Design Creation using MSS Fabric Interfaces in SmartFusion 2 MSS User Guide

Introduction (Ask a Question)

The SmartFusion® 2 Microcontroller Subsystem (MSS) offers four different Fabric Interface Controller (FIC), depending on your device:

- DDR_FIC
- SMC_FIC
- FIC_0
- FIC_1

These interface blocks enable the MSS to interface with logic implemented in the FPGA fabric and vice-versa.

The DDR_FIC is used when you configure the MSS DDR block (MDDR) such that the external DDR memory can be accessed from an FPGA fabric master through an Advanced eXtensible Interface (AXI) or two AHBLite Advanced Microcontroller Bus Architecture (AMBA®) interfaces.

The SMC_FIC is used when you configure the MSS DDR block in the Single Date Rate (SDR) mode. In this configuration, the MSS accesses external SDR Dynamic Random Access Memory (DRAM) or Asynchronous memories through a soft memory controller instantiated in the FPGA fabric, such as CoreSDR_AXI. The SMC_FIC is an AXI or AHBLite target AMBA interface. The DDR_FIC and SMC_FIC interfaces are mutually exclusive; only one is active at a time.

The FIC interfaces enable you to naturally extend the MSS AMBA Bus into the FPGA fabric. There are up to two FIC instances per MSS depending on the selected device. The first instance is named FIC_0 (which is available on every device) and the second is named FIC_1 (may not be present in the smaller devices). You can configure the FIC as either an APB3 or AHBLite AMBA interface depending on your design needs. In each mode, a master and a slave and bus interface is available. That is, a master in the fabric can interface to a slave in the MSS and a master in the MSS can interface to a target in the fabric.

Each Fabric Interface subsystem operates on a different clock frequency, defined as a ratio of the MSS main clock M3_CLK.

The SmartFusion 2 architecture imposes rules related to clocking domains between the fabric interfaces and the FPGA fabric. This document provides guidance on how to properly construct such systems.

Table of Contents

1. Overview (Ask a Question)

The following figures show how the MSS connects to the FPGA fabric through the various Fabric Interface Controllers (FIC). Figure 1-1 shows the block diagram when DDR_FIC is used (external DDR memory) to configure and Figure 1-2 shows a block diagram of when SMC_FIC is used (external SDR memory).

The following figures show the MSS sub-blocks essential to connect the MSS to the FPGA fabric. The FIC sub-block may or may not be used in your application. You may not be using the DDR_FIC or SMC_FIC interfaces in your design. However, this does not change the overall requirements for how to create a design with one or more FIC blocks used in the design.

Figure 1-1. MSS to FPGA Fabric Block Diagram—DDR_FIC Mode

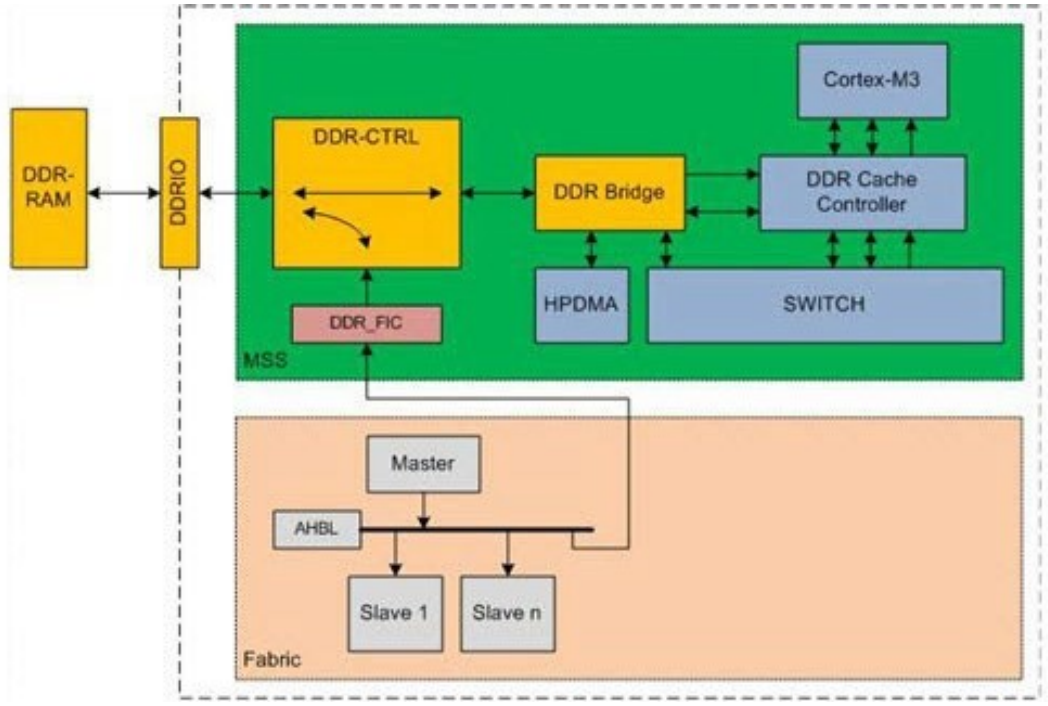
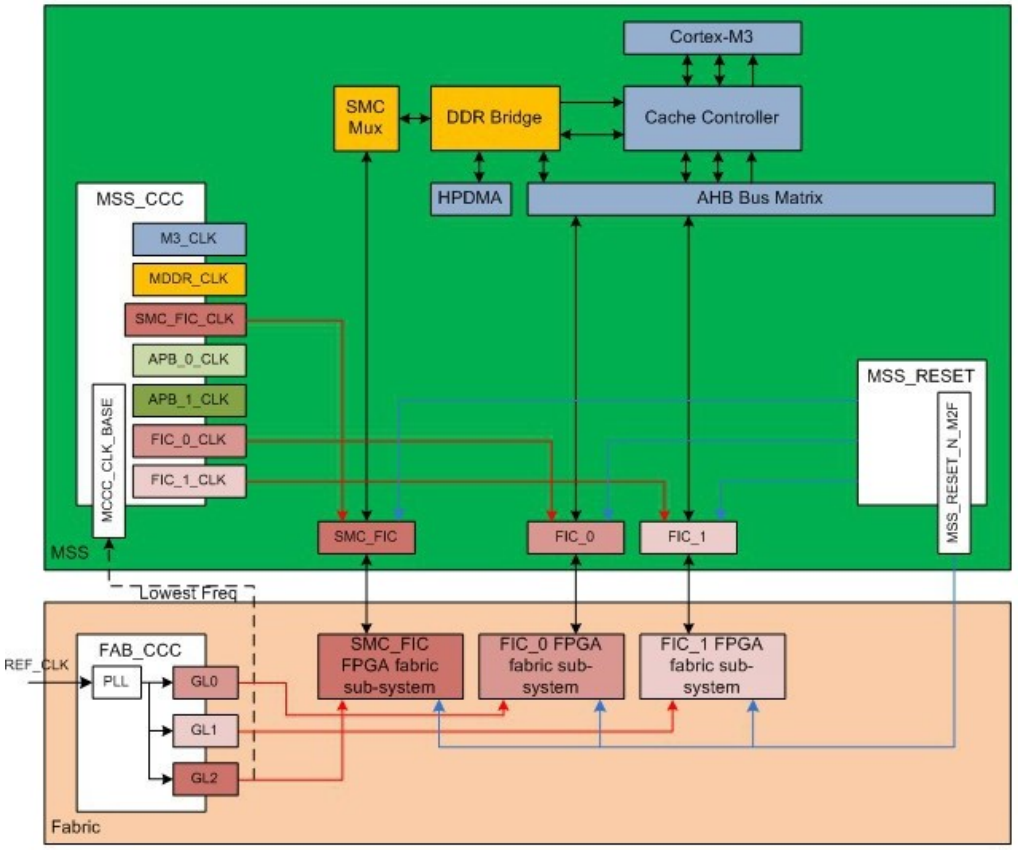


Figure 1-2. MSS to FPGA Fabric Block Diagram—SMC_FIC Mode



The MSS contains the following blocks:

- AHB_Bus matrix: All transactions in the MSS go through this block.
- DDR Controller: Interfaces with the DDR_FIC.
- DDR_FIC (SMC_FIC) sub-block: Used if you enable the FPGA fabric to DDR path in the MSS_MDDR configurator.
- FIC_0 sub-block
- FIC_1 sub-block: In larger devices only.
- FIC_2 sub-block: For APB initialization of SerDes and external DDRs.
- MSS_CCC sub-block needed to configure the FIC clocks relative to the MSS main clock (M3_CLK).
- MSS_RESET sub-block that generates the MSS internal resets as well as the MSS_RESET_N_M2F signal that drives the FPGA fabric.

