SmartFusion2 - Accessing External SDRAM through Fabric - Libero SoC v11.7

TU0311 Tutorial

Microsemi
Power Matters.™

Microsemi
Power Matters.™

Contents

Figures

Tables

1 Preface

1.1 Purpose

This tutorial describes how to create a hardware design for accessing an external single data rate (SDR) synchronous dynamic random access memory (SDRAM) and functionally verify the design using simulation.

1.2 Intended Audience

This tutorial is intended for:

- FPGA designers
- System-level designers

1.3 References

See the following web page for a complete and up-to-date listing of SmartFusion2 device documentation: *http://www.microsemi.com/products/fpga-soc/soc-fpga/sf2docs*

- *CoreAMBA BFM User's Guide*

2 Accessing External SDRAM through Fabric - Libero SoC v11.7

2.1 Introduction

This tutorial describes how to create a hardware design for accessing an external SDR SDRAM and functionally verify the design using simulation. A CoreSDR_AXI intellectual property (IP) is used in SmartFusion®2 system-on-chip (SoC) field programmable gate array (FPGA) device for interfacing the external SDR SDRAM memory with the ARM® Cortex®-M3 processor.

The CoreSDR_AXI IP has a 64-bit AXI bus interface for communicating with the Cortex-M3 processor. The CoreSDR_AXI IP generates the inputs for the SDR SDRAM memory and handles the timing parameters for the input signals of the SDR SDRAM memory.

The tutorial describes the following:

1. Creating a Libero® System-on-Chip (SoC) project using SmartFusion2 SoC FPGA
2. Updating the IP catalog by downloading the latest versions of the IP cores
3. Configuring the various hardware blocks using SmartDesign
4. Configuring the MDDR and CCC blocks of the microcontroller subsystem (MSS) component
5. Generating the microcontroller subsystem (MSS) component
6. Integrating the various hardware blocks in SmartDesign and generating the final top-level component
7. Performing functional level verification of the design using the advanced microcontroller bus architecture (AMBA) advanced extensible interface (AXI) bus functional model (BFM) simulation in Mentor Graphics ModelSim® simulator
8. Using the ModelSim GUI to see the various design signals in the Waveform window of ModelSim

2.2 Design Requirements

Table 1 **• Design Requirements**

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

2.2.1 Project Files

The project files associated with this tutorial can be downloaded from the Microsemi website: *http://soc.microsemi.com/download/rsc/?f=m2s_tu0311_liberov11p7_df*

The project files associated with this tutorial include the following:

- Source
- Solution
- Readme file, which describes the complete directory structure

2.3 Design Overview

The design demonstrates the read/write access to an external slave SDR SDRAM 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. A soft SDRAM controller, CoreSDR_AXI, is implemented inside the FPGA fabric of the SmartFusion2 SoC FPGA. It provides the interface between the Cortex-M3 processor master and slave SDRAM memory. The CoreSDR_AXI IP has a 64-bit AMBA AXI interface on one side, which communicates with the Cortex-M3 processor through the AXI interface. The other side of the CoreSDR_AXI IP has the SDRAM memory interface signals, which are fed as input to the external SDRAM memory through the FPGA I/Os of the SmartFusion2 SoC FPGA. The CoreSDR_AXI IP converts the AXI transactions into the SDRAM memory read/write transactions with appropriate timing generation. It also handles the appropriate command generation for write/read/refresh/precharge operations required for SDRAM memory.

The Cortex-M3 processor resides inside the microcontroller subsystem (MSS) block of the SmartFusion2 SoC FPGA. The MSS contains another block called the double data rate (DDR) Bridge. This block manages the read/write requests from the various masters to the DDR controller in the MSS, called the MDDR block, or interfaces with external bulk memories such as SDR SDRAM through the fabric. This fabric interface for the external bulk memories is called the SMC_FIC.

Either the MDDR controller or SMC_FIC can be enabled at a given time. The MDDR controller is disabled when the SMC_FIC path is active. The fabric side of the SMC_FIC can be configured for one or two 32-bit AHB-Lite interfaces, or an AXI64 interface. The enabling of the SMC_FIC path and its interface towards the fabric side of the SMC_FIC can be configured through the MSS configurator.

In this design, the MDDR block is configured to bring out the 64-bit AXI interface to the fabric through the SMC_FIC.

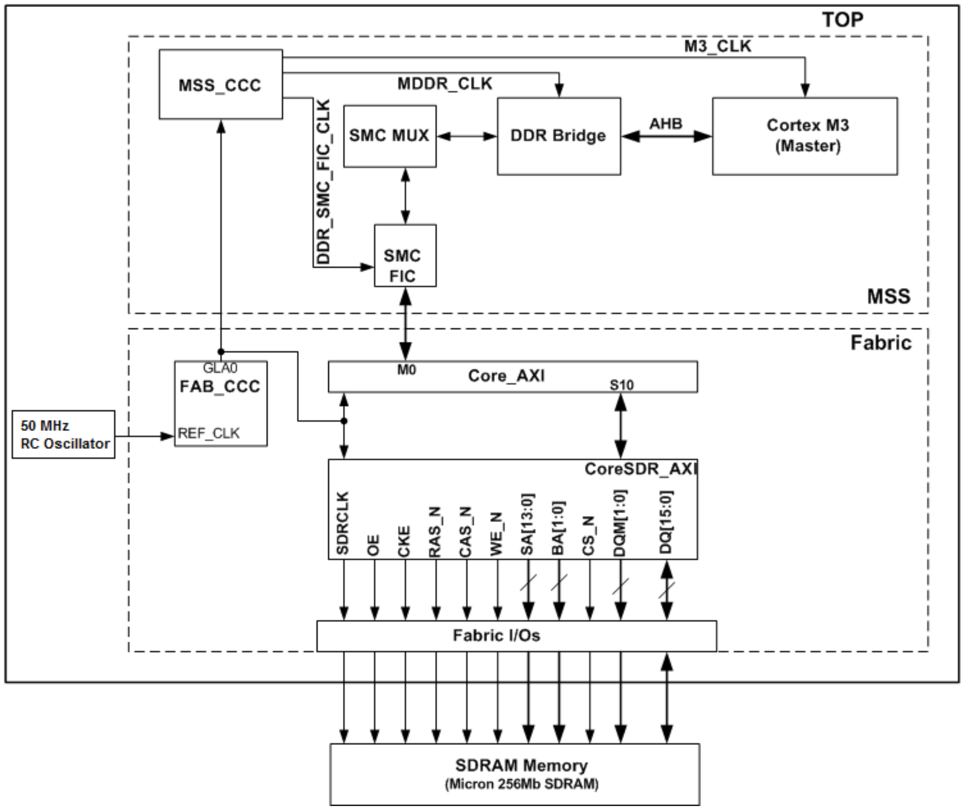
In the SmartFusion2 SoC FPGA device, there are six clock conditioning circuits (CCCs) inside the fabric and one CCC block inside the MSS. Each CCC block has an associated phase-locked loop (PLL). The CCC blocks and their PLLs provide several clock conditioning capabilities such as clock frequency multiplication, clock division, phase shifting, and clock-to-output or clock-to-input delay canceling. The 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 to generate the clocks for the various elements inside the MSS and fabric.

In the SmartFusion2 SoC FPGA device, there are three oscillator sources-an on-chip 25 MHz - 50 MHz RC oscillator, on-chip 1 MHz RC oscillator, and external main crystal oscillator.

In this design, the 25 MHz - 50 MHz on-chip oscillator is configured to provide the clock input for the fabric CCC block, which in turn drives the clocks to all the design blocks, including the MSS block.

Figure 1 shows the top-level design.

Figure 1 • **Top-Level Design**

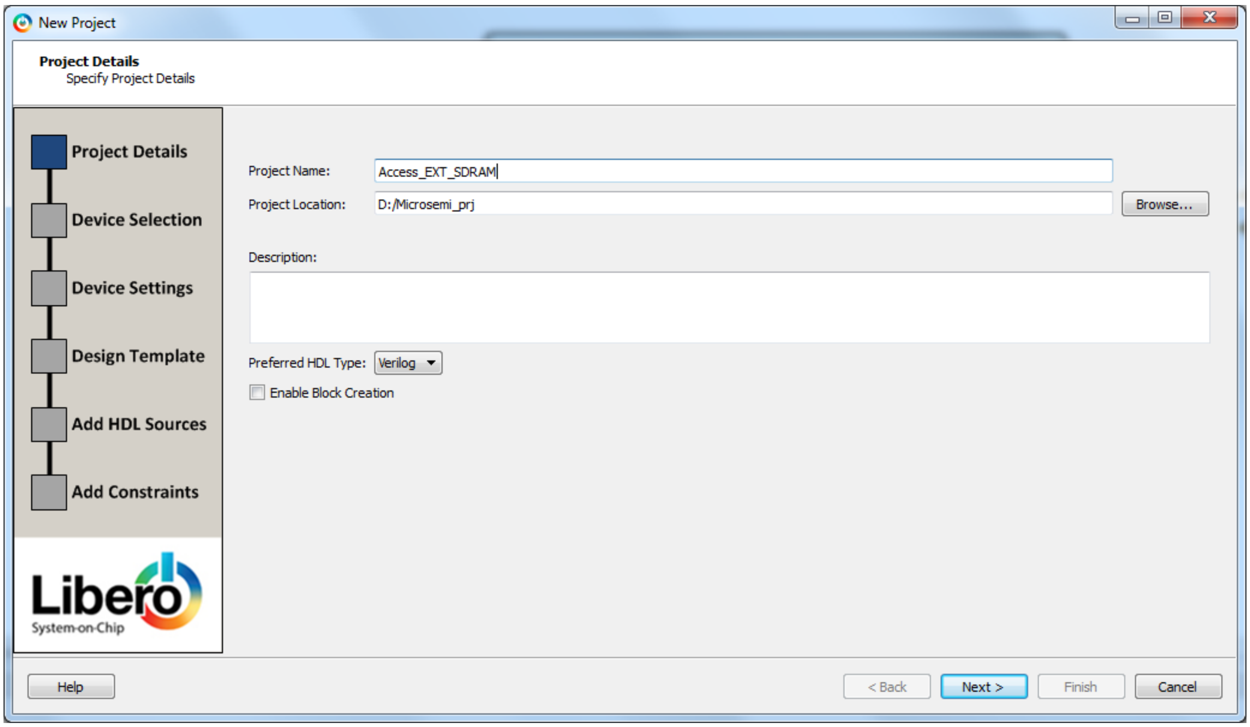


2.4 Design Creation

2.4.1 Step 1: Creating a Libero SoC Project

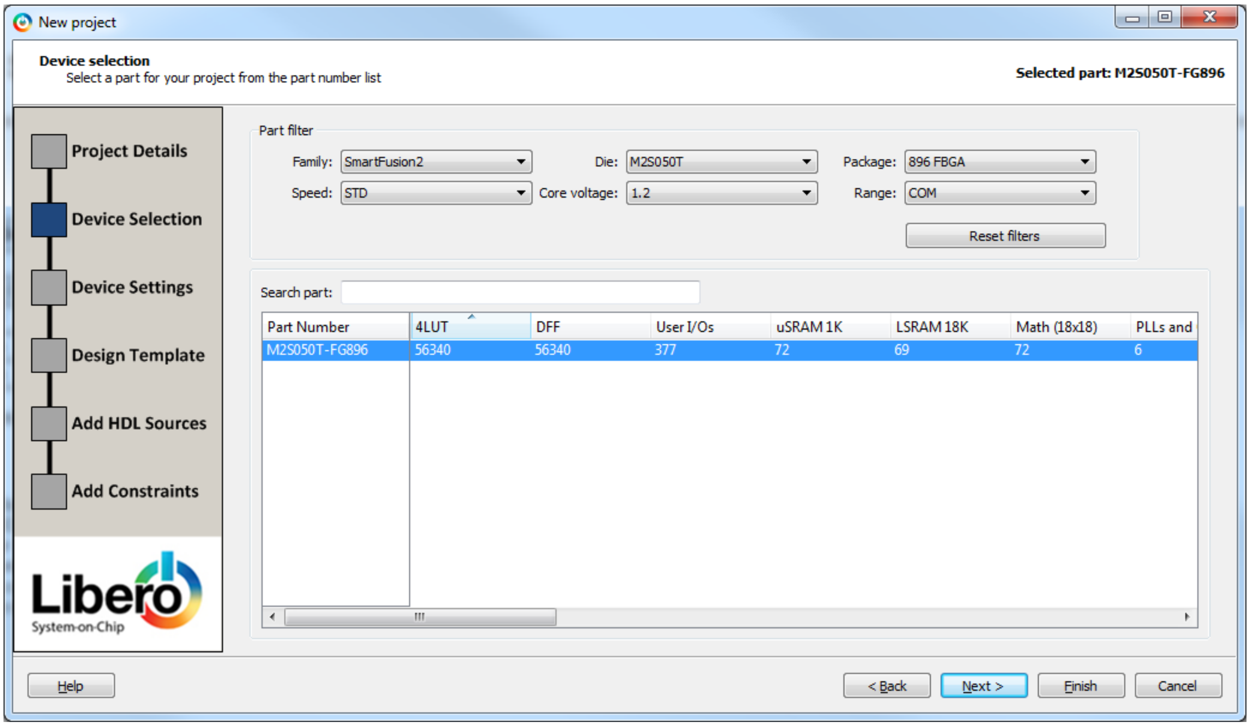
1. Launch Libero SoC v11.7.
2. From the **Project** menu, select **New Project**.
3. Enter the following information in the **Project Details** window, as displayed in Figure 2.
   - **Project Name**: Access_EXT_SDRAM
   - **Project Location**: Select an appropriate location (for example, D:/Microsemi_prj)
   - **Preferred HDL Type**: Verilog

Figure 2 • **New Project - Project Details Page**



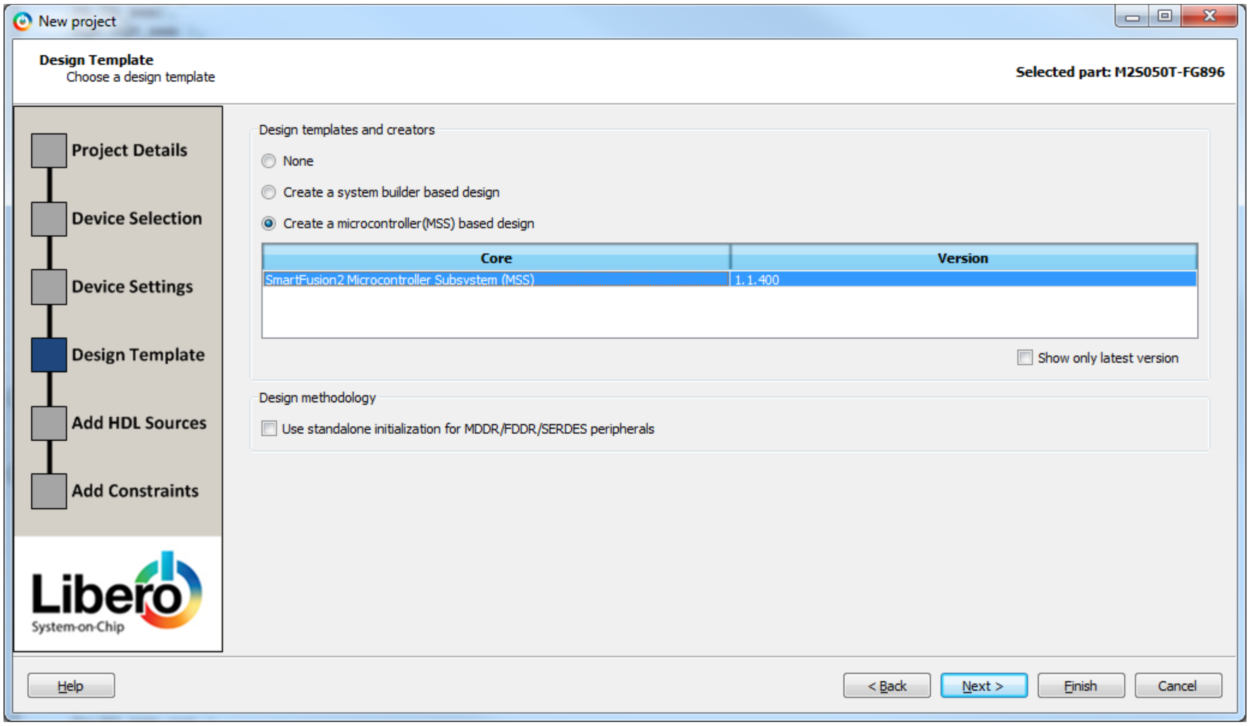
4. Click **Next**. The **Device Selection** window is displayed, as shown in Figure 3.
5. Select the following options from the drop-down list under **Part Filter**:
   - **Family**: SmartFusion2
   - **Die**: M2S050T
   - **Package**: 896 FBGA
   - **Speed**: STD
   - **Core Voltage (V)**: 1.2
   - **Range**: COM

