

TU0308
Tutorial
ARM Cortex-M1 Embedded Processor



Power Matters.™

Microsemi Corporate Headquarters

One Enterprise, Aliso Viejo,
CA 92656 USA

Within the USA: +1 (800) 713-4113

Outside the USA: +1 (949) 380-6100

Fax: +1 (949) 215-4996

Email: sales.support@microsemi.com

www.microsemi.com

© 2017 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.

Microsemi makes no warranty, representation, or guarantee regarding the information contained herein or the suitability of its products and services for any particular purpose, nor does Microsemi assume any liability whatsoever arising out of the application or use of any product or circuit. The products sold hereunder and any other products sold by Microsemi have been subject to limited testing and should not be used in conjunction with mission-critical equipment or applications. Any performance specifications are believed to be reliable but are not verified, and Buyer must conduct and complete all performance and other testing of the products, alone and together with, or installed in, any end-products. Buyer shall not rely on any data and performance specifications or parameters provided by Microsemi. It is the Buyer's responsibility to independently determine suitability of any products and to test and verify the same. The information provided by Microsemi hereunder is provided "as is, where is" and with all faults, and the entire risk associated with such information is entirely with the Buyer. Microsemi does not grant, explicitly or implicitly, to any party any patent rights, licenses, or any other IP rights, whether with regard to such information itself or anything described by such information. Information provided in this document is proprietary to Microsemi, and Microsemi reserves the right to make any changes to the information in this document or to any products and services at any time without notice.

About Microsemi

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

Contents

1	Revision History	1
1.1	Revision 2.0	1
1.2	Revision 1.0	1
1.3	Revision 0	1
2	Introduction	2
2.1	Tutorial Requirements	2
2.1.1	Software Requirements	2
2.1.2	Hardware Requirements	2
2.1.3	Intellectual Property (IP)	3
2.2	CoreAI version 3.0.119 Licensing	3
3	Design Overview	4
3.1	Cortex-M1 System Description	4
3.2	Cortex-M1 Hardware Design Description	4
4	Getting Started	5
4.1	Download Tutorial Files	5
4.2	Install USB-to-UART Driver	6
4.2.1	Setting Up the USB-to-UART Driver	6
5	Creating Design	8
5.1	Step 1 – Create a Libero SoC Project	8
5.2	Step 2 – Create a SmartDesign Component with Libero SoC	9
5.2.1	Obtaining Different Version of Direct Core IP	11
5.2.2	Instantiate Cortex-M1 Processor	12
5.2.3	Instantiate CoreAHBLite Bus	13
5.2.4	Instantiate CoreAhbNvm	13
5.2.5	Instantiate CoreMemCtrl	13
5.2.6	Instantiate CoreAhbSram	14
5.2.7	Instantiate CoreAHB2APB	15
5.2.8	Instantiate CoreAPB	15
5.2.9	Instantiate CoreAI	15
5.2.10	Instantiate CoreUARTapb	18
5.2.11	Instantiate CoreGPIO	19
5.2.12	Instantiate RC Oscillator	19
5.2.13	Instantiate PLL	20
5.2.14	Instantiate AND2 Gate	20
5.2.15	Connect Signals Automatically in SmartDesign	20
5.2.16	Connect Signals Manually in SmartDesign	22
5.2.17	Promote Signals to Top Level	25
5.2.18	Save and Generate SmartDesign System	27
5.3	Step 3 – Create Flash Memory System	28
6	Simulation, Synthesis, and Place-and-Route	31
6.1	Step 4 – Perform Synthesis	31
6.2	Step 5 – Perform Place-and-Route	32
7	Programming	34
7.1	Step 6 – Generate Programming File with Software Code in NVM	34

7.2	Step 7 – Connect to the Target	35
7.3	Step 8 – Program the M1AFS1500 FPGA	36
8	Debugging the Application Using SoftConsole	40
8.1	Step 9 - Building the Software Application through SoftConsole	40
8.2	Step 10 - Debugging the Project	44
9	Appendix: Libero SoC Catalog Settings	48
10	Appendix: Firmware Catalog Settings	49
11	Appendix: Debugging Features in SoftConsole	50

Figures

Figure 12	Block Diagram	4
Figure 13	Tutorial Files	5
Figure 14	Windows Device Manager	7
Figure 15	New Project Dialog Box	8
Figure 16	Selecting SoftConsole as Software IDE	9
Figure 17	Select Smart Design Component	9
Figure 18	Smart Design Canvas Window	10
Figure 19	Opening Options from Catalog Section	11
Figure 20	Catalog - Options: Filter	11
Figure 21	Catalog - Options: Display	12
Figure 22	Configuring Cortex-M1	12
Figure 23	Instantiate and Configure CoreAhbNvm	13
Figure 24	Configuring CoreMemCtrl	14
Figure 25	Configuring CoreAhbSram	15
Figure 26	Configuring CoreAI (Part 1)	16
Figure 27	Configuring CoreAI (Part 2)	16
Figure 28	Configuring CoreAI (Part 3)	17
Figure 29	Configuring CoreAI (Part 4)	18
Figure 30	Configuring CoreUARTapb	19
Figure 31	Configuring the PLL	20
Figure 32	Modify Memory Map for CoreAHBLite_0	21
Figure 33	Configure Memory Map for CoreAPB_0	22
Figure 34	Connecting Signals Manually	23
Figure 35	Replace Driver Warning	23
Figure 36	Edit Slice	24
Figure 37	SmartDesign Canvas	25
Figure 38	Message while Promoting MEMADDR to Top Level	25
Figure 39	MEMADDR Naming	26
Figure 40	SmartDesign Canvas (for Advance Kit)	26
Figure 41	Design Firmware	27
Figure 42	Generating the Design	27
Figure 43	SoftConsole Folder	28
Figure 44	Flash Memory System Window	29
Figure 45	Add Data Storage Client	29
Figure 46	Data Storage Client in Flash Memory System Window	30
Figure 47	Synthesizing the Project	31
Figure 48	After Synthesis	32
Figure 49	Organizing Constraint Files	32
Figure 50	Constraints Organizer Window	33
Figure 51	Design Flow after Verify Timing	33
Figure 52	Designer Window	34
Figure 53	FlashPoint Programming File Generator	34
Figure 54	Designer Desktop	35
Figure 55	COM3 Properties	36
Figure 56	FlashPro Project Flow	37
Figure 57	Fusion Embedded Development Kit Reset System with SW1	37
Figure 58	Fusion Advanced Development Kit Board Reset System with SW1	38
Figure 59	HyperTerminal Window	38
Figure 60	HyperTerminal Display	39
Figure 61	Invoking SoftConsole from Libero SoC	40
Figure 62	SoftConsole Workspace	41
Figure 63	Import Window	41
Figure 64	Linker Flags	42
Figure 65	GNU C Compiler	43

Figure 66	Changes Made to Coreai_cfg.h	44
Figure 67	Debug Configurations	45
Figure 68	Debugger Commands	45
Figure 69	Confirm Perspective Switch	46
Figure 70	Debug Perspective	46
Figure 71	Set Breakpoint	47
Figure 72	Catalog – Options	48
Figure 73	Setting Repositories	48
Figure 74	Setting the Vault Location	48
Figure 75	Firmware Catalog Settings	49

Tables

Table 76	Description of Tutorial Files	5
Table 77	Description of Software Tutorial Files	5
Table 78	Power and USB Connections	6
Table 79	Programming Interface Connections	7
Table 80	Connecting Signals Manually	23
Table 81	VCC and GND Connections	24
Table 82	Signals to Promote to Top Level	25

1 Revision History

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

1.1 Revision 2.0

The following is a summary of the changes in revision 2.0 of this document.

- Information about CoreAI licensing was updated. For more information, see [CoreAI version 3.0.119 Licensing](#), page 3.
- Information about extracting source files was removed.

1.2 Revision 1.0

The following is a summary of the changes in revision 1.0 of this document.

- Modified Introduction section and Software Requirements section (SAR 38531).
- Modified Intellectual Property (IP) section (SAR 38531).
- Modified Instantiate Cortex-M1 Processor section (SAR 38531).
- Replaced Figure 11 (SAR 38531).
- Replaced Figure 17 (SAR 38531).
- Modified Instantiate CoreUARTapb section (SAR 38531).
- Replaced Figure 19 and modified Instantiate CoreGPIO section (SAR 38531).
- Modified Connect Signals Manually in SmartDesign section (SAR 38531).
- Modified Promote Signals to Top Level section (SAR 38531).
- Replaced Figure 30 (SAR 38531).
- Replaced Figure 32 (SAR 38531).
- Modified headings from Step 4 - Perform Synthesis till Step 10 - Debugging the Project (SAR 38531).
- Replaced Figure 49 (SAR 38531).

1.3 Revision 0

Revision 0 was the first publication of this document.

2 Introduction

This tutorial describes how to create a Cortex-M1 processor system that runs on one of the Fusion embedded and development kit boards provided by Microsemi system-on-chip (SoC) Products Group. This design can be used as a starting point for developing the Cortex-M1 embedded system targeting the ProASIC[®]3, IGLOO[®], and Fusion[®] FPGAs of the SoC products group. As this tutorial targets one of the Fusion devices, you need to be familiar with the features and architecture of Fusion. You can download the Fusion datasheet and user's guide from the Microsemi website:

- www.microsemi.com/soc/documents/Fusion_DS.pdf
- www.microsemi.com/soc/documents/Fusion_UG.pdf

After completing this tutorial, you will know the hardware design flow for creating a Cortex-M1 embedded system using Libero[®] system-on-chip (SoC) v10.0 (or later) and SmartDesign tools. This includes the following design steps:

- Instantiating and configuring the Cortex-M1 processor, memory, and peripherals in SmartDesign
- Connecting peripherals and defining the address map in SmartDesign
- Automated generation of the DirectCore RTL
- Synthesis, place-and-route of hardware design, and generating FPGA programming image
- Programming a Microsemi FPGA with Cortex-M1 system ready for software design

For this tutorial, use the binary file from previously developed software. The binaries are written into the memory on the target development kit along with the programming bit files for the hardware designed in this tutorial.

This is followed by the software development flow for the Cortex-M1 processor.

2.1 Tutorial Requirements

2.1.1 Software Requirements

This tutorial requires the following software installed on your PC:

- Libero SoC v10.0 (or later) can be downloaded from the link below:
www.microsemi.com/soc/download/software/libero/default.aspx.

The instructions are based on the Libero SoC v10.0 (or later) software along with the corresponding Synplify and ModelSim OEM software installed on your PC. If you are using a different version, some steps and screen-shots may be different.

- Microsemi SoftConsole v3.3 (or later), which is installed as a part of Microsemi Libero SoC installation or can be downloaded from
www.microsemi.com/soc/download/software/softconsole/default.aspx.
- USB Drivers for USB to UART connection: www.microsemi.com/soc/documents/CP2102_driver.zip.

2.1.2 Hardware Requirements

This tutorial requires the following hardware:

- Fusion Embedded Development Kit Board (M1AFS-EMBEDDED-KIT board)
 - Low-cost programming stick
 - 2 USB cables (USB to mini-USB)
 - PC with 2 USB ports
- Fusion Advanced Development Kit Board (M1AFS-ADV-DEV-KIT board)
 - Low-cost programming stick
 - 2 USB cables (USB to mini-USB)
 - 9 V power supply (provided with kit)
 - PC with 2 USB ports

Everything (except the PC) is provided with the development kit. Refer to the Design Hardware page for more information: www.microsemi.com/soc/products/hardware/default.aspx

2.1.3 Intellectual Property (IP)

This tutorial is based on the DirectCore IP listed below. If you do not have these DirectCore IP versions in the SmartDesign Catalog, you may observe different behavior from what is described in this tutorial.

- Cortex-M1 version 3.1.101
- CoreAHLite version 3.1.102
- CoreAPB version 1.1.101
- CoreAHB2APB version 1.1.101
- CoreAhbNvm version 1.4.110
- CoreAhbSram version 1.4.104
- CoreMemCtrl version 2.0.105
- CoreUARTapb version 5.1.102
- CoreGPIO version 1.2.103

2.2 CoreAI version 3.0.119 Licensing

CoreAI is available with any Libero license. If you do not have the license, you can get it at www.microsemi.com/soc/portal/default.aspx?v=0

1. Sign-in and select **Licenses & Registration** on the left navigation.
2. Click **Request Free License** and select **Libero Evaluation or Silver license**. Follow the instructions in the e-mail sent to you to set up the license on your machine.

3 Design Overview

3.1 Cortex-M1 System Description

The design contains a Cortex-M1 processor system running on a Microsemi Fusion FPGA, which contains an analog block for monitoring analog signals. The system measures various voltages, currents, and temperatures on the target board, processes the sample data, and sends the result over UART.

There is a potentiometer (POT) on the board to change the analog voltage being sampled. In this tutorial you will communicate with the target using HyperTerminal.

3.2 Cortex-M1 Hardware Design Description

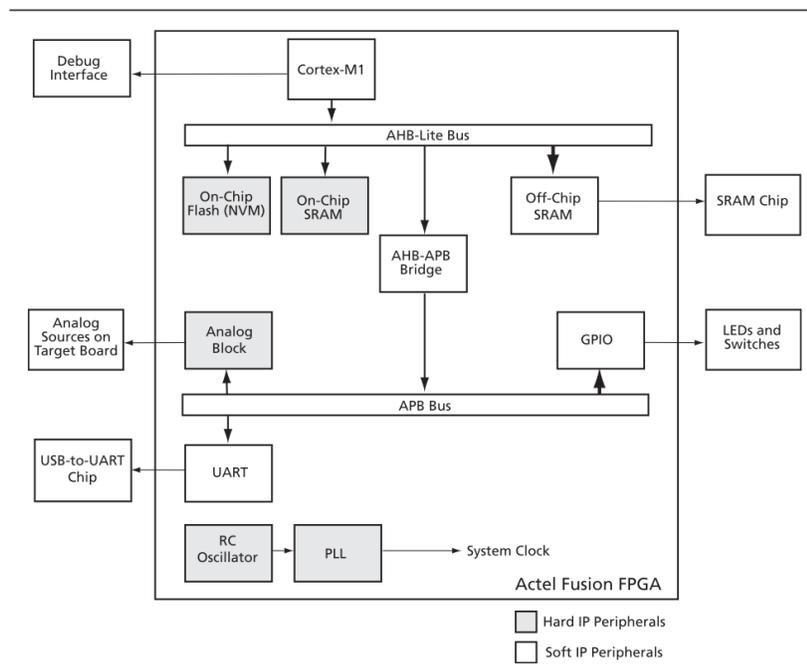
The Cortex-M1 processor system uses CoreAI, which allows the processor to configure, control, and interact with the Analog Block inside the Fusion FPGA. The UART in the system connects to an off-chip USB-to-UART chip, which allows you to communicate with the target system via a COM port on your PC (using HyperTerminal). Also included are 4 output bits to LEDs and 2 input bits from push-buttons or DIP switches (depending on the target board).

The Fusion Advanced Development Kit and the Fusion Embedded Development Kit boards contain a Fusion AFS1500 device that has 1 MB of embedded flash memory (also referred to as nonvolatile memory, or NVM) and 30 KB of internal SRAM. The Fusion Embedded Development Kit Board has 1 MB of SRAM, comprised of two 4 Mbit × 16 bit chips. The Fusion Advanced Development Kit Board has 2 MB of SRAM, comprised of two 1 Mbit × 16 bit chips. This hardware design connects to all of these memories, not necessarily using the entire memory space of each device.

Microsemi Fusion devices have an on-chip 100 MHz RC oscillator. You can feed this clock source to a PLL inside the Fusion device that modifies the clock frequency. The output of the PLL is the system clock. This design enables the JTAG debug interface of the Cortex-M1 processor for software debugging. The software debugging has been explained in the [Debugging the Application Using SoftConsole](#), page 40 section.

The block diagram for the design is shown in the following figure.

Figure 1 • Block Diagram



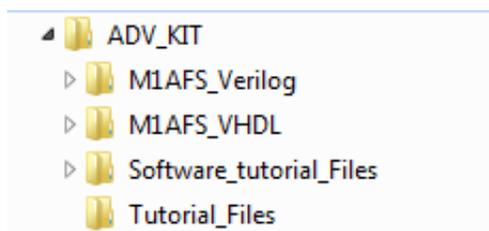
4 Getting Started

4.1 Download Tutorial Files

Before starting the tutorial, you need to download the tutorial files from the website for the board you are targeting. You can download the tutorials from Microsemi website at: (www.microsemi.com/soc/products/hardware/default.aspx).

On the web page of the target board, click on the CortexM1_Proc_Tutorial zip file under the ARM Cortex-M1 Embedded Processor Tutorial document. The supported subdirectories for every board are placed in this zip folder. The directory structure is similar for each board. The following figure shows the sample directory structure for the Fusion Advance Development Kit Board (M1AFS-ADV-KIT).

Figure 2 • Tutorial Files



The M1AFS_Verilog and M1AFS_VHDL subdirectories contain the complete VHDL and Verilog projects for this tutorial design, for reference. The Tutorial_Files subdirectory contains the required source files you need to complete this tutorial and that are described in the following table. The names of the files are based on the targeted board.

Table 1 • Description of Tutorial Files

File Name	Description
M1AFS_EMB/ADV_KIT	Pin constraints for the Fusion Embedded/Advanced Development Kit Board
Cortex M1_Tutorial.HEX	Intel-Hex file that contains the Cortex-M1 software code
M1AFS_EMB/ADV_KIT_<VERILOG/VHDL>.PDC	Constraint to put JTAG reset and clock net on global

The Software_tutorial_files subdirectory contains the source files required to complete the Software tutorial flow. The following table has a description of these files.

Table 2 • Description of Software Tutorial Files

File Name	Description
M1<FPGA>_TUT_TOP.PDB	FPGA programming file containing Cortex-M1 design tutorial (hardware design). It can be found in respective folders of Verilog and VHDL
tutorial.h	Header file with memory map and interrupt assignments targeted for Cortex-M1 FPGA design tutorial
main.c	Top C source file
boot-from-actel-coreahbnvm.ld	Linker script modified to work with Fusion-based tutorial designs

4.2 Install USB-to-UART Driver

At the end of the tutorial, you can program the Fusion FPGA and communicate with the target board. To complete the tutorial, you need to ensure that the USB-to-UART drivers are installed on your PC and identify which COM port the USB port is associated with. Follow the steps given below before starting the tutorial to ensure your PC is properly configured and connected to the target board.

4.2.1 Setting Up the USB-to-UART Driver

Note: You should have Admin privileges to install the USB-RS232 drivers on your PC.

