



XAPP1016 (v1.0) September 13, 2007

Getting Started with the Nucleus PLUS RTOS and EDGE Tools on the MicroBlaze Processor

Author: Mounir Maaref

Abstract

This application note provides an introduction to Nucleus RTOS on the MicroBlaze™ processor using Xilinx Platform Studio (XPS) tools and Mentor Graphics EDGE tools.

This document is a tutorial for building MicroBlaze hardware to run the Nucleus Real Time Operating System, for configuring the BSP (Board Support Package) within XPS (Xilinx Platform Studio), and for using EDGE features, such as the application debug.

The target board for this application note is the Xilinx Spartan™-3E Starter board.

Included Systems

Included with this application note is one reference system:

- www.xilinx.com/bvdocs/appnotes/xapp1016.zip

Introduction

This application note describes the procedure required to get started with Nucleus PLUS RTOs. It provides the necessary tools and setup required to build and debug a Nucleus PLUS based software application targeting the Xilinx MicroBlaze Embedded Processor.

Hardware and Software Requirements

The software requirements are:

- Mentor Graphics EDGE Tools Evaluation or fully Licensed version
- MicroBlaze Nucleus PLUS BSP
- Xilinx Platform Studio 9.1i with all service packs or later
- Xilinx ISE™ 9.1i with all service packs or later
- HyperTerminal or another terminal emulator

The Hardware requirements are:

- Xilinx Spartan™-3E Starter board
- RS232 Serial Cable
- Xilinx Parallel Cable 4 or USB Programming Cable

The design can be ported to any MicroBlaze-capable board.

System Specifics

Nucleus PLUS is a product of Accelerated Technology, a Mentor Graphics Division. Nucleus PLUS is a real-time multitasking kernel. Approximately 95% of Nucleus PLUS is written in ANSI C. This portion of PLUS is identical across all hardware platforms. Hardware dependent code is limited to three assembly code files and one header file.

© 2007 Xilinx, Inc. All rights reserved. All Xilinx trademarks, registered trademarks, patents, and further disclaimers are as listed at <http://www.xilinx.com/legal.htm>. PowerPC is a trademark of IBM Inc. All other trademarks and registered trademarks are the property of their respective owners. All specifications are subject to change without notice.

NOTICE OF DISCLAIMER: Xilinx is providing this design, code, or information "as is." By providing the design, code, or information as one possible implementation of this feature, application, or standard, Xilinx makes no representation that this implementation is free from any claims of infringement. You are responsible for obtaining any rights you may require for your implementation. Xilinx expressly disclaims any warranty whatsoever with respect to the adequacy of the implementation, including but not limited to any warranties or representations that this implementation is free from claims of infringement and any implied warranties of merchantability or fitness for a particular purpose.

Nucleus PLUS RTOS Characteristics

Some of the Nucleus PLUS RTOS characteristics are listed below.

- Small footprint
- High speed, multi-tasking kernel
- Scaleable hard real-time kernel
- Priority, pre-emptive scheduler
- Inter-task communication
- Inter-task synchronization
- memory management
- Dynamic creation and deletion of all objects

Nucleus PLUS RTOS Components

The main components of Nucleus PLUS are listed below.

- Common Services (CS)
- Initialization (IC)
- Thread Control (TC)
- Timer (TM)
- Mailbox (MB)
- Queue (QM)
- Pipe (PI)
- Semaphore (SM)
- Event Group (EV)
- Partition Memory (PM)
- Dynamic Memory (DM)
- Input/Output Driver (IO)
- History (HI)
- Error (ER)
- License (LI)
- Release (RL)

Nucleus PLUS RTOS Architecture

Nucleus PLUS RTOS requires a periodic interrupt to provide time-oriented services such as time-slicing, service call time outs, and application timers. The default setup for the timer interrupt is set up on a 10 ms period.

The Nucleus RTOS is designed to be used as a C library. Nucleus services used inside the SW application are extracted from the PLUS library and combined with the application objects to produce the complete executable image. All files that access Nucleus PLUS services must include the header file `nucleus.h`. The `Application_Initialize` function is the starting point of the user SW application.

Nucleus PLUS defines several standard data types in the include file `nucleus.h`. These data types are guaranteed to remain common across platforms by assigning the appropriate basic data type of the target C compiler. This allows PLUS to perform in an identical manner on multiple platforms.

Nucleus PLUS Integration in Xilinx EDK

As the use of Xilinx FPGAs in embedded systems grows, the need to facilitate the integration of the RTOS Design Flow into the FPGA design cycles increases.

Xilinx provides the EDK (Embedded Developers Kit) Tools with an XPS (Xilinx Platform Studio) user interface. The Embedded Development Kit (EDK) bundle is an integrated software solution for designing embedded processing systems. This pre-configured kit includes the Platform Studio tool suite (XPS) along with the documentation and IP that are required for designing Xilinx Platform FPGAs with embedded PowerPC™ hard processor cores and MicroBlaze soft processor cores, or both.

The Microprocessor Library Definition (MLD) Technology is used for auto-customization of libraries and Board Support Package (BSPs). It is a data-driven process based on user supplied configuration parameters. It supports the specification of specific HW requirements for any SW services support which is a new data-driven capability that generates the RTOS BSP tailored to the defined HW platform.

Xilinx EDK includes the device driver library definition for commercial RTOS, such as Nucleus.

Nucleus provides a device driver abstraction layer for EDK supported IP peripherals.

Based on the user specification, EDK (specifically LibGen Tools in XPS) generates a Nucleus BSP corresponding to the hardware platform design.

Installing the EDGE Environment and MicroBlaze BSP

Mentor Graphics EDGE Tools Overview

The Mentor Graphics Embedded Developers Graphical Environment (EDGE) Tools is a comprehensive embedded tools environment based on Eclipse.

EDGE provides specialized Perspectives such as the Project Perspective which is a fully featured project management solution, a builder with support for multiple tool chains, and source control, including Debug Perspective.

This provides a fully featured debugger with kernel awareness tailored for embedded applications, including Profiler Perspective, a feature rich source profile.

Getting the Nucleus PLUS BSP for MicroBlaze

The Nucleus BSP for MicroBlaze is provided by Mentor Graphics and it is available from Mentor Graphics web site at:

http://www.mentor.com/products/embedded_software/cpu/lv_reg.cfm

Getting the EDGE Tools Evaluation Version for MicroBlaze

Mentor Graphics provides an evaluation version of the EDGE Tools for MicroBlaze which can be downloaded from the Mentor Graphics web site at:

http://www.mentor.com/products/embedded_software/development_tools/nuc_edge/evaluate.cfm

Installing the EDGE Evaluation Version Tools and its License

Once the request for access to the EDGE Tools has been done, a confirmation E-mail which includes a **license.zip** archive is sent. To install, use the following steps.

1. Extract the license.zip archive to a directory.
2. Copy demo_license.txt to the root directory of the downloaded and extracted EDGE installation files.
3. Launch the EDGE Tools setup.exe installer and follow the instructions to install the EDGE Tools.

An evaluation license is installed and configured by the EDGE installer.

When Nucleus EDGE is launched for the first time, the required tools (Xilinx GNU MicroBlaze Toolset v9.1) must be enabled in Nucleus EDGE preferences.

Under the **Windows Preferences** menu, select **Nucleus EDGE Builder Toolsets** and enable the appropriate check box. Confirm the setup by clicking OK.

Installing Nucleus PLUS BSP for MicroBlaze

The zip file, `nucleus_xilinx_microblaze_lv.zip`, contains an auto installer of the Nucleus PLUS BSP builder and the required Nucleus PLUS LV Library for MicroBlaze.

Unzip the File to a temporary directory and run the executable to install the Nucleus PLUS BSP for MicroBlaze.

The Nucleus PLUS LV is a fully-functional version of the RTOS compiled into a library format (rather than the normal source code distribution) with a single restriction — it will stop working after 60 minutes, thus facilitating the evaluation of its full functionality.

With the purchase a full license of Nucleus PLUS from Mentor Graphics, the full source code with no run time restriction is provided.

The LV of Nucleus PLUS RTOS is configured to execute from the off-chip SRAM or SDRAM/DDR module.

Once a full license for the RTOS is available, it can be configured to run from any memory in the system.

The LV of other Nucleus products provided with EDK 9.1i consists of fully functional product compiled into a library format.

Nucleus products contain robust support of Xilinx MLD technology. With the Nucleus evaluation, all configuration options within the *Library/OS Parameters* in the *Software Platform Settings* dialog may not be functional. This restriction is removed when the full version is purchased.

Once the Nucleus PLUS LV BSP builder has been run, it will install the necessary Tools into the `\edk_user_repository\`.

Figure 1 shows the layout of the created directories under:

`\edk_user_repository\Nucleus\bsp\nucleus_v1_15_a\`

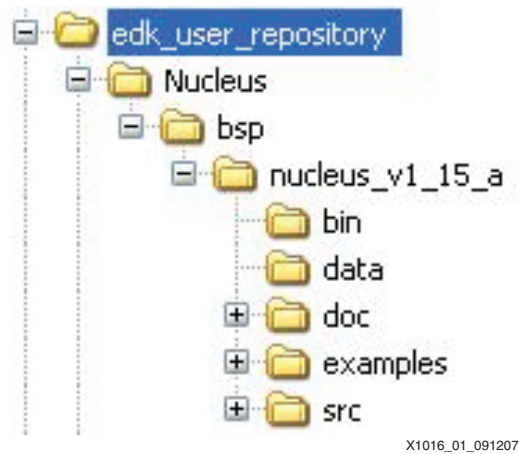


Figure 1: Nucleus PLUS LV BSP Directory Layout

Creating a MicroBlaze System to run Nucleus RTOS

Using the XPS Base System Builder to create the Embedded System

The FPGA based Embedded System consists of the following HW Peripherals:.

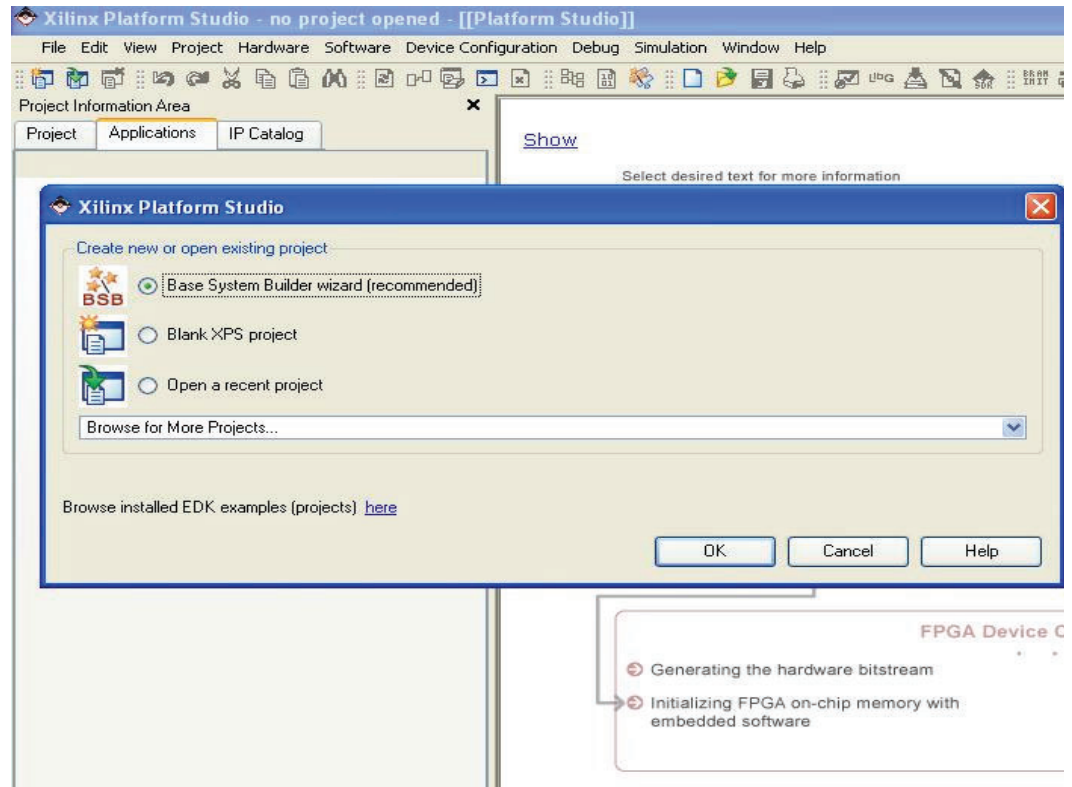
- MicroBlaze processor with 8 Kbyte instruction and data caches
- 8-KB Local on-chip memory (LMB) shared between the instructions and data sides of the MicroBlaze processor
- An OPB MCH controller for the on-board DDR SDRAM
- An OPB EMC controller for the on-board flash memory
- An OPB timer
- An OPB UART as input/output user interface
- An OPB interrupt controller
- An OPB MDM with an FSL connection to MicroBlaze for the system debug

To design the above system with EDK, the Base System Builder of XPS is used as described by Figure 4 through Figure 17.

Figure 4 shows how to launch the BSB Wizard.

When XPS is first launched, click on **OK** in [Figure 2](#) to launch the BSB.

If XPS is already open, select **File** from the XPS menu, then **New Project** to launch the BSB Wizard.



X1016_02_091207

Figure 2: Launching the BSB Wizard

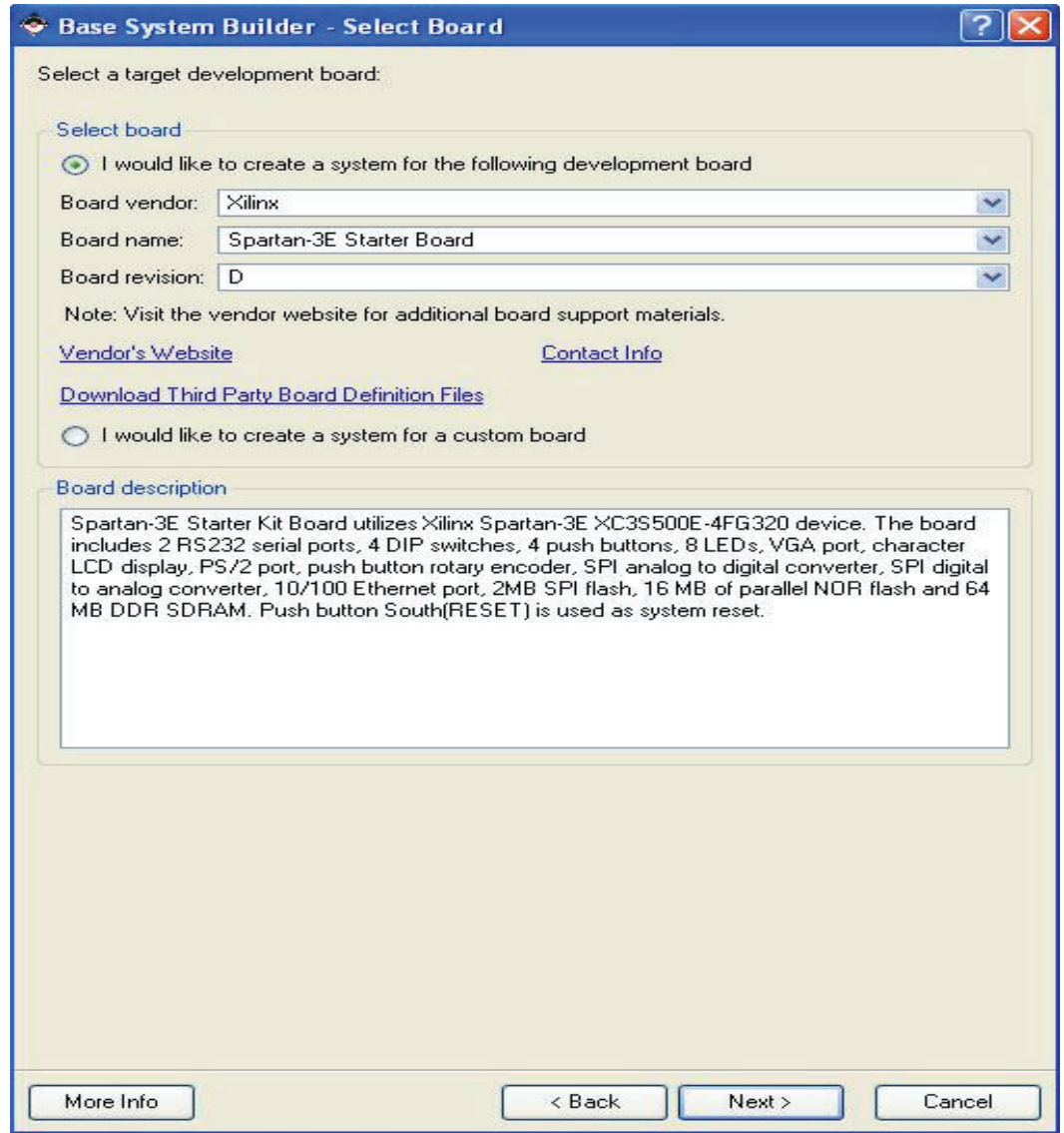
Figure 3 shows how to start the design using the BSB Wizard.



