SmartFusion2 - Accessing External SDRAM through Fabric - Libero SoC v11.4 Tutorial

Superseded

Microsemi

Table of Contents

Superseded

Abbreviations Used

- cSoC – Customizable system-on-chip
- MSS – Microcontroller subsystem
- SDR SDRAM – Single data rate synchronous dynamic Random Access Memory
- SMC_FIC – Soft Memory Controller – Fabric Interface Controller
- CCC – Clock conditioning circuits
- MSS CCC – CCC block inside the MSS component
- Fabric CCC – CCC block instantiated inside the FPGA fabric
- DDR – Double data rate memory controller
- MDDR – DDR controller inside the MSS component.
- BFM – Bus functional model

Superseded

Accessing External SDRAM through Fabric - Libero SoC v11.4

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 IP is used in SmartFusion®2 system-on-chip (SoC) field programmable gate array (FPGA) devices 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 to 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:

- Creating a Libero® System-on-Chip (SoC) v114 project using a SmartFusion2 SoC FPGA.
- Updating the IP catalog by downloading the latest versions of the IP cores.
- Configuring the various hardware blocks using SmartDesign.
- Configuring the MDDR and CCC blocks of the microcontroller subsystem (MSS) component.
- Generating the microcontroller subsystem (MSS) component.
- Integrating the various hardware blocks in SmartDesign and generating the final top-level component.
- Performing functional level verification of the design using AMBA AXI bus functional model (BFM) simulation in Mentor Graphics ModelSim® simulator.
- Using the ModelSim GUI to see the various design signals in the Waveform window of ModelSim.

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	11.4

Project Files

The project files associated with this tutorial can be downloaded from Microsemi® website:
http://soc.microsemi.com/download/rsc/?f=sf2_accessing_external_sdram_through_fabric__liberov11p4_tu_df

The project files associated with this tutorial include the following:

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

Superseded

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 go 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 MSS block of the SmartFusion2 SoC FPGA. The MSS contains another block called the DDR Bridge. This block is responsible for managing the read/write requests from the various masters to the DDR controller in the MSS, called the MDDR block, or interfacing with external bulk memories such as SDR SDRAM via 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 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 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 all 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 the 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.

Superseded

Figure 1 shows the top-level design.

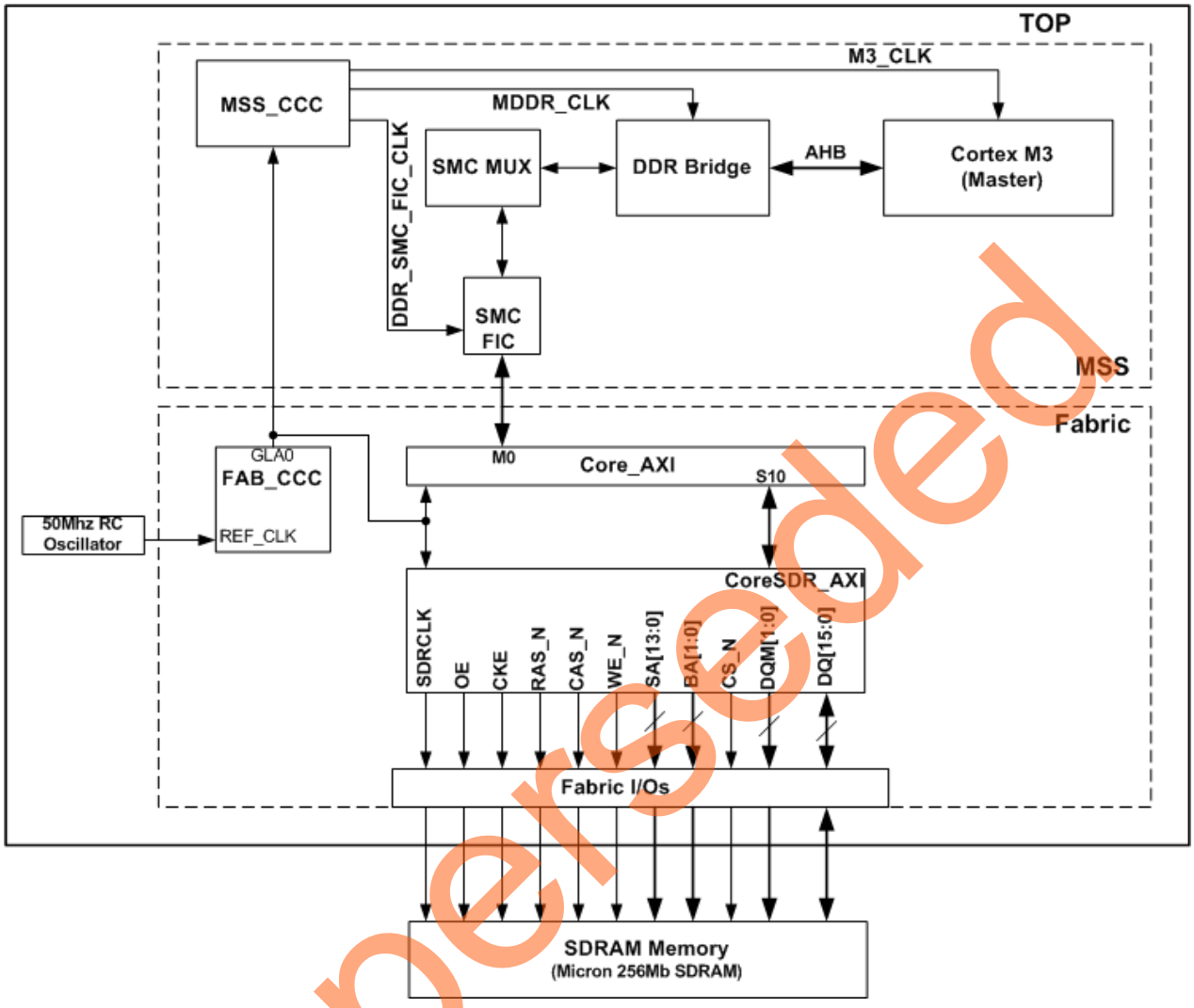


Figure 1. Top-Level Design

Superseded

Design Steps

Creating a Libero SoC Project

1. Launch Libero SoC v114.
2. From the **Project** menu, select **New Project**. Enter the information as displayed in Figure 3.
   - Name : Access_EXT_SDRAM
   - Location: Select an appropriate location (for example, C:/Microsemi_prj)
   - Preferred HDL Type: Verilog
   - Family: SmartFusion2
   - Die: M2S050T
   - Package: 896 FBGA
   - Speed: STD
   - Core Voltage (V): 1.2
   - Operating Conditions: COM
   - PLL Supply Voltage (V): 2.5
3. Select the MSS core in the **Design Templates and Creators** section of the **New Project** window. If the selected MSS core version appears in italics, it means that the selected MSS Core is not available in the vault and it needs 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 2, it indicates that the MSS core is present in vault.
4. Click **OK** and close the **New Project** window.

   Note: The download process requires internet connection to the machine.

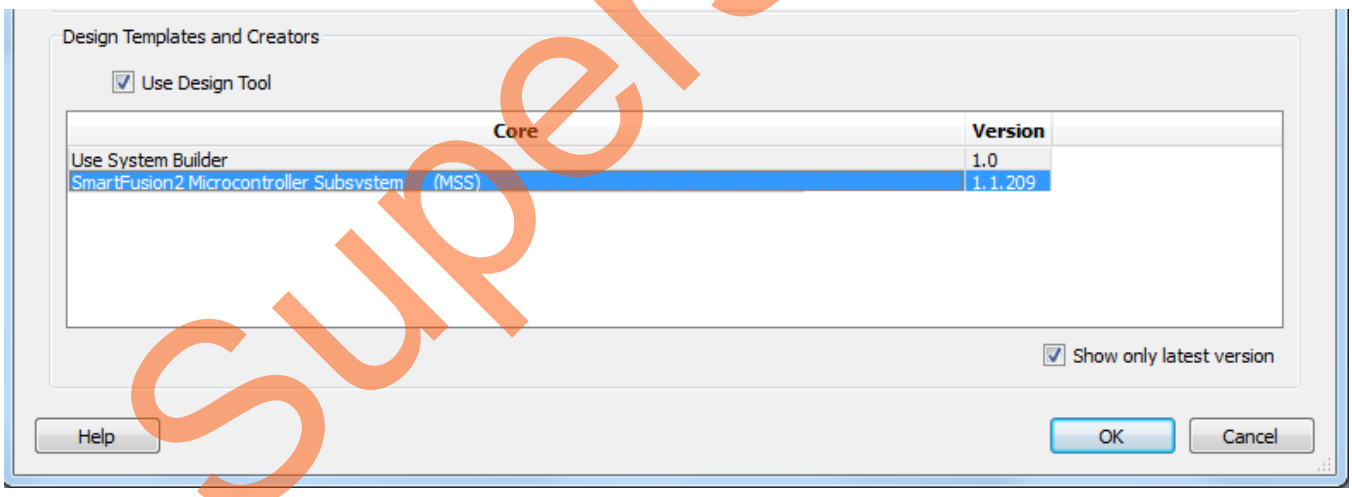


Figure 2. MSS Version

Superseded

Figure 3 shows the **New Project** window.

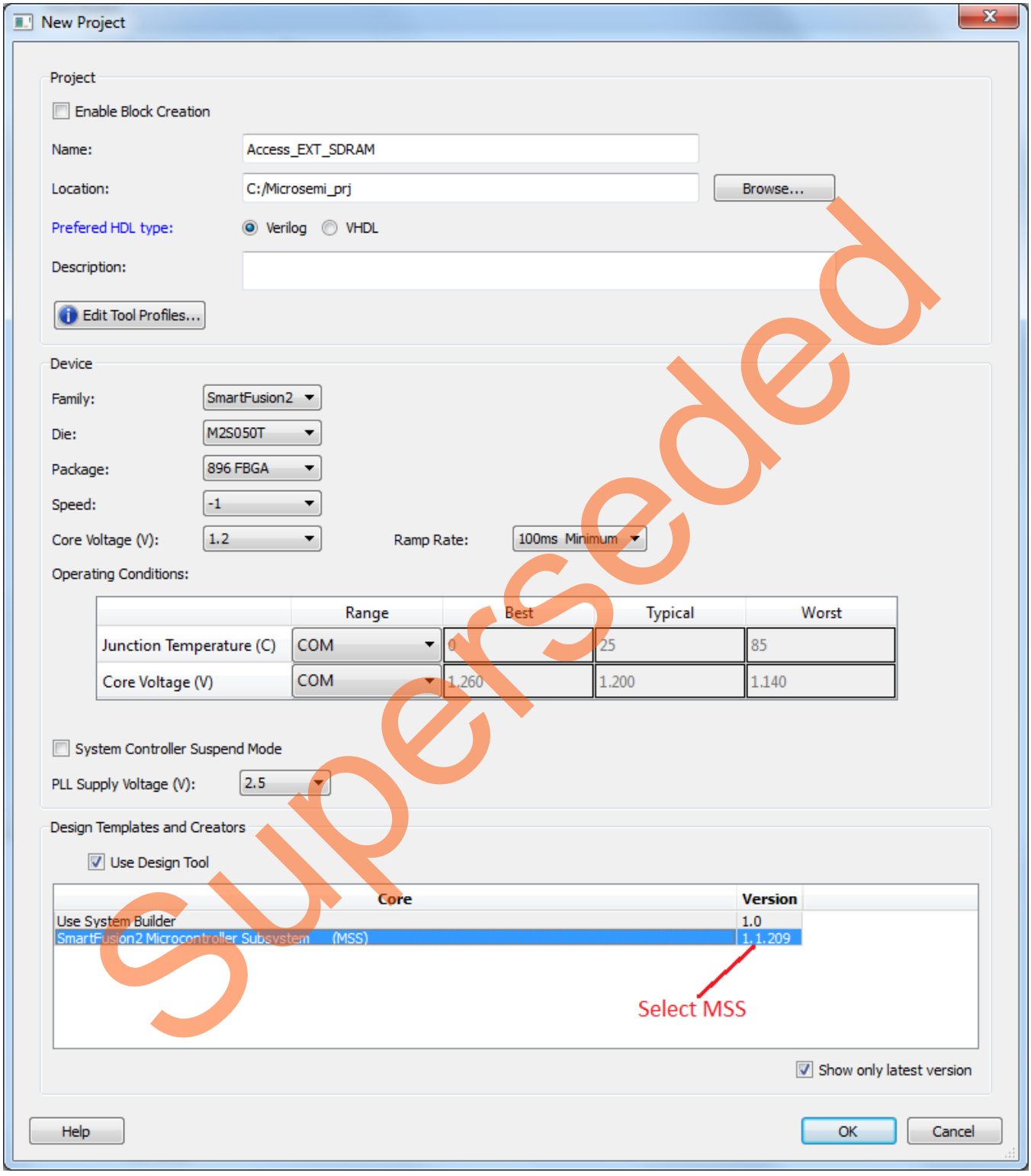


Figure 3. New Project Window

Updating IP Catalog

1. The project is created and the Libero SoC window is displayed as shown in Figure 4. The **SmartDesign** window opens and a project **Access_EXR_SDRAM** is created with the instantiation of the MSS component.

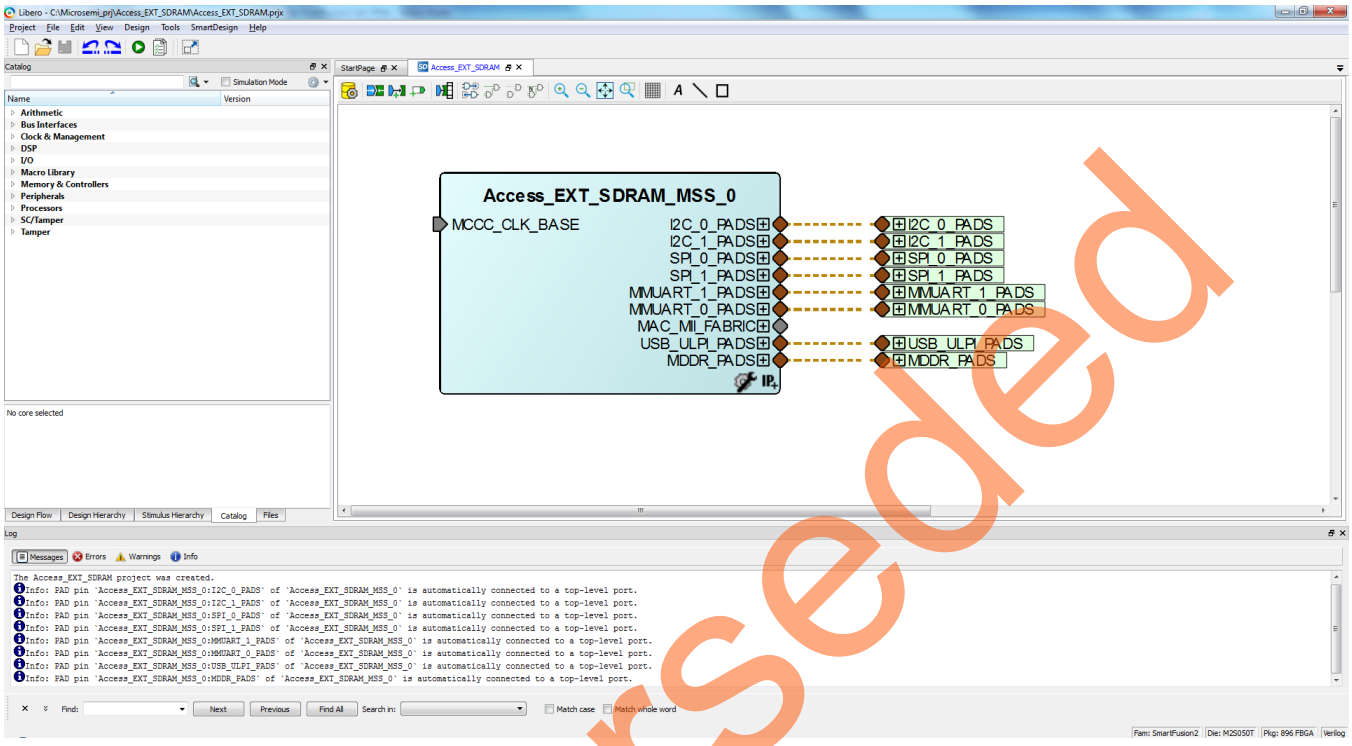


Figure 4. Libero Window on Completion of New Project Creation Wizard

Superseded

2. Click the **Catalog** tab, as shown in Figure 5. If a message is displayed "**New cores are available**", click **Download them now!**, and download the latest versions of the IP cores.

