ARM® Cortex™-M1 Embedded Processor Hardware Development Tutorial

for Fusion Mixed-Signal FPGAs

Printed in the United States of America

Part Number: 50200154-2

Release: November 2009

Trademarks

Table of Contents

Introduction

This document shows you how to create a Cortex™-M1 processor system that runs on one of the Actel Fusion embedded development kit boards. This design can be used as a starting point for developing your Cortex-M1 embedded system targeting Actel's Fusion FPGA devices.

Because this tutorial targets one of Actel's Fusion devices, you need to become familiar with Fusion features and architecture. The Fusion datasheet and user's guide are on the Actel website:

http://www.actel.com/documents/Fusion_DS.pdf

http://www.actel.com/documents/Fusion_UG.pdf

After completing this tutorial, you will know the hardware design flow for creating a Cortex-M1 embedded system using Libero® Integrated Design Environment (IDE) 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
- Functional simulation using a bus functional model (BFM)
- Synthesis, place-and-route of hardware design, and generating FPGA programming image
- Programming an Actel FPGA with Cortex-M1 system ready for software design

For this tutorial, we will use the binary file from previously developed software. The binaries will be written into the memory on the target development kit, along with the programming bit files for the hardware designed in this tutorial. If you want to learn about the software development flow for the Cortex-M1 processor, you can study the *ARM Cortex-M1 Embedded Processor Software Development Tutorial*:

http://www.actel.com/documents/CortexM1_Proc_SW_Tutorial_UG.pdf.

Requirements for this Tutorial

Hardware

To complete this tutorial, you need one of the following boards with the corresponding 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 you need (except the PC) is provided with the development kit. For more information about the development boards, refer to Design Hardware page:

www.actel.com/products/hardware/default.aspx

Software

The instructions are based on the Actel Libero IDE v8.6 SPA 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.

Intellectual Property (IP)

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

- Cortex-M1 version 2.7.103
- CoreAHBLite version 2.0.140
- CoreAPB version 1.1.101
- CoreAHB2APB version 1.1.101
- CoreAhbNvm version 1.3.135
- CoreAhbSram version 1.3.103
- CoreMemCtrl version 2.0.105
- CoreUARTapb version 4.0.120
- CoreGPIO version 1.2.103
- CoreAI version 3.0.119

Licensing

You will need a license for these IP cores. If you do not have a license, go to the Customer Portal at https://www.actel.com/portal/, sign in and choose *Licenses & Registration* on the left navigation. Click the **Request Free License** button and choose the **Libero Gold** license. Follow the instructions in the email sent to you to set up the license on your machine.

1 – Design Overview

Cortex-M1 System Description

The design contains a Cortex-M1 processor system running on an Actel 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 sampled data, and sends the result over UART. There is a potentiometer on the board to change the analog voltage being sampled. In this tutorial you will communicate with the target using Hyper Terminal.

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 Actel 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 machine (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 an Actel Fusion AFS1500 device that has 1 MByte of embedded flash memory (also referred to as nonvolatile memory, or NVM) and 30 KBytes of internal SRAM. The Fusion Embedded Development Kit board has 1 Mbyte of SRAM, comprised of two 4 Mbit × 16 bit chips. The Fusion Advanced Development Kit board has 2 Mbytes of SRAM, comprised of two 1 Mbit x 16 bit chips. This hardware design connects to all of these memories, not necessarily using the entire memory space of each device.

Actel's Fusion devices have an on-chip 100 MHz RC oscillator. You will 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 debug interface will not be used in this hardware design tutorial. However, it is required for the *ARM Cortex-M1 Embedded Processor Software Development Tutorial for Fusion Mixed-Signal FPGAs* (www.actel.com/documents/CortexM1_Proc_SW_Tutorial_UG.pdf), which uses hardware created in this design tutorial.

A block diagram of the design is shown in Figure 1-1.

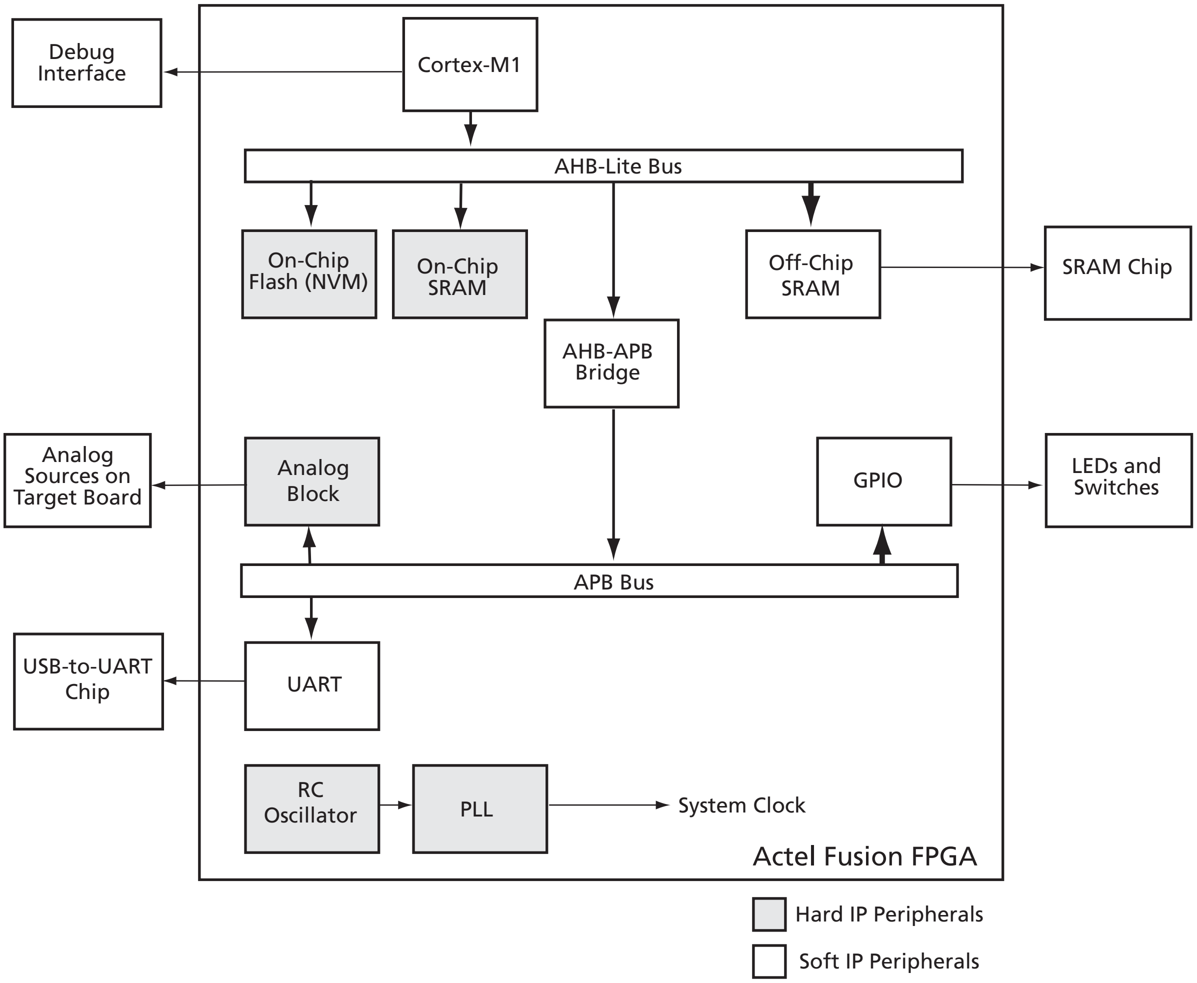

Figure 1-1 • **Block Diagram**

2 – Before You Get Started

Download Tutorial Files

Before starting the tutorial, you need to download the tutorial files from the website for the board you are targeting. Go to Actel's Design Hardware web page and click the link for the board you are targeting (http://www.actel.com/products/hardware/default.aspx). On the web page of the target board, click on the zip file under the *ARM Cortex-M1 Embedded Processor Hardware Development Tutorial* document.

The files are placed in a folder called CortexM1_Fusion_HW_Tutorial.

If you browse the contents of the zip file, you will see a subdirectory for every board this tutorial supports. The directory structure is similar for each board. As an example, Figure 2-1 shows a picture of the directory structure for the Fusion Embedded Development Kit board (M1AFS-EMBEDDED-KIT).

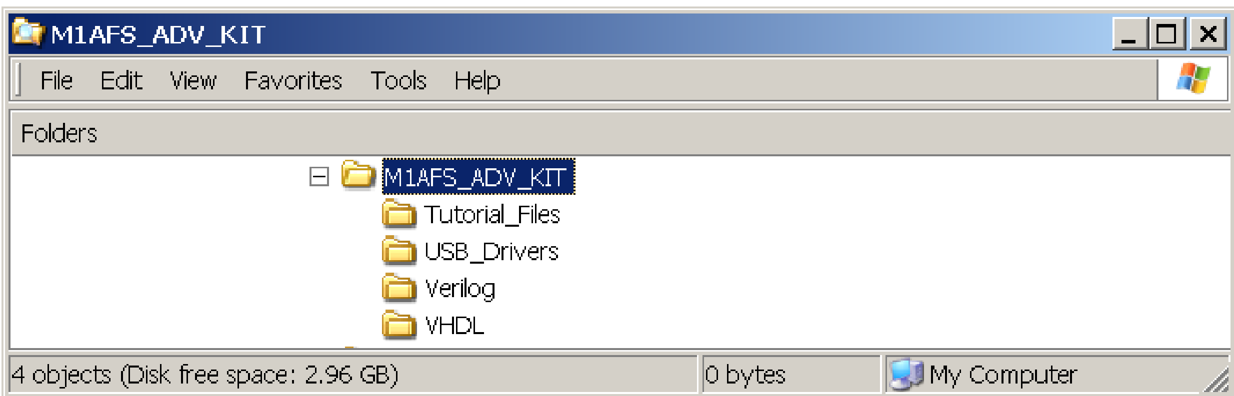

Figure 2-1 • **Tutorial Files**

The *Verilog* and *VHDL* subdirectories contain the complete VHDL and Verilog projects for this tutorial design, for your reference. The *Tutorial_Files* subdirectory contains the source files you need to complete this tutorial. Table 2-1 gives a description of these files. The names of the files will be slightly different depending on the board you are targeting.

Table 2-1 • **Description of Tutorial Files**

File Name	Description
M1AFS_EMB_TUT	Pin constraints for the Fusion Embedded Development Kit board
TUTORIAL.HEX	Intel-Hex file which contains Cortex-M1 software code
M1AFS_EMB_TUT_<VERILOG/VHDL>.PDC	Constraint to put JTAG reset and clock net on global

The *USB_Drivers* subdirectory contains the driver for the USB-to-UART chip on the target board. This driver allows you to communicate with the target board over USB by treating the USB port as a COM port. Therefore, you can use HyperTerminal to connect to a UART in the design running on the board.

Install USB-to-UART Driver

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

Setting Up the USB-to-UART Driver

1. Make sure that a terminal emulation program such as HyperTerminal, which is included with Windows,® is installed on your PC.
2. Install the drivers for the USB to RS-232 Bridge by double-clicking on the **PreInstaller.exe** executable located in the *USB_Drivers* subfolder. Accept the default installation folder and press the **Install** button. Press the **Continue Anyway** button if prompted.
3. Connect USB and power cables according to Table 2-2.

Table 2-2 • **Power and USB Connections**

Board	Power and USB Connections
M1AFS-EMBEDDED-KIT	Make sure you have 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 in the ON position to supply power to the board.

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

4. Select **Install the software automatically** (recommended) and press **Next**. Once installation has completed, press **Finish**. Repeat the driver installation steps a second time (if prompted). Press the **Continue Anyway** button if prompted.
5. 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 2-2 it is assigned to COM8).

If you do not see SFE USB to RS232 Controller in the Device Manager (Figure 2-2), you may need to reboot your machine.

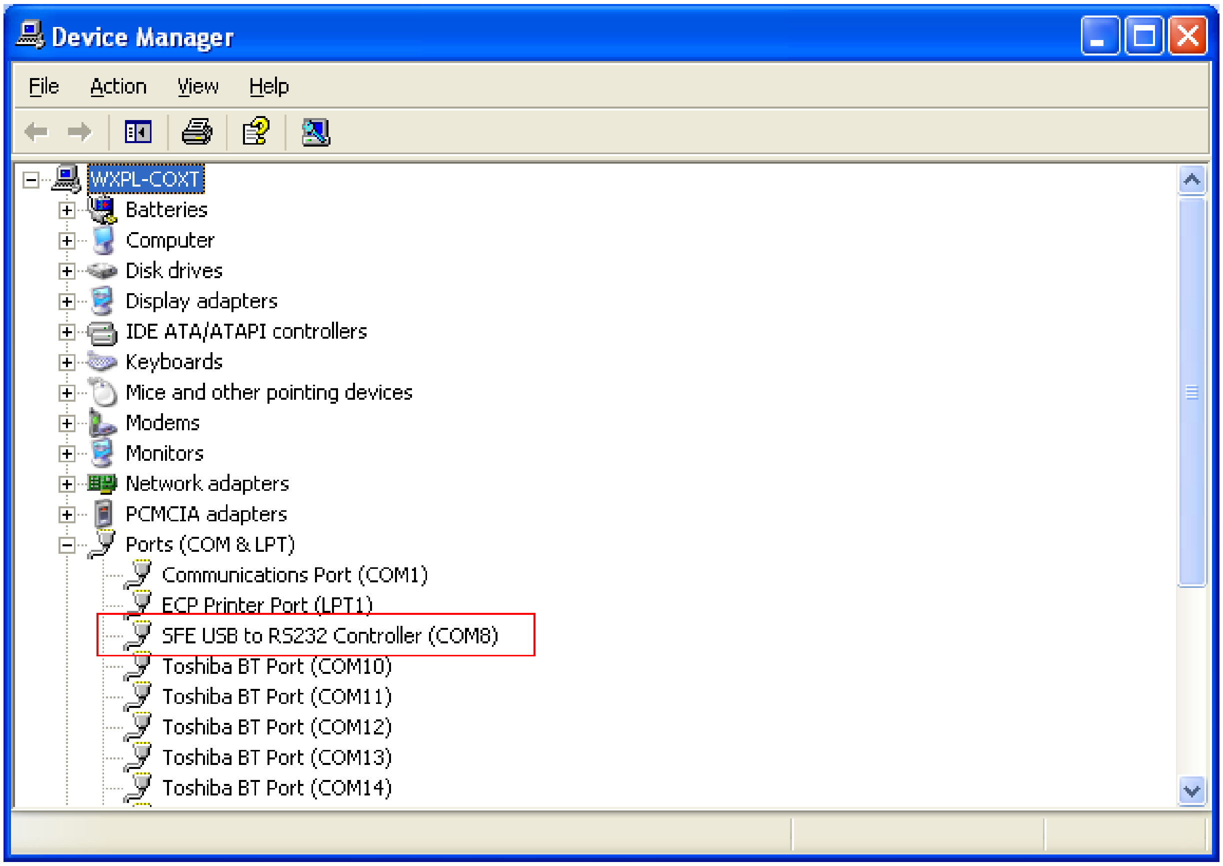

Figure 2-2 • **Windows Device Manager**

1. Connect the FlashPro3 programming interface according to Table 2-3.

Table 2-3 • **Programming Interface Connections**

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

2. If you are prompted to install drivers for the FlashPro3 hardware, refer to the *FlashPro User's Guide at* http://www.actel.com/documents/flashpro_ug.pdf.

3 – Tutorial – Create Design

Step 1 – Create an Actel Libero IDE Project

Open the Libero IDE tool and complete the following steps to create a Libero IDE project.

1. Go to **Project** > **New Project**. The New Project Wizard window opens.
2. Type **M1AFS_EMB_TUT** (for M1AFS-EMBEDDED-KIT) or **M1AFS_EMB_TUT** (for M1AFS-ADV-DEV-KIT) in the Project name field.
3. Click the **Browse** button and choose a location on your machine for the project.
4. Select your preferred HDL language. Your window should look similar to Figure 3-1.

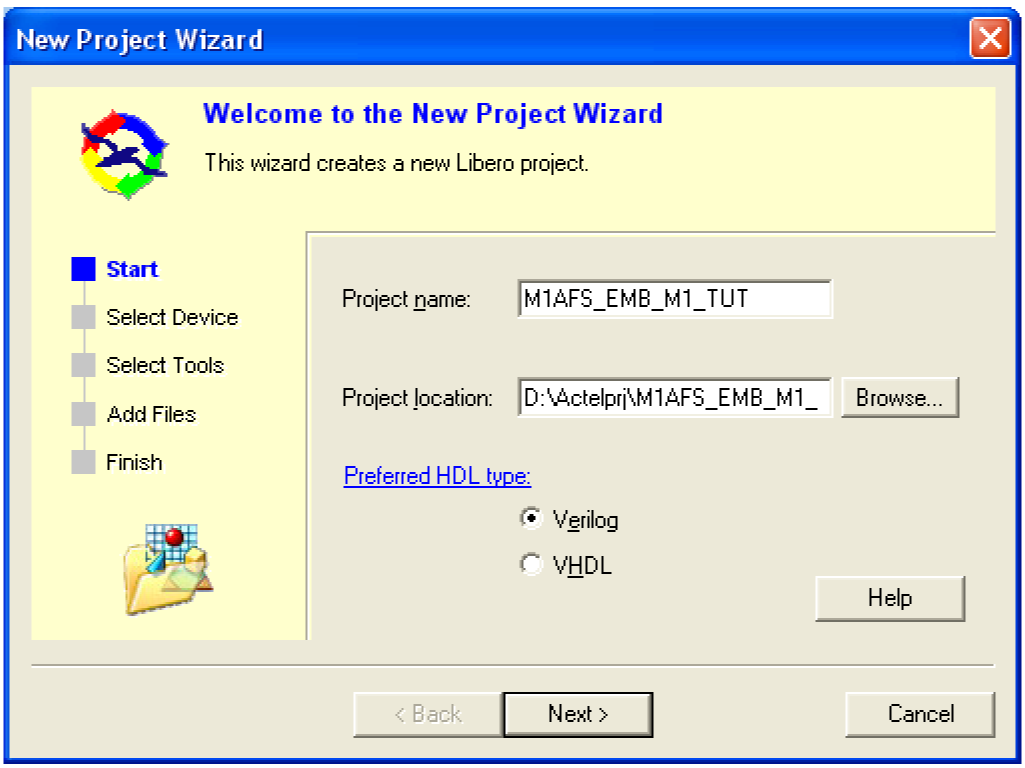

Figure 3-1 • **New Project Wizard**

5. Click **Next**. The Family, Die and Package options appear (Figure 3-2 on page 3-14).
6. Select **Fusion** for the Family.
7. Select **M1AFS1500** for the Die.