X1016_03_091207

Figure 3: Starting the Nucleus PLUS HW Platform design using the BSB

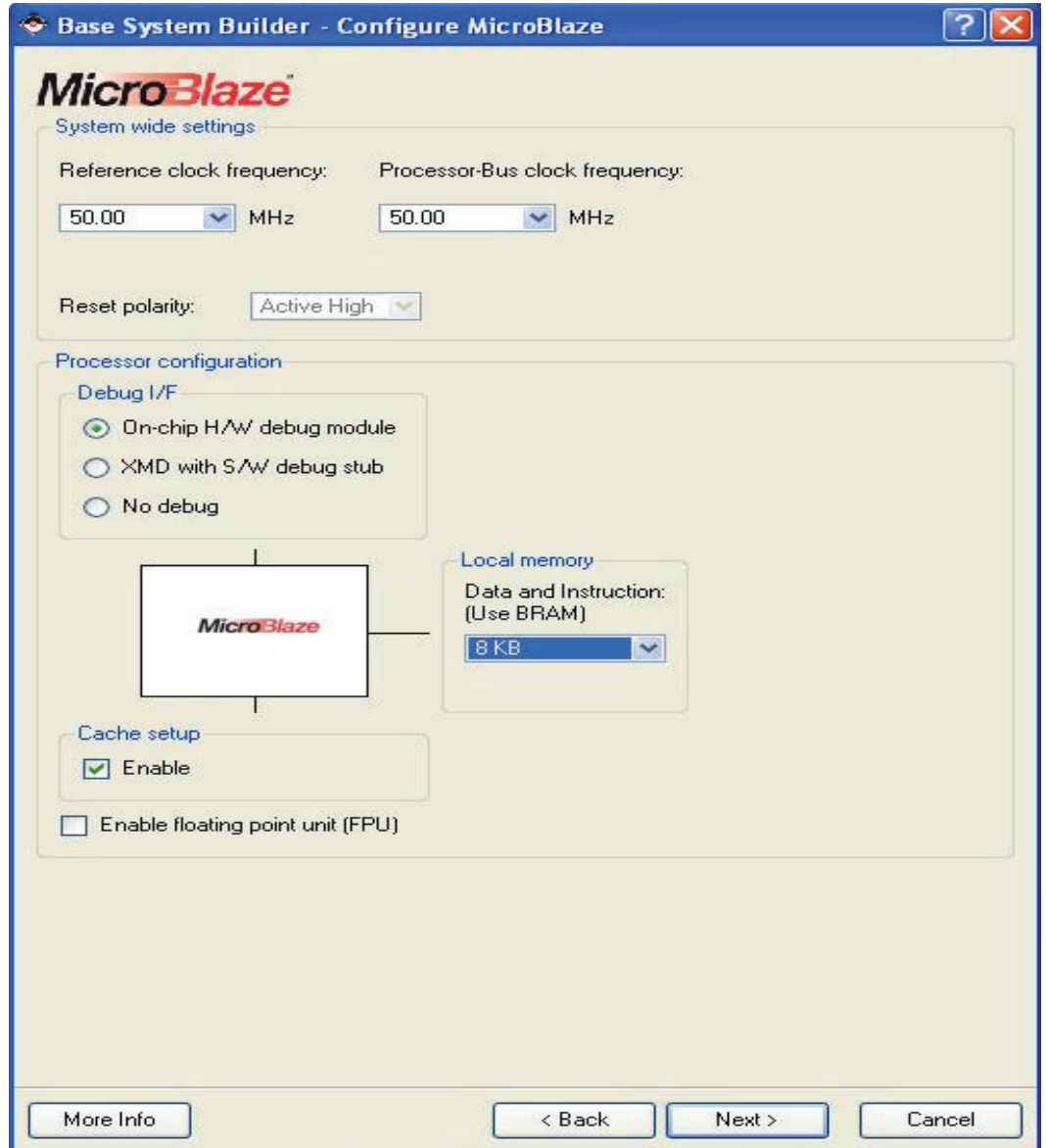
Figure 4 illustrates how to select the **Spartan3E Starter Board** as a target board.



X1016_04_091207

Figure 4: **Spartan-3E Starter Board Selection in the BSB Wizard**

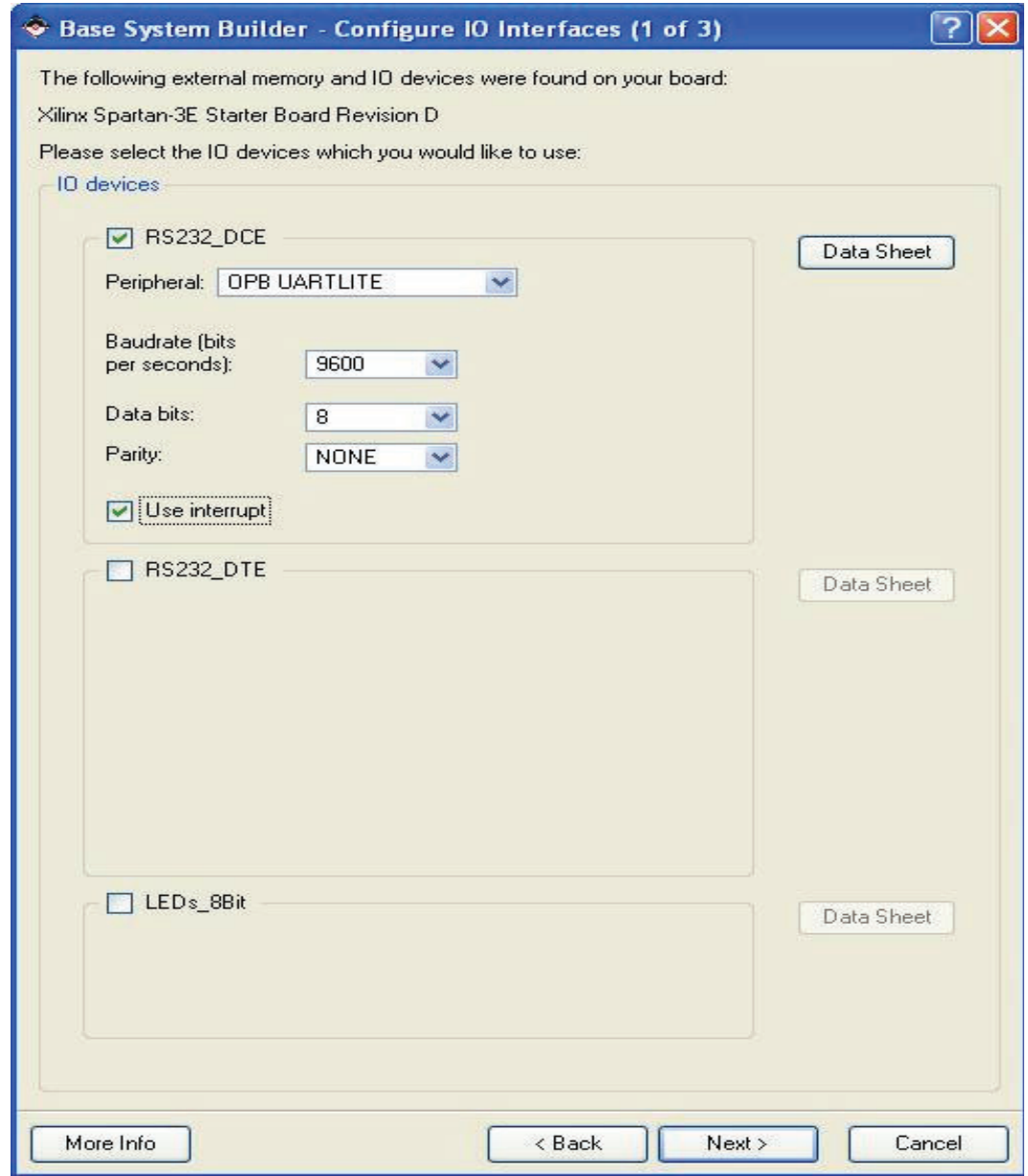
Figure 5 shows how to configure the MicroBlaze processors and some of the system features.



X1016_05_091207

Figure 5: Setting up the MicroBlaze Soft Processor

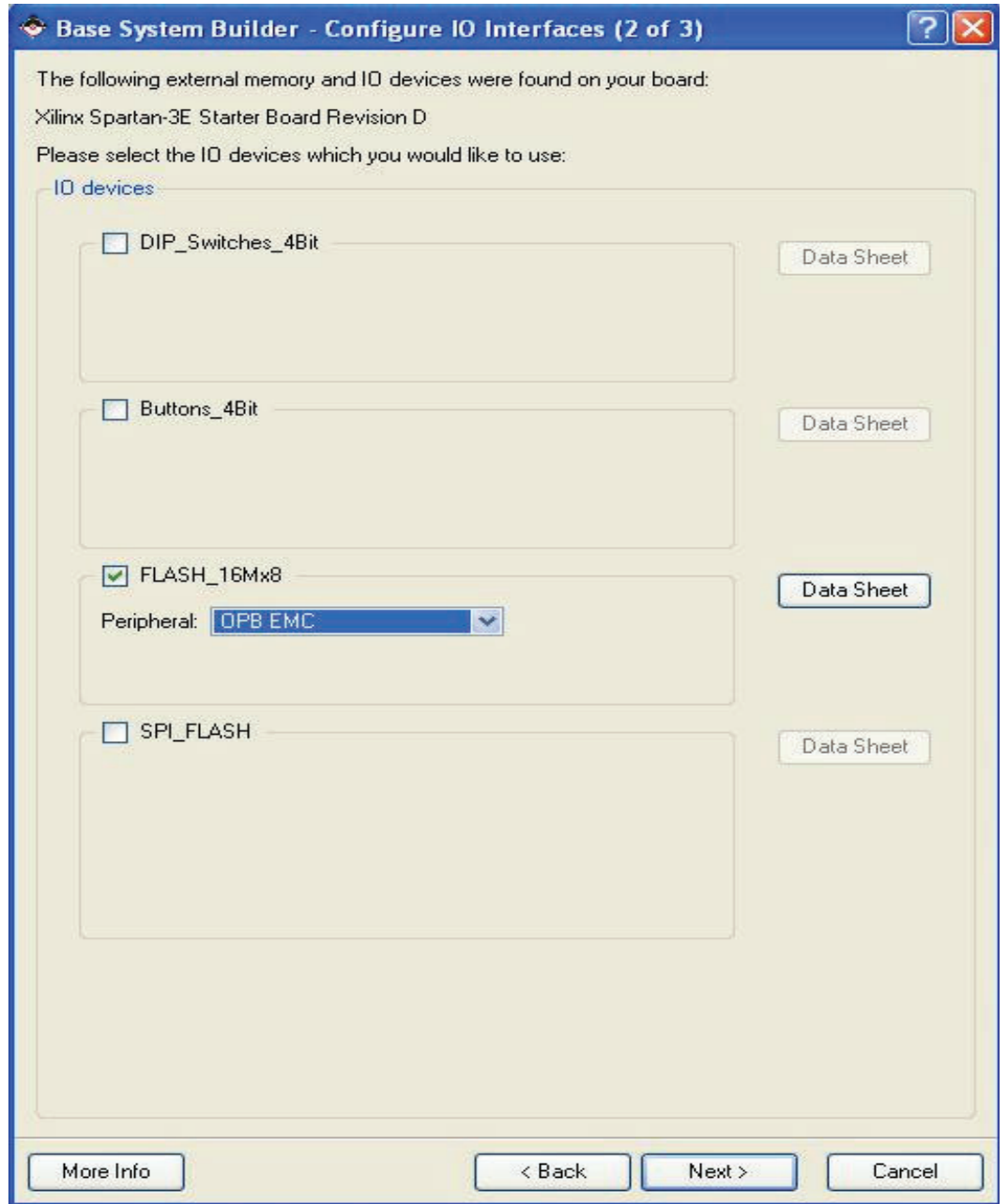
Figure 6 shows how to select the OPB UART Lite peripheral and setup its parameters.



X1016_06_091207

Figure 6: OPB UART Lite Selection and Setup

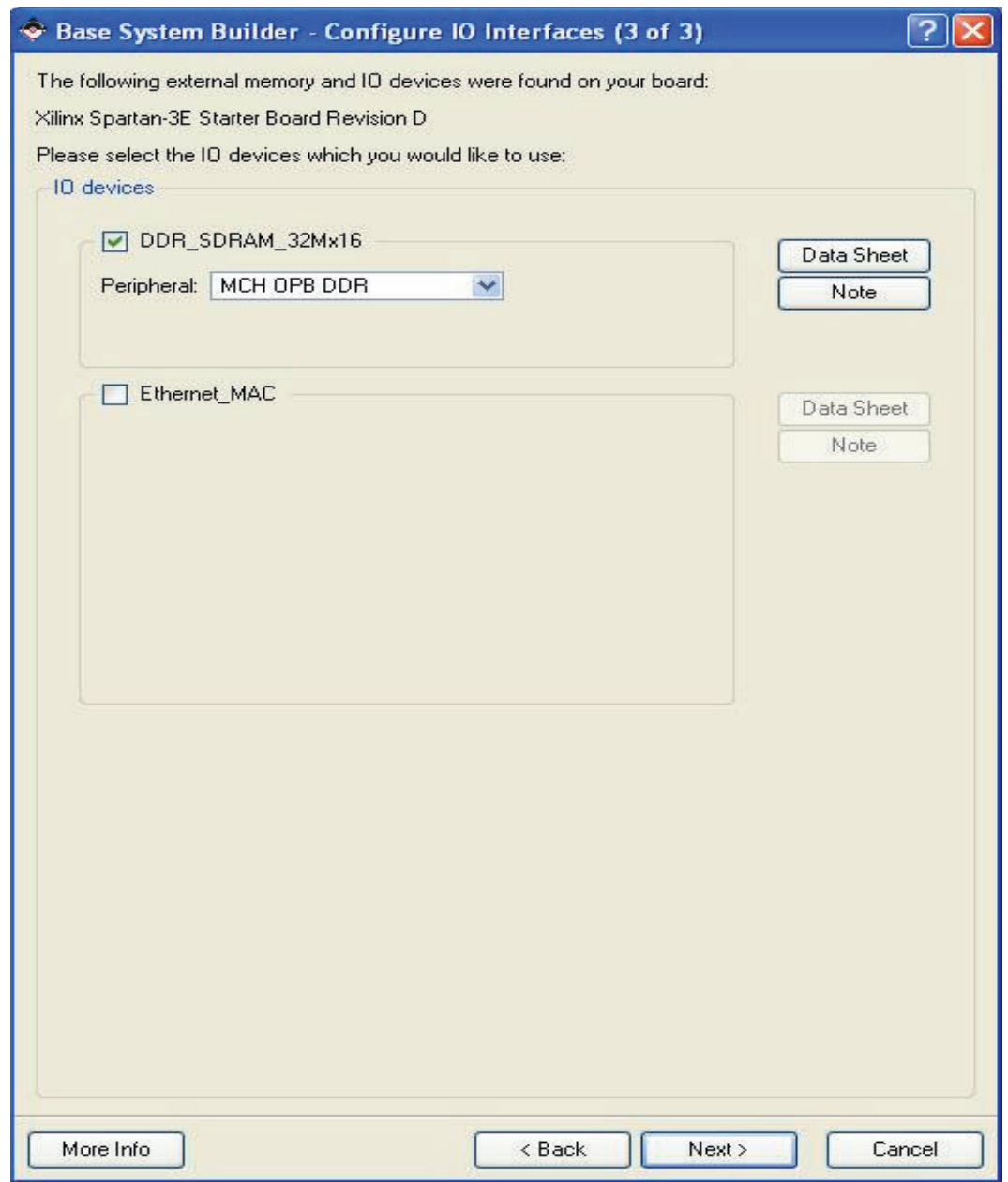
Figure 7 illustrates how to select the OPB EMC to interface to the flash memory.