To install drivers for the USB to RS232 Bridge:

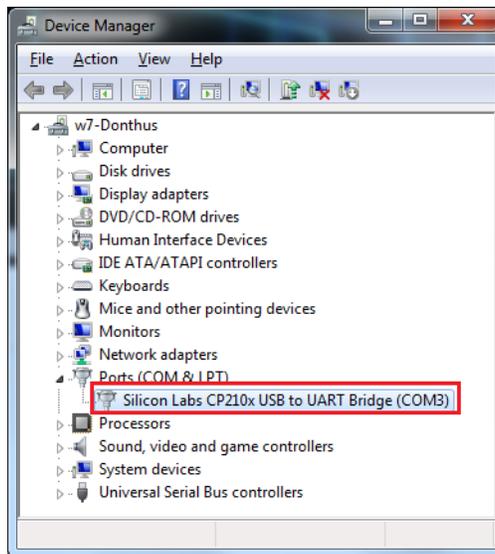
1. Download the USB to RS232 bridge drivers from www.microsemi.com/soc/documents/CP2102_driver.zip.
2. Unzip the cp2102_drivers.zip file.
3. Double-click (Run) the *.exe file.
4. Accept the default installation location and click **Install**. Click **Continue Anyway** if prompted.
5. When the installation is complete, click **OK**. The Ports (COM & LPT) section of the Device Manager lists Silicon Labs CP210x USB to UART Bridge under the Ports section of Device Manager.
6. Make sure that a terminal emulation program such as HyperTerminal, which is included with Windows, is installed on your PC.
7. Connect USB and power cables according to the following table.

Table 3 • Power and USB Connections

Board	Power and USB Connections
M1AFS-EMBEDDED-KIT	Make sure you have the J40 set to use USB power (not V5IN). Connect a USB cable to J2, which provides power and the USB-to-UART communication. Make sure JP10 is set to use the 1.5 V external regulator (connect pins 2 and 3).
M1AFS-ADV-DEV-KIT	Connect the 9 V power supply, provided with the kit, to J3 on the board. Connect a USB cable to J2, (this provides the USB-to-UART communication) and make sure SW7 is ON to supply power to the board.

If Windows prompts you to connect to Windows Update, select **No, not at this time** and click **Next**.

8. Select **Install the software automatically** (recommended) and click **Next**. Once installation is completed, click **Finish**. Repeat the driver installation steps again (if prompted). Click **Continue Anyway** if prompted.
9. Open the Windows Device Manager by selecting **Start > Control Panel > System > Hardware > Device Manager**. Expand the Ports (COM and LPT) section and take note of the COM port assignment for the SFE USB to RS232 Controller (in [Figure 3](#), page 7 it is assigned to COM3). If you do not see SFE USB to RS232 Controller in the Device Manager ([Figure 3](#), page 7), you may need to reboot your PC.

Figure 3 • Windows Device Manager

10. Connect the FlashPro programming interface according to the following table.

Table 4 • Programming Interface Connections

Board	Flash Pro Programming Interface Connection
M1AFS-EMBEDDED-KIT	Connect the FlashPro3 Low-Cost Programming Stick (LCPS) to J1.
M1AFS-ADV-DEV-KIT	Connect a USB cable between the LCPS and the host machine.

11. If you are prompted to install drivers for the FlashPro hardware, refer to the FlashPro User's Guide at http://www.microsemi.com/soc/documents/flashpro_ug.pdf.

5 Creating Design

5.1 Step 1 – Create a Libero SoC Project

1. Launch Libero SoC v10.0 or later.
2. From the **Project** menu, select **New Project**. Enter the information shown below in the Libero **New Project** dialog box.

Under **Project**:

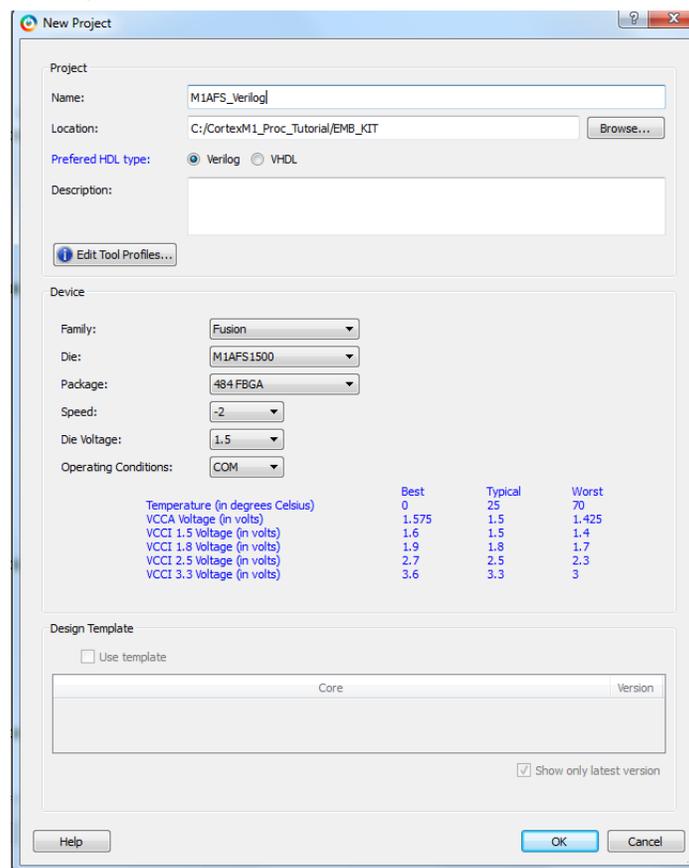
- Name: M1AFS_Verilog/VHDL
- Location: <.> (Example: C:\CortexM1_Proc_Tutorial\EMB_KIT)
- Preferred HDL Type: Choose VERILOG or VHDL

Under **Device**:

- Family: Fusion
- Die: M1AFS1500
- Package: 484 FBGA

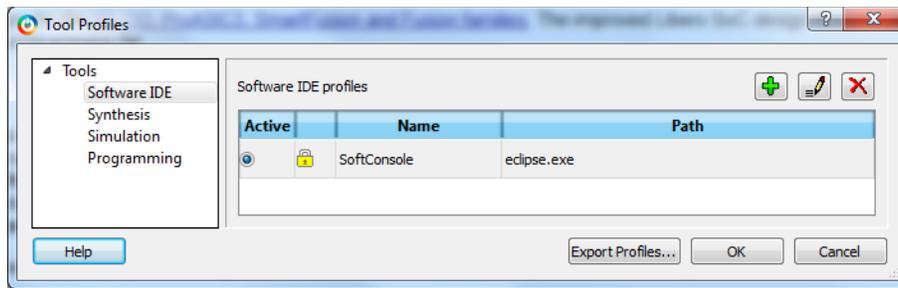
Leave others as default and click **OK**.

Figure 4 • New Project Dialog Box



3. Click **Edit Tool Profiles** and add SoftConsole by clicking **Software IDE**, as shown in the following figure.

Figure 5 • Selecting SoftConsole as Software IDE



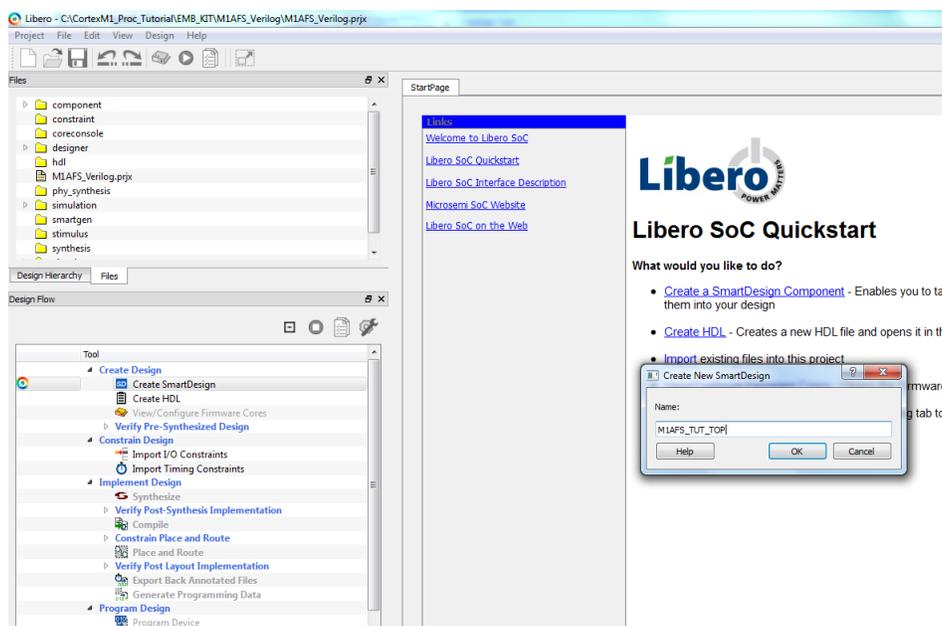
4. After adding the Profile, click **OK** to close the **Add Profile** dialog window.
 5. Repeat steps above for Synthesis, Simulation, and Programming and then click **OK** to close the **Tool Profiles** dialog window.
 6. In order to import the PDC files, use **File > Import files** and Browse to the following directory: CortexM1_Proc_Tutorial\<board>\Tutorial_Files. Also select the type of the file as ***.pdc**.
 7. Select **M1AFS_EMB_KIT.PDC** and **M1AFS_EMB_KIT_<VERILOG/VHDL>.PDC** or **M1AFS_ADV_KIT.PDC** and **M1AFS_ADV_KIT_<VERILOG/VHDL>.PDC**, depending on the board you are targeting and the project language used (Verilog or VHDL).
- Note:** These PDC files contain pin assignments for the FPGA design and are specific to this board. This design does not need timing constraints file (SDC) because the internal clock source from the RC oscillator is constrained automatically by SmartTime.
8. Click **Open**. If the system prompts to organize file constraints, click **NO**. You should see the file you selected under **Constraint** in the **Files** section.

5.2 Step 2 – Create a SmartDesign Component with Libero SoC

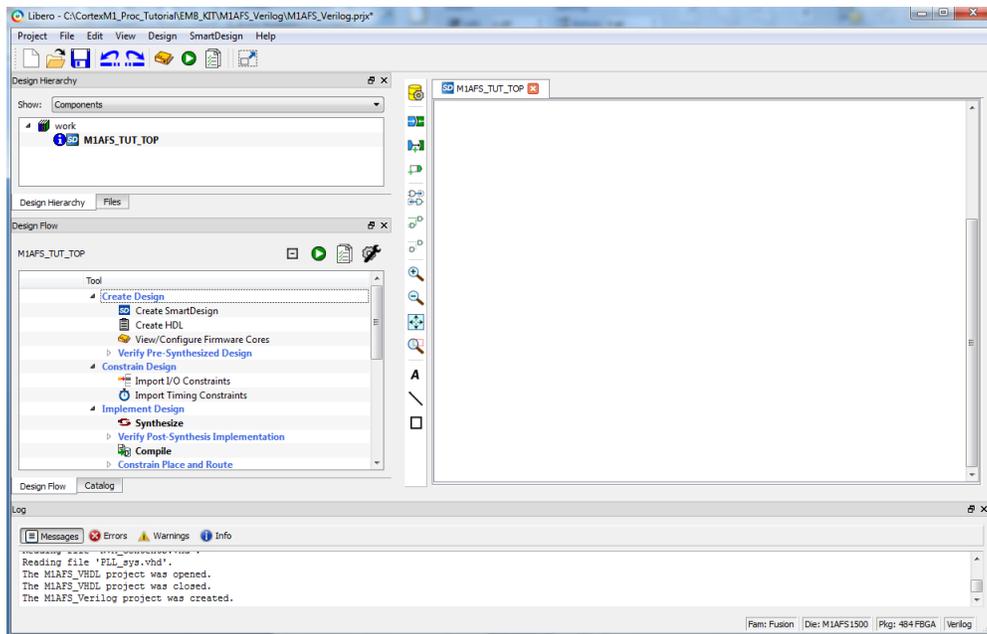
Follow the steps below to create a SmartDesign component for your hardware design:

1. In the **Design Flow** tab right-click **Create SmartDesign** and click **Run**. The **Create New SmartDesign** window is displayed. Enter name of the design as **M1AFS_TUT_TOP** as shown in the following figure.

Figure 6 • Select Smart Design Component



2. Click **OK**. A blank canvas opens in SmartDesign as shown in the following figure. The **M1AFS_TUT_TOP** is created under **Design Hierarchy > work** tab.

Figure 7 • Smart Design Canvas Window

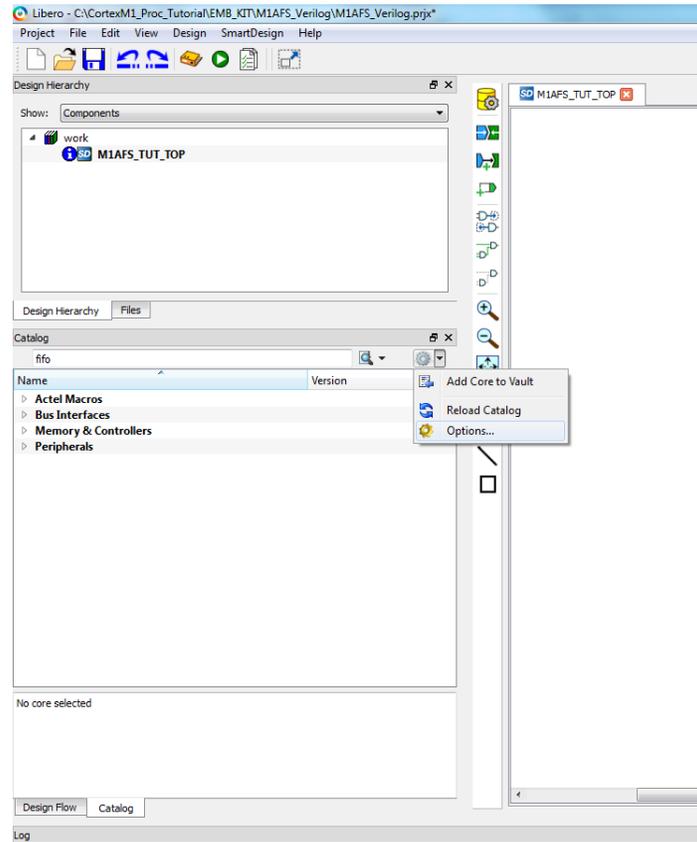
In the following steps use SmartDesign to create a Cortex-M1 system, as shown in [Figure 11](#), page 12. When the instructions tell you to instantiate a component, instantiate it in the SmartDesign Canvas.

When you instantiate components in SmartDesign, it is recommended to use the same version of DirectCoreIPs that are mentioned in this tutorial.

5.2.1 Obtaining Different Version of Direct Core IP

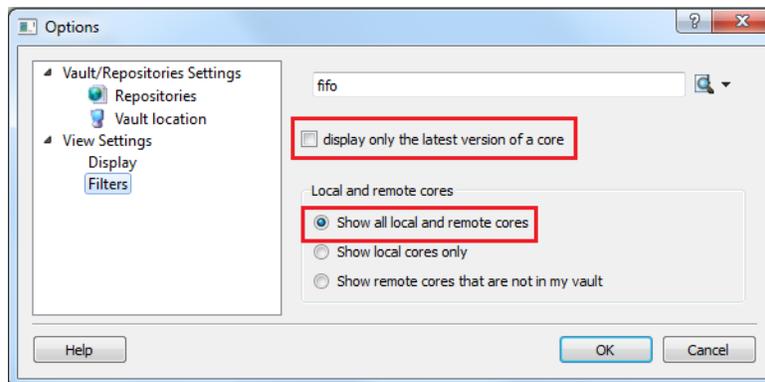
1. Click **Options** in **Catalog** section as shown in the following figure.

Figure 8 • Opening Options from Catalog Section



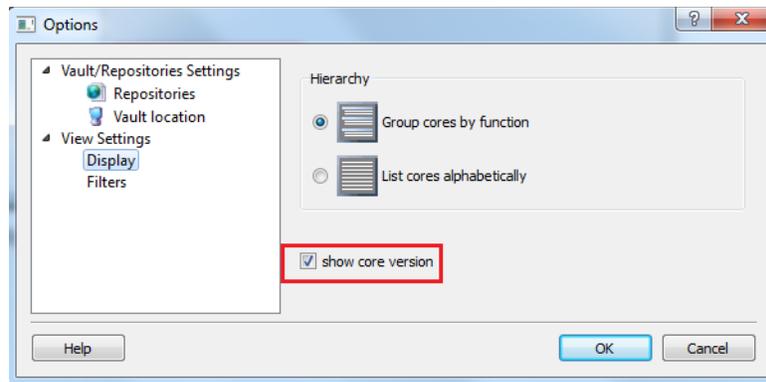
2. Click **Filters** and clear the **display only the latest version of a core** check-box and select **show all local and remote cores** option as shown in the following figure.

Figure 9 • Catalog - Options: Filter



3. Click **Display** and select **show core version** check-box as shown in the following figure.

Figure 10 • Catalog - Options: Display



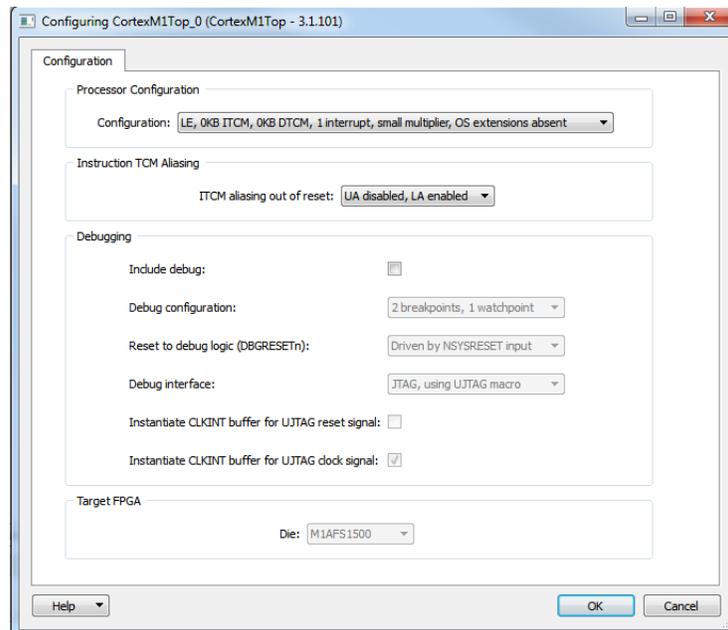
5.2.2 Instantiate Cortex-M1 Processor

The Cortex-M1 processor is a 32-bit soft ARM® processor designed for implementation in an FPGA. Use the following to instantiate the Cortex-M1 processor:

1. From **Catalog**, select **Processors**.
2. Right-click **Cortex-M1 Version 3.1.101** and then selects **Instantiate in M1AFS_TUT_TOP**.
3. If the specific IP version is grayed out or not highlighted in the tool, right-click the specific version and click **Download**.
4. Select **Include Debug** and select **JTAG using UJTAG macro** from **Debug Interface** drop-down list. This allows software debugging over JTAG using SoftConsole® development tool. Select default settings for other options.

Note: If you want to use third-party tools (Keil MDK-ARM, IAR Embedded Workbench, ARM Real View, etc.), select **Real View JTAG** from **Debug Interface** drop-down list. However, keep in mind that this document assumes that you are using SoftConsole development tool.