Figure 3 • **New Project - Device Selection Page**



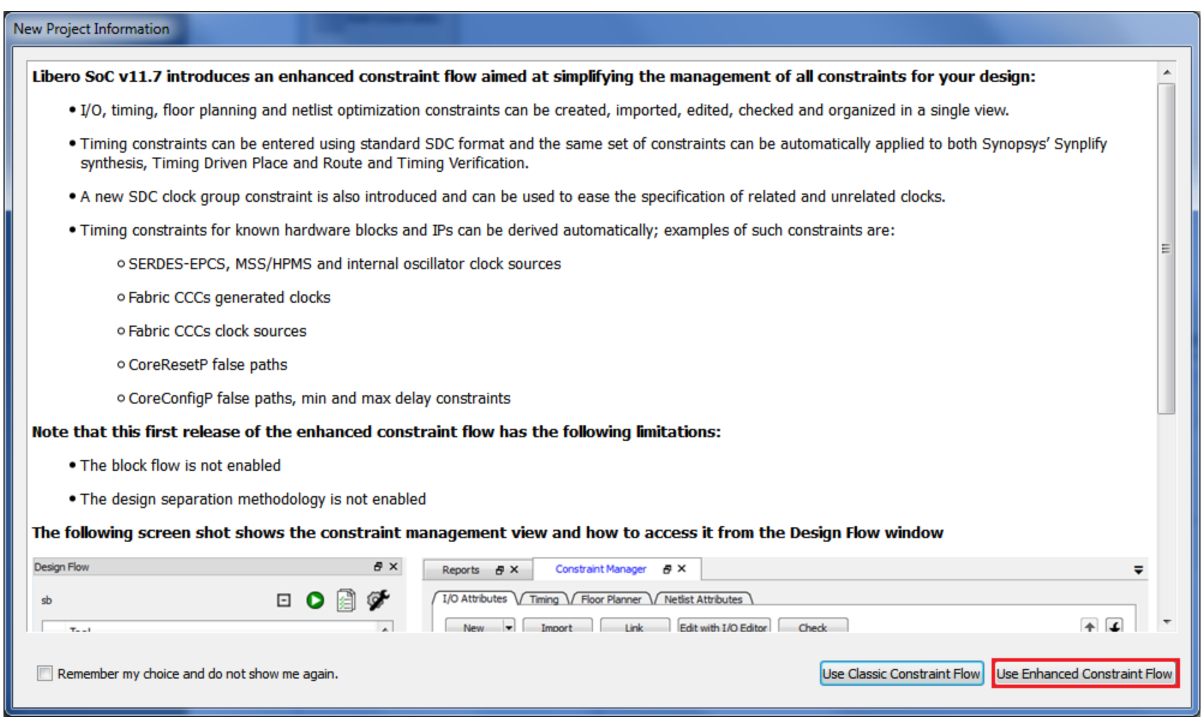
6. Click **Next**. The **Device Settings** page is displayed. Do not change the default settings.
7. Click **Next**. The **Design Templates and Creators** window is displayed, as shown in Figure 4.
8. Under **Design Templates and Creators**, select the **Create a Microcontroller (MSS) based design** check box. If the selected MSS core version appears in italics, it indicates that the selected MSS Core is not available in the vault and it requires to be downloaded. To download, select the MSS core and click **OK**. The tool prompts for downloading the MSS core. Click **Yes** on the message prompt. The tool downloads the selected MSS core.
   If the selected MSS core appears in normal font, as shown in Figure 4 on page 11, it indicates that the MSS core is present in vault.

Figure 4 • **New Project - Design Template Page**



9. Click **Finish**.
10. In the **New Project Information** window, click **Use Enhanced Constraint Flow** as shown in Figure 5.

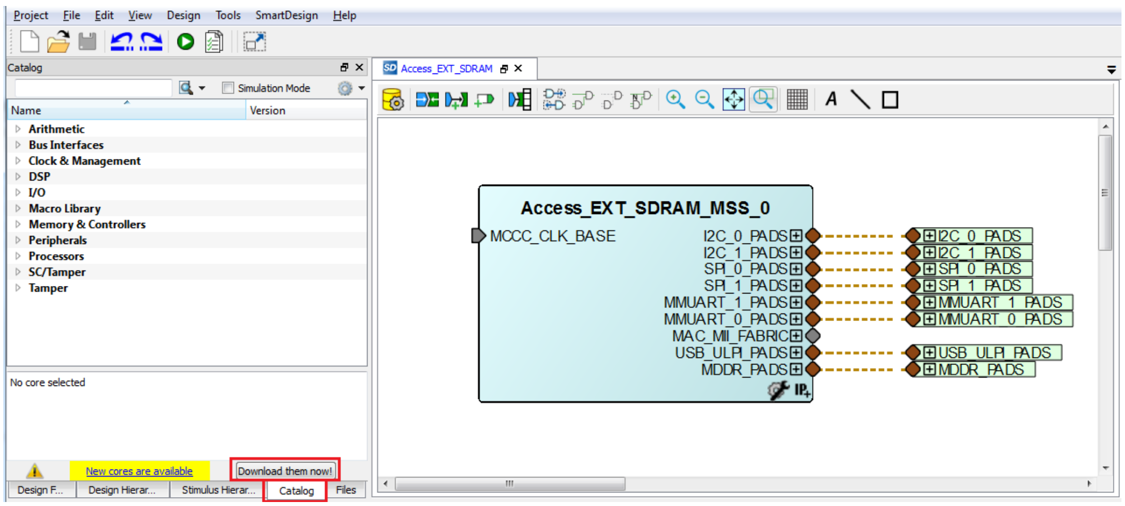
Figure 5 • **New Project Information Window**



2.4.2 Step 2: Updating IP Catalog

The project is created and the Libero SoC window is displayed as shown in Figure 6. The **SmartDesign** window opens and a project **Access_EXT_SDRAM** is created with the instantiation of the MSS component.

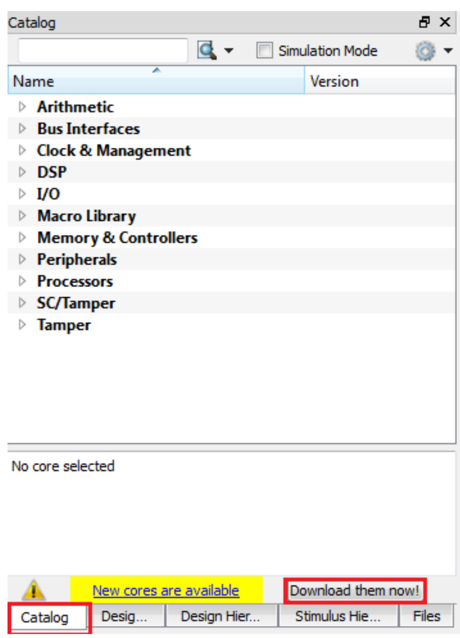
Figure 6 • **Libero Window on Completion of New Project Creation Wizard**



Click the **Catalog** tab, as shown in Figure 7. If a message **New cores are available** is displayed, click **Download them now!**, and download the latest versions of the IP cores.

Note: The download process requires internet connection.

Figure 7 • **Updating the Catalog**



2.4.3 Step 3: Configuring MSS Peripherals

1. Double-click **Acess_EXT_SDRAM_MSS_0** to configure the MSS. The MSS is displayed in the SmartDesign canvas in a new tab, as shown in Figure 8.
   The enabled MSS blocks are highlighted in blue and can be configured to be included in the hardware.

   The disabled peripherals are shown in gray.

   To disable a peripheral, right-click the peripheral block and clear the Disable check box, as shown in Figure 9, or clear the check box in the lower right corner of the peripheral box. The box turns gray to indicate that the peripheral are disabled. Disabled peripherals can be enabled by selecting the check box in the lower right corner of the peripheral box as shown in Figure 10 on page 14.

Figure 8 • **MSS in SmartDesign Canvas**

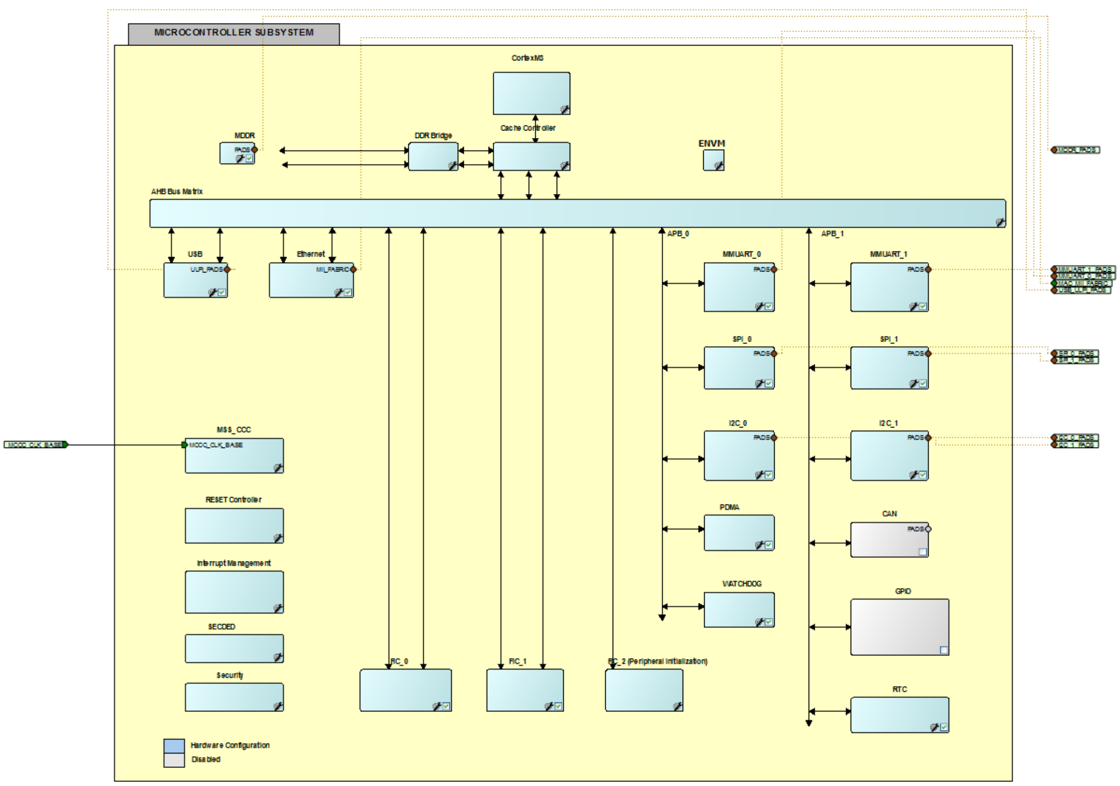
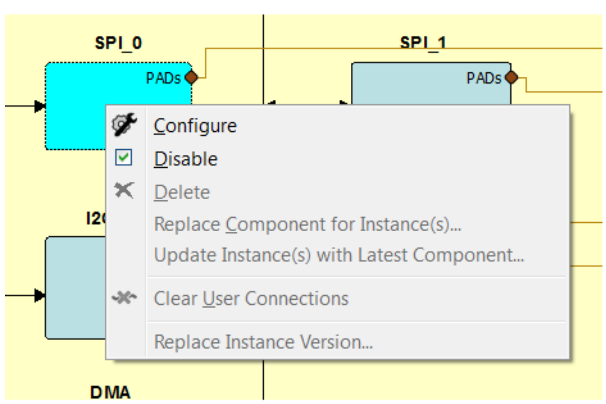
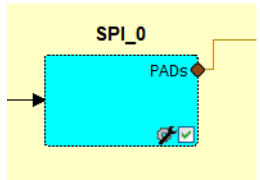


Figure 9 • **Right-Click and Disable Peripheral Block**



An enabled peripheral is shown in Figure 10.

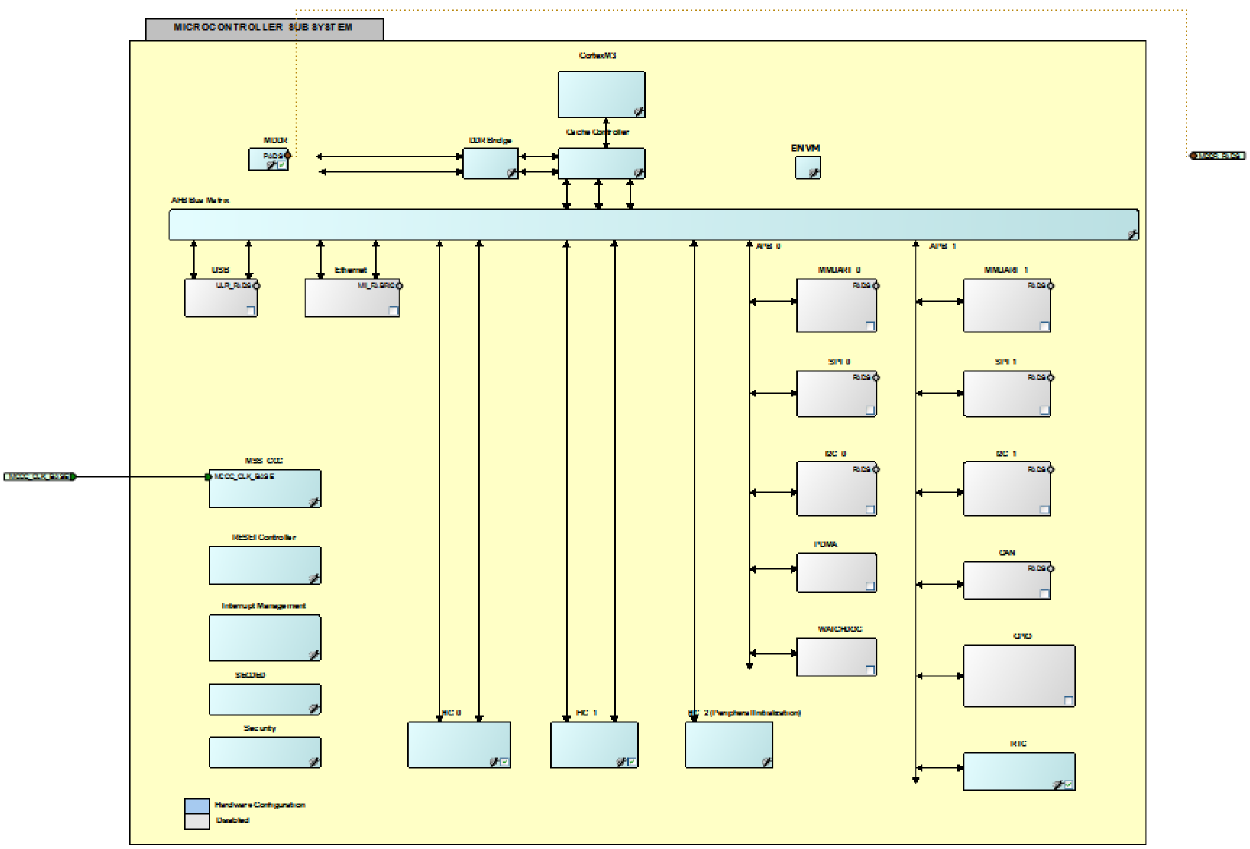
Figure 10 • **Enabling the Peripheral**



2. Disable the following peripherals on the MSS canvas:
   - MMUART_0 and MMUART_1
   - SPI_0 and SPI_1
   - I2C_0 and I2C_1
   - PDMA
   - WATCHDOG
   - FIC_0 and FIC_1
   - USB
   - Ethernet

   Figure 11 shows the **MSS Configuration** window after disabling the above components.

