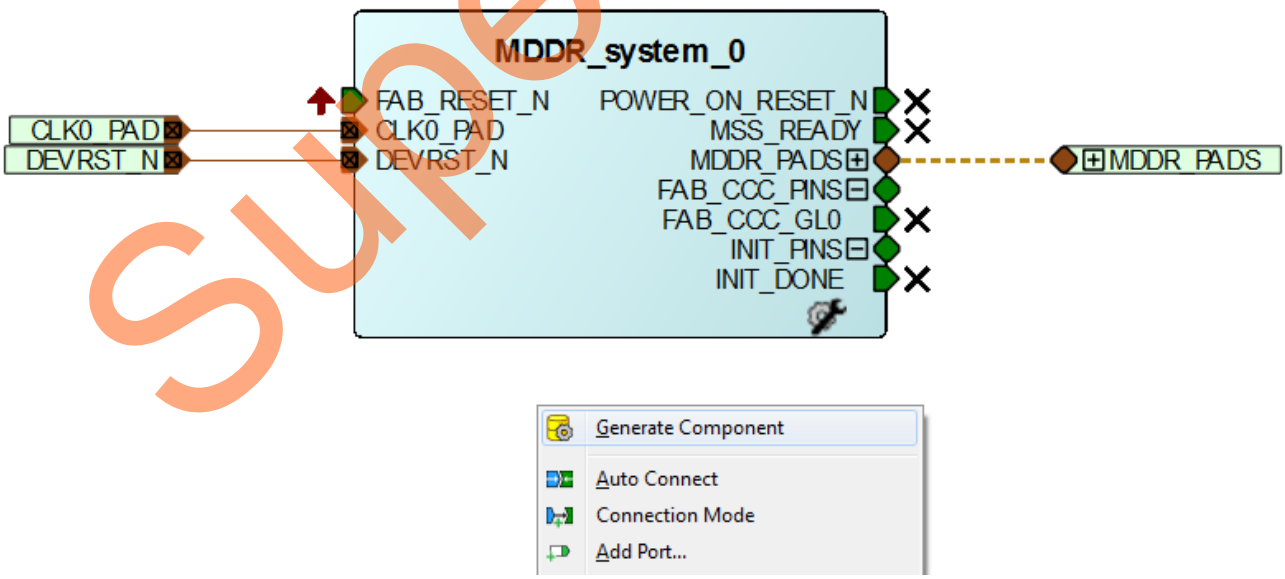


Figure 9. SmartFusion2 SoC FPGA Generated Final System

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 the 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 9. You can dive into that block to see the individual blocks that make-up the entire design. To do so, you can 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 10.

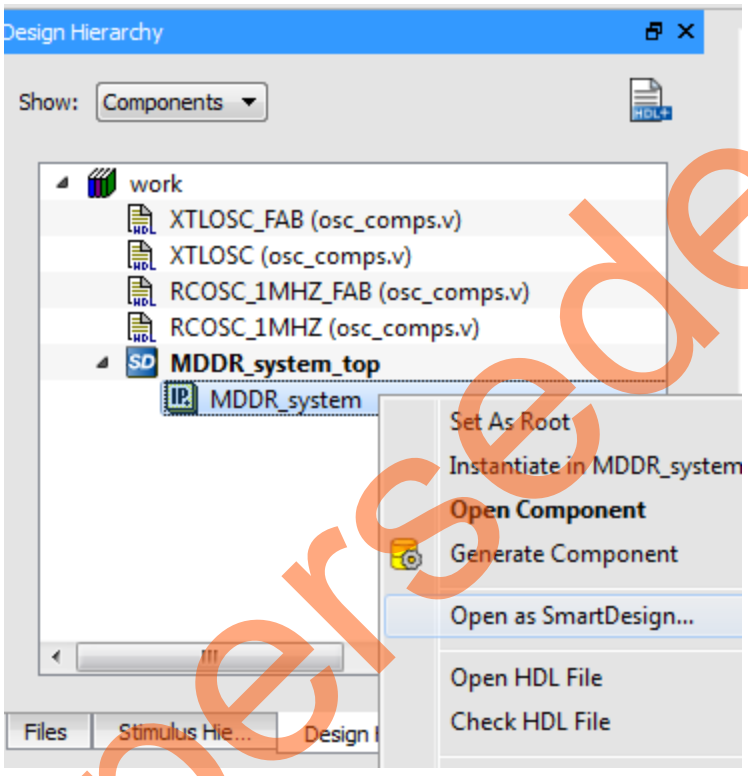


Figure 10. Open as SmartDesign Option

The system will be converted to a SmartDesign component and you will get the message, as shown in Figure 11.

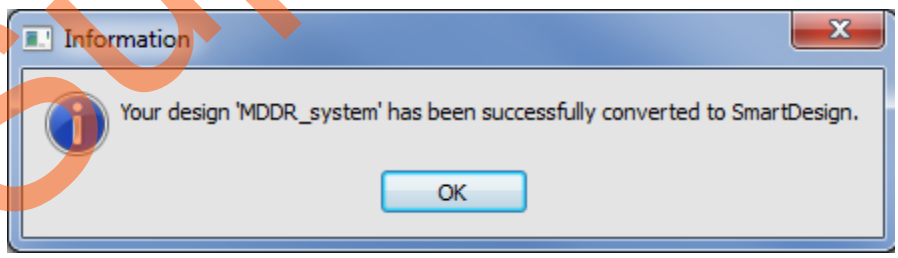


Figure 11. Successful Conversion of System Builder to a SmartDesign Message

3. Click **OK**. The entire system will be shown in the SmartDesign canvas, as shown in Figure 12.

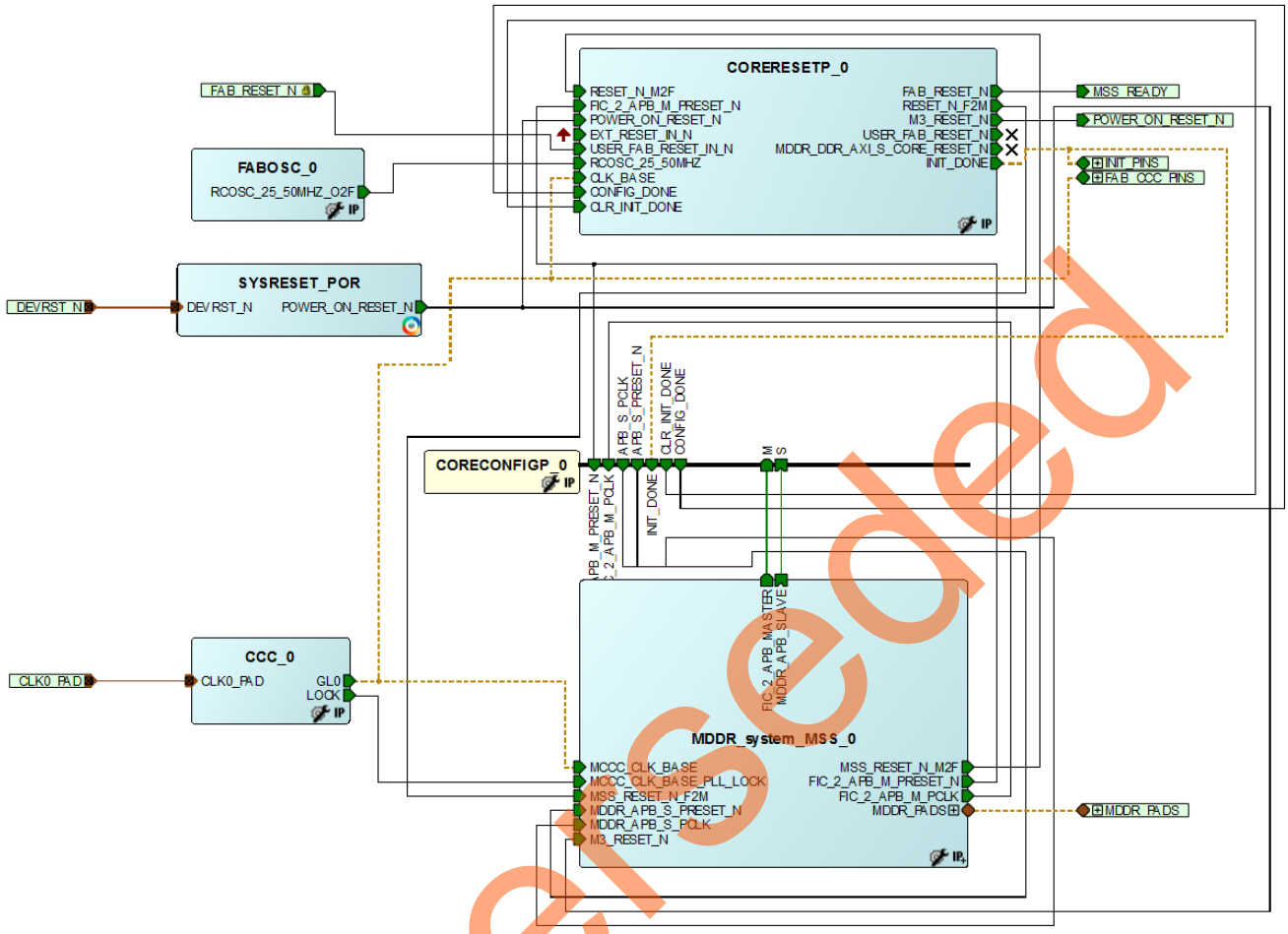


Figure 12. System Builder Generated System Opened in the SmartDesign

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

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

Step 2: Generating the Testbench

In this step you will 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 13. The IP catalog will display 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.