Figure 11 • Configuring Cortex-M1



5. Click **OK**. You should see an instance of the Cortex-M1 processor on the SmartDesign canvas.

5.2.3 Instantiate CoreAHLite Bus

The AHB-Lite bus is a multi-master 32-bit bus (data and address) which allows you to connect upto 2 masters and 16 slave peripherals, where each peripheral consumes one slot (numbered 0 through 15). Each slot is 256 MBs in the processor's memory space (for a total of 4 GBs).

1. From **Catalog**, select **Bus Interfaces**.
2. Instantiate **CoreAHLite** version 3.1.102 with the slots 0,1,2,12 enabled (select these slots under Enable Master 0 AHBlite Slave slots and Enable Master 1 AHBlite Slave slots). Choose default settings for other options.

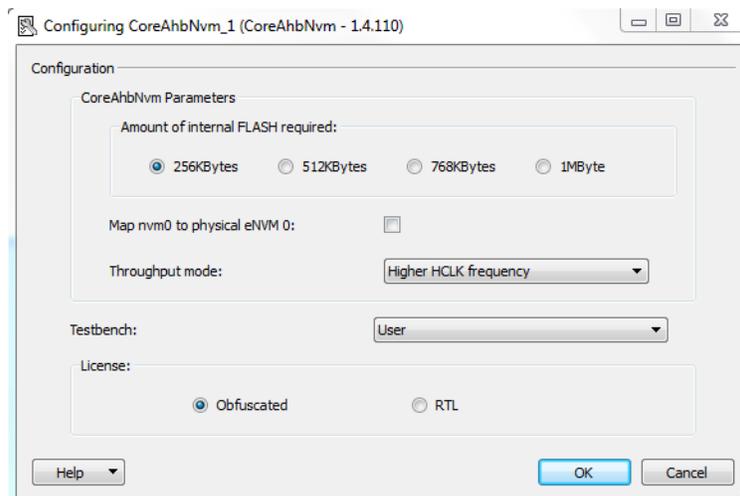
5.2.4 Instantiate CoreAhbNvm

CoreAhbNvm provides the processor with an AHB interface to access the internal **NVM** (nonvolatile memory, also known as embedded flash memory) of the Fusion device. The M1AFS1500 device has 1 MB of internal NVM composed of 4 embedded flash memory blocks, each 256 KBs in size. However, for this design you need only 1 embedded flash memory block, 256 KBs.

Follow the steps below to instantiate the CoreAhbNvm.

1. From **Catalog**, select **Memory & Controllers**.
2. Instantiate and configure CoreAhbNvm version 1.4.110 with a size of 256 KBs, as shown in the following figure.

Figure 12 • Instantiate and Configure CoreAhbNvm



5.2.5 Instantiate CoreMemCtrl

CoreMemCtrl creates an interface to off-chip SRAM and/or flash with shared data and address buses. The AHB slot is divided in half with the flash at the bottom half and SRAM at the upper half of the slot. By setting the Remap input to '1', you can swap the flash and SRAM locations. For this design, you only need to use the SRAM interface. The system clock in this design is 20 MHz, which has a period of 50 ns. The asynchronous SRAM has an access time of 10 ns. Therefore, you can set the wait states to their minimum values: 0 for read wait states and 1 for write wait states.

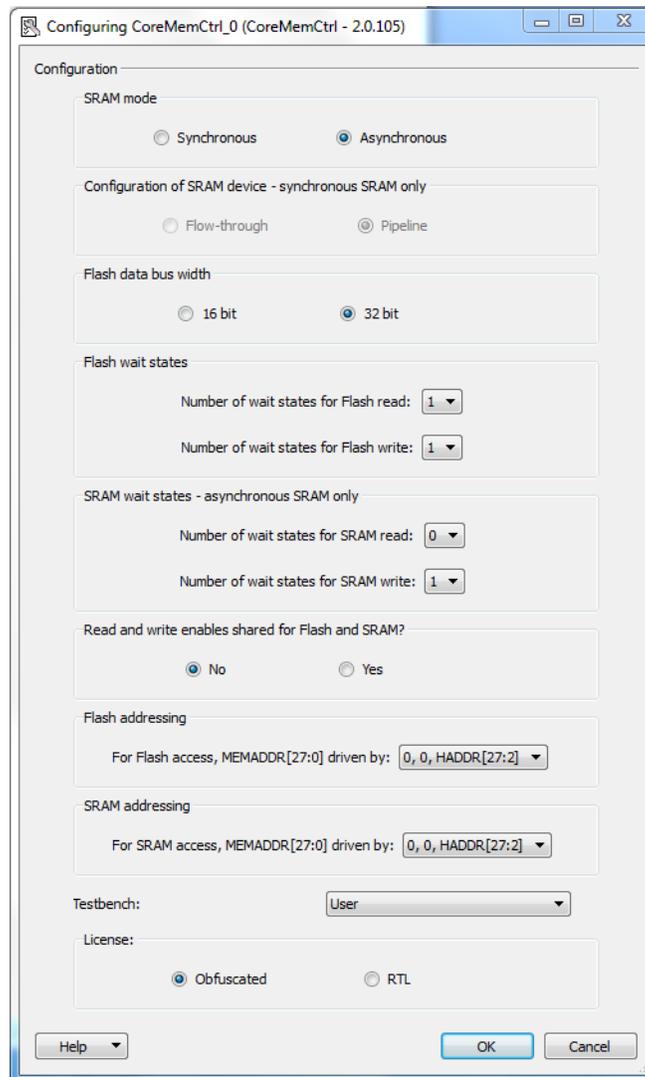
1. From **Catalog**, select **Memory & Controllers**.
2. Instantiate **CoreMemCtrl** version **2.0.105** to interface to the off-chip asynchronous SRAM on the development board.
3. Make the following selections in the CoreMemCtrl configuration window:
 - SRAM mode: Asynchronous
 - Number of wait states for SRAM read: 0
 - Number of wait states for SRAM write: 1
 - Read and write enables shared for Flash and SRAM: No

The flash and SRAM addressing options determine which bit of the AHB-Lite bus is connected to bit 0 of the off-chip memory. The default value of "0, 0, HADDR[27:2]" means that the off-chip memory address is

word addressed (32-bit word). The other settings refer to half-word and byte addressing. Leave the other options as default setting since this design does not use this flash interface.

The corresponding CoreMemCtrl settings are shown in the following figure.

Figure 13 • Configuring CoreMemCtrl

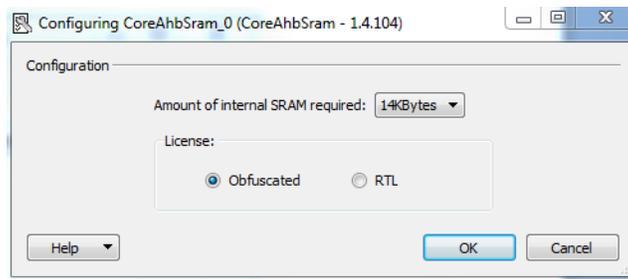


5.2.6 Instantiate CoreAhbSram

CoreAhbSram provides the processor with an AHB interface to access the internal SRAM of the Fusion device. The Fusion M1AFS1500 device has 30 KBs of internal SRAM consisting of 60 RAM blocks each 0.5 KB in size. The Cortex-M1 uses 2 KBs and the CoreUARTApb (which you instantiate later) uses 1 KB in FIFO mode. Therefore, CoreAHBS RAM should not be more than 27KBs. You need to configure it for 14 KBs. Follow the steps below to instantiate the CoreAhbSram.

1. From **Catalog**, select **Memory & Controllers**.
2. Instantiate and configure **CoreAhbSram version 1.4.104** with a size of 14 Kbytes. The CoreAhbSram settings window is shown in the following figure.

Figure 14 • Configuring CoreAhbSram



5.2.7 Instantiate CoreAHB2APB

The CoreAHB2APB is a bridge from the AHB bus to APB bus. It has an AHB slave which consumes one AHB slot and an APB master which can master an APB bus. Read and write transfers on the AHB bus are converted to APB transfers to the APB bus. Follow the steps below to instantiate the CoreAHB2APB.

1. From **Catalog**, select **Bus Interfaces**.
2. Instantiate **CoreAHB2APB version 1.1.101** to bridge to the APB bus with default settings.

5.2.8 Instantiate CoreAPB

The CoreAPB bus is a 32-bit bus (data and address) which allows you to connect up to 16 slave peripherals where each peripheral consumes one slot (numbered 0 through 15). Each slot is 16 MBs in the processor's memory space for a total of 256 MBs. Due to the performance, typically slower, low priority peripherals are placed in the APB bus. Follow the steps below to instantiate the CoreAPB.

1. From **Catalog**, select **Bus Interfaces**.
2. Instantiate the **CoreAPB version 1.1.101** bus with all slots enabled. Leave others as default.

Note: You can disable slots that you are not using. For this tutorial, leave them all enabled.

5.2.9 Instantiate CoreAI

CoreAI allows the Cortex-M1 processor to access and control the Analog Block in the Fusion device. You will configure this instance of CoreAI to interface with the analog features on the target development board. The frequency of the ACM clock in the Analog Block must be 10 MHz or less. The PCLK (clock of the APB bus) is divided down to create the ACM clock. The appropriate divider value is calculated based on the PCLK frequency parameter. For this design, the PCLK frequency is 20 MHz.

1. From **Catalog**, select **Peripherals**.
2. Instantiate **CoreAI version 3.0.119** to give the Cortex-M1 processor access to the Analog Block in the Fusion device.
3. Configure CoreAI according to the list of settings below and leave the other settings as default.
 - APB data bus width (in bits): 32
 - PCLK frequency (up to nearest MHz): 20
 - Interrupt output: Disabled (tied low)
 - Analog Quad 0
 - AV0 input: Voltage Monitor: 0 V to 4 V
 - AC0 input: Current Monitor
 - Analog Quad 1
 - AV1 input: Voltage Monitor: 0 V to 2 V
 - AC1 input: Current Monitor
 - Analog Quad 2
 - AT2 input: Temperature Monitor
 - Analog Quad 4
 - AC4 input: Voltage Monitor: 0 V to 4 V

Figure 15, page 16 through Figure 18, page 18 show the instance of CoreAI with the corresponding parameters set.

Figure 15 • Configuring CoreAI (Part 1)

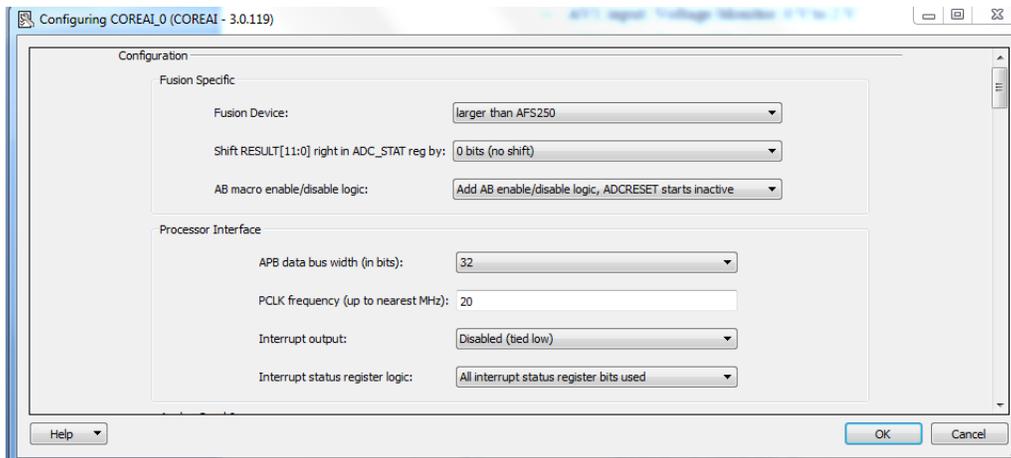


Figure 16 • Configuring CoreAI (Part 2)

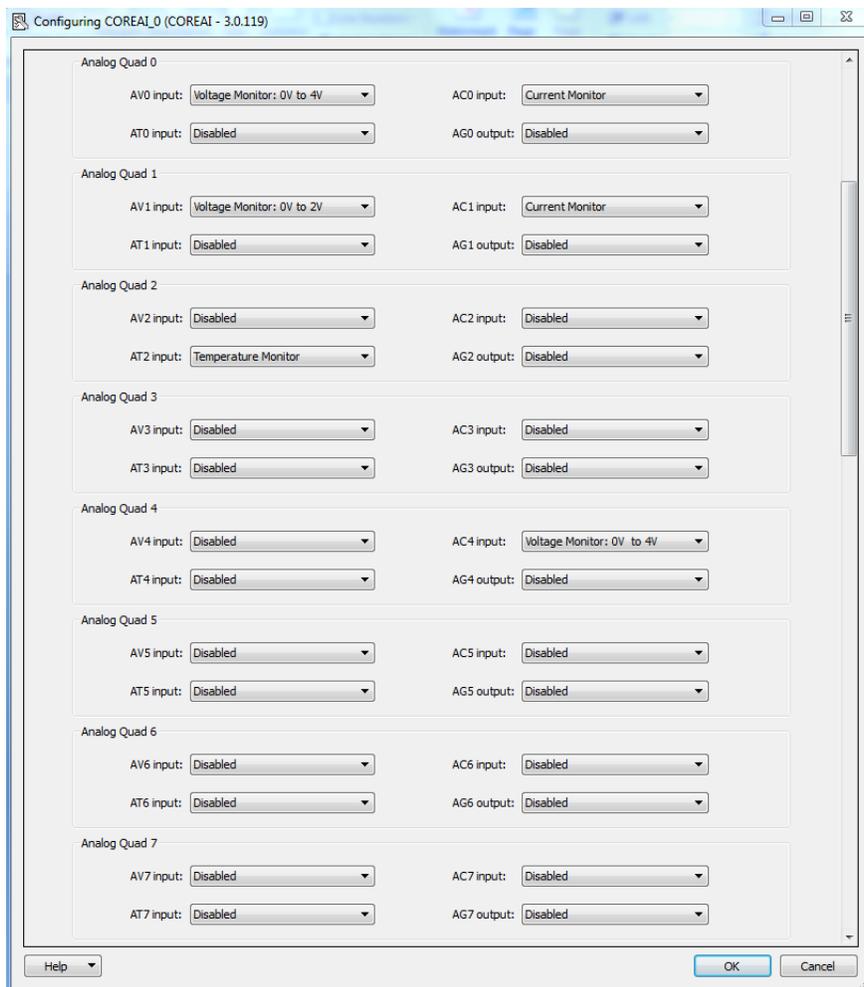
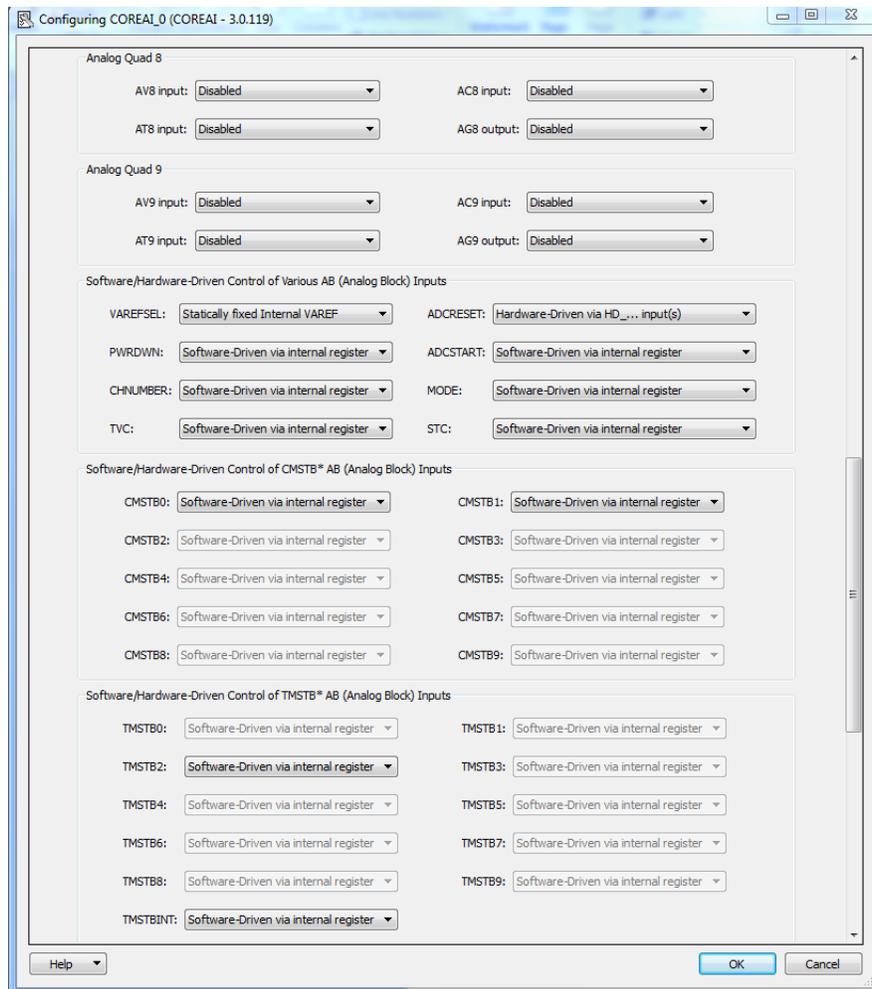
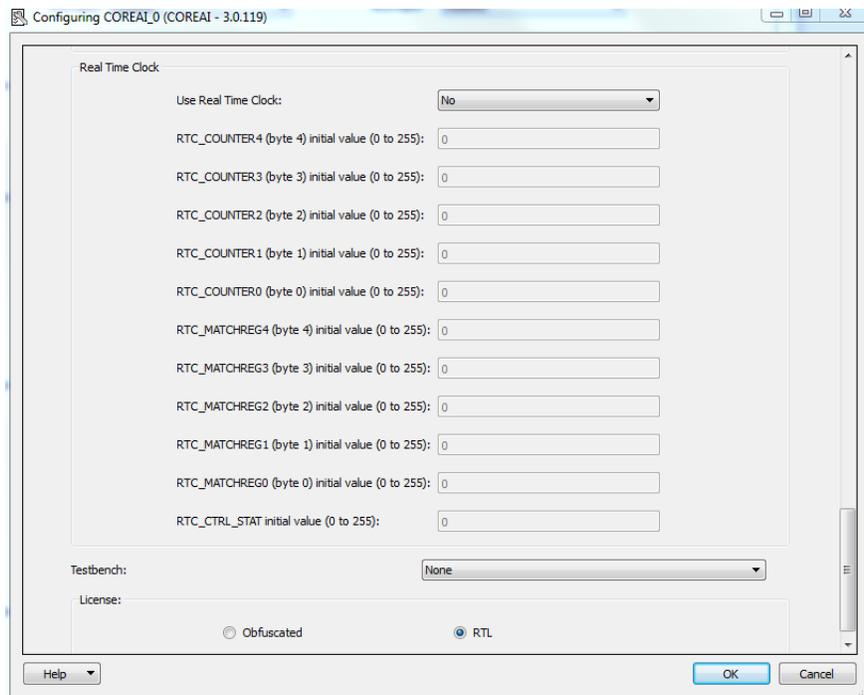


Figure 17 • Configuring CoreAI (Part 3)



Note: Many of the parameters are software driven, which means that the Cortex-M1 processor can change these parameters by writing to registers.