X1016_07_091207

Figure 7: OPB EMC Selection

Figure 8 shows how to select the MCH OPB DDR peripheral to interface to the DDR SDRAM Memory.



X1016_08_091207

Figure 8: MCH OPB DDR Controller Selection

Figure 9 shows how to add the OPB Timer to the system.

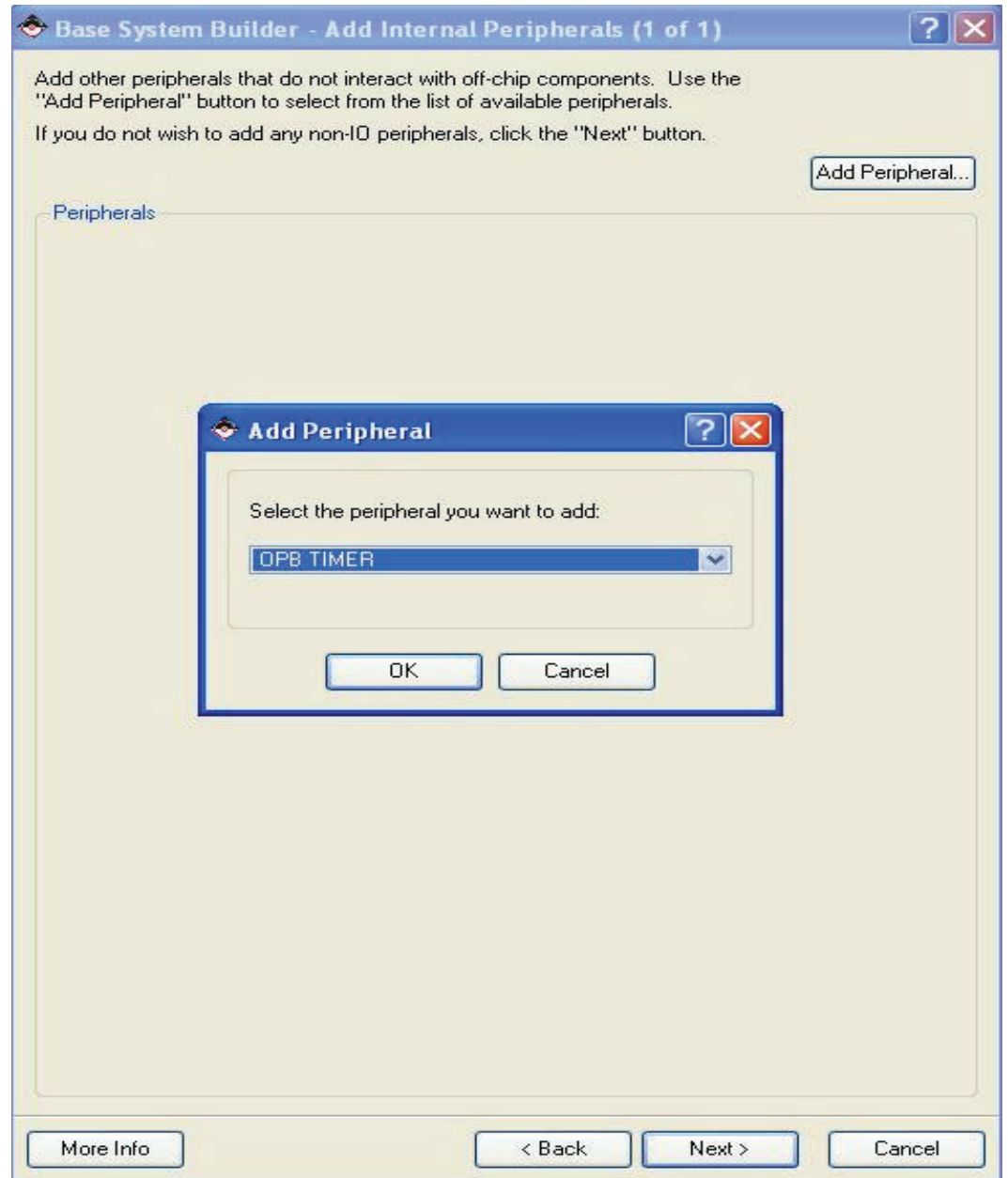
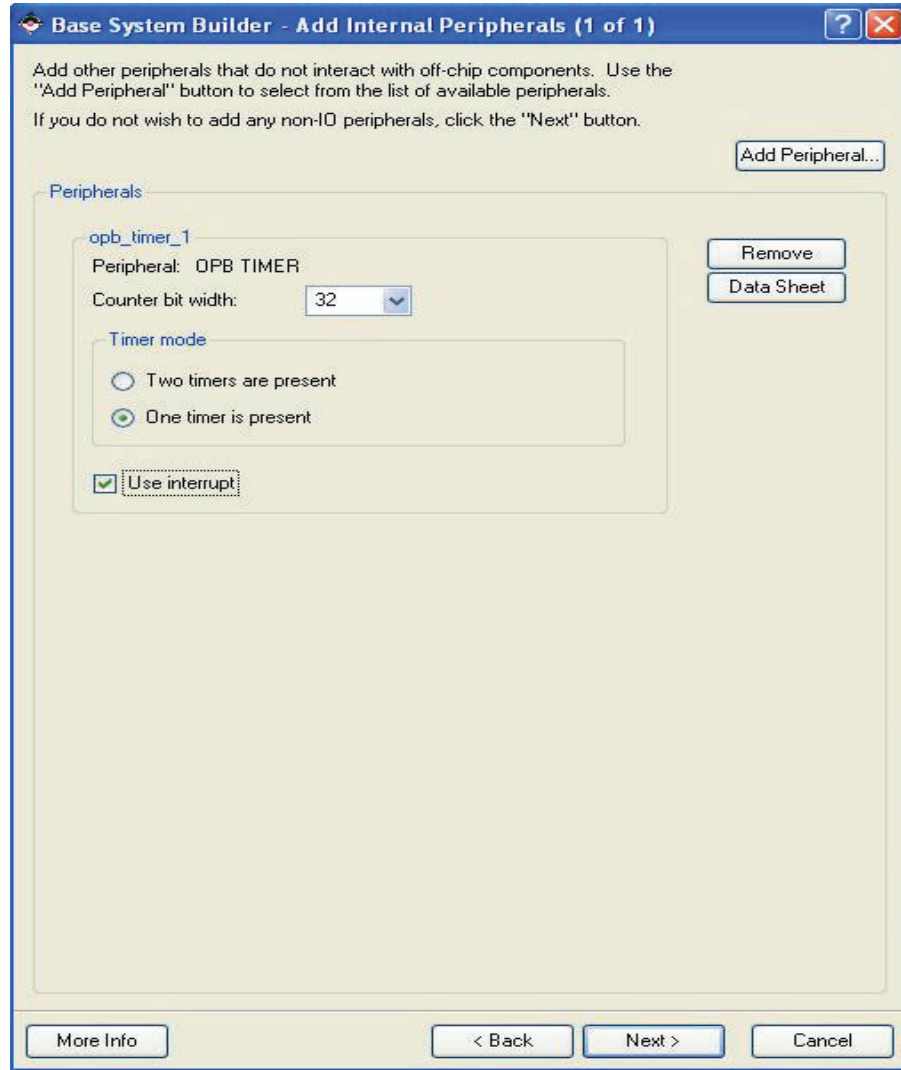


Figure 9: Adding the OPB Timer to the System

Figure 10 shows how to configure the OPB timer.

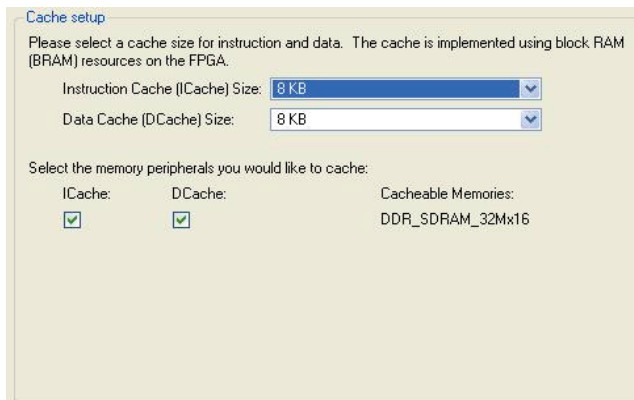


X1016_10_091207

Figure 10: Configuring the OPB Timer

Figure 11 shows how to setup the MicroBlaze caches.

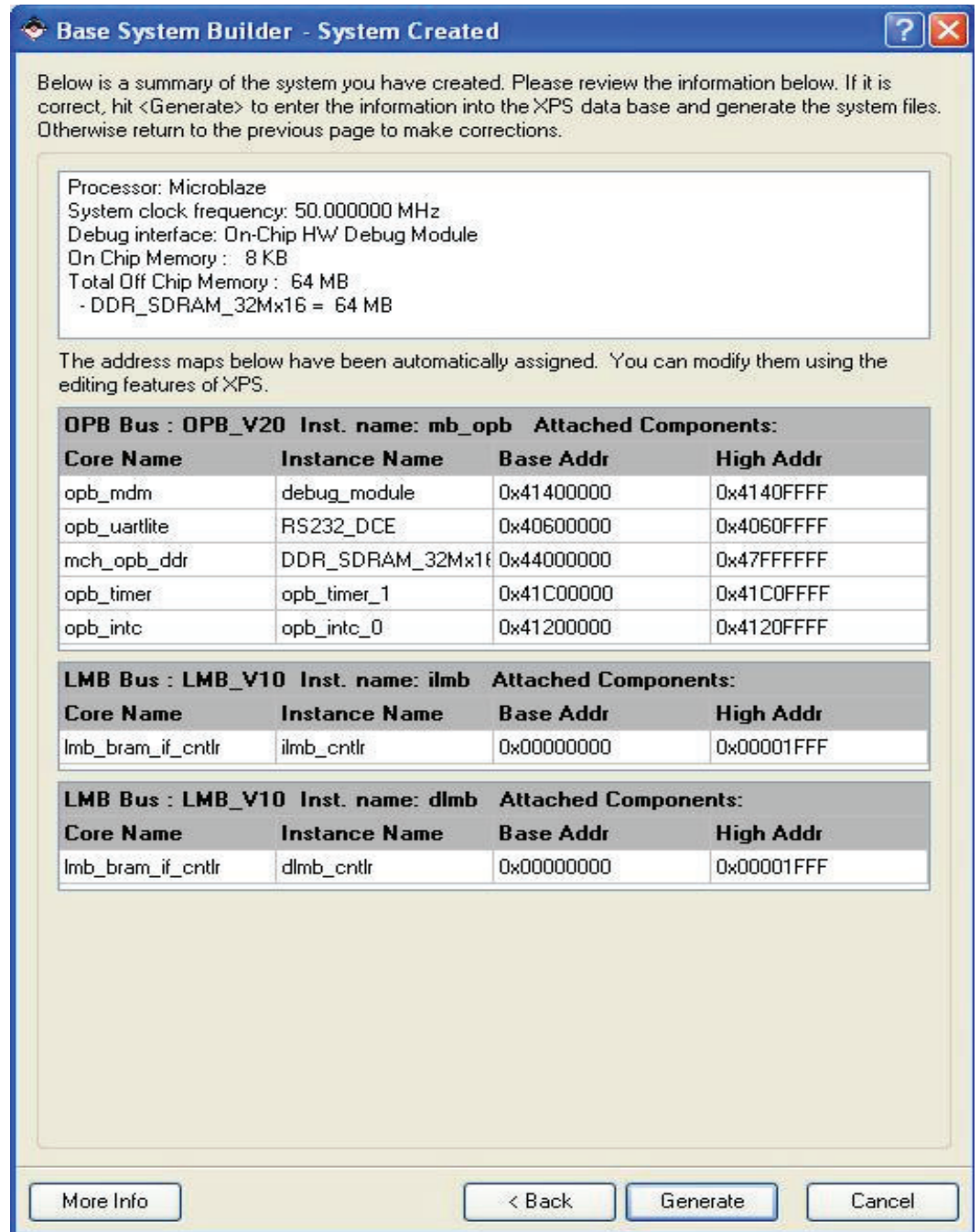
A Cache Size of any allowed value can be selected depending on the SW Application needs and the available BRAM resources in the target FPGA.



X1016_11_091207

Figure 11: Configuring the MicroBlaze Caches

Figure 12 shows the BSB generated system.



X1016_12_091207

Figure 12: BSB Generated System

Configuring the generated System in XPS

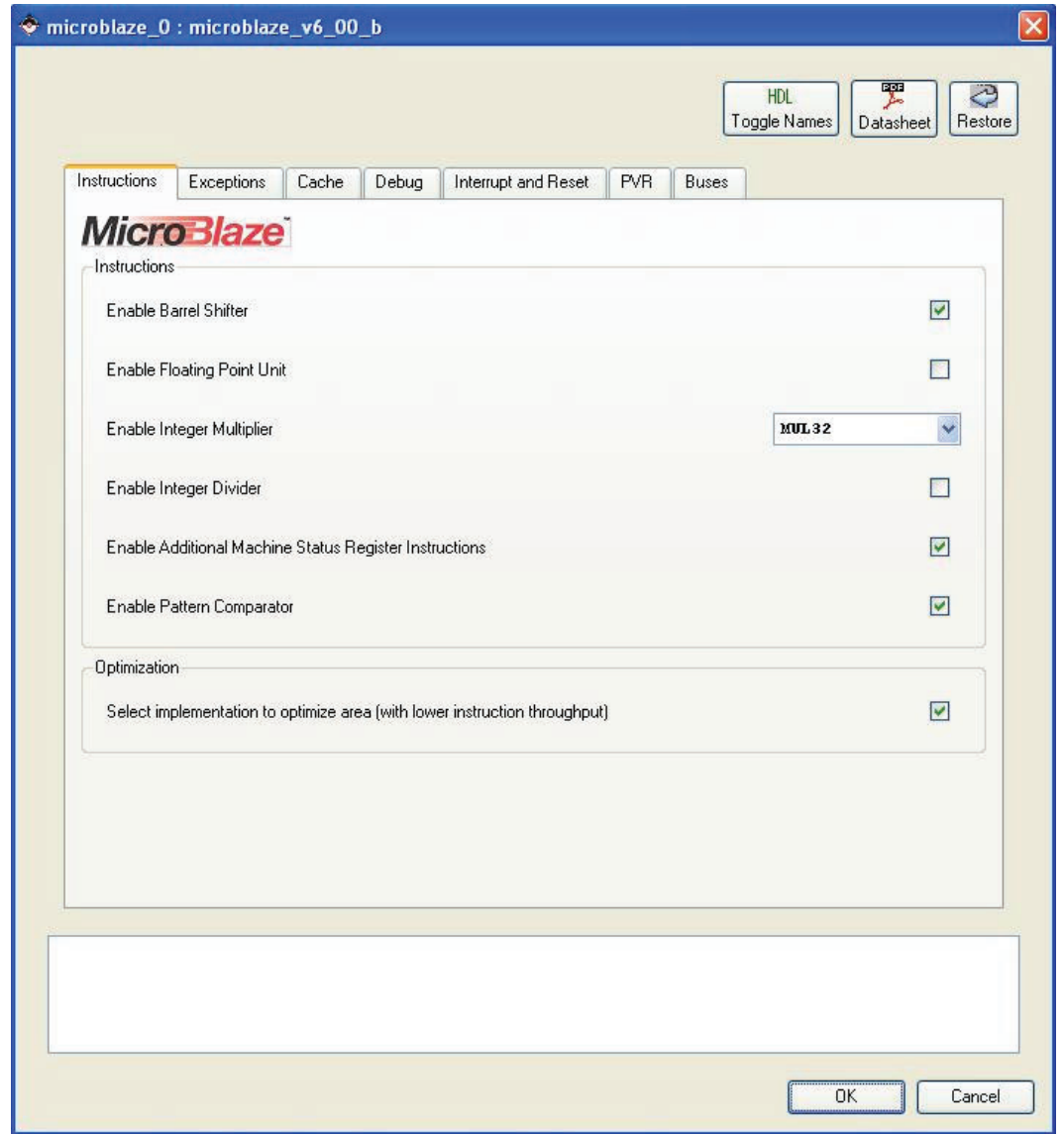
At this stage, the required HW design to run the Nucleus PLUS RTOS based applications SW is almost finished. To decrease the download time of the SW application images via the JTAG cable to the system main memory, an optional FSL (Fast Simplex Link) can be added to the system. This FSL Link is a point to point connection between the MicroBlaze soft processor and the OPB MDM peripheral.