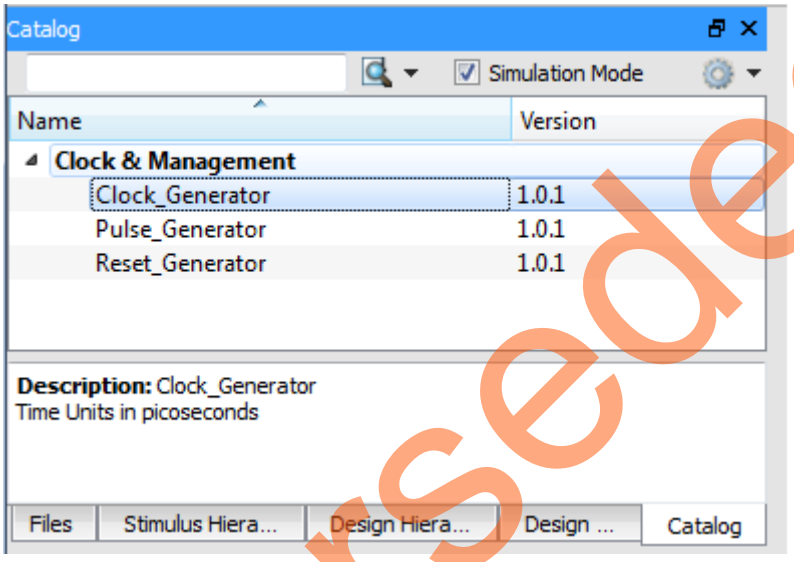


Figure 13. Simulation Cores in the Libero IP Catalog

2. Double-click the **Create SmartDesign Testbench** in the Libero Design Flow window, as shown in Figure 14.

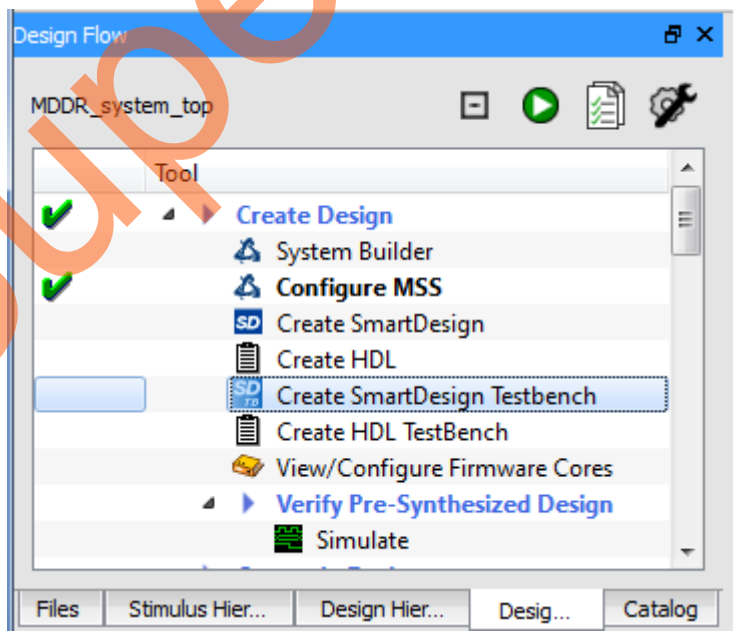


Figure 14. Opening SmartDesign Testbench

3. The **Create New SmartDesign Testbench** dialog box is displayed, as shown in Figure 15.

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

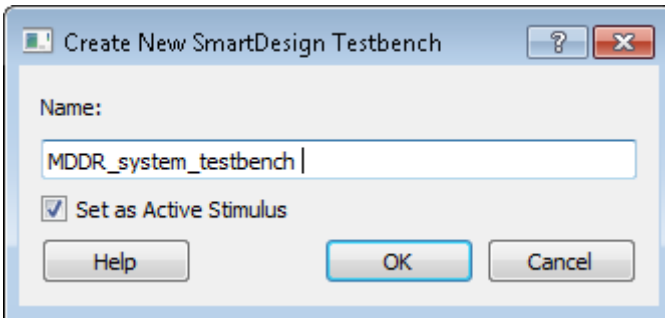


Figure 15. Create New SmartDesign Testbench Dialog Box

5. The SmartDesign canvas will open with the MDDR_system_top_0 component instantiated, as shown in Figure 16.

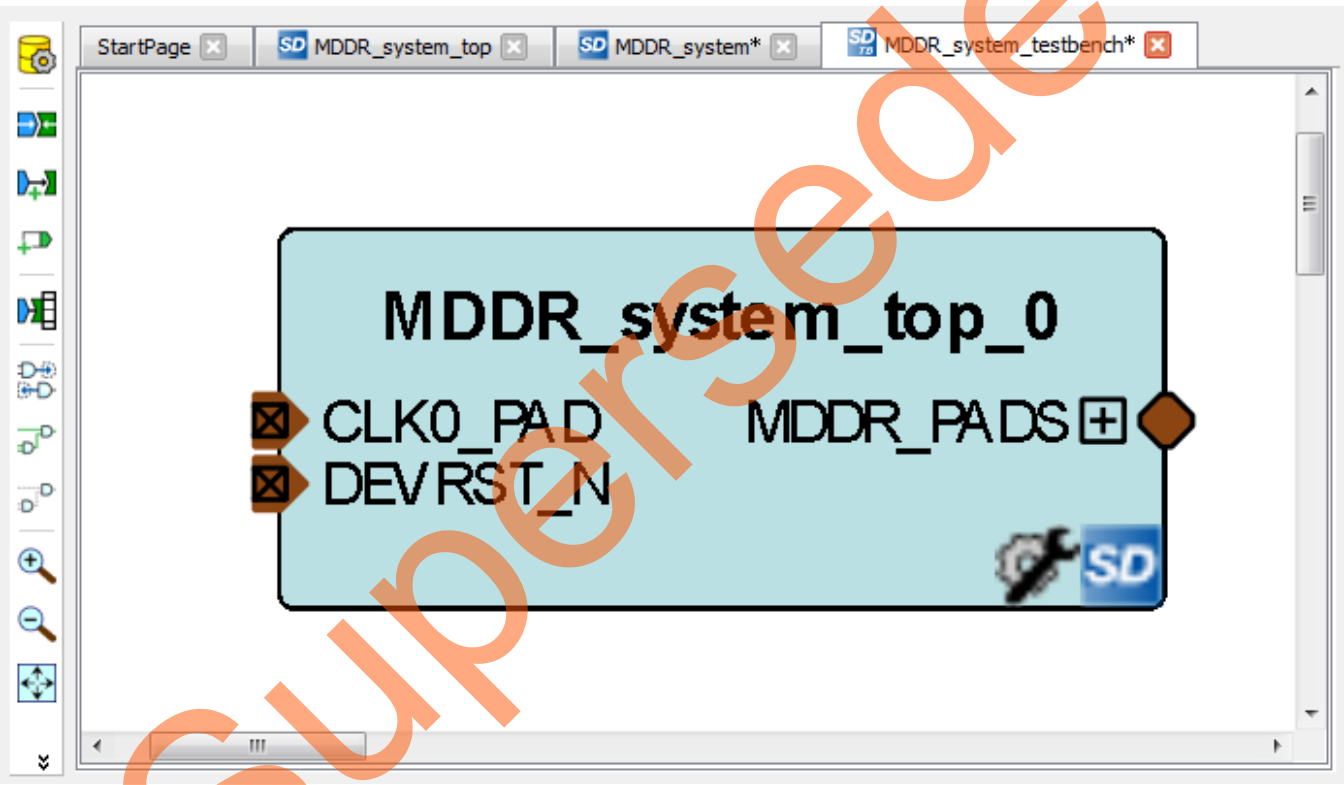


Figure 16. SmartDesign Testbench Canvas

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 17, is set in the RESET_GEN_0 configurator and click **OK**:
   - Level: ACTIVE LOW (default)
   - Programmable Delay (ns): 1000

The reset generator will provide the reset pulse for the simulation.