Figure 18 • Configuring CoreAI (Part 4)

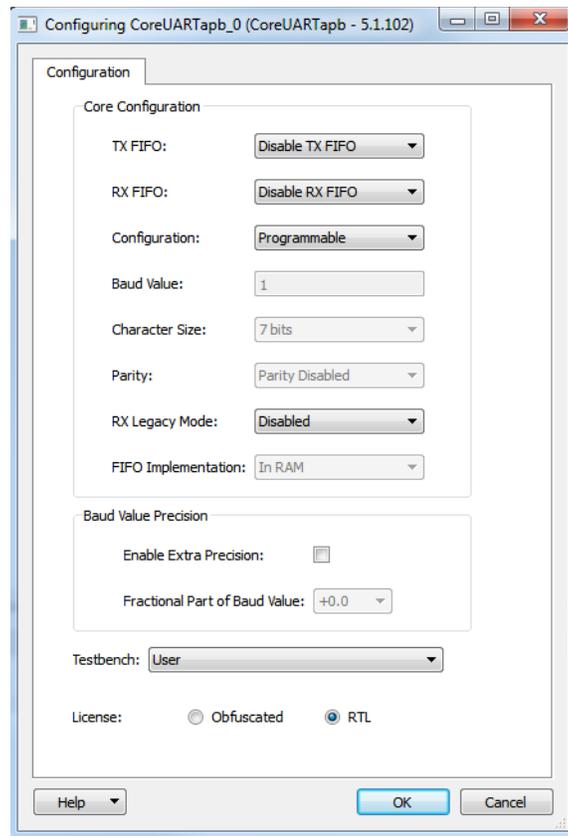
5.2.10 Instantiate CoreUARTapb

CoreUARTapb is a configurable UART peripheral with an APB slave interface. Follow the steps below to instantiate the CoreUARTapb.

1. From **Catalog**, select **Peripherals**.
2. Instantiate and configure **CoreUARTapb version 5.1.102** with the following details:
 - TX FIFO: Enable TX FIFO
 - RX FIFO: Enable RX FIFO
 - Configuration: Programmable

The CoreUARTapb configuration is shown in the following figure.

Figure 19 • Configuring CoreUARTapb



The Baud value, Character Size and Parity parameters are grayed out since **Configuration** is set to Programmable. These details can be modified by the processor when the UART is initialized. The following equation is used to calculate the Baud Value parameter based on the desired baud rate (57600) and system clock frequency (20 MHz) in Hz:

$$\text{Baud Value} = \frac{\text{Clock}}{(16 \times \text{baud rate})} - 1 = \frac{20 \times 10^6}{(16 \times 57600)} - 1 = 21 (\text{rounded to the nearest integer})$$

5.2.11 Instantiate CoreGPIO

CoreGPIO allows you to access up to 32 general purpose inputs and 32 general purpose outputs. Follow the steps below to instantiate CoreGPIO.

1. From **Catalog**, select Peripherals.
2. Instantiate **CoreGPIO version 1.2.103** with the default settings (32 inputs and 32 outputs).

5.2.12 Instantiate RC Oscillator

Microsemi Fusion devices have a 100 MHz on-chip RC oscillator which does not require any extra components. You can use this as an input to the PLL inside the FPGA. Follow the steps below to instantiate the RC Oscillator.

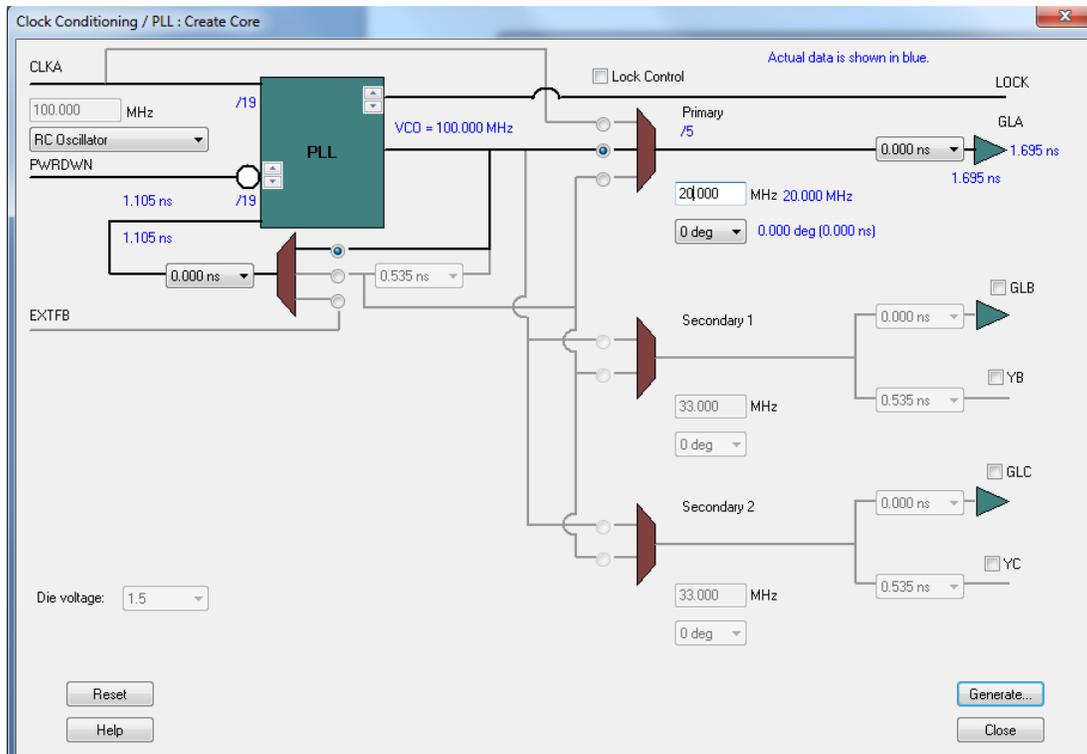
1. From **Catalog**, select **Fusion Peripherals**.
2. Instantiate **Oscillator - RC** with clock conditioning circuit selected and click **Generate**.
3. Name it **myRCOSC**.

5.2.13 Instantiate PLL

The PLL inside the FPGA can be used to generate new clock signals from a source clock. For this design, you can configure the PLL to use the RC Oscillator (from the previous step) as the input clock and generate a 20 MHz output clock (as the system clock). Follow the steps below to instantiate the PLL.

1. From **Catalog**, select **Clock & Management**.
2. Instantiate **PLL - Static** with the following settings:
 - Clock input: RC Oscillator (on the left below the MHz text box)
 - GLA Output: 20 MHz (no delay)
3. Click **Generate**. Name the PLL as **PLL_sys**.

Figure 20 • Configuring the PLL



5.2.14 Instantiate AND2 Gate

Microsemi recommends that you hold the system in reset until the PLL has locked. You can use this AND2 gate to accomplish this. Later you can make the connections to other components. Follow the steps below to instantiate the AND2 gate.

1. From **Catalog**, select **Actel Macros**.
2. Instantiate **AND2**.

5.2.15 Connect Signals Automatically in SmartDesign

SmartDesign has the ability to automatically connect together standard interfaces for components with the Auto Connect feature. Many of the interfaces to standard components are well-defined and permit automatic connection. These include clock and reset signals and Advanced Microcontroller Bus Architecture (AMBA) master/slave connections. Follow the steps to have SmartDesign automatically connect signals and to configure the memory maps.

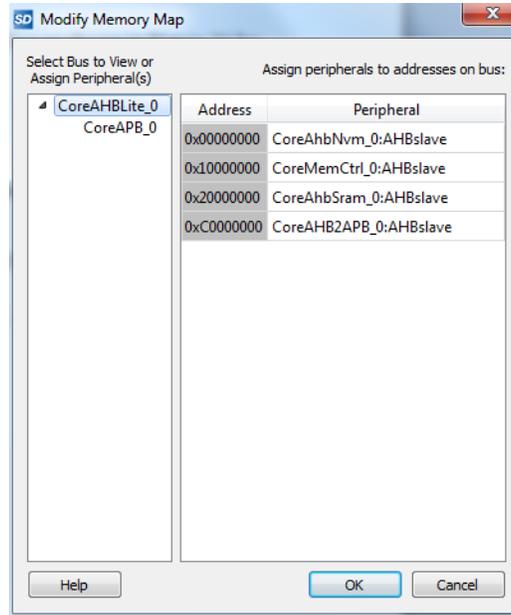
1. Perform **AutoConnect** (right-click canvas and select **Auto Connect**). The Modify Memory Map window is displayed.
2. Close it and go to the canvas. From canvas, click CoreAPB_0 and check all the slots (Slot 0-15). Ensure Slots 0,1,2,12 are enabled in Core AHBLite_0 (under **Enable Master0/1 AHBLite slave slots**).

3. Right-click and select **Modify Memory Map** on SmartDesign canvas, the Modify Memory Map window is displayed as shown in the following figure.
4. Click **CoreAHLite_0**. Configure the memory map for the CoreAHLite bus as shown in the following figure by following the steps below for each peripheral that you need to move.
 - Select the **Peripheral**. To remove the existing peripheral address, select blank from the drop-down list.
 - Select the name of the **Peripheral** from the list.

If you swap two peripherals, you must select blank for both the address locations and then select each peripheral at the corresponding addresses.

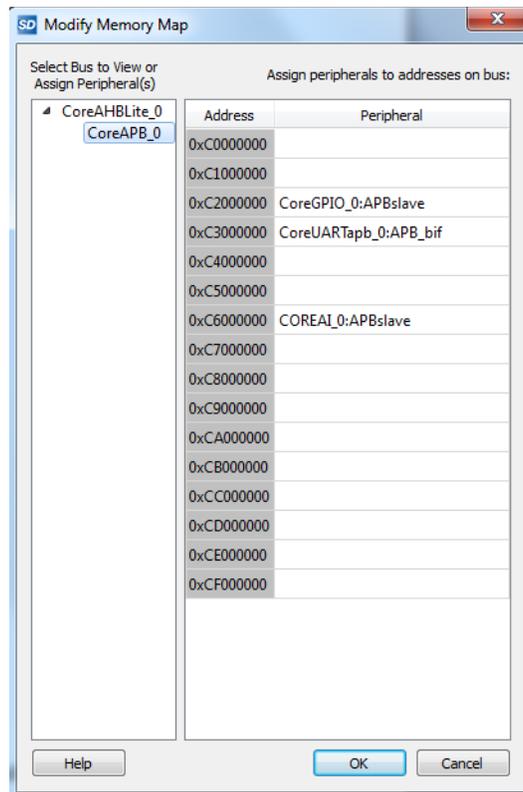
Both SRAM and flash peripherals are added automatically when using CoreMemCtrl.

Figure 21 • Modify Memory Map for CoreAHLite_0



5. Click **CoreAPB_0**. Configure the memory map for the CoreAPB bus as explained above. Click **OK**.

Figure 22 • Configure Memory Map for CoreAPB_0



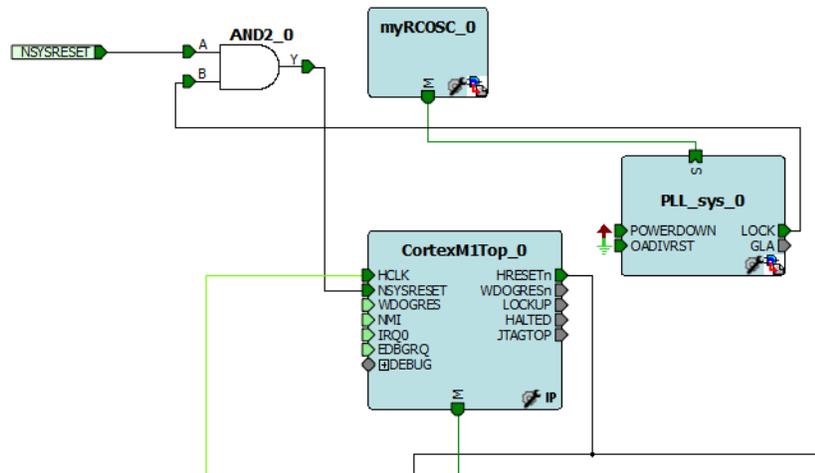
- After closing the Memory Map window recheck all the slots of CoreAPB_0 are enabled and GPIO, UARTapb, COREAI are connected to slot 2, 3, and 6 respectively. Configure the Memory Map as shown in [Figure 21](#), page 21 and [Figure 22](#), page 22. This directly defines the subsystem memory map and has a direct impact on the Cortex-M1 software code. Right-click and select the Auto Arrange Instances feature in SmartDesign to place all the peripherals in an orderly manner on the canvas.

5.2.16 Connect Signals Manually in SmartDesign

Some signals must be manually connected. Use the Canvas in a graphical manner and connect the signals by following the steps below.

- Click **Signal** (For example: LOCK of PLL_sys_0)
- Press the **Ctrl** key and click the second signal to be connected (For example: B of AND2_0)
- Right-click the signal to be connected (Signal B of AND2_0), and select **Connect** as shown in the following figure.

Figure 23 • Connecting Signals Manually



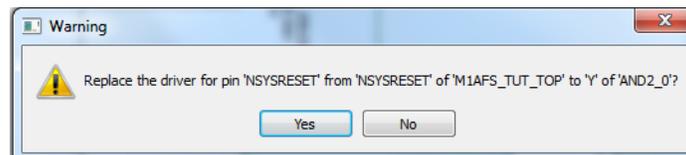
- Follow steps 1–3 for the rest of the signals.

Table 5 • Connecting Signals Manually

FROM		TO	
Instance	Signal	Instance	Signal
PLL_sys_0	LOCK	AND2_0	B
(top level port)	NSYSRESET	AND2_0	A
AND2_0	Y	CortexM1Top_0	NSYSRESET

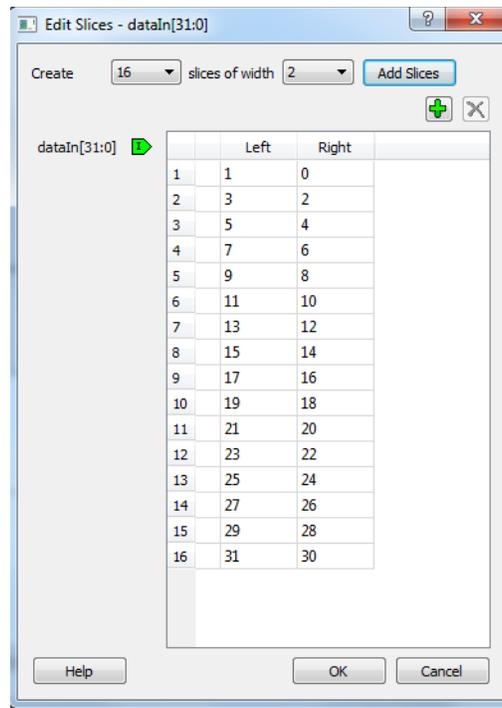
- When making the last connection in the table, a message is displayed as shown in the following figure.

Figure 24 • Replace Driver Warning



- Click **Yes**. You receive this message because you are changing the driver for NSYSRESET of the Cortex-M1 processor.
- Some signals need to be connected to GND or VCC. Follow the steps below to connect signals according to [Table 5](#), page 23 using the SmartDesign Canvas.
- Right-click the dataIn[31:0] signal from the CoreGPIO_0 instance and select **Edit Slice**.
- Select slice width as 2 and create 16 slices as shown in the following figure.
- Click **Add Slices** and then click **OK**.

Figure 25 • Edit Slice



11. Click '+' icon near dataIn[31:0] on the Smart Canvas and press **Ctrl** and select all the signals from dataIn[30:31] to dataIn[3:2]. Right-click and select **Tie Low** option. This is because these bits are not being used in the design.
12. Right-click **EDBGRQ** signal in the CortexM1Top_0 instance, and select **Tie Low**. Follow steps 10–11 for the remaining signals in the following table. However, remember to choose **Tie High** for the POWERDOWN signal of the PLL_sys_0 instance.

Table 6 • VCC and GND Connections

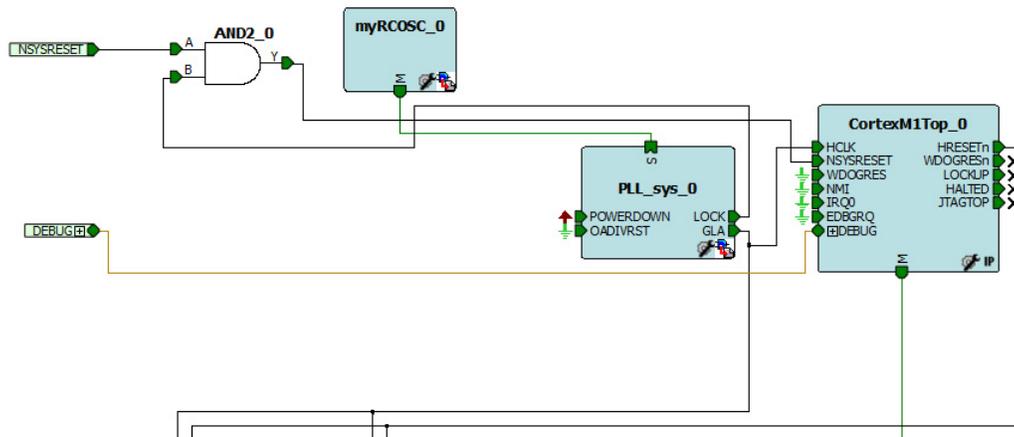
Instance	Signal	Connection
CoreGPIO_0	dataIn[31:2]	GND
CortexM1Top_0	EDBGRQ	GND
	NMI	GND
	IRQ0	GND
	WDOGRES	GND
PLL_sys_0	POWERDOWN	VCC
	OADIVRST	GND
CoreAHLite_0	REMAP_M0	GND

Follow the steps below to change the clock source from the auto-generated SYSCLK top-level port to the output of the PLL:

1. Right-click **SYSCLK** top level port and select **Disconnect**.
2. Highlight **HCLK** input of the Cortex-M1.
3. Press **Ctrl** and left-click to highlight the GLA output of the PLL.
4. Right-click **GLA** and select **Connect**.
5. Delete **SYSCLK** port as it is not required in this design.

Now, the GLA output of the PLL should be the clock driver for the clock port of all peripherals. Your Canvas should be as shown in the following figure.

Figure 26 • SmartDesign Canvas



5.2.17 Promote Signals to Top Level

The SmartDesign component is the top level of this design. Therefore, all of the signals connected to a pin of the FPGA must be at the top level of the SmartDesign canvas. Follow the steps below to promote the signals in the following table to the top level.