XPS is used to add the download FSL Link and to customize some other features of the MicroBlaze soft processor.

This download feature is enabled in the MicroBlaze soft processor by adding a Slave FSL interface to the core as shown by [Figure 14](#).

[Figure 13](#) shows how to customize the MicroBlaze soft core features using XPS.

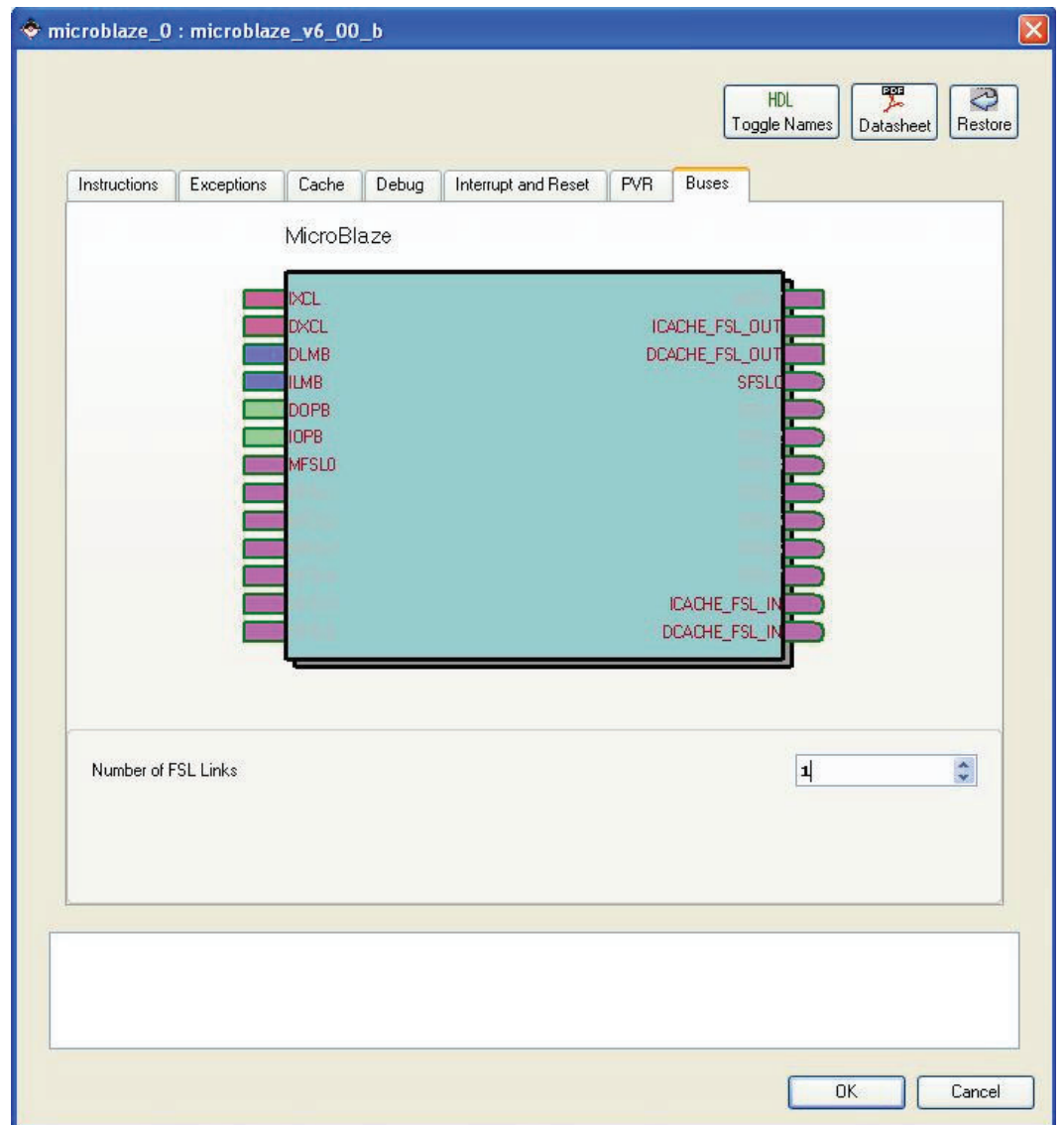
To access this GUI, in the System Assembly view, double-click on the **microblaze_0** processor instance, then select **Enable Barrel Shifter**.



X1016_13_091207

Figure 13: Customizing MicroBlaze Core in XPS

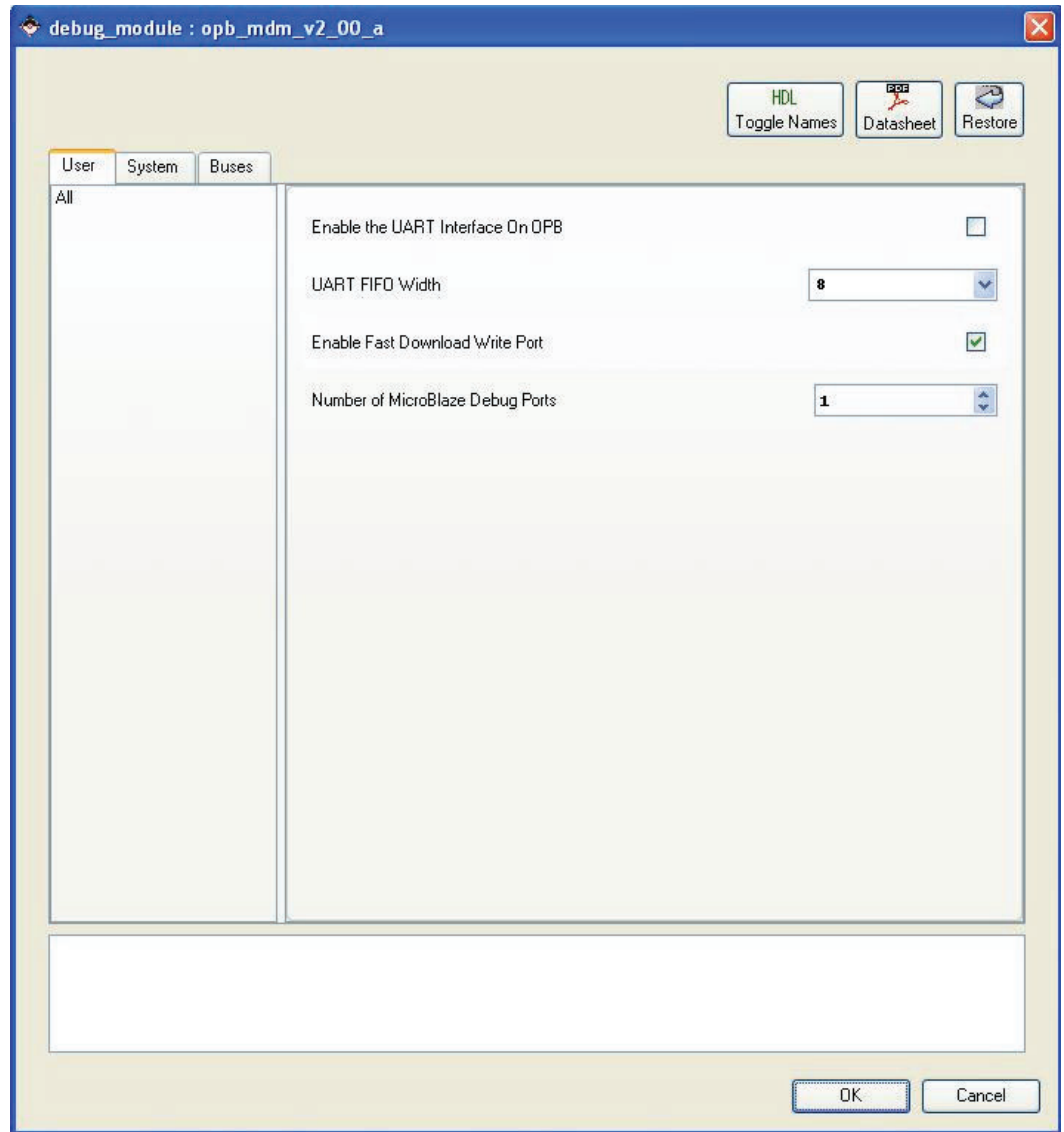
Figure 14 shows how to add the Slave FSL interface to the MicroBlaze soft core. In the same GUI shown in Figure 13, select the Buses tab, and select 1 in the Number of FSL Links field.



X1016_14_091207

Figure 14: Adding an FSL Interface to the MicroBlaze Soft Core

Figure 15 shows how to enable the Fast Download Link in the OPB MDM Core. In the System Assembly view, double-click on the **debug_module** instance. Check the **Enable Fast Download Write Port**.



X1016_15_091207

Figure 15: Enabling the OPB MDM JTAG Fast Download Link

Figure 16 shows how to add the FSL to the System.

In the IP Catalog tab, right click on **Fast Simplex Link (FSL) Bus**, then Add IP in the System Assembly view.

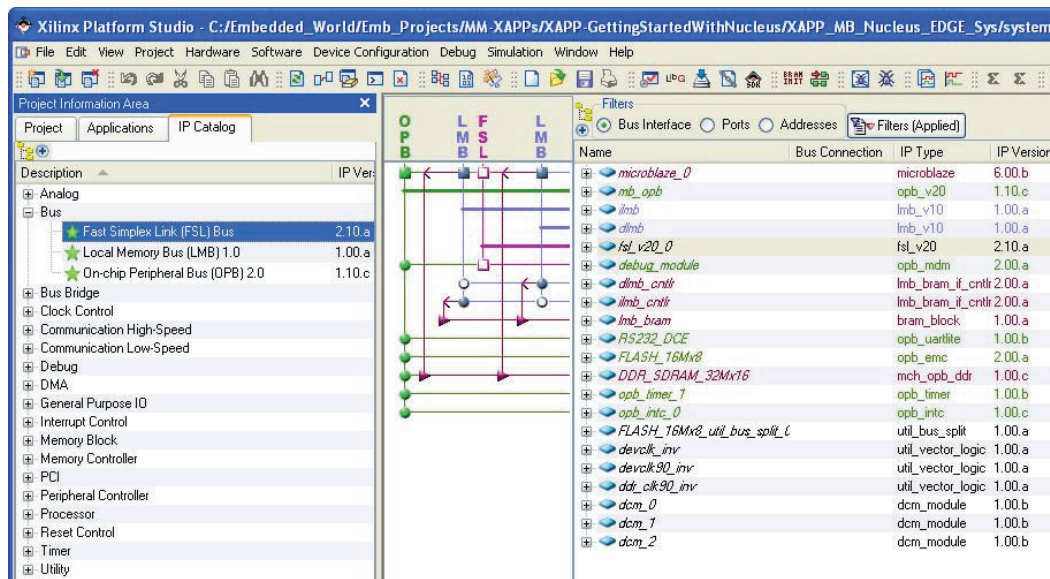


Figure 16: Adding and FSL Link to the System

Figure 17 shows how to connect the added FSL Link to the MicroBlaze soft core slave FSL interface and to the OPB MDM FSL master interface.

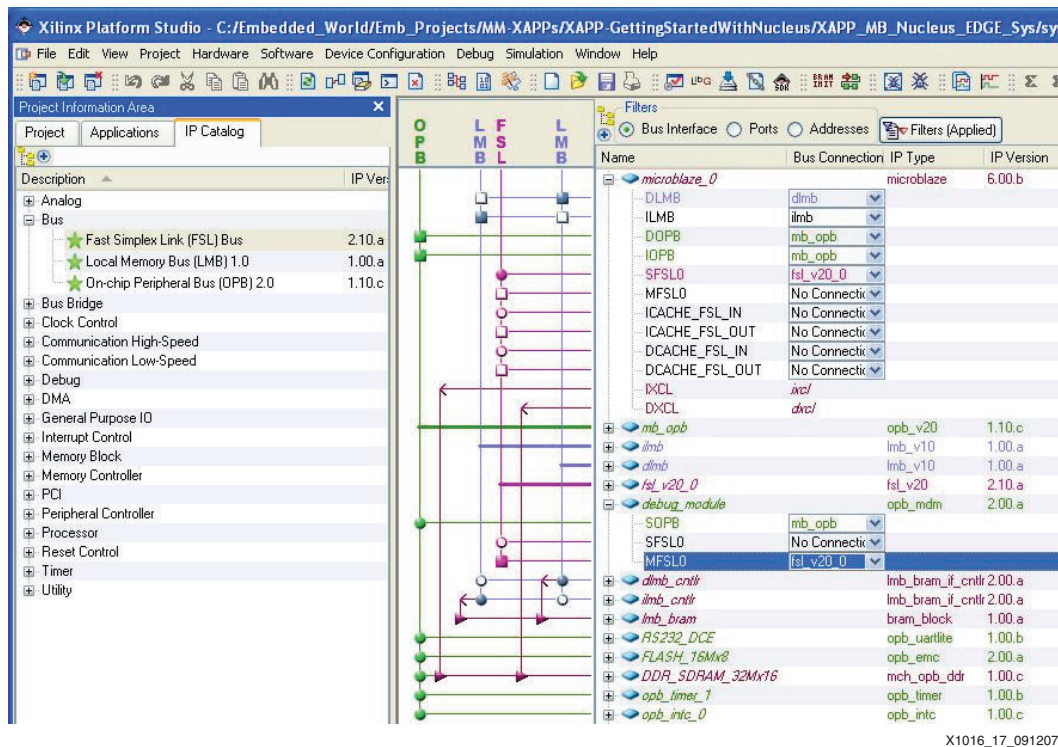


Figure 17: Making the FSL Connections for JTAG Fast Download

Implementing the Design in XPS

In XPS, select **Device Configuration** → **Update Bitstream** to generate the `download.bit` bitstream.

This FPGA Bitstream version has the auto-generated TestApp_Memory Standalone SW Application Executable as an initialization file embedded inside the MicroBlaze LMB BRAM memory. Upon FPGA configuration, this SW application is executed by the MicroBlaze processor to test the SDRAM DDR memory on the S3E starter kit.