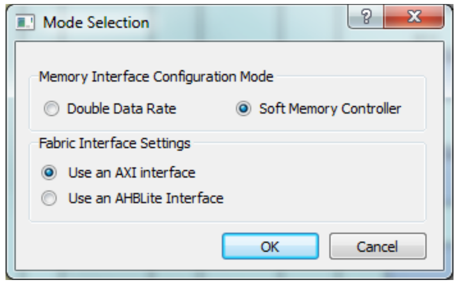
Figure 11 • **Enabled and Disabled MSS Components**



3. Double-click the **MDDR** block and configure the following as shown in Figure 11.
   - Select Soft **Memory** Controller as Memory Interface Configuration Mode.
   - Select **Use an AXI Interface** as Fabric Interface Settings.

   This selection configures the SMC_FIC interface inside the MDDR as a 64-bit AXI interface for the FPGA fabric from the DDR Bridge.
   - Click **OK**.

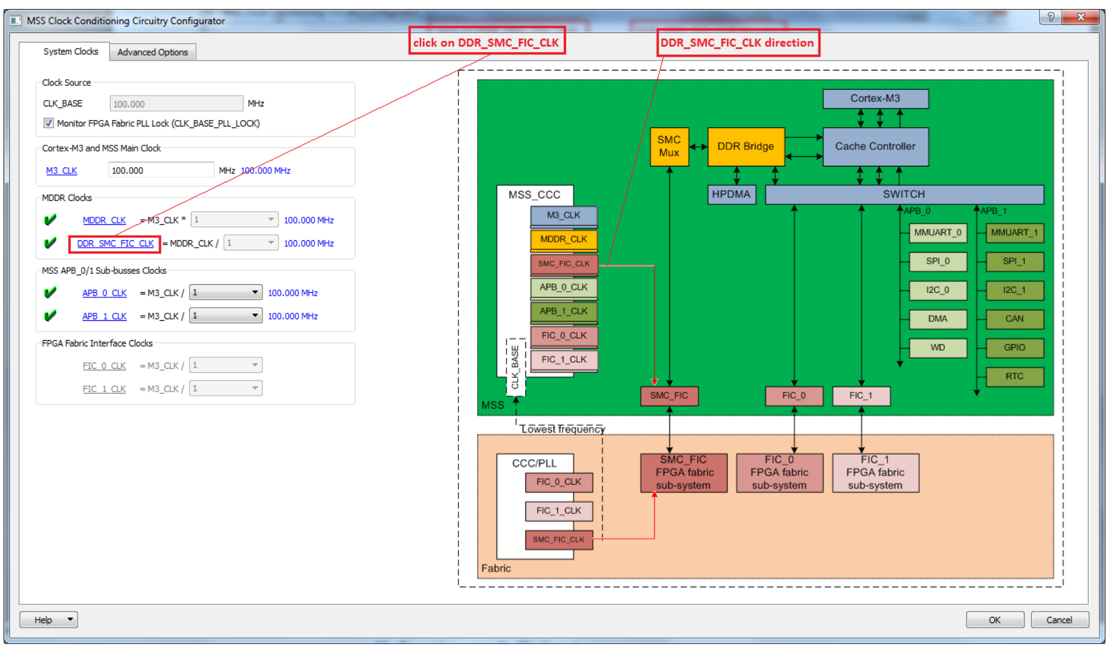
Figure 12 • **Mode Selection**



4. Double-click the **MSS_CCC** block and configure the following as shown in Figure 13.
   The clock input is by default selected as CLK_BASE with the input frequency of 100 MHz.
   - Select the check box for Monitor FPGA Fabric PLL Lock (CLK_BASE_PLL_LOCK).
   - Leave the default frequency of 100 MHz for M3_CLK.
   - Click DDR_SMC_FIC_ CLK to see the clock direction in the GUI. By default, DDR_SMC_FIC_CLK is set to the same frequency as that of M3_CLK (M3_CLK divided by 1; that is, 100 MHz).
   - Leave the remaining options as default.
   - Click **OK**.

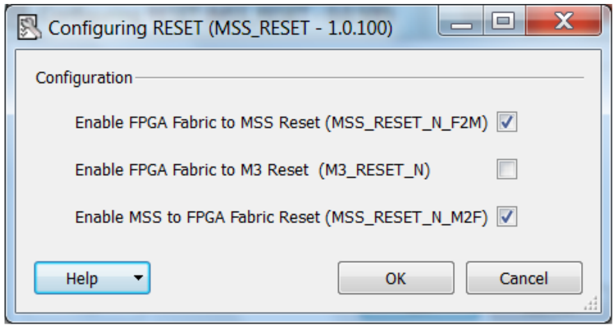
   The above selection configures the MSS CCC to receive the input clock from the fabric CCC. The lock input of the MSS CCC is configured to be received from the fabric CCC block.

Figure 13 • **MSS Clock Configurator**



5. Double-click **Reset Controller** and select **Enable MSS to Fabric Reset** and **Enable Fabric to MSS Reset**, as shown in Figure 14. This enables the MSS to generate the Reset signal for all the fabric blocks. The MSS reset comes through a system reset pin on the Fabric I/O.
6. Click **OK**.

Figure 14 • **MSS RESET Configurator**

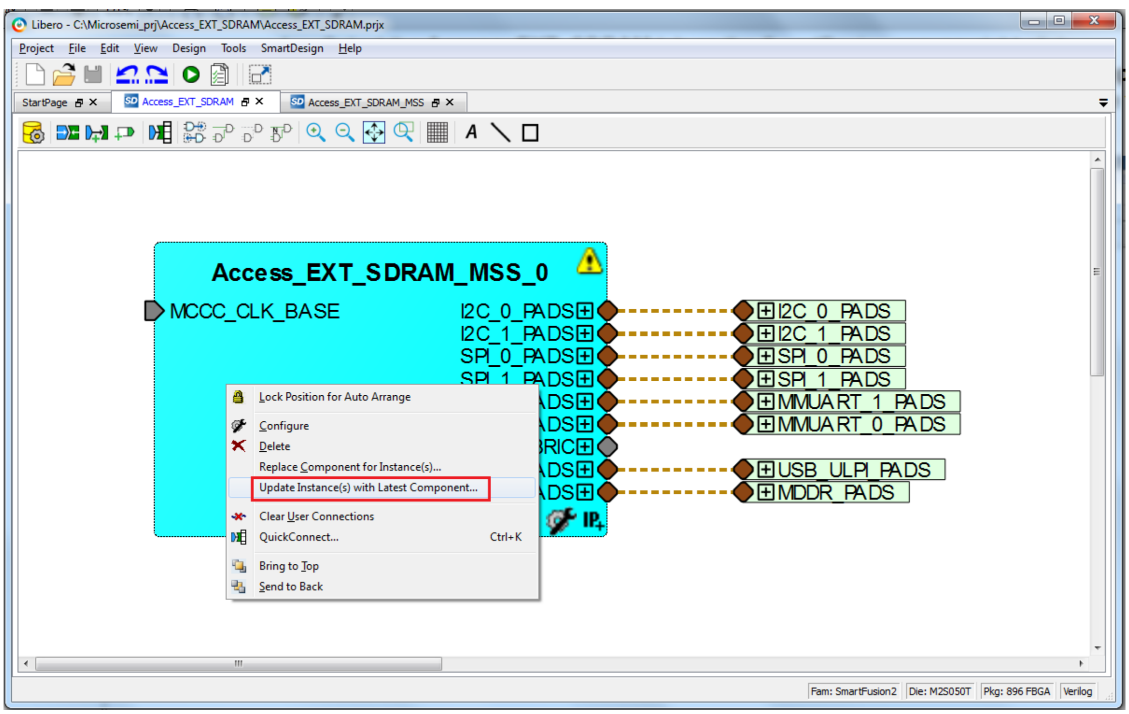


7. Select **File > Save** to save **Access_EXT_SDRAM_MSS**. This completes the configuration of the MSS.

2.4.4 Step 4: Updating MSS Component Instance

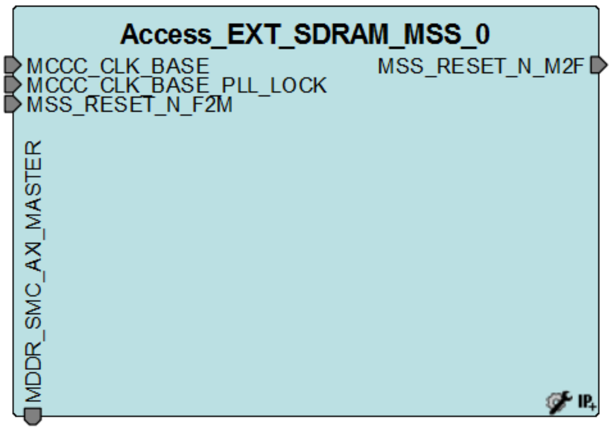
1. Select the **Access_EXT_SDRAM** tab on the SmartDesign canvas, right-click **Access_EXT_SDRAM_MSS_0**, and select **Update Instance(s) with Latest Component,** as shown in Figure 15.

Figure 15 • **Updating the MSS**



The **Access_EXT_SDRAM_MSS_0** instance after successful update is shown in Figure 16.

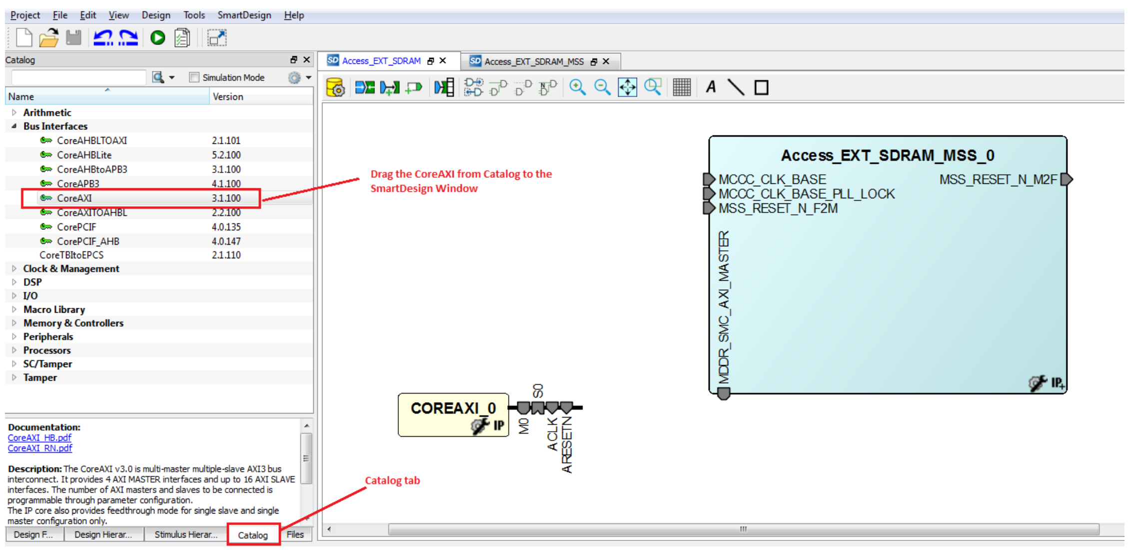
Figure 16 • **Updated MSS Instance**



2.4.5 Step 5: Configuring Fabric Components

1. In the **Catalog** tab, under **Bus Interfaces**, drag the **CoreAXI** IP onto the **Access_EXT_SDRAM** tab, as shown in Figure 17.

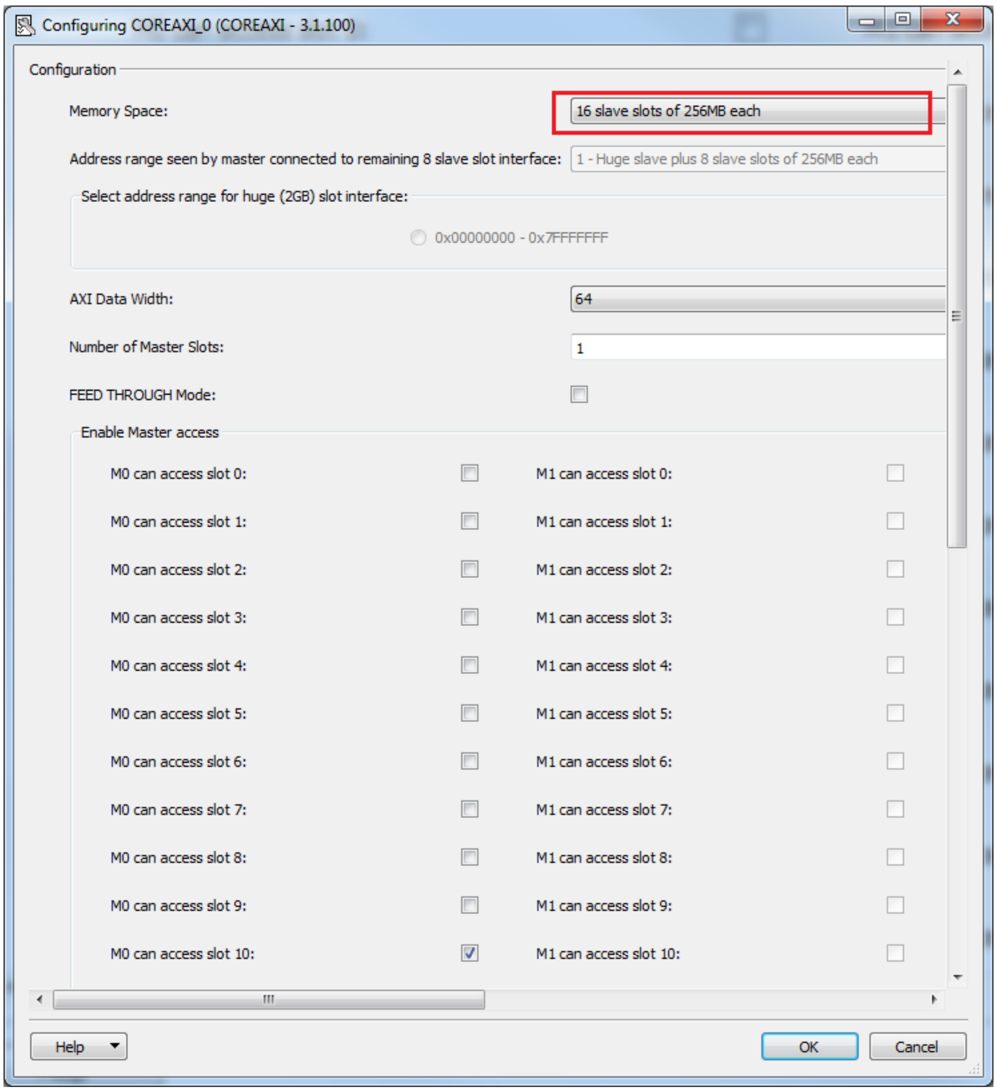
Figure 17 • **CoreAXI IP from the Catalog**



2. Double-click the **COREAXI_0** instance on the SmartDesign pane to open its configuration window. Configure the following items, as shown in Figure 18.
   - Leave the **Memory Space** field as 16 slave slots of 256 MB each, which is default.
   - Leave the **AXI Data Width** field as 64, which is default.
   - Leave the **Number of Master Slots** field as 1.
   - Clear the SLAVE0 for **Enable Master Access** checkbox.
   - Select the SLAVE10 for **Enable Master Access** checkbox.
   - Leave the remaining options as default.
   - Click **OK**.