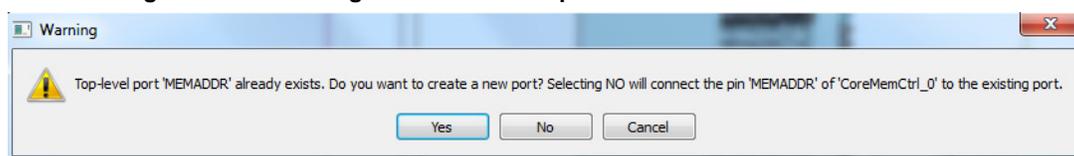
1. From **Canvas**, right-click **DEBUG** group of the **CortexM1Top_0** instance and select **Promote to Top Level**.
2. Click '+' icon near dataIn[31:0] bus of CoreGPIO_0 instance and right-click dataIn[1:0]. Then select **Promote to Top Level**.
3. Right-click dataOut[31:0] and click **Edit Slice**. Put the width of slice as 4 and create 8 slices by clicking **Add slice**.
4. Click '+' icon at dataOut [31:0] and promote dataOut[3:0] to top level.
5. Right-click and select **Promote to Top Level** to promote the rest of the signals in the following table.

Table 7 • Signals to Promote to Top Level

Instance	Group	Signal
CortexM1Top_0	DEBUG	All signals
CoreGPIO_0	dataIn[31:0]	dataIn[1:0]
	dataOut[31:0]	dataOut[3:0]
CoreMemCtrl_0	ExternalMemoryInterface	MEMADDR[17:0] (requires slice with slice width 2), MEMDATA[31:0], SRAMBYTEN[3:0], SRAMCSN, SRAMOEN, SRAMWEN
CoreUARTapb_0		RX
		TX

6. Promote only one pin at a time in an order starting with **MEMADDR[1:0]** till MEMADDR[17:16]. While promoting **MEMADDR [17:0]**, you might get a message as shown in the following figure.

Figure 27 • Message while Promoting MEMADDR to Top Level



5.2.18 Save and Generate SmartDesign System

The final step within SmartDesign is saving and generating the HDL. Follow the steps below to generate the HDL for your design. Besides generating the HDL design, a test bench file is created along with all the necessary files to run a simulation, using the bus functional model (BFM) and AMBA transfers.

1. Before generating the design go to **Design > Configure Firmware**. Disable the firmware which is not required by clearing the checkboxes against them. The version of the firmware should be as shown in the following figure. Ensure you use Version 2.1.102 for the GPIO driver.
2. Also, ensure if all the drivers are available and no warning or error appears saying that the firmware is missing from the Vault. If so, refer to [Appendix: Firmware Catalog Settings](#), page 49.

Figure 30 • Design Firmware

Generate	Instance Name	Core Type	Version	Compatible Hardware Instance
<input checked="" type="checkbox"/>	CoreAI_Driver_0	CoreAI_Driver	3.0.101	M1AFS_TUT_TOP:COREAI_0
<input checked="" type="checkbox"/>	CoreAHbNvm_Driver_0	CoreAHbNvm_Driver	2.1.102	M1AFS_TUT_TOP:CoreAHbNvm_0
<input checked="" type="checkbox"/>	CoreGPIO_Driver_0	CoreGPIO_Driver	2.1.102	M1AFS_TUT_TOP:CoreGPIO_0
<input checked="" type="checkbox"/>	CoreIARTapb_Driver_0	CoreIARTapb_Driver	3.0.105	M1AFS_TUT_TOP:CoreIARTapb_0
<input checked="" type="checkbox"/>	HAL_0	HAL	2.1.102	M1AFS_TUT_TOP:CortexM1Tap_0

3. Click **Generate** as shown in [Figure 31](#), page 27 for generating the project.
4. See the log window for the information if the design has been generated or not.
5. Ensure the DRC report has only one warning about not connecting CoreAHLite_0 M1 pin. Also, make sure there are no other warnings. You can ignore this warning.
6. After generation, a SoftConsole folder is created in the **Files** tab along with few subfolders as shown in [Figure 32](#), page 28.

Figure 31 • Generating the Design

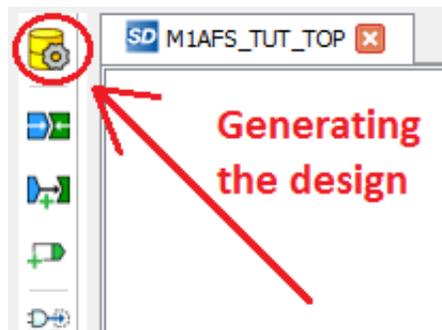
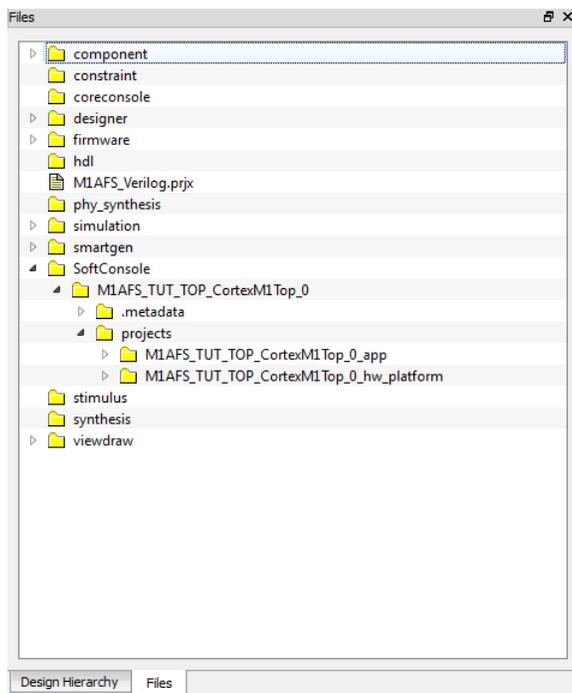


Figure 32 • SoftConsole Folder

5.3 Step 3 – Create Flash Memory System

You are creating this Flash Memory System and Data Storage client that allow you to include software code, as an Intel-Hex file, in the FPGA programming file. This allows the Cortex-M1 processor to boot and run from the NVM.

1. From **Catalog** expand **Fusion Peripherals** category.
2. Double-click **Flash Memory System Builder**. The Flash Memory System window is displayed as shown in [Figure 33](#), page 29.
3. Double-click **Data Storage Client** in the available client types list. The Add Data Storage Client window is displayed as shown in [Figure 34](#), page 29.
4. Enter NVM as the **Client name**.
5. Click **Browse** and select the **CortexM1_Tutorial.hex** file from the CortexM1_Proc_Tutorial\<board>\Tutorial_Files directory.
6. The memory content file should be in Intel-Hex format.

Figure 33 • Flash Memory System Window

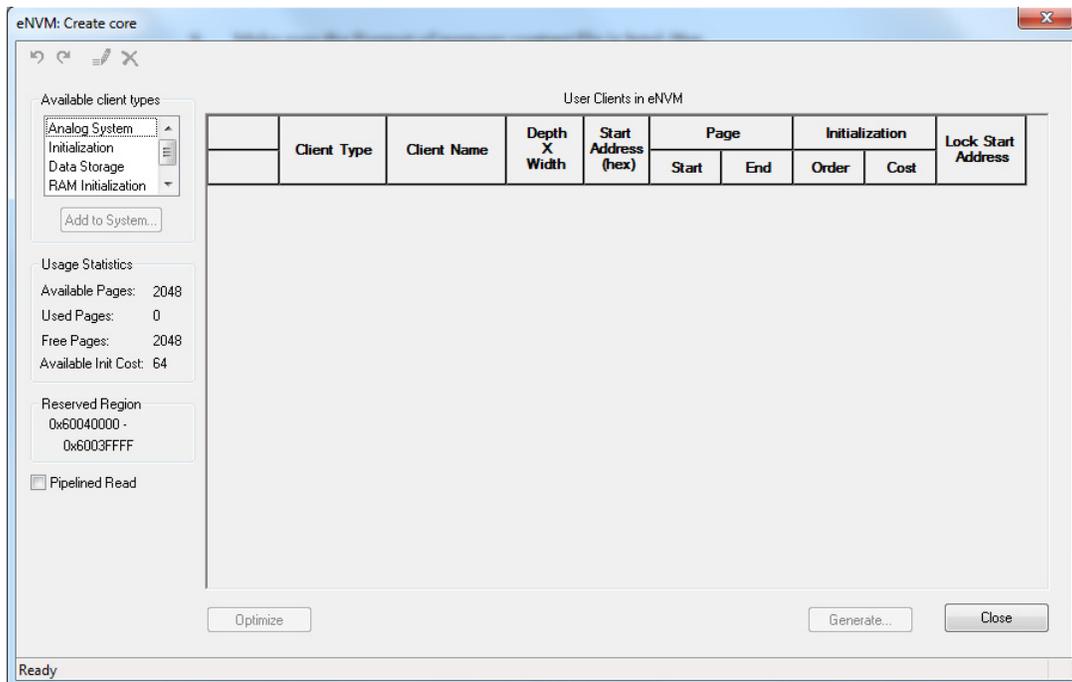
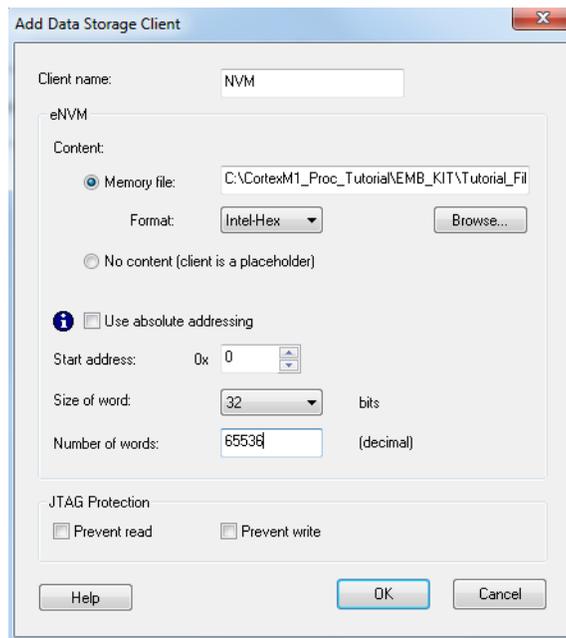
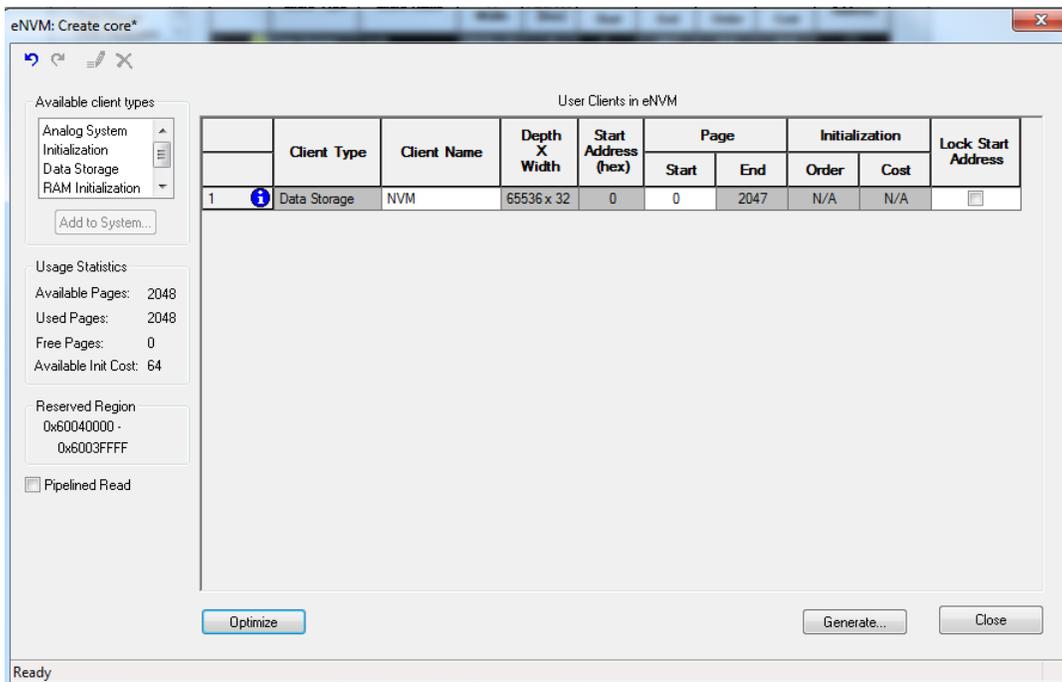


Figure 34 • Add Data Storage Client



7. Click **OK**. Now the created Data Storage client is displayed in the Flash Memory System window as shown in the following figure.

Figure 35 • Data Storage Client in Flash Memory System Window



8. Click **Generate** to generate the component. The Generate Core window is displayed.
9. Enter NVM_contents as the **Core name**.
10. Click **OK**.

Note: Ensure that no component is instantiated on SmartDesign canvas called NVM contents. If it does then it has to be deleted and the project has to be regenerated.

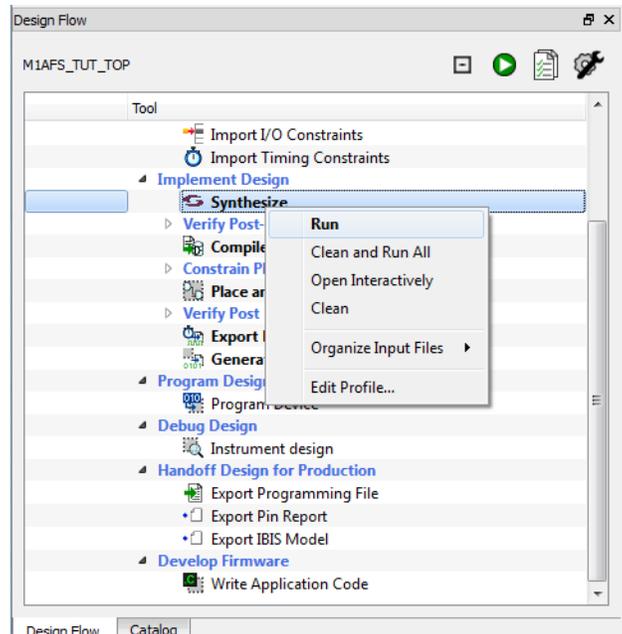
6 Simulation, Synthesis, and Place-and-Route

6.1 Step 4 – Perform Synthesis

After generating the project, synthesis can be performed. Follow the steps below to synthesize the design.

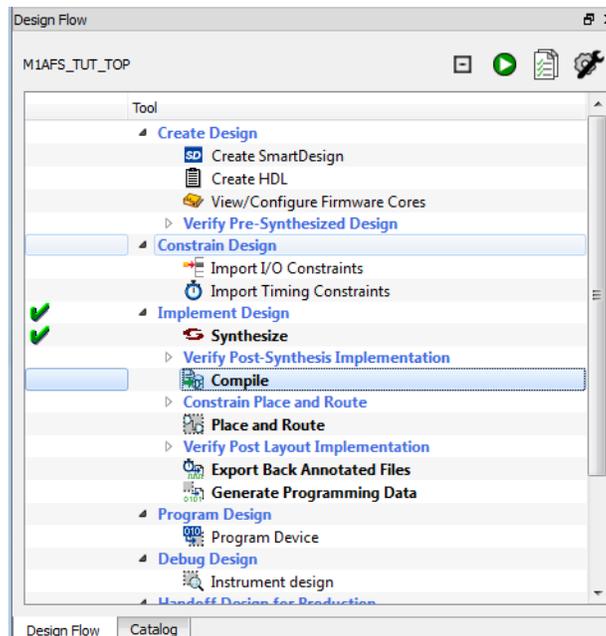
1. Right-click **Synthesize** under **Implement Design** and select **Run** to synthesize the design as shown in the following figure.

Figure 36 • Synthesizing the Project



2. Once the synthesis is complete warning messages are displayed which can be ignored. However, if any errors are present, they need to be corrected prior to continuing. Once synthesis is completed, a green tick mark displayed near the Synthesize icon as shown in the following figure.

Figure 37 • After Synthesis

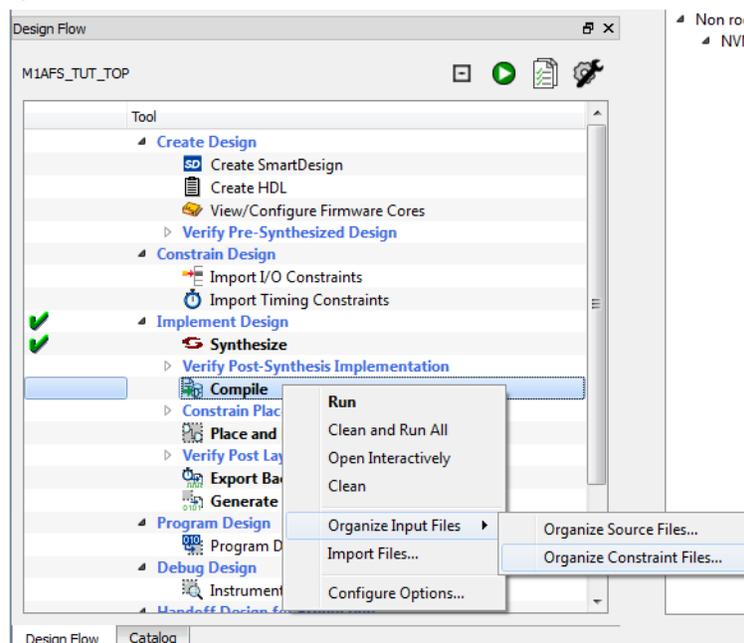


6.2 Step 5 – Perform Place-and-Route

Now that you have synthesized the design, you can place-and-route the design. Follow the steps below to place-and-route the design.

1. Right-click **Compile**, select **Organize Input Files**, and then select **Organize Constraint** as shown in the following figure.

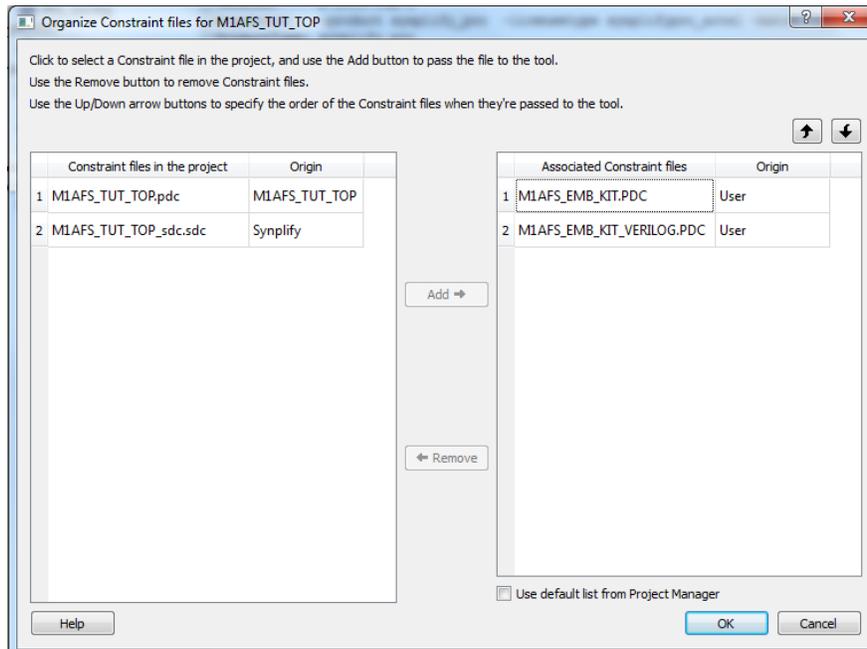
Figure 38 • Organizing Constraint Files



2. The first two files created automatically by SmartDesign and Synplify are not required. Select the files and click **Remove**. Only the two imported PDC files are required. The constraints organizer should look as shown in the following figure.

