

1 Tutorial: Creating a Complex FIR Filter on a Virtex-5 FX Platform (EDK 10.1)



Overview

This detailed tutorial will demonstrate how to use Impulse C to create, compile and optimize a digital signal processing (DSP) example for the **PowerPC** platform. We will also show how to make use of the **Auxiliary Processor Unit (APU)** and **Fabric Co-processor Bus (FCB)** provided in the PowerPC platform.

The goal of this application will be to compile the algorithm (a **Complex FIR Filter** function) as hardware on the FPGA. The PowerPC will be used to run test code (producer and consumer processes) that will pass text data into the algorithm and accept the results.

This example makes use of the **Xilinx ML507 Evaluation Platform**. The board features a **Virtex-5 FXT FPGA** with a **PowerPC 440** embedded processor core. This tutorial also assumes you are using the **Xilinx EDK 10.1i** (or later) development tools.

This tutorial will require approximately one hour to complete, including software run times.

*Note: this tutorial is based on a sample DSP application developed by Bruce Karsten of Xilinx, Inc. A more complete description of the algorithm can be found in the **Impulse C User Guide**, in the Getting Started Tutorial #2. This tutorial assumes that you have are familiar with the basic steps involved in using the Xilinx EDK tools. For brevity this tutorial will omit some EDK details that are covered in introductory EDK and Impulse C tutorials.*

Note also that most of the detailed steps in this tutorial only need to be performed once, during the initial creation of your PowerPC application. Subsequent changes to the application do not require repeating these steps.

Steps

- [Loading the Complex FIR Application](#)
- [Understanding the Complex FIR Application](#)
- [Compiling the Application for Simulation](#)
- [Building the Application for the Target Platform](#)
- [Creating the Platform Using the Xilinx Tools](#)
- [Configuring the New Platform](#)
- [Exporting Files from CoDeveloper](#)
- [Importing the Generated Hardware](#)
- [Generating the FPGA Bitmap](#)
- [Importing the Application Software](#)
- [Running the Application](#)

1.1 Loading the Complex FIR Filter Application

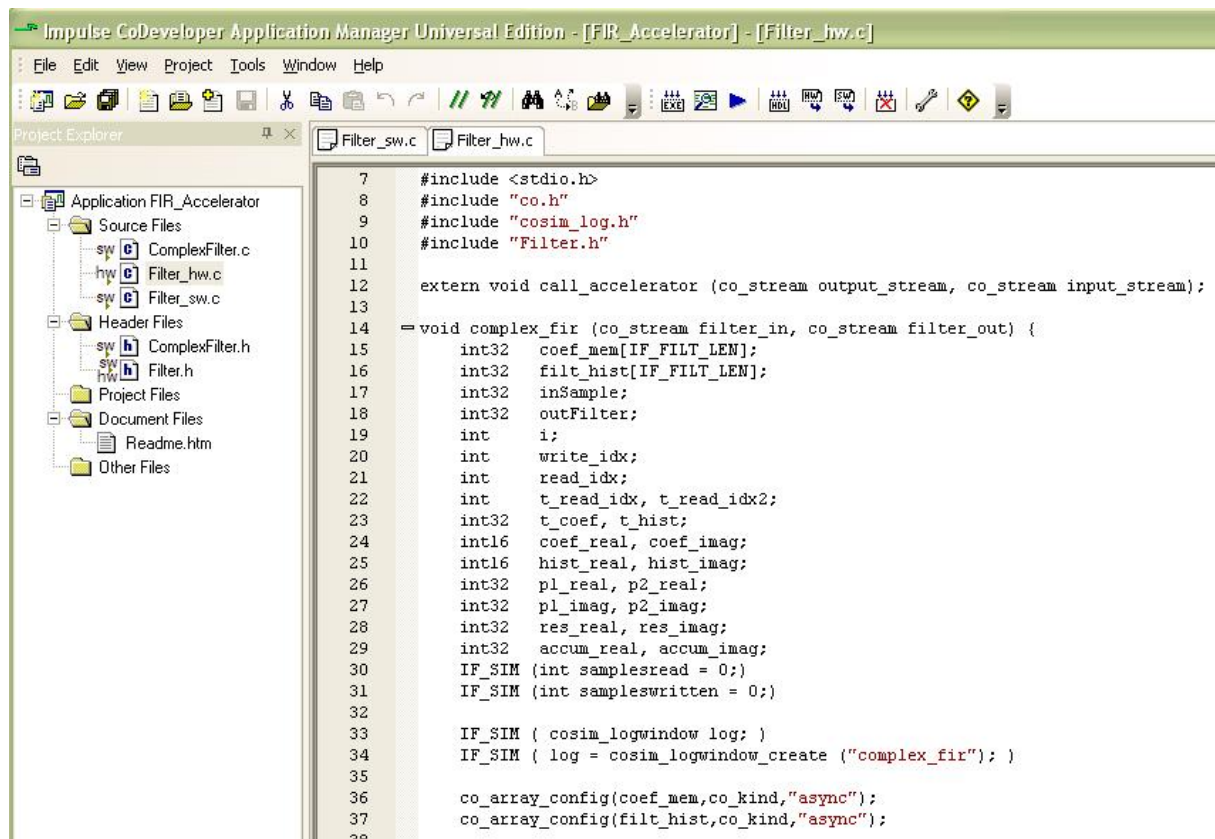
Complex FIR Filter Tutorial for PowerPC, Step 1