   With the above settings, configure the **COREAXI_0** instance as a 64-bit AXI interface with Slave 10 slot enabled for Master0.

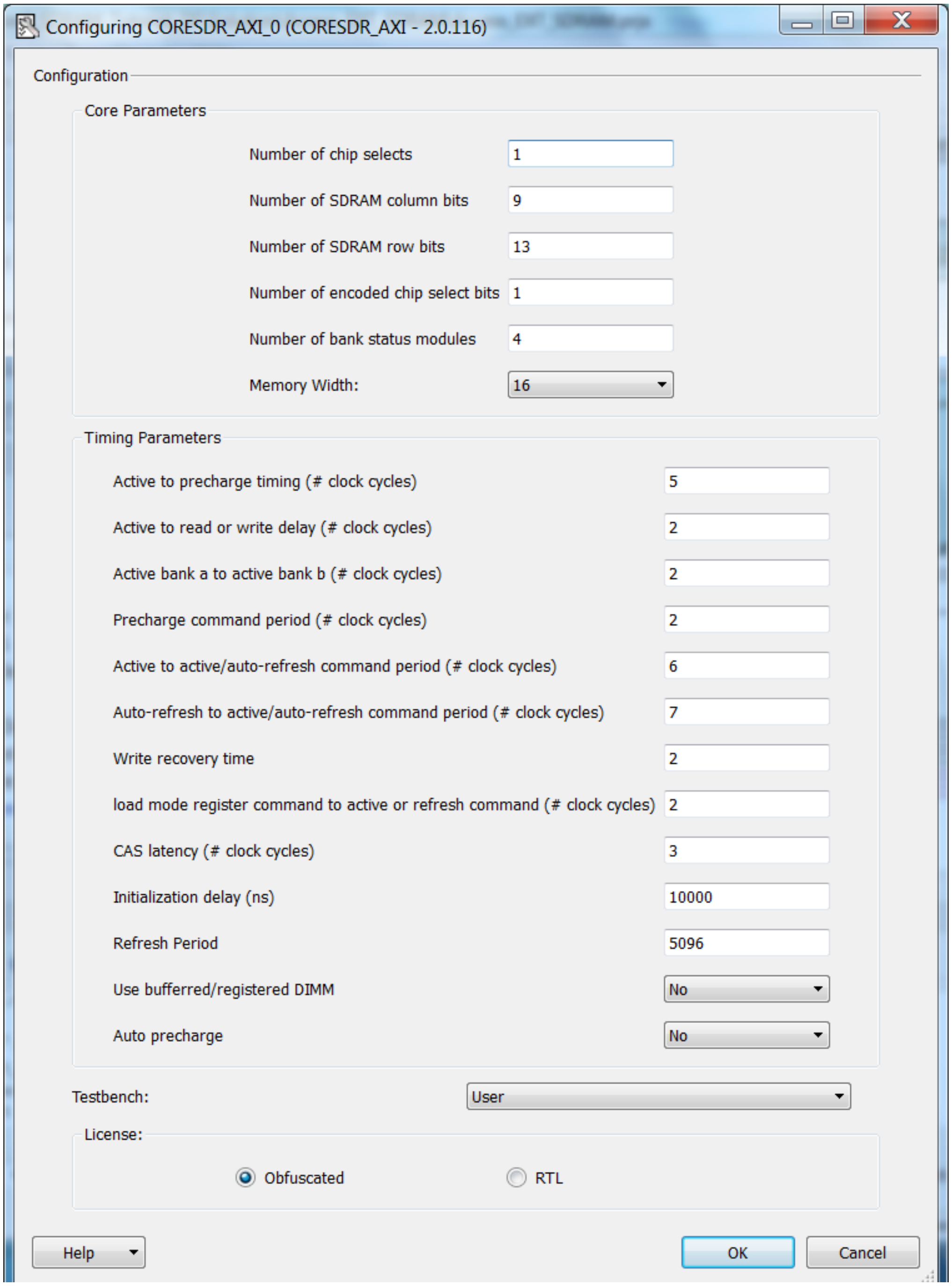
Figure 18 • **CoreAXI Configurator**



3. In the **Catalog** tab, under **Peripherals**, drag the **CoreSDR_AXI** IP onto the **Access_EXT_SDRAM** tab. Double-click the **CORESDR_AXI_0** instance to access its configuration window. Enter the details in the configuration window, as shown in Figure 19. These details are filled as per the datasheet of the Micron 256 MB SDRAM simulation model, which is used for functional simulation. The part number of the SDRAM is MT48LC16M16A2. It is a 4 Meg x 16 x 4 banks SDRAM.

Note: If any other SDRAM simulation model is used, configure **CORESDR_AXI** according to the specific SDRAM memory datasheet.

Figure 19 • **CoreSDR_AXI Configuration Window**



4. In the **Catalog** tab, under **Clock & Management**, drag the clock conditioning circuitry (CCC) block onto the **Access_EXT_SDRAM** tab. Double-click the **FCCC_0** instance to open up its configuration window.
   Configure the following items on the configuration window:
   - Select the **Advanced** tab as shown in Figure 20.
   - Select the clock source as **Oscillators > 25/50 MHz Oscillator**, as shown in Figure 21.
   - Leave the output frequency as **100 MHz**.
   - Leave the remaining options as default.
   - Select the **PLL Options** tab and select the **Expose PLL_ARST_N and PLL_POWERDOWN_N** check box, as shown in Figure 22 on page 21.
   - Click **OK**.

Figure 20 • **Advanced Tab of the FAB CCC Configurator**

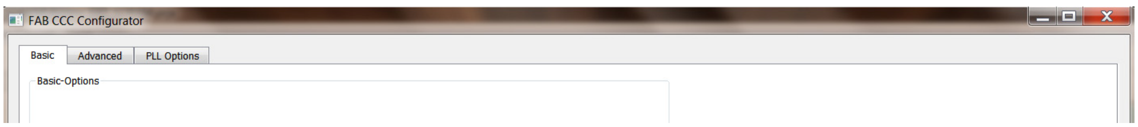


Figure 21 • **Selecting Clock Source**

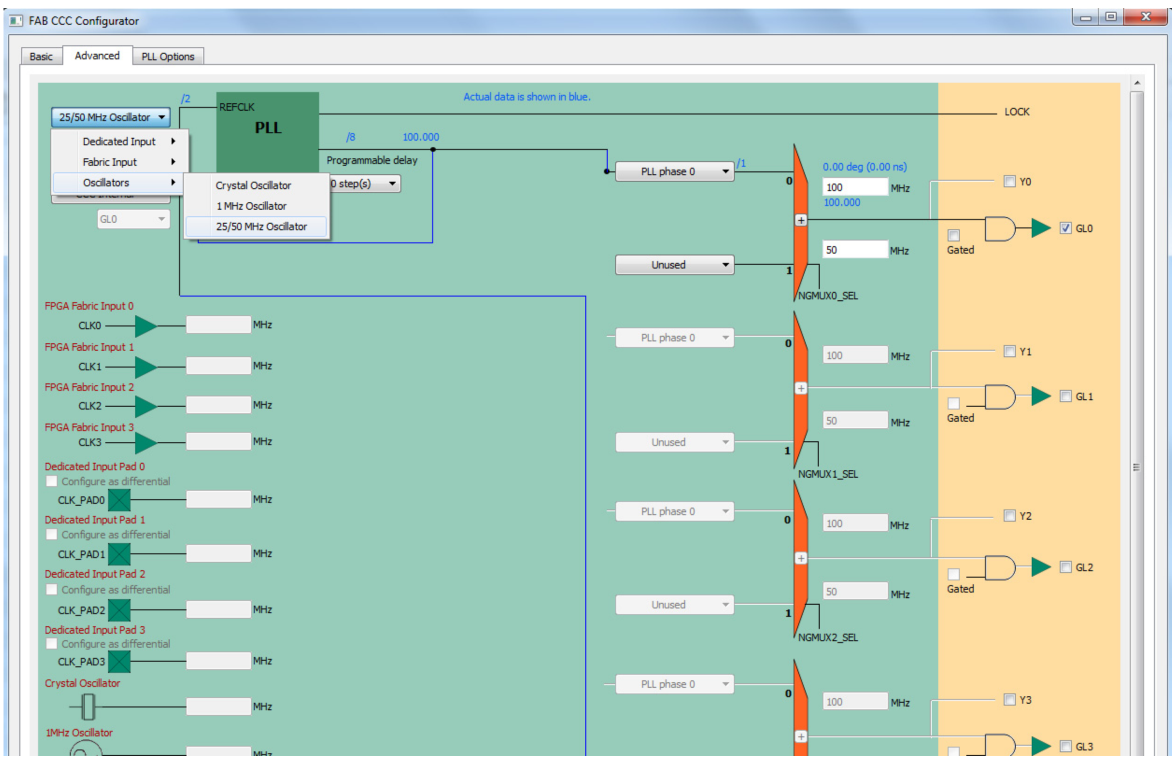
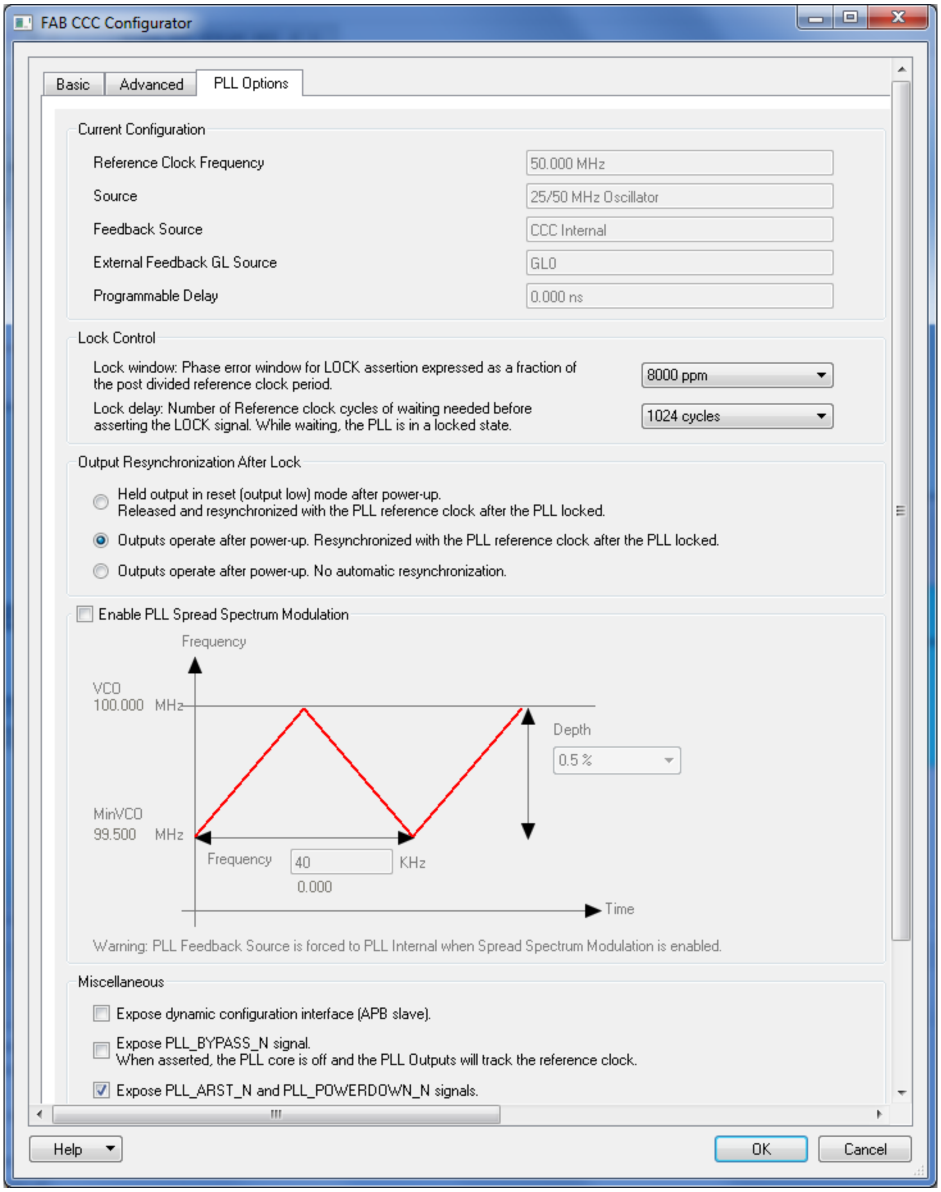


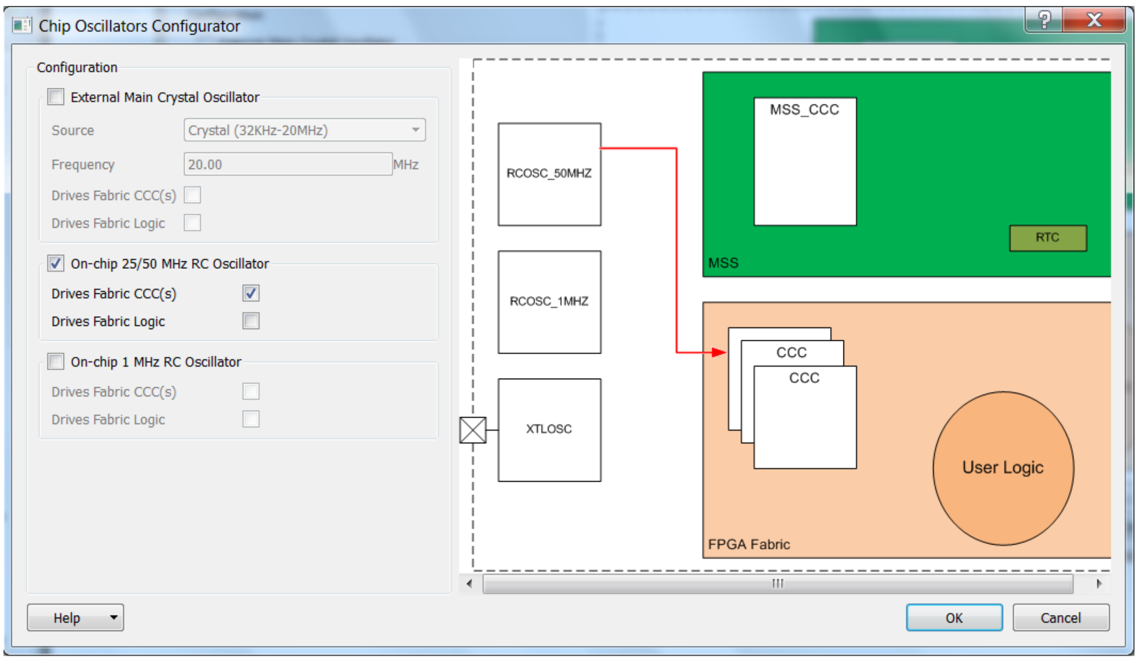
Figure 22 • **Exposing PLL Reset and Power-down Signals**



5. In the **Catalog** tab, under **Clock & Management**, drag the **Chip Oscillators** IP onto the **Access_EXT_SDRAM** tab. Double-click the **OSC_0** instance to open up its configuration window. Configure the following items, as shown in Figure 23:
   - Select the **On Chip 25/50 MHz RC Oscillator** check box.
   - Clear the **Drives MSS** check box.
   - Select the **Drives Fabric CCC(s)** check box.
   - Leave the remaining options as default.
   - Click **OK**.