Start a HyperTerminal session with the following settings.

com1

Bits per second: **9600**

Data bits: **8**

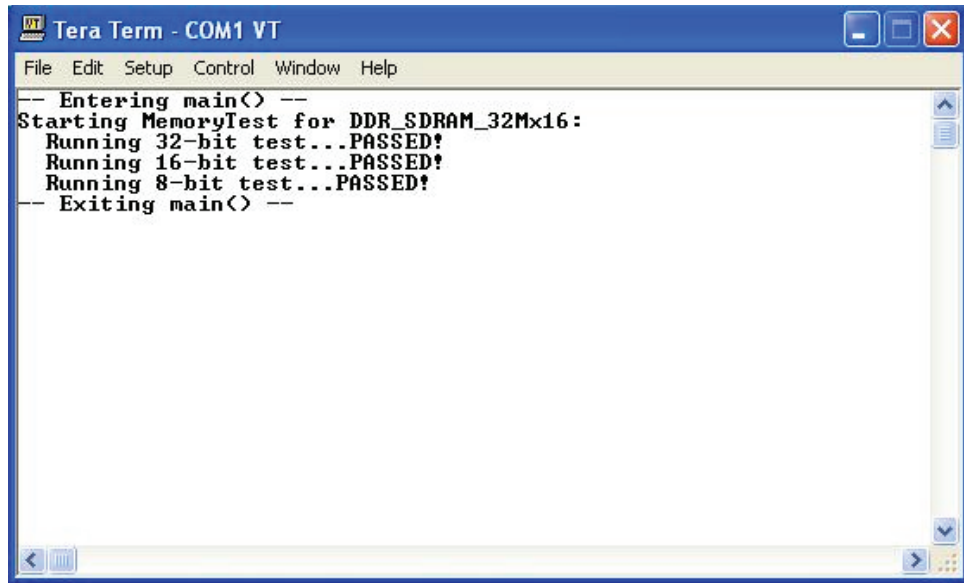
Parity: **none**

Stop bits: **1**

Flow control: **none**

Download the bitstream Device Configuration Download Bitstream.

The memory test output appears on the HyperTerminal as shown by [Figure 18](#).



```

Tera Term - COM1 VT
File Edit Setup Control Window Help
-- Entering main<> --
Starting MemoryTest for DDR_SDRAM_32Mx16:
Running 32-bit test...PASSED!
Running 16-bit test...PASSED!
Running 8-bit test...PASSED!
-- Exiting main<> --

```

X1016_18_091207

Figure 18: Running the TestApp_Memory on the S3E Starter Kit

Another bitstream version is generated. This version will include a simple BootLoop only as an initialization file in the MicroBlaze LMB BRAM Memory.

Select the **microblaze_0_bootloop** application for block RAM initialization, and deselect the **TestApp_Memory** application from the block RAM initialization.

Select **Device Configuration** → **Update Bitstream** to generate the **download.bit**.

Download the bitstream Device Configuration Download Bitstream.

Configuring the BSP in XPS

At this stage the HW Platform has been built and the BSP may be generated automatically using the XPS Libgen and Mentor Graphics MLD based BSP builder previously installed into the /edk_user_repository/.

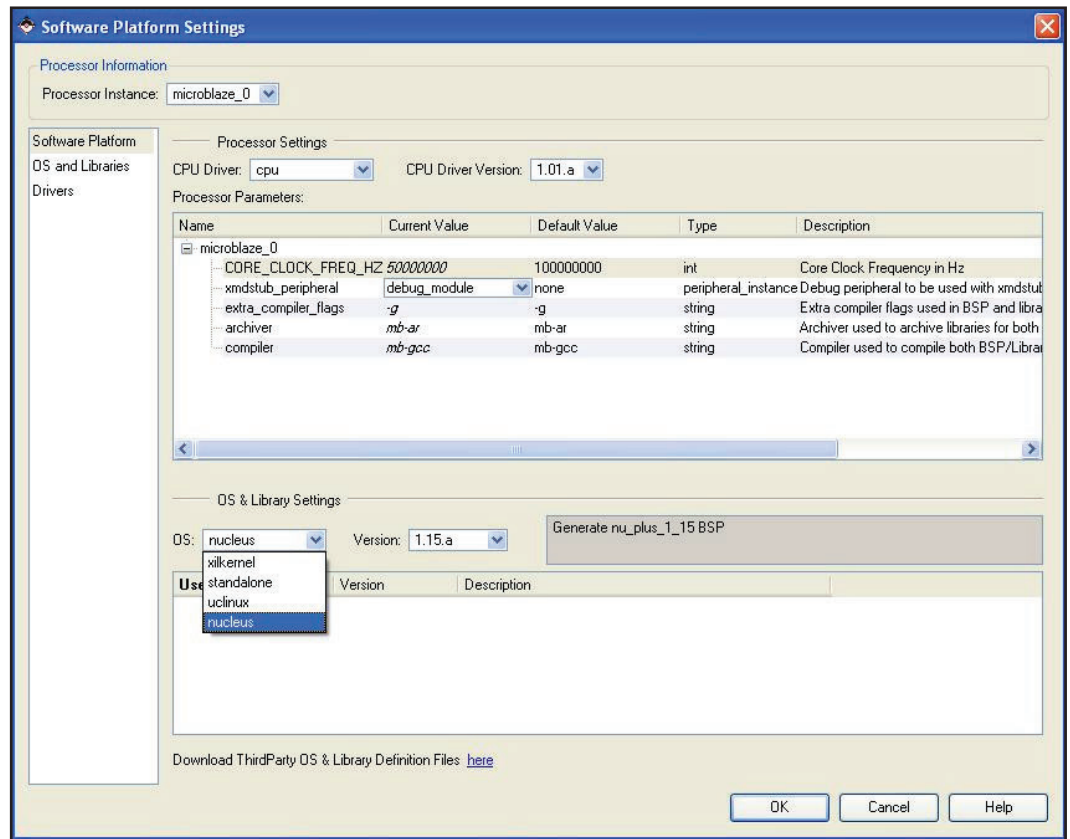
Setting up the MSS File

The first step in BSP generation process is to configure the BSP in the Microprocessor Software Specification (MSS) file.

The XPS Software Platform Setting is the GUI used for configuring the Nucleus PLUS BSP for the system.

To launch the Software Platform Setting, in XPS select **Software** → **Software Platform Setting**.

Figure 19 shows the Software Platform Settings window. In the Processor Parameters sub-window, set the MicroBlaze core clock frequency value to 50 MHz and select Nucleus PLUS as the OS in the OS & Library Settings sub-window.

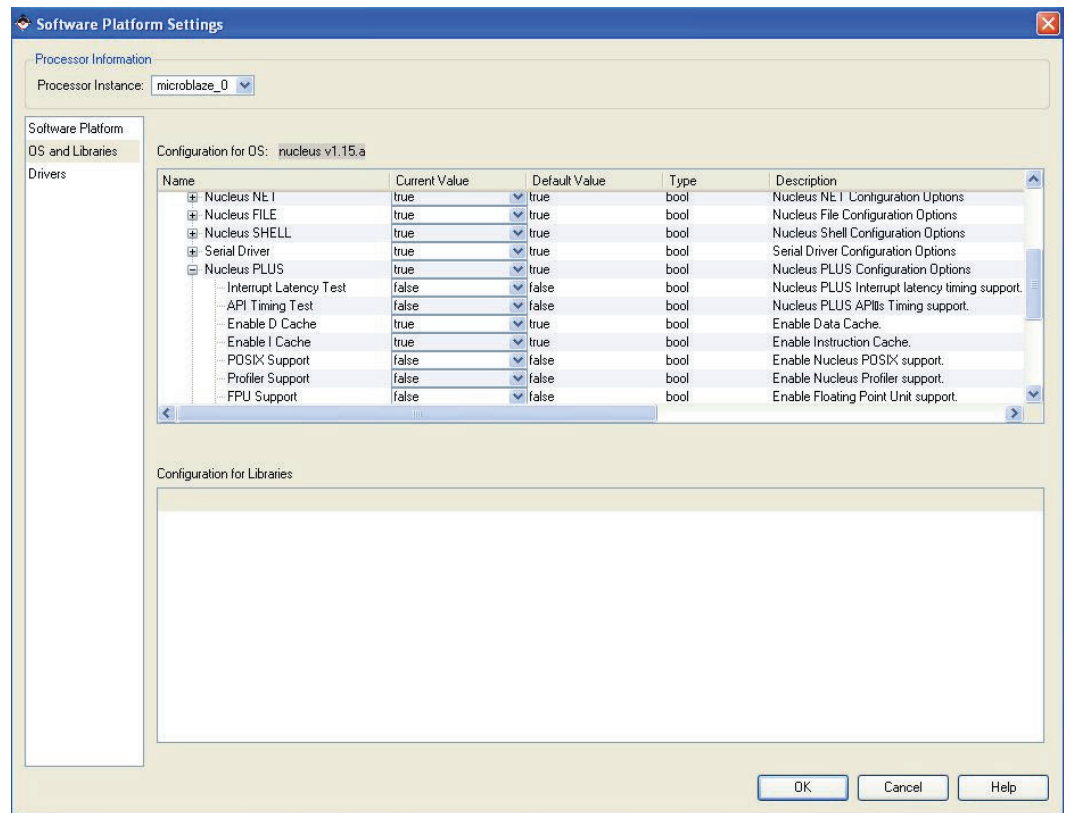


X1016_19_091207

Figure 19: Core Clock Frequency Setup & RTOS Selection

Select the OS and Libraries Tab, then configure the Nucleus BSP as shown in [Figure 20](#).

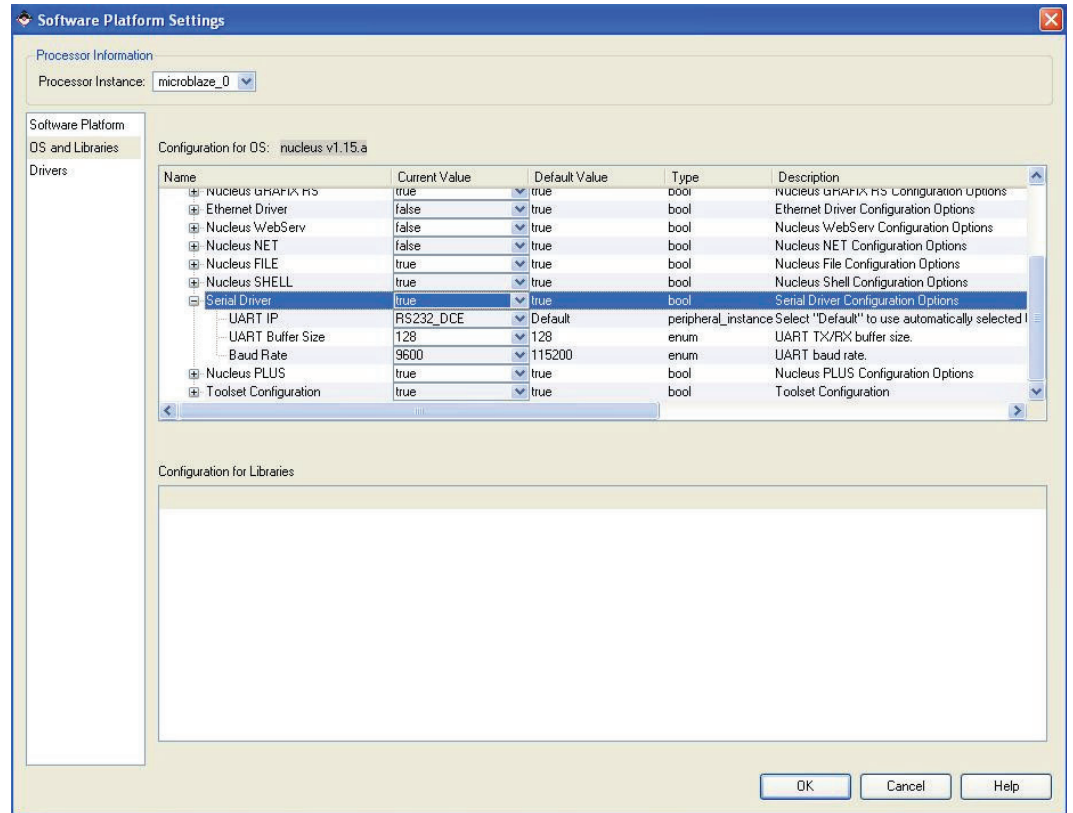
In this window, Nucleus Library Services can be included or excluded according to the Application needs. Template Projects and services Demos could also be requested at this stage. By default, every Nucleus Library Service Demo Project is selected.



X1016_20_091207

Figure 20: Nucleus PLUS BSP Configuration

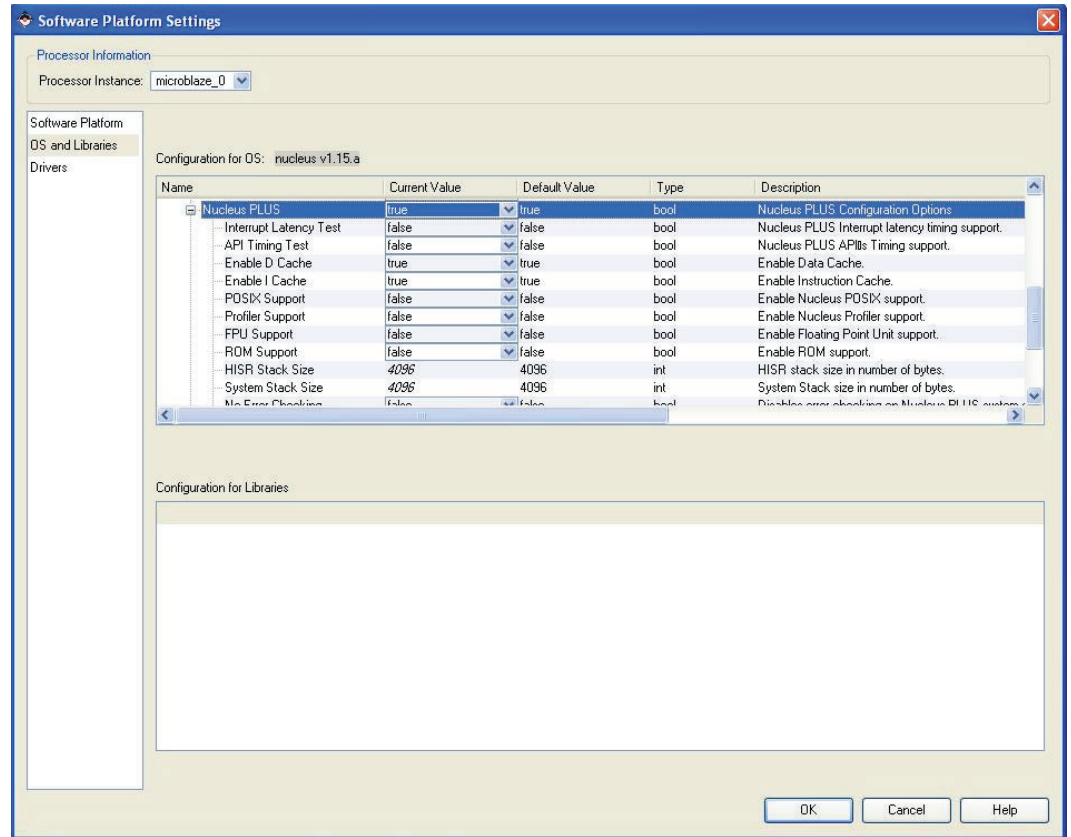
Figure 21 shows how to set up the Serial Driver.



X1016_21_091207

Figure 21: Nucleus PLUS Serial Drivers Configuration

Figure 22 shows some of the Nucleus PLUS LV Library setup and features.