   Note: The download process requires internet connection to the machine.

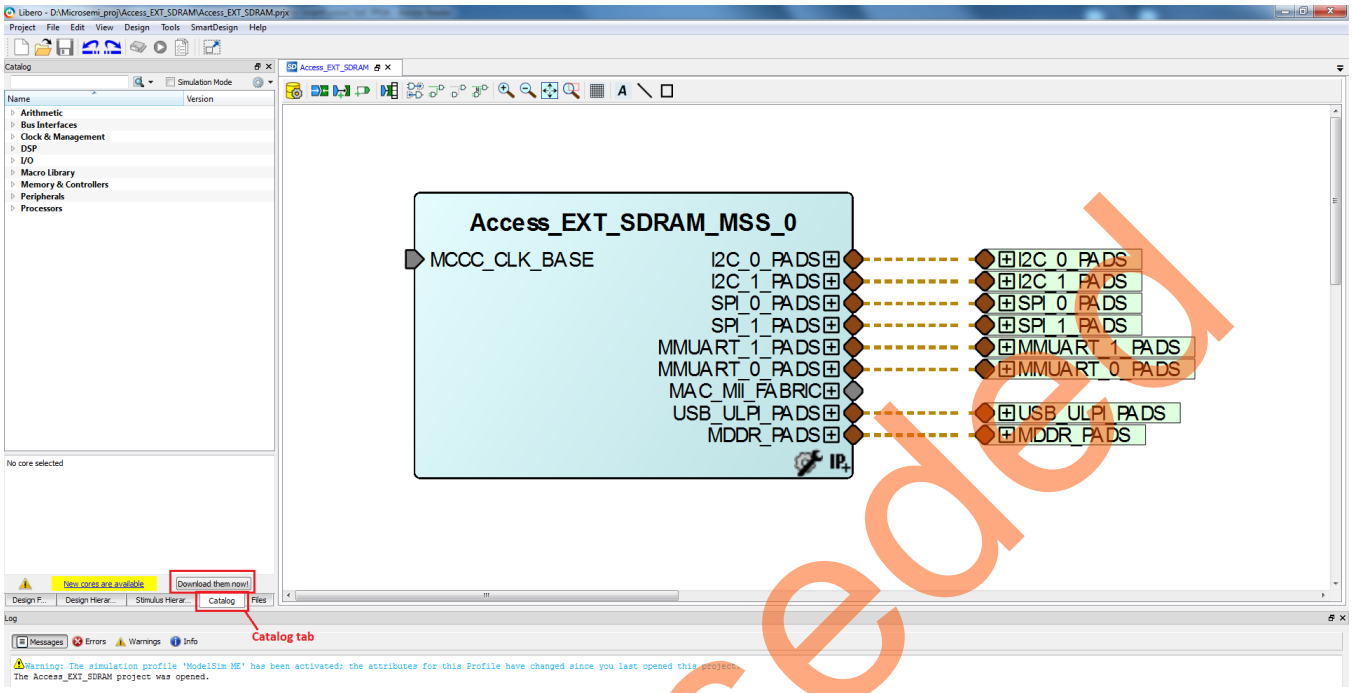


Figure 5. Updating the Catalog

Superseded

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 6.

   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 that is not required, right-click the peripheral block and clear the **Disable** check box, as shown in Figure 7, or clear the check box in the lower right corner of the peripheral box. The box turns gray to indicate that the peripheral has been disabled. Disabled peripherals can be enabled by selecting the check box in the lower right corner of the peripheral box (see Figure 8) or by right-clicking the peripheral block and selecting the **Enable** check box (see Figure 7).

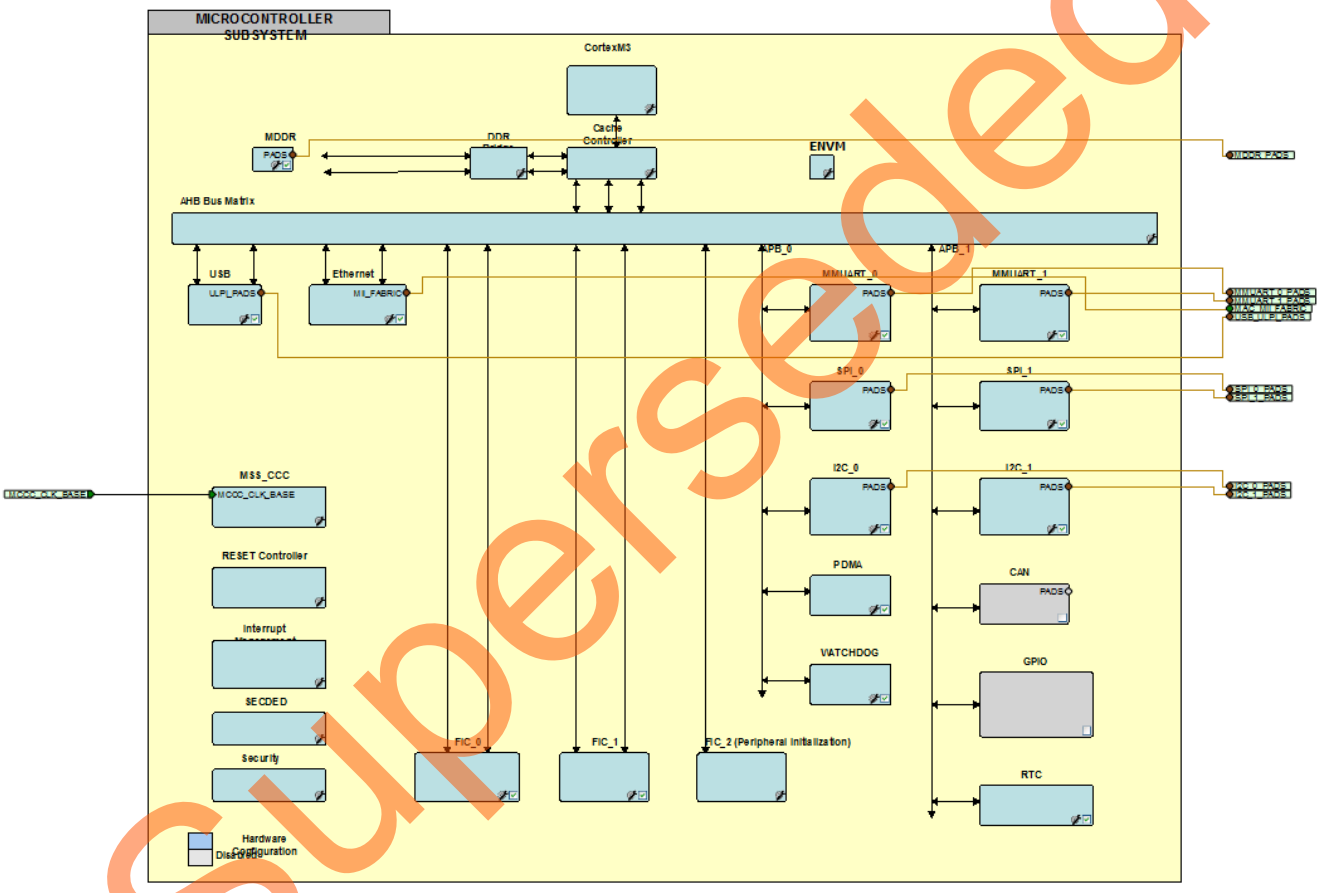


Figure 6. MSS in SmartDesign Canvas

Superseded

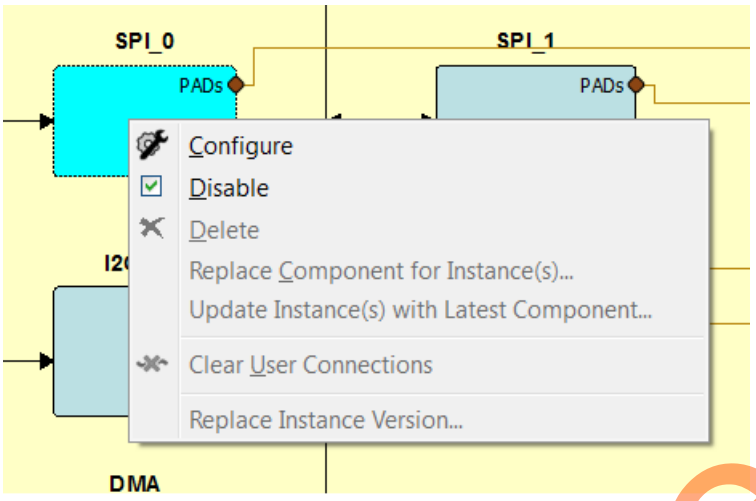


Figure 7. Right-click and Enable Peripheral Block

An enabled peripheral is shown in Figure 8.

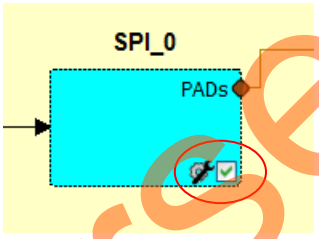


Figure 8. Enabling the Peripheral

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

Superseded

Figure 9 shows the **MSS Configuration** window (after disabling the above components).

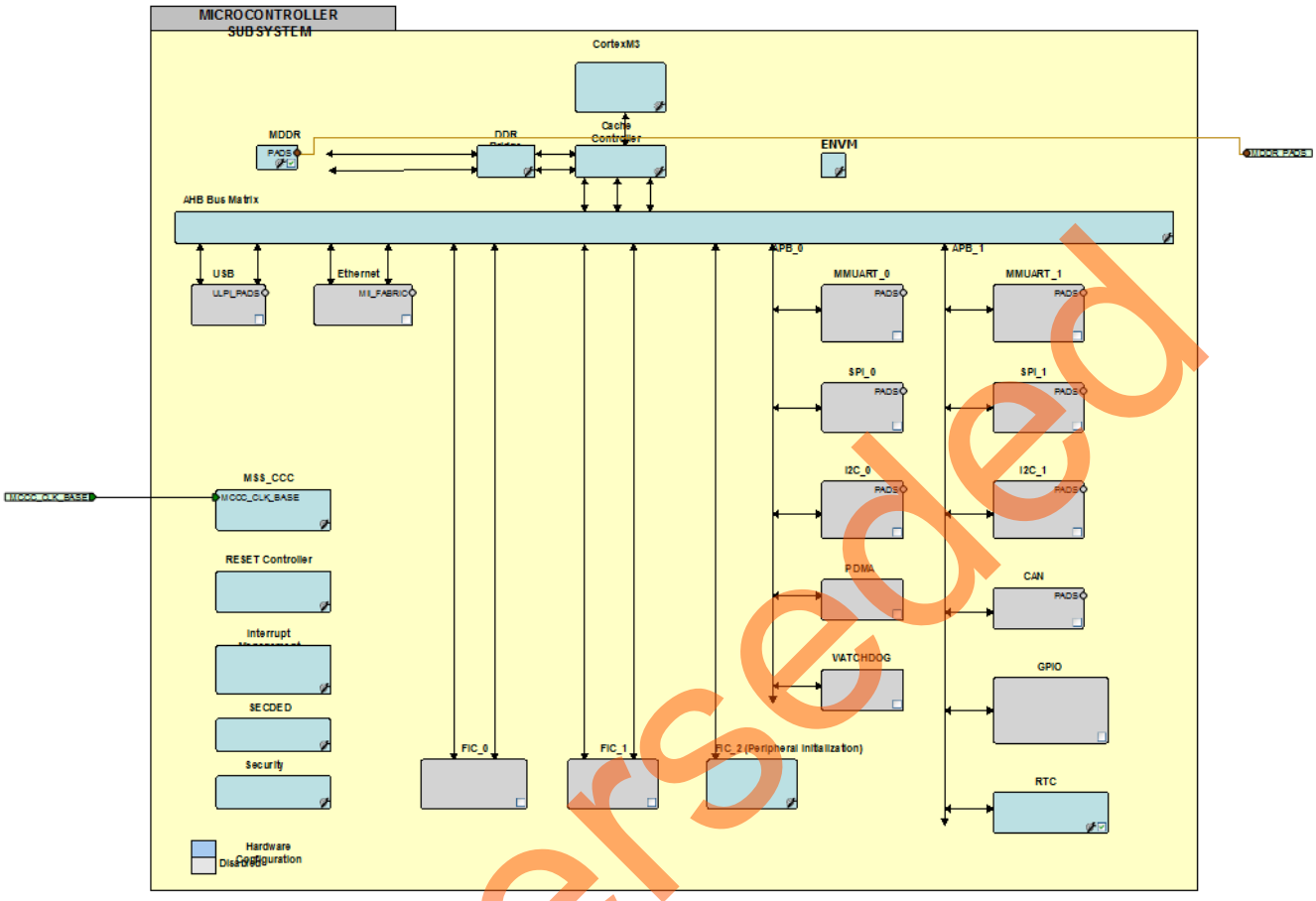


Figure 9. Enabled and Disabled MSS Components

Superseded

3. Double-click the **MDDR** block and configure as shown in Figure 10.
   - 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** and complete the configuration.

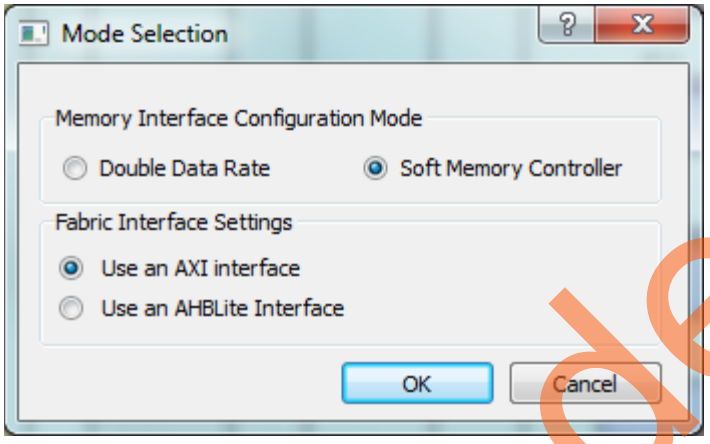


Figure 10. Mode Selection

4. Double-click the **MSS_CCC** block and configure as shown in Figure 11.
   - 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; i.e. **100 MHz**).
   - Leave the rest as default.
   - Click **OK** and complete the clock configuration.

   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.

Superseded

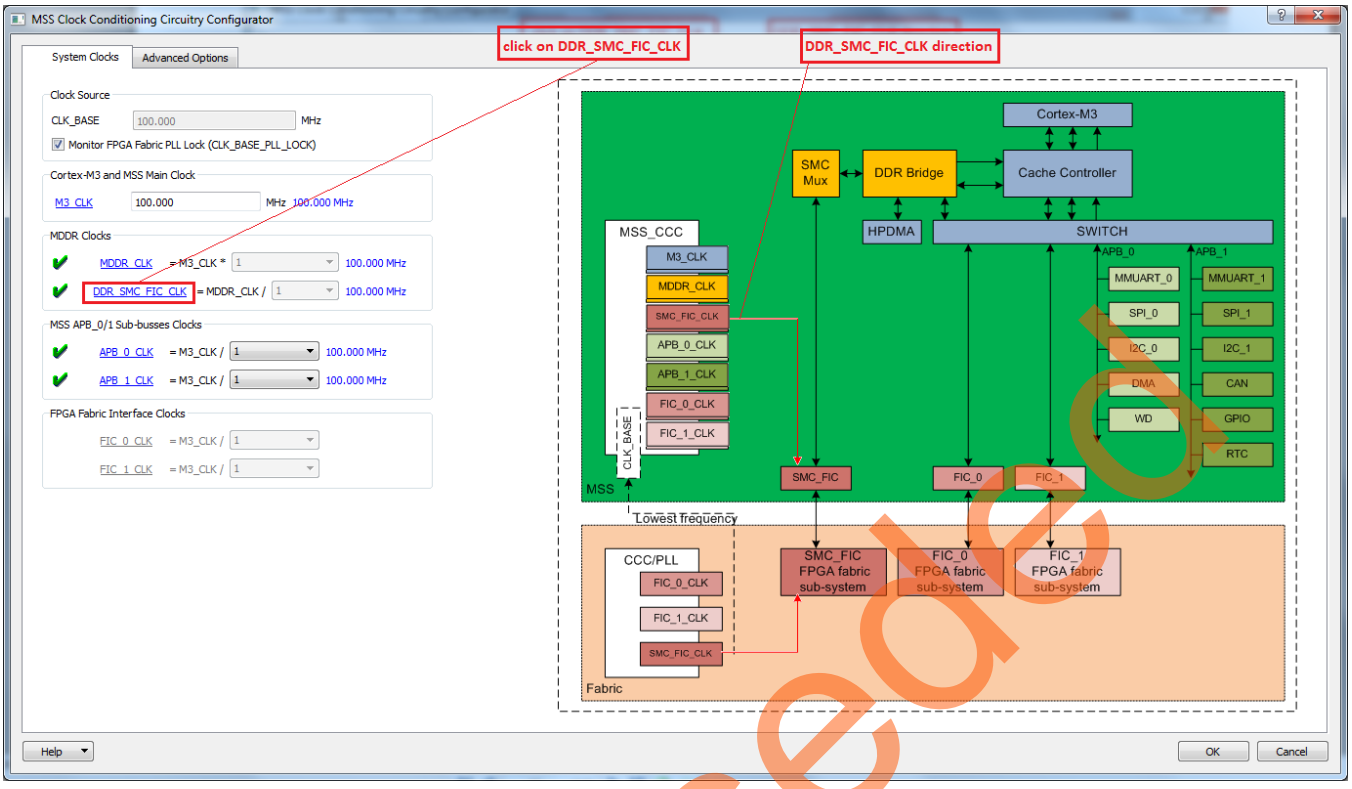


Figure 11. MSS Clock Configurator

5. Double-click the **Reset Controller** and select the **Enable MSS to Fabric Reset** and **Enable Fabric to MSS Reset,** as shown in Figure 12. This enables the MSS to generate the Reset signal for all the Fabric blocks. The MSS reset itself comes through a system reset pin on the Fabric I/O.
Click **OK**.