8. Select **484 FBGA** for the Package

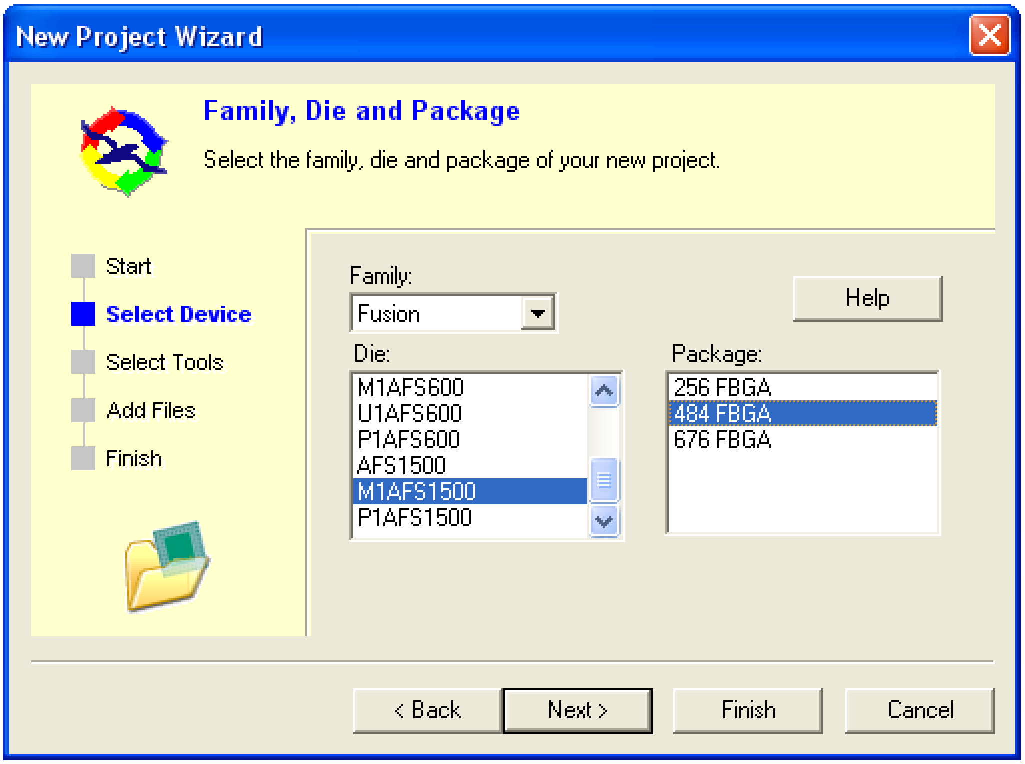

Figure 3-2 • **Select Family, Die, and Package**

9. Click **Next** two times. The Add files to your project options appear.
10. Click the **Add Files** button.
11. Select **Physical Design Constraint Files (*.pdc)** for the Files of type box.
12. Browse to the following directory:
    CortexM1_Fusion_Tutorial\<board>\Tutorial_Files
13. Select **M1AFS_EMB_TUT.PDC** and **M1AFS_EMB_TUT_<VERILOG/VHDL>.PDC** or **M1AFS_ADV_TUT.PDC** and **M1AFS_ADV_TUT_<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 a timing constraints file (SDC) because the internal clock source from the RC oscillator is constrained automatically by SmartTime.

14. Click **Add**. You should see the file you selected under the Constraint Files section.
15. Click **Finish** to create the Libero IDE project.

Step 2 – Create a SmartDesign Component within Actel Libero IDE

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

1. Click the SmartDesign icon in the Design Entry Tools area (in the Project Flow tab, as shown in Figure 3-3.

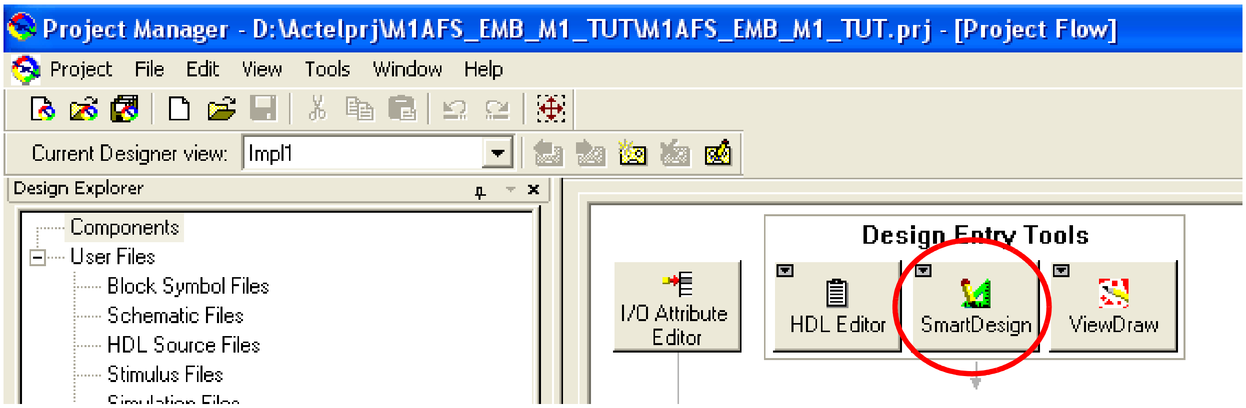

Figure 3-3 • **SmartDesign Icon**

2. Type **M1AFS_TUT_TOP** in the Name field, as shown in Figure 3-3.

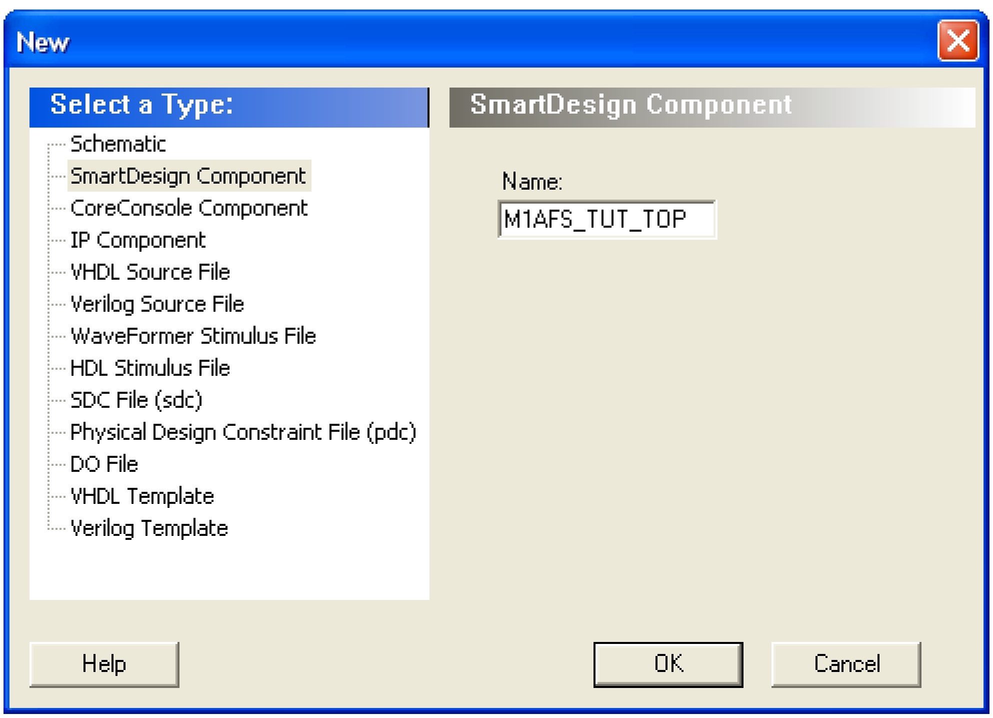

Figure 3-4 • **Select SmartDesign Component**

3. Click **OK**. A blank canvas opens in SmartDesign, which looks like Figure 3-5. Notice the new M1AFS_TUT_TOP tab (next to the Project Flow tab) which contains a Canvas tab. This is your new SmartDesign canvas window.

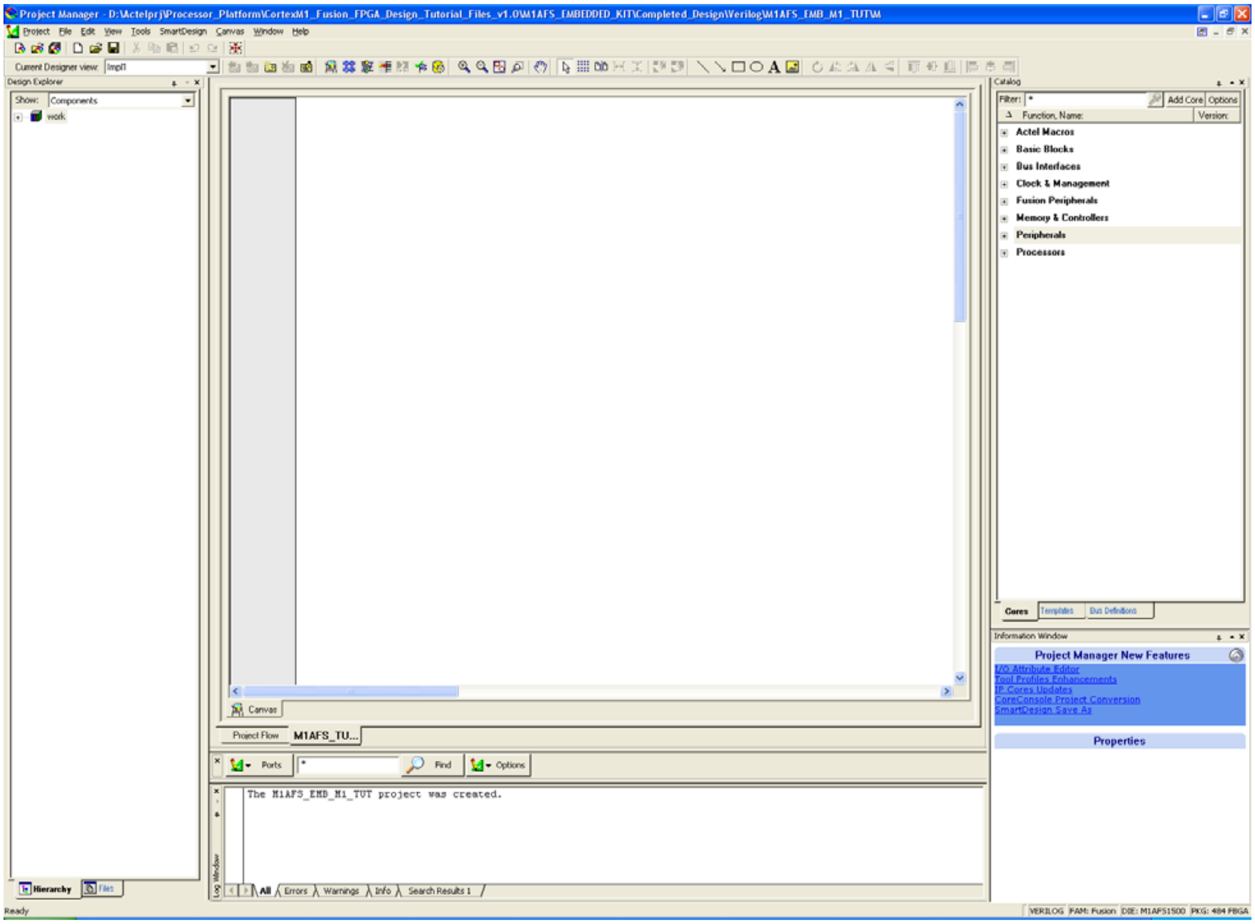

Figure 3-5 • SmartDesign Canvas Window

In the following steps you use SmartDesign to create a Cortex-M1 system, as shown Figure 3-7 on page 3-18. When the instructions tell you to instantiate a component, instantiate it in the SmartDesign Canvas.

As you are instantiating the components, Actel recommends you use the same version of DirectCore IP. Otherwise, you may receive different results.

Obtaining a Different Version of DirectCore IP

1. Click the **Options** button in the Catalog. You will see the Catalog Display Options window (Figure 3-6).

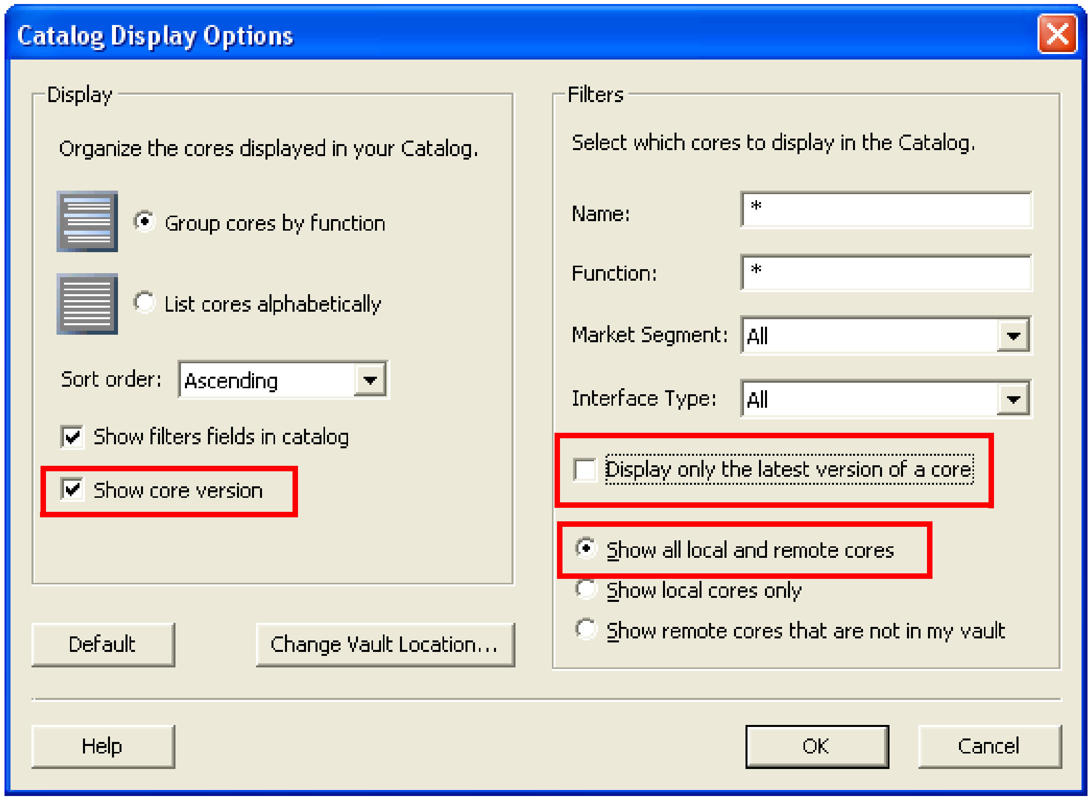

Figure 3-6 • **Catalog Display Options**

2. Clear the check box for **Display only the latest version of a core**. Make sure you have selected both **Show core version** and **Show all local and remote cores**.
3. Click **OK**. If you still do not see the version you need in the Catalog, then proceed to the next step.
4. Click the **Add Core** button and choose **From Web Repository**. The Add Cores to Vault window appears.
5. Select the core(s) that you need and click **Add**.
6. Click **Done** once the new cores have been downloaded.

Instantiate Cortex-M1 Processor

The Cortex-M1 processor is a 32-bit soft ARM® processor designed to be implemented in an FPGA. Follow the steps below to instantiate the Cortex-M1 processor.

1. Go to the Catalog and expand the Processors category.
2. Instantiate by clicking and dragging **Cortex-M1 version 2.7.103** onto the SmartDesign canvas.
3. Select FlashPro3 as the Debug Interface. This allows software debugging over JTAG using Actel's SoftConsole® development tool.

Note: If you are planning to use third-party tools (Keil MDK-ARM, IAR Embedded Workbench, ARM RealView, etc.), select RealView JTAG as the Debug Interface. However, keep in mind that this document assumes you are using Actel's SoftConsole development tool. Also, the succeeding Cortex-M1 Software Design Tutorial requires the FlashPro3 debug interface.