   The on-chip 50 MHz RC oscillator is selected to drive the input of the fabric CCC block instantiated earlier.

Figure 23 • **Oscillator Configuration**



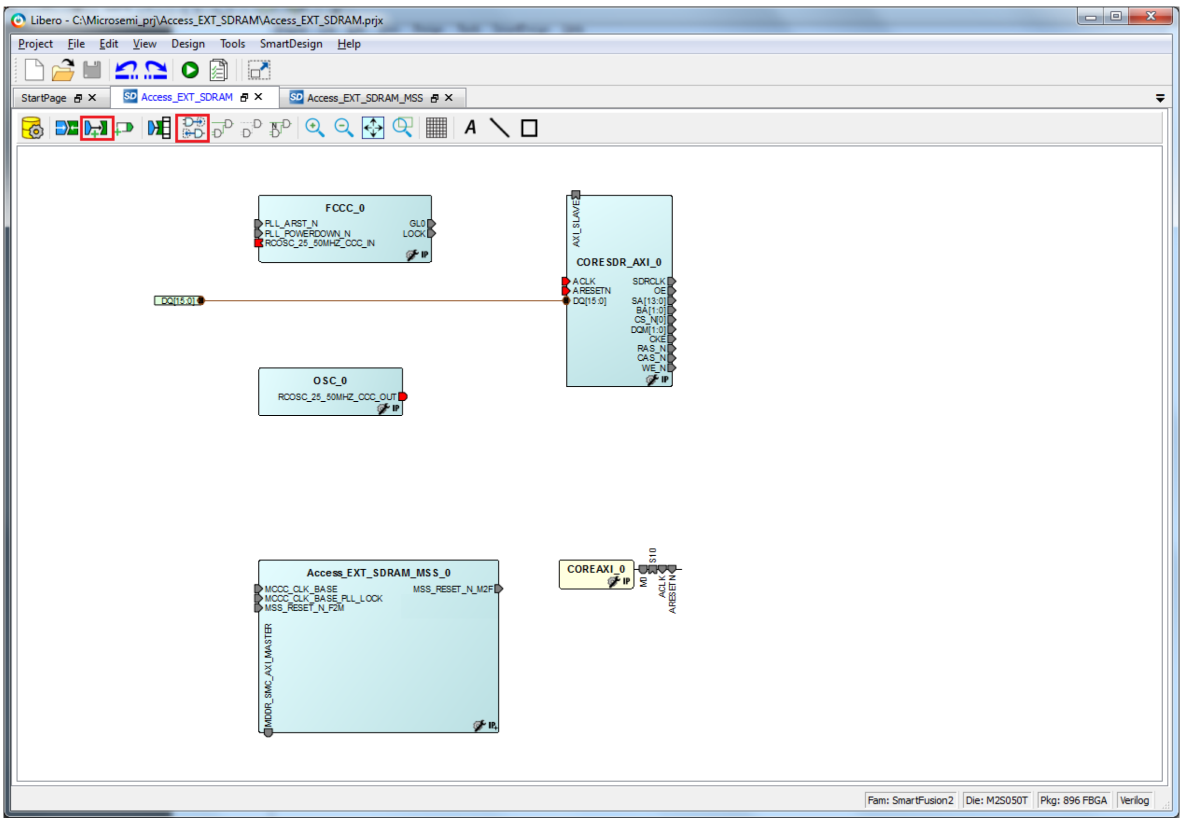
All the IPs for the fabric of the SmartFusion2 SoC FPGA device required in this design are configured. Arrange the IPs as required before connecting them.

2.4.6 Step 6: Interconnecting All Components

After arranging all the components on the **SmartDesign** window, connect the pins of all the blocks as described:

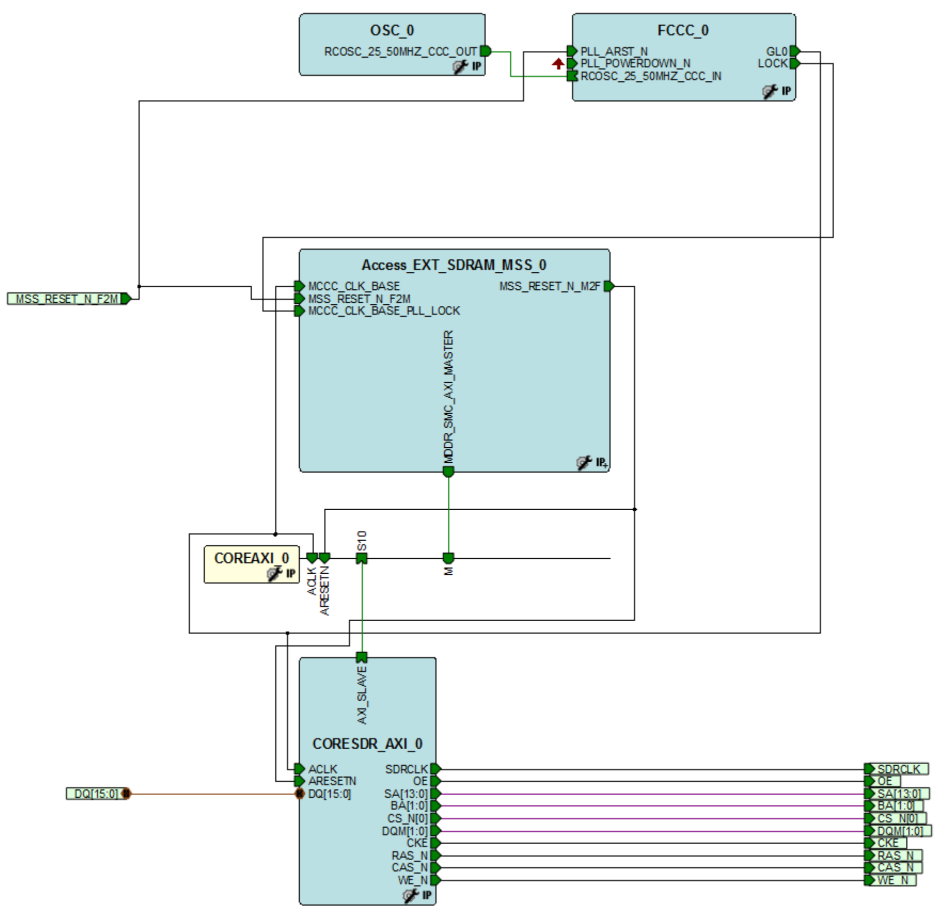
1. Use **Auto Arrange Instances** on the SmartDesign canvas to arrange the various instances, automatically. There are two ways to connect the components:
   - The first method is by using the Connection Mode option. Change SmartDesign to the connection mode by clicking **Connection Mode** on the **SmartDesign** window, as shown in Figure 24. The cursor changes from the normal arrow shape to the connection mode icon shape. Select the first pin and drag it to the second pin that needs to be connected.
   - The second method is by selecting the pins to be connected together and selecting **Connect** from the context menu. To select multiple pins to be connected together, hold the **CTRL** key as you select the pins. Right-click the input source signal and select **Connect** to connect all the signals together. In the same way, select the input source signal, right-click and select **Disconnect** to disconnect the signals.

Figure 24 • **Changing to Connection Mode**



2. Connect the following components as described below:
    - Connect ROSC_25_50MHZ_CCC_OUT(M) of OSC_0 to ROSC_25_50MHZ_CCC_IN(S) of FCCC_0.
    - Connect GL0 of FCCC_0 to MCCC_CLK_BASE of Access_EXT_SDRAM_MSS_0, ACLK of COREAXI_0, and ACLK of CORESDR_AXI_0. The fabric CCC clock output clocks all the blocks inside the fabric and is the input source clock for the MSS CCC block.
    - Connect LOCK of FCCC_0 to MCCC_CLK_BASE_PLL_LOCK input of Access_EXT_SDRAM_MSS_0.
    - Connect MSS_RESET_N_M2F of Access_EXT_SDRAM_MSS_0 to ARESETN of COREAXI_0 and ARESETN of CORESDR_AXI_0.
    - Connect M of COREAXI_0 to MDDR_SMC_AXI_MASTER of Access_EXT_SDRAM_MSS_0.
    - Connect S10 of COREAXI_0 to AXI_Slave of CORESDR_AXI_0.
    - Connect PLL_POWERDOWN_N inputs of FCCC_0 to logic '1'. Right-click each input signal, and select Tie High.
    - Promote the input signal of MSS_RESET_N_F2M of Access_EXT_SDRAM_MSS_0 to top-level. To do this, right-click the input signal, and select Promote to Top Level.
    - Select the top-level signal of MSS_RESET_N_F2M and the input signal PLL_ARST_N of the FCCC_0 instance and connect them. This connects the resets of the MSS and Fabric CCC to the top-level system reset input.
    - Promote all the output signals of CORESDR_AXI_0 to the top level. Hold the CTRL key and select each of them, right-click and select Promote to Top Level.
3. Click **Auto arrange instances** to arrange the instances, as shown in Figure 25. Save the design by selecting **File > Save**.

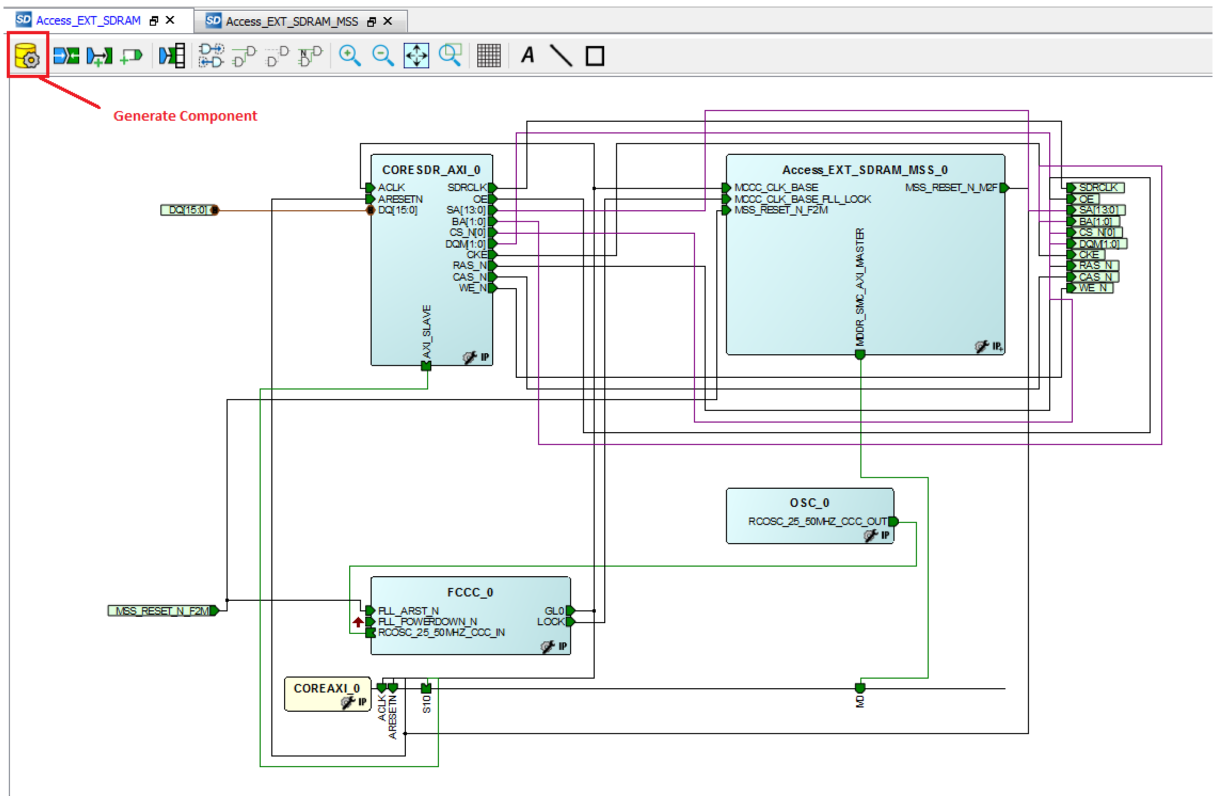
Figure 25 • **After Making the Top-Level Connection**



2.4.7 Step 7: Generating MSS and Top-Level Design

1. Select the **Access_EXT_SDRAM** tab on the **SmartDesign** canvas and click **Generate Component** on the SmartDesign pane, as shown in Figure 26 or select from **SmartDesign > Generate Component**.

Figure 26 • **Generating MSS Component**



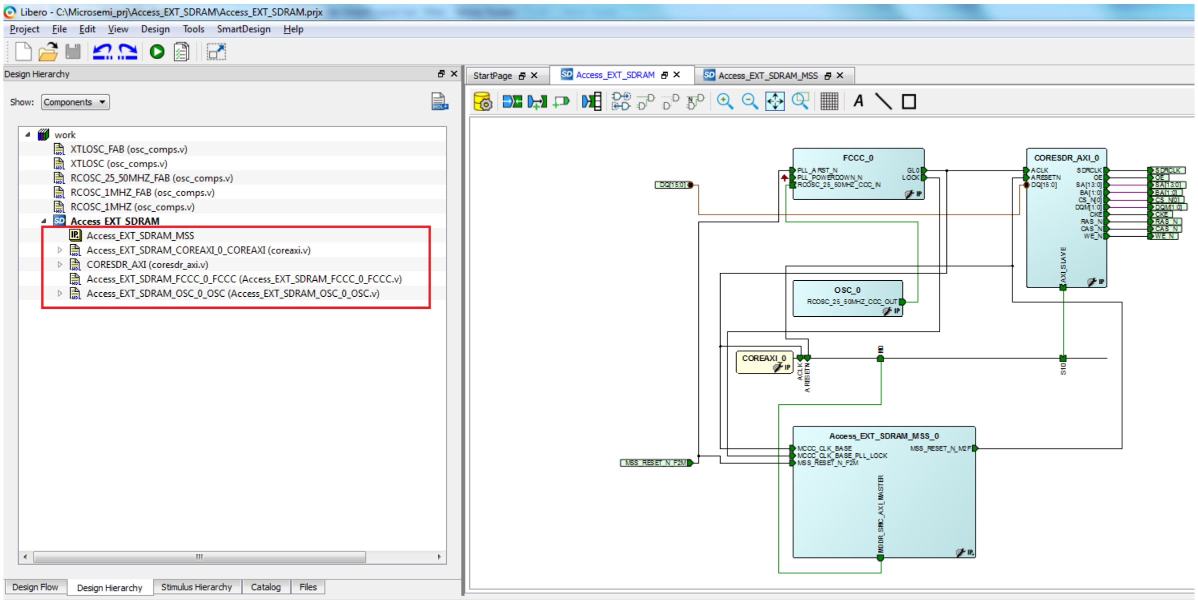
After successful generation of all the components, the following message is displayed on the log window:

Info: 'Access_EXT_SDRAM' was successfully generated.