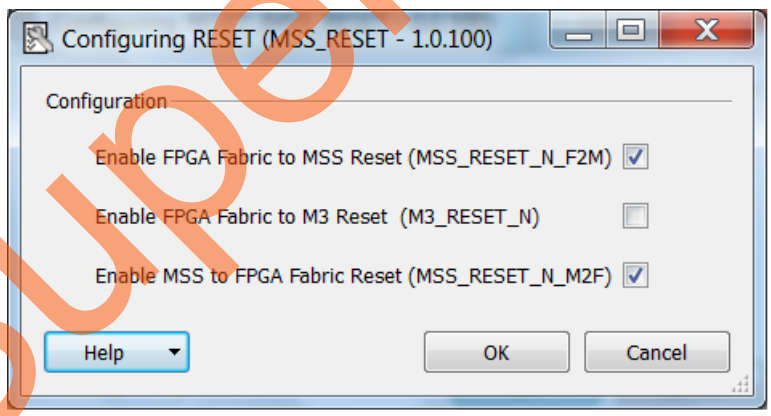


Figure 12. MSS RESET Configurator

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

Superseded

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 13.

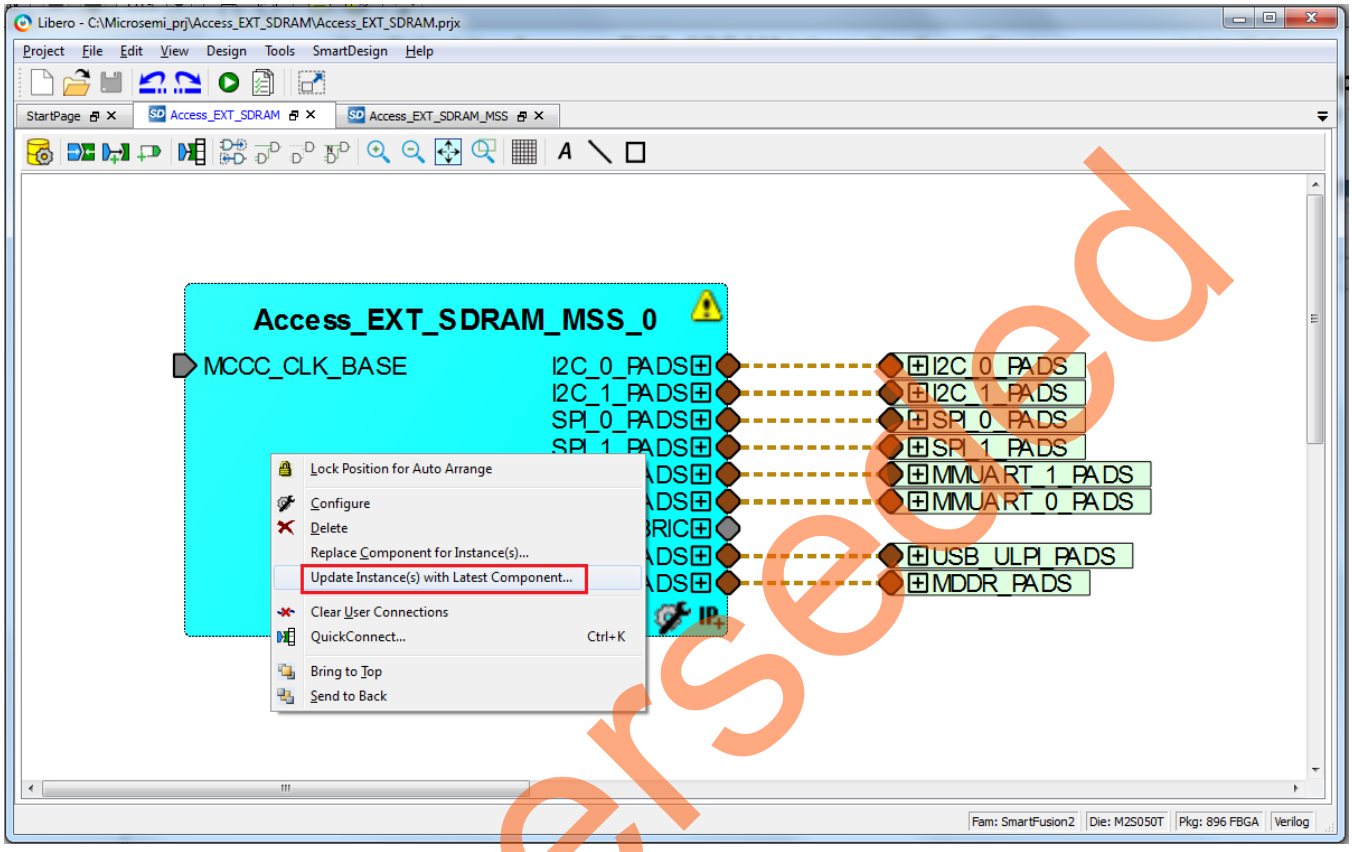


Figure 13. Updating the MSS

The **Access_EXT_SDRAM_MSS_0** instance (after successful updating) is shown in Figure 14.

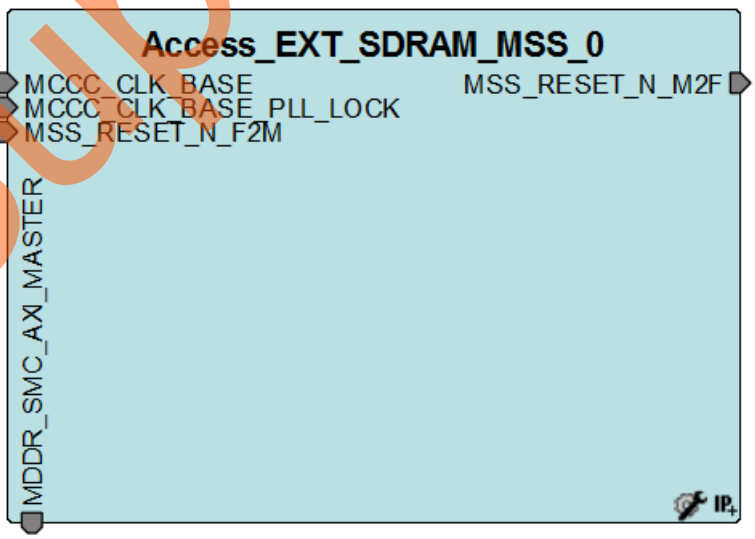


Figure 14. Updated MSS Instance

Superseded

Configuring Fabric Components

1. Select the **CoreAXI** IP from the Bus Interface sub-section of the IP Catalog, as shown in Figure 15, and drag it onto the **Access_EXT_SDRAM SmartDesign** tab.

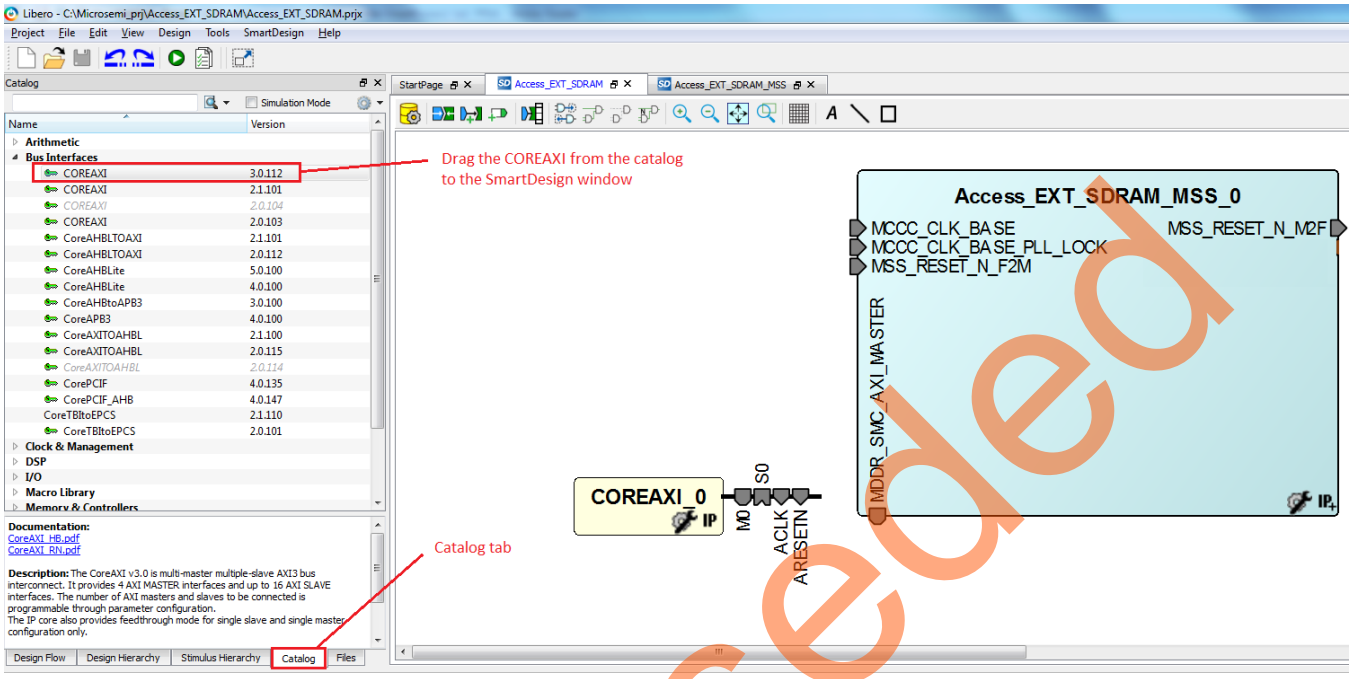


Figure 15. CoreAXI IP from the Catalog

2. Double-click the **COREAXI_0** instance on the SmartDesign pane to open its configuration window. Configure the core, as shown in Figure 16.
   - Leave the **Memory Space** field as **16, 256 MB slots beginning at address** 0x00000000 (which is default), as shown in Figure 16.
   - Leave the **AXI Data Width** field as **64** (which is default), as shown in Figure 16.
   - Leave the **Number of Master slots field** as **1**.
   - Clear the option of **SLAVE0 for Enable Master Access**.
   - Select the option **of SLAVE10 for Enable Master Access**.
   - Leave the rest as default.
   - Click **OK** in the configuration window to complete the configuration.

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

Superseded

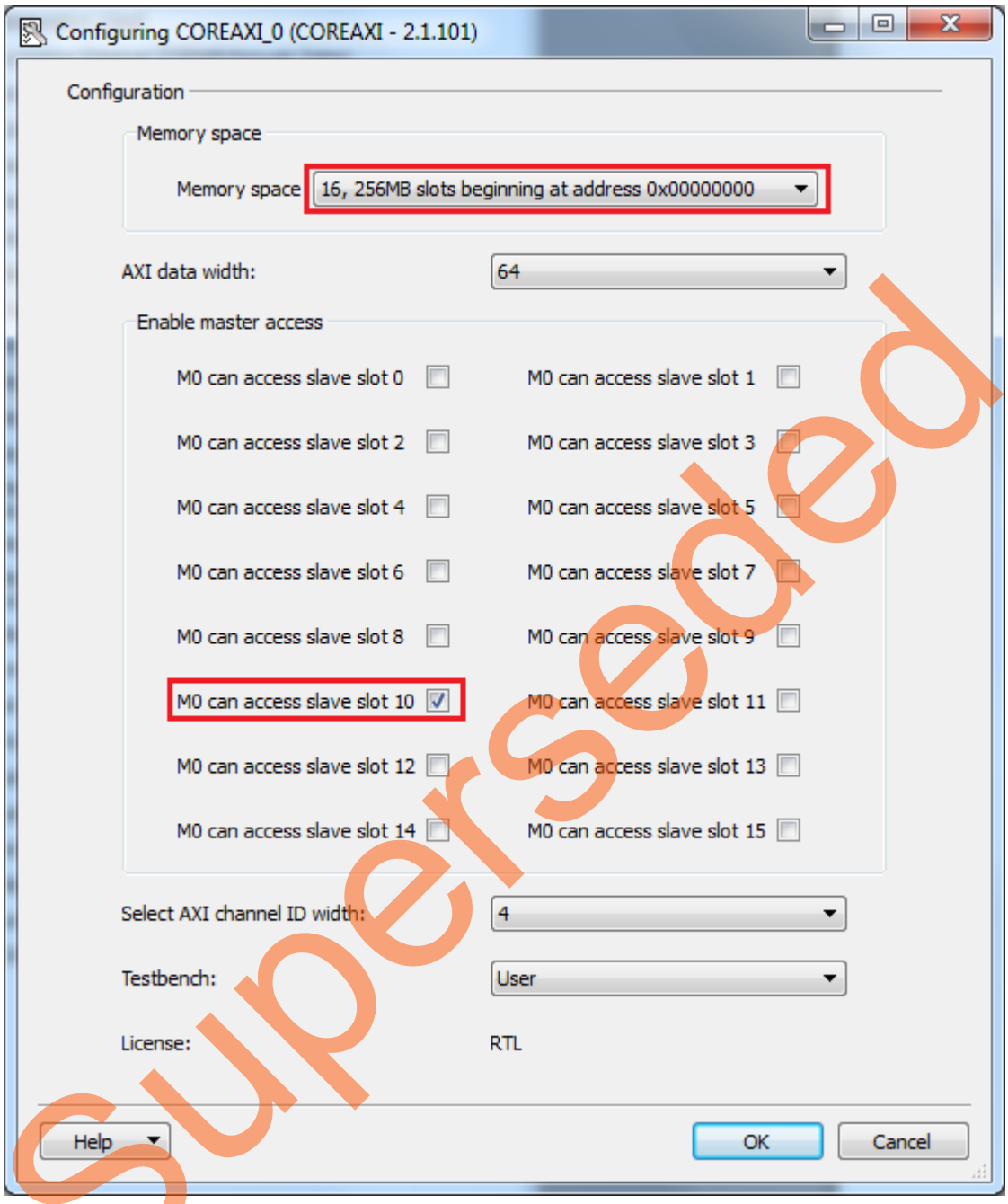
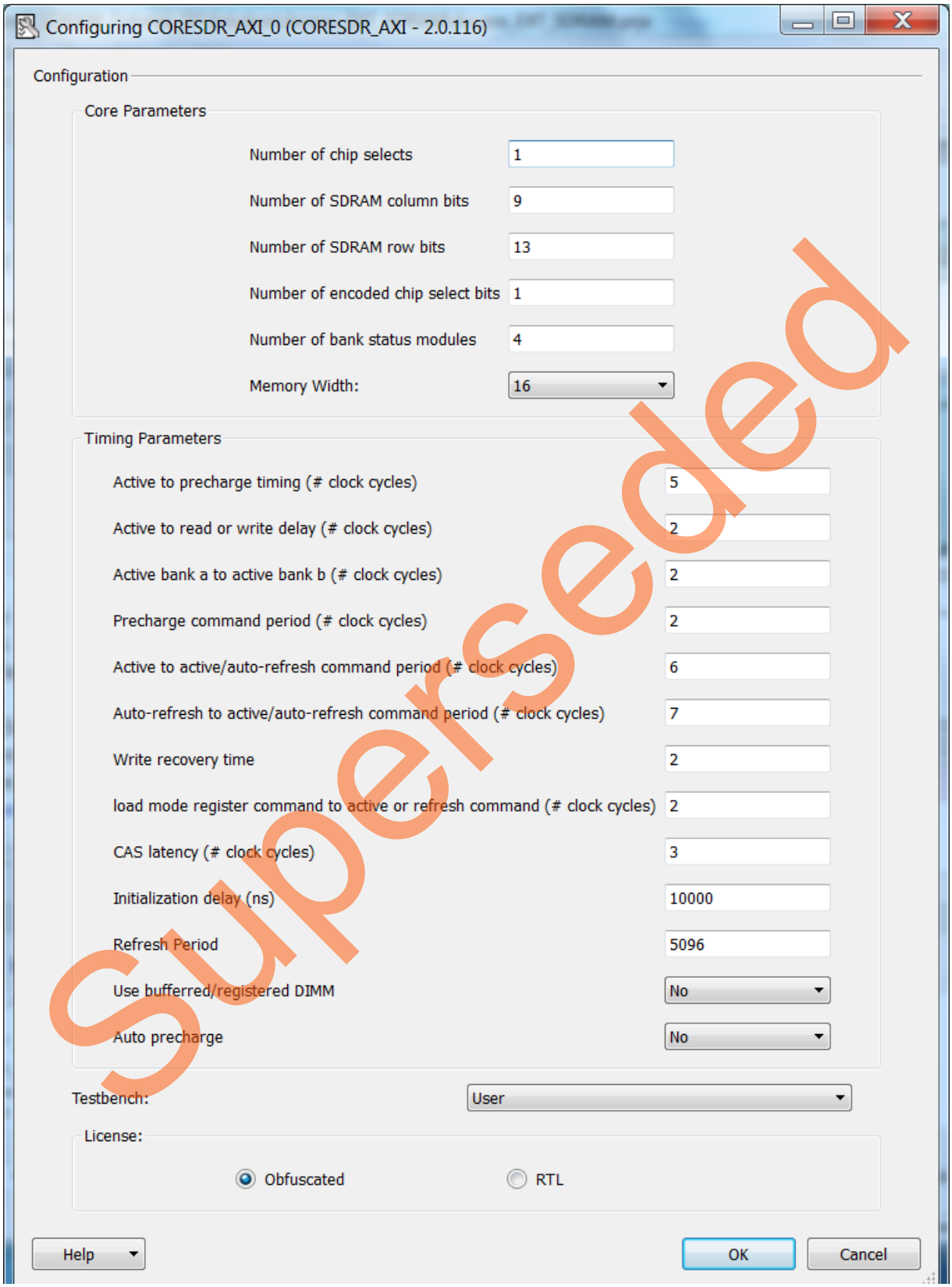