The FPGA fabric contains:

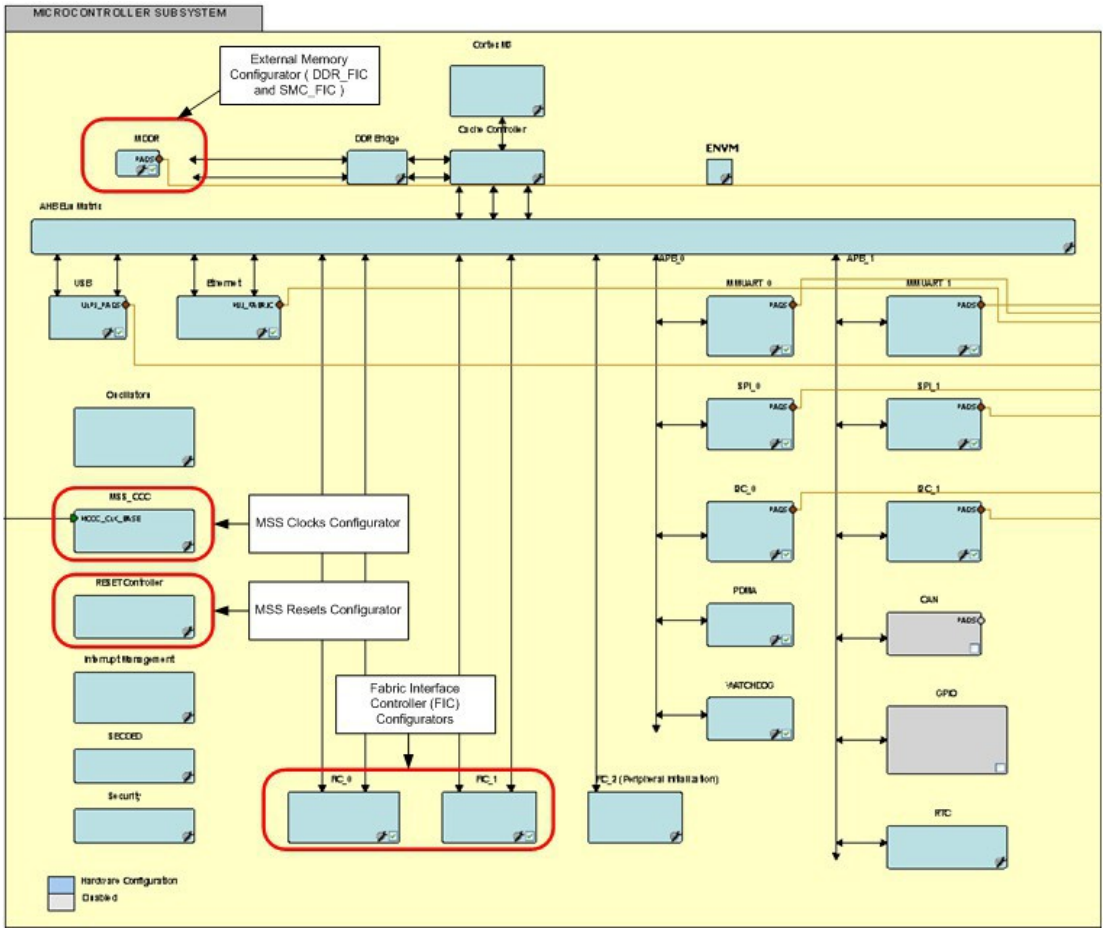
- Three FIC subsystems (DDR_FIC, FIC_0 and FIC_1)
- Fabric CCC and FAB_CCC: Generates the clocks that drive the FPGA fabric FIC sub\u0002system as well as the MCCC_CLK_BASE port on the MSS block.
- FAB_CCC reference clock signal: This is one of the on-chip oscillators, an external crystal oscillator, a clock generated internally to the fabric or from outside the chip.

The block diagrams also show the clock network in red and the reset network in dark blue. This document discusses each of these blocks and describes how to connect the clocks and resets.

2. MSS Configurator (Ask a Question)

This document refers to certain MSS sub-blocks that are configured as part of creating a design where the MSS interfaces with the FPGA fabric through the FIC sub-blocks. These blocks are highlighted in red in the following figure.

Figure 2-1. MSS Configurator



3. SmartDesign and MSS Configurator Actions (Ask a Question)

There are several common actions referenced in this document; they are summarized in the following list:

Tip: If you are familiar with SmartDesign and the MSS, you can skip to the next section.

- Instantiating a core: The action of selecting a core from the Libero® SoC IP catalog, dragging and dropping that core onto a SmartDesign Canvas.
- Configuring a core: The action of opening the configurator for an instance of that core on the canvas. In the configurator dialog box, select a particular configuration and click OK to commit. As a result of that action, only ports pertinent to the current configuration are visible and available for use.
- Instantiating a component: The action of selecting a component in the Libero SoC Design Hierarchy, dragging and dropping that component onto a SmartDesign canvas.
- Instantiating a custom AMBA compliant component: The action of selecting an HDL+ component in the Libero SoC Design Hierarchy, dragging and dropping that component onto a SmartDesign canvas. Using the HDL+ feature enables you to add AMBA compliant Bus Interfaces (BIFs) to your regular HDL module.
- Configuring a component:The action of opening the configurator for a component instantiated on a SmartDesign Canvas or from the Design Hierarchy. In the configurator dialog box, select a particular configuration and click OK to commit. As a result of that action, all instances on that component in the Libero SoC project are affected and need to be updated. When an instance is updated with the latest component, only ports pertinent to the current configuration are visible and available for use.
- Enabling an MSS sub-block: Some MSS sub-blocks can be enabled or disabled, indicating that they are used in the current application. All unused sub-blocks must be disabled.
- Configuring an MSS sub-block: The action of opening the configurator for that sub-block in the MSS configurator. In the configurator dialog box, select a particular configuration and click OK to commit, then save the MSS configuration. As the result of these actions, the MSS component has a new configuration and potentially a new port interface. When the instance of the MSS is updated with the latest component, only the MSS ports pertinent to the current configuration are visible and available for use.
- Creating a FPGA fabric subsystem: The action of instantiating, configuring cores and components in a SmartDesign canvas and connecting them together and to the MSS component instantiated already in that particular SmartDesign.

4. Configuring the DDR_FIC Subsystem (Ask a Question)

To configure or create a DDR_FIC subsystem, perform the following steps:

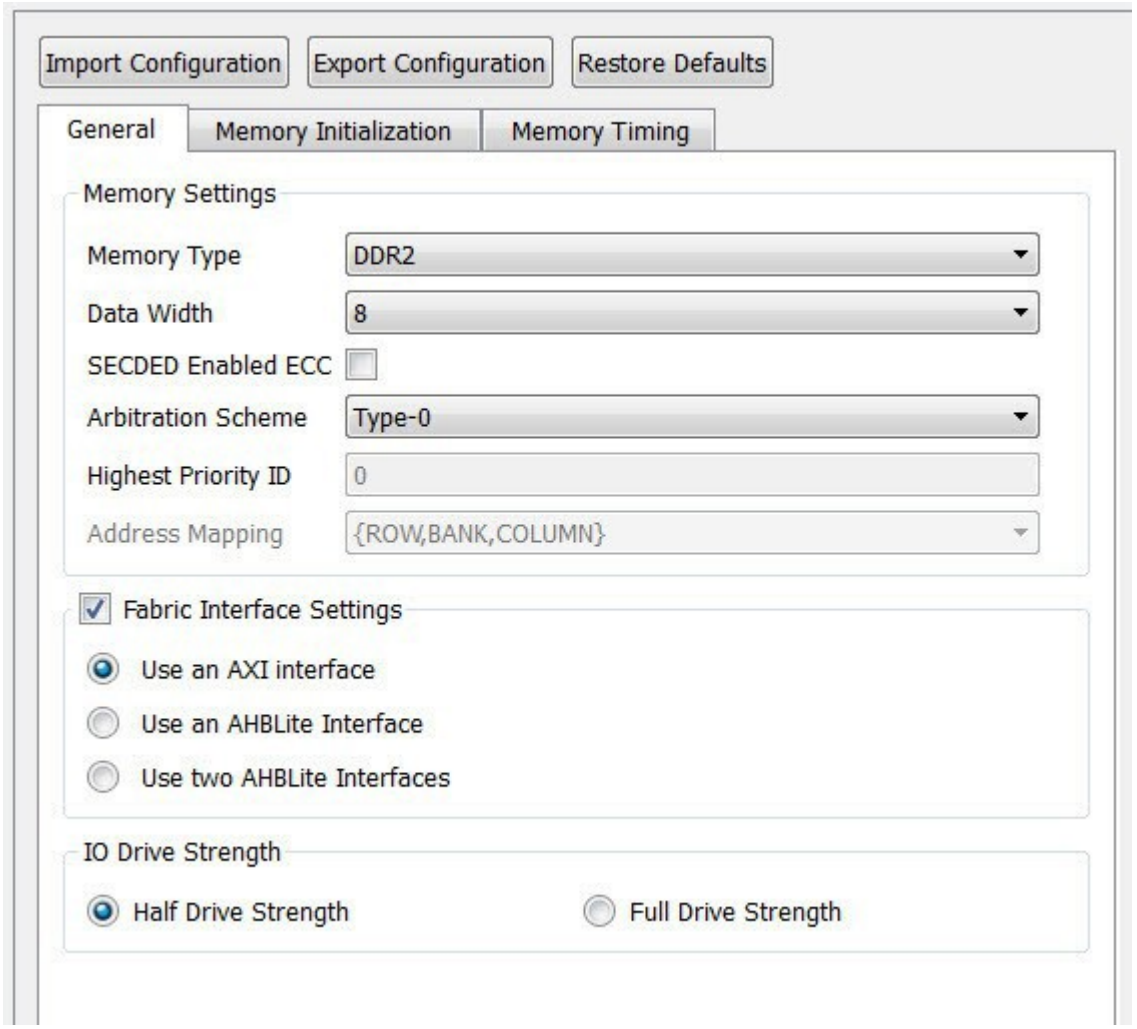
1. Configure the MSS MDDR to expose the DDR_FIC interface.
2. Create the FPGA fabric DDR_FIC subsystem including instantiation/configuration/connectivity for the following buses:
   - AXI or AHBLite bus
   - AXI or AHBLite bus master(s)
   - Other masters and peripherals on the bus as required by your application.
   - Clocks and resets. For more infromation, see Configuring the FIC Subsystem Clocks and Configuring the FIC Subsystem Reset sections.