- Click **OK**. Also unselect the **Use default list** from **Project Manager** check-box to make the changes.

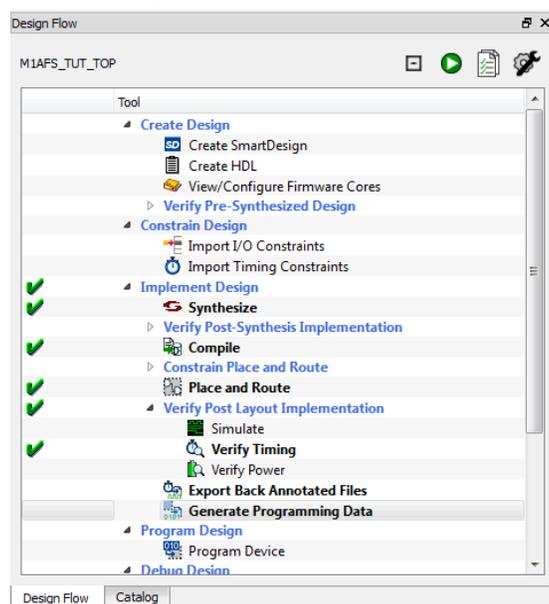
Figure 39 • Constraints Organizer Window



- Right-click **Compile** under **Implement Design**, and then click **RUN**. Once the compilation is completed, a green tick mark is displayed at the Compile.
- Right-click **Place-and-Route** and click **RUN**. Once the place-and-route is completed, a green tick mark is displayed against the **Place and Route** icon.
- Right-click **Verify Timing** under **Verify Post Layout Implementation**, and then click **RUN**. After this the design flow should be as shown in the following figure.

Note: The timing requirement for the system clock (myPLL_0/Core:GLA) is automatically created because you are using a PLL. If any of the clocks in the design are not meeting the constraints set in the constraint files, a red X is displayed next to the verify timing. To set this right-click **Verify Timing** and select **Open Interactively**. Ensure that red X is not displayed for any of the clocks.

Figure 40 • Design Flow after Verify Timing



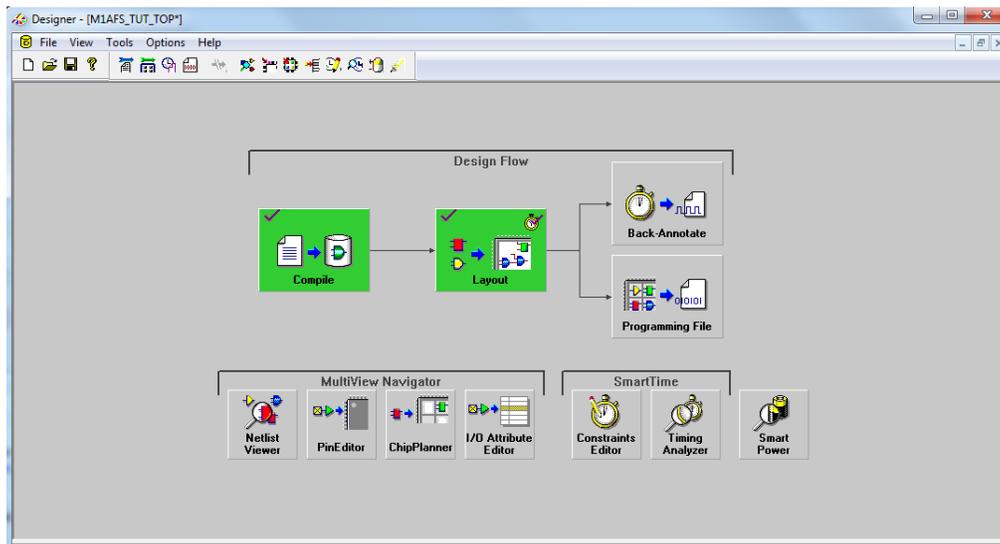
7 Programming

7.1 Step 6 – Generate Programming File with Software Code in NVM

In this section, you will associate the Data Storage client you created with the embedded flash memory inside the Fusion device. This places the software code and an Intel-Hex file into the embedded flash memory in the Fusion device and includes it in the FPGA programming file. Use the steps below.

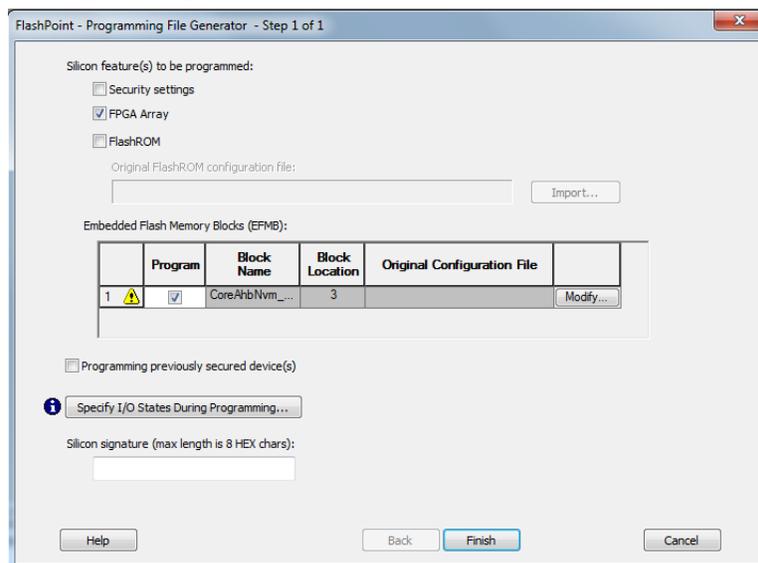
1. Right-click **Generate Programming Data** and select **Open Interactively**. A Designer window is displayed as shown in the following figure.

Figure 41 • Designer Window



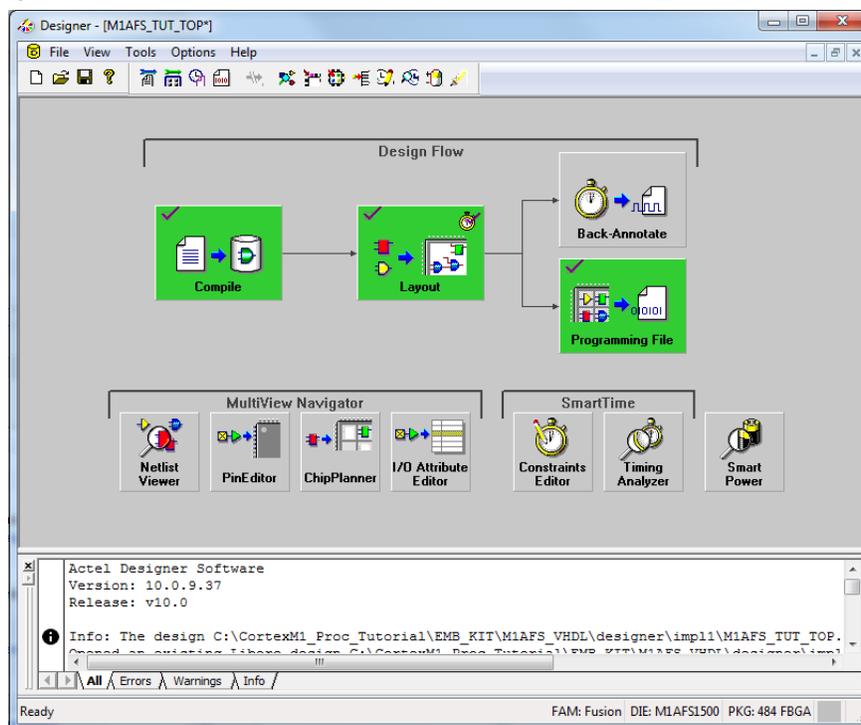
2. Click **Programming File** and **Flash Point**; the **Programming File Generator** opens as shown in the following figure.

Figure 42 • FlashPoint Programming File Generator



3. Select **CoreAhbNvm** check-box under **Embedded Flash Memory Blocks (EFMB)**. Click **Modify**, the Modify Embedded Flash Memory Blocks window is displayed.
4. Click **Import Configuration File**. The Import window appears.
5. Browse to <project directory>/smartgen/NVM_contents/NVM_contents.efc.
6. Click **Import**, and then click **OK**.
7. Click **Finish**. The Generate Programming Files window is displayed.
8. Ensure that Programming Data File (*.pdb) is selected.
9. Click **Generate**.
Once the Programming File icon in the Designer changes to green, the programming file for the Fusion device is generated and you are ready to program the device with your Cortex-M1 design. A completed Designer desktop is shown in the following figure.

Figure 43 • Designer Desktop

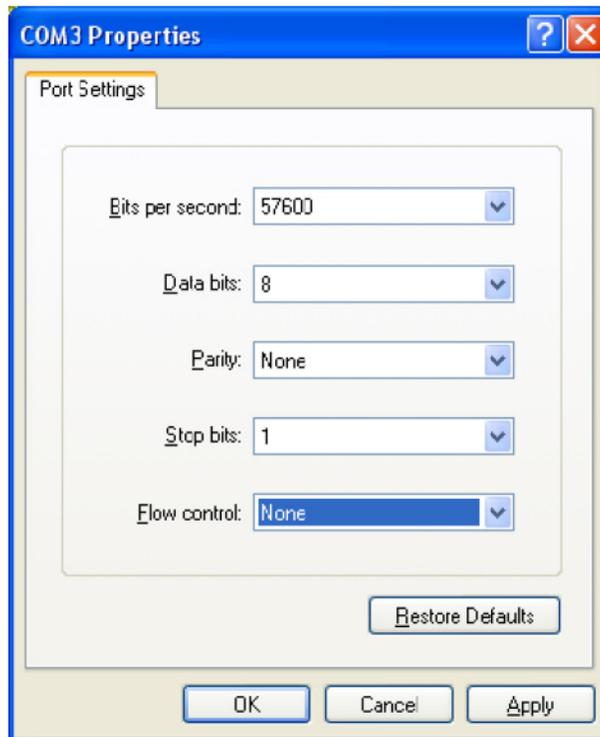


10. Close Designer window. Click **Yes** if prompted to save changes to M1AFS_TUT_TOP.adb. On generating the file, a green tick is displayed against the **Generate Programming file**.

7.2 Step 7 – Connect to the Target

Before programming the FPGA, you need to connect to the target board and setup HyperTerminal to communicate over the UART in the design. Perform the following steps to setup the communication.

1. Open the **HyperTerminal** application (**Start > Programs > Accessories > Communications > HyperTerminal**).
2. Enter **Name as M1AFS_Tutorial** in the Connection Description dialog box and click **OK**.
3. Select COM port you identified in the [Getting Started](#), page 5 section of this tutorial and click **OK**.
4. Enter the following properties as shown in the following figure:
 - Bits per second: 57600
 - Data bits: 8
 - Parity: None
 - Stop bits: 1
 - Flow control: None
5. If your PC does not have HyperTerminal, use any free serial terminal emulation program as PuTTY or Tera Term. Refer to the [Configuring Serial Terminal Emulation Programs](#) tutorial for configuring the HyperTerminal, Tera Term, and PuTTY.

Figure 44 • COM3 Properties

6. Click **OK**. HyperTerminal is now connected with the appropriate settings to communicate with the UART on the target design.

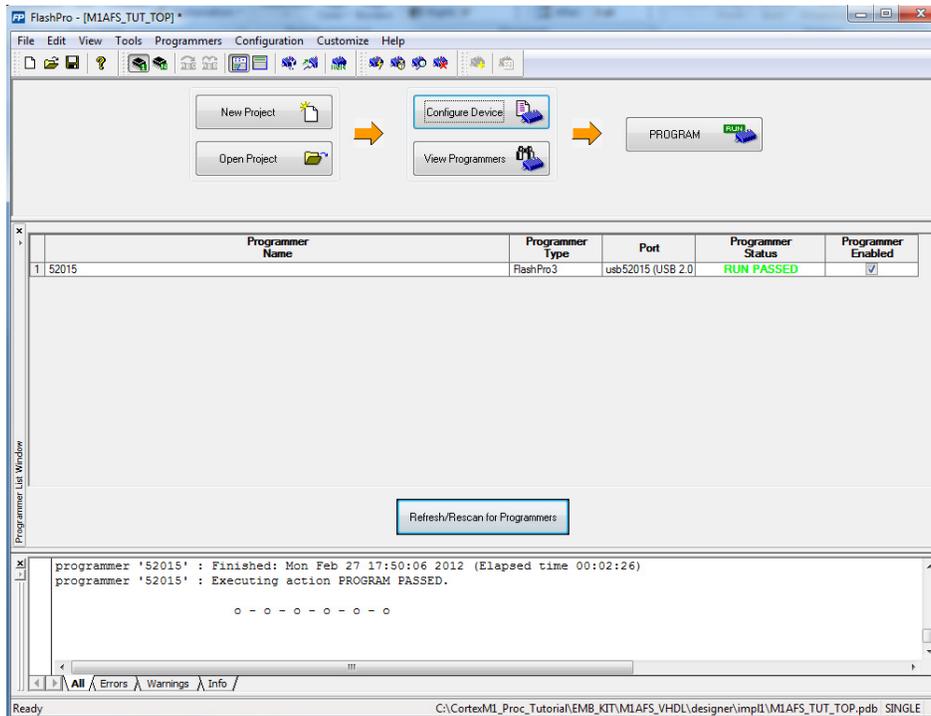
7.3 Step 8 – Program the M1AFS1500 FPGA

Now the Cortex-M1 processor system is created and the corresponding FPGA programming file is generated, you can program the FPGA with your design. Follow the steps below to download the programming file to the FPGA.

1. Confirm that the FlashPro programmer is connected as described in [Table 4](#), page 7.

Note: Do not interrupt power during the programming because the device might get damaged.

2. Right-click **Program Device** (FlashPro) and select **Open** interactively in the Design Flow window. A message is displayed if you want to proceed without IO constraints. Click **YES**. The FlashPro window is displayed as shown in the following figure.

Figure 45 • FlashPro Project Flow

Note: If you receive any messages about new hardware being detected on your PC, complete the USB driver installation as described in the “Getting Started” section of the FlashPro User’s Guide (www.microsemi.com/soc/products/hardware/program_debug/flashpro/default.aspx#docs).

3. Click **PROGRAM** to program the Fusion device. Once the programming is completed, the Programmer Status is shown as **RUN PASSED** in green color as shown in the preceding figure.
4. Press **SW1** on the board to reset the system (Figure 46, page 37 and Figure 47, page 38).

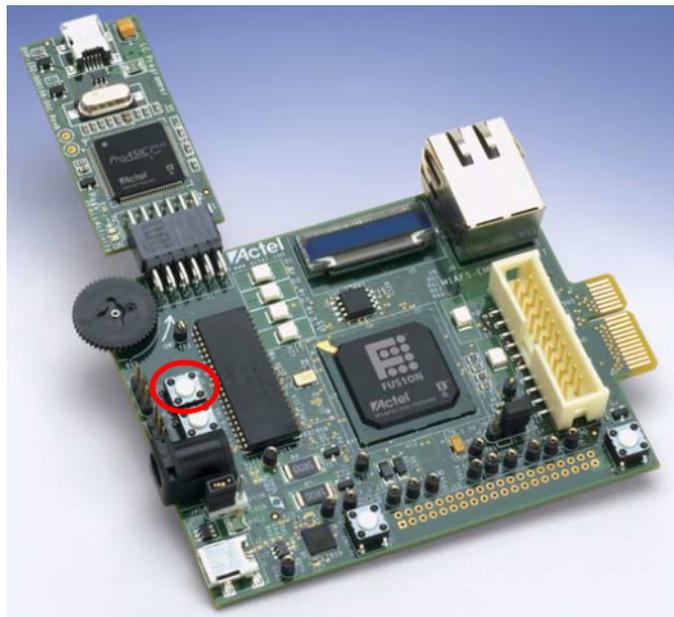
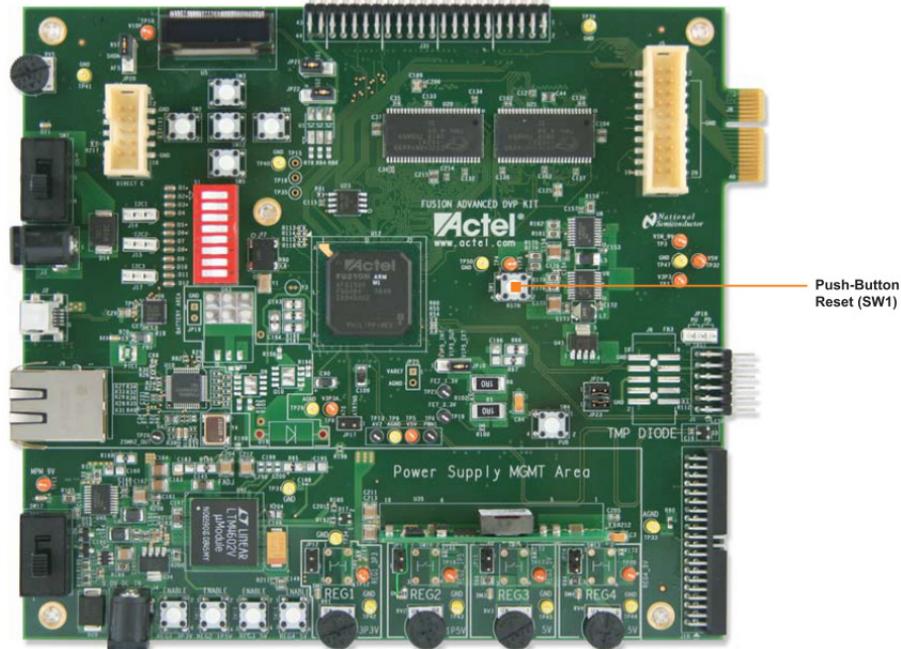
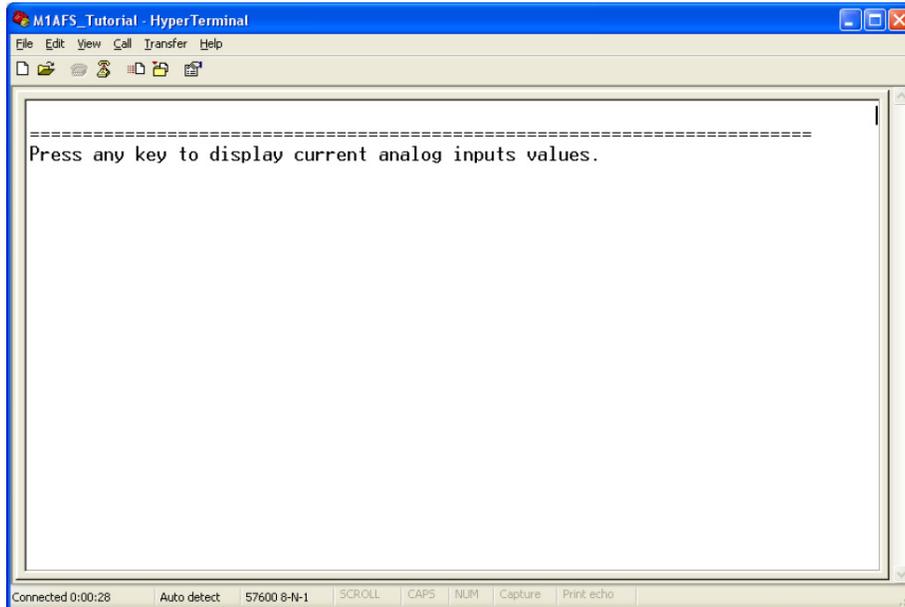
Figure 46 • Fusion Embedded Development Kit Reset System with SW1