Figure 16. CoreAXI Configurator

3. Drag the **CoreSDR_AXI** IP from the **Peripherals** sub-section of the IP **Catalog**. Double-click on the **CORESDR_AXI_0** instance to access its configuration window. Enter the details in the configuration window, as shown in Figure 17. 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.

Superseded



Superseded

Figure 17. CoreSDR_AXI Configuration Window

4. Drag the clock conditioning circuitry (CCC) block from the Clock & Management sub-section of the IP Catalog. Double-click the F**CCC_0** instance to open up its configuration window.
   Configure the following items on the configuration window:
   - Select the **Advanced** tab as shown in Figure 18.
   - Select the clock source as **Oscillators > 25/50 MHz Oscillator**, as shown in Figure 19.
   - Leave the output frequency as **100 MHz**.
   - Leave the rest as default.
   - Select the **PLL Options** tab and select **Expose PLL_ARST_N and PLL_POWERDOWN_N** check box, as shown in Figure 20.
   - Click **OK** to complete the configuration.

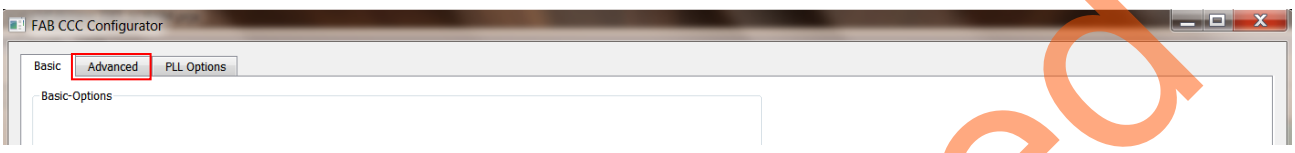


Figure 18. Advanced Tab of the FAB CCC Configurator

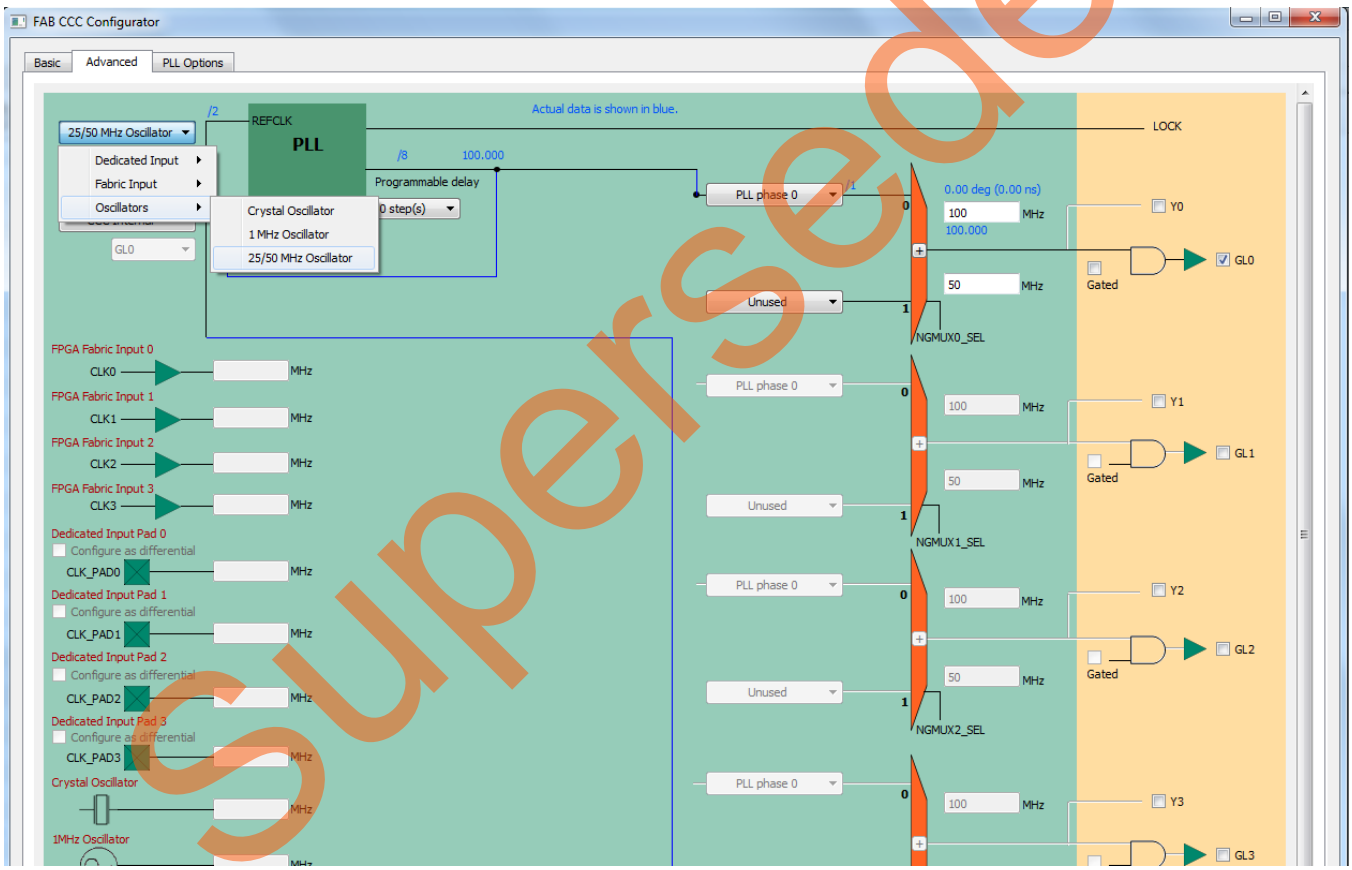


Figure 19. Selecting Clock Source

Superseded

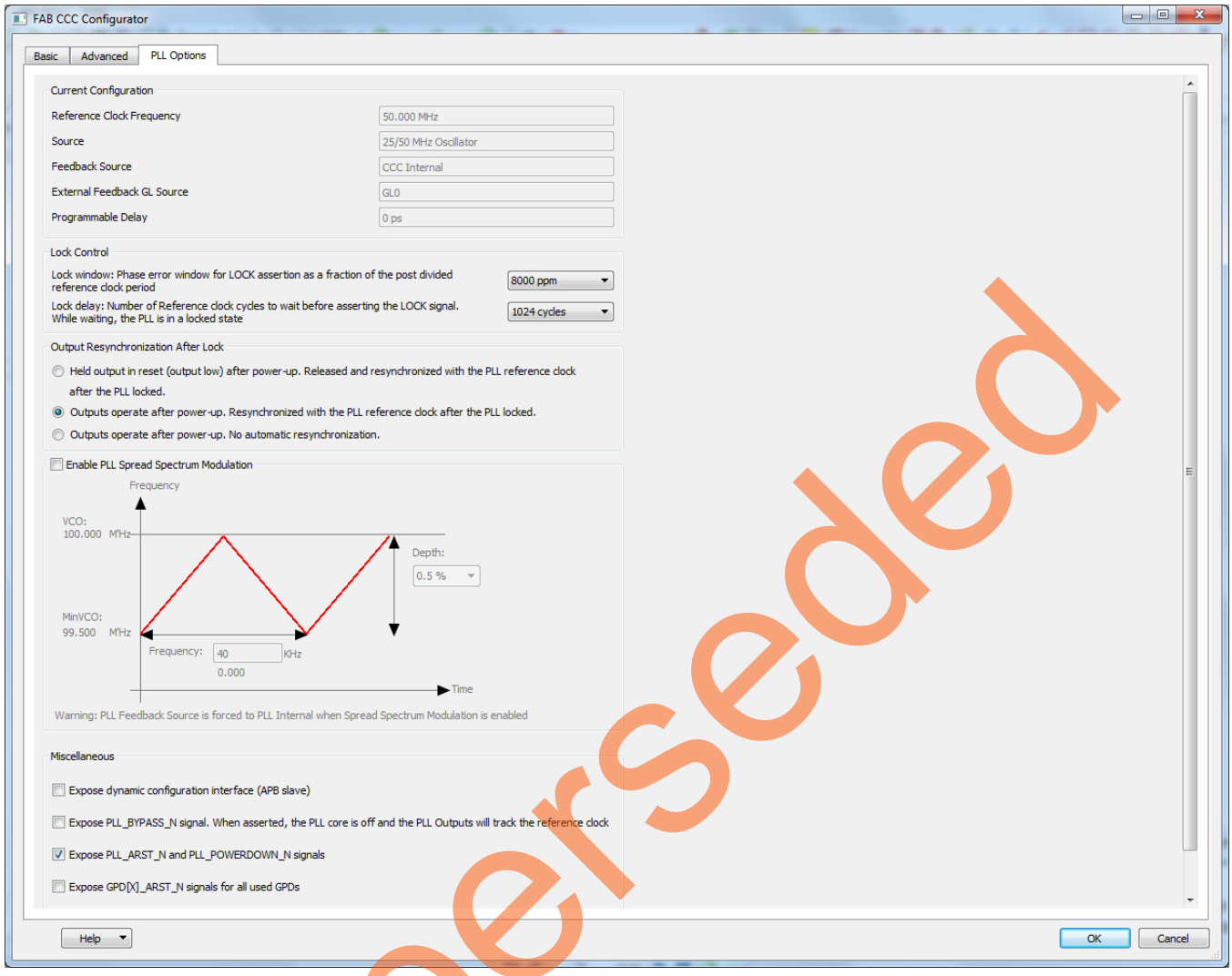


Figure 20. Exposing PLL Reset and Power-down Signals

5. Drag the **Chip Oscillators** IP from the Clock & Management sub-section of the IP Catalog into the SmartDesign. Double-click the **OSC_0** instance to open up its configuration window. Select the following as shown in Figure 21:
   - Select the check box **On Chip 25/50 MHz RC Oscillator**.
   - Clear the **Drives MSS** check box.
   - Select the check box for **Drives Fabric CCC(s)**.
   - Leave the rest as default.
   - Click **OK** to complete the configuration.

   With this, the On-chip 50 MHz RC oscillator has been selected to drive the input of the fabric CCC block instantiated earlier.

Superseded

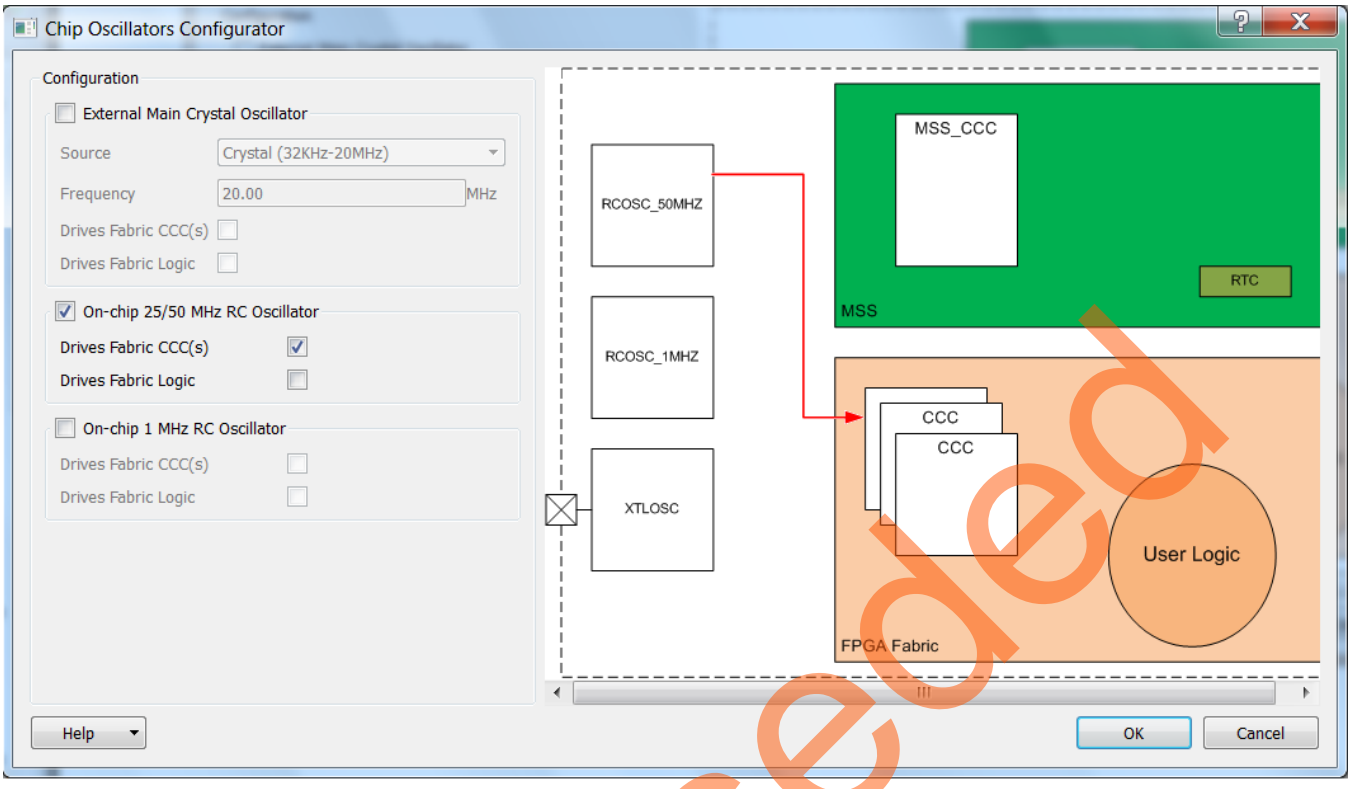


Figure 21. Oscillator Configuration

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

Interconnecting All Components

1. After re-arranging all the components on the **SmartDesign** window, connect the pins of all the blocks as described below.
2. Use **Auto Arrange Instances** on the SmartDesign canvas to arrange the various instances, automatically. There are two ways to make the connections:
   - The first method is by using the Connection Mode option. Change the SmartDesign to connection mode by clicking the **Connection Mode** button on the **SmartDesign** window, as shown in Figure 22. The cursor changes from the normal arrow shape to the connection mode icon shape. Click on the first pin and drag-drop 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 it and select **Disconnect** to disconnect the signals already connected.