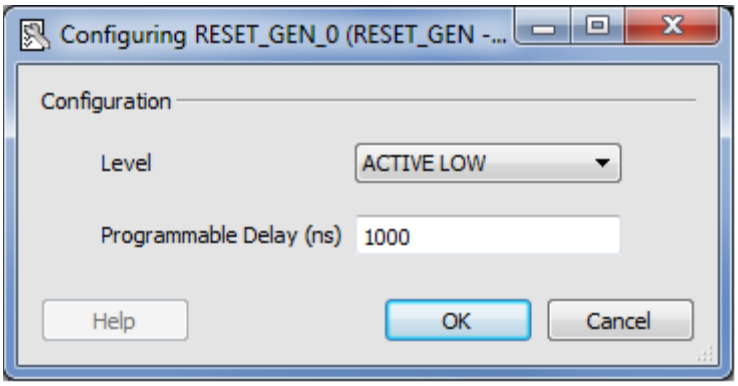


Figure 17. RESET_GEN Configuration

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 18, is set in the CLK_GEN_0 configurator and click **OK**:
   - Clock Period (ps): 10000
   - Duty Cycle (%): 50

Since you indicated that the System Clock is equal to 100 MHz, as shown in Figure 7, the clock generator period is set to 10000 ps to generate this 100 MHz clock.

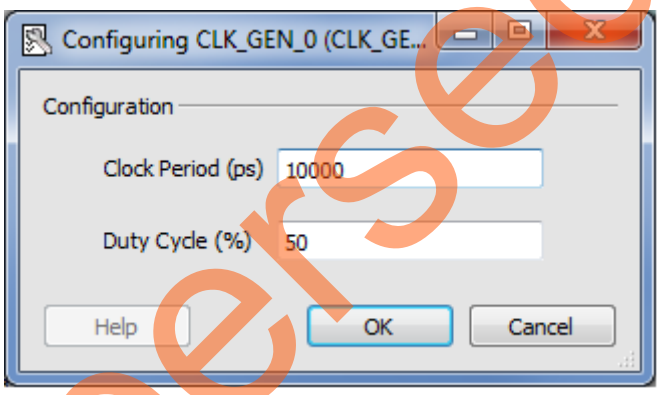


Figure 18. CLK_GEN Configuration

9. Import the provided DDR3 models into the Libero SoC project then 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 19.
   - Select the provided **ddr3.v** and the **ddr3_parameters.v** files and click **Open**, as shown in Figure 19. The files are located under the <project directory>\DDR3_SmartFusion2_Tutorial\Source_files \DDR3_folder.

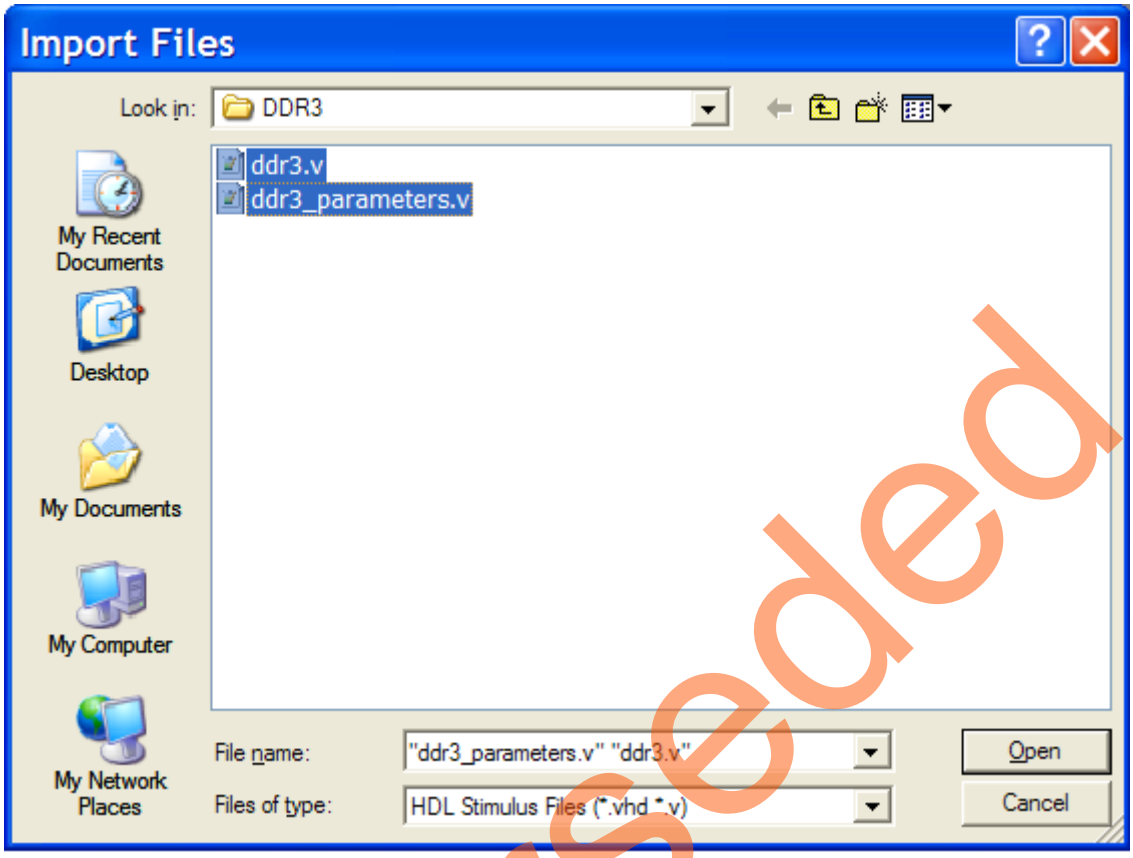


Figure 19. Import the DDR3 Models as Stimulus Files

Verify that the files are imported correctly as stimulus files by checking under the Stimulus folder in the Files tab, as shown in Figure 20.

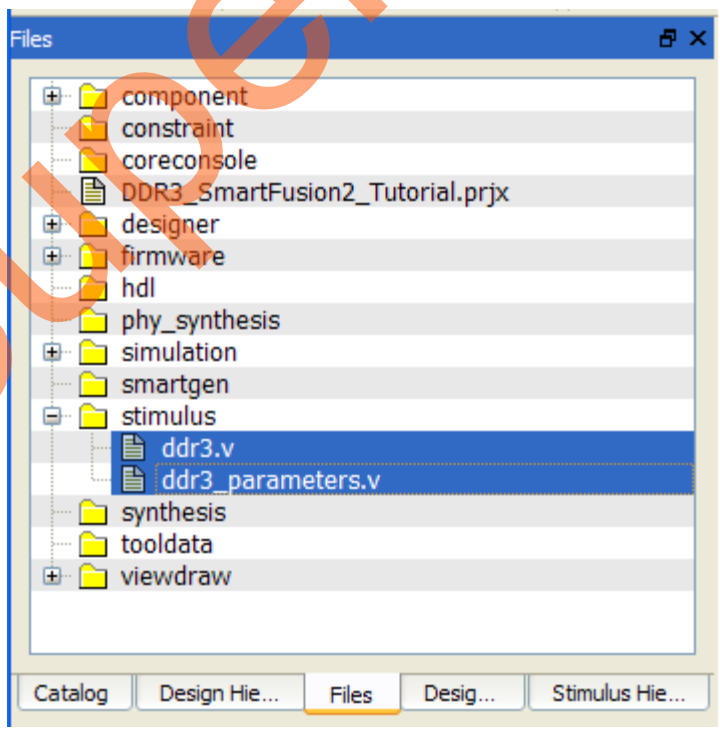


Figure 20. Imported DDR3 in Stimulus Folder

When the file is imported as a Stimulus, the file will also show in the **Stimulus Hierarchy** window, as shown Figure 21.

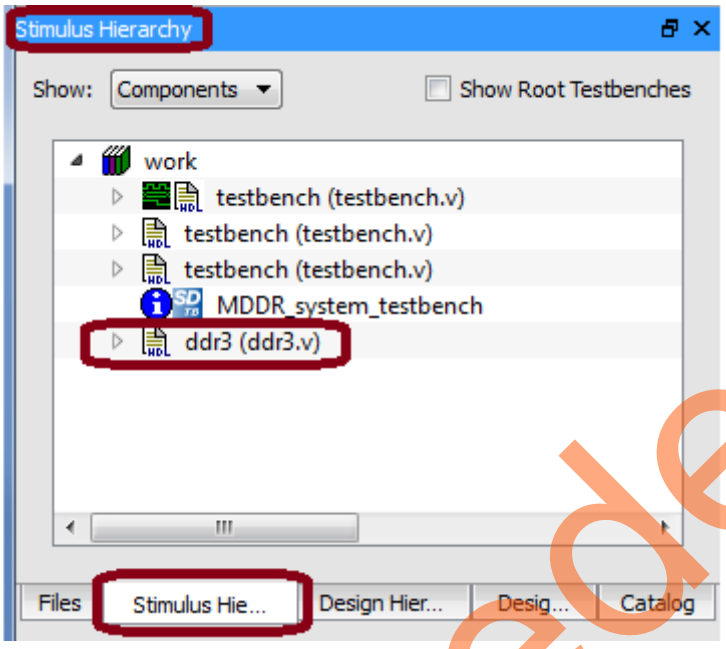


Figure 21. Stimulus Hierarchy Window

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 22.