Figure 47 • Fusion Advanced Development Kit Board Reset System with SW1

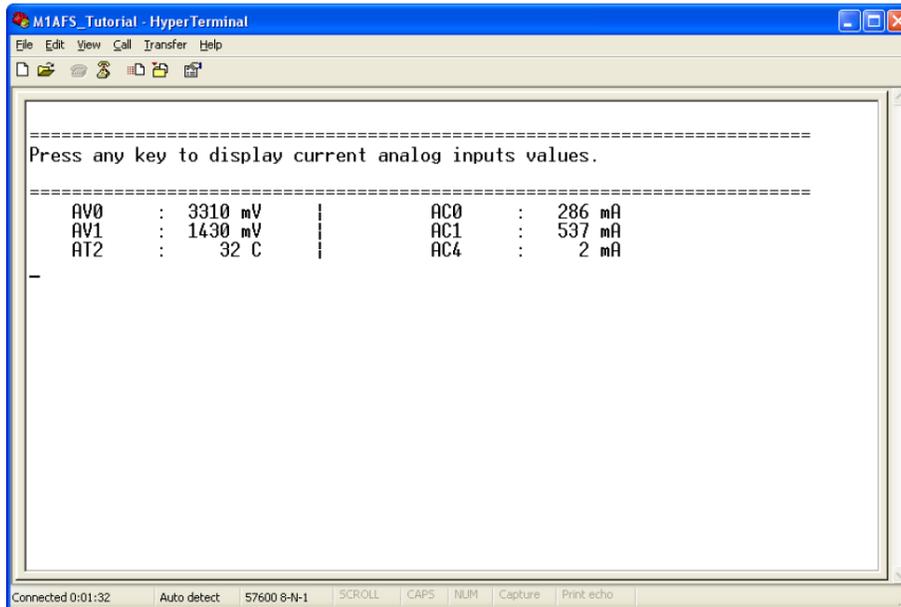


5. After pressing the reset button (SW1, you should see some of the LEDs counting. The message is displayed in the HyperTerminal window as shown in the following figure.

Figure 48 • HyperTerminal Window



6. Press any key. The window is displayed as shown in the following figure.

Figure 49 • HyperTerminal Display

```
=====  
Press any key to display current analog inputs values.  
=====  
AV0 : 3310 mV | AC0 : 286 mA  
AV1 : 1430 mV | AC1 : 537 mA  
AT2 : 32 C | AC4 : 2 mA  
-  
=====  
Connected 0:01:32 Auto detect 57600 8-N-1 SCROLL CAPS NUM Capture Print echo
```

7. Change the potentiometer on the development board which is on the left corner on M1AFS-EMBEDDED-KIT and near power supply for M1AFS-ADV-DEV-KIT.
8. Press any key. The AC4 value should get changed.
9. The design is measuring the voltage across the potentiometer. You should be able to measure a range of about 0 V to 3.3 V across the potentiometer.

Congratulations! You have successfully created a Cortex-M1 system. Now you can develop the software for your Cortex- M1 design. If you want to learn how to get started developing the software for Cortex-M1, refer to [Debugging the Application Using SoftConsole](#), page 40.

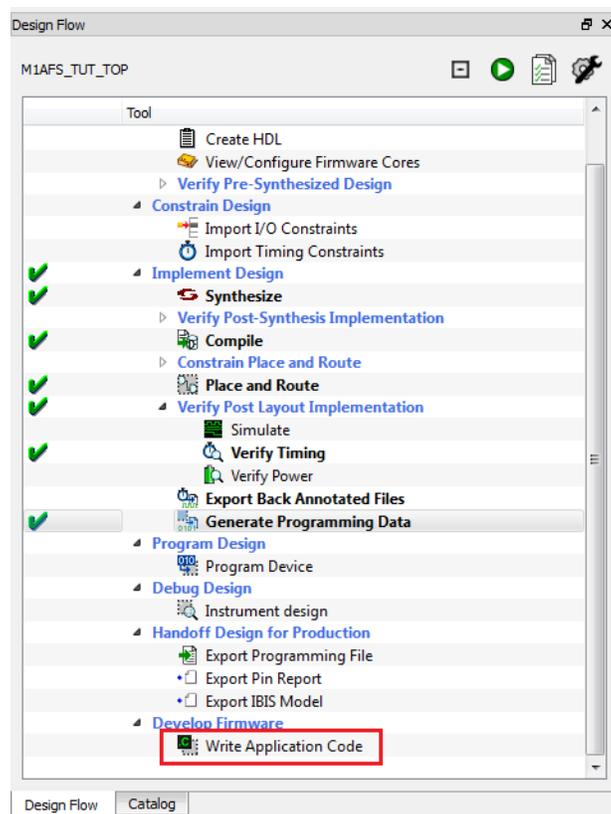
8 Debugging the Application Using SoftConsole

8.1 Step 9 - Building the Software Application through SoftConsole

Before Launching SoftConsole ensure you have programmed the target device using FlashPro and the device is connected to the PC. You can also program the device with the programming file (*.pdb) given in Software_tutorial_files by following [Step 8 – Program the M1AFS1500 FPGA](#), page 36.

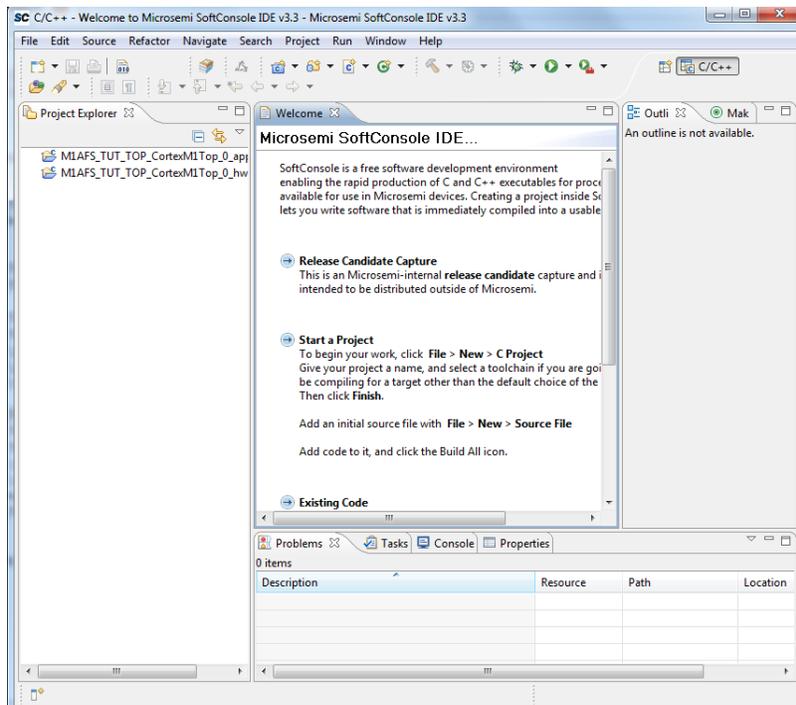
1. From Libero SoC open the SoftConsole project by double-clicking **Write Application Code** under **Develop Firmware in Design Flow** window.

Figure 50 • Invoking SoftConsole from Libero SoC



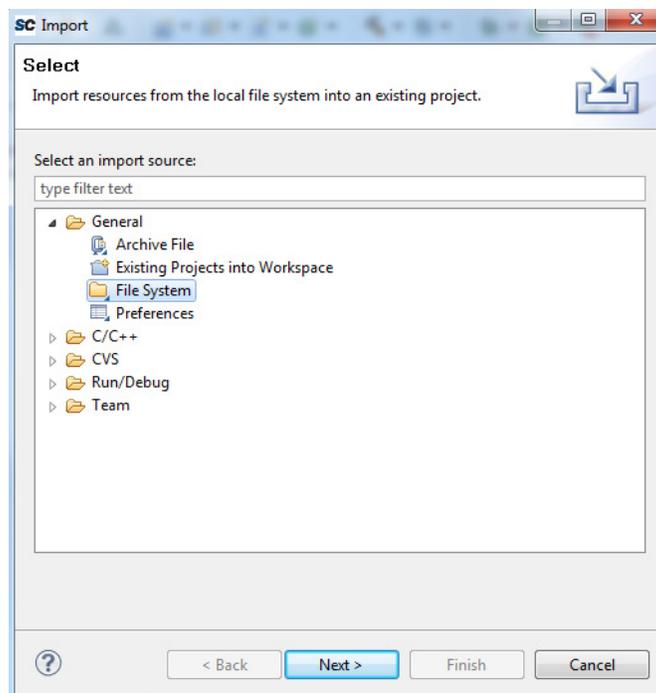
2. The SoftConsole window is displayed as shown in the following figure.

Figure 51 • SoftConsole Workspace



3. Right-click **M1AFS_TUT_TOP_Cortex M1_Top_0_app** and click **Import**. The Import window is displayed.
4. Select **File System** under General and click **Next** as shown in the following figure.

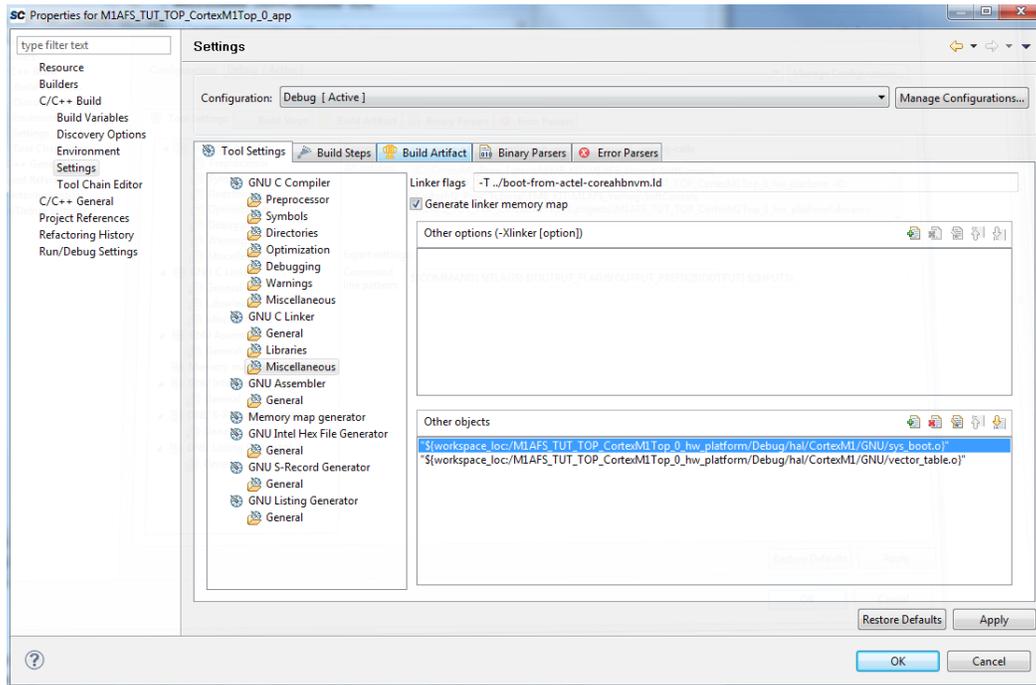
Figure 52 • Import Window



5. Click **Browse**.
6. Browse and select the tutorial source files: <zip file location>/CortexM1_Proc_Tutorial/<KIT>/Software_tutorial_files.
7. Click **OK** and select main.c, tutorial.h, and boot-from-actel-coreahbnvm.ld.

8. Ensure that the **Create selected folders only** check-box is selected.
9. Click **Finish**. Click **Yes** if it asks to overwrite main.c. If you expand the project in the Navigator, you can see the new files that have been imported into the project.
10. Right-click **M1AFS_TUT_TOP_Cortex M1_Top_0_app** and select **Properties**. The Properties for M1AFS_TUT_TOP_Cortex M1_Top_0_app window are displayed as shown in the following figure.
11. Expand **C/C++ Build** on the left pane and select **Settings**.
12. Click **GNU C Linker** under **Miscellaneous**.
13. Enter Linker flags as **T ../boot-from-actel-coreahbnvm.ld**.

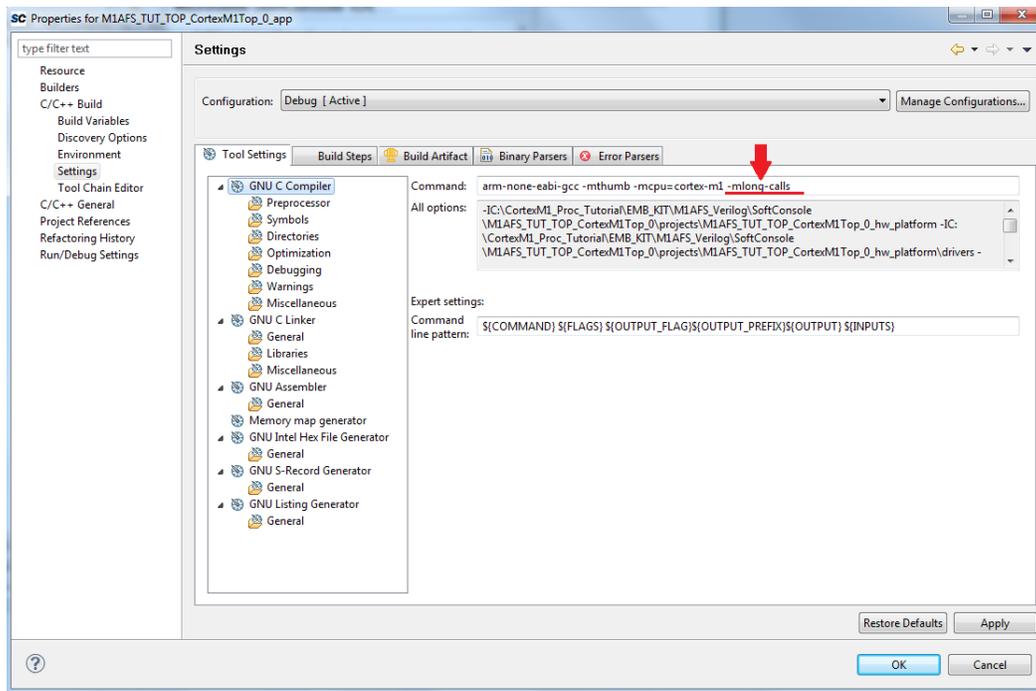
Figure 53 • Linker Flags



Example linker scripts are provided in the SoftConsole installation in the <SoftConsole install>/src/Cortex- M1/linker-script-examples directory. This linker script was copied from this directory and modified for the target system. The system boots from on-chip NVM, copies the application code and data to external SRAM, and then runs the application code from external SRAM.

14. Click **GNU C Compiler**.
15. Add **-mlong-calls** to the command in Command, as shown in the following figure.

Figure 54 • GNU C Compiler



This flag is required only when booting from NVM and running from SRAM (for example, using boot-from-actel-coreahbnvm.ld). It is required to support the call to main() from _start since _start is in NVM, whose base address is 0x00000000, and main() is being executed from SRAM, whose base address is 0x18000000. For more information, refer to the GNU GCC documentation (**Start > Programs > MicrosemiSoftConsole > Reference Documentation > GNU GCC Compiler Manual or <SoftConsole install>\Sourcery-G++\share\doclarm-arm- none-eabi \pdf\gcc\gcc.pdf**).

16. Click **OK** to save the project settings.
17. Open the coreai_cfg.h by clicking **M1AFS_TUT_TOP_Cortex M1_Top_0_hw_platform > drivers > CoreAI > coreai_cfg.h**
18. Make the following changes as shown in the following figure.
 - #define AC0_CONFIG AC_DISABLED change to #define AC0_CONFIG AC_CURRENT_MONITOR
 - #define AC1_CONFIG AC_DISABLED change to #define AC1_CONFIG AC_CURRENT_MONITOR
 - #define AC4_CONFIG AC_DISABLED change to #define AC4_CONFIG AC_4V
 - #define AT2_CONFIG AT_DISABLED change to #define AT2_CONFIG AT_TEMPERATURE_MONITOR
 - #define AV0_CONFIG AV_DISABLED change to #define AV0_CONFIG AV_4V
 - #define AV1_CONFIG AV_DISABLED change to #define AV1_CONFIG AV_2V

Figure 55 • Changes Made to Coreai_cfg.h

```

18#define AC0_CONFIG AC_CURRENT_MONITOR
19#define AC1_CONFIG AC_CURRENT_MONITOR
20#define AC2_CONFIG AC_DISABLED
21#define AC3_CONFIG AC_DISABLED
22#define AC4_CONFIG AC_4V
23#define AC5_CONFIG AC_DISABLED
24#define AC6_CONFIG AC_DISABLED
25#define AC7_CONFIG AC_DISABLED
26#define AC8_CONFIG AC_DISABLED
27#define AC9_CONFIG AC_DISABLED
28#define AG0_CONFIG AG_DISABLED
29#define AG1_CONFIG AG_DISABLED
30#define AG2_CONFIG AG_DISABLED
31#define AG3_CONFIG AG_DISABLED
32#define AG4_CONFIG AG_DISABLED
33#define AG5_CONFIG AG_DISABLED
34#define AG6_CONFIG AG_DISABLED
35#define AG7_CONFIG AG_DISABLED
36#define AG8_CONFIG AG_DISABLED
37#define AG9_CONFIG AG_DISABLED
38#define AT0_CONFIG AT_DISABLED
39#define AT1_CONFIG AT_DISABLED
40#define AT2_CONFIG AT_TEMPERATURE_MONITOR
41#define AT3_CONFIG AT_DISABLED
42#define AT4_CONFIG AT_DISABLED
43#define AT5_CONFIG AT_DISABLED
44#define AT6_CONFIG AT_DISABLED
45#define AT7_CONFIG AT_DISABLED
46#define AT8_CONFIG AT_DISABLED
47#define AT9_CONFIG AT_DISABLED
48#define AV0_CONFIG AV_4V
49#define AV1_CONFIG AV_2V
50#define AV2_CONFIG AV_DISABLED
51#define AV3_CONFIG AV_DISABLED
52#define AV4_CONFIG AV_DISABLED

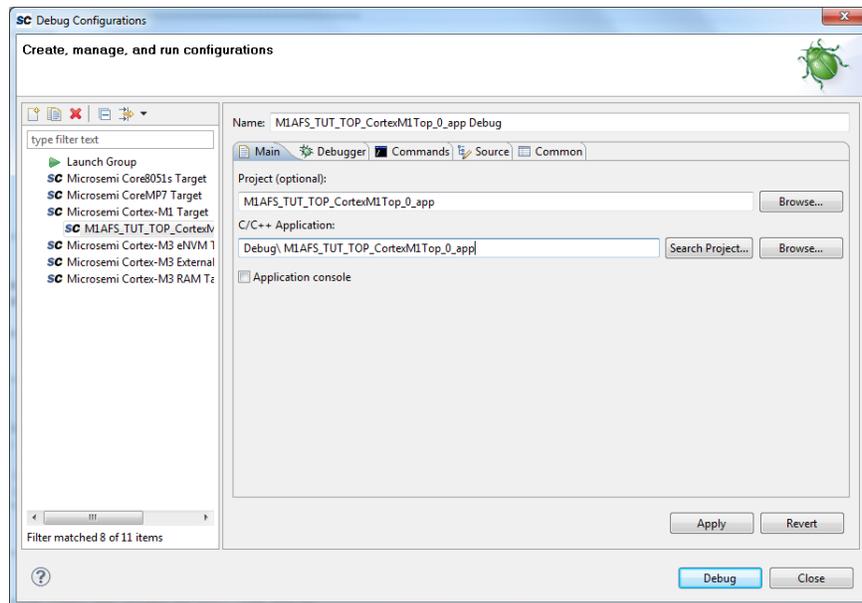
```

19. Save the changes. Perform a clean build by selecting Clean all Projects from **Project > Clean**. Leave others as default in the **Clean** dialog box and click **OK**. Ensure there are no errors and warnings.