Superseded

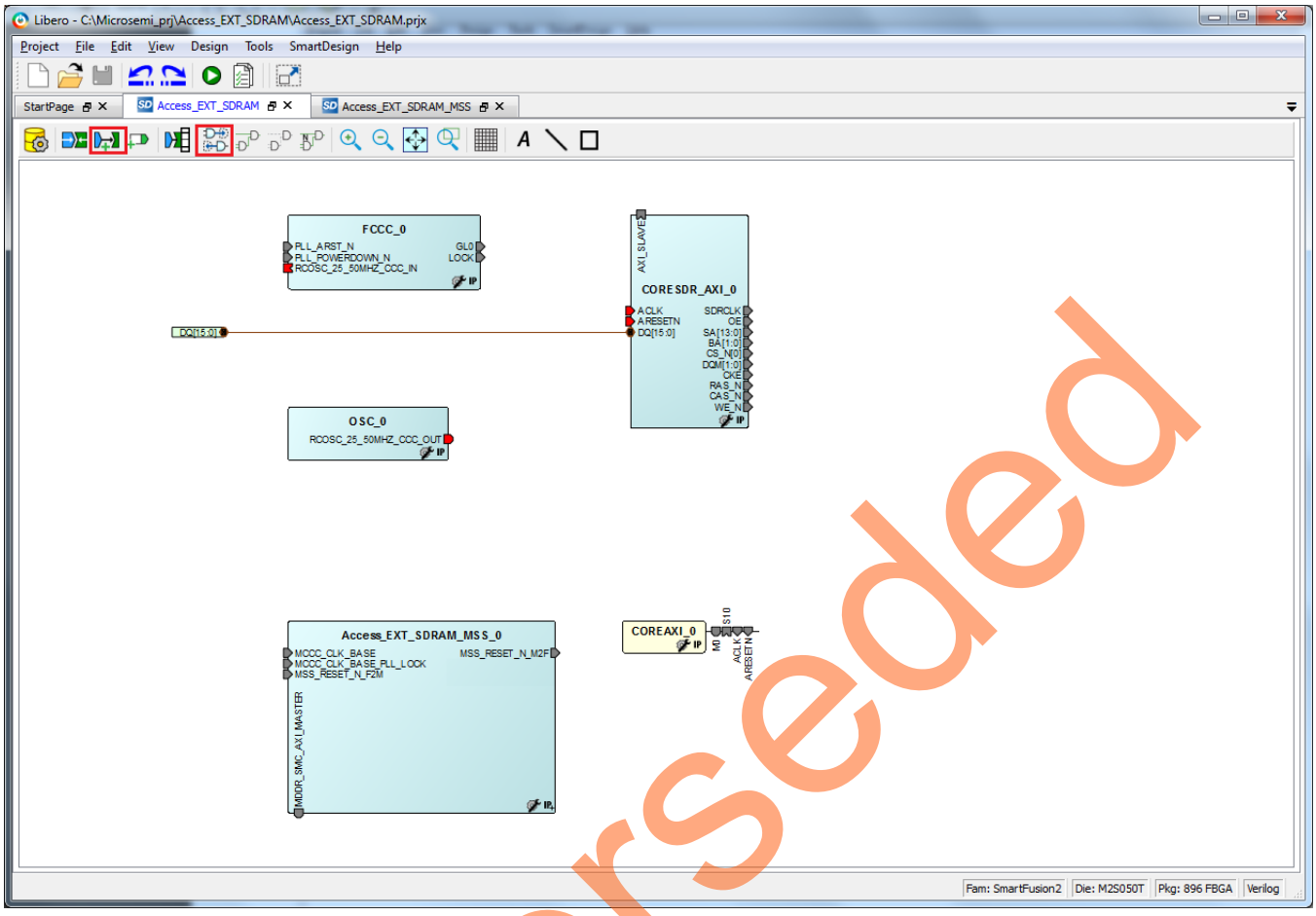


Figure 22. Changing to Connection Mode

3. Connect the following components as described below:
   - Connect **ROSC_25_50MHZ_CCC_OUT(M)** of the **OSC_0** to the **ROSC_25_50MHZ_CCC_IN(**S) of the F**CCC_0**.
   - Connect **GL0** of the F**CCC_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 input source clock for the MSS CCC block.
   - Connect **LOCK** of F**CCC_0** to **MCCC_CLK_BASE_PLL_LOCK** input of the **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 the **Access_EXT_SDRAM_MSS_0**.
   - Connect **S10** of **COREAXI_0** to **AXI_Slave** of **CORESDR_AXI_0**.
   - Connect **PLL_POWERDOWN_N** inputs of F**CCC_0** to logic '1'. Select each input signal, right-click the 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, select the input signal, right-click it, 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 F**CCC_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 the **CORESDR_AXI_0** to the top level. Hold the **CTRL** key and select each of them, right-click and select **Promote to Top Level**.

Superseded

4. Click **Auto arrange instances** to arrange the instances, as shown in Figure 23. Save the design by selecting **File > Save**.

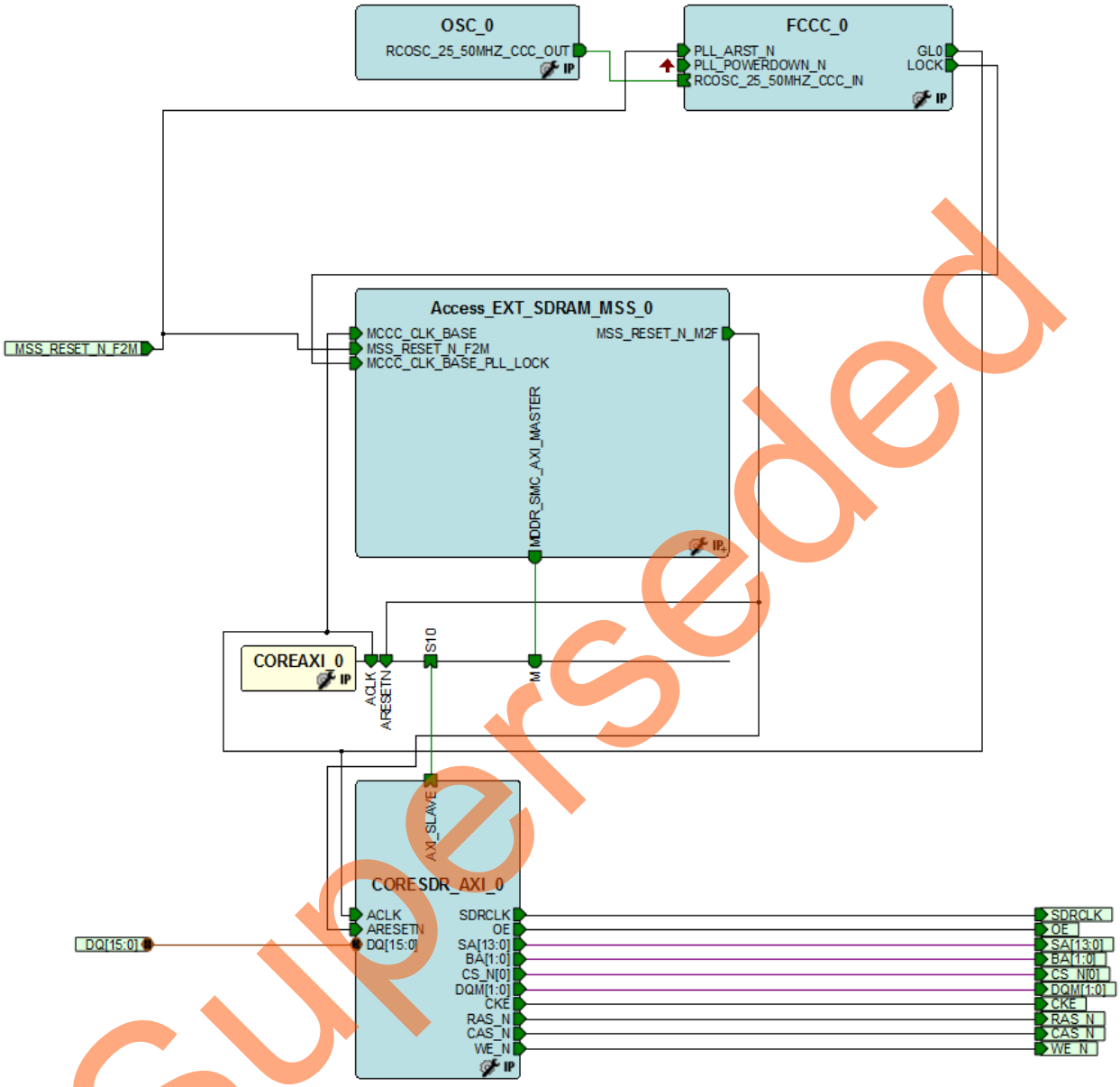


Figure 23. After Making the Top-Level Connection

Superseded

Generating MSS and Top-Level Design

1. Select **Access_EXT_SDRAM** tab on the **SmartDesign** canvas and click **Generate Component** on the SmartDesign pane (as shown in Figure 24) or select from **SmartDesign > Generate Component**.

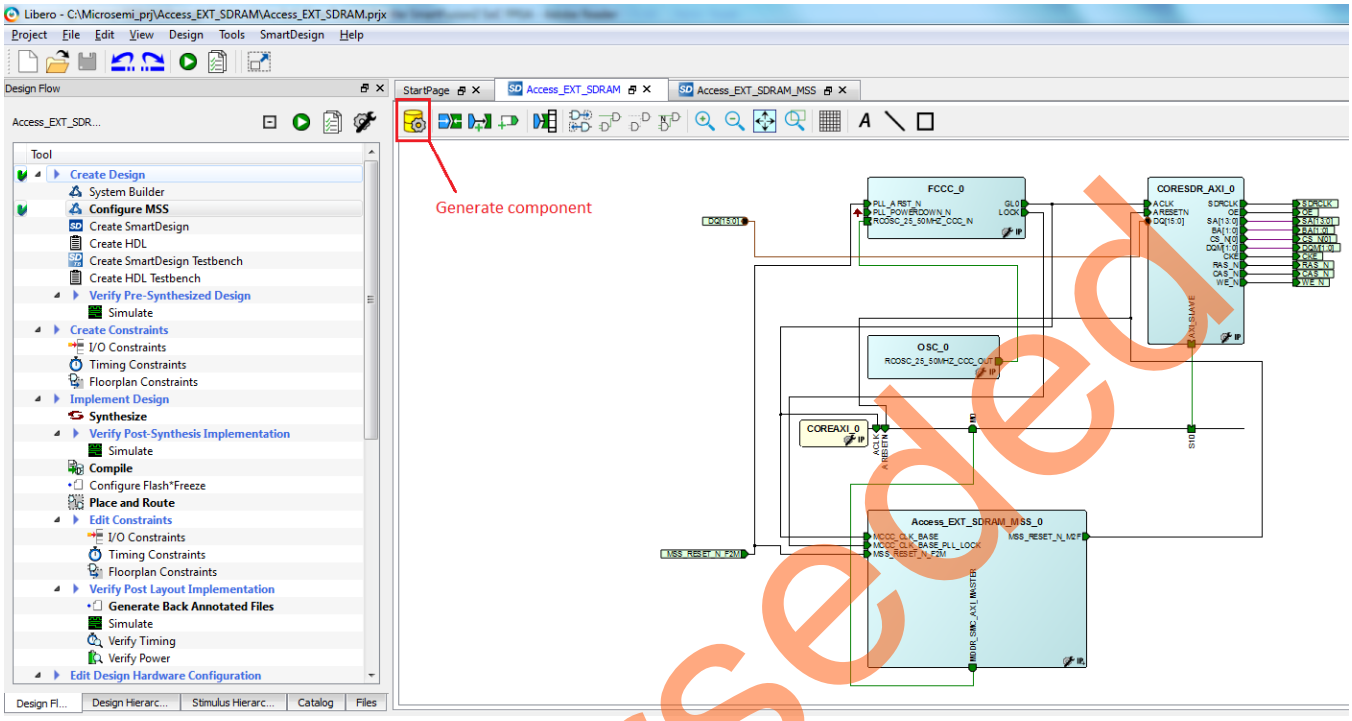


Figure 24. Generating MSS Component

2. 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

3. After generation, the design hierarchy can be found in the **Design Hierarchy** pane of the Libero SoC, as shown in Figure 25.

Superseded

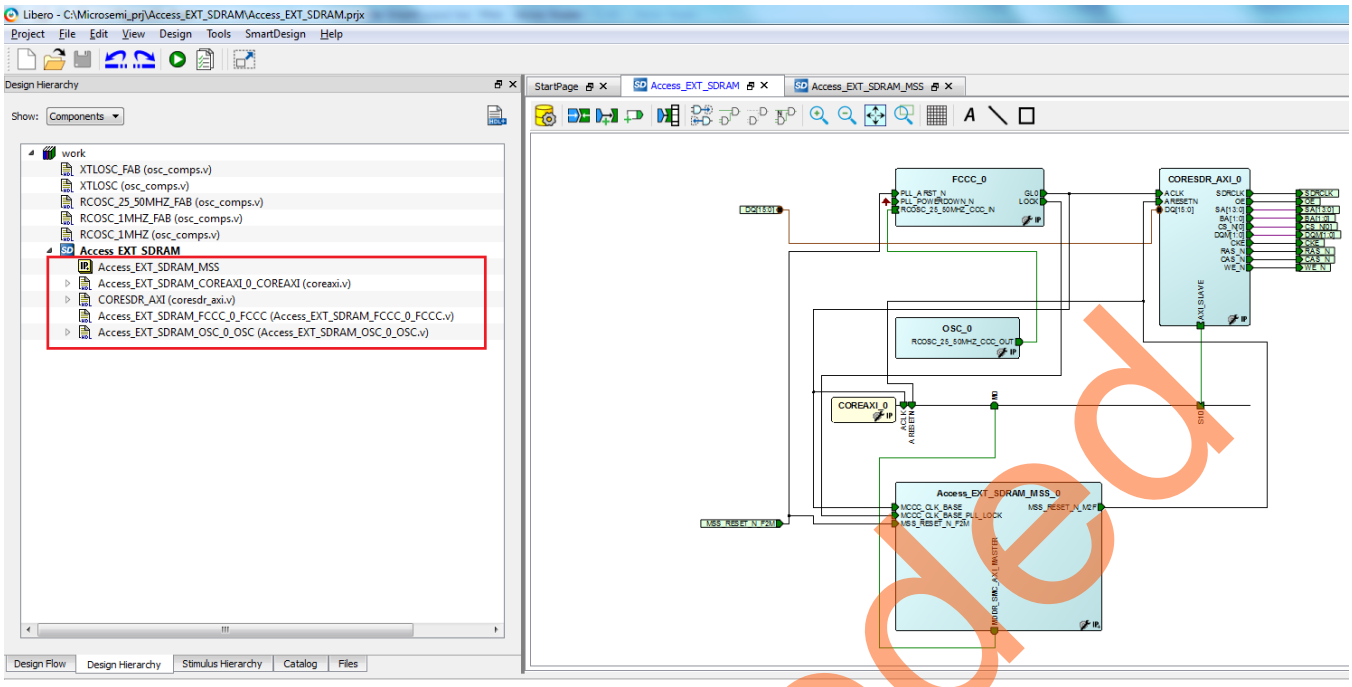


Figure 25. Design Hierarchy

4. After generation, you can see the Memory Map for the CORESDR_AXI_0 component. Right-click the **Access_EXT_SDRAM** SmartDesign window and select **Modify Memory Map**. Figure 26 shows the resultant memory map. The starting address of the MDDR Space 0 is 0xa0000000 in the Cortex-M3 processor address space.