These steps are described in detail in the following sections:

4.1 Configure the MSS MDDR Sub-block to Expose the DDR_FIC Bus Interface (Ask a Question)

The DDR_FIC interface is exposed when your application needs to access the external DDR memory from the FPGA fabric. In this configuration, the MDDR subblock exposes the DDR_FIC interface, which is a slave (Advanced eXtensible Interface (AXI)) or AHB-Lite Bus Interface (BIF). The following figure shows the MSS DDR configuration with access from FPGA Fabric.

Figure 4-1. MSS DDR Configuration with Access from FPGA Fabric



4.2 Create the FPGA Fabric DDR_FIC Subsystem (Ask a Question)

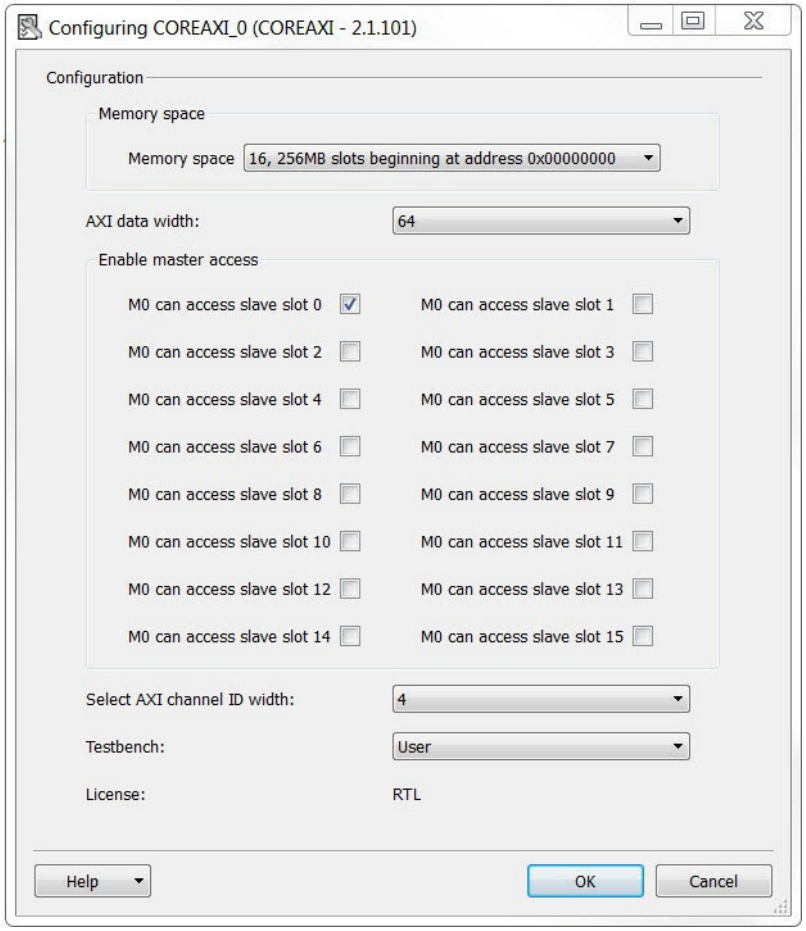
Based on the FPGA bus connection type you selected—AXI, Single AHBLite or double AHB-Lite—you must create a subsystem that matches your selection.

4.2.1 DDR_FIC/AXI Subsystem (Ask a Question)

To configure/create a DDR_FIC/AXI subsystem, perform the following steps:

1. From the **Catalog** window, instantiate and configure the CoreAXI IP core. Enable the slots you plan to use for your application as well as the amount of memory per slot that matches your design requirements. Since you are addressing an external DDR memory, your slot size selection must match the space that you plan on addressing from the FPGA fabric master, see the following figure.

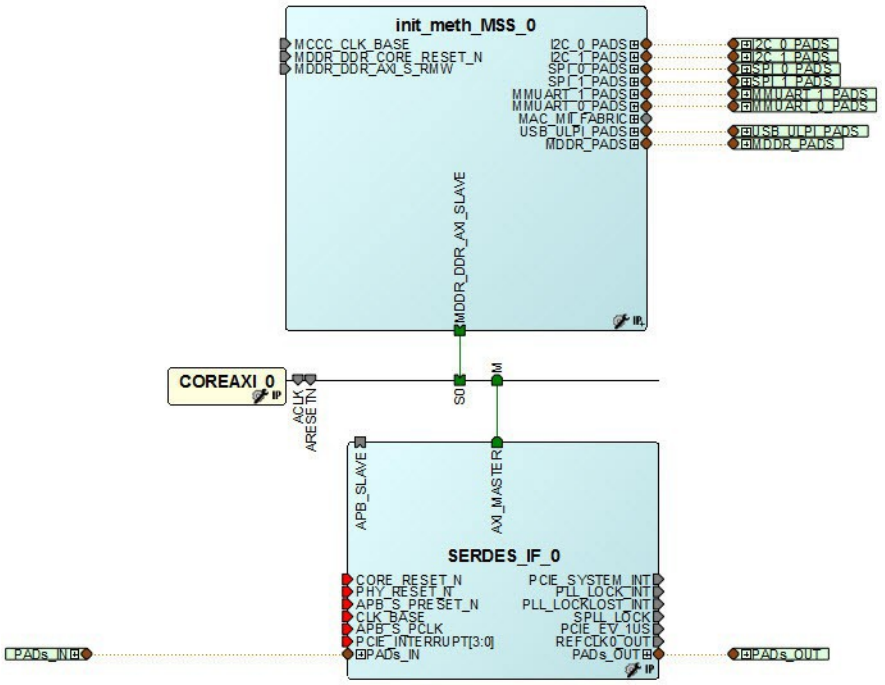
Figure 4-2. CoreAXI Configuration



2. Instantiate and configure the AMBA AXI-compliant master core or component that is intended to master to the AXI bus. If your application requires more than one master onto the CoreAXI bus, instantiate the second master also.
3. Connect the subsystem:
   a. Connect the CoreAXI mirrored-master Bus Interface (BIF) port M0 (M1) to the master BIF port of your master core instance(s).
   b. Connect the MSS DDR_FIC slave BIF port—MDDR_DDR_AXI_SLAVE—to the proper CoreAXI bus mirrored-slave slot as per your memory map requirement. If you have other slaves on that bus, connect them, as per your memory map.
   c. Clocks and resets. For more information, see Configuring the FIC Subsystem Clocks and Configuring the FIC Subsystem Reset sections.

   The following figure shows the DDR_FIC AXI subsystem.

Figure 4-3. DDR_FIC AXI Subsystem

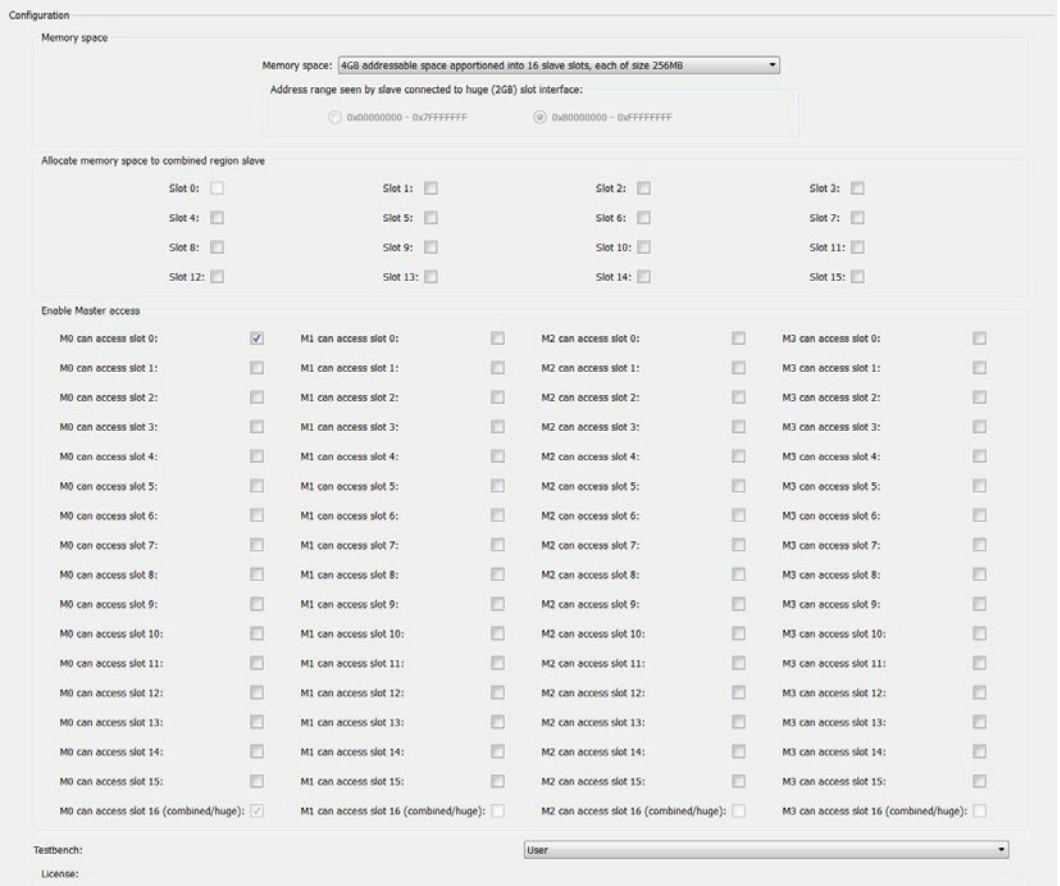


4.2.2 DDR_FIC/Single-AHBLite Subsystem (Ask a Question)

To configure/create a DDR_FIC/Single-AHBLite subsystem, perform the following steps:

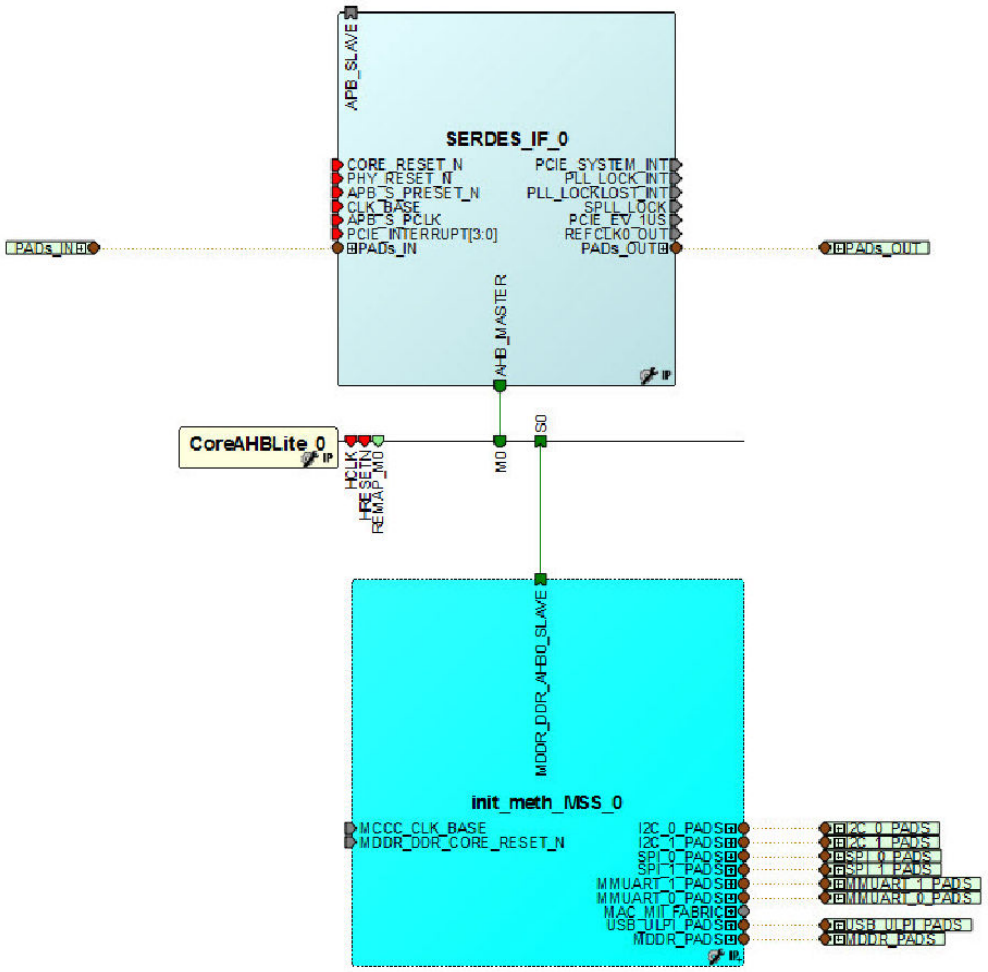
1. From the **Catalog** window, instantiate and configure the CoreAHBLite IP core. Enable the slots that you plan to use for your application as shown the following figure. In this example, CoreAHBLite is configured to address one 4GB of DDR RAM memory space using slot0 from master M0. Since you are addressing an external DDR memory, your slot size selection must match the amount of DDR memory space that you plan on addressing from the FPGA fabric master.

Figure 4-4. Core AHBLite—Combined Region Master Configuration



2. Instantiate and configure the AMBA AHBLite compliant master core or component that is intended to Master to the CoreAHBLite bus. If your application requires more than one master onto the CoreAHBLite bus, instantiate the additional masters also. Up to four masters are supported on the CoreAHBLite bus.
3. Connect the subsystem:
   a. Connect the CoreAHBLite mirrored-master BIF port M0 (M1) to the master BIF port of your master core instance(s).
   b. Connect the MSS DDR_FIC slave BIF port—MDDR_DDR_AHB0_SLAVE—to the proper CoreAHBLite bus mirrored-slave slot (S0 in this example) as per your memory map requirement. If you have other slaves on that bus, connect them as per your memory map. Clocks and resets. For more information, see Configuring the FIC Subsystem Clocks and Configuring the FIC Subsystem Reset sections.

Figure 4-5. DDR_FIC AHBLite Subsystem



4.2.3 DDR_FIC/Two AHBLite Subsystem (Ask a Question)

When you select the two **AHBLite Interfaces** option for the MDDR, an additional BIF—MDDR_DDR_AHB1_SLAVE BIF—is exposed at the MSS component for you to connect to the new slave.

For this configuration, repeat the steps for a single AHBLite configuration for the MDDR_DDR_AHB1_SLAVE BIF interface exposed on the MSS component.

5. Configuring the SMC_FIC Subsystem (Ask a Question)

Although the SMC_FIC can be used as an AXI or AHBLite bus interface, this document only describes how to use the SMC_FIC interface configured in AXI mode to connect to the CoreSDR_AXI core. That core is an AXI-based SDR RAM controller used to connect, in this case, the MSS to an external SDR memory component. You can easily infer how to use the AHBLite interface from the following description; the steps are very similar. The AXI interface is a more efficient interface and is the preferred option.

To configure/create a SMC_FIC subsystem, perform the following steps:

1. Configure the MSS MDDR to expose the SMC_FIC interface.
2. Create the FPGA fabric DDR_FIC subsystem, including instantiation/configuration/connectivity for:
   - CoreAXI bus
   - CoreSDR_AXI
   - Other masters and peripherals on the bus as required by your application.
   - Clocks and resets. For more information, see Configuring the FIC Subsystem Clocks and Configuring the FIC Subsystem Reset sections.

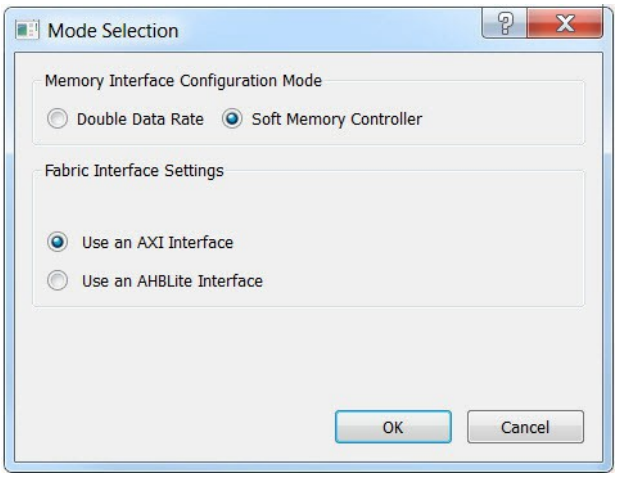
These steps are described in detail in the following sections:

5.1 Configuring the MSS MDDR Sub-block to Expose the SMC_FIC Bus Interface (Ask a Question)

The SMC_FIC interface is exposed when your application needs to access an external SDR memory through the FPGA fabric. In this configuration, the MDDR sub-block exposes the SMC_FIC interface, which is a master AXI or AHBL BIF, see the following figure.

1. Right-click the MDDR controller inside the MSS configurator and choose **Configure**.
2. Select **Use an AXI Interface**.
3. Click **OK**.

Figure 5-1. MDDR Soft Memory Controller Configuration



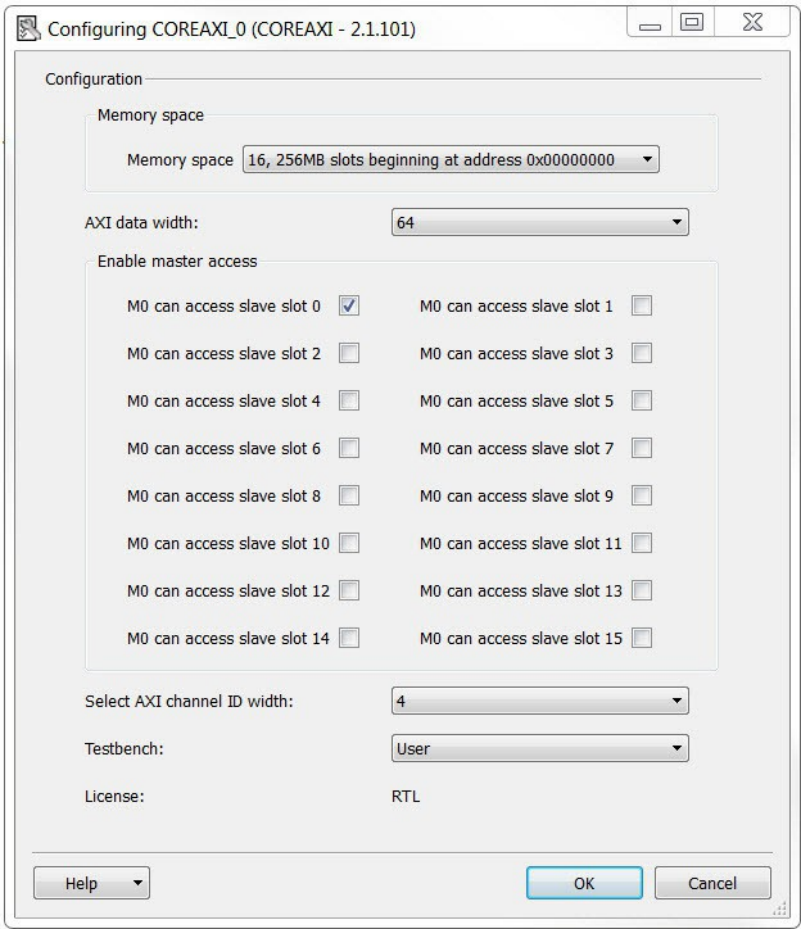
4. To update the MSS component, right-click the MSS Component and select **Update Instances with Latest Component** option. The MDDR_SMC_AXI_MASTER is exposed as a BIF port of the MSS component.