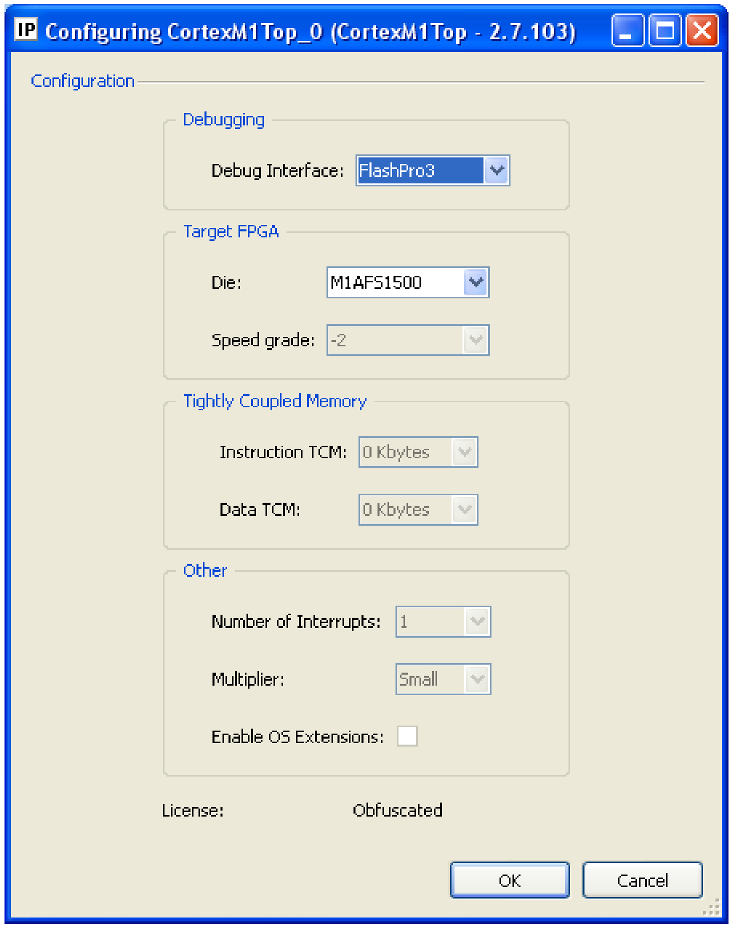

Figure 3-7 • **Configuring Cortex-M1 Processor**

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

Instantiate CoreAHBLite Bus

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

1. Go to the Catalog and expand the Bus Interfaces category.
2. Instantiate **CoreAHBLite version 2.0.140** with the default settings (all slots enabled).

Note that the configuration window allows you to disable slots that you are not using. For this tutorial design, leave all slots enabled (the default).

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 MByte of internal NVM composed of 4 embedded flash memory blocks, each 256 Kbytes in size. However, for this design you will use only 1 embedded flash memory block, 256 Kbytes. Follow the steps below to instantiate the CoreAhbNvm.

1. Go to the Catalog and expand the Memory & Controllers category.
2. Instantiate and configure **CoreAhbNvm version 1.3.135** with a size of 256 Kbytes. The CoreAhbNvm settings are shown in Figure 3-8.

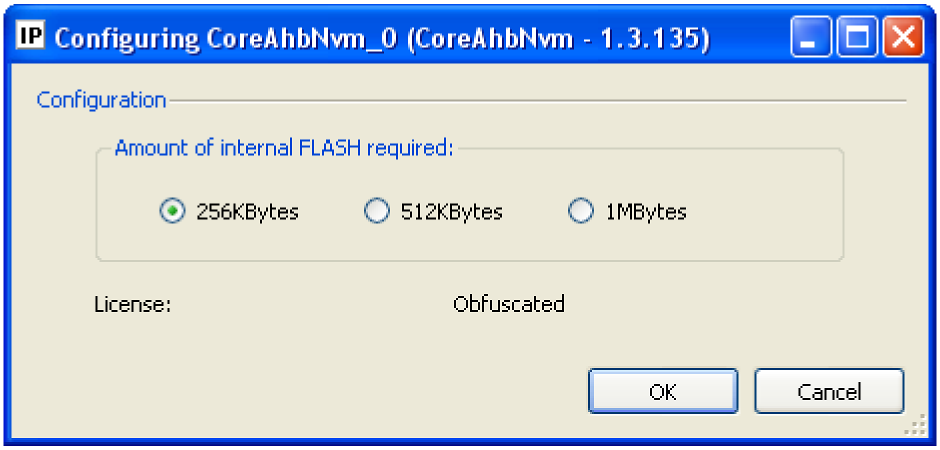

Figure 3-8 • **Configuring CoreAhbNvm**

Instantiate CoreMemCtrl

CoreMemCtrl creates an interface to off-chip SRAM and/or flash with shared data and address busses. 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 will use the SRAM interface only. The system clock in this design is 20 MHz, which has a period of 50 ns. The asynchronous SRAM has a 10 ns access time. Therefore, you can set the wait states to their minimum values: 0 for read wait states and 1 for write wait states.

1. Go to the Catalog and expand the Memory & Controllers category.
2. Instantiate **CoreMemCtrl version 2.0.105** to interface to the off-chip asynchronous SRAM on the development board.
3. Take 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 the off-chip memory address is word addressed (32-bit word). The other settings refer to half-word and byte addressing. You can leave the other options at their default setting since this design does not use this flash interface.

The corresponding CoreMemCtrl settings are shown in Figure 3-9.

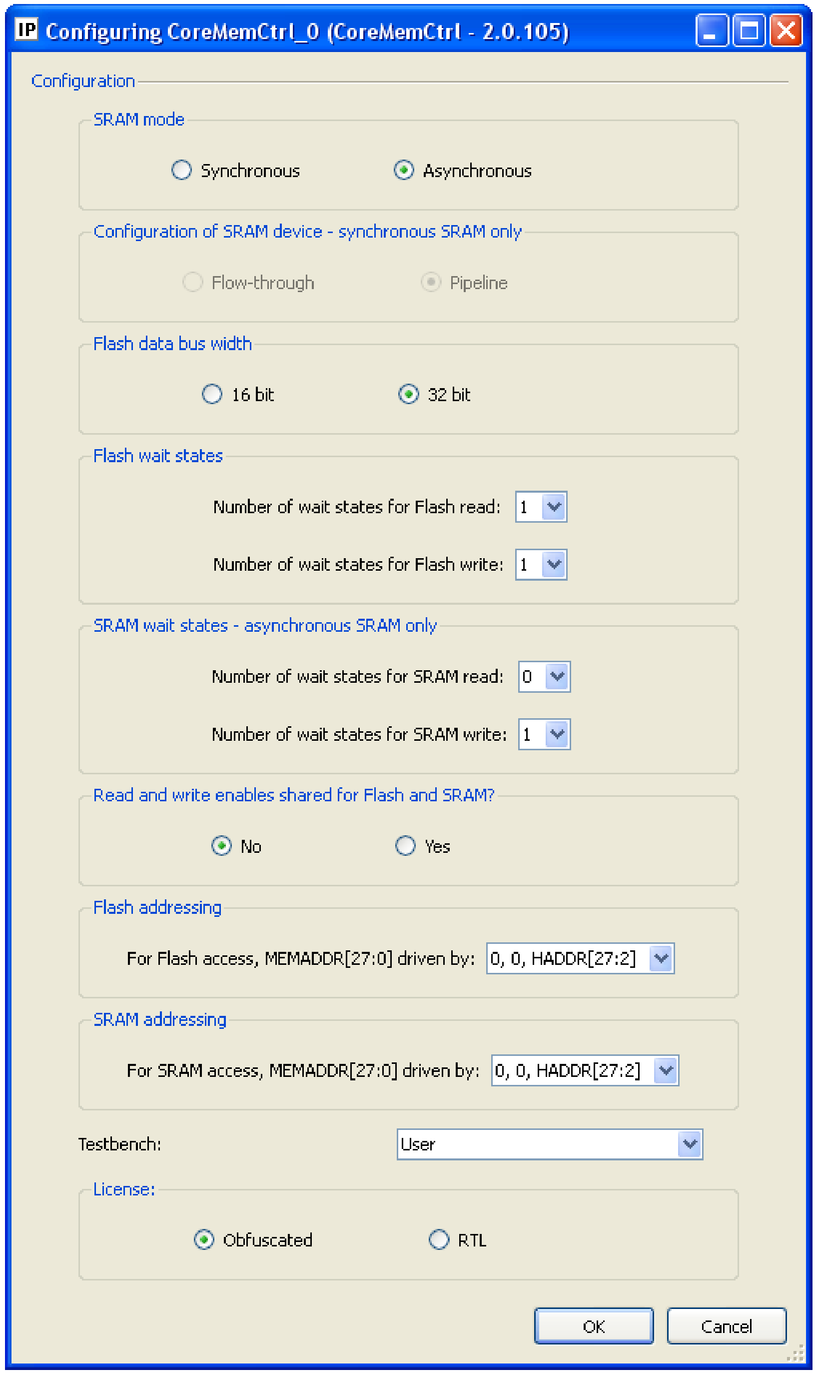

Figure 3-9 • **Configuring CoreMemCtrl**

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 Kbytes of internal SRAM consisting of 60 RAM blocks each 0.5 Kbyte in size. The Cortex-M1 uses 2 Kbytes and the CoreUARTApb (which you instantiate later) uses 1 Kbyte in FIFO mode. Therefore, CoreAHBSram must use no more than 27 Kbytes. You will configure it for 14 Kbytes. Follow the steps below to instantiate the CoreAhbSram.

1. Go to the Catalog and expand the Memory & Controllers category.
2. Instantiate and configure **CoreAhbSram version 1.3.103** with a size of 14 Kbytes. The CoreAhbSram settings window is shown in Figure 3-10.

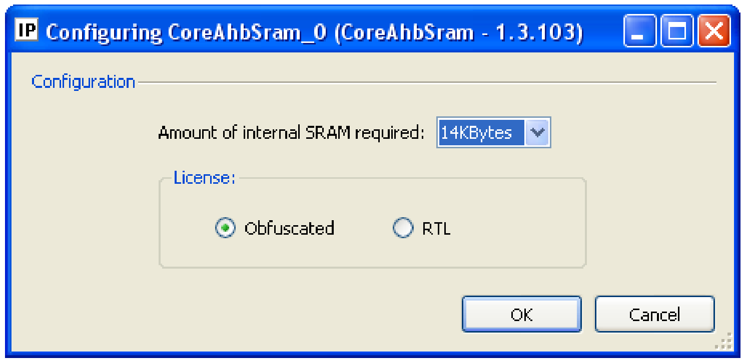

Figure 3-10 **• Configuring CoreAhbSram**

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. Go to the Catalog and expand the Bus Interfaces category.
2. Instantiate the **CoreAHB2APB version 1.1.101** to bridge to the APB bus.

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 MBytes in the processor's memory space for a total of 256 MBytes. Due to the performance, typically slower, low priority peripherals are placed in the APB bus. Follow the steps below to instantiate the CoreAPB.

1. Go to the Catalog and expand the Bus Interfaces category.
2. Instantiate the **CoreAPB version 1.1.101** bus with the default settings (all slots enabled).

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

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. Go to the Catalog and expand the Peripherals category.
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 at their default values.

- 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 3-11 through Figure 3-14 on page 3-25 show the instance of CoreAI with the corresponding parameters set.