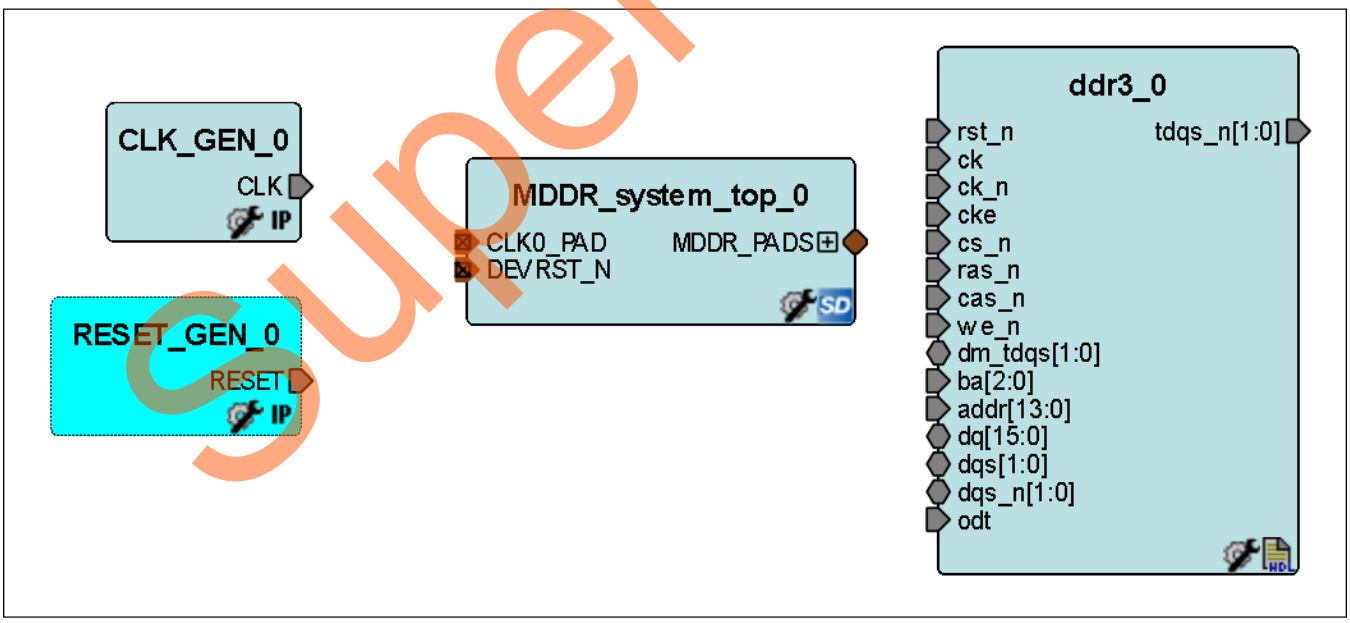


Figure 22. System Testbench Canvas with DDR3 Models Instantiated

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 23. The cursor will change 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.

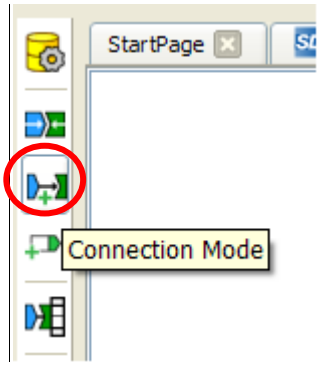


Figure 23. Enabling the Connection Mode Option

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_GN_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 1.

Table 1. 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. In order 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. In order 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 24. The dialog box is displayed, as shown in Figure 25.

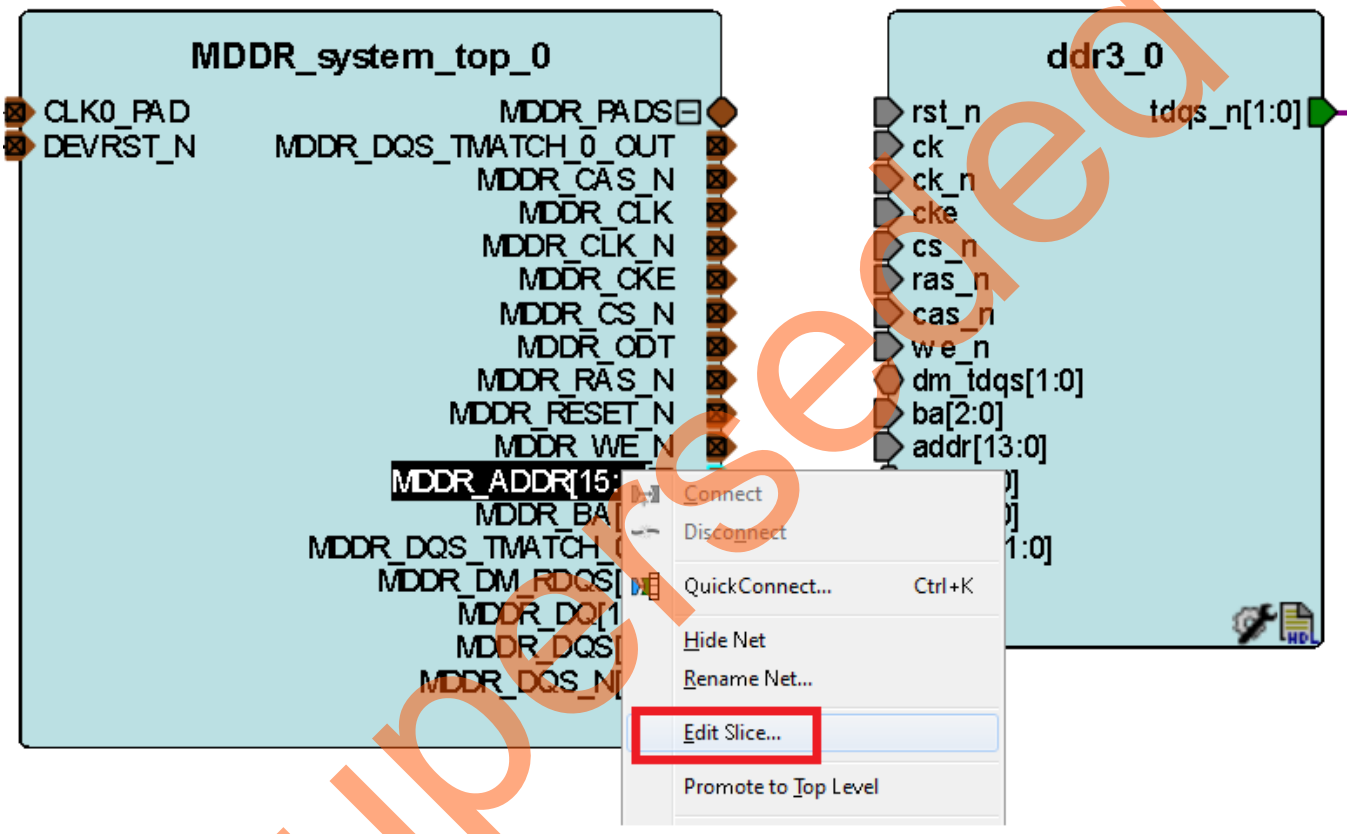


Figure 24. Creating a Bus Slice

- Click icon (circled in Figure 25) to add a slice. Since you need to add two slices, click icon twice. Then add the slices, as shown in Figure 25.

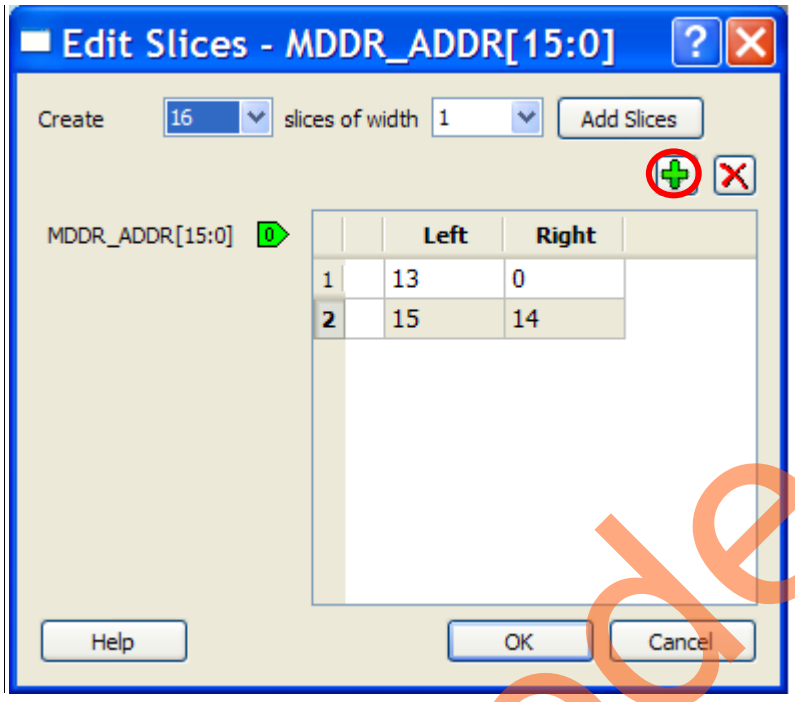


Figure 25. Edit Slices Dialog Box

- Click **OK**. This creates two slices of the MDDR_ADDR bus, as shown in Figure 26.