Open datasheet for details.

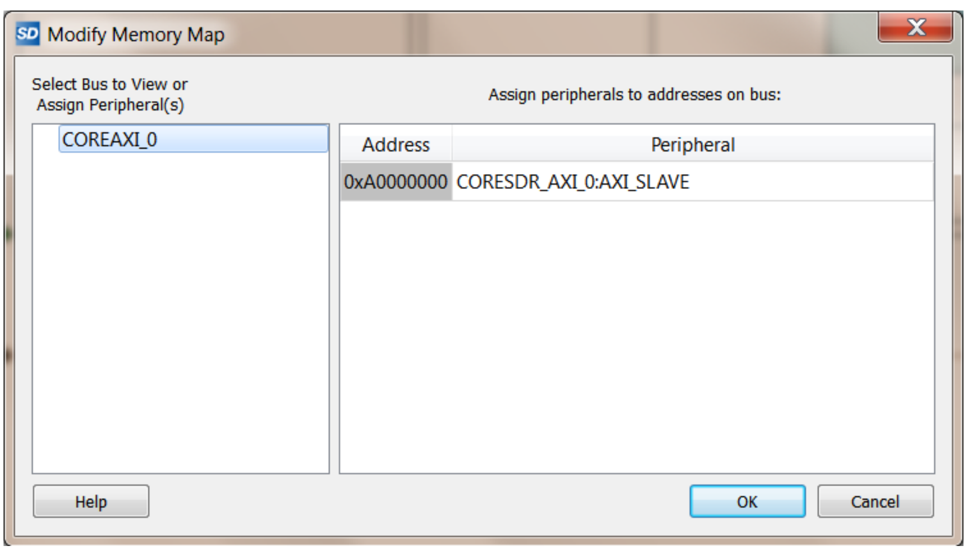
The design hierarchy can be found in the **Design Hierarchy** pane of Libero SoC, as shown in Figure 27 on page 26.

Figure 27 • **Design Hierarchy**



2. After generation, you can see the Memory Map for the CORESDR_AXI_0 component. Right-click the **Access_EXT_SDRAM** tab and select **Modify Memory Map**.
   Figure 28 shows the resultant memory map. The starting address of the MDDR Space 0 is 0xA0000000 in the Cortex-M3 processor address space.

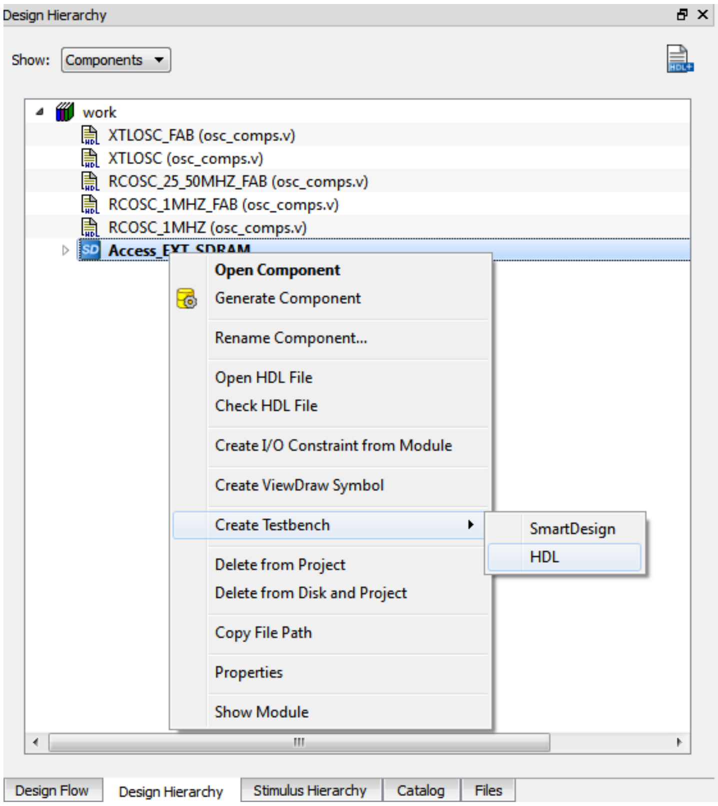
Figure 28 • **CORESDR_AXI_0 Memory Address**



2.4.8 Step 8: Generating Testbench and Adding SDR SDRAM Simulation Model

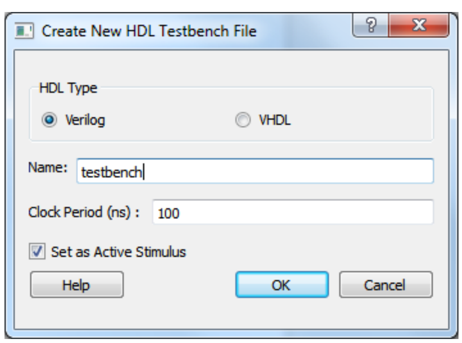
1. In the **Design Hierarchy** tab, right-click **Access_EXT_SDRAM** and go to **Create Testbench > HDL**, as shown in Figure 29.

Figure 29 • **Create Testbench – HDL**



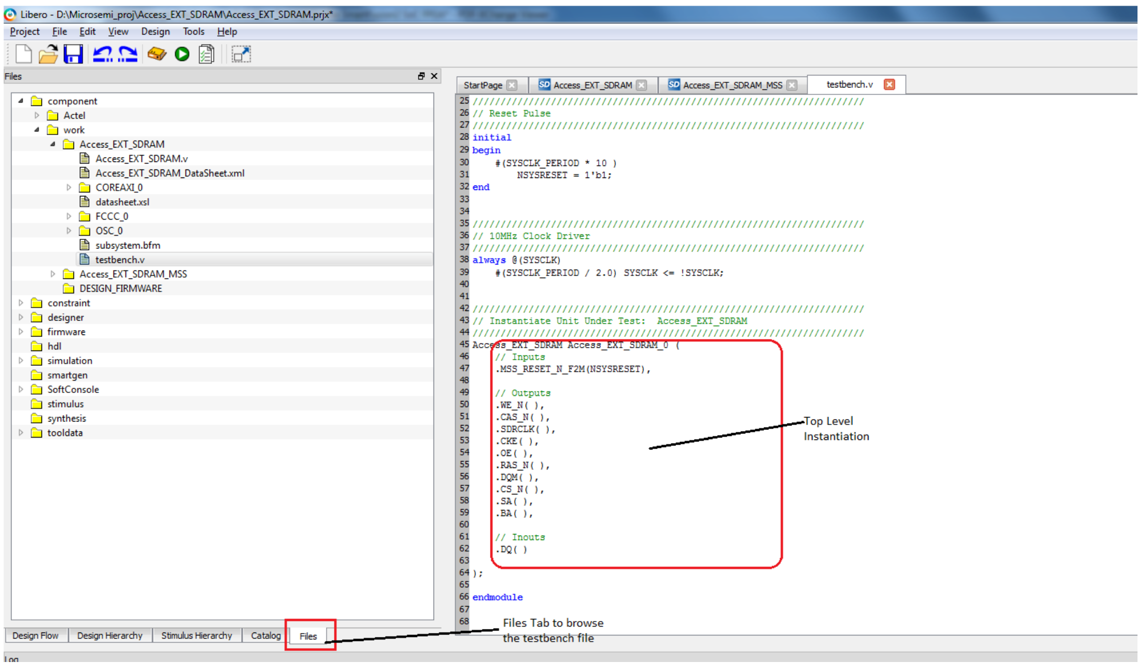
2. Enter the name as **testbench** in the **Create New HDL Testbench File** window and click **OK**, as shown in Figure 30.

Figure 30 • **Default Testbench**



3. In the generated testbench, add the external SDR SDRAM simulation model and map the port with the top-level design SDRAM interface signals. Double-click **testbench.v** in the **Files** tab to open the file, as shown in Figure 31.

Figure 31 • **Default Testbench**



4. Add the following lines of Verilog code to the testbench:
At the top of the file, include the SDR SDRAM simulation file.

```
`include "mt48lc16m16a2.v"
```

Declare the following signals in the testbench module:

```
// CORESDR_AXI signals
wire CAS_N_mem;
wire OE_mem;
wire WE_N_mem;
wire CS_N_mem;
wire [1:0] BA_mem;
wire SDRCLK_mem;
wire CKE_mem;
wire RAS_N_mem;
wire [13:0] SA_mem;
wire [15:0] DQ_mem;
wire [1:0] DQM_mem;
// SDR SDRAM interface signals with the CORESDR_AXI
wire CAS_N_mem_out;
wire WE_N_me_out;
wire CS_N_mem_out;
wire [1:0] BA_mem_out;
wire CKE_mem_out;
wire RAS_N_mem_out;
wire [13:0] SA_mem_out;
```

```
wire [15:0] DQ_mem_out;
wire [1:0] DQM_mem_out;
```

Modify the top-level instantiation of **Access_EXT_SDRAM**, as shown below:

```
///////////////////////////////////////////////////////////////////////
// Instantiate Unit Under Test:  Access_EXT_SDRAM
///////////////////////////////////////////////////////////////////////
Access_EXT_SDRAM Access_EXT_SDRAM_0 (
    // Inputs
    . MSS_RESET_N_F2M (NSYSRESET),
    // Outputs
    .CAS_N(CAS_N_mem ),
    .OE(OE_mem ),
    .WE_N(WE_N_mem ),
    .CS_N(CS_N_mem ),
    .BA(BA_mem ),
    .SDRCLK(SDRCLK_mem ),
    .CKE(CKE_mem ),
    .RAS_N(RAS_N_mem ),
    .SA(SA_mem),
    .DQM(DQM_mem ),
    // Inouts
    .DQ(DQ_mem)
);
```

SDRAM uses source-synchronous clock. Ensure that the SDRAM signals are received after the rising edge of the clock. A delay of 1 ns is added to the SDR SDRAM interface signals with the CORESDR_AXI, as shown below:

```
assign #1 CKE_mem_out = CKE_mem;
assign #1 RAS_N_mem_out = RAS_N_mem;
assign #1 CAS_N_mem_out = CAS_N_mem;
assign #1 WE_N_mem_out = WE_N_mem;
assign #1 SA_mem_out = SA_mem;
assign #1 CS_N_mem_out = CS_N_mem;
assign #1 BA_mem_out = BA_mem;
assign #1 DQM_mem_out = DQM_mem;
assign #1 DQ_mem_out =  OE_mem ? DQ_mem: {16{1'bz}};
assign DQ_mem = OE_mem ? {16{1'bz}}: DQ_mem_out;
```

Micron's **MT48LC16M16A2** SDR SDRAM is instantiated in the testbench as shown below:

```
///////////////////////////////////////////////////////////////////////
// Instantiate SDR SDRAM
///////////////////////////////////////////////////////////////////////
 mt48lc16m16a2 mt48lc16m16a2_0 (
 // Inputs
    .Addr(SA_mem_out[12:0]),
    .Ba(BA_mem_out ),
    .Clk(SDRCLK_mem  ),
    .Cke(CKE_mem_out ),
    .Cs_n(CS_N_mem_out),
```

```
        .Ras_n(RAS_N_mem_out ),
        .Cas_n(CAS_N_mem_out ),
        .We_n(WE_N_mem_out ),
        .Dqm(DQM_mem_out ),

        // Inouts
        .Dq(DQ_mem_out )
);
```

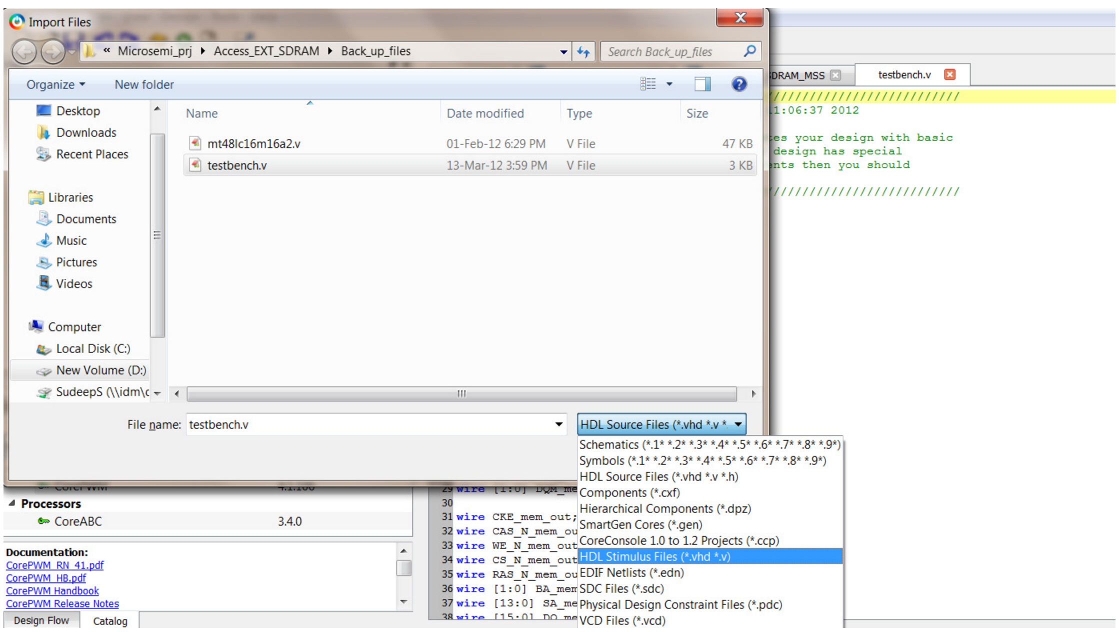
Save the file by selecting **File > Save testbench.v**

Note: The modified `testbench.v` file is provided in the following location in the attached compressed project:

`<Project_directory>\ ACCESS_EXT_SDRAM\Source`

To use the provided modified `testbench.v` file, import it as a stimulus file by selecting **File > Import Files**. In the **Import Files** dialog box, select the file type as **HDL Stimulus Files** (`*.vhd, *.v`). Browse to the above location of `testbench.v` and import it, as shown in Figure 32. The `testbench.v` file is shown under the **Stimulus** folder in the **Files** tab.

Figure 32 • **File Import to Stimulus Folder**



5. Import the `mt48lc16m16a2.v` file from the location in the attached compressed project `<Project_directory>\ ACCESS_EXT_SDRAM\Source` to the project's **Stimulus** folder location as follows:

   Select **File > Import File**. In the **Import Files** dialog box, select the file type as

   HDL Stimulus Files (`*.vhd, *.v`). Browse to the above mentioned location of the mt48lc16m16a2.v file and import it. The `mt48lc16m16a2.v` file is seen under the **Stimulus** folder in the **Files** tab.

   After saving the modified testbench file, it can be checked for syntax errors. On the testbench.v source window, right-click and select the **Check HDL** file. It checks the `testbench.v` file for any syntax errors.