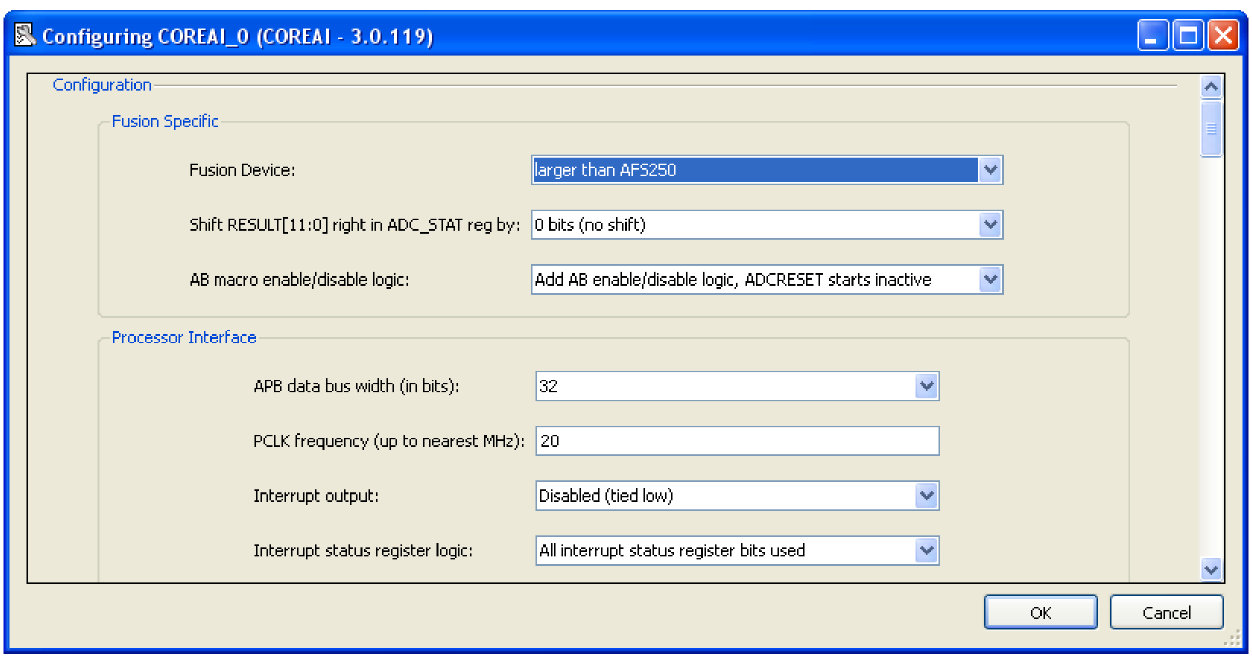

Figure 3-11 • **Configuring CoreAI (part 1)**

Figure 3-12 • **Configuring CoreAI (part 2)**

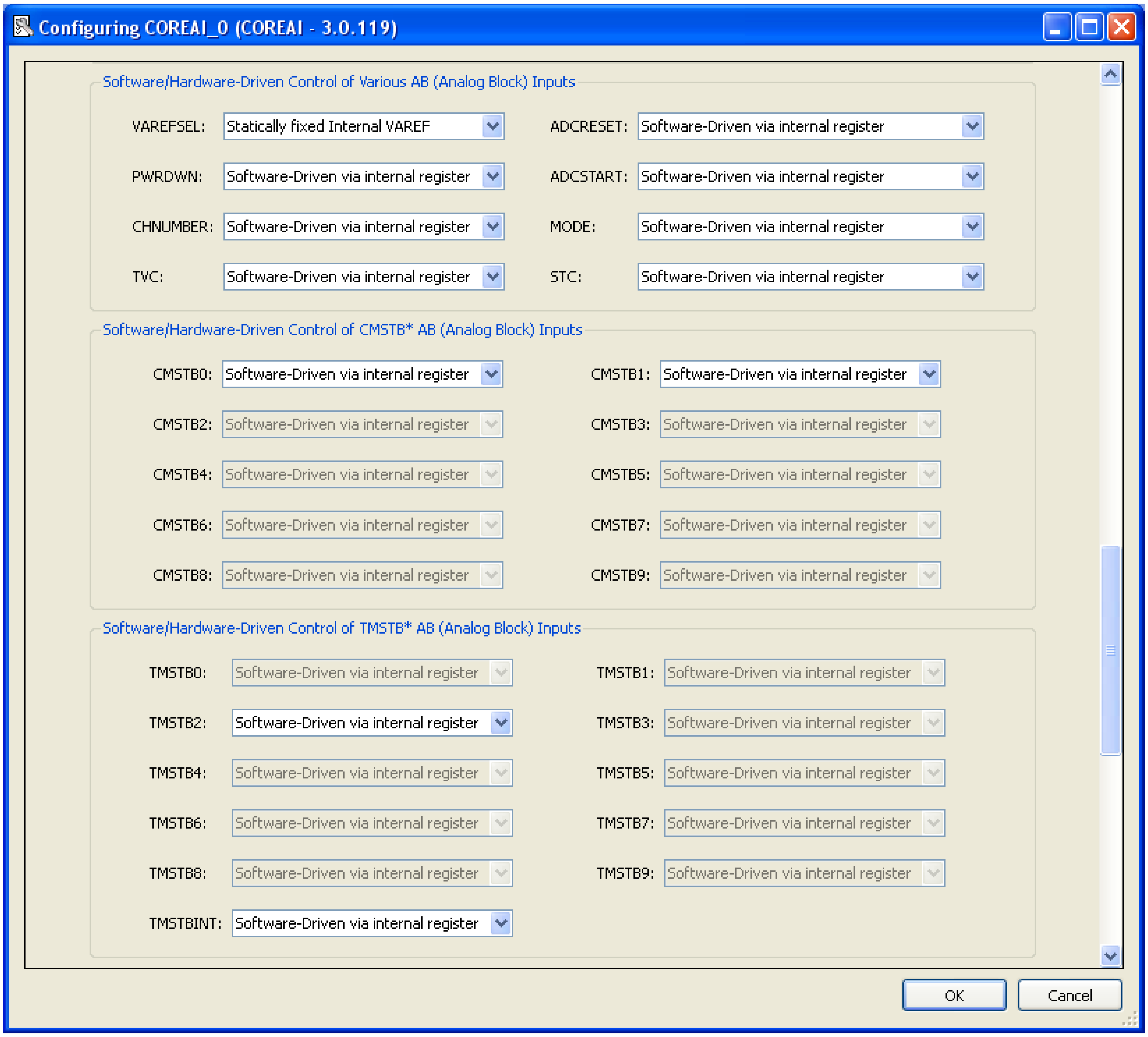

***Figure 3-13* • Configuring CoreAI (part 3)**

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

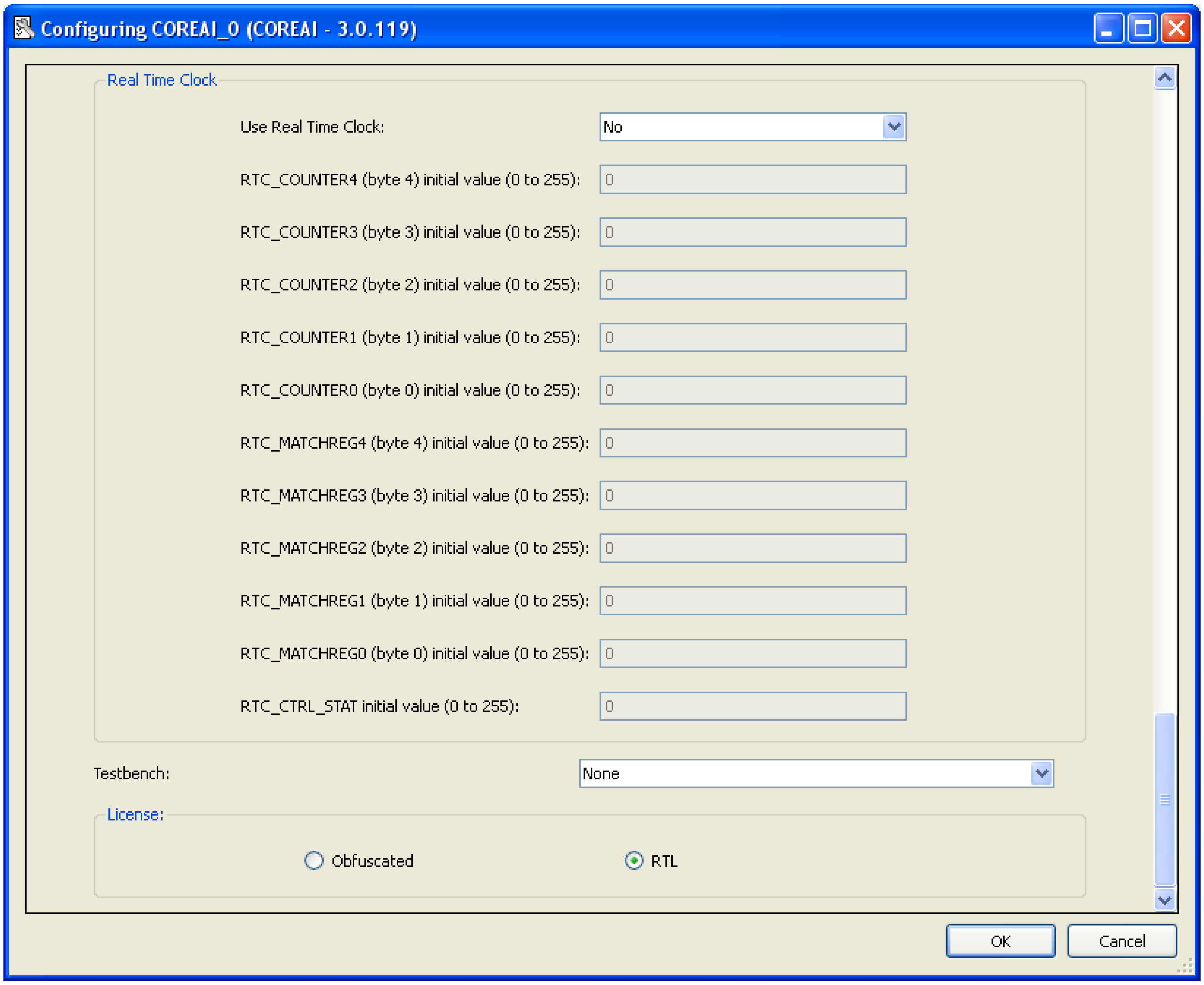

Figure 3-14 • **Configuring CoreAI (part 4)**

Instantiate CoreUARTapb

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

1. Go to the Catalog and expand the Peripherals category.
2. Instantiate and configure **CoreUARTapb version 4.0.120** with the following selections:
   – TX FIFO: Enable TX FIFO
   – RX FIFO: Enable RX FIFO
   – Configuration: Programmable

The CoreUARTapb configuration is shown in Figure 3-15.

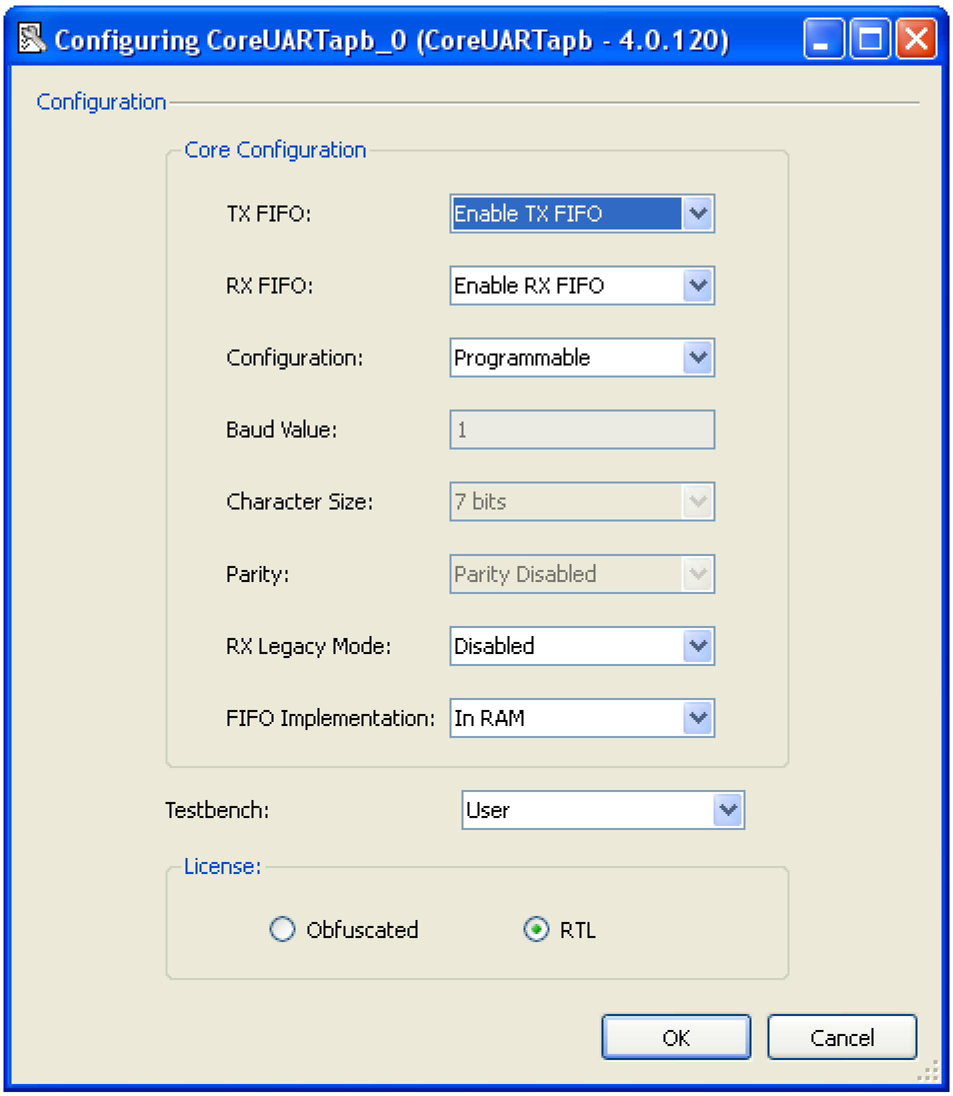

***Figure 3-15* • Configuring CoreUARTApb**

The Baud value, Character Size and Parity parameters are greyed out since Configuration is set to Programmable. They can be modified by the processor when the UART is initialized. EQ 3-1 is used to calculate the Baud Value parameter based on the desired baud rate (57600) and system clock frequency (20 MHz) in Hz:

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

EQ 3-1

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. Go to the Catalog and expand the Peripherals category.
2. Instantiate **CoreGPIO version 1.2.103** with the default settings (32 inputs and 32 outputs).

Instantiate RC Oscillator

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

1. Go to the Catalog and expand the Fusion Peripherals category.
2. Instantiate Oscillator - RC with Clock conditioning circuit selected.
3. Name it myRCOSC.

Instantiate PLL

The PLL inside of the FPGA can be used to generate new clock signals from a source clock. For this design, you will 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. Go to the Catalog and expand the Clock & Management category.
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)

Figure 3-16 shows the PLL configuration window with the correct settings.

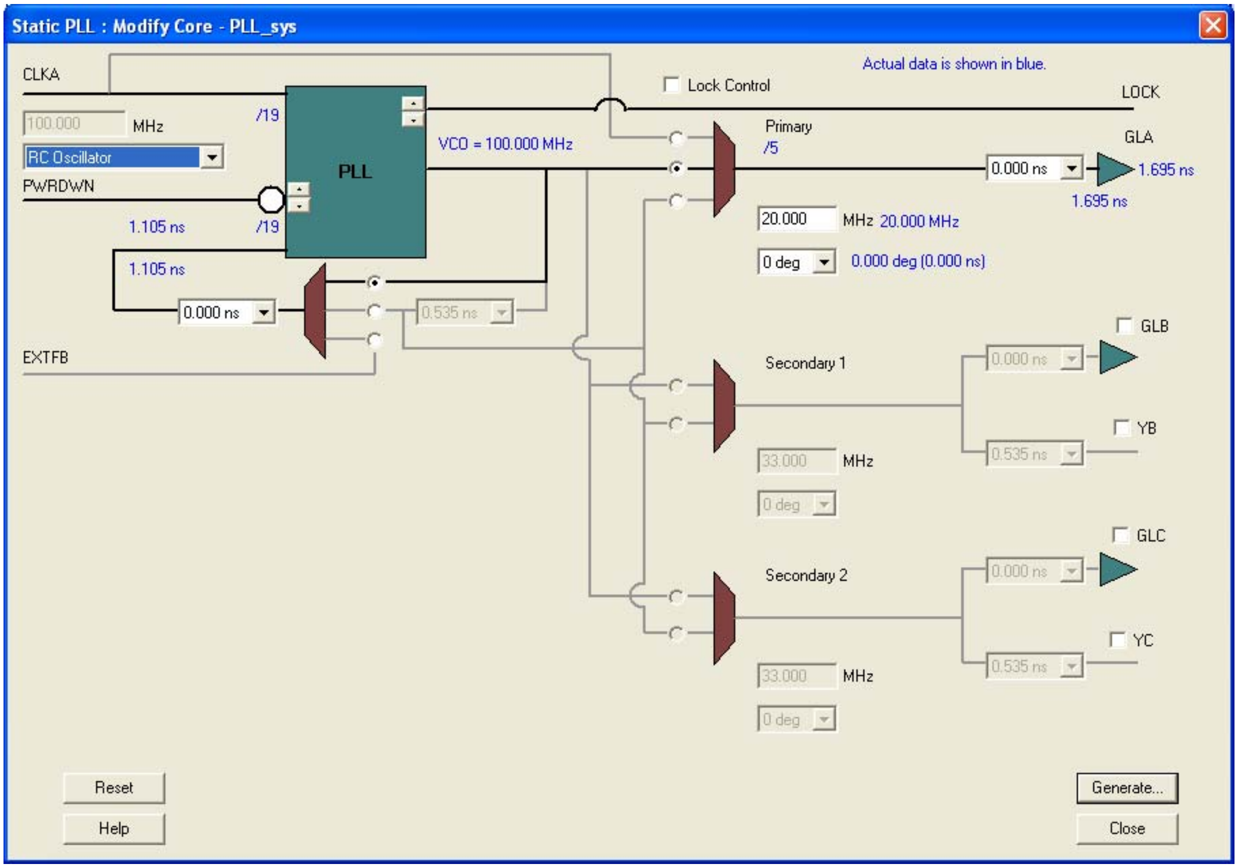

Figure 3-16 **• Configuring the PLL**

3. Name the PLL PLL_sys.

Instantiate AND2 Gate

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

1. Go to the Catalog and expand the Actel Macros category.
2. Instantiate AND2.

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 on canvas and choose **Auto Connect)**. The Modify Memory Map window will open (Figure 3-17).
2. Click on **CoreAHBLite_0**. Configure the memory map for the CoreAHBLite bus as shown in Figure 3-17 by following the steps below for each peripheral that you need to move.
   - Click on the name of the peripheral in the Peripheral column.
   - Select the blank to remove this peripheral from this address.
   - Click on the Peripheral column next to the desired address of this peripheral.
   - Select the name of the peripheral from the list.

   If you swap two peripherals, you must blank both address locations and then select each peripheral at the corresponding addresses. Note that you will have to remove the CoreMemCtrl_0:AHBslave_flash peripheral from the memory map, since it is not being used. Both SRAM and flash peripherals are added in automatically when using CoreMemCtrl.

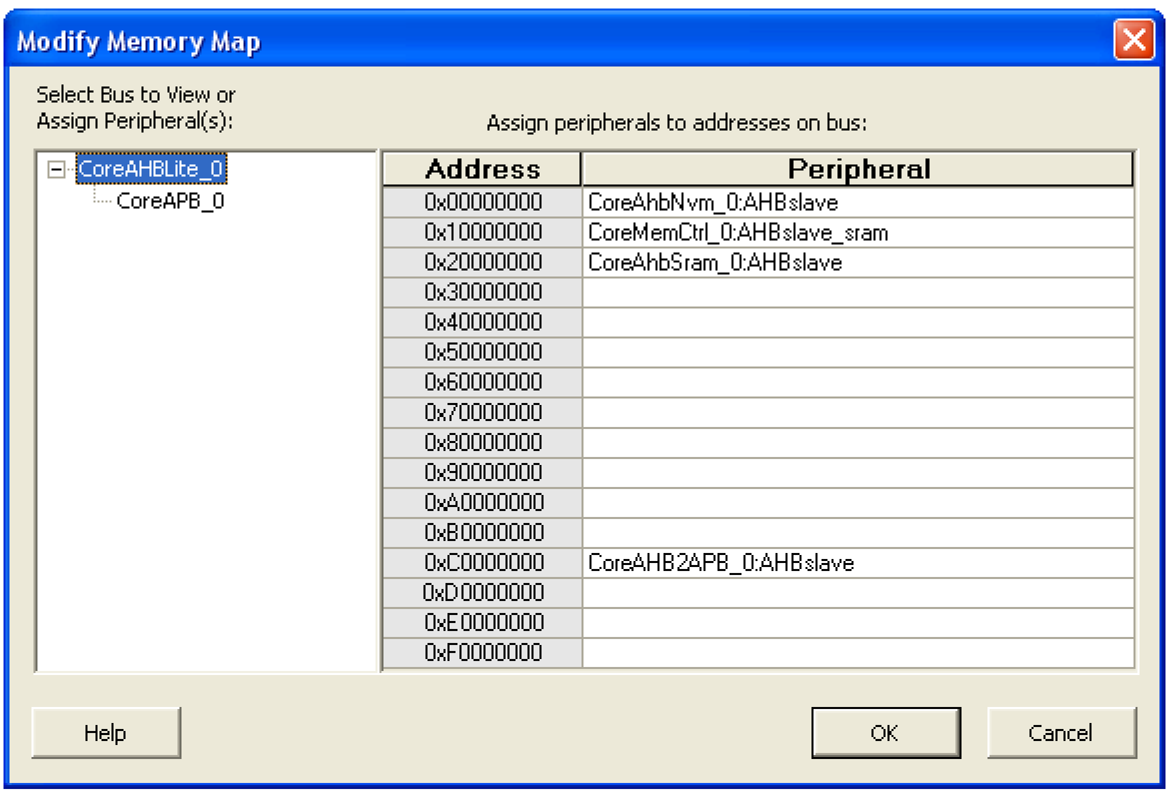

***Figure 3-17* • Modify Memory Map for CoreAHBLite_0**

3. Click on CoreAPB_0. Configure the memory map for the CoreAPB bus as shown in Figure 3-18 on page 3-30 using the same steps as above.

4. Click **OK**.

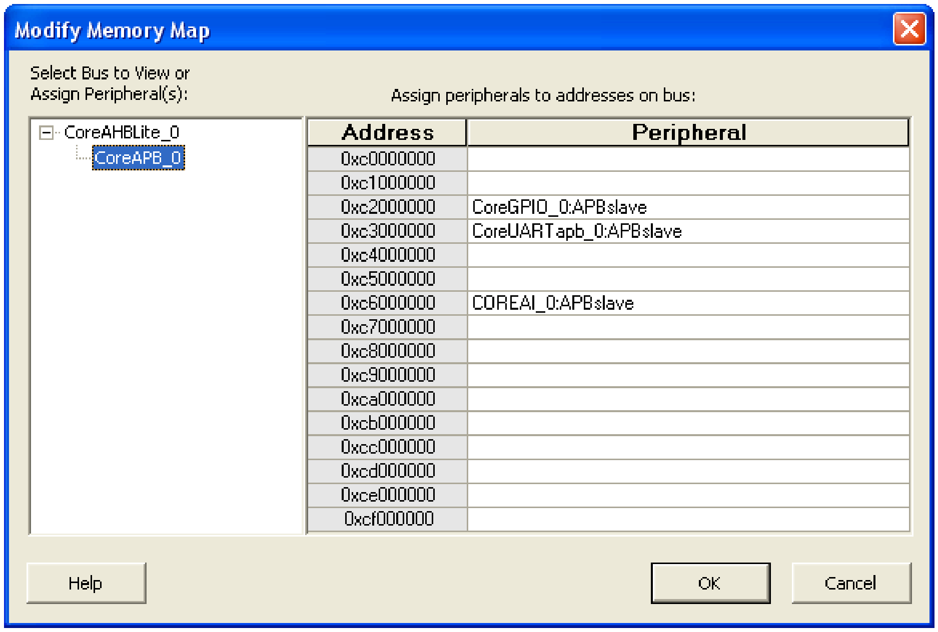

***Figure 3-18* • Configure Memory Map for CoreAPB_0**

Take careful note to configure Memory Map dialog box as shown in Figure 3-17 on page 3-29 and Figure 3-18 as this directly defines the subsystem memory map and has a direct impact on the Cortex-M1 software code.

You may want to use the Auto Arrange Instances features in SmartDesign by right-clicking and choosing **Auto Arrange Instances**. This will place the peripherals in an orderly manner on the canvas.

Connect Signals Manually in SmartDesign

Some signals must be manually connected. There are two ways to manually connect signals in Smart Design. You can use the Canvas in a graphical manner or use the Grid in a spreadsheet manner. For this tutorial, use the Grid to connect the signals in Table 3-1 following the steps below.