5.2 Create the FPGA Fabric SMC_FIC Subsystem (Ask a Question)

To configure/create the FPGA Fabric SMC_FIC subsystem, perform the following steps:

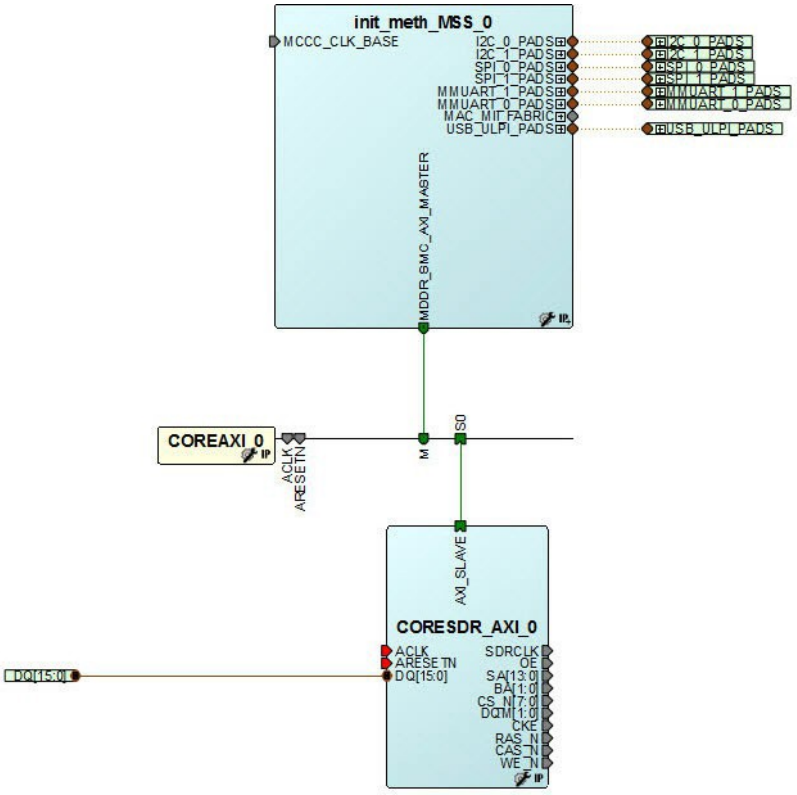
1. Instantiate and configure the CoreAXI such that the master slot M0 is enabled for the slave slot S0, see the following figure. Since you are addressing an external memory through a soft memory controller, your slot size selection must match the amount of external memory space that you plan on addressing from the Arm® Cortex®-M3 processor or any master writing to that external memory through the MSS DDR bridge.

Figure 5-2. CoreAXI Configuration—SMC_FIC Mode



2. From the IP Catalog, instantiate and configure CoreSDR_AXI to match your external memory parameters.
3. Connect the subsystem, as shown in the following figure:
   - Connect the MSS SMC_FIC master BIF port—MDDR_SMC_AXI_MASTER—to the CoreAXI bus mirrored-master M0.
   - Connect the CoreAXI mirrored-slave port S0 to the slave BIF port of the CoreSDR_AXI core instance.
   - Clocks and resets. For more information, see Configuring the FIC Subsystem Clocks and Configuring the FIC Subsystem Reset sections.

Figure 5-3. SMC_FIC Subsystem Connections



6. Configuring the FIC Subsystems (Ask a Question)

To configure/create an FIC subsystem, perform the following steps:

1. Configure the MSS FIC to expose the FIC interface.
2. Create the FPGA fabric FIC subsystem including instantiation/configuration/connectivity for:
   - APB3 or AHBLite bus
   - APB3 and AHBLite compliant master and/or peripherals configuration and connection onto the bus as required by your application.
   - Clocks and resets. For more information, see Configuring the FIC Subsystem Clocks and Configuring the FIC Subsystem Reset sections.

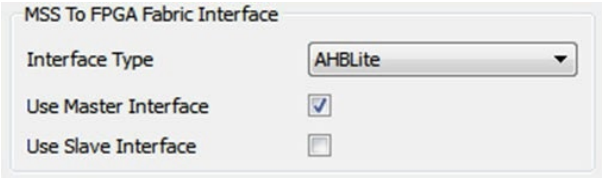
These steps are described in detail in the following sections.

6.1 Configure the MSS FIC Sub-Block (Ask a Question)

To configure the MSS FIC sub-block, perform the following steps:

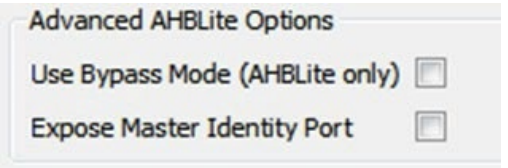
1. Invoke the FIC configurator and right-click on FIC_0 or FIC_1 to open the FIC configurator.
2. In the configurator for the MSS to FPGA Fabric Interface configuration group select the following options:
   - **Interface Type**: The type of AHBLite or APB3 interface.
   - **Use Master Interfaces**: This option let you select whether you intend to use the interface as a master of the FPGA fabric.
   - **Use Slave Interfaces**: This option let you select whether you intend to use the interface as a slave mastered by the FPGA fabric.

Figure 6-1. MSS to FPGA Fabric Interface Options



3. If you are using an AHBLite Interface you can also use the Advanced AHBLite Options option to select the bypass mode or expose the master ID port if you selected the interface to act as a master of the fabric, see the following figure.

Figure 6-2. Advanced AHBLite Options



6.2 Create the FPGA Fabric FIC Subsystem (Ask a Question)

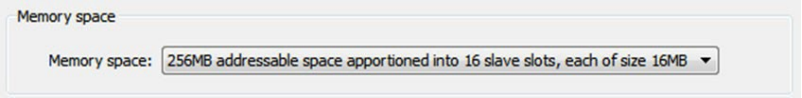
For each FIC interface exposed—Master and Slave, you must instantiate a bus (CoreAHBLite or CoreAPB3) that matches the type you selected. Depending on the interface role (master/slave) and type (AHBLite/APB3). The following configurations must be applied to the bus.

6.2.1 Master/AHBLite (Ask a Question)

To instantiate and configure the CoreAHBLite bus, perform the following steps:

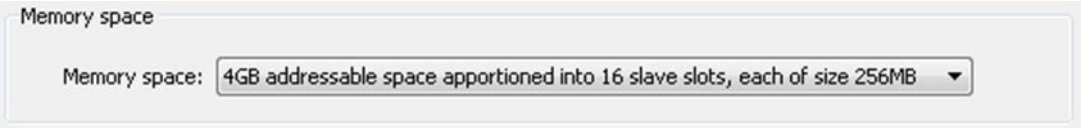
1. Select the **Memory Space** option that matches your requirements.
   - If you need less than 16 MB of address space of all your peripherals, select the option in the following figure. This mode provides 16 16 MB slots that connects up to 16 AHBLite slaves.

     Figure 6-3. Master/AHBLite Memory Space Configuration—16 MB per Slot

   - If you need more than 16 MB and less than 256 MB of address space for any of your peripherals, select the option as shown in the following figure. This mode provides 16 256 MB slots that connects up to 16 AHBLite slaves.

     Figure 6-4. Master/AHBLite Memory Space Configuration—256 MB per Slot

2. Enable the slots that you are using for your application as per the following figure. It is recommended to use the M1 to slot accesses, see the following figure.

Important: Use M1 if you create a multi master subsystem where you have a master in the fabric that requires the remap feature and thus needs to connected to M0.

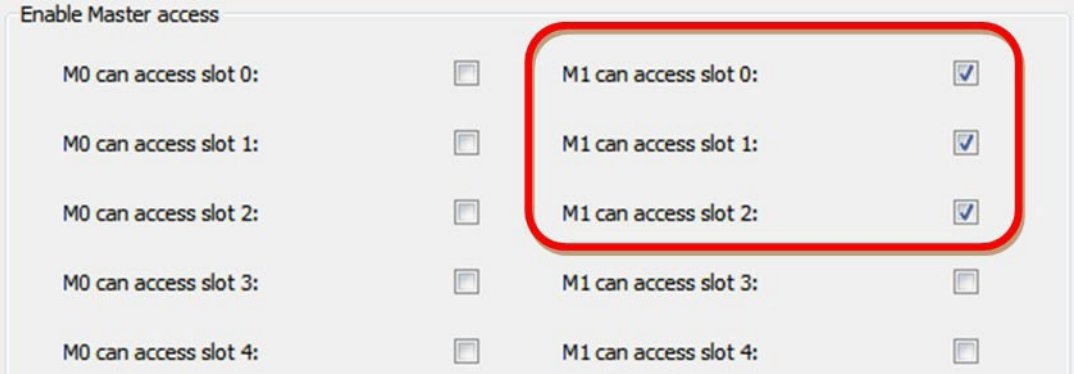
- If you have selected the 16 MB per slot option, there are no restrictions on which slots is used.
- If you have selected the 256 MB per slot option, only the slots compatible with the FIC instance fabric memory address regions selection can be used. Each FIC memory address region is 256 MB in size. The six FIC memory regions are summarized in the following figure.