X1016_22_091207

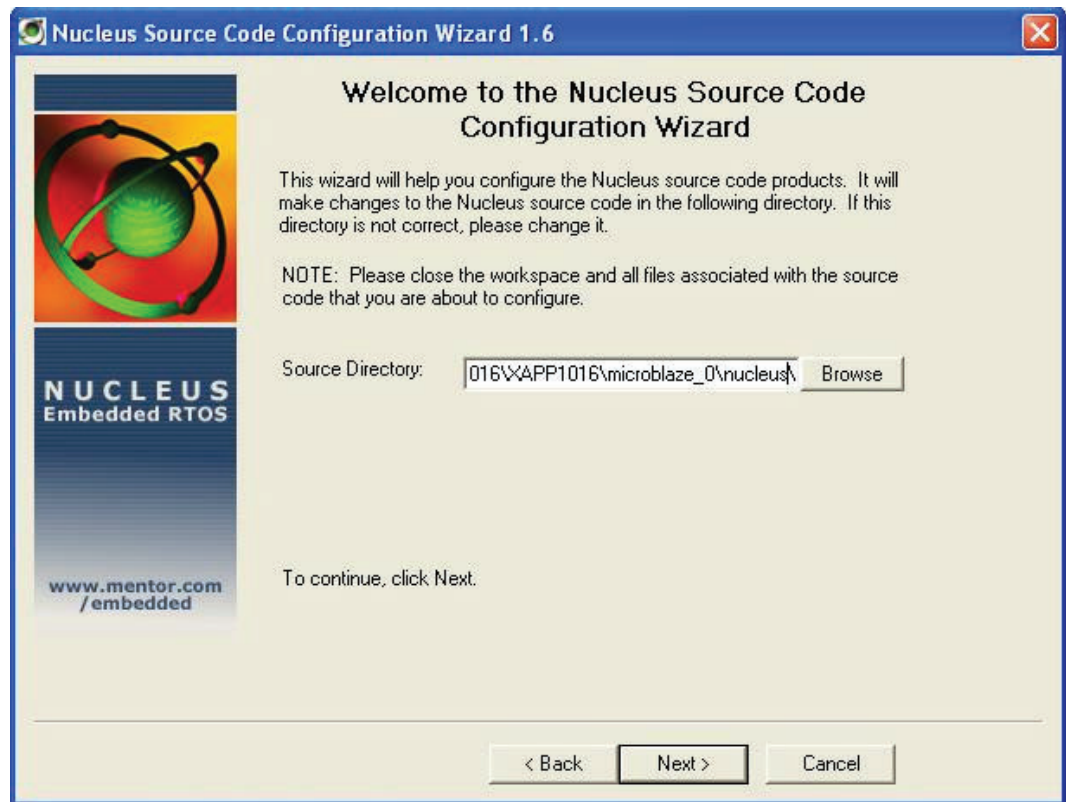
Figure 22: Nucleus PLUS Library Setup

Click **OK** to save the configuration of the Nucleus PLUS LV BSP settings in the MSS file.

Generating the Nucleus PLUS LV Libraries and BSP

At this stage the Nucleus PLUS LV Libraries and BSP can be generated by running Libgen in XPS from **Software** → **Generate Libraries and BSPs**.

Once LibGen is run, a Nucleus Source Code Configuration Wizard is launched automatically as shown in [Figure 23](#). Other Nucleus Services can be configured if they were selected during the setup process.



X1016_23_091207

Figure 23: Nucleus Source Code Configuration Wizard

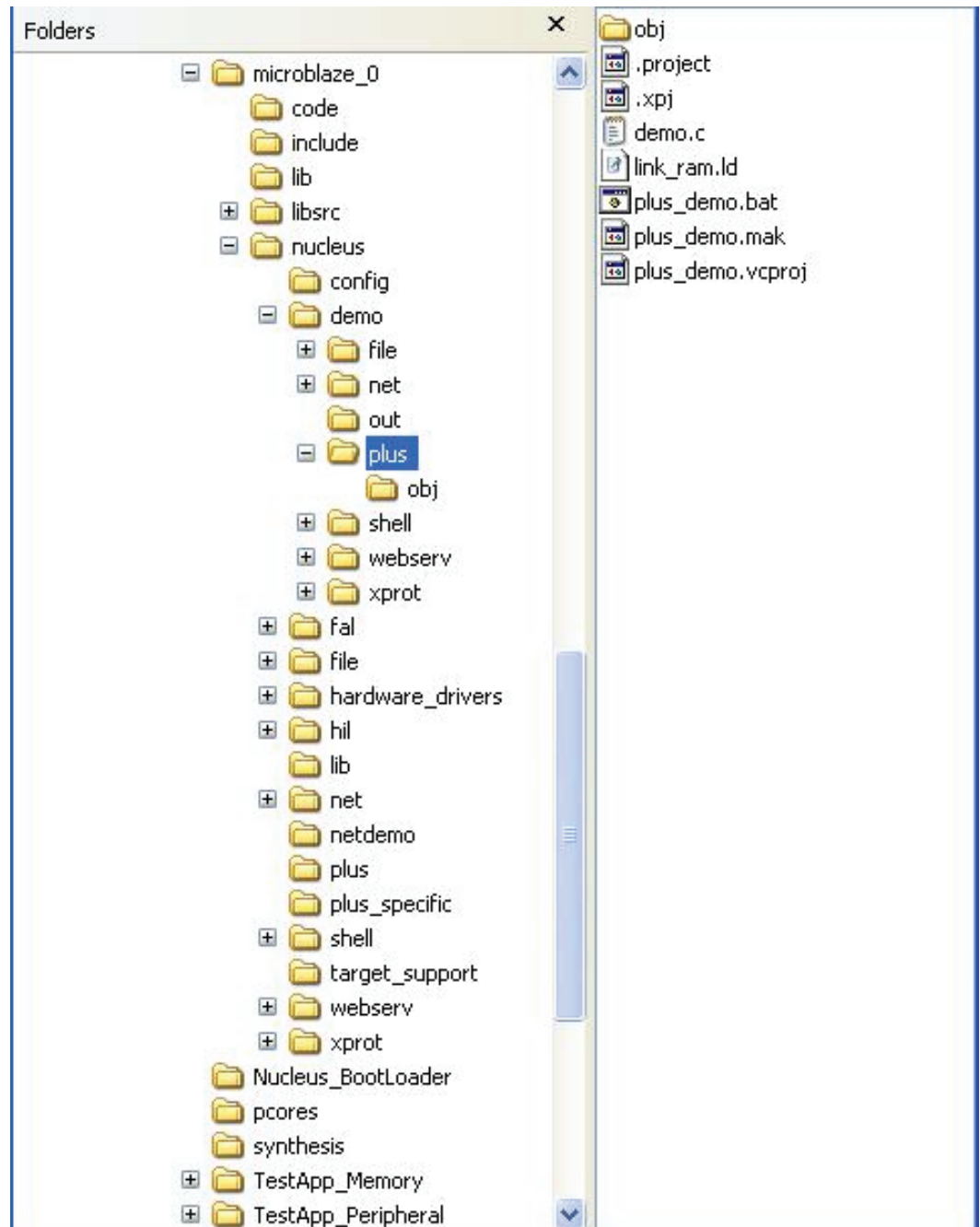
Click **Next** to reach the last screen of the BSP generation Wizard.

Creating a Nucleus Project in EDGE

Once the Generate Libraries and BSPs step is completed in XPS, a `nucleus` folder is created under the Processor Instance Name (`microblaze_0`) folder within the XPS Project directory.

Figure 24 shows the hierarchy of the created folder.

Under the `plus` folder, an EDGE Template Project tailored to the associated XPS Project is created. This Template Project can be used as a starting point for the SW Application development process.



X1016_24_091207

Figure 24: Nucleus PLUS Folder layout

Building the SW Application in EDGE

The plus_demo EDGE project is used.

Launch the EDGE software, then select a Workspace for EDGE, if necessary, as shown in Figure 25.

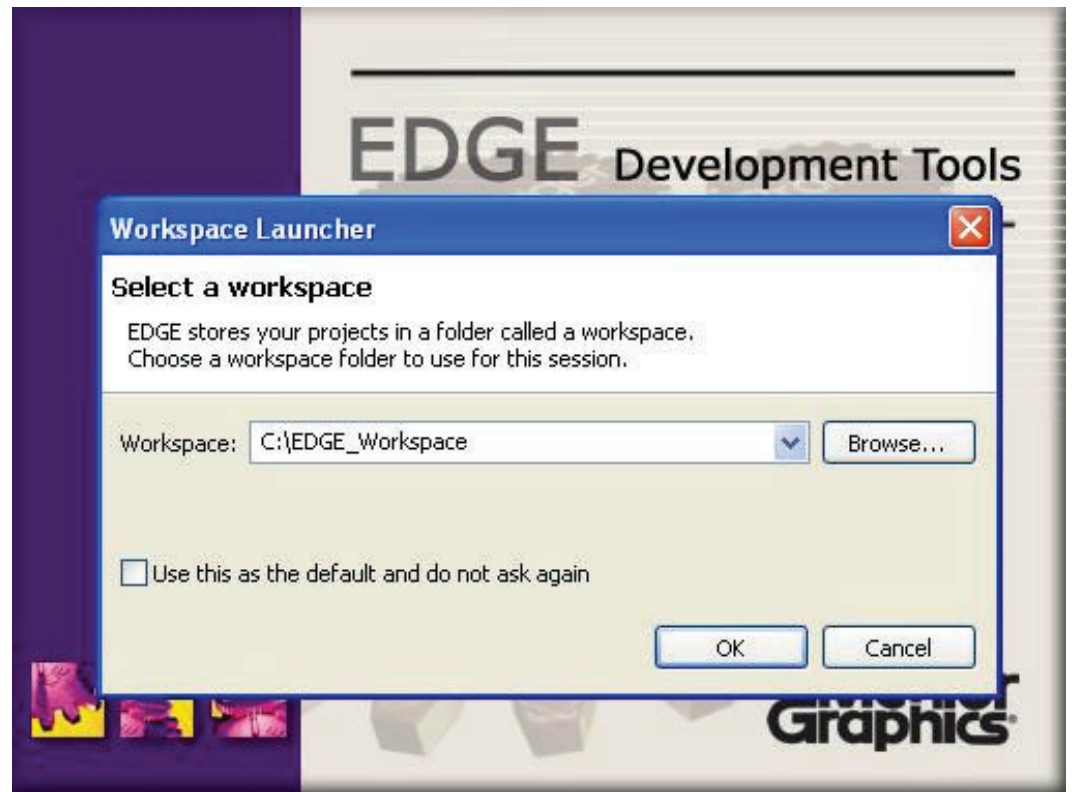


Figure 25: Launching EDGE and Specifying its Workspace

Import the plus_demo project to the EDGE Workspace as illustrated by [Figure 26](#) and [Figure 27](#).

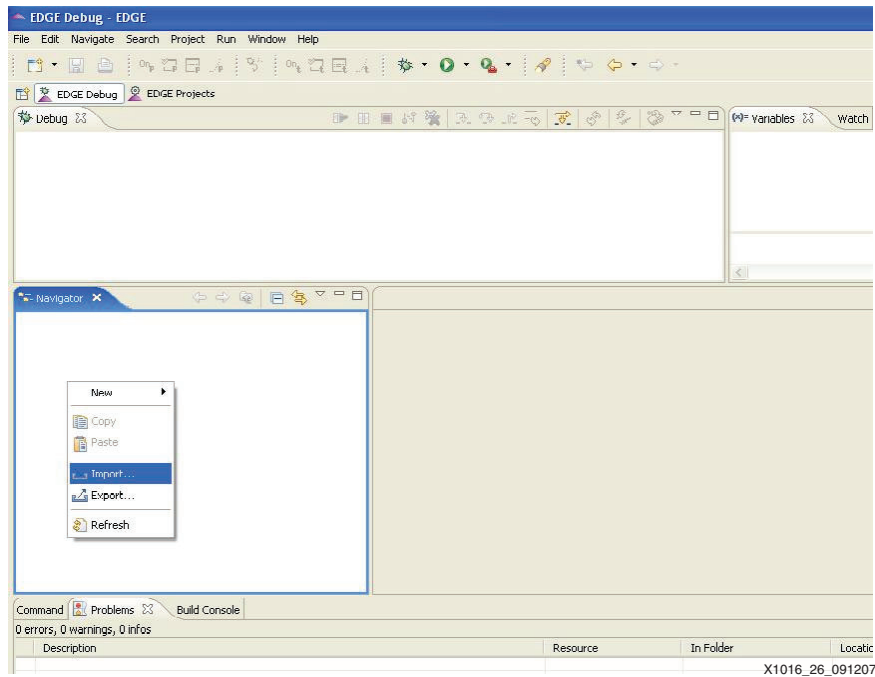


Figure 26: Importing an existing EDGE Project into the Workspace – Step-1

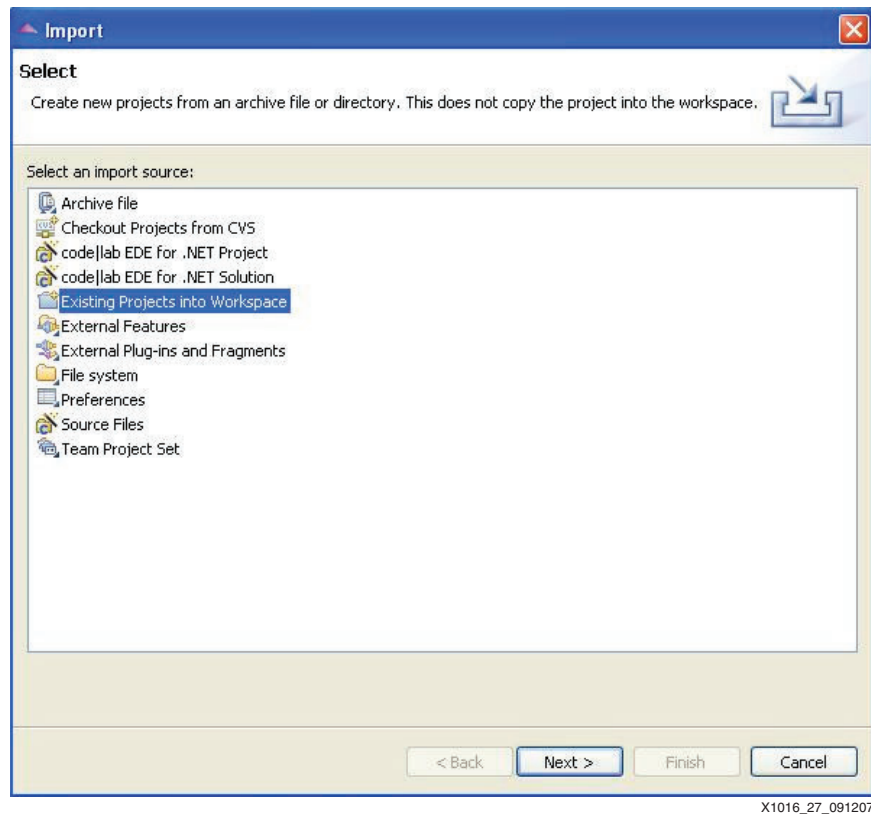


Figure 27: Importing an Existing EDGE Project into the Workspace – Step-2

Specify the `plus_demo` project directory to import into the Workspace as shown in Figure 28 and Figure 29.