2.4.9 Step 9: Adding BFM Commands to Perform Simulation

1. The user BFM commands are added in a file named `user.bfm`, which can be found in the following location in the project:
   `<Project_directory>\Access_EXT_SDRAM\simulation`

   Browse to the *user.bfm* file under simulation file in the **Files** tab in Libero SoC and double-click it to open the file. Add the following commands to it:

   Before **procedure user_main**, add the following command:

   ```
   memmap CORESDR_AXI_0   0xA0000000;
   ```

   Comment the following line in the `user.bfm` file using hash (#)

   ```
   "include "subsystem.bfm""
   ```

   Under the **procedure user_main** section, add the BFM commands that are in the red boxes below.

   ```
   int i
   # perform subsystem initialization routine
   #call subsystem_init;
   print "M_DDR0_CTRL_REGS TEST START";
   loop i 0 110 1
   wait 100ns
   endloop
   # add your BFM commands below:
   write w CORESDR_AXI_0 0x0000 0xA1B2C3D4 ;
   write w CORESDR_AXI_0 0x0004 0x10100101 ;
   write w CORESDR_AXI_0 0x0008 0xA5DEF6E7 ;
   write w CORESDR_AXI_0 0x000C 0xD7D7E1E1 ;

   readcheck w CORESDR_AXI_0 0x0000 0xA1B2C3D4 ;
   readcheck w CORESDR_AXI_0 0x0004 0x10100101 ;
   readcheck w CORESDR_AXI_0 0x0008 0xA5DEF6E7 ;
   readcheck w CORESDR_AXI_0 0x000C 0xD7D7E1E1 ;
   print "M_DDR0_CTRL_REGS TEST ENDS";
   print ""
   ```

   Save the `user.bfm` file by selecting **File > Save**.

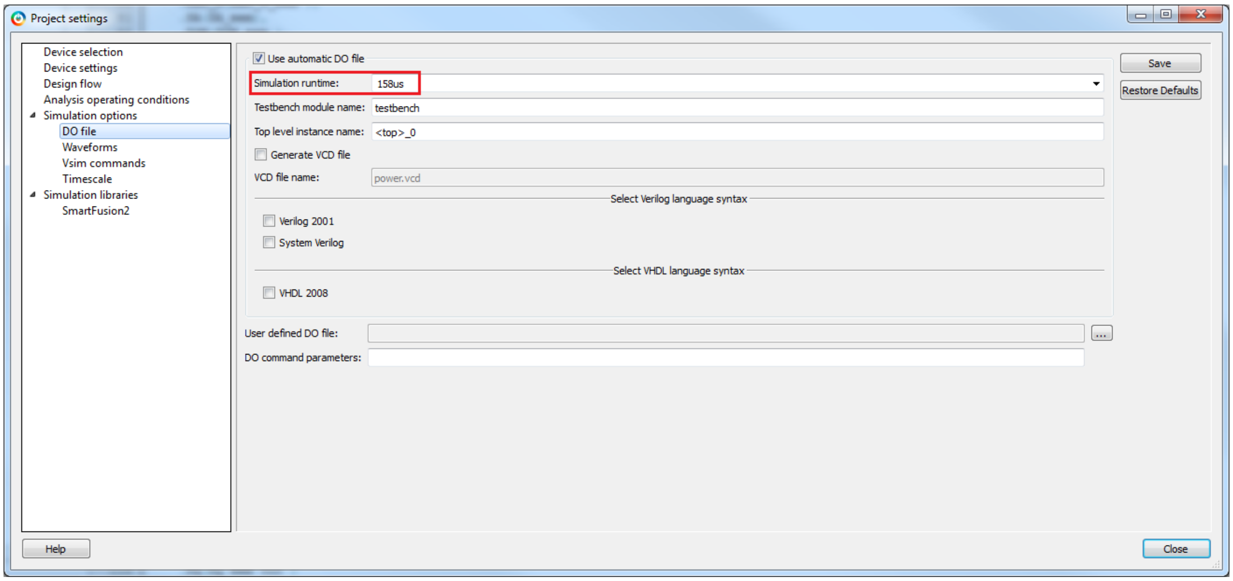
   See the *CoreAMBA BFM User's Guide* for more information about the BFM commands. The sample *user.bfm* file can be found in the following location in the attached compressed project:

   `<Project_directory>\ ACCESS_EXT_SDRAM\Source`

2.4.10 Step 10: Setting up Simulation and Opening Simulation Tool

1. The simulation tool must be set up before opening to load with the desired settings. Select **Project > Project Settings > Simulation Options > Do File**.
2. Set **Simulation Runtime** to **158 µs**, as shown in Figure 33.

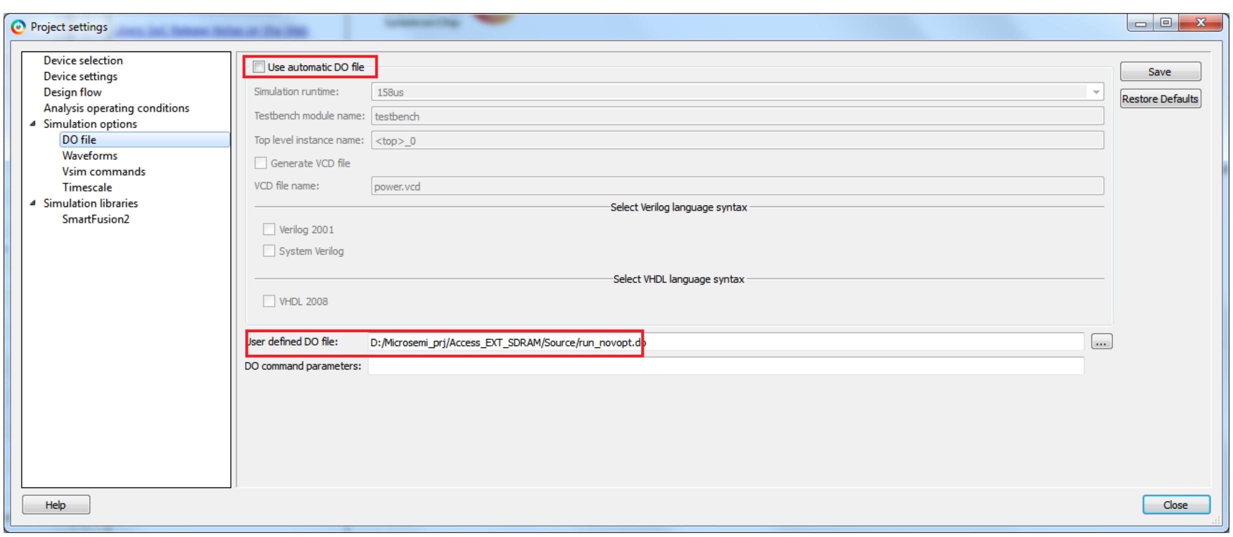
Figure 33 • **Simulation Runtime**



Note: For VHDL flow:

- Micron SDRAM memory models are only available in Verilog. For VHDL flow, use the ModelSim full version, for example, ModelSim SE, as ModelSim AE does not support mixed-language flow. Compile with **-novopt** switch, if ModelSim full version is used.
- A .do file, `run_novopt.do`, which has the switch already set, is provided along with the source files in the tutorial zip files. To use the provided `run_novopt.do` file, clear the **Use automatic DO** file check box and browse to the location of the provide `run_novopt.do` file, as shown in Figure 34.

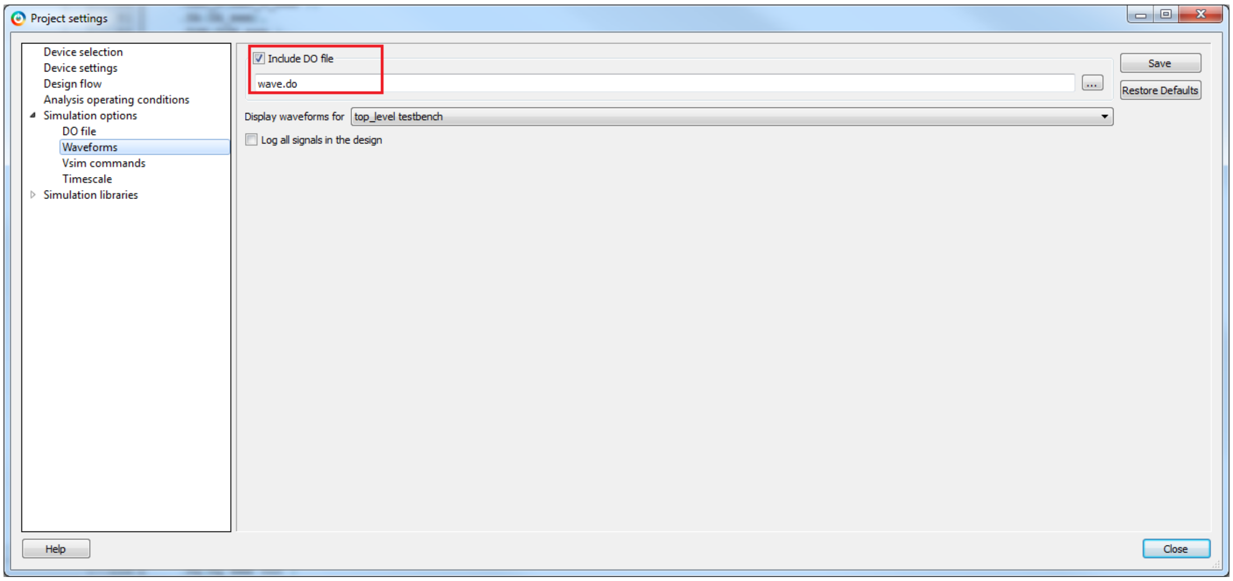
Figure 34 • **Specifying run_novopt.do for VHDL ModelSim Full Version**



3. Select the waveforms under **Simulation Options** and select the **Include DO File** option.
   This option allows to specify a custom macro file, which sets up the **ModelSim Wave** window with the required signals added to the **Wave** window. A custom macro file, *wave.do*, is provided at the following location in the attached compressed project:
   `<Project_directory>\ ACCESS_EXT_SDRAM\Source`
   This **DO** file adds all the AXI bus signals and the CORESDR_AXI interface signals with the external SDR SDRAM memory.
   Browse to the `wave.do` file from the above specified location, as shown in Figure 35.

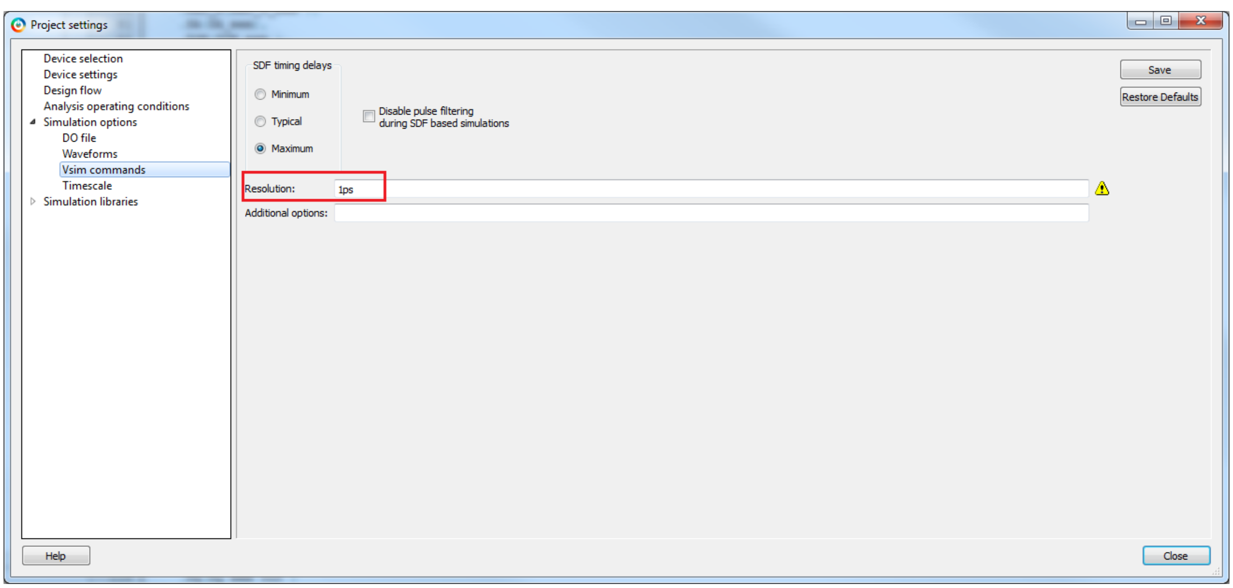
Note: To add your signals in the **ModelSim Wave** window during simulation, do not select the **Include DO File** check box.

Figure 35 • **Adding Custom DO File for ModelSim Wave Window**



4. Select the **Vsim Commands** option under the **Simulation Options**, and modify the **Resolution** to **1ps**, as shown in Figure 36. This option sets the simulation resolution to 1 ps.