The six FIC memory regions and their slots are listed in the following table.

Table 6-1. FIC Memory Regions

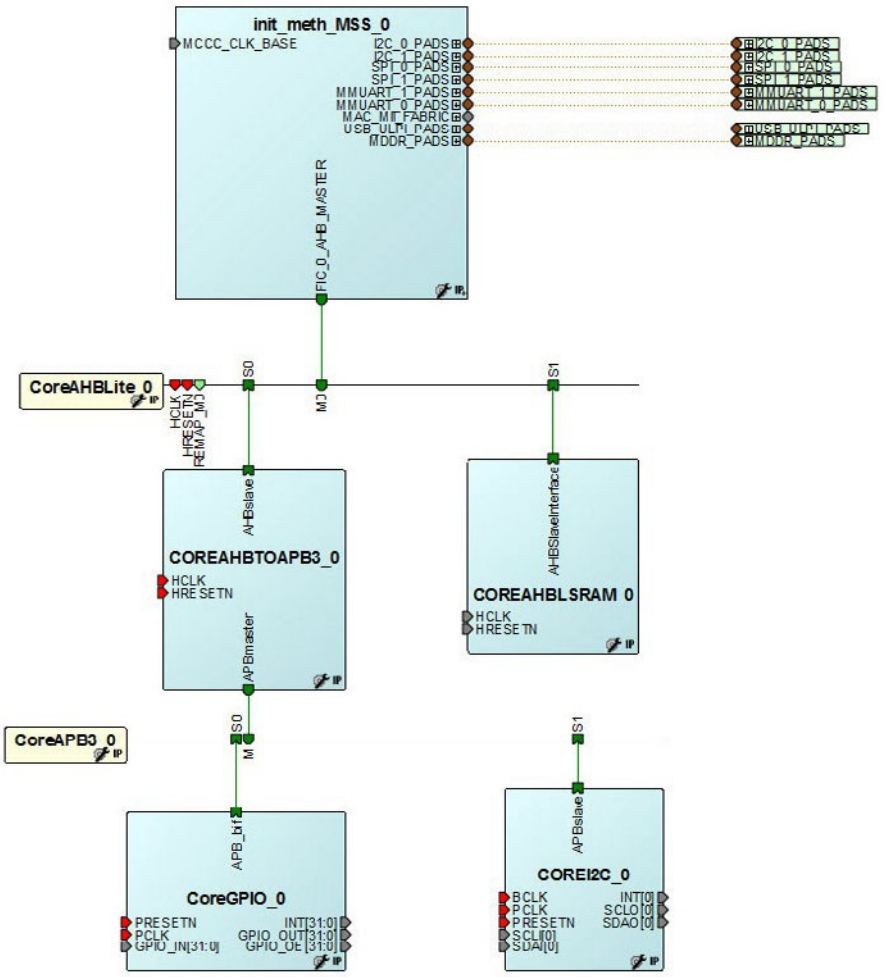
Memory Address Region	Compatible Slots
30000000-3FFFFFFF	3
50000000-5FFFFFFF	5
70000000-7FFFFFFF	7
80000000-8FFFFFFF	8
90000000-9FFFFFFF	9
F0000000-FFFFFFFF	15 (F)

Figure 6-5. Master/AHBLite—Master Access Configuration



3. Instantiate and configure AHBLite compliant peripheral cores and/or custom AHBLite compliant components.
4. To connect the subsystem, perform the following steps:
   a. Connect the CoreAHBLite mirrored-master BIF port M1 to the MSS master BIF port—FIC_0/1_AHB_MASTER, see the following figure.
   b. Connect the AHBLite slaves to the proper slots as per your memory map requirement. Clocks and resets. For more information, see Configuring the FIC Subsystem Clocks and Configuring the FIC Subsystem Reset sections.

Figure 6-6. FIC Master/AHBLite Subsystem



6.2.2 Master/APB3 (Ask a Question)

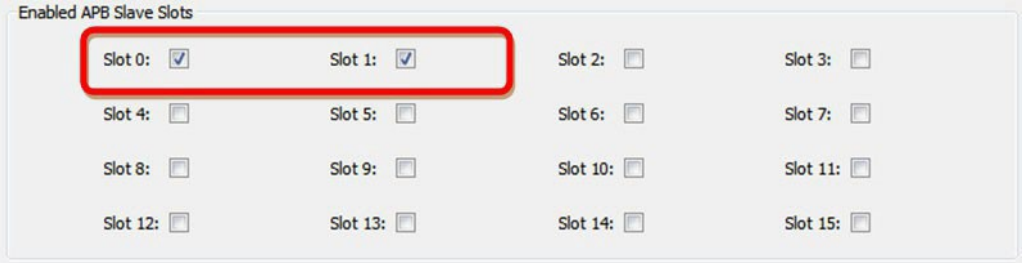
To instantiate and configure the CoreAPB3 bus, perform the following steps:

1. Select the **Address Configuration** options as shown in the following figure. This mode provides sixteen 16 MB slots that connects up to 16 APB3 compliant slaves. If you need slots with more memory, you can combine multiple slaves to build a larger slot. For more information, see CoreAPB3 User's Guide.

   Figure 6-7. Master/APB3 Address Configuration

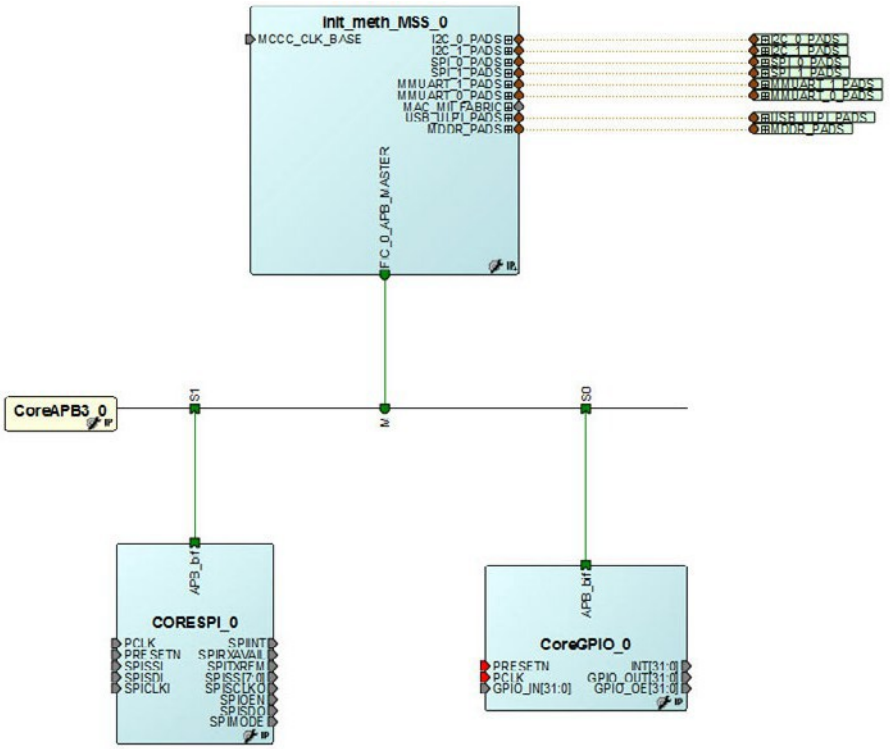
2. Enable the slots that you are using for your application, as shown in the following figure.

Figure 6-8. Master/APB3 Slave Slots Configuration



3. Instantiate and configure APB3 compliant peripheral cores and/or custom APB3 compliant components.
4. To connect the subsystem, perform the following steps:
   a. Connect the CoreAPB3 mirrored-master BIF port to the MSS master BIF port—FIC_0/1_APB_MASTER, see the following figure.
   b. Connect the APB3 slaves to the proper slots as per your memory map requirement. For more information on clocks and resets, see Configuring the FIC Subsystem Clocks and Configuring the FIC Subsystem Reset sections.

Figure 6-9. FIC Master/APB3 Subsystem



6.2.3 Slave/AHBLite (Ask a Question)

Slave/APB3 interfaces are available for configuration. A fabric AHBLite master connected through a CoreAHBLite bus and the FIC Slave AHBLite interface can access MSS peripherals and other memory spaces.

6.2.4 Slave/APB3 (Ask a Question)

Slave/APB3 interfaces are available for configuration. A fabric APB3 master connected through a CoreAPB3 bus and the FIC Slave APB3 interface can access MSS peripherals and other memory spaces.