To begin, start the **CoDeveloper** Application Manager:

Start -> Programs -> Impulse Accelerated Technologies -> CoDeveloper -> CoDeveloper Application Manager

*Note: this tutorial assumes that you have already read and understand the Complex FIR example and tutorial presented in the main **CoDeveloper** help file.*

Open the **Xilinx PowerPC ComplexFIR** sample project by selecting **Open Project** from the **File** menu, or by clicking the **Open Project** toolbar button. Navigate to the `.\Examples\Embedded\ComplexFIR_Xilinx\` directory within your CoDeveloper installation. (You may wish to copy this example to an alternate directory before beginning.) The project file is also available online at <http://impulsec.com/ReadyToRun/>. Opening the project will result in the display of a window similar to the following:



Files included in the **Complex FIR** project include:

Source files ComplexFilter.c, Filter_hw.c and Filter_sw.c - These source files represent the complete application, including the **main()** function, consumer and producer software processes and a single hardware process.

See Also

[Understanding the Complex FIR Application](#)

1.2 Understanding the Complex FIR Filter Application

Complex FIR Filter Tutorial for PowerPC, Step 2

Before compiling the Complex FIR application to hardware, let's first take a moment to understand its basic operation and the contents of its primary source files, and in particular *Filter_hw.c*.

The specific process that we will be compiling to hardware is represented by the following function (located in *Filter_hw.c*):

```
void complex_fir(co_stream filter_in, co_stream filter_out)
```

This function reads two types of data:

- Filter coefficients used in the Complex FIR convolution algorithm.
- An incoming data stream

The results of the convolution are written by the process to the stream **filter_out**.

The **complex_fir** function begins by reading the coefficients from the **filter_in** stream and storing the resulting data into a local array (**coef_mem**). The function then reads and begins processing the data, one at a time. Results are written to the output stream **filter_out**.

The repetitive operations described in the **complex_fir** function are complex convolution algorithm.

The complete test application includes test routines (including **main**) that run on the PowerPC processor, generating test data and verifying the results against the legacy C algorithm from which **complex_fir** was adapted.

The configuration that ties these modules together appears toward the end of the **Filter_hw.c** file, and reads as follows:

```
void config_filt (void *arg) {
    int i;

    co_stream  to_filt, from_filt;
    co_process cpu_proc, filter_proc;

    to_filt    = co_stream_create ("to_filt",    INT_TYPE(32), 4);
    from_filt  = co_stream_create ("from_filt",  INT_TYPE(32), 4);

    cpu_proc   = co_process_create ("cpu_proc",   (co_function)
call_accelerator, 2, to_filt, from_filt);
    filter_proc = co_process_create ("filter_proc", (co_function)
complex_fir,      2, to_filt, from_filt);

    co_process_config (filter_proc, co_loc, "PE0");
}
```

This configuration function describes the connectivity between instances of each previously defined process.

Only one process in this example (**filter_proc**) will be mapped onto hardware and compiled by the Impulse C compiler. This process (**filter_proc**) is flagged as a hardware process through the use of the **co_process_config** function, which appears here at the last statement in the configuration function. **Co_process_config** instructs the compiler to generate hardware for **complex_fir** (or more accurately, the *instance* of **complex_fir** that has been declared here as **filter_proc**).

The **ComplexFilter.c** generates a set of complex FIR coefficients and also a group of input data being processed.

The **Filter_sw.c** will run in the PowerPC embedded processor, controlling the stream flow and printing results.

See Also

[Compiling the Application for Simulation](#)

1.3 Compiling the Application for Simulation

Complex FIR Filter Tutorial for PowerPC, Step 3

Simulation allows you to verify the correct operation and functional behavior of your algorithm before attempting to generate hardware for the FPGA. When using Impulse C, simulation simply refers to the process of compiling your C code to the desktop (host) development system using a standard C compiler, in this case the gcc compiler included with the Impulse CoDeveloper tools.