1. Open the Grid by going to **SmartDesign** > **Show Grid View**.
2. Click the Lock signal cell from the PLL_sys_0 instance on the left column.
3. Scroll to the right to find the AND2_0 instance (on the top).
4. Click the cell which is the intersection of these two.
5. Select the B signal from the list box.
6. Follow steps 2-5 for the rest of the signals in Table 3-1.

***Table 3-1* • Manual Connections**

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, you will receive the message window shown in Figure 3-19.

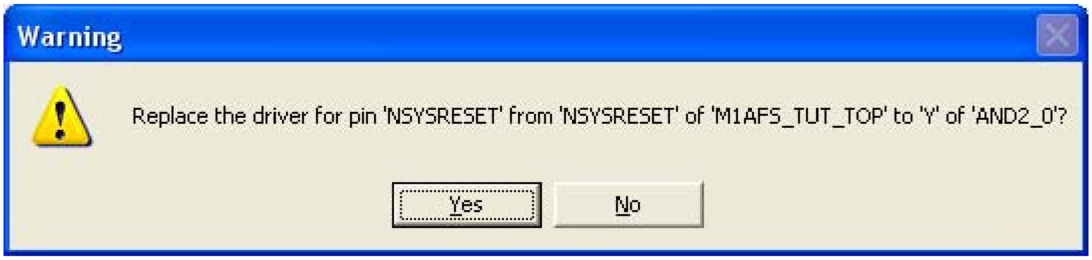

Figure 3-19 **• Replace Driver Warning**

7. Click **Yes.** You receive this message because you are changing the driver for NSYSRESET of the Cortex-M1 processor.
8. Some signals need to be connected to GND or $V_{CC.}$ Follow the steps below to connect signals according to Table 3-2 on page 3-32 using the Smart Design Grid.
9. Right- click the **dataIn[31:0]** signal from the CoreGPIO_0 instance and choose **Add Slice**.
10. Type 31 and 2 (for [31:2]), as shown in Figure 3-20.

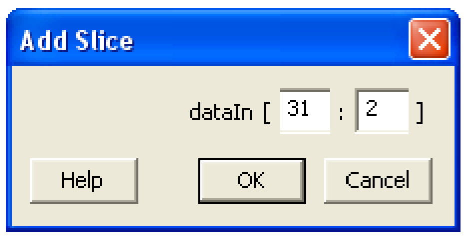

Figure 3-20 **• Add Slice**

11. Click **OK**.
12. Right-click on the [31:2] slice you just created and choose **Tie Low** because these bits are not being used in the design.
13. Left-click the cell in the Attribute column next to the EDBGRQ signal in the CortexM1Top_0 instance.
14. Choose **Tie Low**.
15. Follow steps 12-13 for the rest of the signals in Table 3-3, except remember to choose **Tie High** for the POWERDOWN signal of the PLL_sys_0 instance (as indicated in Table 3-2 on page 3-32).

Table 3-2 • **VCC and GND Connections**

Instance	Signal	Connection
CoreGPIO_0	dataIn[31:2]	GND
CortexM1Top_0	EDBGRQ	GND
	NMI	GND
	Irq[0..31]	GND
	RV_TCK	GND
	RV_TDI	GND
	RV_TMS	GND
	WDOGRES	GND
PLL_sys_0	POWERDOWN	VCC
	OADIVRST	GND
CoreAHBLite_0	REMAP_M0	GND

Switch to the Smart Design Canvas, 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 the SYSCLK top level port and choose **Disconnect**.
2. Highlight the **HCLK** input of the Cortex-M1.
3. CTRL + Click to highlight the **GLA** output of the PLL.
4. Right-click on GLA and choose **Connect**.
5. Delete the SYSCLK port, since you do not need it 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 look like Figure 3-21.

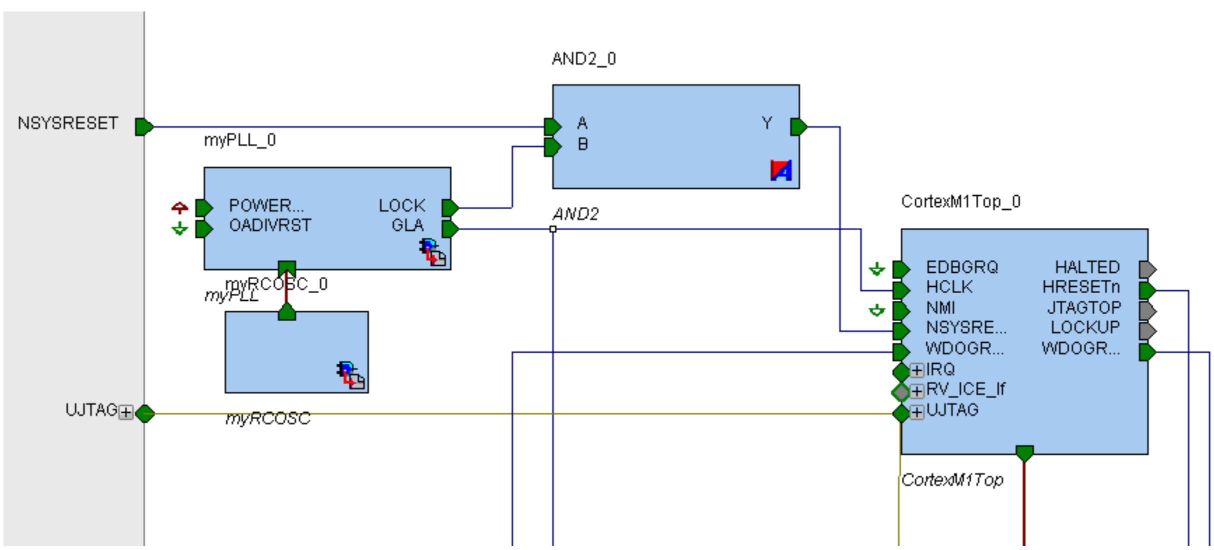

Figure 3-21 • **SmartDesign Canvas**

The naming of the groups, such as UJTAG, is not important, as the group names are not carried through during design generation.

Promote Signals to Top Level

This 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 Table 3-3 to the top level.

1. Go to the Canvas.
2. Right-click the UJTAG group of the CortexM1Top_0 instance and choose **Promote to Top Level**.
3. Right-click the dataIn[31:0] bus of CoreGPIO_0 instance and choose **Add Slice**.
4. Type 1 and 0 (for [1:0]), as shown in Figure 3-22.

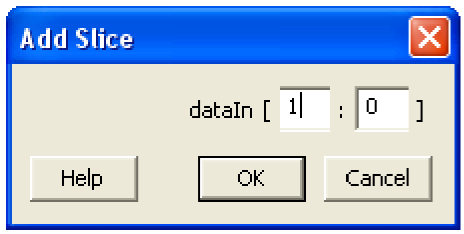

Figure 3-22 • **Add Slice Dialog Box**

5. Click **OK**.
6. Right-click on **dataIn[1:0]** and select **Promote to Top Level**.
7. Follow steps 3-6 to promote dataOut[3:0] of Core_GPIO_0 to the top.
8. Right-click and choose **Promote to Top Level** to promote the rest of the signals in Table 3-3

Table 3-3 • **Signals to Promote to Top Level**

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

Now, your Smart Design Canvas should resemble the one shown in Figure 3-23 on page 3-34.

If you are targeting the M1AFS-ADV-DEV-KIT board perform the following steps to add a second chip select signal for the off-chip SRAM device:

- Right-click to the right of the canvas and choose **Add Port**.
- Type **SRAMCS** in the Name field.
- Select **Output** for the direction.
- Click *OK*. The new SRAMCS port will be added to your canvas.
- Press CTRL and click to select SRAMCS and SRAMCSN.
- Right-click SRAMCS or SRAMCSN and choose **Connect**.
- Right-click SRAMCS only (not SRAMCSN) and choose **Invert**.

Note: Make sure SRAMCSN is not highlighted when doing this step.

You should see an inverter placed near the SRAMCS port.

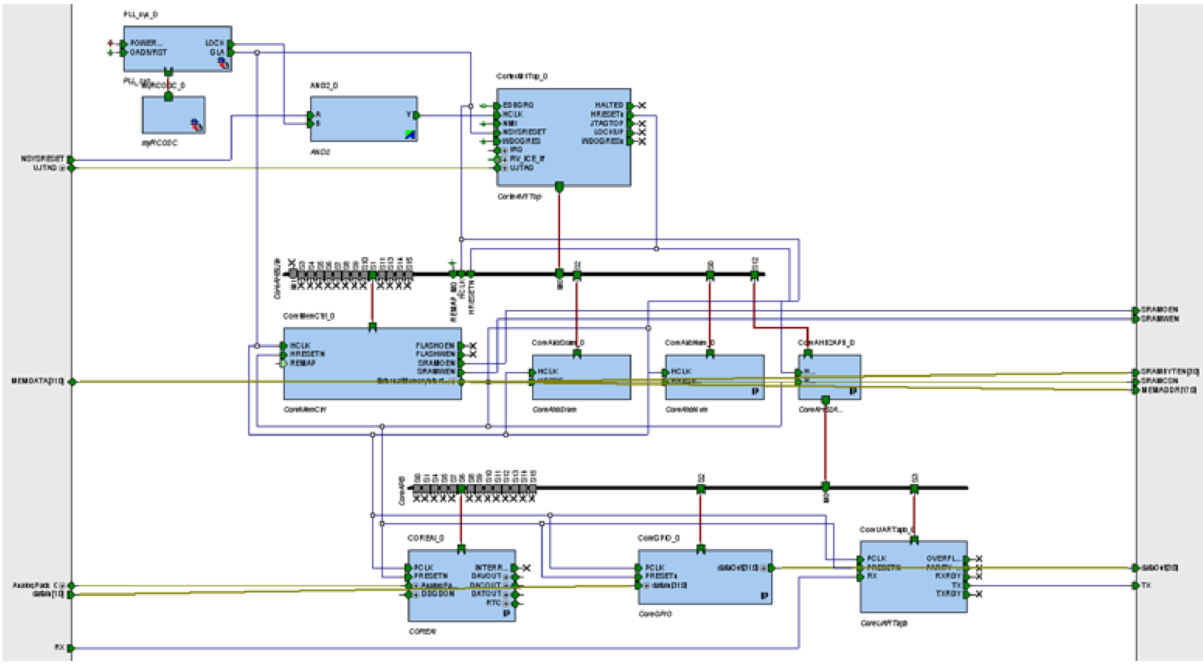

Figure 3-23 • SmartDesign Canvas

9. Go to **SmartDesign** > **Show Memory Map / Data Sheet.**

An HTML file opens which has the system memory map and register maps for the peripherals. Make sure this matches the memory map shown in Figure 3-24.

CortexM1Top_0 Subsystem

Master(s) on this bus:

- CortexM1Top_0

	Base Address	
	NoRemap	M0_SwapSlots0and1
CoreAhbNvm_0 : NVM	0x00000000	0x10000000
CoreMemCtrl_0 : FLASH	0x10000000	0x00000000
CoreAhbSram_0 : IRAM	0x20000000	
CoreGPIO_0 : RegisterMap	0xc2000000	
CoreUARTapb_0 : RegisterMap	0xc3000000	
COREAI_0 : RegisterMap	0xc6000000	

subsystem list, top of page

Figure 3-24 • **Memory Map**

Note: The memory map for CoreMemCtrl_0 is not complete. The SRAM interface for CoreMemCtrl_0 is at address 0x18000000.

Save and Generate the SmartDesign System

The final step within SmartDesign is saving and generating the HDL. Follow the steps below to generate the HDL for your design.

1. Right-click on the Canvas and choose **Generate Design** to generate the HDL design.
2. Click **OK** in the Information window after generation is complete (you receive warnings).

   Prior to generating HDL code, SmartDesign checks the connectivity of the signals in the design and produces errors or warnings as appropriate. You should not receive any errors. However, you will receive warnings concerning some unconnected outputs. You can safely ignore these. If you want to see which signals the warnings are concerned with, you can go to the SmartDesign menu and choose **Check Design Rules.** This will launch a read-only Grid-like window which will show you the signals related to the warnings.

   If you want to get rid of the warnings, you can mark the floating drivers and unconnected bus interfaces as unused. You will need to create slices for busses which have some bits connected and some floating. Then, you can go to the Design Rules Check grid, highlight all of the unconnected and floating drivers, right-click under the Port Name column, and choose **Mark as Unused**.

3. Close the SmartDesign by clicking on the small **x** in the upper right corner as in Figure 3-25.

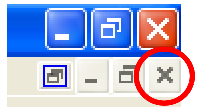

Figure 3-25 **•** Close SmartDesign

Besides generating the HDL design, a testbench file is created along with all the necessary files to run a simulation, using the bus functional model (BFM) and AMBA transfers.

Step 3 – Create Flash Memory System

You are creating this Flash Memory System and Data Storage client to 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. Go to the Catalog and expand the Fusion Peripherals category.
2. Double-click **Flash Memory System Builder**. You will see the Flash Memory System window shown in Figure 3-26.

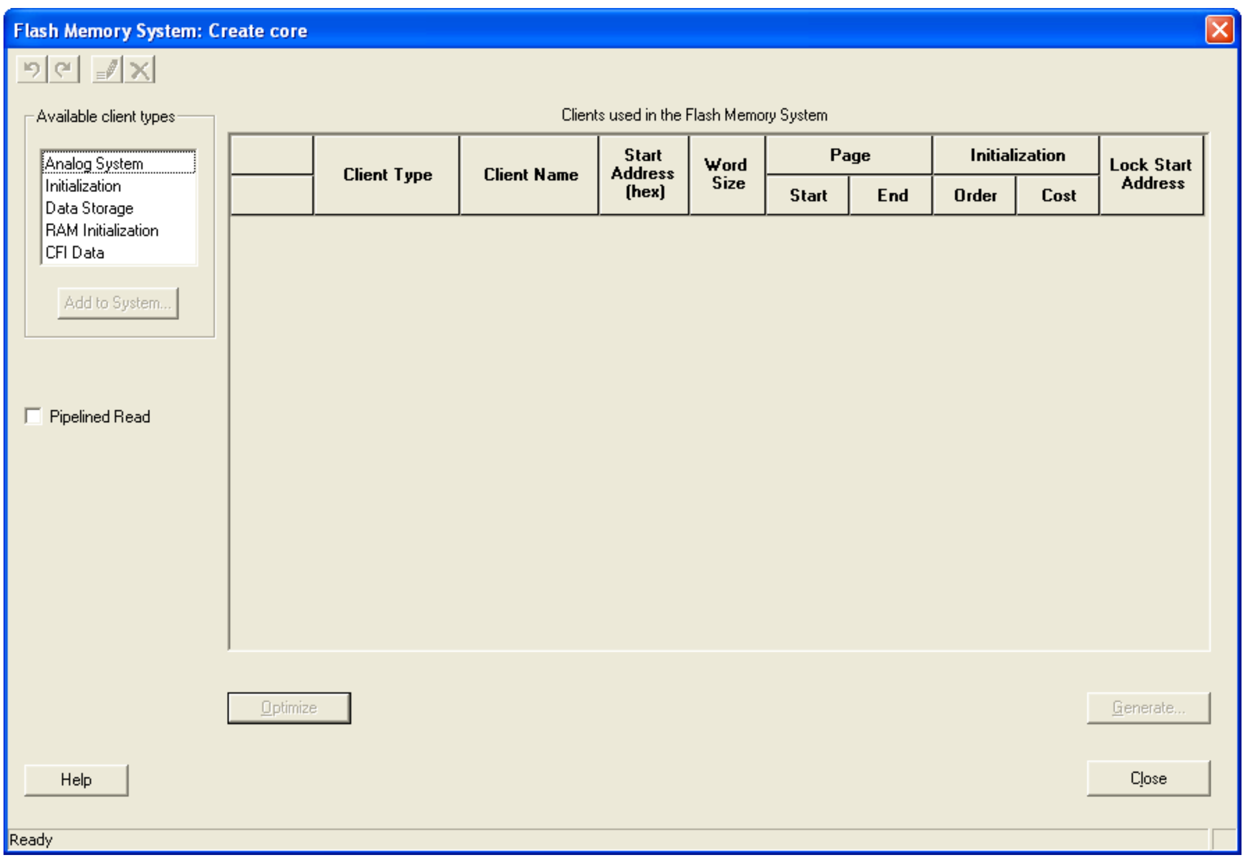

Figure 3-26 **• Flash Memory System Window**

3. Double-click on **Data Storage Client** in the Available client types list. You will see the Add Data Storage Client window shown in Figure 3-27.