7. Configuring the FIC Subsystem Clocks (Ask a Question)

To create the proper clock configuration and connectivity, perform the following settings:

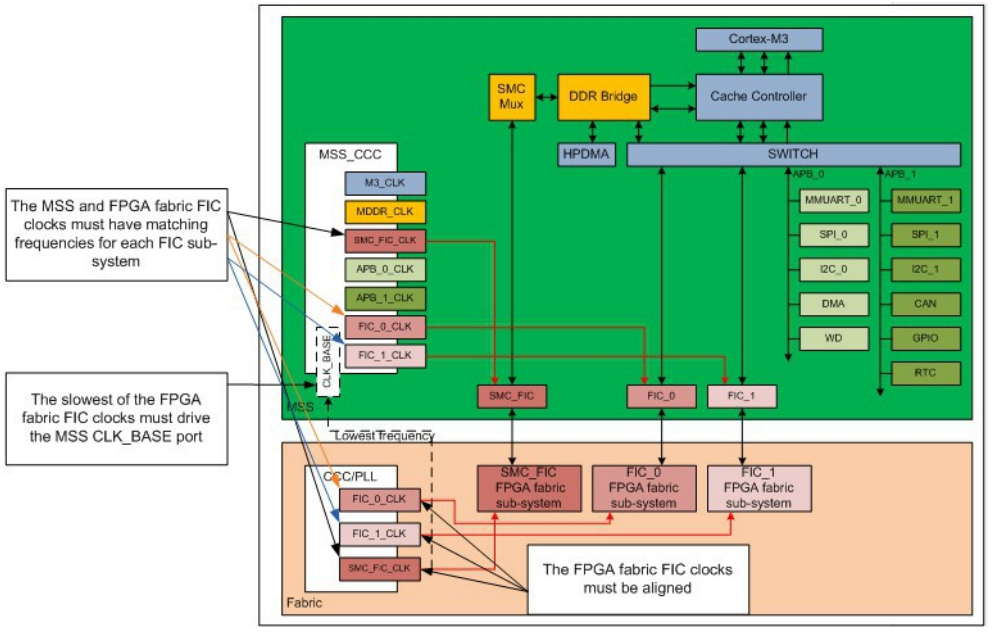
- Configure the MSS CCC FIC clocks.
- Instantiate and configure an FPGA fabric CCC core.
- Connect the clock networks for each FIC subsystem.
- Connect the MSS CLK_BASE port to the correct FPGA fabric FIC subsystem clock network.

The SmartFusion 2 architecture imposes a number of rules that must be followed for all FIC subsystems to interact between the MSS logic and FPGA fabric logic. The following are the defined rules:

- Each FPGA fabric FIC subsystem must be driven by a clock whose clock frequency matches the frequency defined, for that particular subsystem, in the MSS CCC configurator.
- All FPGA fabric FIC subsystem clocks must be precisely aligned. The clocks may be of different frequencies, but the rising edges of the slower clocks must be aligned to the rising edges of the fastest clocks.
- The FPGA fabric FIC subsystem clock with the smallest frequency must drive the MSS CLK_BASE.
- If the fabric clocks are derived from a fabric CCC (with PLL), the fabric CCC LOCK output must be connected to the MSS_CCC_CLK_BASE_PLL_LOCK port. The MSS CCC Fabric Alignment Clock Circuitry (FACC) monitors the CLK_BASE PLL LOCK signal to esnure that the CLK_BASE is stable before switching from the standby clock (clock used during device boot up) to the user-configured clock derived from CLK_BASE.

The following figure shows the subsystem clock rules.

Figure 7-1. Subsystem Clock Rules

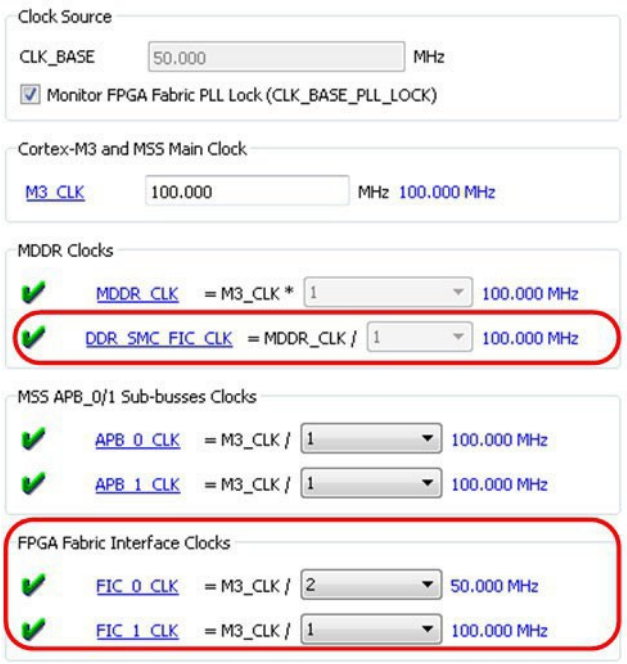


To configure the clock networks for all your FIC subsystems, perform the following steps:

7.1 Configure the MSS CCC Sub-Block (Ask a Question)

For each FIC block (FIC_0, FIC_1 and DDR_FIC/SMC_FIC) used in your design, select the clock divisors in the MSS Clock Configurator (MSS_CCC) as shown in the following figure.

Figure 7-2. MSS CCC FIC Clock Configuration



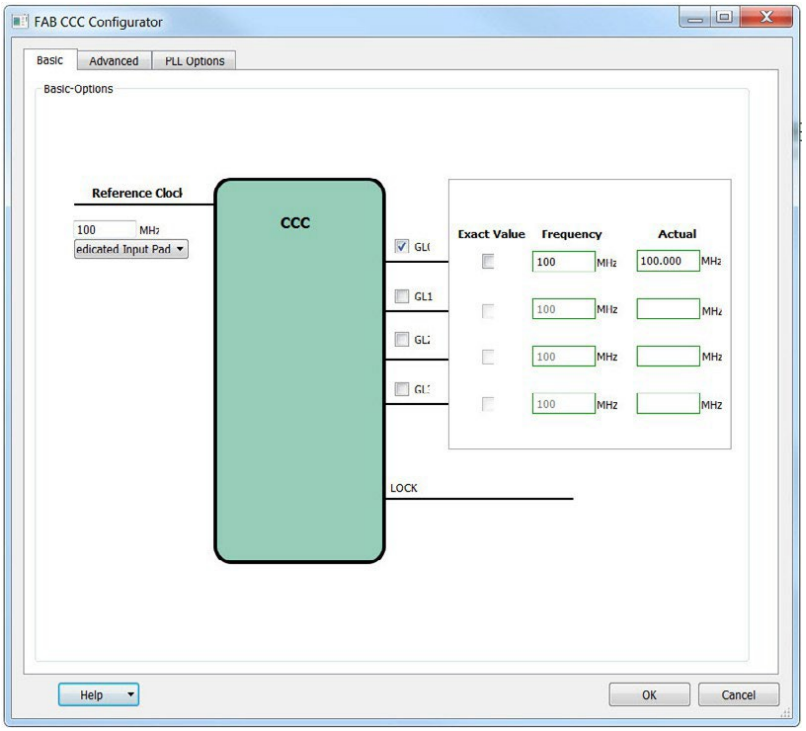
Important: The CLK_BASE field is not editable. The CLK_BASE frequency, as imposed by the SmartFusion 2 architecture, must be the minimum frequency of all FIC clock frequencies and is automatically computed by the MSS CCC configurator. For more details on CLK_BASE configuration, see Connect the FPGA Fabric FIC Subsystems Clock Networks section.

7.2 Configure the FPGA Fabric FIC Clocks (Ask a Question)

Instantiate a fabric CCC (with PLL) and configure it to satisfy the FIC subsystem clock rules described in the Configuring the FIC Subsystem Clocks section. You typically need to associate a global output (GLx) for each of the FIC clocks, specify for each output its frequency (matching the frequencies defined in the MSS CCC) and have all global outputs derived from the output of the PLL to ensure the phase alignment see the following figure.

Important: If two FIC subsystems have the same frequencies, you do not need to generate two independent global outputs from the fabric CCC, one is sufficient.

Figure 7-3. Fabric CCC with PLL



7.3 Connect the FPGA Fabric FIC Subsystems Clock Networks (Ask a Question)

Connect each fabric CCC global output GLx to the respective FIC subsystem.

7.4 Connect the MSS CLK_BASE Port (Ask a Question)

Connect the slowest of the fabric CCC global output GLx to the MSS CLK_BASE port.

7.5 Connect the MSS MCCC_CLK_BASE_PLL_LOCK Port (Ask a Question)

Connect the fabric CCC LOCK output to the MSS MCCC_CLK_BASE_PLL_LOCK port.

7.6 Timing Analysis Requirements (Ask a Question)

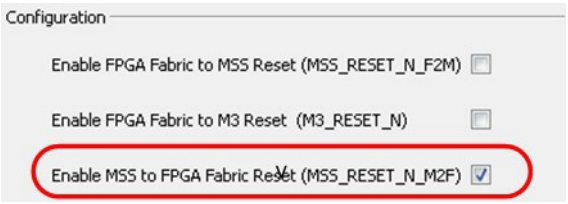
Perform post-layout static timing analysis to make sure that the design meets the timing requirements defined in the MSS CCC and FPGA fabric CCC Configurators. You may have to change M3_CLK or increase the clock ratio between the MSS and the fabric to get a design that passes static timing analysis.