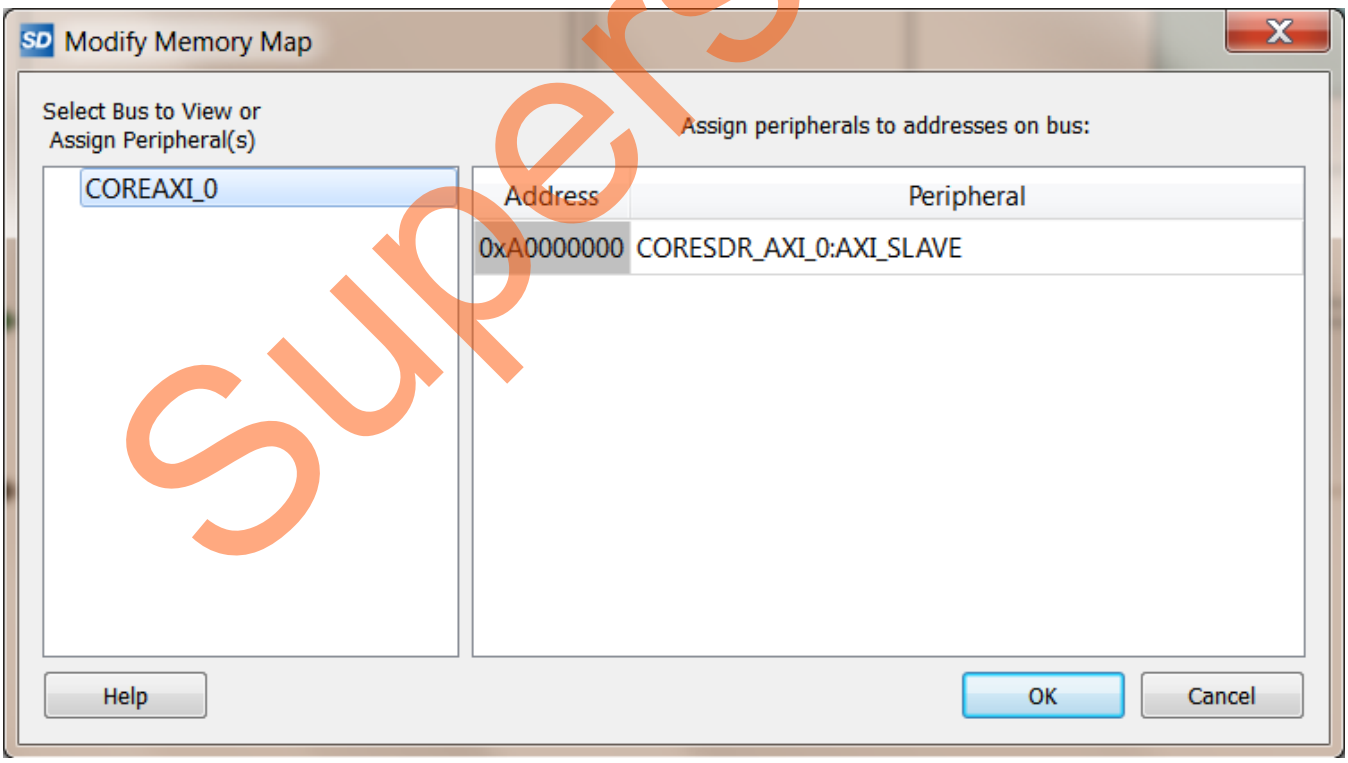


Figure 26. CORESDR_AXI_0 Memory Address

Superseded

Generating Testbench and Adding SDR SDRAM Simulation Model

1. In the **Design Hierarchy** tab, right-click on **Access_EXT_SDRAM > Create Testbench > HDL to generate the testbench**.

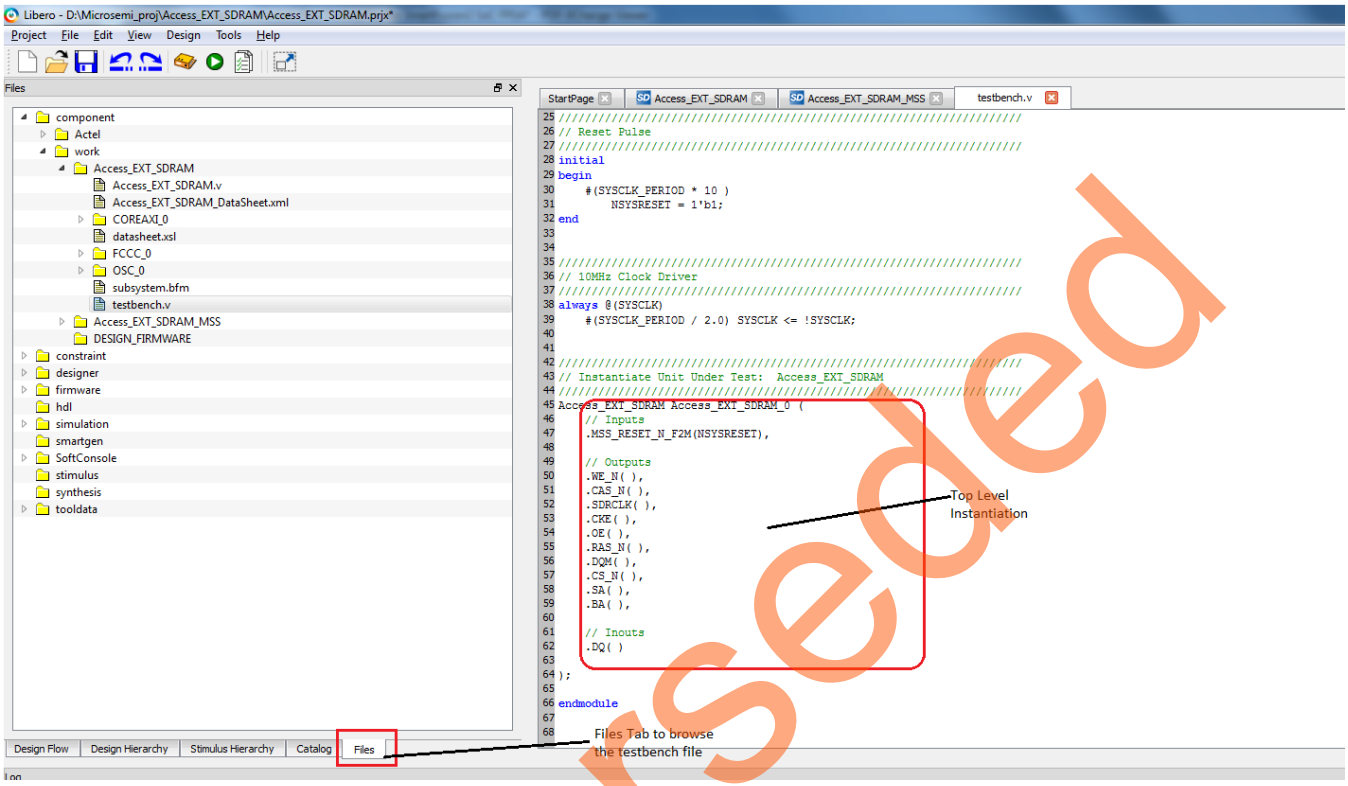


Figure 27. Default Testbench

2. Enter the name as **testbench** in the popup window and click **OK.**
3. In the generated testbench, the external SDR SDRAM simulation model needs to be added and port mapped with the top-level design SDRAM interface signals. Double-click ***testbench.v*** in the Files tab to open the file. Add the following lines of Verilog code in this testbench.

   At the top of the file, include the SDR SDRAM simulation file:

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

   Now 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;
```

Superseded

```
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 the **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;
```

Superseded

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*, import it as a stimulus file by selecting **File > Import Files**. In **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 28. The testbench.v file is shown under the Stimulus folder in the Files tab.

Superseded

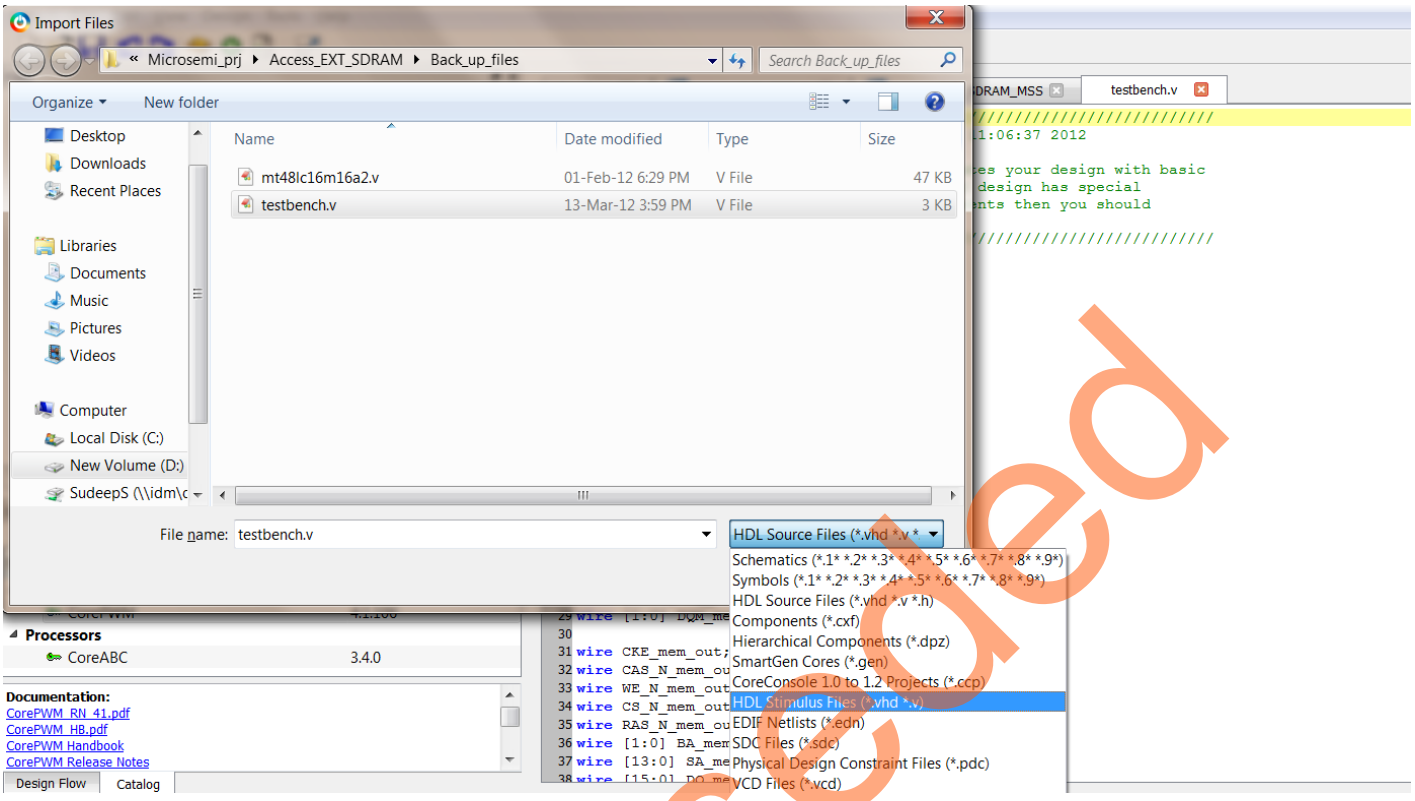


Figure 28. File Import to Stimulus Folder

4. 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 follow:

   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 now shows under the Stimulus folder in the Files tab.

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

Superseded

Adding BFM Commands to Perform Simulation

1. The user BFM commands are added in to a file named *user.bfm,* which can be found in the following location in the project:

   <Project_directory>\Access_EXT_SDRAM\simulation

   Browse the user.bfm 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 the "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 which are circled 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 ""
   ```

   Comment this line in the bfm file

   Save the *user.bfm* file after adding the above lines by selecting **File > Save**.

   Refer to the CoreAMBA BFM User's Guide for more information about the above BFM commands. www.microsemi.com/soc/ipdocs/CoreAMBA_BFM_UG.pdf

   Note: The sample user.bfm file can be found in the following location in the attached compressed project: <Project_directory>\ ACCESS_EXT_SDRAM\Source

Superseded

Setting up Simulation and Invoking Simulation Tool

1. The simulation tool must be set up before invoking so that it loads with the desired settings. Select **Project > Project Settings > Simulation Options > Do File**. Set **Simulation Runtime** to **158 us**, as shown in Figure 29.

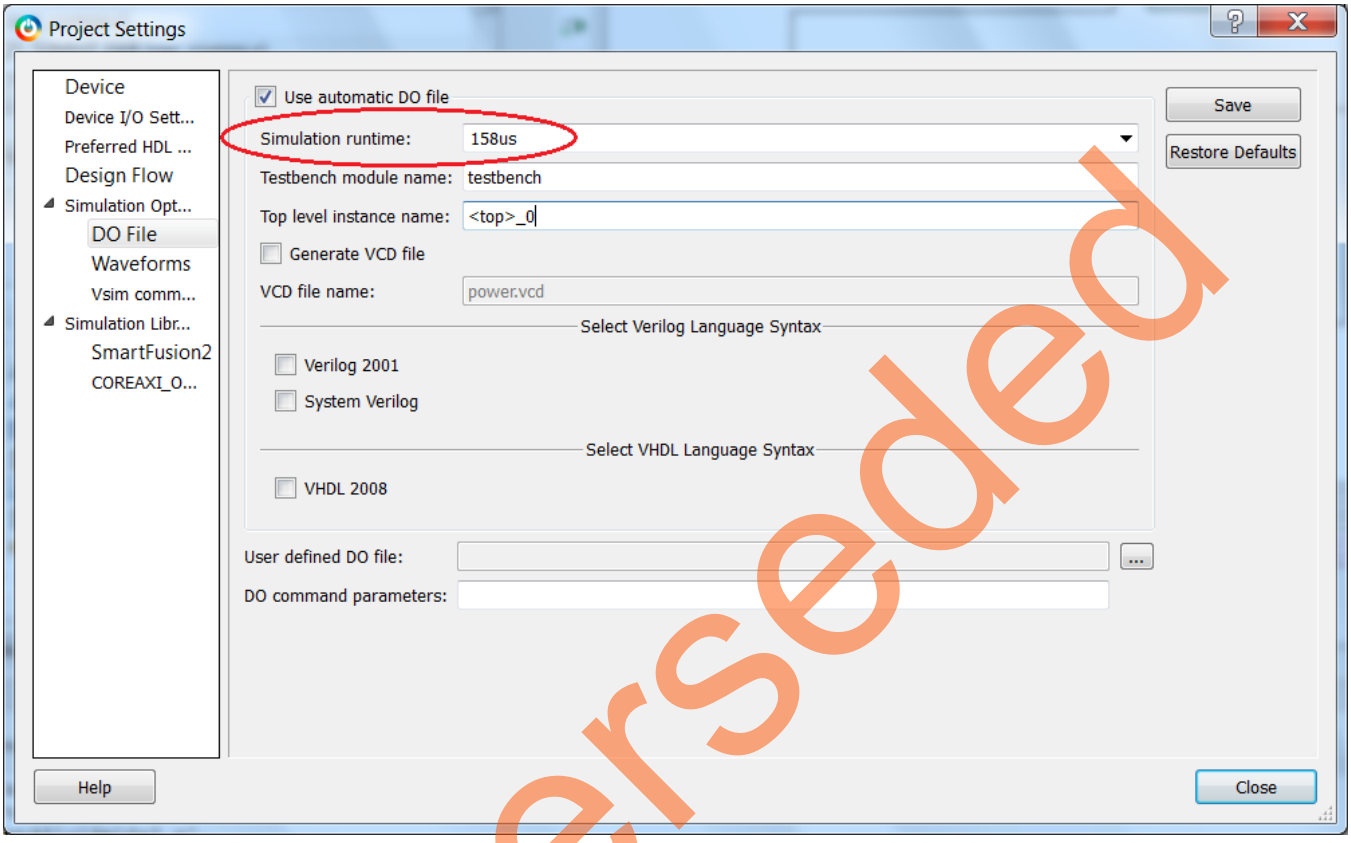


Figure 29. Simulation Runtime

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, since ModelSim AE does not support mixed-language flow. Compile with **-novopt** switch, if ModelSim full version is used.
- A .do file, **run_novopt.do** that 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 30.