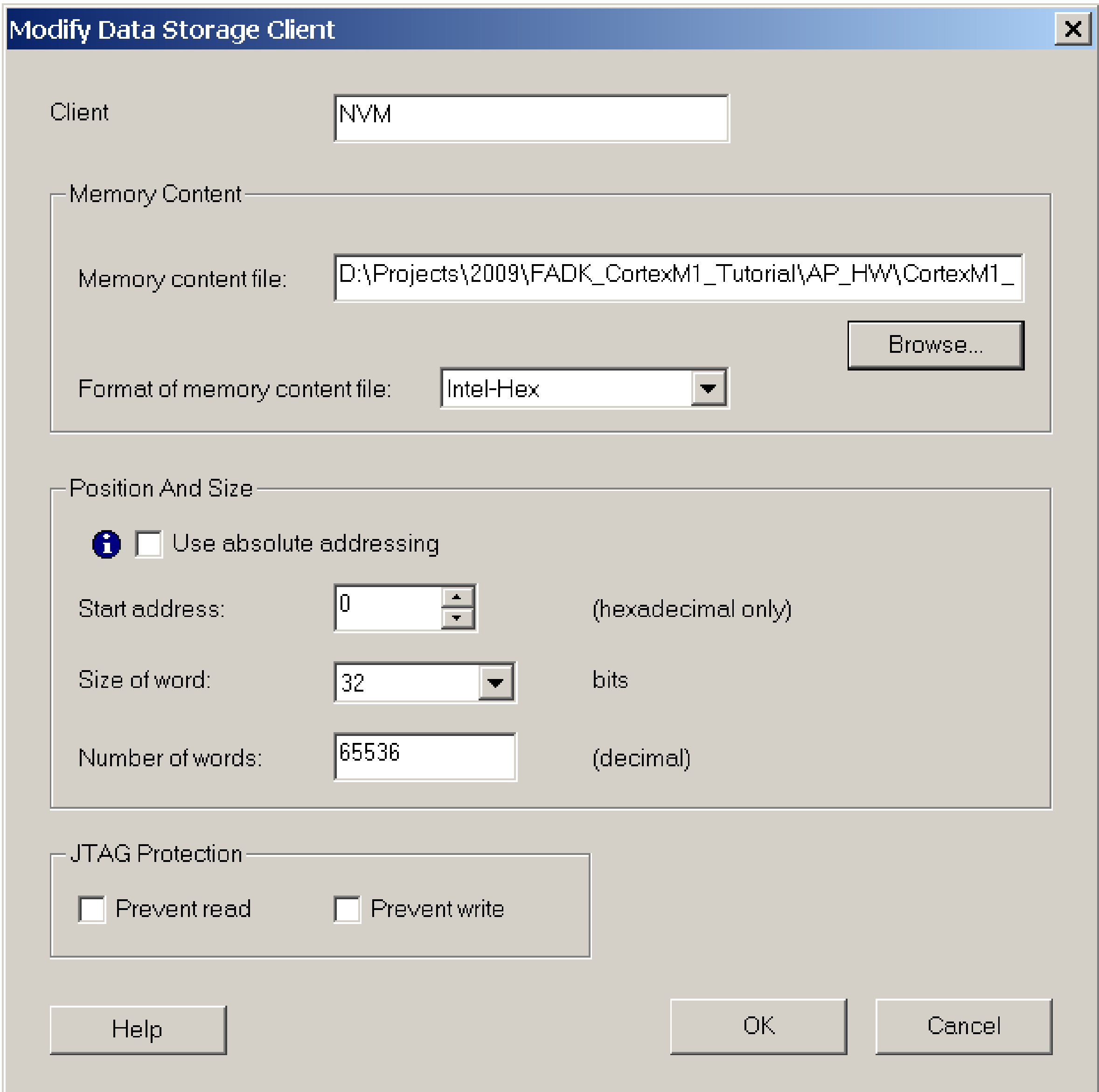

Figure 3-27 **• Add Data Storage Client**

4. Enter NVM in the Client name text box.
5. Leave the Start address at 0.
6. Change the Size of word to 32 bits.
7. Enter the 65536 for the Number of words.
8. Click the **Browse** button and select the *CortexM1_Tutorial.hex* file from the *CortexM1_Fusion_HW_Tutorial\<board>\Tutorial_Files* directory.
9. Make sure the Format of memory content file is Intel-Hex.

10. Click **OK**. Now the Data Storage client you just created appears in the Flash Memory System window (Figure 3-28).

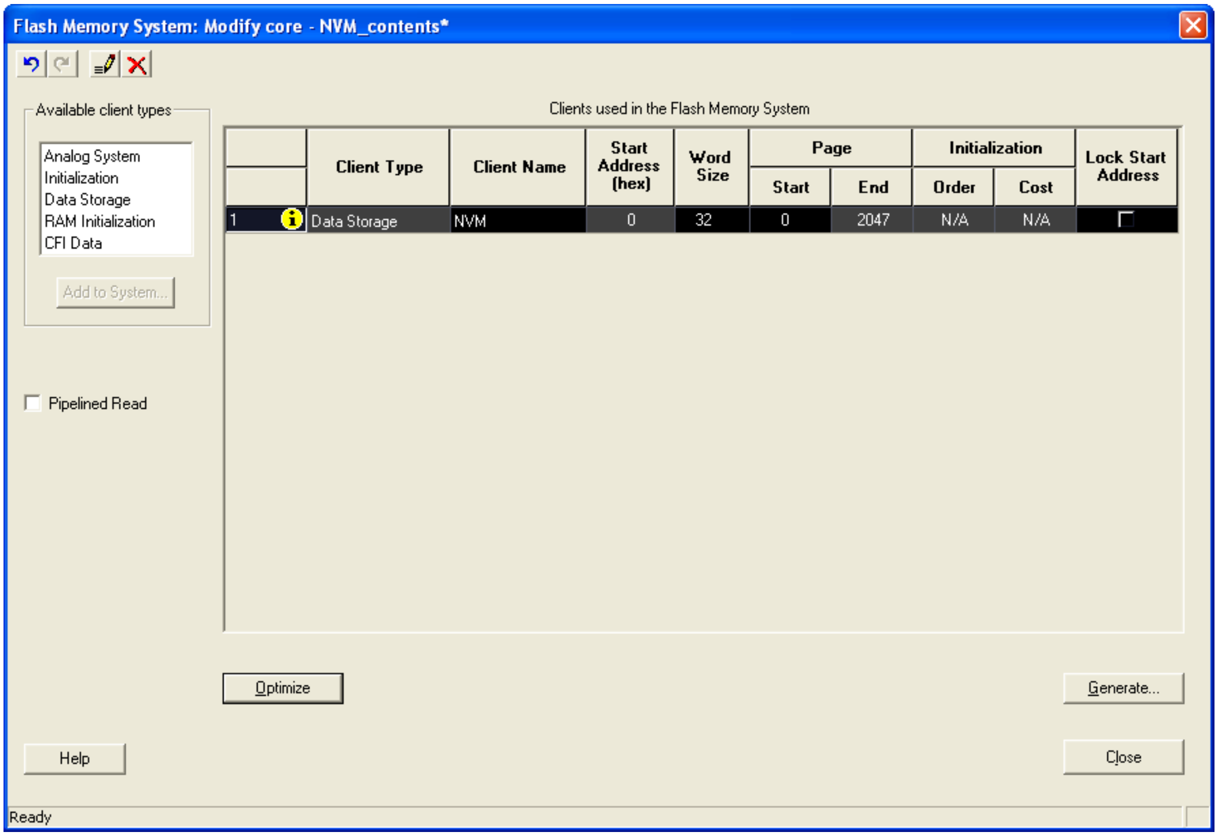

Figure 3-28 • Data Storage Client in Flash Memory System Window

11. Click **Generate** to generate the component. You will see the Generate Core window.
12. Enter NVM_contents in the Core name box.
13. Click **OK**.

4 – Tutorial – Simulation, Synthesis, and Place-and-Route

Step 4 – Perform Pre-Synthesis Functional Simulation

In this section you will run a pre-synthesis functional simulation to exercise the Bus Functional Model (BFM) to verify AMBA transactions with the peripherals in your system. This step is not necessary, since the Actel cores have already been verified. However, running the BFM is useful when you want to verify the AMBA interface of peripherals you have created.

1. Go to **Project** > **Settings**.
2. Click on the Simulation tab.
3. Change the Value for Simulation runtime to **-all**. Make sure your simulation settings look like those in Figure 4-1.

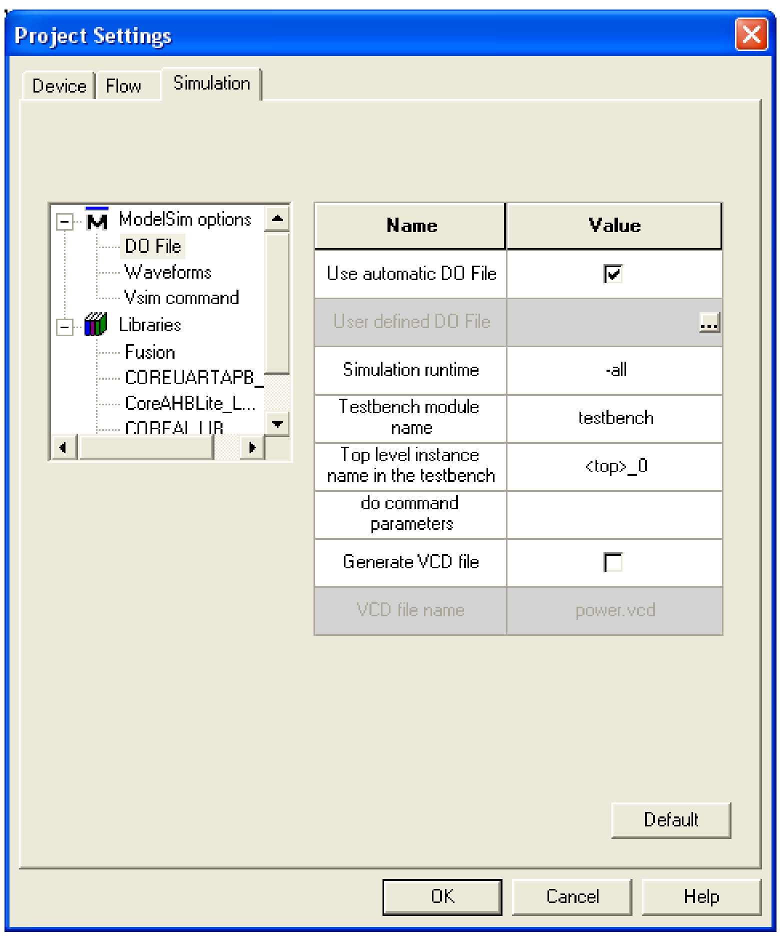

Figure 4-1 • **Simulation Settings**

4. Click **OK**
5. Click on the **ModelSim** icon.

ModelSim is launched and the necessary source files are compiled, including the testbench and peripheral components. At the conclusion of the compilation, the Bus Functional Model (BFM) scripts will be executed to exercise and test the AMBA-based peripherals.

A sequence of reads and writes will occur to the peripherals and the results displayed in the ModelSim log window, as shown in Figure 4-2.

```
rm/coreuartapb_0
# --------- Execution of BFM Script Started ---------
# N-cycle: Write AABBCCDD to address 10000010
# N-cycle: Read ZZZZZZZZ from address 10000010
# N-cycle: Write AABBCCDD to address 10000010
# N-cycle: Read ZZZZZZZZ from address 10000010
# N-cycle: Write 12345678 to address 20000000
# N-cycle: Read 12345678 from address 20000000
# N-cycle: Write 9A to address 20000000
# N-cycle: Read 1234569A from address 20000000
# N-cycle: Read 9A from address 20000000
# N-cycle: Read 56 from address 20000001
# N-cycle: Read 34 from address 20000002
# N-cycle: Read 12 from address 20000003
# N-cycle: Read 569A from address 20000000
# N-cycle: Read 1234 from address 20000002
# N-cycle: Write BC to address 20000001
# N-cycle: Write DE to address 20000002
# N-cycle: Write F0 to address 20000003
# N-cycle: Read F0DEBC9A from address 20000000
# N-cycle: Write AABBCCDD to address 20000004
# N-cycle: Read AABB from address 20000006
# N-cycle: Write 9876 to address C0000000
# N-cycle: Read 9876 from address C0000004
# N-cycle: Read 9876 from address C0000000
# N-cycle: Write AB to address C1000000
# N-cycle: Write FF to address C1000004
# N-cycle: Write 87654321 to address C1000018
# N-cycle: Write FFFFFFFF to address C100001C
# N-cycle: Read 00 from address C1000000
# N-cycle: Read 00 from address C1000004
# N-cycle: Read 00000000 from address C1000018
# N-cycle: Read 00000000 from address C100001C
# N-cycle: Write 99 to address C3000074
# N-cycle: Read 00 from address C3000080
# N-cycle: Write 9876 to address C4000000
# N-cycle: Read 9876 from address C4000004
# N-cycle: Read 9876 from address C4000000
# N-cycle: Write ABCD to address CC000000
# N-cycle: Read ABCD from address CC000004
# N-cycle: Read ABCD from address CC000000
# N-cycle: Write 01 to address CF000000
# N-cycle: Read 01 from address CF000000
# N-cycle: Read 01 from address CF000000
# N-cycle: Write 00 to address CF000000
# N-cycle: Read 00 from address CF000000
# --------- Successful Execution of BFM Script Complete ---------
# ** Failure: Breakpoint encountered - normal completion of BFM-driven simulation
#    Time: 4926495 ns  Iteration: 1  Process: /testbench/m1a3psocdk_platform_uitag_0/u
```

Figure 4-2 • **ModelSim Log Window**

If you receive some warnings, you can ignore them.

For more information about the BFM, refer to the Cortex-M1 Handbook on the Actel website:

http://www.actel.com/documents/CortexM1_HB.pdf.

Step 5 – Perform Synthesis

After the functional simulation has completed, synthesis can be performed. Perform the steps below to synthesize the design.

1. Launch the Synplicity® Synplify® tool by clicking on the **Synthesis** button in the Libero IDE Project Flow window. Synplify will launch and load your project. If you see the Organize constraints for Synthesis window, click **OK**.
2. Click on the **Run** button in Synplify to synthesize the design.

   Once synthesis is complete you will see many warnings, which are safe to ignore. However, if any errors are present, those need to be corrected prior to continuing. A screenshot of the Synplify tool is shown in Figure 4-3.

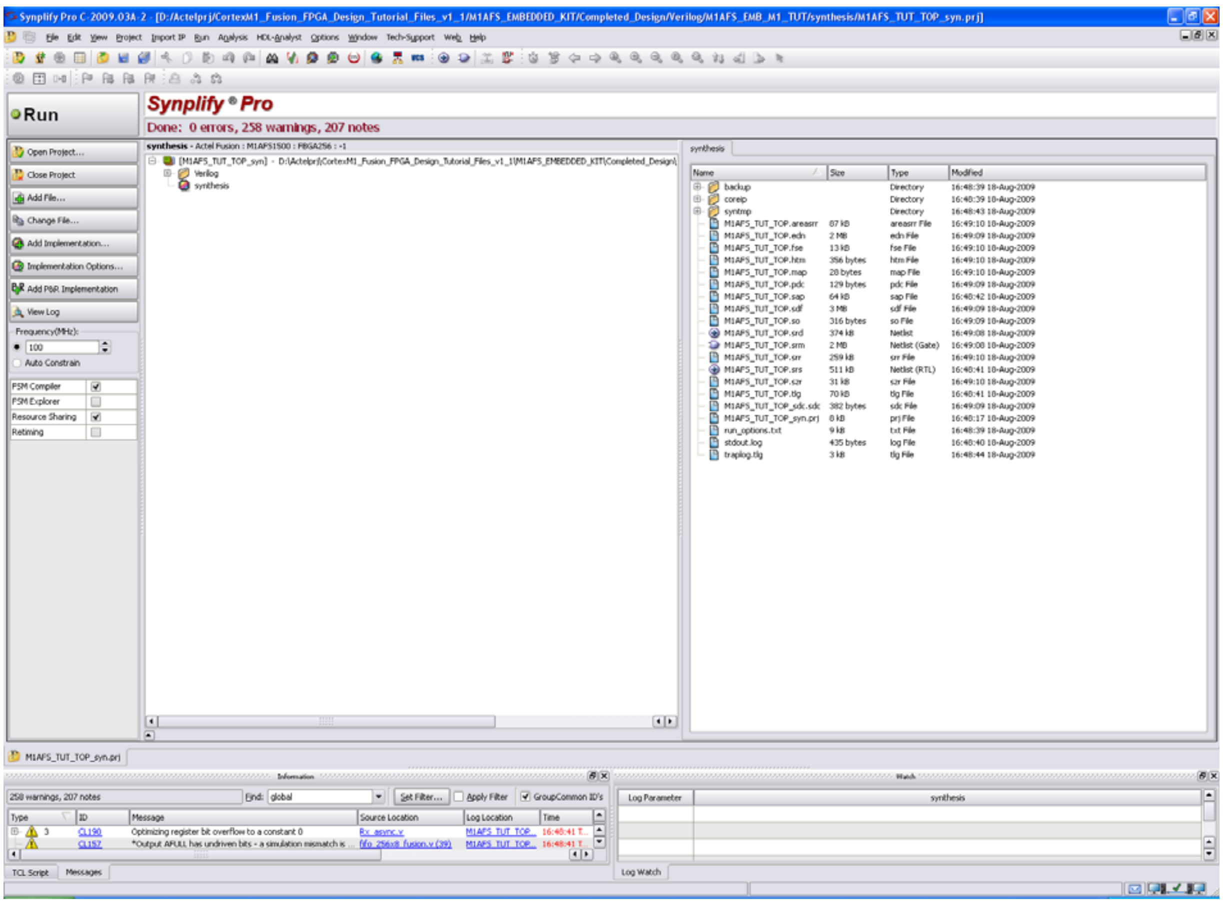

Figure 4-3 • **Synplify Pro**

Note that if you are using Synplify (instead of Synplify Pro) your screen will look different. Close Synplify before continuing.

Step 6 – Perform Place-and-Route

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

1. Open Designer by clicking on the **Place & Route** icon on the Libero IDE Design Flow manager.
2. The Organize Constraint dialog box comes up, as shown in Figure 4-4.