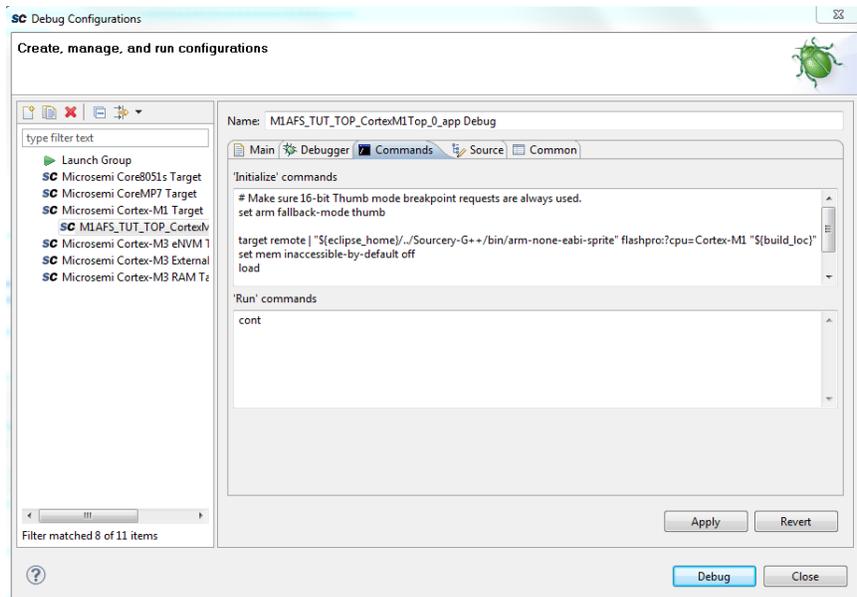
8.2 Step 10 - Debugging the Project

Use the following steps to debug the application project using SoftConsole:

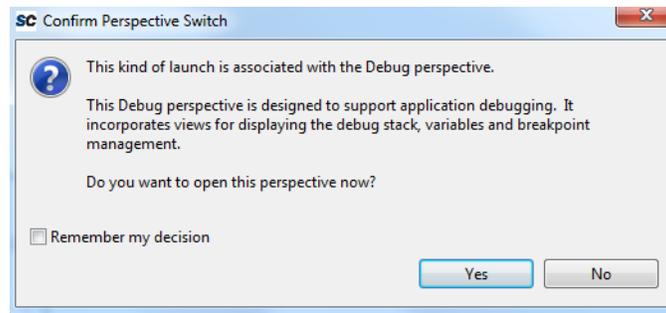
1. From the **Run** menu of the SoftConsole, select **Open Debug Dialog/Debug Configurations**. The Debug dialog is displayed. The M1AFS_TUT_TOP_CortexM1Top_0_app project should be highlighted in the Project explorer window while doing this.
2. Double-click **Microsemi Cortex-M1 Target**. The Debug Configurations window is displayed as shown in the following figure.

Figure 56 • Debug Configurations

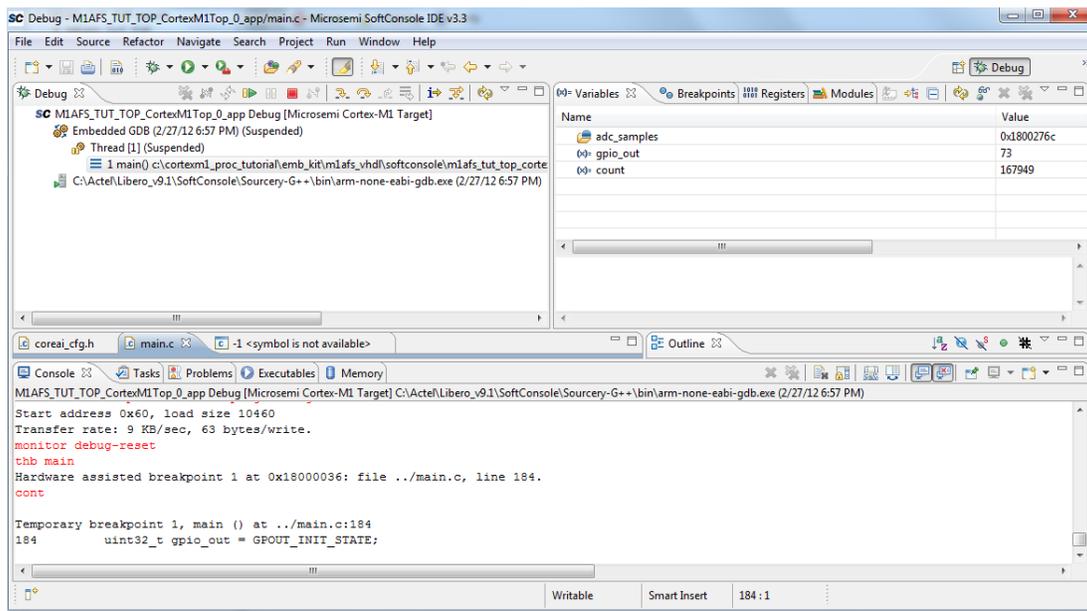
3. Confirm that the following details appear on the **Main** tab in the Debug window:
 - Name: M1AFS_TUT_TOP_CortexM1Top_0_app Debug
 - Project: M1AFS_TUT_TOP_CortexM1Top_0_app
 - C/C++ application: Debug\M1AFS_TUT_TOP_CortexM1Top_0_app
4. Click the **Commands** tab. Confirm that Initialize and Run commands should have the commands as shown in the following figure.

Figure 57 • Debugger Commands

5. Click **Apply** and **Debug**.
6. Click **Yes** when prompted for **Confirm Perspective Switch**. This displays the debug view mode.

Figure 58 • Confirm Perspective Switch

7. The Debug window should be displayed as shown in the following figure.

Figure 59 • Debug Perspective

8. Ensure that HyperTerminal is configured as shown in [Step 7 – Connect to the Target](#), page 35. The Debug View in the upper left shows the debug communication client running as well as the call stacks in the application.

Note: The processor is currently suspended and the **Resume**  and **Step** buttons  are enabled.

A breakpoint is always set at the first executable line of code in main().

You see this line of code highlighted in main.c.

The various processor views, including the current variables, breakpoints, and processor registers and modules, can be seen on the top-right corner. You can see these values change as you go through the code.

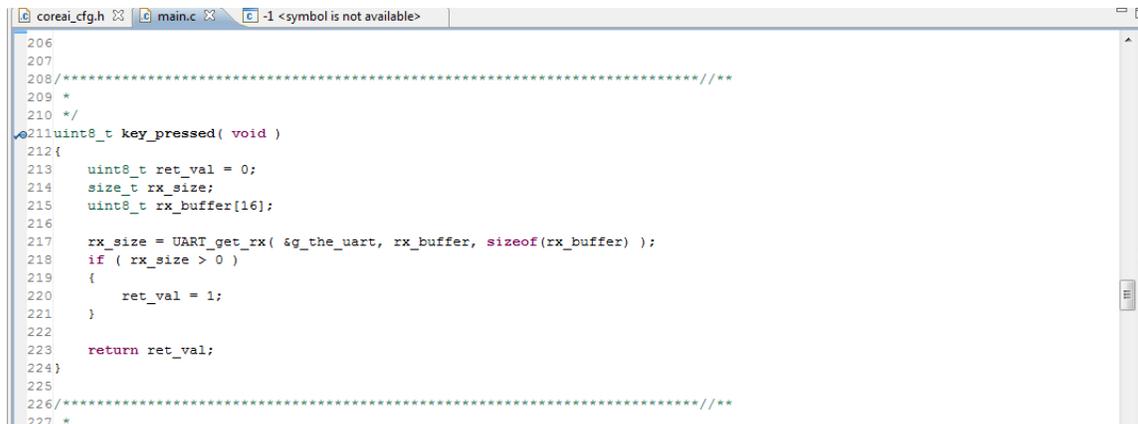
Some views may not be displayed by default. From **Window** click **Show View** to add them. If you add the Memory view, you can see a **Memory** tab at the bottom. Here you can view the values stored in memory. You can view the values and modify the memory in the target. You can add a memory monitor by clicking the green plus icon. Enter **Address** and then click **OK**. You can add multiple memory monitors.

The application code initializes the UART peripheral, and the analog quads and ADC of the analog block of the Fusion FPGA. Then a message is sent to the UART, prompting the user to press any key.

The Cortex-M1 processor reads and displays the following information from the analog sources on the board (analog channel in parentheses):

- 3.3 V supply voltage (AV0)
 - 3.3 V supply current (AV0 and AC0)
 - 1.5 V supply voltage (AV1)
 - 1.5 V supply current (AV1 and AC1)
 - Ambient temperature (AT2)
 - Voltage of potentiometer RV1 (AC4)
9. To run the application, from **Run** select **Resume** on the SoftConsole toolbar.
 10. The output is displayed on HyperTerminal as shown in [Figure 49](#), page 39.
 11. Change the potentiometer (RV1) on the target board.
 12. Go back to HyperTerminal and press any key. You should see AC4 change value, based on the potentiometer (RV1). The full range is 0 V to 3.3 V.
 13. Click **Pause**  to stop the code execution. Notice that the call stack is updated and the next line of C code is highlighted.
 14. Set a breakpoint at the line of code highlighted in [Figure 60](#), page 47 by double-clicking in the left margin next to the line of code. This line of code is near line 200.

Figure 60 • Set Breakpoint



```

206
207
208/*****
209 *
210 */
211uint8_t key_pressed( void )
212{
213     uint8_t ret_val = 0;
214     size_t rx_size;
215     uint8_t rx_buffer[16];
216
217     rx_size = UART_get_rx( &g_the_uart, rx_buffer, sizeof(rx_buffer) );
218     if ( rx_size > 0 )
219     {
220         ret_val = 1;
221     }
222
223     return ret_val;
224}
225
226/*****
227 *

```

A dot appears in the margin indicating that a breakpoint is set on that line of code.

15. Click **Resume**  to continue executing the code. The debugger should stop at the line of code where you set a breakpoint.
16. To open the Registers view, from **Windows** select **Show View**, and then select **Registers**. You cannot see the Register option, if the Register view is already open. You can expand the register to view the values.
17. Click Step Over . The values of the registers change on clicking Step Over. The changed register values are highlighted.

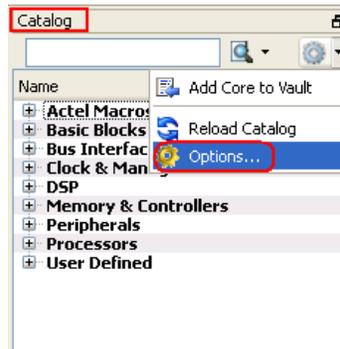
Congratulations! You have successfully created a Cortex-M1 software project and debugged it, running on a Microsemi FPGA development kit. For more information on the features of SoftConsole, refer to [Debugging the Application Using SoftConsole](#), page 40.

9 Appendix: Libero SoC Catalog Settings

The following steps show how to configure your vault location and set up the repositories in Libero SoC.

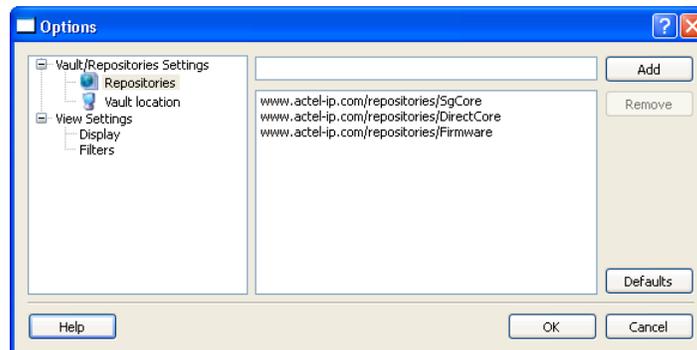
1. On the **Catalog** window, click **Settings** and then click **Options**.

Figure 61 • Catalog – Options



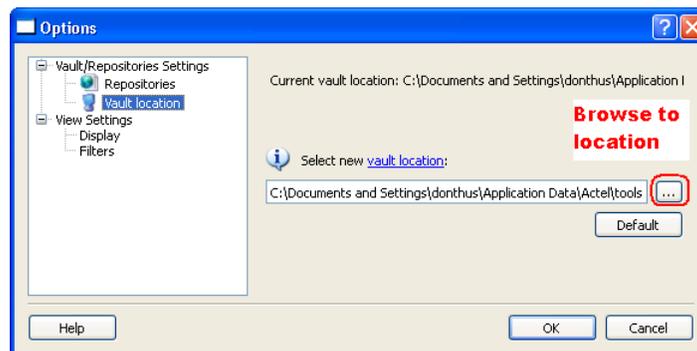
2. The Settings window is displayed as shown in the following figure.
3. Click **Repositories** and add the following in the address field:
 - www.actel-ip.com/repositories/SgCore
 - www.actel-ip.com/repositories/DirectCore
 - www.actel-ip.com/repositories/Firmware
4. Click **Add** after entering each path.

Figure 62 • Setting Repositories



5. Click **Vault location** in the **Options** window. Browse to a location on your PC to set the vault location where the IPs can be downloaded from the repositories.
6. Click **OK**.

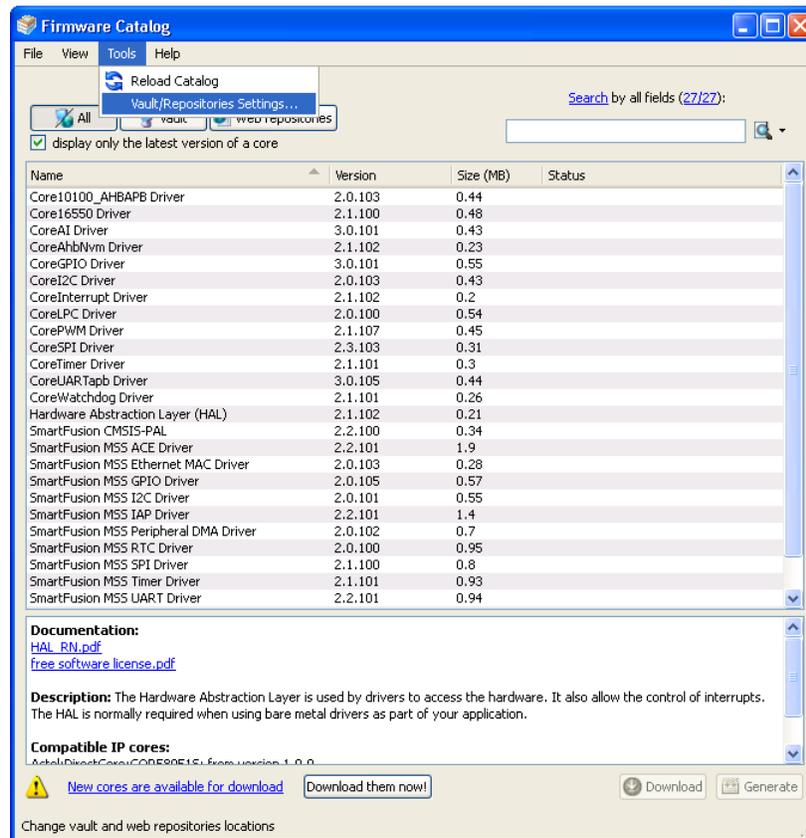
Figure 63 • Setting the Vault Location



10 Appendix: Firmware Catalog Settings

1. Open <Libero Installation directory>\Designer\bin\catalog.exe.
2. Select **Tools > Vault/Repositories Settings**, from the Firmware Catalog widow.

Figure 64 • Firmware Catalog Settings



3. Select **Repositories** under **Vault/Repositories Settings** in the Options dialog box as shown in [Figure 63](#), page 48.
4. Confirm that the following repositories are displayed (add them if needed):
 - www.actel-ip.com/repositories/SgCore
 - www.actel-ip.com/repositories/DirectCore
 - www.actel-ip.com/repositories/Firmware
5. Add the above mentioned paths in the address field if required by selecting the repository and clicking **Add**.
6. Click **Download them now!** to download the new cores to the vault.

11 Appendix: Debugging Features in SoftConsole

You can set a breakpoint using any of the following methods:

1. Place the cursor on the line of code and select **Run > Toggle Breakpoint**.
2. Right-click the line of code and select **Toggle Breakpoint**.
3. Double-click the left margin next to the line of code.
The Breakpoints, Registers, and Variables tabs are displayed on the top right corner. Elements are highlighted if the values are changed.

To single step through the source code:

1. From **Run** select **Step Into** or click  icon and from **Run** select **Step Over** or click  icon.
2. The Step Return steps to the next instruction of the calling function. From **Run** select **Step Return** or click  icon.
3. To view disassembly, choose **Window > Show View > Disassembly**. You can then click the **Instruction Stepping Mode**  icon to step by processor instruction.