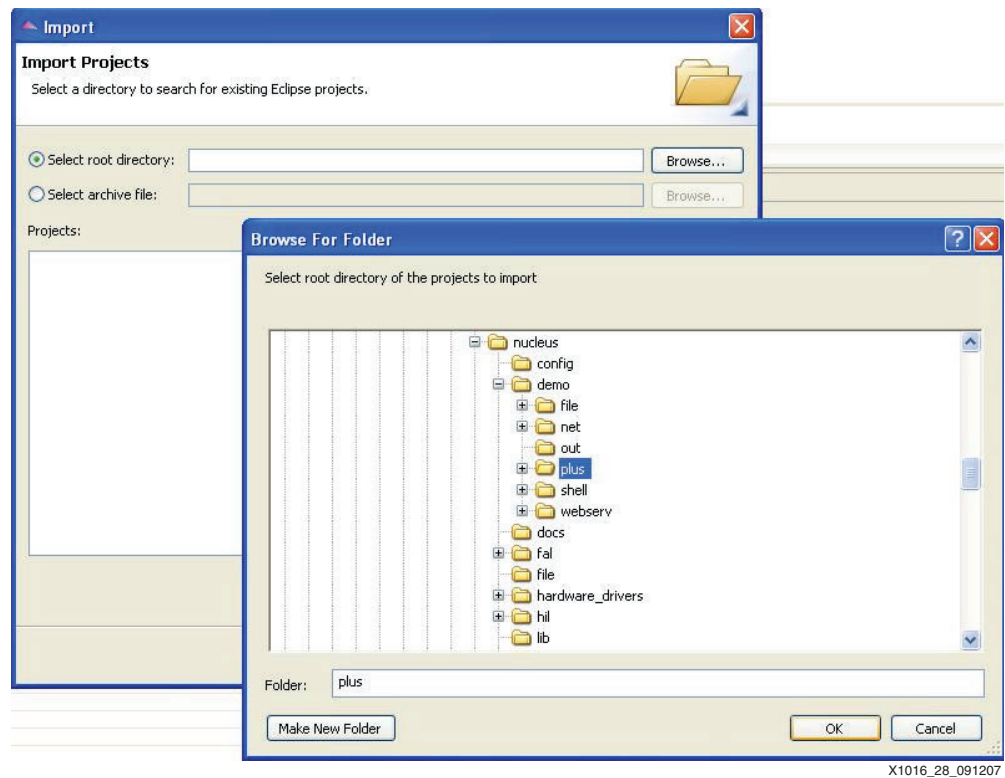


Figure 28: Specifying the `plus_demo` Project Directory – Step-1

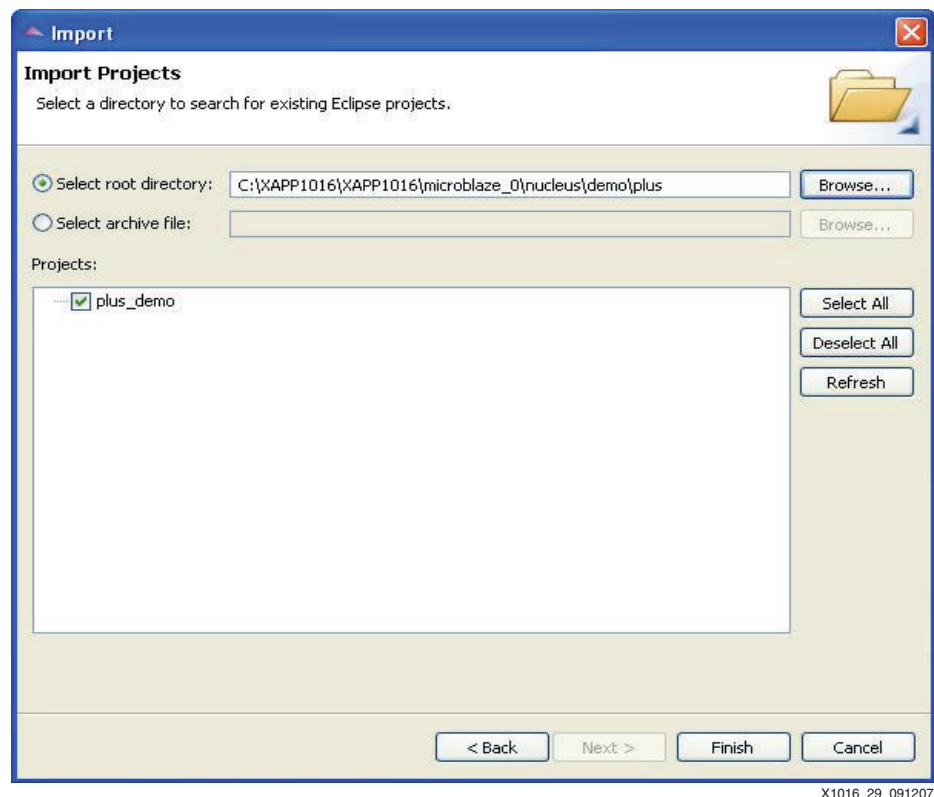
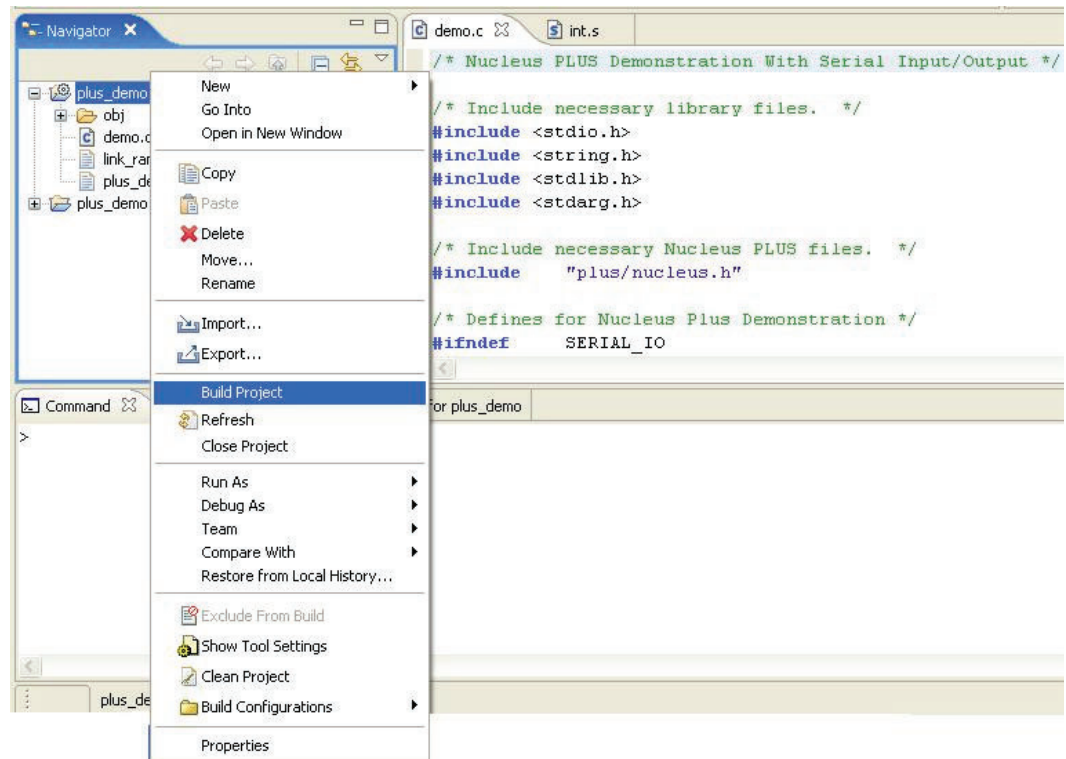


Figure 29: Specifying the `plus_demo` Project Directory – Step-2

Click **Finish**.

At this stage the Project can be built in EDGE as shown by [Figure 30](#).



X1016_30_091207

Figure 30: Building the plus_demo Project in EDGE

Downloading Nucleus PLUS Applications to the HW target using XMD

At this stage the Nucleus PLUS SW image can be downloaded to the HW target to test the system functionality.

XMD (Xilinx Microprocessor Debugger) can be used to download the SW image to the SDRAM DDR memory on the Spartan-3E Starter Kit.

XMD is a tool that facilitates program debugging and system verification using the PowerPC™ 405GP (Virtex-II Pro & Virtex-4) or MicroBlaze microprocessors. Use XMD to debug programs running on a hardware board, Cycle-Accurate Instruction Set Simulator (ISS), or on a MicroBlaze Cycle-Accurate Virtual Platform (VP) system.

XMD provides a Tool Command Language (Tcl) interface which can be used for command line control and debugging of the target, as well as for running complex verification test scripts to directly test a complete system.

XMD supports GNU Debugger (GDB) Remote TCP protocol to control debugging of a target.

Some graphical debuggers use this interface for debugging, including PowerPC and MicroBlaze GDB (powerpc-eabi-gdb and mb-gdb), the Platform Studio Software Development Kit (SDK), and EDK Eclipse-based Software IDE.

Mentor Graphics EDGE Debugger also uses this protocol.

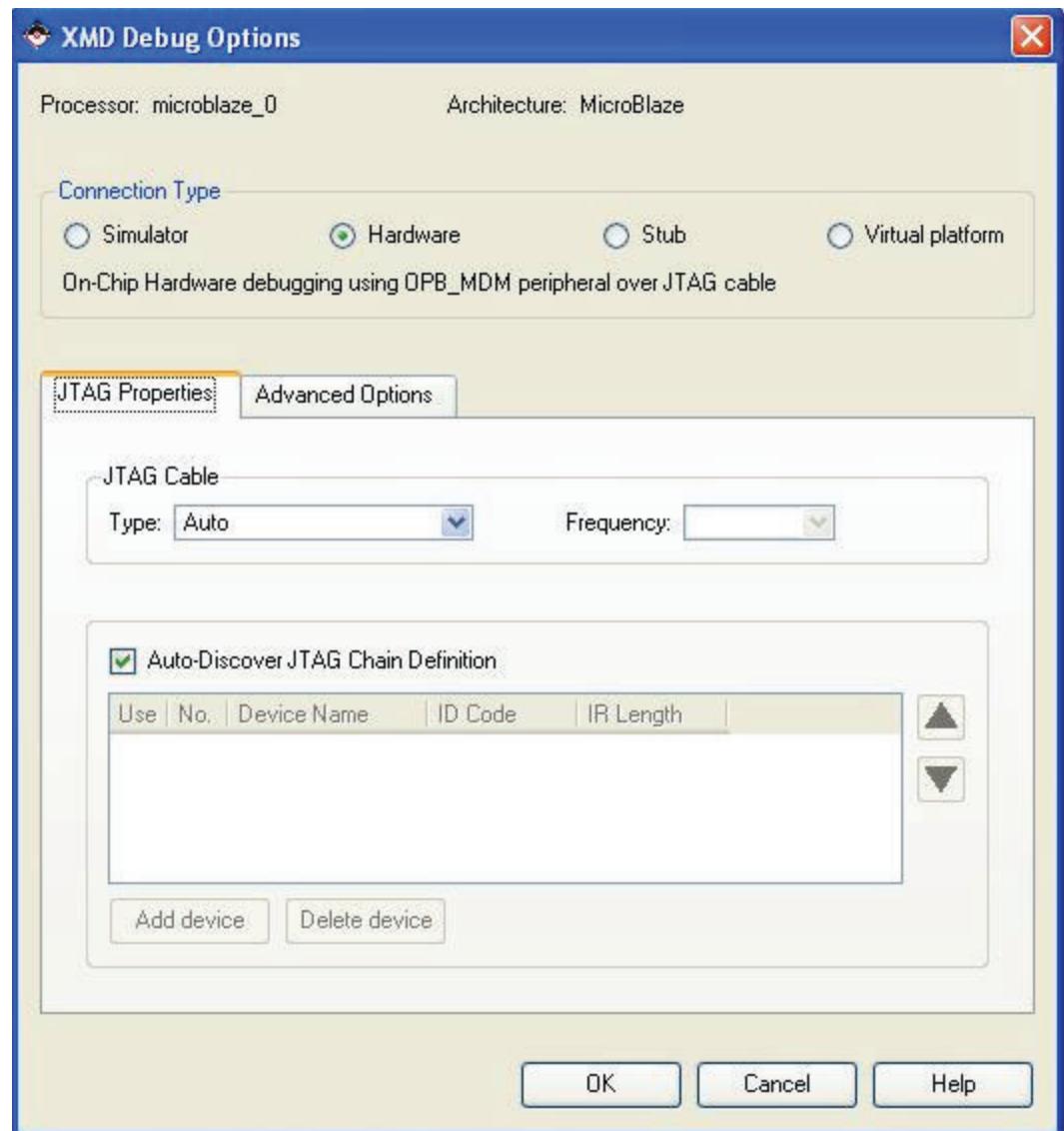
In either case, the debugger connects to XMD running on the same computer or on a remote computer on the network.

The FPGA has been previously configured with the Embedded System designed earlier in XPS.

To connect to the HW Target, the XMD Debug Options must be set up first.

From XPS menu, click **Debug** → **XMD Debug Options**. This will open up the XMD Debug Options setup Window.

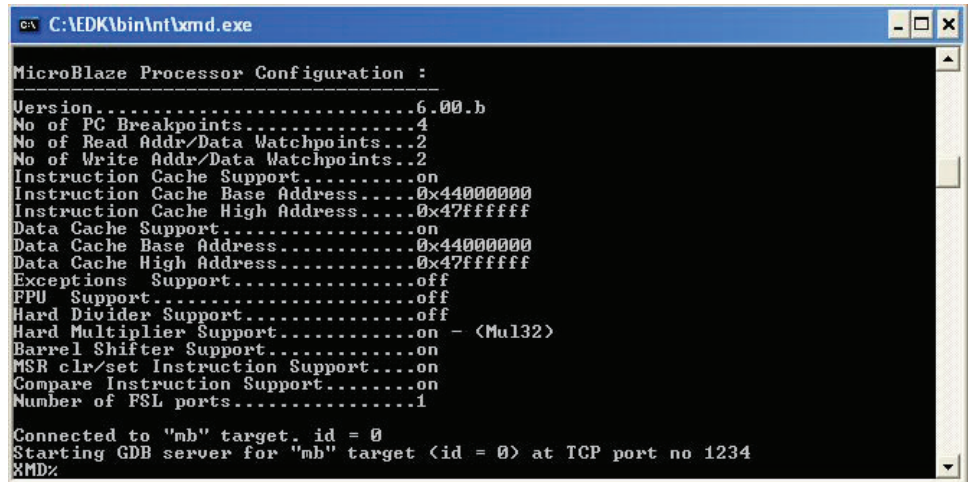
Select **Hardware** as the Connector Type as shown by [Figure 31](#), then click **OK**.



X1016_31_091207

Figure 31: Selecting the Hardware as an XMD connection Target

Launch XMD from XPS to connect to the MicroBlaze processor over the OPB MDM JTAG interface as shown by Figure 32.



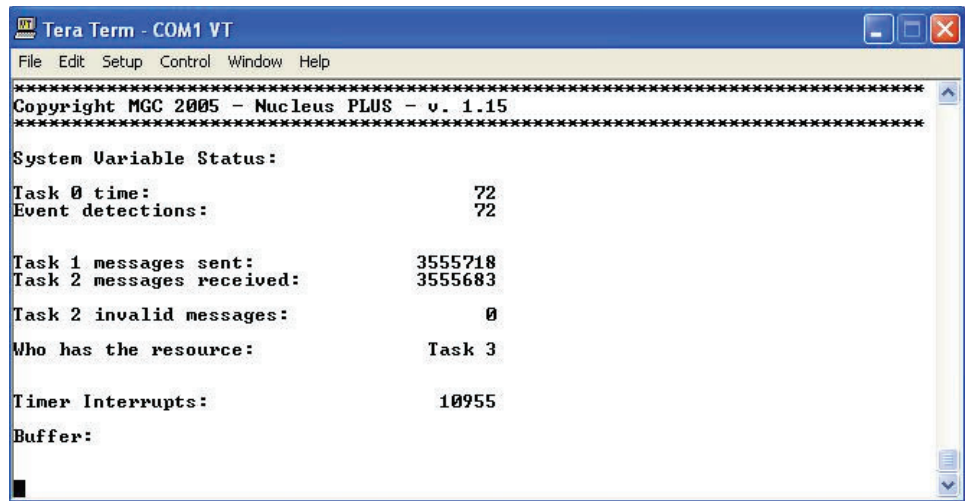
X1016_32_091207

Figure 32: XMD connection to the Hardware Board

Use the XMD dow command to download the plus_demo.out executable file to the Target. Use command dow microblaze_0\nucleus\demo\out\plus_demo.out.

Start executing the SW on the target from XMD by using the con XMD command.

At this stage, the outputs on the Terminal Window which had been opened previously, appear as shown in Figure 33.



X1016_33_091207

Figure 33: Output of plus_demo to the UART Console

Stop executing the SW on the target from XMD by using the stop XMD command.

Exit XMD by using the exit command.