8. Configuring the FIC Subsystem Reset (Ask a Question)

To configure the FIC subsystem reset, perform the following steps:

1. To expose the MSS_RESET_N_M2F port, from the MSS Configurator, configure the MSS reset sub block.

   Figure 8-1. Configure the MSS Reset Sub-Block

2. Connect the MSS_RESET_N_M2F port to all FPGA fabric FIC subsystems reset ports.

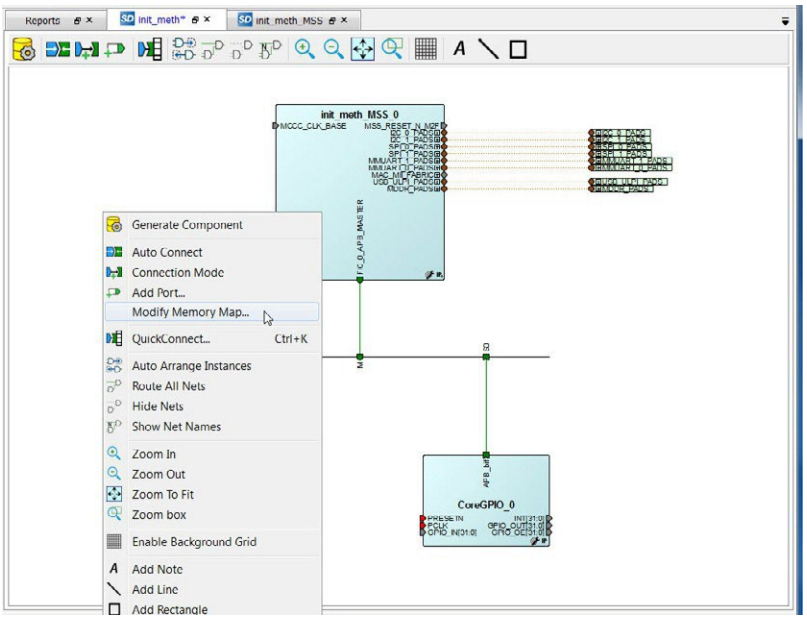
9. Configuring the System Memory Map (Ask a Question)

Each peripheral (AMBA® AXI, AHBLite and APB3 slaves) is identified by an address from the FIC subsystem's master point of view. We usually refer to the overall relationships between masters and slaves of a subsystem as the memory map of that subsystem. The memory map of a subsystem can be edited in SmartDesign. You can also view the final memory map of your system when you generate the entire system. The memory map is part of the Datasheet generated for the "root" of your design upon generation.

9.1 Configuring the Memory Map (Generic SmartDesign Behavior) (Ask a Question)

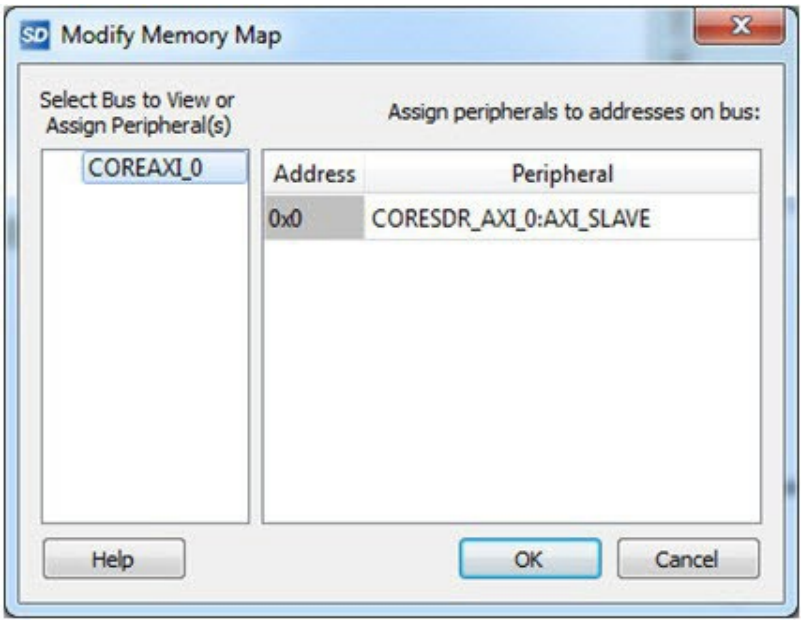
In SmartDesign, a peripheral is assigned an address on a bus based on the base address of that bus in the subsystem and the slot number on that bus times the slot size. Changing the slot number for a peripheral affects its address accordingly. Changing the slot assignment for a peripheral is done by manually by connecting the peripheral's slave BIF to a particular bus mirrored slave BIF (slot) using the SmartDesign connectivity tools available in the canvas. This is also done by editing the memory map using the **Modify Memory Map**, as shown in the following figure.

Figure 9-1. Modify Memory Map from SmartDesign Canvas



The **Modify Memory Map** dialog box appears.

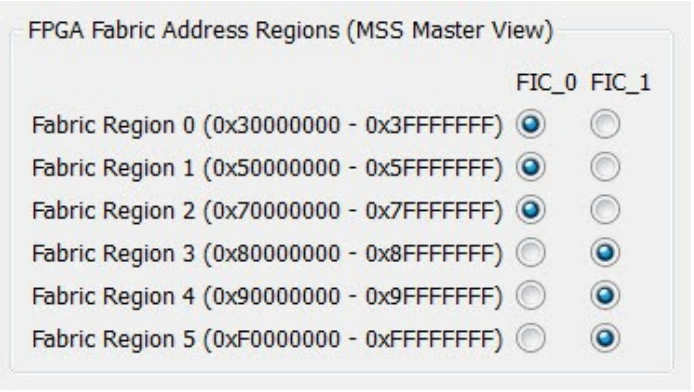
Figure 9-2. Modify Memory Map Dialog Box



9.2 Configuring the Memory Regions for the FIC Interfaces (MSS Master View) (Ask a Question)

There are six 256 MB regions defined as FIC Regions 0 to 5 in the MSS memory map. Each of these regions are allocated to the FIC_0 or FIC_1 slave interfaces in a mutually exclusive fashion. You can select to which FIC (0 or 1) slave interface you assign those regions by using the radio button next to each region in the FPGA Fabric Address Regions (MSS Master View) group box in the following figure.

Figure 9-3. FPGA Fabric Address Regions—MSS Master View



9.3 Memory Map Computation General Formula (Ask a Question)

The possible base addresses for the FIC fabric interfaces are 0x30000000, 0x50000000, 0x70000000, 0x80000000, 0x90000000 and 0xF0000000. For details on how to configure each FIC instance to map to these addresses, See Configuring the FIC Subsystems.

For AHBLite, the slot size can either be 16 MB (0x01000000) or 256 MB (0x10000000).

When using 16 MB per slot, all slots from 0 to 16 can be used; the address of client peripheral can be computed as the FIC memory region base address + (slot number × 0x01000000). In this configuration, all regions are aliases of each other as the AHBLite core does not decode the address bits [31:28]; the slots are decoded using address bits [27:24].

When using 256 MB per slot, only slots 3, 5, 7, 8, 9 and F are used; the address of client peripheral is computed as the FIC memory region base address + (slot number × 0x10000000). In this configuration, all regions uniquely address different slots as the AHBLite core decodes the slots using address bits [31:28]. In this configuration, to simplify the memory map equation, the base address is 0x00000000, instead of one of the six fixed address defined.

Example 1 (16 MB Slot Configuration):
Using memory region 0x50000000, if the peripheral is at slot number 7, then its address is: 0x50000000 + (0x7 × 0x01000000) = 0x57000000

Example 2 (256 MB Slot Configuration):
Using memory region 0x50000000, if the peripheral is at slot number 15, then its address is: 0x00000000 + (0xF × 0x10000000) = 0xF0000000

9.3.1 Viewing the Final Memory Map (Ask a Question)

At this time, Libero SoC does not generate a correct memory map for systems using the SmartFusion 2 MSS.

10. Revision History (Ask a Question)

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

Revision	Date	Description
A	12/2024	The following is the list of changes made in revision A of the document: • Migrated the document to the Microchip template. • Updated the document number to DS50003789 from UG0370.

Microchip FPGA Support

Microchip FPGA products group backs its products with various support services, including Customer Service, Customer Technical Support Center, a website, and worldwide sales offices. Customers are suggested to visit Microchip online resources prior to contacting support as it is very likely that their queries have been already answered.

Contact Technical Support Center through the website at www.microchip.com/support. Mention the FPGA Device Part number, select appropriate case category, and upload design files while creating a technical support case.

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, **650.318.8044**

Microchip Information

Trademarks

The "Microchip" name and logo, the "M" logo, and other names, logos, and brands are registered and unregistered trademarks of Microchip Technology Incorporated or its affiliates and/or subsidiaries in the United States and/or other countries ("Microchip Trademarks"). Information regarding Microchip Trademarks can be found at https://www.microchip.com/en-us/about/legal-information/microchip-trademarks.

ISBN: 979-8-3371-0178-1

Legal Notice

This publication and the information herein may be used only with Microchip products, including to design, test, and integrate Microchip products with your application. Use of this information in any other manner violates these terms. Information regarding device applications is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. Contact your local Microchip sales office for additional support or, obtain additional support at www.microchip.com/en-us/support/design-help/client-support-services.

THIS INFORMATION IS PROVIDED BY MICROCHIP "AS IS". MICROCHIP MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION INCLUDING BUT NOT LIMITED TO ANY IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE, OR WARRANTIES RELATED TO ITS CONDITION, QUALITY, OR PERFORMANCE.

IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE, INCIDENTAL, OR CONSEQUENTIAL LOSS, DAMAGE, COST, OR EXPENSE OF ANY KIND WHATSOEVER RELATED TO THE INFORMATION OR ITS USE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS IN ANY WAY RELATED TO THE INFORMATION OR ITS USE WILL NOT EXCEED THE AMOUNT OF FEES, IF ANY, THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THE INFORMATION.

Use of Microchip devices in life support and/or safety applications is entirely at the buyer's risk, and the buyer agrees to defend, indemnify and hold harmless Microchip from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights unless otherwise stated.

Microchip Devices Code Protection Feature

Note the following details of the code protection feature on Microchip products:

- Microchip products meet the specifications contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is secure when used in the intended manner, within operating specifications, and under normal conditions.
- Microchip values and aggressively protects its intellectual property rights. Attempts to breach the code protection features of Microchip products are strictly prohibited and may violate the Digital Millennium Copyright Act.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of its code. Code protection does not mean that we are guaranteeing the product is “unbreakable”. Code protection is constantly evolving. Microchip is committed to continuously improving the code protection features of our products.