Superseded

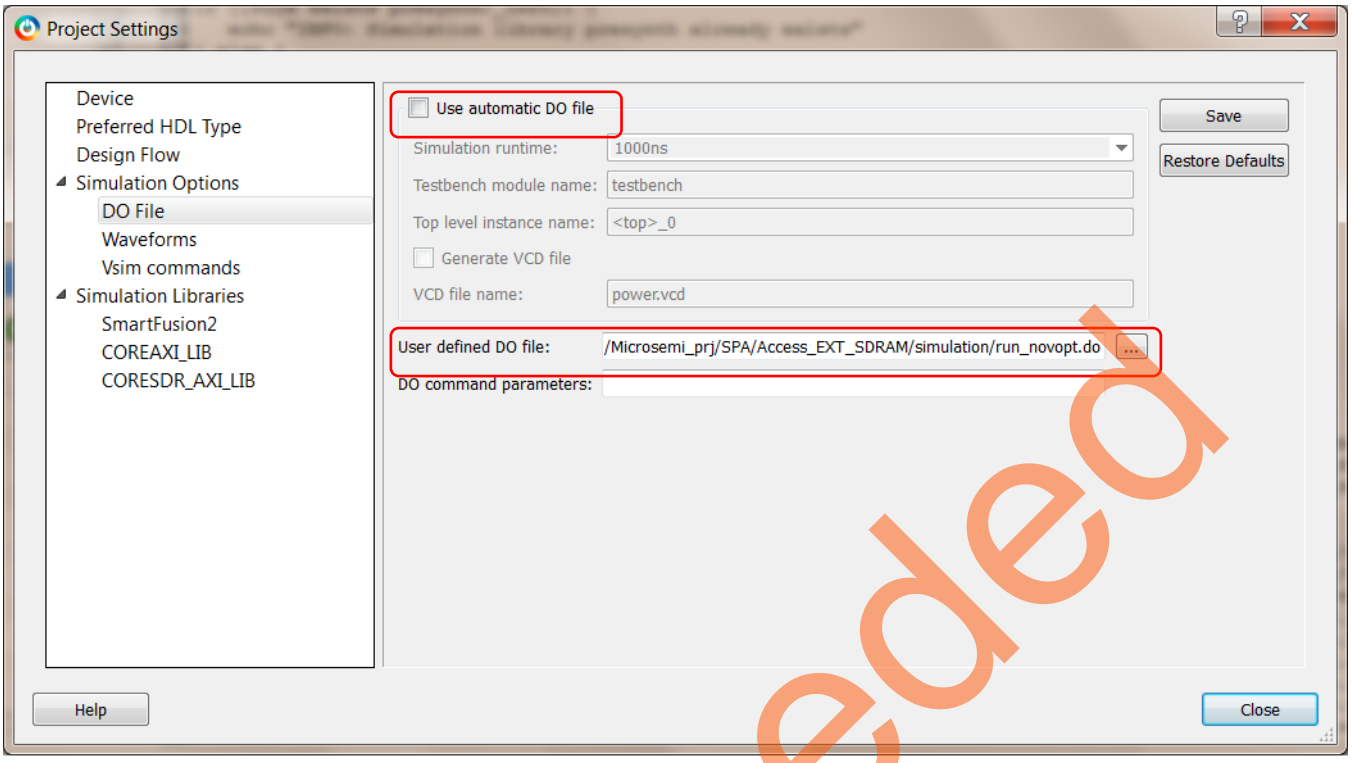


Figure 30. Specifying run_novopt.do for VHDL ModelSim Full Version

2. 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 external SDR SDRAM memory.

   Browse **wave.do** file from the above specified location, as shown in Figure 31.

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

Superseded

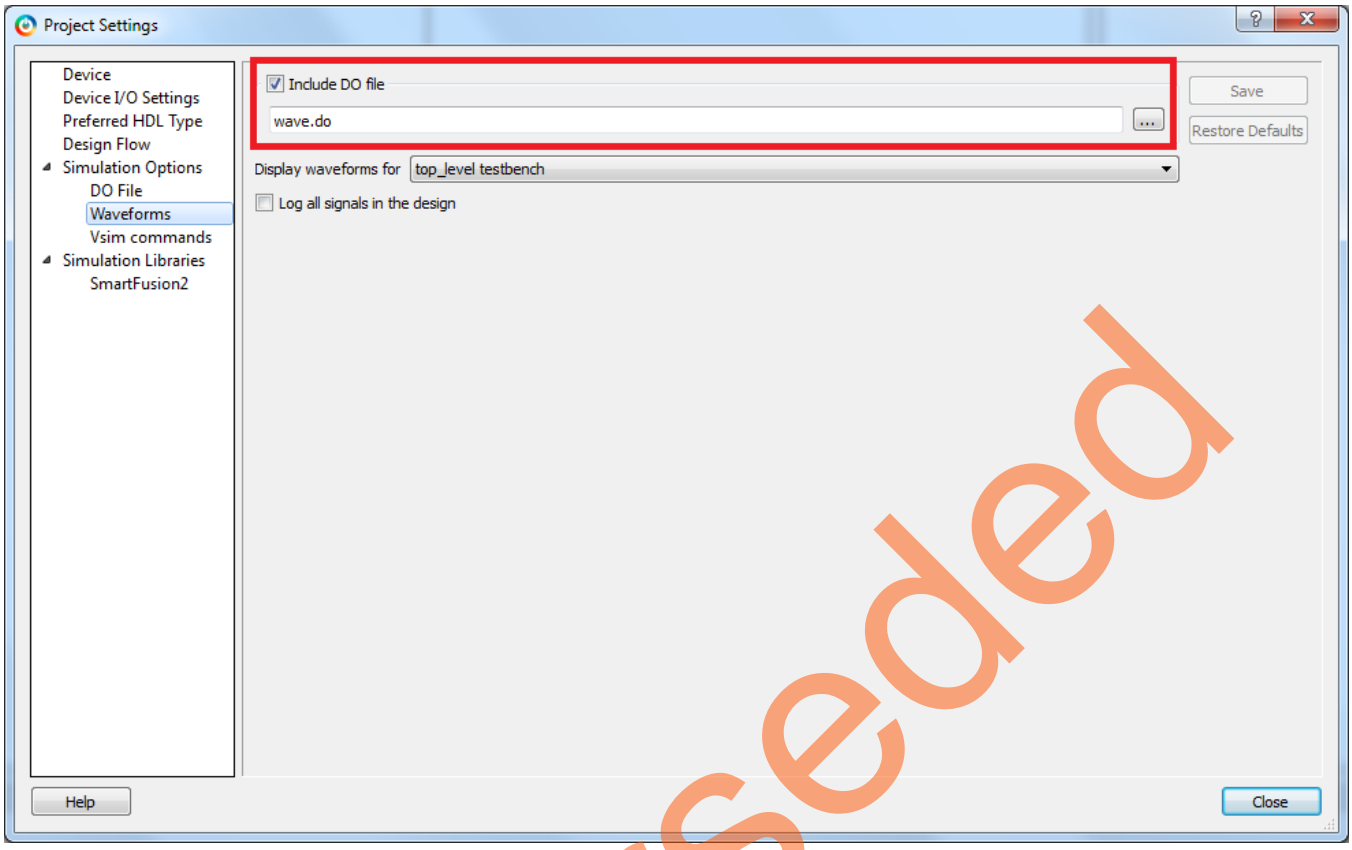


Figure 31. Adding Custom DO File for ModelSim Wave Window

3. Select the **Vsim Commands** option under the **Simulation Options,** and modify the **Resolution** to **1ps**, as shown in Figure 32. This option sets the simulation resolution to 1ps.

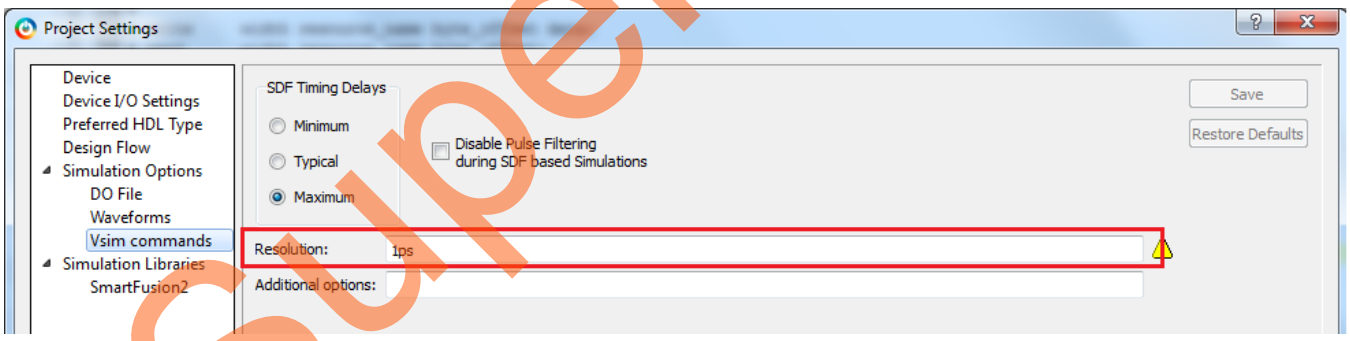


Figure 32. Simulation Resolution

4. Click **Save** and **Close** to exit the **Project Settings** window.
5. In the **Design Flow** tab on Libero SoC, expand the **Verify Pre-Synthesized Design** option and select the **Simulate** option under it:
   - Specify the testbench ModelSim to be used during simulation. To do so, right-click the **Simulate** option and select **Organize Input Files > Organize Stimulus Files**. The **Organize Stimulus files of Access_EXT_SDRAM for Simulate tool** window opens.
   - Change the **Use List of files organized by** option from **Libero** to **User**.
   - Select ***testbench.v*** in the **Associated Stimulus files** and click **Remove**.
   - Select the ***testbench.v*** and ***mt48lc16m16a2.v*** files under **Stimulus files in the project** and click **Add** to add them to the Associated Stimulus files as shown in Figure 33.

Superseded

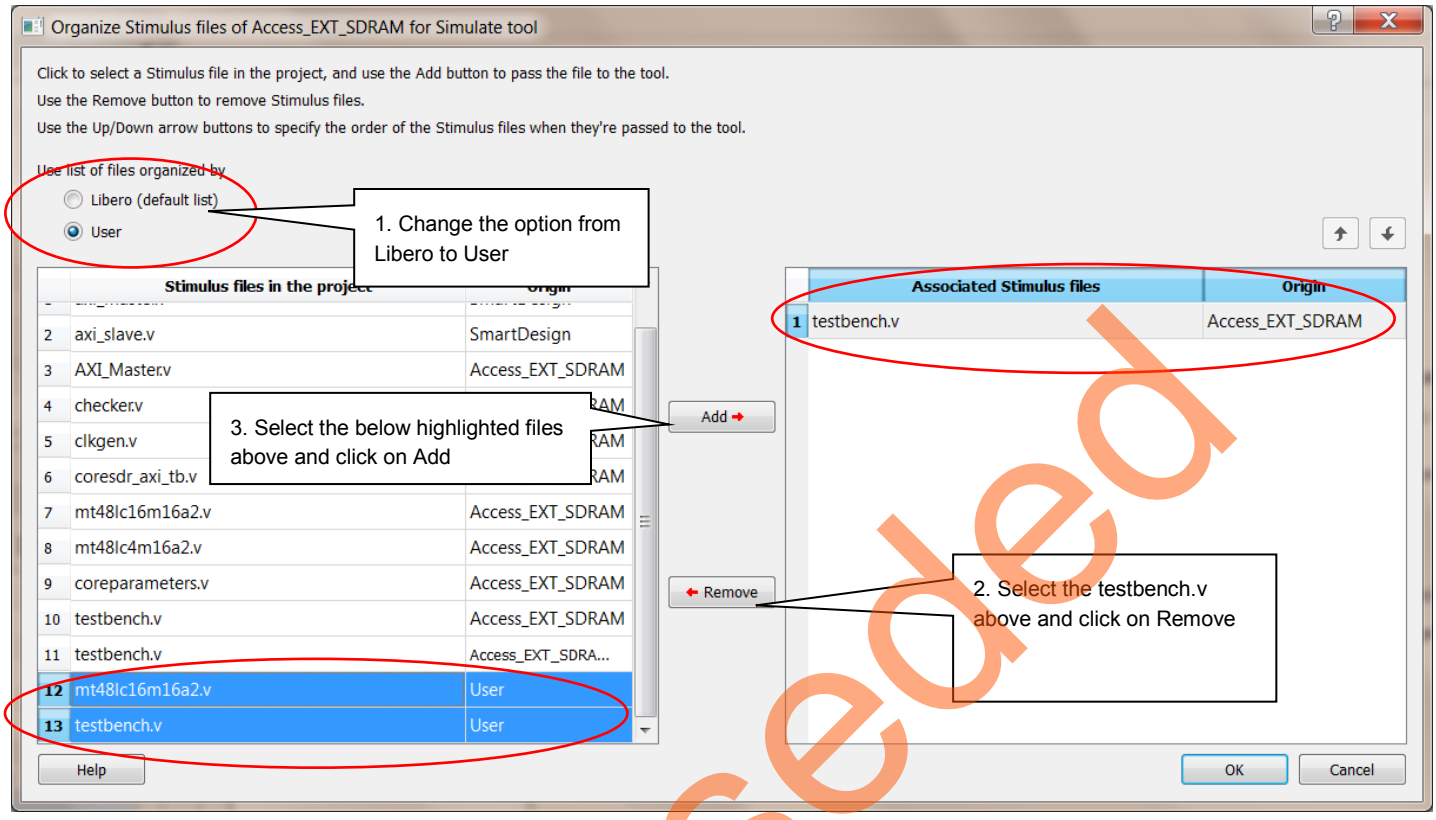


Figure 33. Organizing Stimulus Files

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

Superseded

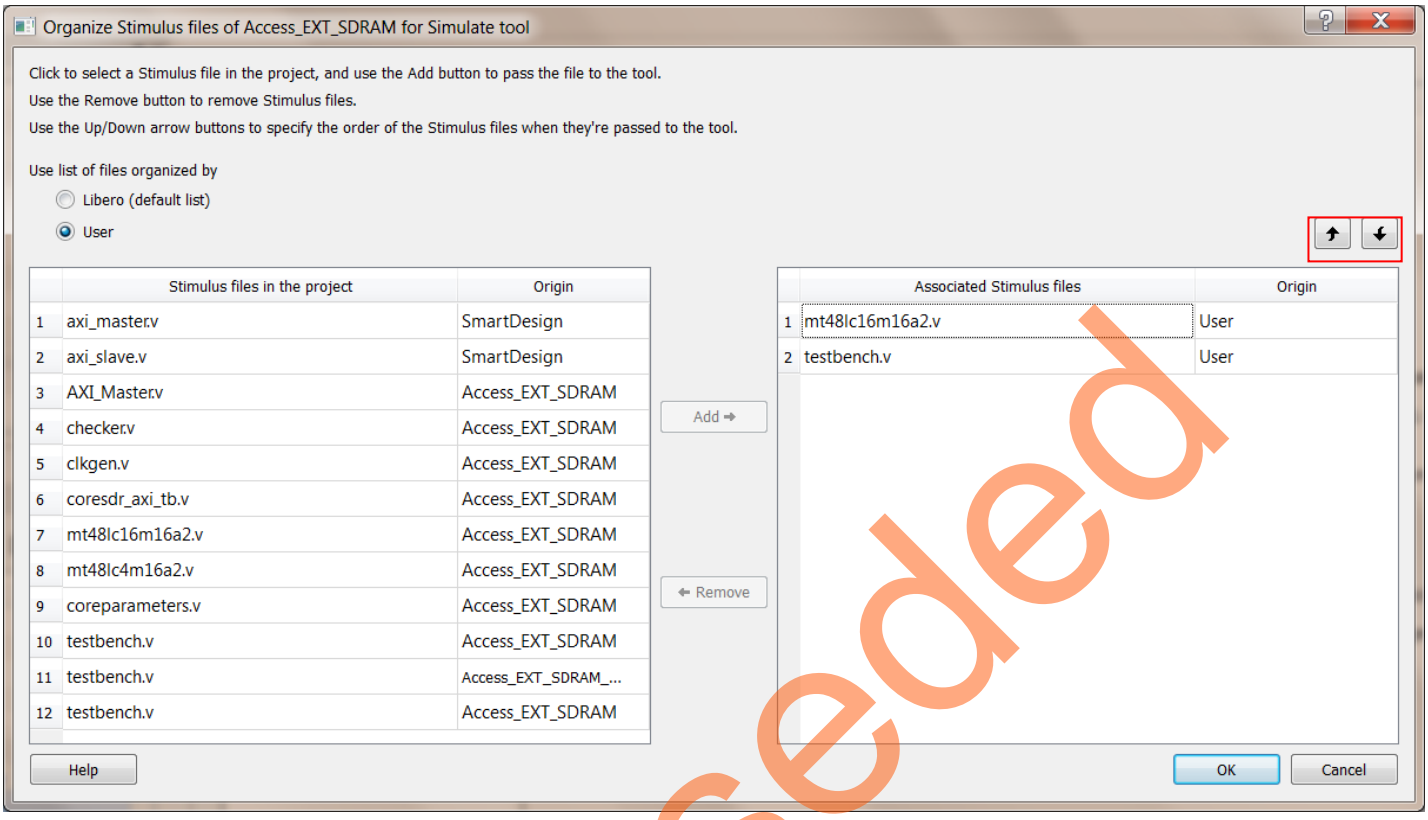


Figure 34. Organized Stimulus Files for the Simulation

6. Click **OK** and close the **Organize Stimulus files** dialog box.
7. After specifying the testbench stimulus file, expand the **Verify Pre-Synthesized Design** option, select the **Simulate** option under it, right-click and select **Open Interactively** to invoke ModelSim, as shown in Figure 35. ModelSim is invoked and the design is loaded.