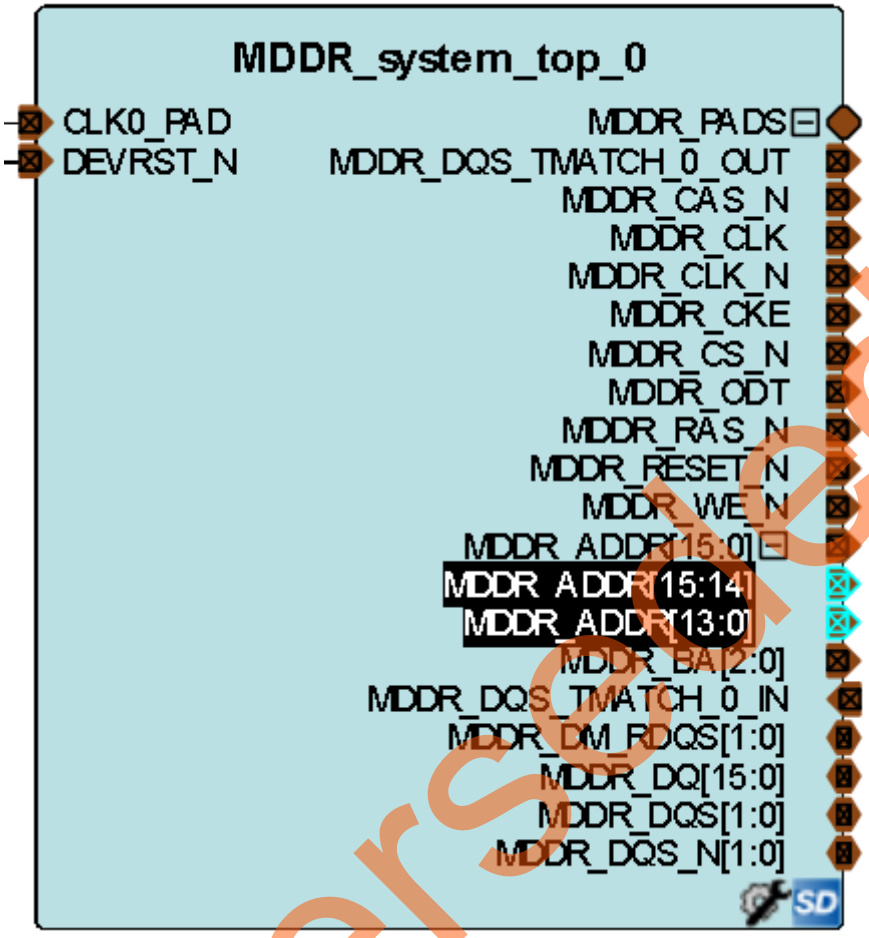


Figure 26. MDDR_ADDR Created Slices

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 27.

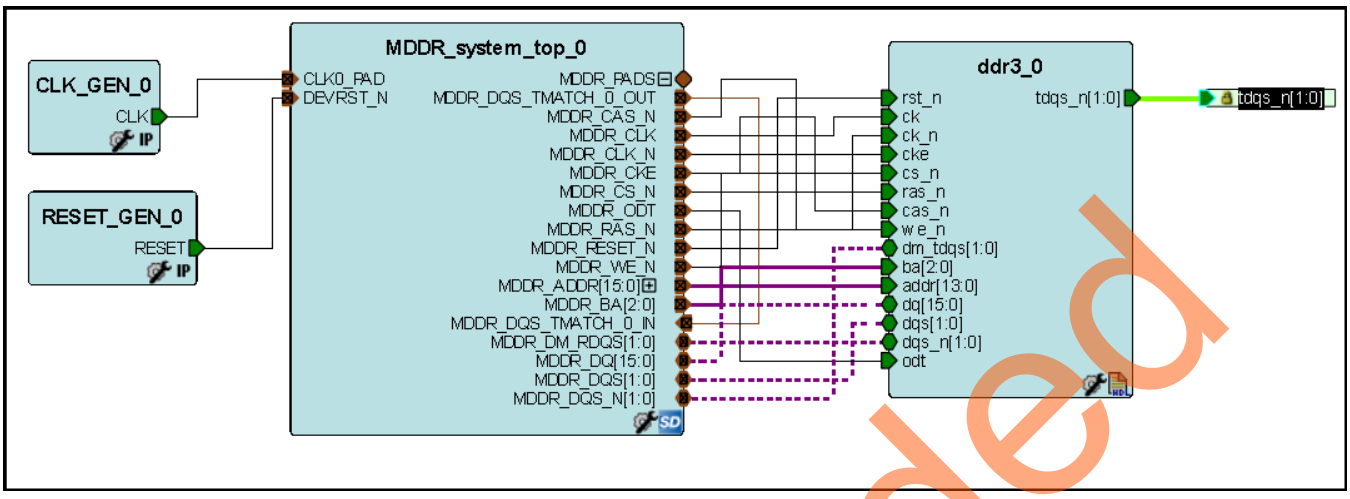


Figure 27. Fully Connected MDDR System

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 28.

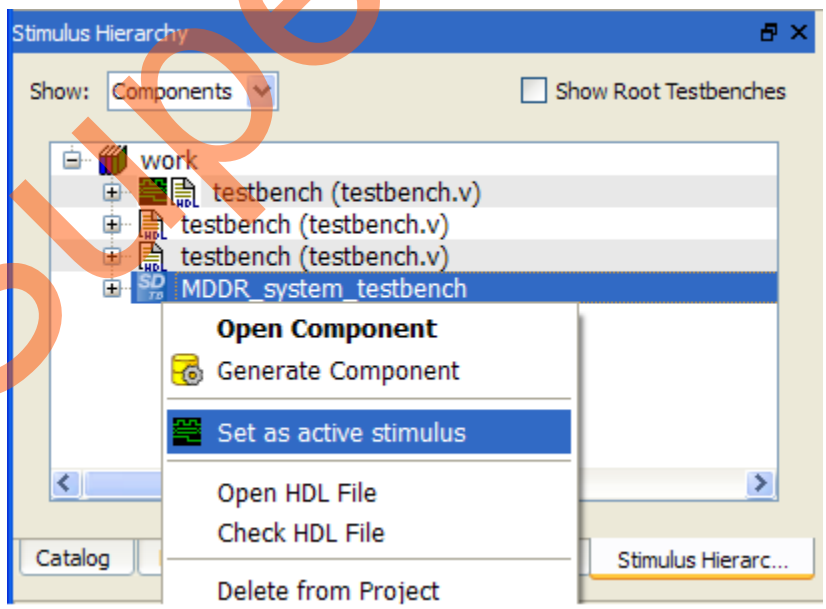


Figure 28. Setting a testbench as Active testbench

Step 3: Modifying the BFM Scripts

In this step you will modify the BFM script (user.bfm) file that was generated by the SmartDesign. The BFM script file simulates Cortex-M3 processor writing/reading to/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 will open, as shown in Figure 29.

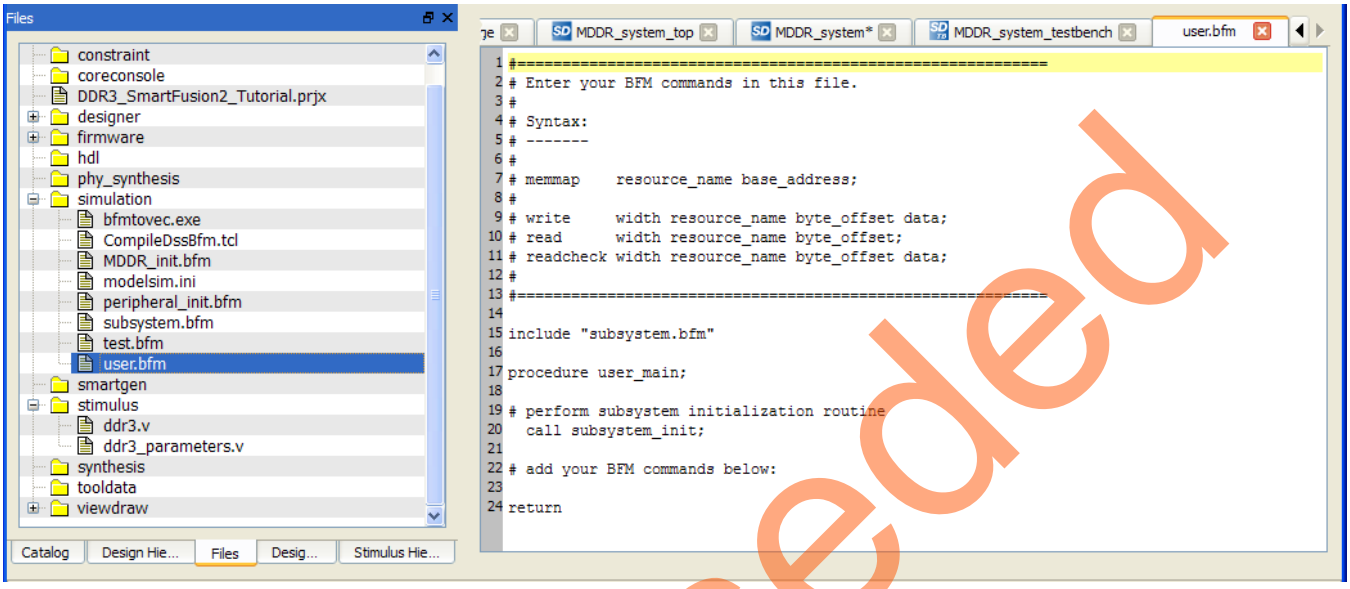


Figure 29. SmartDesign Generated user.bfm File

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 30.