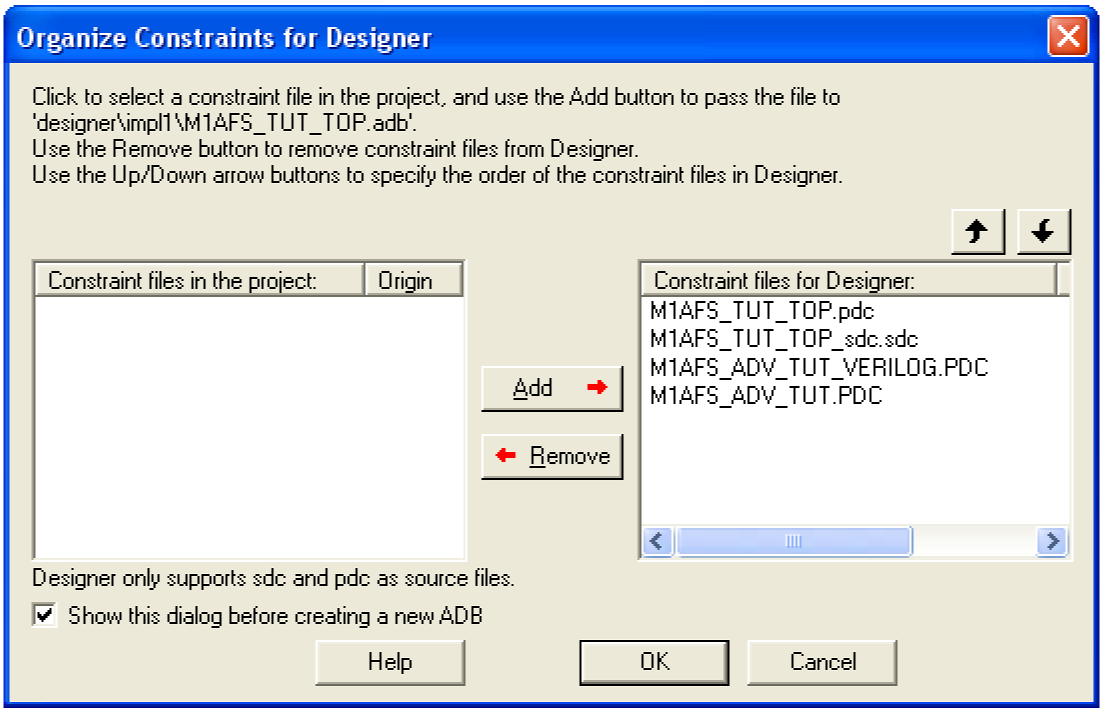

Figure 4-4 • **Organize Constraints for Designer Window**

3. We do not need the first two files created automatically by SmartDesign and Synplify. Highlight the files not needed and click **Remove**. We will need only the two PDF files we have imported. The constraints organizer should look like Figure 4-5. Click **OK**.

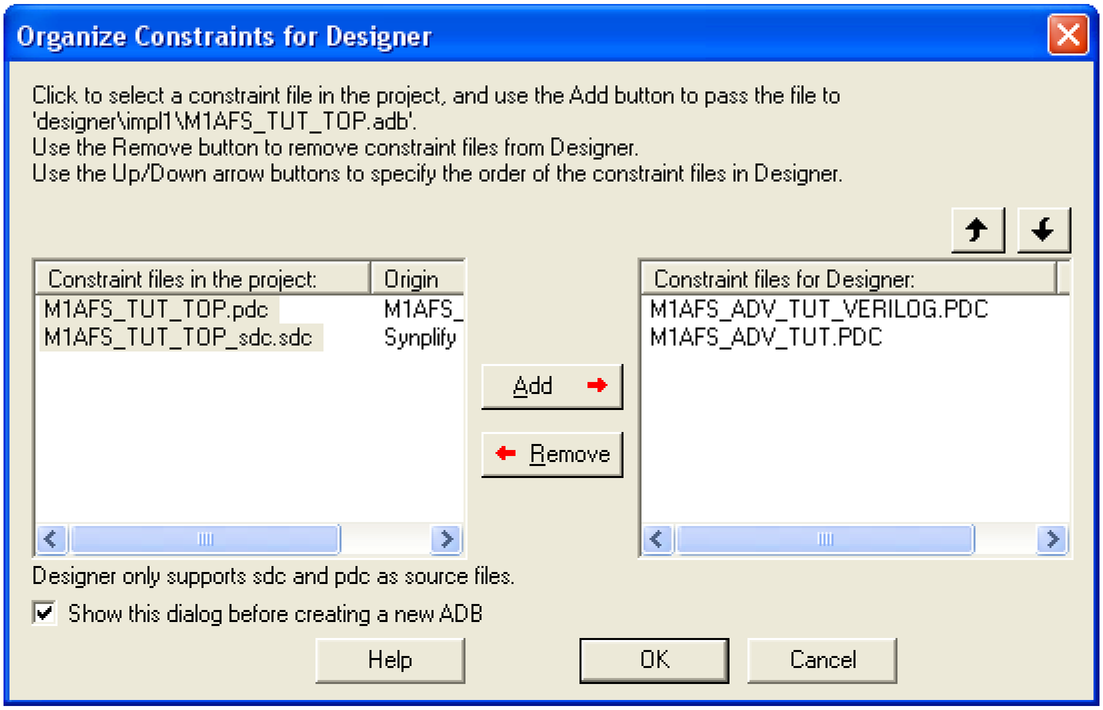

Figure 4-5 • **Constraints Organizer Window**

4. When the device selection wizard comes up (Figure 4-6), select **Next** twice and then **Finish** to set the default selection.

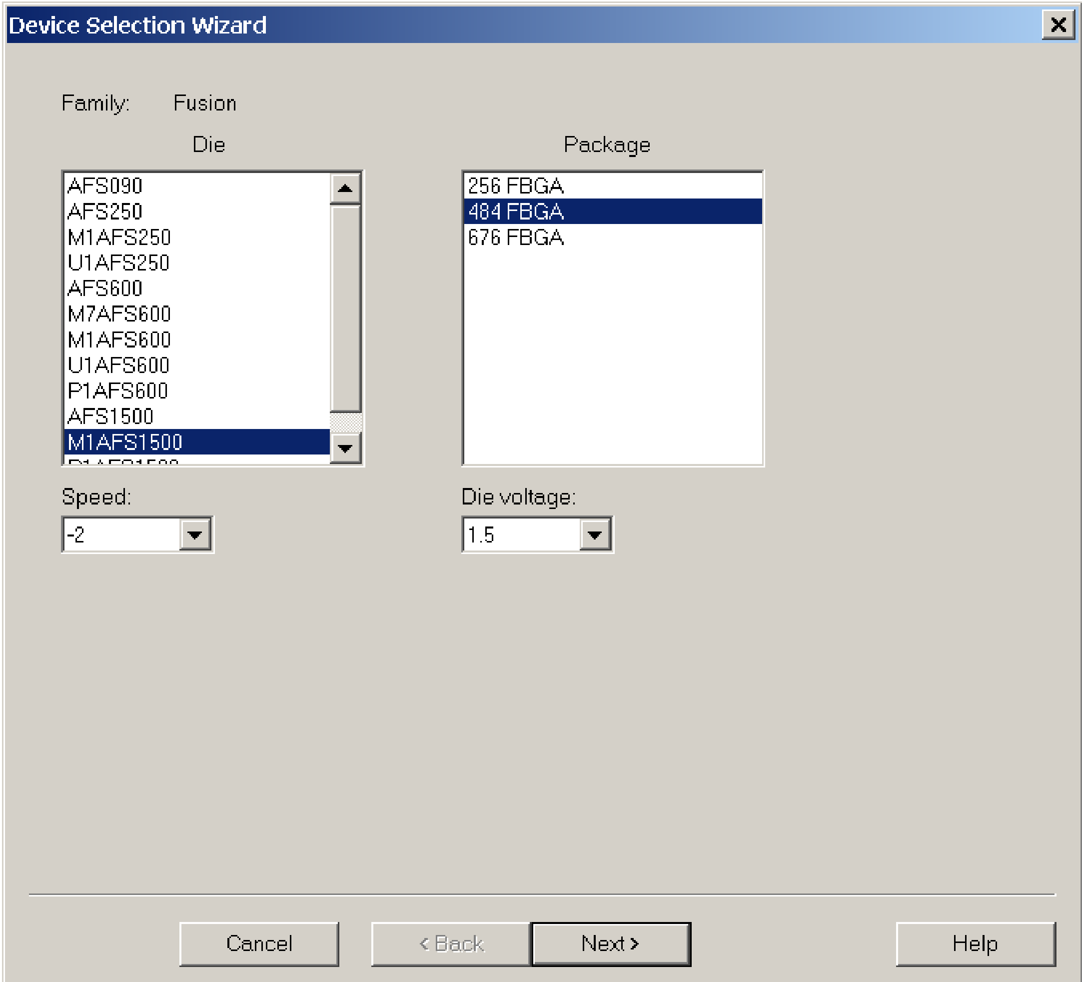

Figure 4-6 • **Device Selection Wizard**

5. Click **Compile** and **OK** with default setting to compile the design.
6. Click the **Layout** icon in Designer and accept all default settings by clicking **OK** in the windows that appear. First Compile and then Layout will be performed. Once the Layout has finished, you can proceed to the next step. This will take longer than the compilation of the previous step.
7. Click the **Timing Analyzer** icon to analyze the timing of the design. Note that 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, you will see a red X next to the clock in the window in the left upper corner. Make sure that none of the clocks have a red X next to them. Close the Timing Analyzer window.

5 – Tutorial – Programming

Step 7 – Generate Programming File with Software Code in NVM

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

Click the Programming File icon in the Designer window. You will see a window similar to the one in Figure 5-1.

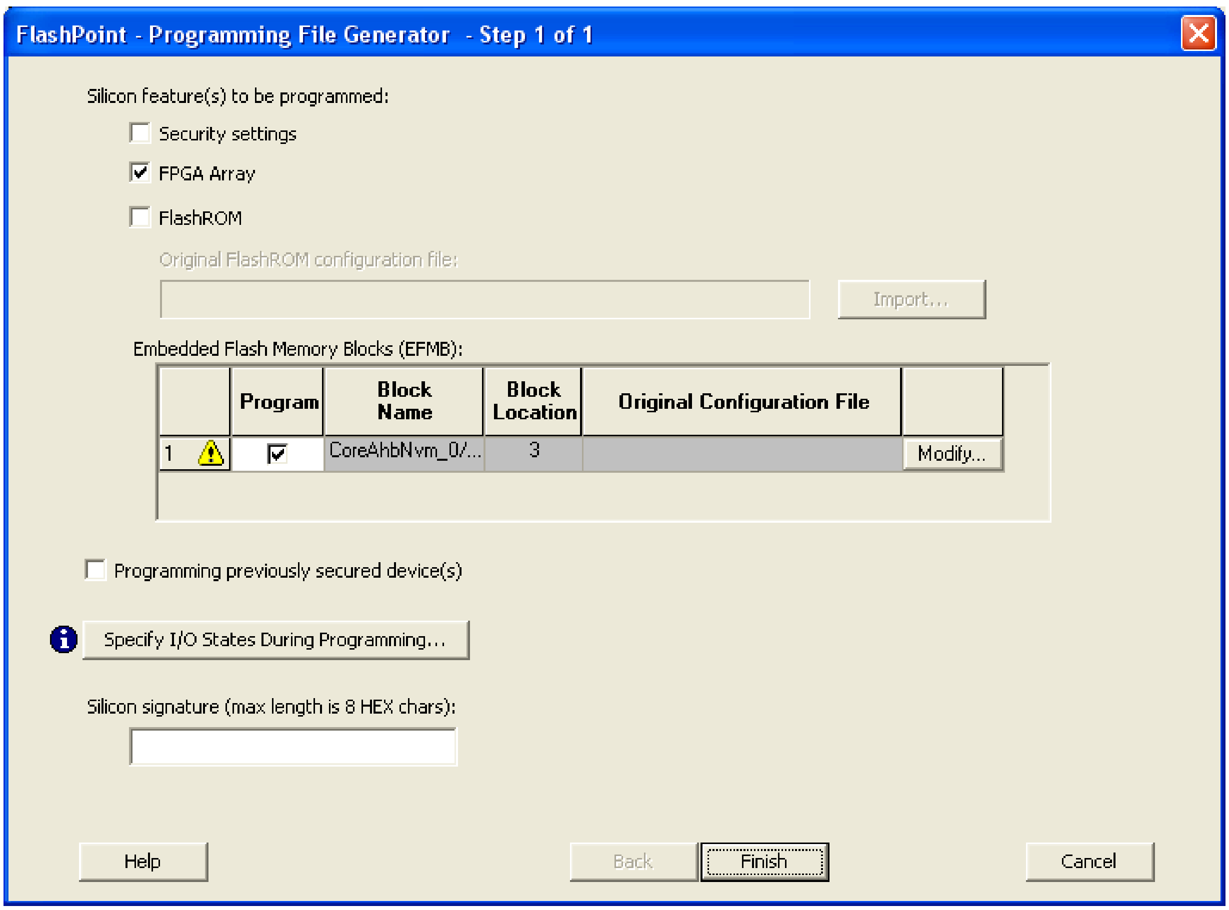

Figure 5-1 • **FlashPoint Programming File Generator**

6. Click the **Modify** button. The Modify Embedded Flash Memory Block window appears.
7. Click the **Import Configuration File** button. The Import window appears.
8. Browse to <project directory>/smartgen/NVM_contents/NVM_contents.efc.
9. Click **Import**.
10. Click **OK**.
11. Click **Finish**. The Generate Programming Files window appears.

12. Make sure that Programming Data File (*.pdb) is checked.
13. Click **Generate**.

    Once the Programming File icon in Designer turns 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 Actel Designer desktop is shown in Figure 5-2.
14. Close Designer. Click **Yes** if prompted to save changes to M1AFS_TUT_TOP.adb.

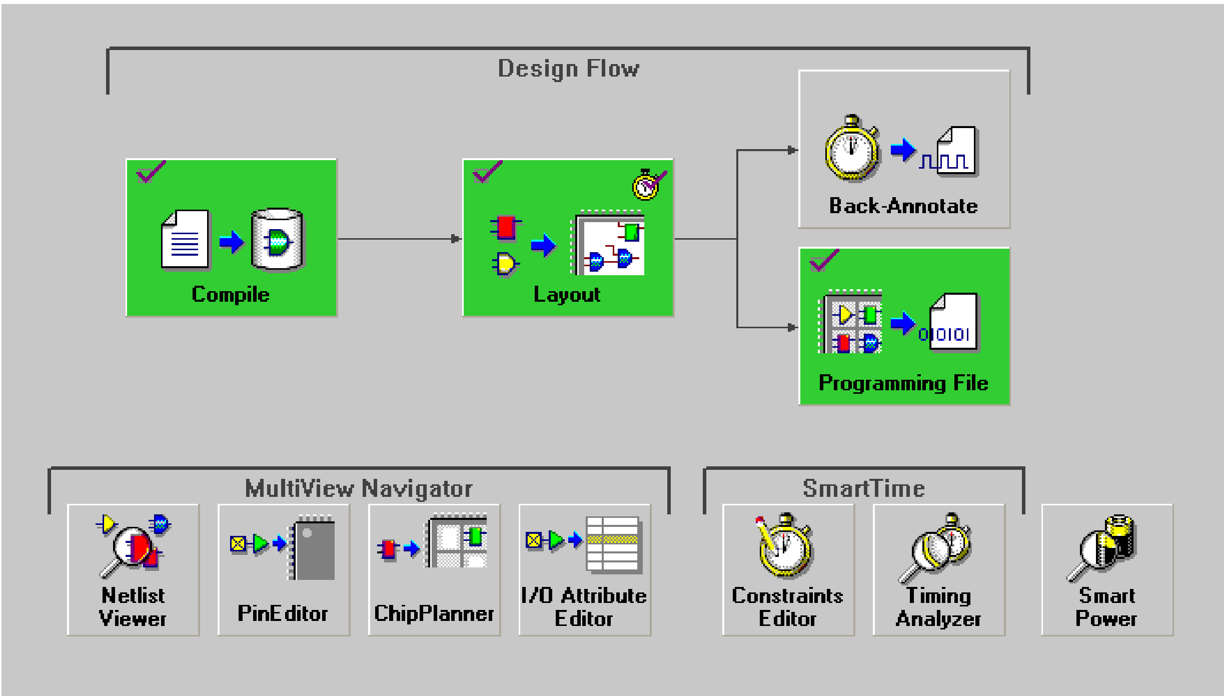

Figure 5-2 • **Actel Designer Desktop**

Step 8 – Connect to the Target

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

1. Open the HyperTerminal application (**Start** > **Programs** > **Accessories** > **Communications** > **HyperTerminal)**. Enter M1AFS_Tutorial on the Name field in the Connection Description dialog box and click **OK**.
2. Select the COM port you identified in the Getting Started Section of this tutorial and click **OK**.
3. Enter the following for the properties (Figure 5-3):
   - Bits per second:57600
   - Data bits: 8
   - Parity: None
   - Stop bits: 1
   - Flow control: None

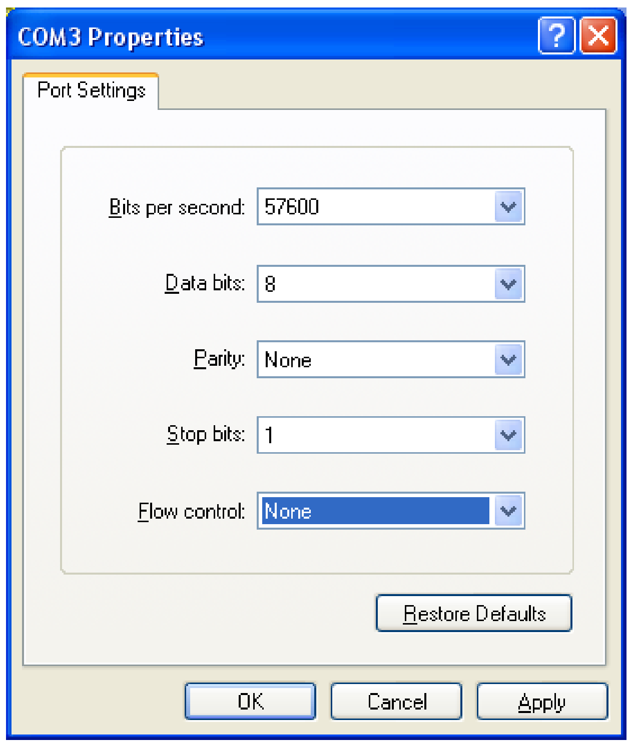

Figure 5-3 • **COM3 Properties**

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

Step 9 – Program the M1AFS1500 FPGA

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

1. Confirm that the FlashPro3 programmer is connected as described in Table 2-3 on page 2-11.
2. Important: Do not interrupt power during programming since damage to the device might occur.
3. Click on the Programming (FlashPro) icon in the Project Flow tab in Libero IDE Project Manager.