Superseded

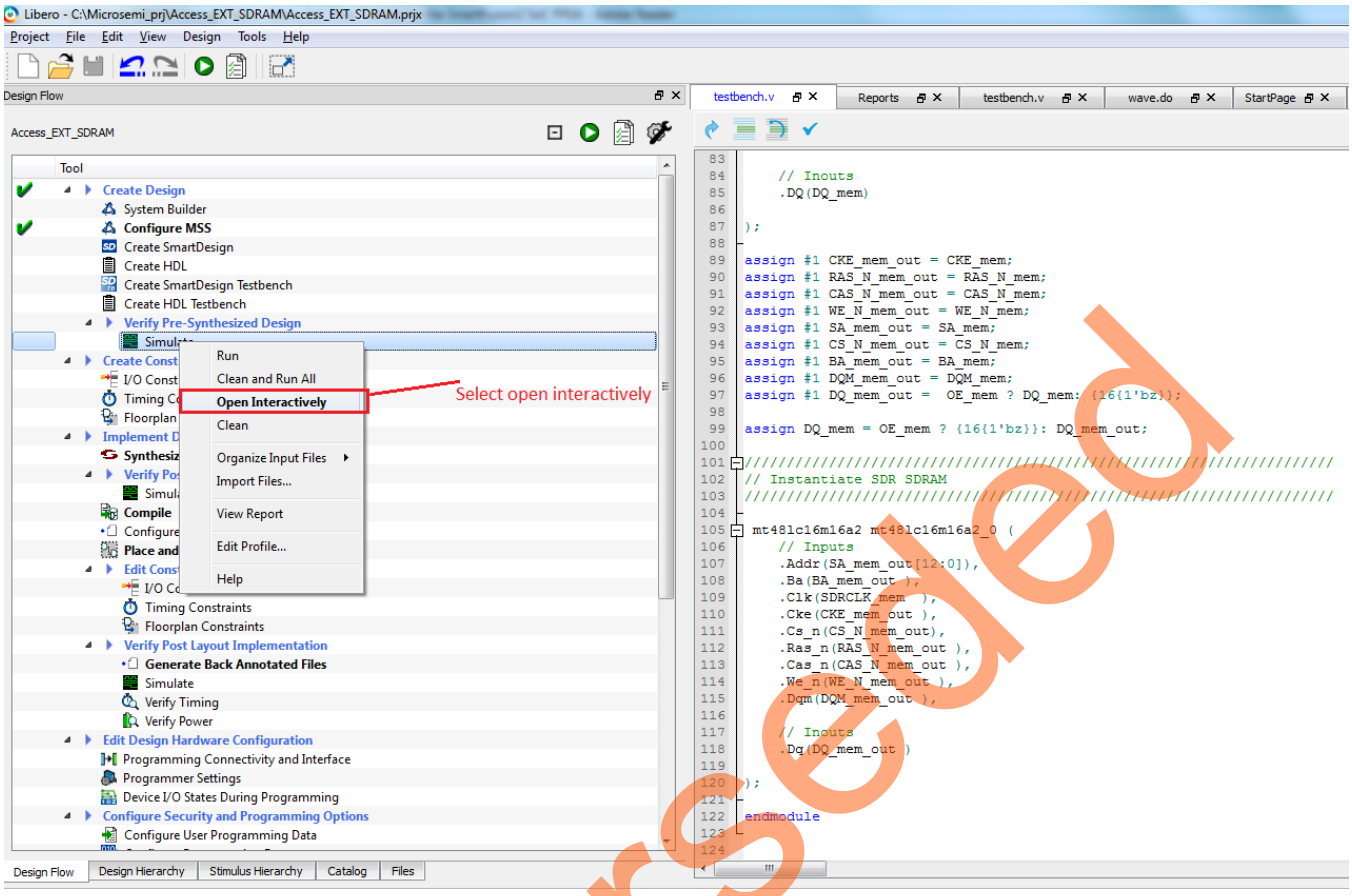


Figure 35. Invoke ModelSim

Viewing Simulation Results

1. ModelSim runs the design for about 158 us, as specified in the **Project Settings** window. Once the simulation has run completely, undock the **Wave** window by clicking the **Dock/Undock** button on the **Wave** window, as shown in Figure 36.

Figure 36. Dock/Undock Button in Wave Window

2. Click the **Zoom Full** button to fit all the waveforms in the single view (Figure 37).

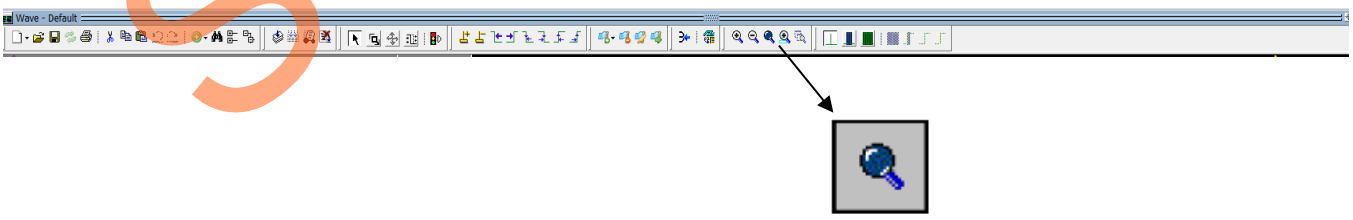


Figure 37. Zoom Full Button

Superseded

3. Place the cursor at 114 us on the **Wave** window and click the **Zoom In on the Active Cursor** button to zoom in at that location, as shown in Figure 38. Click as needed until complete write and read transactions to the external SDR SDRAM is seen on the **Wave** window, as shown in Figure 39.

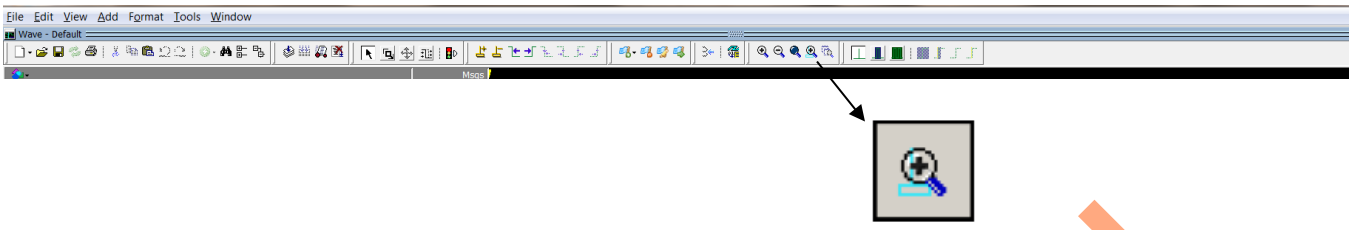


Figure 38. Zoom In on the Active Cursor

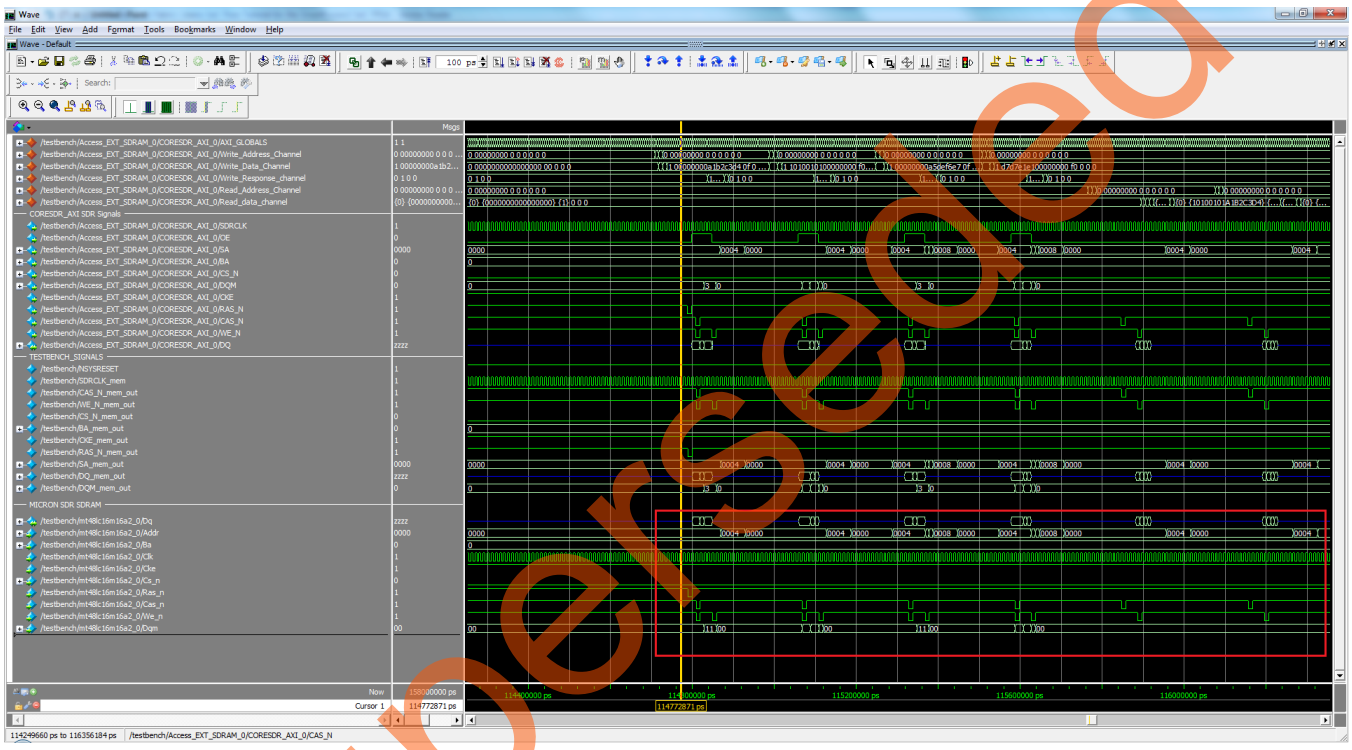
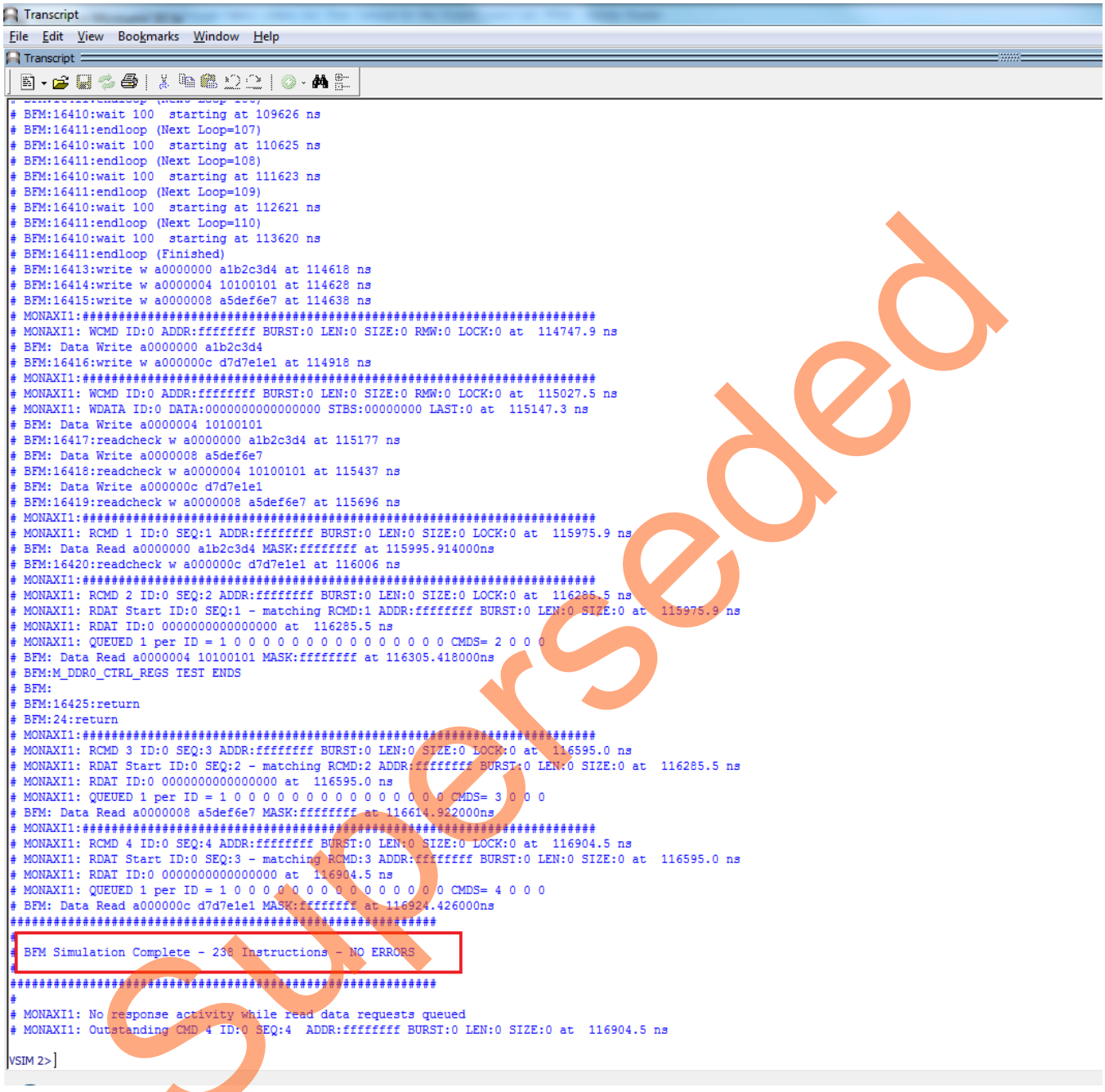


Figure 39. Write/Read Transactions

4. Analyze the Read and Write transactions on the **Wave** window by expanding the required signals.

Superseded

5. The simulation results can also be seen on the **Transcript** window of ModelSim, as shown in Figure 40.



Superseded

Figure 40. Transcript Window

The following message is displayed in the Transcript window:

```
# BFM: Data Read a0000000 a1b2c3d4 MASK:ffffffff at 115995.914000ns
# BFM: Data Read a0000004 10100101 MASK:ffffffff at 116305.418000ns
# BFM: Data Read a0000008 a5def6e7 MASK:ffffffff at 116614.922000ns
# BFM: Data Read a000000c d7d7e1e1 MASK:ffffffff at 116924.426000ns
```

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 along with the readcheck command. If the value read does not match, the simulation results in an error.

6. Go to **File > Quit** and quit the ModelSim simulator.

Conclusion

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

Superseded

List of Changes

Revision	Changes	Page
Revision 8 (September 2014)	Updated the document for Libero version 11.4 (SAR 60226).	NA
Revision 7 (May 2014)	Updated the document for Libero version 11.3 (SAR 56971).	NA
Revision 6 (November 2013)	Updated the document for Libero version 11.2 (SAR 52903).	NA
Revision 5 (April 2013)	Updated the document for 11.0 production SW release (SAR 47102).	NA
Revision 4 (March 2013)	Updated the document for Libero 11.0 Beta SP1 software release (SAR 44867).	NA
Revision 3 (November 2012)	Updated the document for Libero 11.0 beta SPA software release (SAR 42845).	NA
Revision 2 (October 2012)	Updated the document for Libero 11.0 beta launch (SAR 41584).	NA
Revision 1 (May 2012)	Updated the document for LCP2 software release (SAR 38953).	NA

Superseded

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.

Superseded

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.

Superseded

Superseded

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

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**.

© 2014 Microsemi Corporation. All rights reserved. Microsemi and the Microsemi logo are trademarks of Microsemi Corporation. All other trademarks and service marks are the property of their respective owners.

50200311-8/09.14