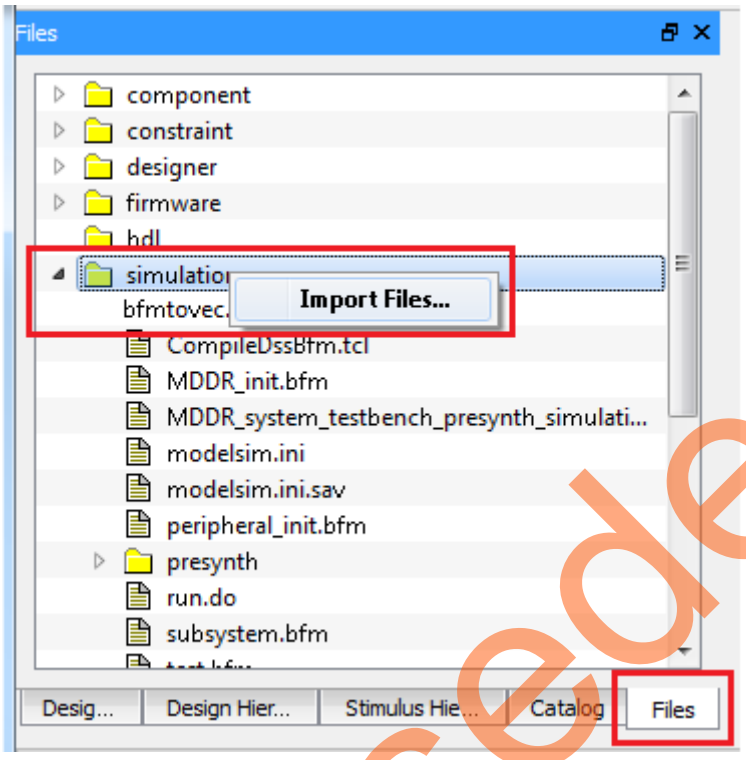


Figure 30. Importing bfm Source File

- Browse to <project directory>\DDR3_SmartFusion2_Tutorial\Source_files and select **user.bfm** and select **Open**.
- A warning as shown in Figure 31 will come up. Select **Yes**.

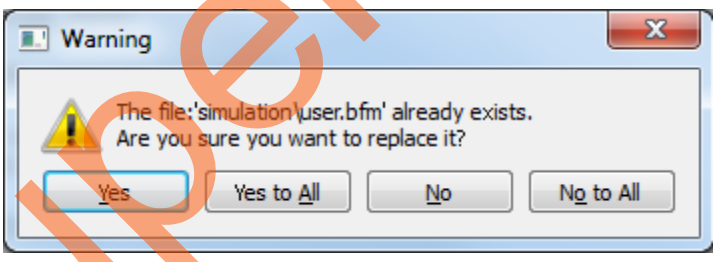


Figure 31. Replacing Existing user.bfm File

- If the user.bfm file is already open in your Libero window, a warning as shown in Figure 32 will come up. Select **Yes**. If the user.bfm is not already open in Libero window, the message will not show up.

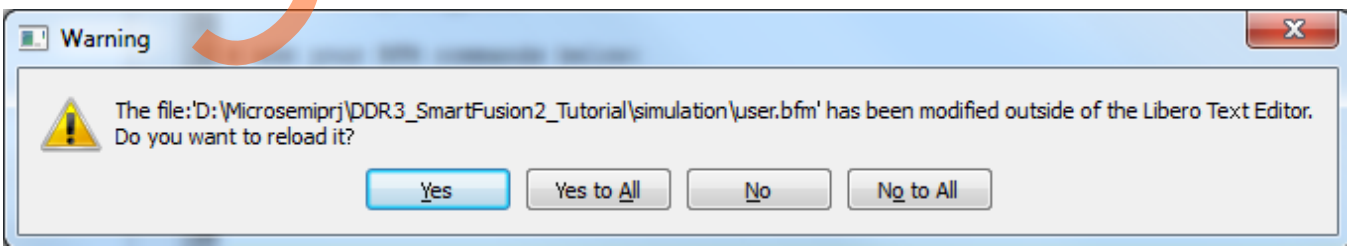
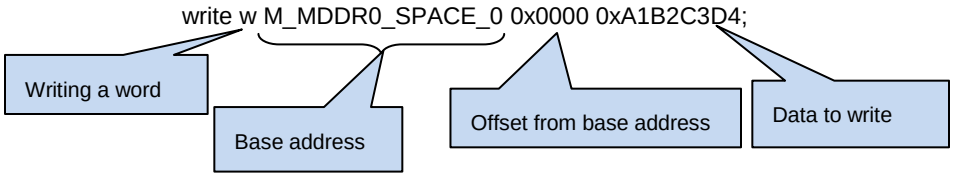


Figure 32. Reloading and Updating user.bfm File

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 33.

user.bfm

```
# add your BFM commands below:


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


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


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


print "TEST ENDS";

return
```

Figure 33. 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

1. In this step you will use the SmartDesign testbench and BFM script file to simulate the design. Open the Libero SoC project settings (**Project > Project Settings**).
2. Select **Do File** under Simulation Options in the Project Settings window. Change the **Simulation runtime** to 260us, as shown in Figure 34.

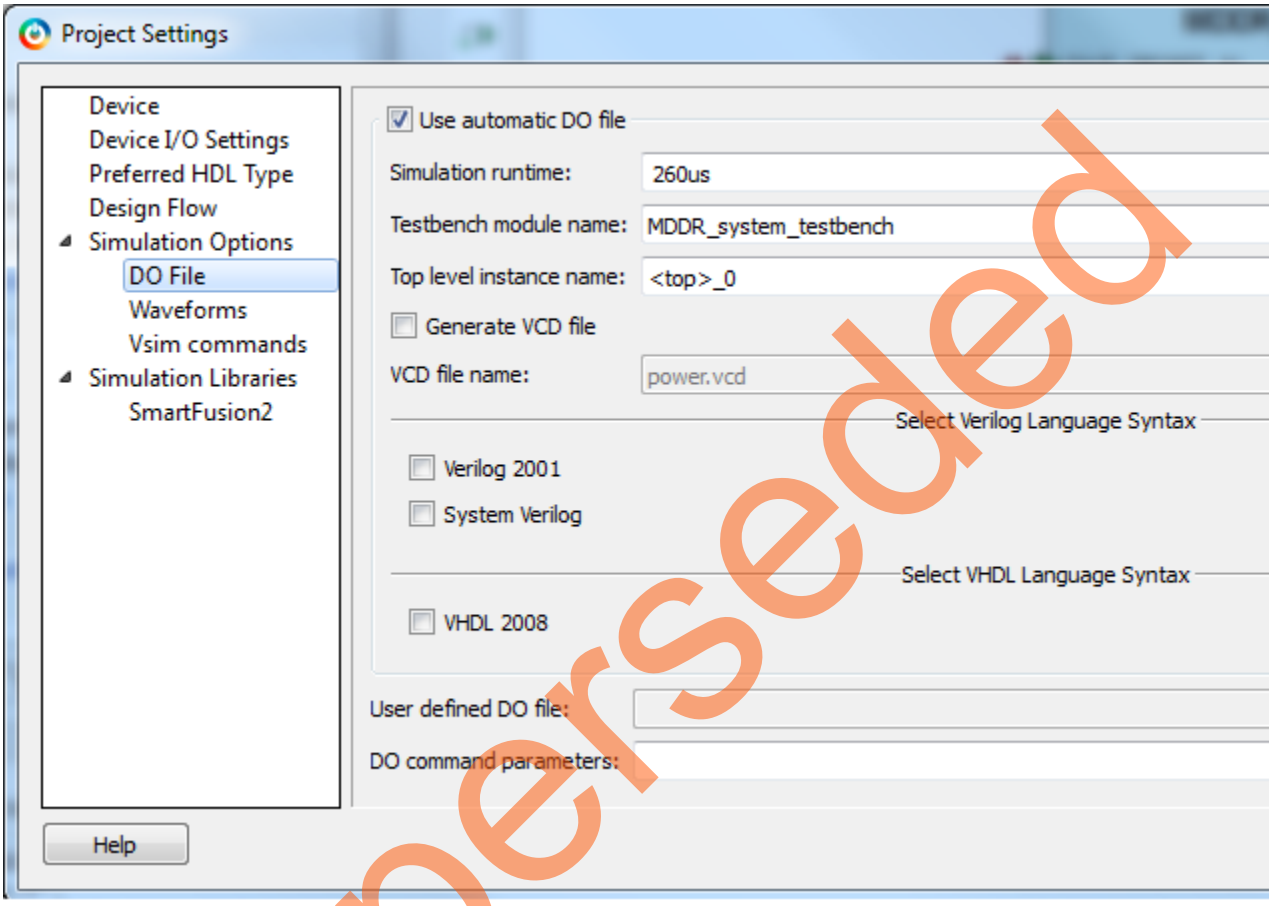


Figure 34. Project Setting – Do File Simulation Runtime Setting

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 35. In this file the list of signals that are required are 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.