To compile and simulate the application for the purpose of functional verification:

1. Select **Project -> Build Simulation Executable** (or click the **Build Simulation Executable** button) to build the **ComplexFIR.exe** executable. The **Build** console window will display the compile and link messages as shown below:

```
Build
===== Building target 'build_exe' in file _Makefile =====
"C:/Impulse/CoDeveloper3_30/MinGW/bin/gcc" -g "I:/Impulse/CoDeveloper3_30/Include" "I:/Impulse/CoDeveloper3_30/StageMaster/include" -DWIN32
"I:/Impulse/CoDeveloper3_30/MinGW/include" -o ComplexFilter.o -c ComplexFilter.c
"C:/Impulse/CoDeveloper3_30/MinGW/bin/gcc" -g "I:/Impulse/CoDeveloper3_30/Include" "I:/Impulse/CoDeveloper3_30/StageMaster/include" -DWIN32
"I:/Impulse/CoDeveloper3_30/MinGW/include" -o Filter_hw.o -c Filter_hw.c
"C:/Impulse/CoDeveloper3_30/MinGW/bin/gcc" -g "I:/Impulse/CoDeveloper3_30/Include" "I:/Impulse/CoDeveloper3_30/StageMaster/include" -DWIN32
"I:/Impulse/CoDeveloper3_30/MinGW/include" -o Filter_sw.o -c Filter_sw.c
"C:/Impulse/CoDeveloper3_30/MinGW/bin/gcc" -g ComplexFilter.o Filter_hw.o Filter_sw.o "C:/Impulse/CoDeveloper3_30/Libraries/ImpulseC.lib" -o FIR_Accelerator.exe
===== Build of target 'build_exe' complete =====
```

2. You now have a Windows executable representing the ComplexFIR application implemented as a desktop (console) software application. Run this executable by selecting **Project -> Launch Simulation Executable**. A command window will open and the simulation executable will run as shown below:

```

C:\WINDOWS\system32\cmd.exe
'D:\TestingExamples\ComplexFIR_PPC440\FIR_Accelerator.exe'
Begin Filtering a Slot
Complete Filtering a Slot
Begin Filtering a Slot
Complete Filtering a Slot
COMPLETE APPLICATION
Press Enter to continue...

```

Verify that the simulation produces the output shown. Note that although the messages indicate that the **ComplexFIR** algorithm is running on the FPGA, the application (represented by hardware and software processes) is actually running entirely in software as a compiled, native Windows executable. The messages you will see have been generated as a result of instrumenting the application with simple printf statements such as the following:

```

#ifdef IMPULSE_C_TARGET
    // Print Acceleration Numbers
    printf ("\r\n--> Acceleration factor: %dX\r\n\n", TimeSA/TimeHA);
    printf ("-----> Visit www.ImpulseC.com to learn more!");

    #if defined(XPAR_MICROBLAZE_ID)
        // Disable DCache
        microblaze_disable_dcache();
        microblaze_init_dcache_range(0, XPAR_MICROBLAZE_0_DCACHE_BYTE_SIZE);
        // Disable ICache
        microblaze_disable_icache();
        microblaze_init_icache_range(0, XPAR_MICROBLAZE_0_CACHE_BYTE_SIZE);

    #elif defined(XPAR_PPC440_VIRTEX5_ID)
        // Disable DCache
        XCache_DisableDCache();
        // Disable ICache
        XCache_DisableICache();
    #endif

#else
    printf ("COMPLETE APPLICATION\r\n");
    printf ("Press Enter to continue...\r\n");
    c = getc(stdin);
#endif

```

Notice in the above C source code that **#ifdef** statements have been used to allow the software side of the application to be compiled either for the embedded PowerPC processor, or to the host development system for simulation purposes.

See Also

[Building the Application for the Target Platform](#)