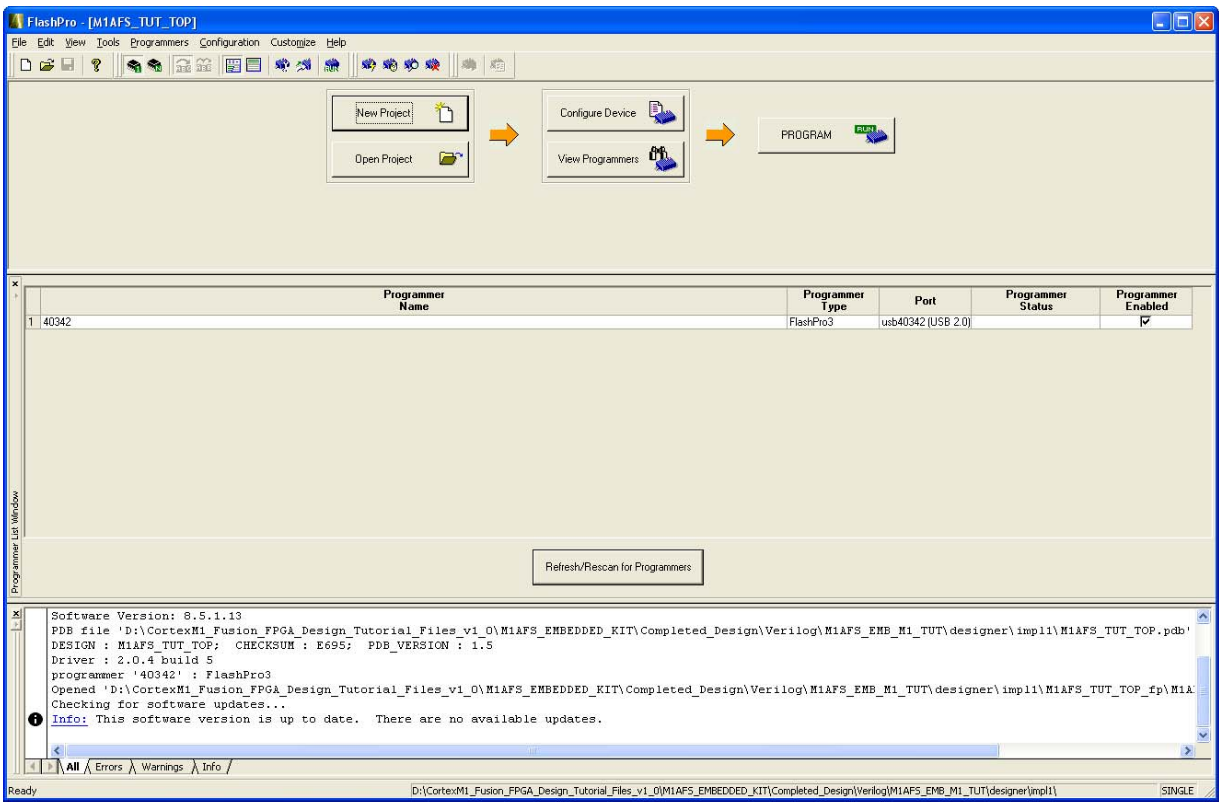

Figure 5-4 • **FlashPro Project Flow**

Note: If you receive messages about new hardware detected on your machine, complete the USB driver installation as described in the "Getting Started" section of the FlashPro User's Guide (www.actel.com/products/hardware/program_debug/flashpro/default.aspx#docs).

4. Click the PROGRAM button to program the Fusion device (Figure 5-4). Once programming is complete, the Programmer Status will show **RUN PASSED** in green.

5. Press SW1 on the board to reset the system (Figure 5-5 and Figure 5-6 on page 5-50).

Figure 5-5 • **Fusion Embedded Development Kit Reset System with SW1**

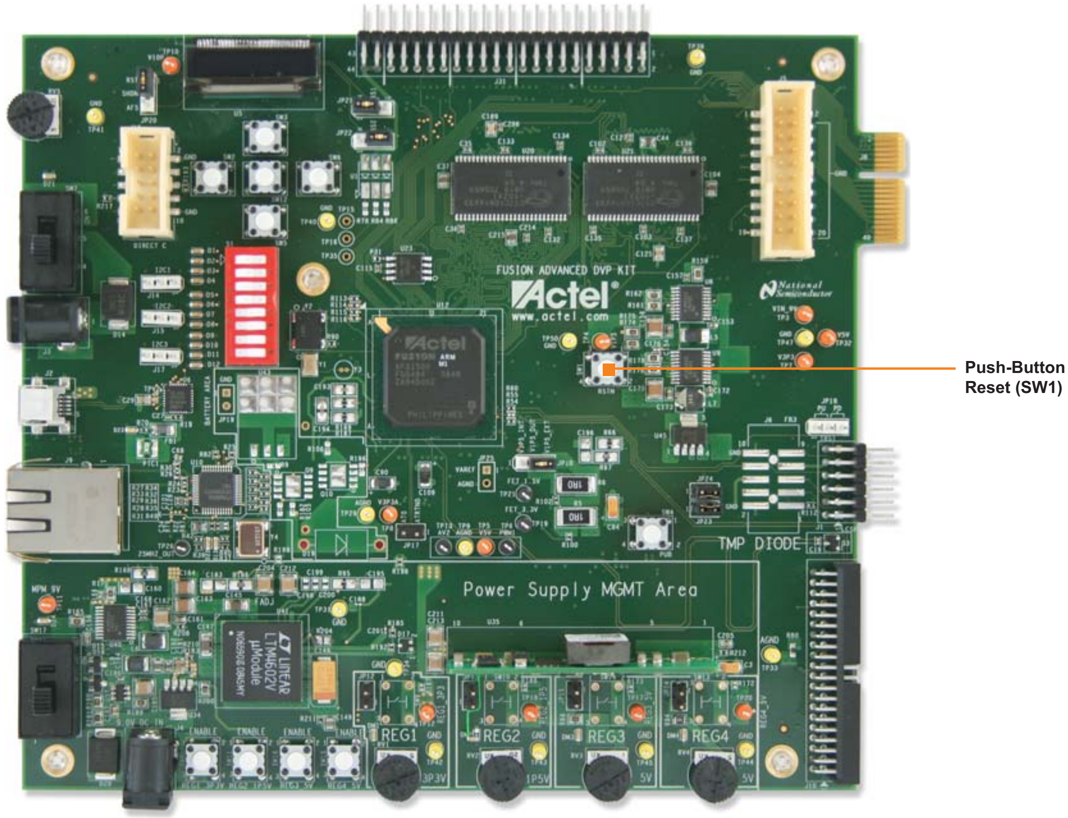

Figure 5-6 • **Fusion Advanced Development Kit Board Reset System with SW1**

6. After pressing the reset button (SW1), you should see some of the LEDs counting.

7. Go to the HyperTerminal window. You should see the message shown in Figure 5-7 in the HyperTerminal window.

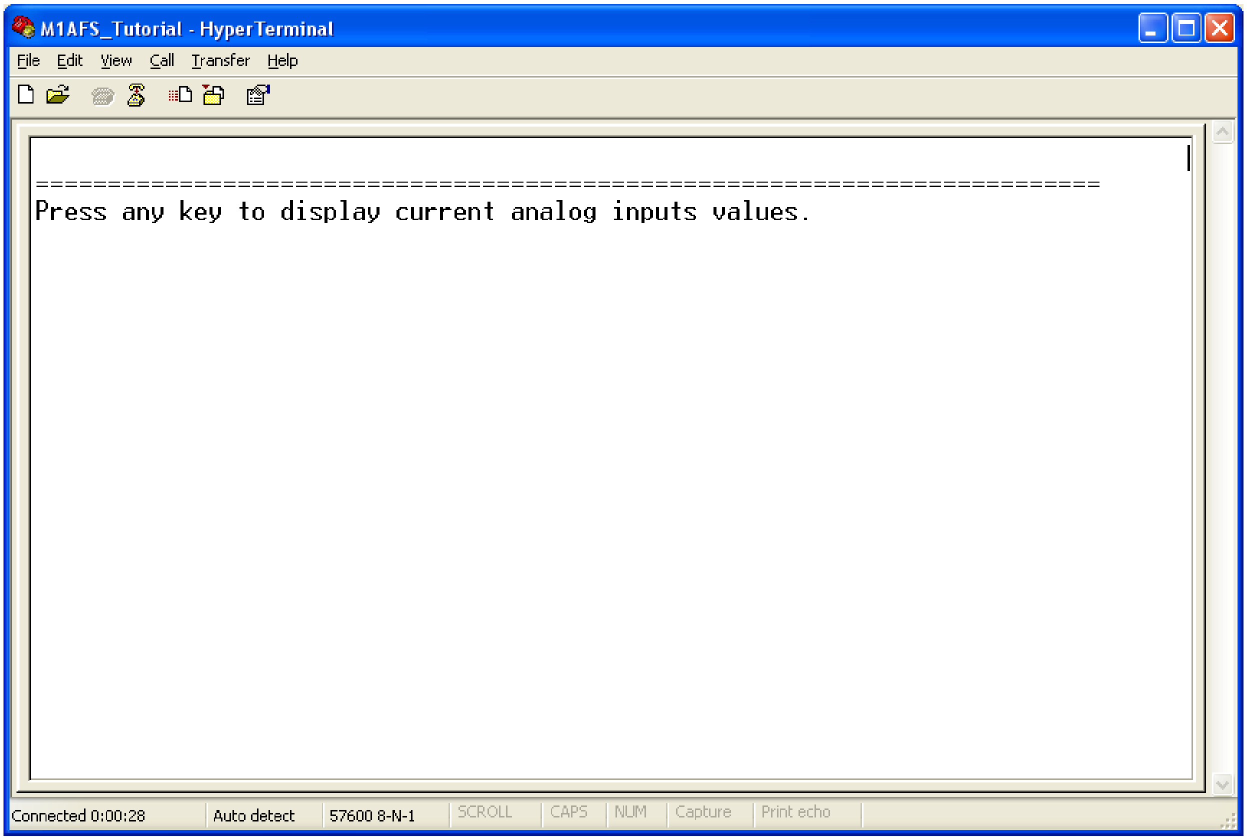

Figure 5-7 • **HyperTerminal Window**

8. Press any key. You should see something like the window shown in Figure 5-8.

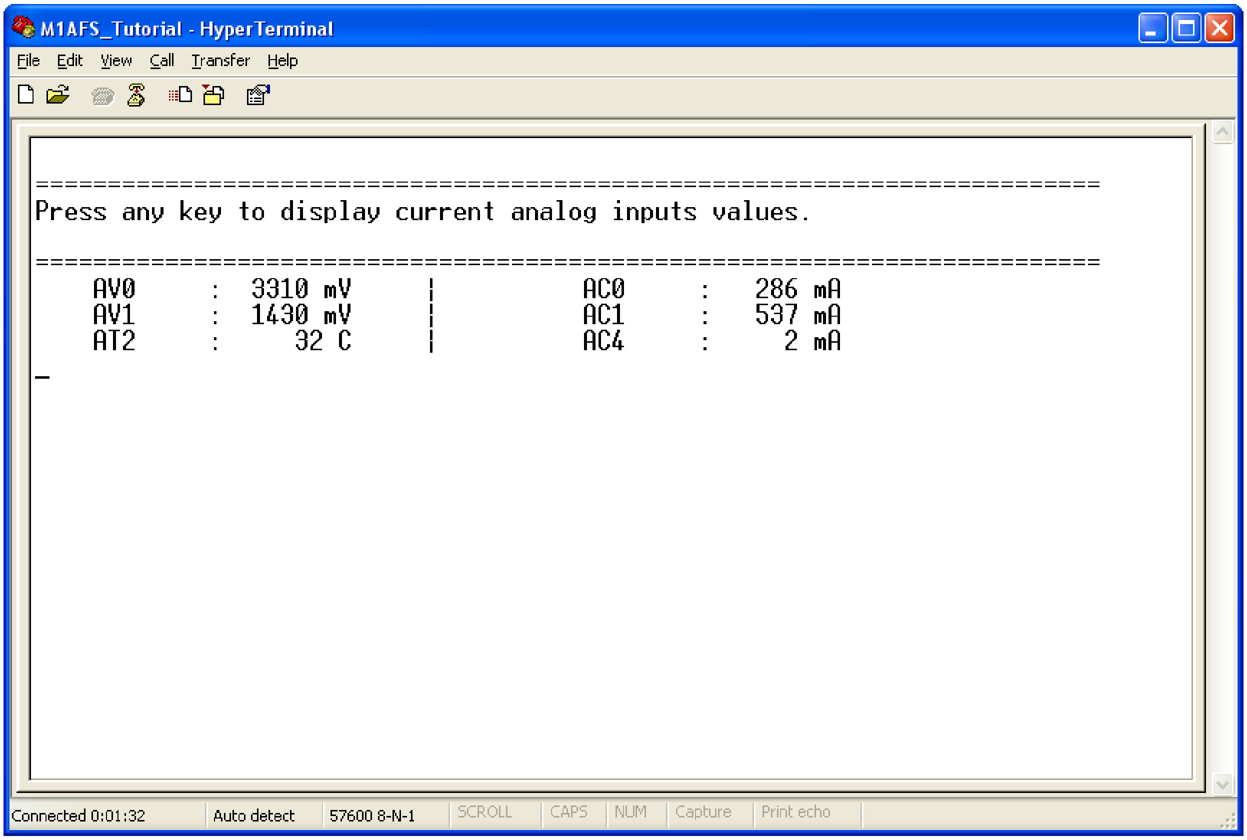

Figure 5-8 • **HyperTerminal Display**

9. Change the potentiometer on the development board (upper left corner on M1AFS-EMBEDDED-KIT and near power supply for M1AFS-ADV-DEV-KIT)).
10. Press any key. The AC4 value should change.
11. 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 are ready to develop software for your Cortex-M1 design. If you want to learn how to get started developing software for the Cortex-M1, refer to the *ARM Cortex-M1 Embedded Processor Software Development Tutorial for Fusion Mixed-Signal FPGAs* (www.actel.com/documents/CortexM1_Proc_SW_Tutorial_UG.pdf).

6 – List of Document Changes

The following table lists critical changes that were made in the current version of the document.

Previous Version	Changes in Current Version (50200162-1)	Page
50200162-1 (March 2009)	The "Requirements for this Tutorial" section was revised. FlashPro v8.6 or newer is required. Several of the versions were updated in the "Intellectual Property (IP)" section.	5
	The "Before You Get Started" section was revised to change the names of the files and file hierarchy.	9
	The "Tutorial – Create Design" section was revised to match the new file names and screen shots were updated as appropriate.	13
	The "Instantiate CoreAI" section was significantly changed.	22
	The "Instantiate CoreUARTapb" section was updated, including EQ 3-1.	26
	Table 3-2 • VCC and GND Connections was revised.	32
	Figure 4-3 • Synplify Pro was replaced.	41
	"Step 6 – Perform Place-and-Route" was significantly revised.	42
	The "Save and Generate the SmartDesign System" section was updated with additional information.	35
50200162-0 (March 2009)	The "Download Tutorial Files" section was revised to guard against problems created by a directory name that is too long.	9

A – Product Support

Actel backs its products with various support services including Customer Service, a Customer Technical Support Center, a web site, an FTP site, electronic mail, and worldwide sales offices. This appendix contains information about contacting Actel and using these support services.

Customer Service

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

From Northeast and North Central U.S.A., call **650.318.4480**
From Southeast and Southwest U.S.A., call **650. 318.4480**
From South Central U.S.A., call **650.318.4434**
From Northwest U.S.A., call **650.318.4434**
From Canada, call **650.318.4480**
From Europe, call **650.318.4252** or **+44 (0) 1276 401 500**
From Japan, call **650.318.4743**
From the rest of the world, call **650.318.4743**
Fax, from anywhere in the world **650.318.8044**

Actel Customer Technical Support Center

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

Actel Technical Support

Visit the Actel Customer Support website (www.actel.com/support/search/default.aspx) for more information and support. Many answers available on the searchable web resource include diagrams, illustrations, and links to other resources on the Actel web site.

Website

You can browse a variety of technical and non-technical information on Actel's home page, at www.actel.com.

Contacting the Customer Technical Support Center

Highly skilled engineers staff the Technical Support Center from 7:00 a.m. to 6:00 p.m., Pacific Time, Monday through Friday. Several ways of contacting the Center follow:

Email

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

The technical support email address is tech@actel.com.

Phone

Our Technical Support Center answers all calls. The center retrieves information, such as your name, company name, phone number and your question, and then issues a case number. The Center then forwards the information to a queue where the first available application engineer receives the data and returns your call. The phone hours are from 7:00 a.m. to 6:00 p.m., Pacific Time, Monday through Friday. The Technical Support numbers are:

650.318.4460
800.262.1060

Customers needing assistance outside the US time zones can either contact technical support via email (tech@actel.com) or contact a local sales office. Sales office listings can be found at www.actel.com/company/contact/default.aspx.

Index

T

W

POWER MATTERS

Actel is the leader in low-power FPGAs and mixed-signal FPGAs and offers the most comprehensive portfolio of system and power management solutions. Power Matters. Learn more at www.actel.com.

Actel Corporation
2061 Stierlin Court
Mountain View, CA
94043-4655 USA
Phone 650.318.4200
Fax 650.318.4600

Actel Europe Ltd.
River Court,Meadows Business Park
Station Approach, Blackwater
Camberley Surrey GU17 9AB
United Kingdom
Phone +44 (0) 1276 609 300
Fax +44 (0) 1276 607 540

Actel Japan
EXOS Ebisu Buillding 4F
1-24-14 Ebisu Shibuya-ku
Tokyo 150 Japan
Phone +81.03.3445.7671
Fax +81.03.3445.7668
http://jp.actel.com

Actel Hong Kong
Room 2107, China Resources Building
26 Harbour Road
Wanchai, Hong Kong
Phone +852 2185 6460
Fax +852 2185 6488
www.actel.com.cn

50200154-2/11.09