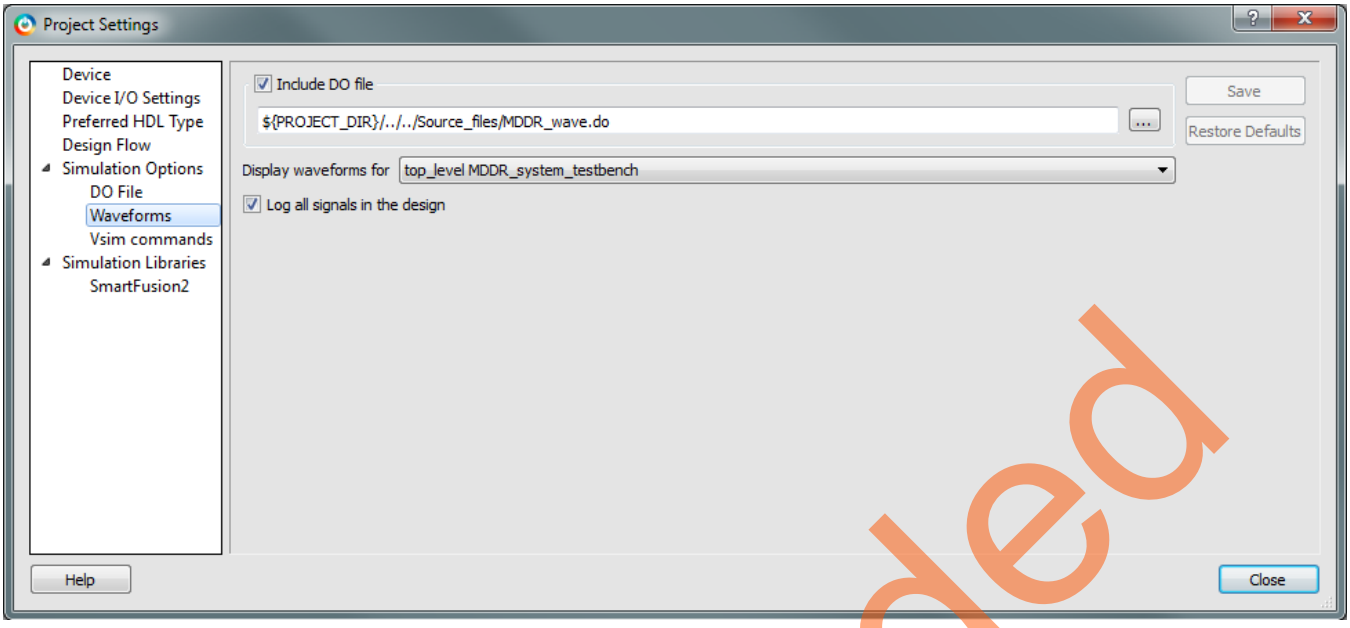


Figure 35. Project Setting – Specifying the MDDR_wave.do File Location

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.

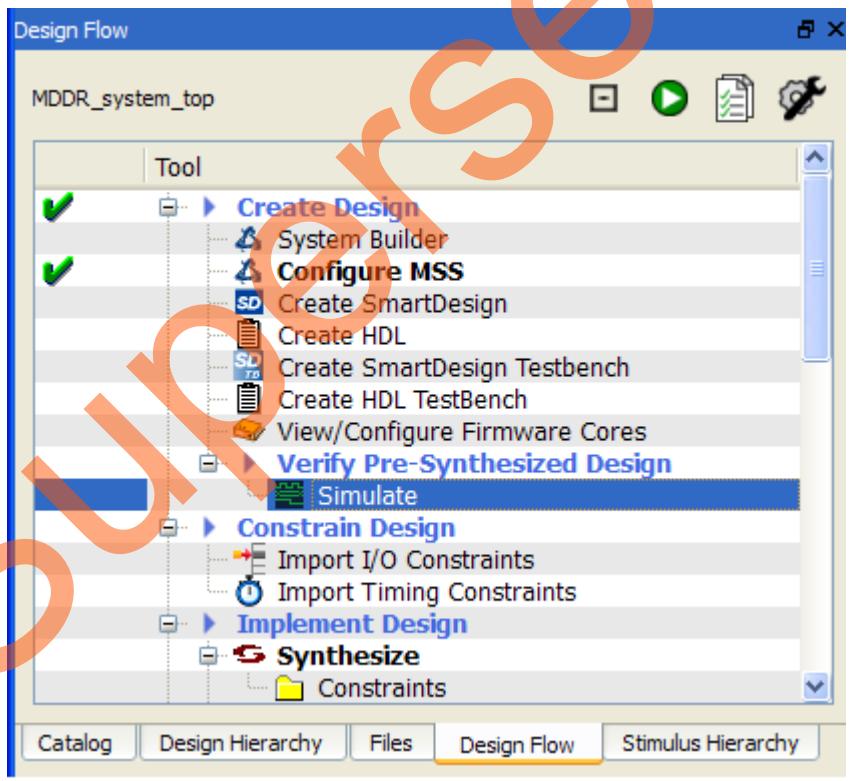


Figure 36. Starting Pre-Synthesis Simulation

Step 5: Validating the Simulation Results

1. ModelSim runs the design for about 260 us, as specified in the Project Settings window.

   This will go through the initial process of initializing 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 5. You can write and read from the external DDR.

2. The ModelSim transcript window will display 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 whether the data read matches with the value provided along with the readcheck command. If the value read does not match, the simulation will show an error.

Figure 37. 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 38. 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

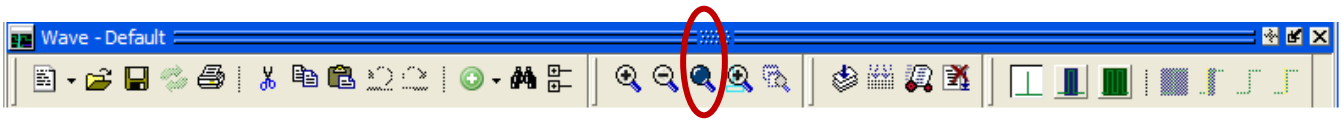


Figure 39. Zoom Full Option

4. Place the cursor around 243 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 and 0.

Figure 40. Zoom In on Active Cursor Button

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

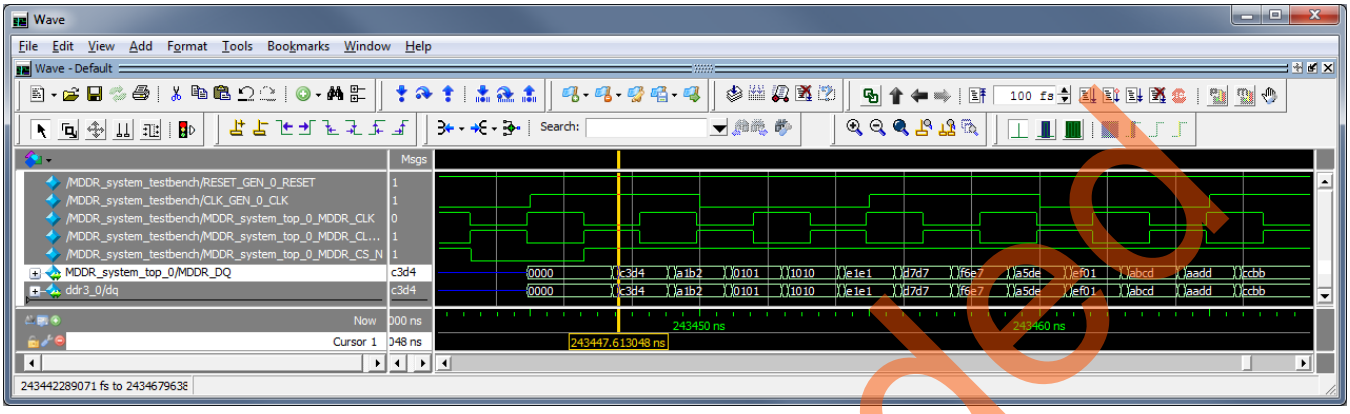


Figure 41. Write/Read Data

That concludes this tutorial.