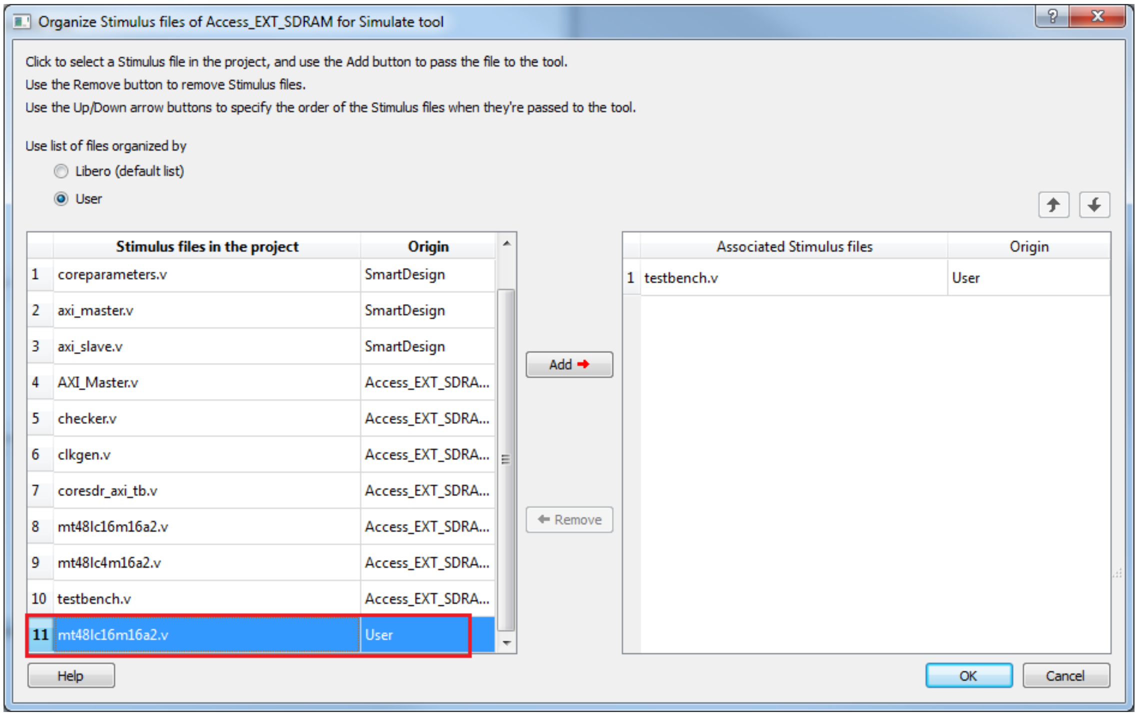
Figure 36 • **Simulation Resolution**



5. Click **Save** and **Close** to exit the **Project Settings** window.

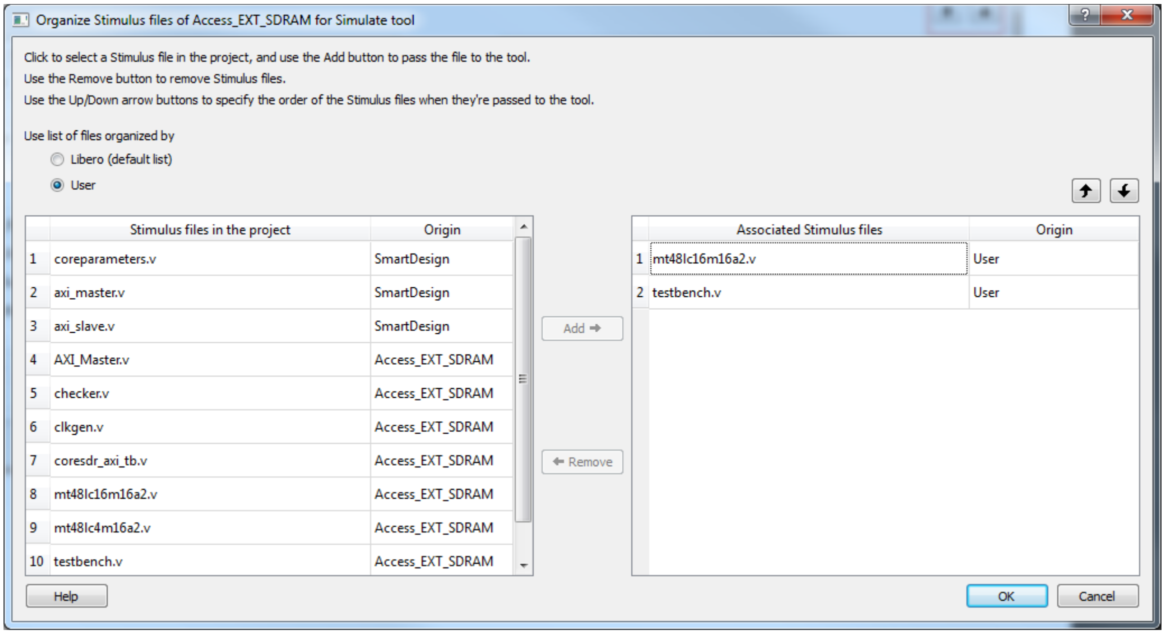
6. In Libero SoC, on the **Design Flow** tab Libero SoC, expand **Verify Pre-Synthesized Design,** select **Simulate** and do the following:
   - Specify the testbench ModelSim to be used during simulation. To do so, right-click **Simulate** and select **Organize Input Files > Organize Stimulus Files**. The **Organize Stimulus files of Access_EXT_SDRAM for Simulate tool** window opens.
   - Under **Stimulus files in the project**, select the `mt48lc16m16a2.v` file and click **Add** to add the file to the **Associated Stimulus files**, as shown in Figure 37.

Figure 37 • **Organizing Stimulus Files**



After organizing the stimulus file, the above window looks similar to Figure 37 on page 34. If the files are not in the order, as shown in Figure 38, use up and down arrows to move the files in the correct order.

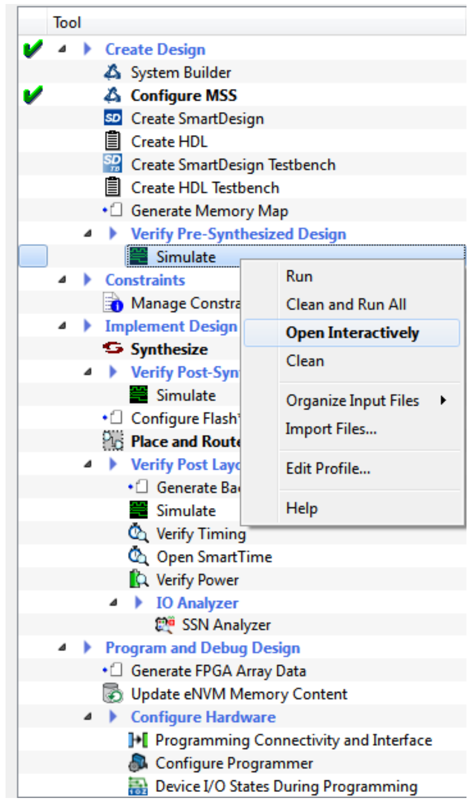
Figure 38 • **Organized Stimulus Files for the Simulation**



7. Click **OK**.

8. After specifying the testbench stimulus file, expand **Verify Pre-Synthesized Design**, right-click **Simulate**, and select **Open Interactively** to invoke ModelSim, as shown in Figure 39. ModelSim is invoked and the design is loaded.

Figure 39 • **Invoke ModelSim**



2.4.11 Step 11: Viewing Simulation Results

1. ModelSim runs the design for about 158 µs, as specified in the **Project Settings** window. After the simulation has run completely, undock the **Wave** window by clicking **Dock/Undock** on the **Wave** window, as shown in Figure 40.

Figure 40 • **Dock/Undock Button in Wave Window**

2. Click **Zoom Full** to fit all the waveforms in the single view (Figure 41).

Figure 41 • **Zoom Full Button**

3. Place the cursor at 114 µs on the **Wave** window and click **Zoom In on the Active Cursor** to zoom in at that location, as shown in Figure 42. Click as needed until all write and read transactions to the external SDR SDRAM are seen on the **Wave** window, as shown in Figure 43 on page 37.

Figure 42 • **Zoom In on the Active Cursor**

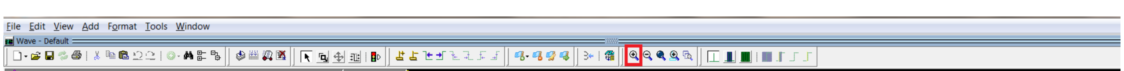
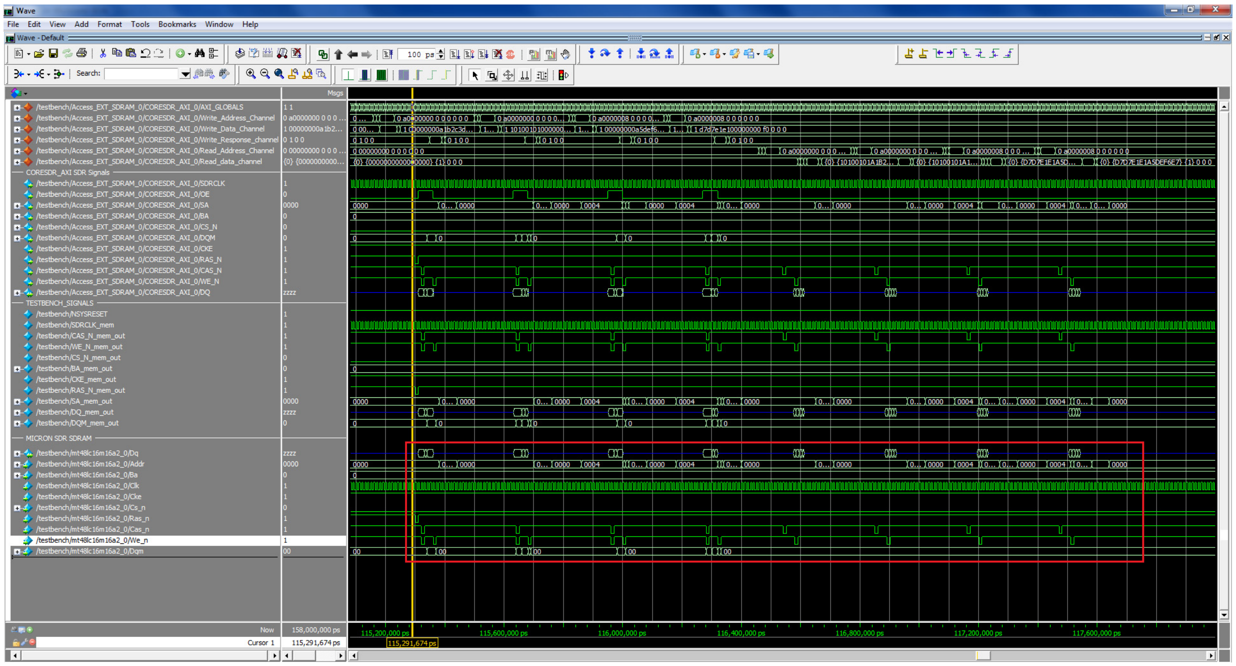


Figure 43 • **Write/Read Transactions**



4. Analyze the read and write transactions on the **Wave** window by expanding the required signals.

5. The simulation results can also be seen on the **Transcript** window of ModelSim, as shown in Figure 44.

Figure 44 • **Transcript Window**

```
Transcript
# BFM:32793:wait 100  starting at 113070 ns
# BFM:32794:endloop (Next Loop=110)
# BFM:32793:wait 100  starting at 114070 ns
# BFM:32794:endloop (Finished)
# BFM:32796:write w a0000000 a1b2c3d4 at 115070 ns
# BFM:32797:write w a0000004 10100101 at 115080 ns
# BFM:32798:write w a0000008 a5def6e7 at 115090 ns
# MONAXI1:######################################################################
# MONAXI1: WCMD ID:0 ADDR:ffffffff BURST:0 LEN:0 SIZE:0 RMW:0 LOCK:0 at  115200.0 ns
# BFM: Data Write a0000000 a1b2c3d4
# BFM:32799:write w a000000c d7d7e1e1 at 115430 ns
# MONAXI1:######################################################################
# MONAXI1: WCMD ID:0 ADDR:ffffffff BURST:0 LEN:0 SIZE:0 RMW:0 LOCK:0 at  115540.0 ns
# MONAXI1: WDATA ID:0 DATA:0000000000000000 STBS:00000000 LAST:0 at  115720.0 ns
# BFM: Data Write a0000004 10100101
# BFM:32800:readcheck w a0000000 a1b2c3d4 at 115750 ns
# BFM: Data Write a0000008 a5def6e7
# BFM:32801:readcheck w a0000004 10100101 at 116070 ns
# BFM: Data Write a000000c d7d7e1e1
# BFM:32802:readcheck w a0000008 a5def6e7 at 116390 ns
# MONAXI1:######################################################################
# MONAXI1: RCMD 1 ID:0 SEQ:1 ADDR:ffffffff BURST:0 LEN:0 SIZE:0 LOCK:0 at  116670.0 ns
# BFM: Data Read a0000000 a1b2c3d4 MASK:ffffffff at 116690.010000ns
# BFM:32803:readcheck w a000000c d7d7e1e1 at 116700 ns
# MONAXI1:######################################################################
# MONAXI1: RCMD 2 ID:0 SEQ:2 ADDR:ffffffff BURST:0 LEN:0 SIZE:0 LOCK:0 at  116980.0 ns
# MONAXI1: RDAT Start ID:0 SEQ:1 - matching RCMD:1 ADDR:ffffffff BURST:0 LEN:0 SIZE:0 at  116670.0 ns
# MONAXI1: RDAT ID:0 0000000000000000 at  116980.0 ns
# MONAXI1: QUEUED 1 per ID = 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 CMDS= 2 0 0 0
# BFM: Data Read a0000004 10100101 MASK:ffffffff at 117000.010000ns
# BFM:M_DDR0_CTRL_REGS TEST ENDS
# BFM:
# BFM:32809:return
# BFM:24:return
# MONAXI1:######################################################################
# MONAXI1: RCMD 3 ID:0 SEQ:3 ADDR:ffffffff BURST:0 LEN:0 SIZE:0 LOCK:0 at  117290.0 ns
# MONAXI1: RDAT Start ID:0 SEQ:2 - matching RCMD:2 ADDR:ffffffff BURST:0 LEN:0 SIZE:0 at  116980.0 ns
# MONAXI1: RDAT ID:0 0000000000000000 at  117290.0 ns
# MONAXI1: QUEUED 1 per ID = 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 CMDS= 3 0 0 0
# BFM: Data Read a0000008 a5def6e7 MASK:ffffffff at 117310.010000ns
# MONAXI1:######################################################################
# MONAXI1: RCMD 4 ID:0 SEQ:4 ADDR:ffffffff BURST:0 LEN:0 SIZE:0 LOCK:0 at  117600.0 ns
# MONAXI1: RDAT Start ID:0 SEQ:3 - matching RCMD:3 ADDR:ffffffff BURST:0 LEN:0 SIZE:0 at  117290.0 ns
# MONAXI1: RDAT ID:0 0000000000000000 at  117600.0 ns
# MONAXI1: QUEUED 1 per ID = 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 CMDS= 4 0 0 0
# BFM: Data Read a000000c d7d7e1e1 MASK:ffffffff at 117620.010000ns
###########################################################
#
# BFM Simulation Complete - 238 Instructions - NO ERRORS
#
###########################################################
#
# MONAXI1: No response activity while read data requests queued
# MONAXI1: Outstanding CMD 4 ID:0 SEQ:4  ADDR:ffffffff BURST:0 LEN:0 SIZE:0 at  117600.0 ns

VSIM 2>
```

The following message is displayed in the Transcript window:

```
# BFM: Data Read a0000000 a1b2c3d4 MASK:ffffffff at 116690.010000ns
# BFM: Data Read a0000004 10100101 MASK:ffffffff at 117000.010000ns
# BFM: Data Read a0000008 a5def6e7 MASK:ffffffff at 117310.010000ns
# BFM: Data Read a000000c d7d7e1e1 MASK:ffffffff at 117620.010000ns
```

In the BFM script provided in the user.bfm file earlier, the readcheck command reads the data from the AXI bus and verifies whether the data read matches with the value provided with the readcheck command. If the value read does not match, the simulation results in an error.

6. Go to **File > Quit**.

2.5 Conclusion

In this tutorial, a new project is created in Libero SoC, the MSS component is configured to access an external SDR SDRAM memory through the fabric, the CoreSDR_AXI IP is added and configured in the fabric, and the IP is connected to the MSS component. The fabric and MSS CCC blocks are configured to generate the clocks. The design in ModelSim using the AMBA AXI BFM simulation is also verified.

3 Revision History

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

Revision	Changes
Revision 11 (April 2016)	Updated the document for Libero SoC v11.7 software release (SAR 77066).
Revision 10 (October 2015)	Updated the document for Libero SoC 11.6 software release (SAR 72419).
Revision 9 (February 2015)	Updated the document for Libero SoC 11.5 software release (SAR 64191).
Revision 8 (September 2014)	Updated the document for Libero version 11.4 (SAR 60226).
Revision 7 (May 2014)	Updated the document for Libero version 11.3 (SAR 56971).
Revision 6 (November 2013)	Updated the document for Libero version 11.2 (SAR 52903).
Revision 5 (April 2013)	Updated the document for 11.0 production SW release (SAR 47102).
Revision 4 (March 2013)	Updated the document for Libero 11.0 Beta SP1 software release (SAR 44867).
Revision 3 (November 2012)	Updated the document for Libero 11.0 beta SPA software release (SAR 42845).
Revision 2 (October 2012)	Updated the document for Libero 11.0 beta launch (SAR 41584).
Revision 1 (May 2012)	Updated the document for LCP2 software release (SAR 38953).

4 Product Support

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

4.1 Customer Service

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

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

4.2 Customer Technical Support Center

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

4.3 Technical Support

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

4.4 Website

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

4.5 Contacting the Customer Technical Support Center

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

4.5.1 Email

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

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

4.5.2 My Cases

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

4.5.3 Outside the U.S.

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

4.6 ITAR Technical Support

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

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

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

50200311-11/4.16