Debugging Nucleus PLUS Applications using EDGE

Connecting to the HW Board

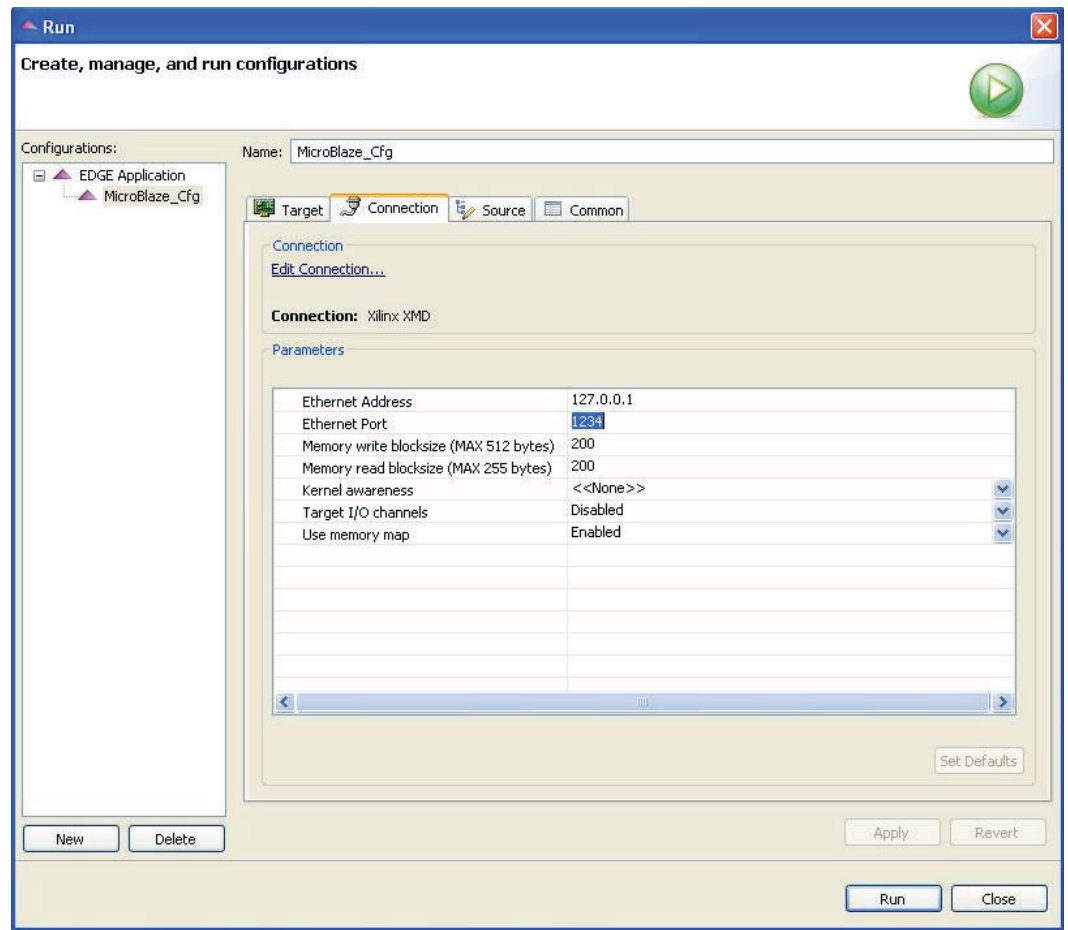
To debug the Application using EDGE Debugger, launch XMD to connect to the hardware board from XPS menu, Click on **Debug Launch XMD** as detailed above.

Note: The provided TCP Port 1234 by XMD must be used within EDGE Debugger Settings to connect to XMD.

Connecting EDGE Debugger to the target

In EDGE menu, click on **Run Run** to open up the MicroBlaze Target Settings.

Select the Connection Tab. Confirm that Ethernet Value is set to 1234, the TCP Port number returned by XMD at the Board connection stage. [Figure 34](#) illustrates this step.



X1016_34_091207

Figure 34: Launching EDGE Debugger

At this stage the SW application can be debugged in EDGE debugger.

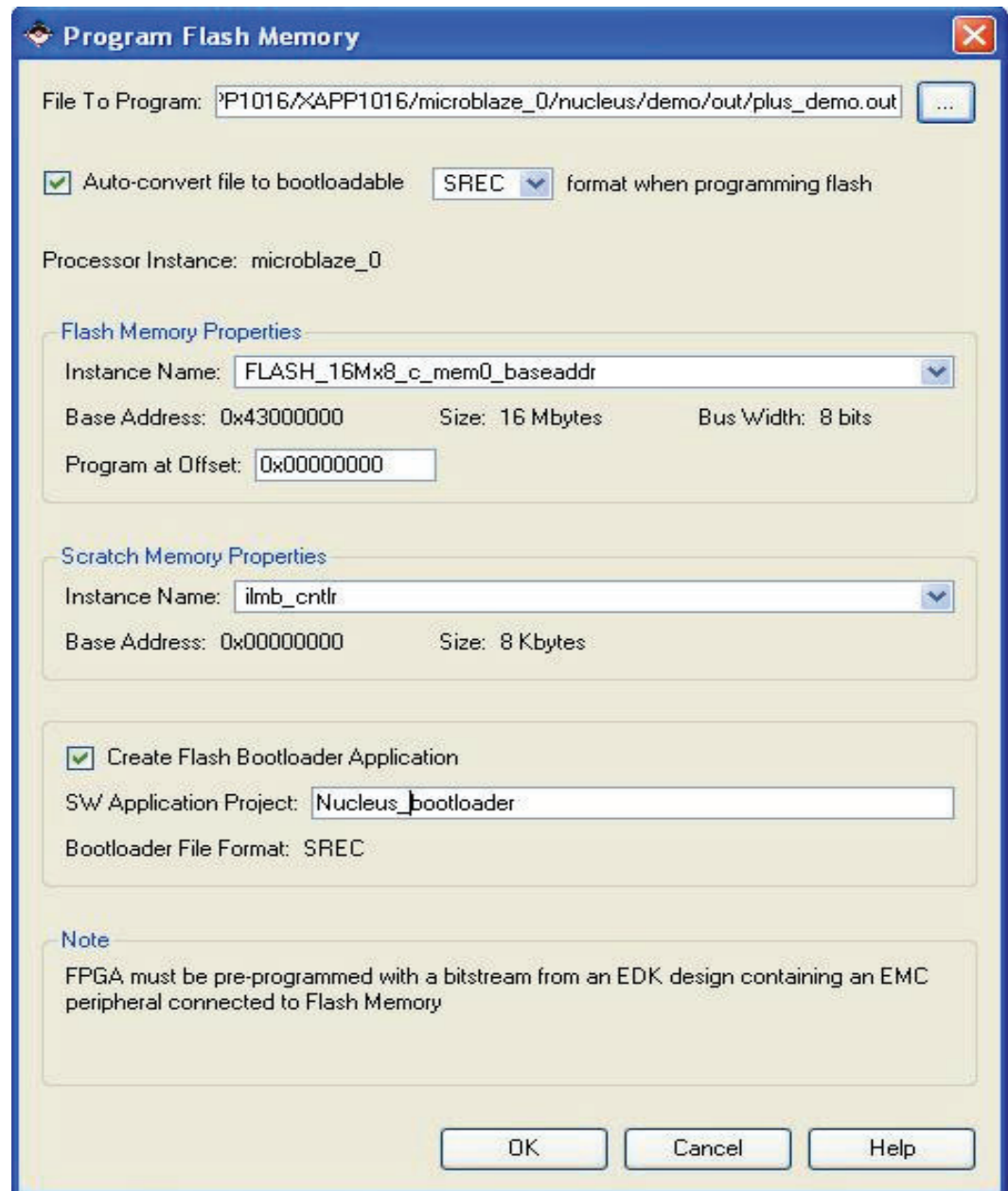
Adding a Bootloader from Flash memory to DDR memory to the XPS Design

Generating a bootloadable image of the Nucleus PLUS SW application in XPS

XPS can be used to generate a bootable S-Records Image of the Nucleus PLUS SW application using the XPS Flash Writer utility.

Before starting this step, the Board should be configured and XMD not connected to the target.

1. From XPS menu, select **Device Configuration** → **Program Flash Memory**
2. From the Program Flash Memory wizard, browse to the location of the Nucleus PLUS SW Image previously built by EDGE Tools as shown by [Figure 35](#).

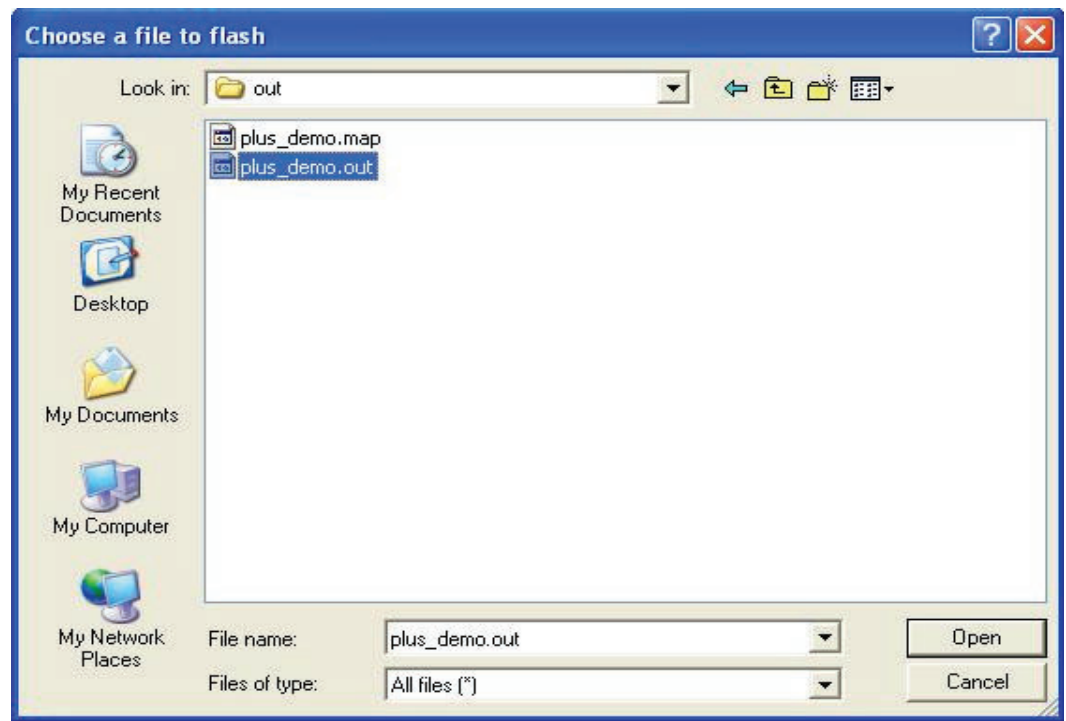


X1016_35_091207

Figure 35: Launching XPS Flash Writer Wizard

3. Browse to the Nucleus PLUS **plus_demo.out** location under the folder listed below.
[\\microblaze_0\nucleus\demo\out\plus_demo.out](#)

Note: Specify **All Files** in the Files of type field to be able to specify an *.out file to the S-Records generation utility as shown by [Figure 36](#).



X1016_36_091207

Figure 36: Specifying the Nucleus PLUS SW Image to Convert into S-Records

4. Click **Open**.
5. Check **Auto-convert file to bootloadable** and select **SREC** format when programming flash in the wizard as shown in [Figure 35](#).

Programming the Flash with the bootloadable S-Records Image of the Nucleus PLUS SW application from XPS

In the Programming Flash wizard shown in [Figure 35](#), the location of where the S-Records Image of the Nucleus PLUS SW Application should be programmed can be changed, if necessary.

Under Scratch Memory Properties, select **ilmb_cntlr** in the Instance Name field.

At this stage, a Bootloader SW Application can be created by the Programming Flash wizard by selecting the Create Flash Bootloader Application.

The name of the Bootloader SW Application can be changed to **Nucleus_Bootloader**, for example, as shown by [Figure 35](#).

Click **OK**, to automatically accomplish the following procedures.

1. Convert the SW Image `plus_demo.out` file to a bootable S-Records file, `plus_demo.out.srec`.
2. Create a Bootloader SW Application in XPS to bootload the `plus_demo.out.srec` file from flash to SDRAM DDR Memory.
3. Program the Flash with the `plus_demo.out.srec` file.

Customizing the Bootloader and building it in XPS

As selected earlier in the Flash Programming Interface (Figure 35), the “Nucleus_Bootloader” SW application has been automatically added to XPS SW Projects. This is a simple Bootloader that converts the S-Records SW Image into its executable format while copying it from the Flash memory to the SDRAM DDR memory (to where the executable image was originally built to run from).

The `bootloader.c` source file can be edited to customize the Bootloader application.

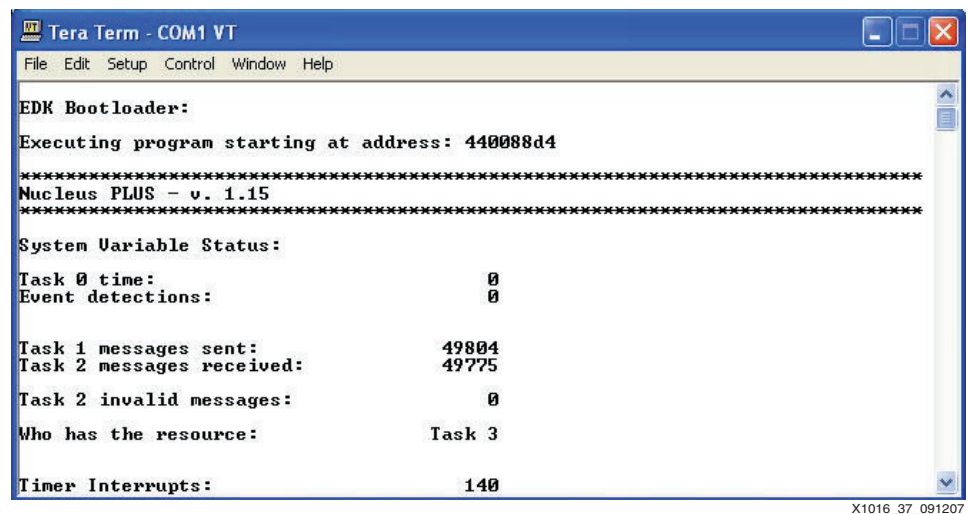
1. In XPS Applications Tab, expand the sources under the Nucleus_Bootloader SW Project.
2. Open up the `bootloader.c` source file, and go to line 45, then uncomment the file named `// #define DISPLAY_PROGRESS`
3. Save and Close the file.

This will speed up the boot from Flash to SDRAM DDR memory process by disabling the Bootloader displaying every S-Records line conversion.

1. In XPS, select **Software** → **Software Platform Settings**.
2. Choose Standalone as the OS.
3. Select the OS and Libraries Tab, and make sure that the “stdin” and “stdout” are set to RS232_DCE,
4. Click OK to save the settings in the MSS file.
5. Build the “Nucleus_Bootloader” SW Application in XPS.
6. Make sure that no other SW Application is selected to initialize the BRAMs.
7. Select the “Nucleus_Bootloader” SW Application to initialize the BRAMs.

At this stage the “Nucleus_Bootloader” Application is ready to boot the Nucleus SW image from Flash to SDRAM DDR Memory.

1. In XPS, select **Device Configuration** → **Download Bitstream**, the Bootloader is launched once the FPGA is reconfigured as shown by Figure 37.



```

Tera Term - COM1 VT
File Edit Setup Control Window Help

EDK Bootloader:
Executing program starting at address: 440088d4
*****
Nucleus PLUS - v. 1.15
*****
System Variable Status:
Task 0 time:                0
Event detections:          0

Task 1 messages sent:      49804
Task 2 messages received: 49775
Task 2 invalid messages:  0
Who has the resource:      Task 3
Timer Interrupts:         140
  
```

Figure 37: Bootloading the Nucleus SW Application from Flash to DDR Memory

Conclusion

This application note describes how to use XPS Base System Builder (BSB) to build a hardware platform capable of running a Nucleus PLUS RTOS based application. It describes the steps needed to use EDGE Tools in order to build the SW application and debug it. It provides a way to load the Nucleus PLUS SW application to the target board using XPS XMD interface. It provides an example Bootloader for the Nucleus PLUS SW application and the required steps to program the Flash memory with a bootloadable S-Records image of the SW Application.

This application note can be ported to any HW board capable of hosting a MicroBlaze based embedded system.

References

For more details on Nucleus PLUS characteristics, components, and architecture, see the Mentor Graphics web site at www.mentor.com.

http://www.mentor.com/products/embedded_software/nucleus_rtos/index.cfm

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
9/13/07	1.0	Initial Xilinx release.