6. 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 finally connected the different blocks using the SmartDesign tools.

Finally you verified the design in ModelSim using AMBA BFM simulation.

Appendix: VHDL Flow

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

1. Copy the precompiled VHDL simulation library folder **smartfusion2** from the Libero SoC install area (<Libero install> \Designer\lib\modelsim\precompiled\vhdl\) to a different folder on your disk (for example, E:\Microsemi_prj\)
2. Remove the **Read-Only** attribute from the **smartfusion2** folder at the new location.

   Note: The reason for steps 1 and 2 is that ModelSim full version needs to refresh the precompiled library. Those steps are to enable ModelSim full version to refresh the precompiled library and to ensure that the original precompiled library, which is installed with the Libero SoC, is unchanged.
3. Simulate with automatic design optimization option disabled (-novopt) and point to the new precompiled library location (for example, E:\Microsemi_prj\smartfusion2) in the **Project Settings** window as shown below:
   (a) Select **Project Settings** from the **Project** menu
   (b) Select **Vsim commands** under **Simulation Options**. Add the –novopt option into the **Additional options** field, as shown in Figure 36. The –novopt option disables the automatic design optimization run.

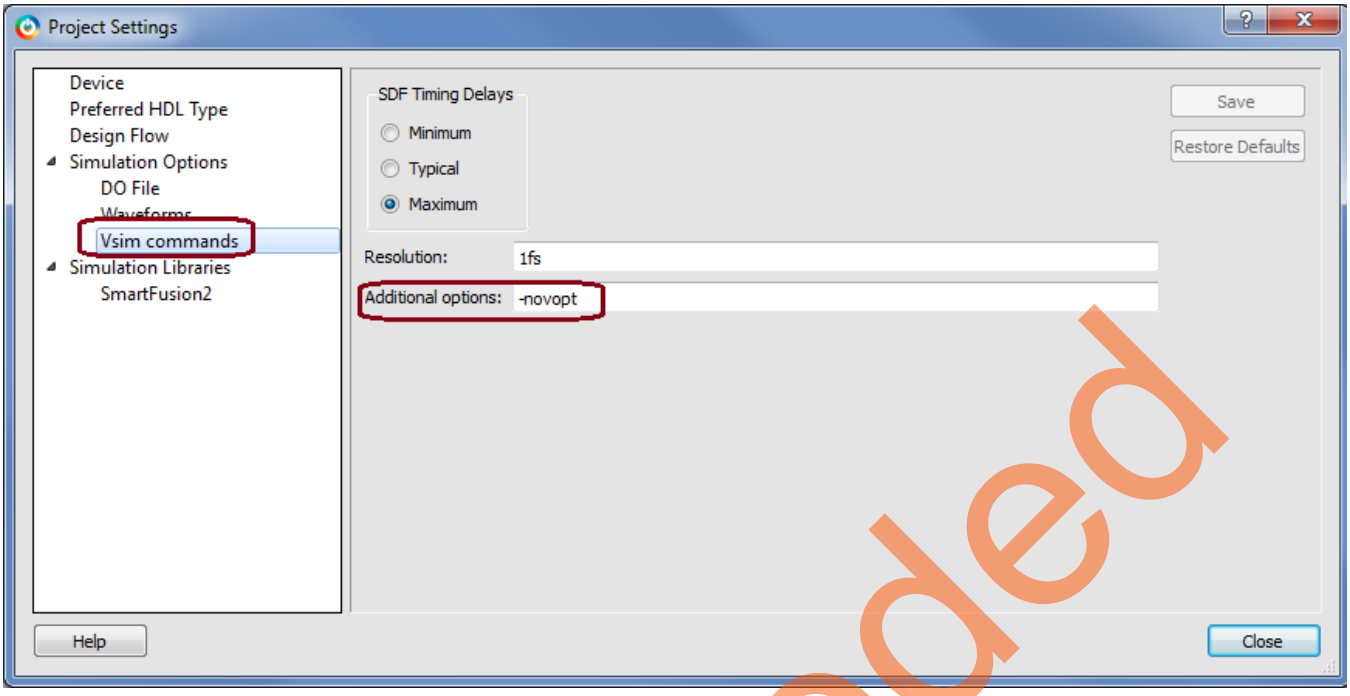


Figure 42. Project Settings – Specifying **–novopt** Simulation Option

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

I. Clear **Use default library path** option

II. In the **Library path**, enter the new location where you copied the precompiled library (for example, E:/Microsemi_prj/smartfusion2), as shown in Figure 37.

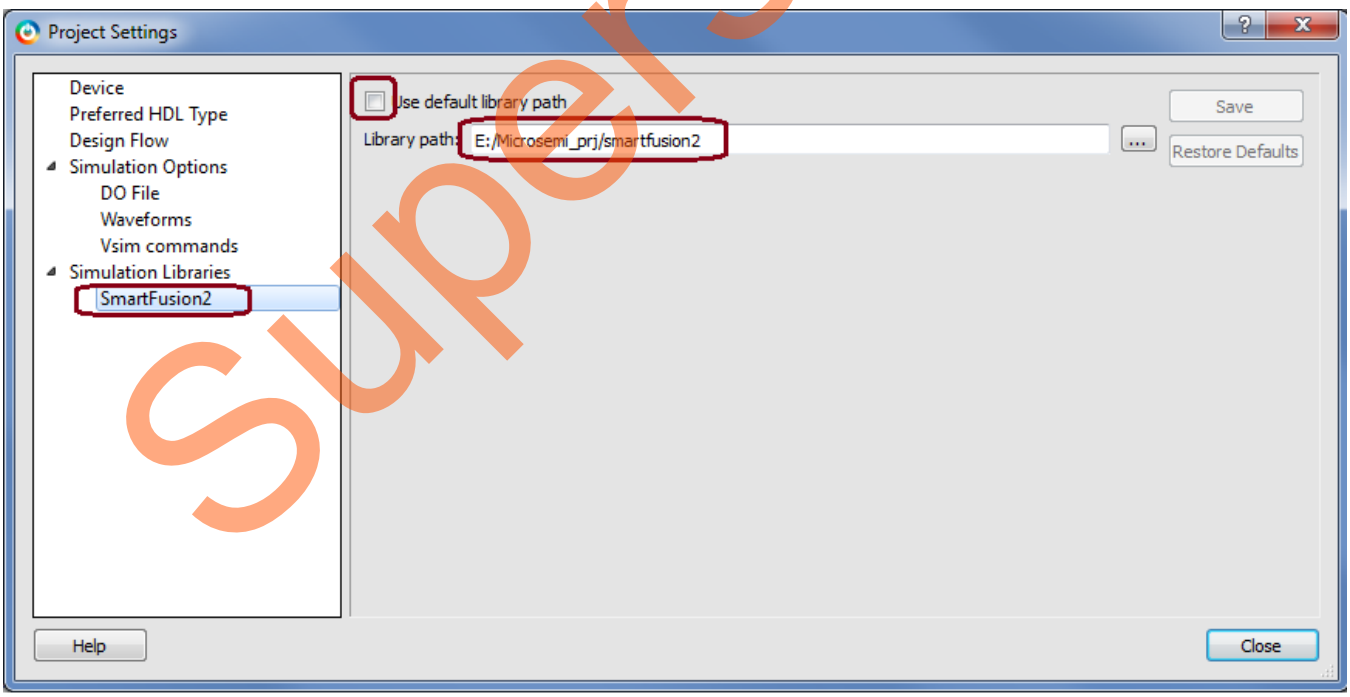


Figure 43. Project Settings – Specifying Precompiled Library Path

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

List of Changes

Revision	Changes	Page
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

Visit the Microsemi SoC Products Group Customer Support website for more information and support (http://www.microsemi.com/soc/support/search/default.aspx). Many answers available on the searchable web resource include diagrams, illustrations, and links to other resources on website.

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.

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) or contact a local sales office. Sales office listings can be found at 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. Alternatively, within My Cases, select **Yes** in the ITAR drop-down list. For a complete list of ITAR-regulated Microsemi FPGAs, visit the ITAR web page.

Microsemi Corporate Headquarters
One Enterprise, Aliso Viejo CA 92656 USA
Within the USA: +1 (800) 713-4113
Outside the USA: +1 (949) 380-6100
Sales: +1 (949) 380-6136
Fax: +1 (949) 215-4996
E-mail: sales.support@microsemi.com

Microsemi Corporation (Nasdaq: MSCC) offers a comprehensive portfolio of semiconductor and system solutions for communications, defense and security, aerospace, and industrial markets. Products include high-performance and radiation-hardened analog mixed-signal integrated circuits, FPGAs, SoCs, and ASICs; power management products; timing and synchronization devices and precise time solutions, setting the world's standard for time; voice processing devices; RF solutions; discrete components; security technologies and scalable anti-tamper products; Power-over-Ethernet ICs and midspans; as well as custom design capabilities and services. Microsemi is headquartered in Aliso Viejo, Calif. and has approximately 3,400 employees globally. Learn more at **www.microsemi.com**.

50200372-7/03.14