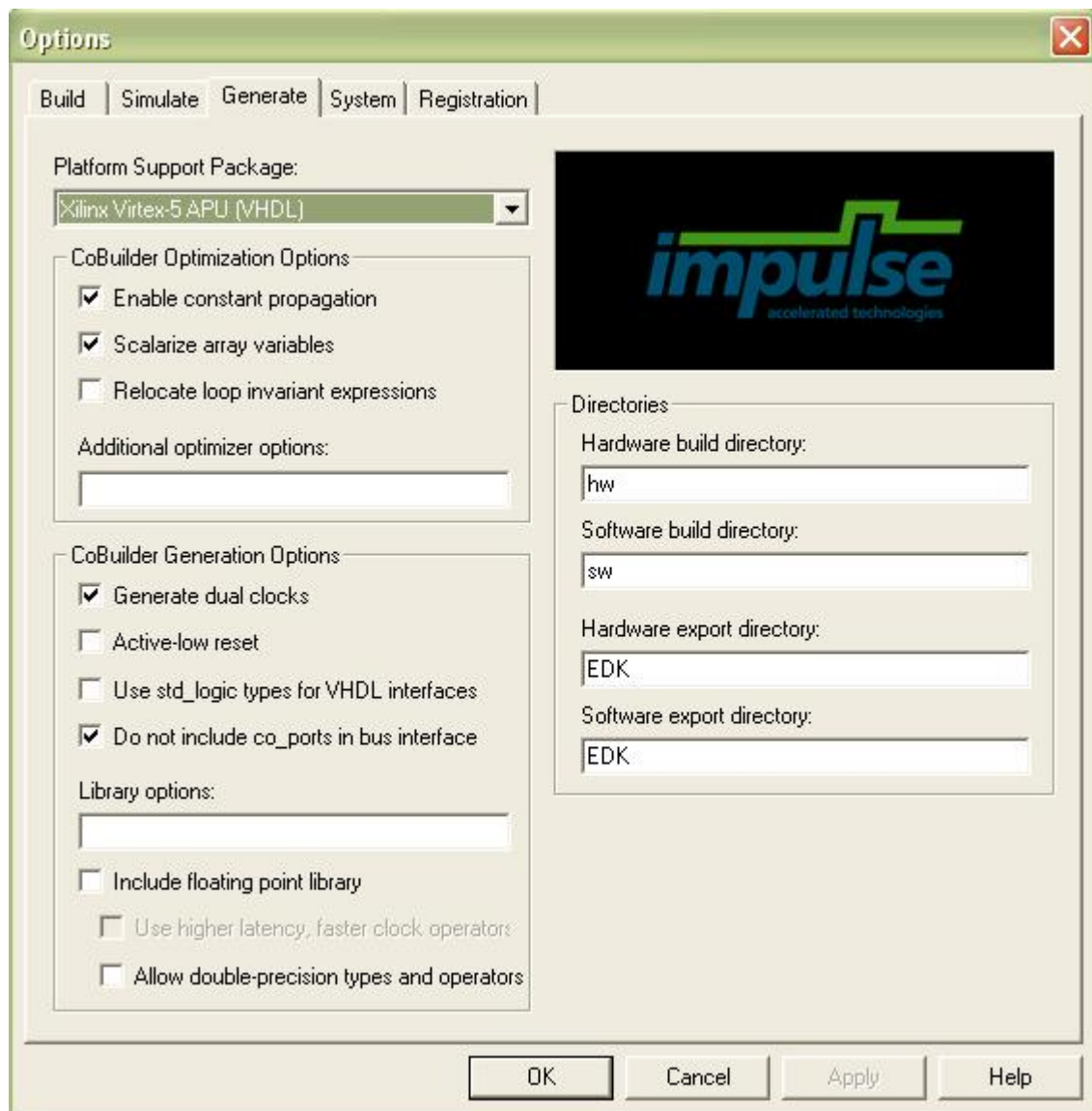
1.4 Building the Application for the Target Platform

Complex FIR Filter Tutorial for PowerPC, Step 4

The next step in the tutorial is to create FPGA hardware and related files from the C code found in the **Filter_hw.c** source file. This requires that we select a platform target, specify any needed options, and initiate the hardware compilation process.

Specifying the Platform Support Package

To specify a platform target, select from the menu **Project -> Options** to open the **Generate Options** dialog as shown below:



Specify **Xilinx Virtex-5 APU (VHDL)** as the **Platform Support Package**. Also specify **hw** and **sw** for the hardware and software directories as shown, and specify **EDK** for both the hardware and software export directories. Also ensure that the **Generate dual clocks** option is checked, which will allow the

generated hardware core to run at a different clock speed than the system bus speed on the FPGA.

Click **OK** to save the options and exit the dialog.

Generate HDL for the Hardware Process

To generate hardware in the form of HDL files, and to generate the associated software interfaces and library files, select from the menu **Project -> Generate HDL**. A series of processing steps will run in the **Build** console window as shown below:

```
Build
| Max. Unit Delay: 0
| Block #11:
| Stages: 1
| Max. Unit Delay: 0
|-----
| Operators:
| 10 Adder(s)/Subtractor(s) (32 bit)
| 4 Multiplier(s) (16 bit)
| 1 Comparator(s) (3 bit)
| 5 Comparator(s) (32 bit)
|-----
| Total Stages: 20
| Max. Unit Delay: 33
| Estimated DSPs: 4
|-----
Writing output ... done
"C:/Impulse/CoDeveloper3_30/bin/impulse_arch" "-aC:/Impulse/CoDeveloper3_30/Architectures/xilinx_v5_apu.xml" -dc -no_port_bus_connect -swdirsw -files
"FIR_Accelerator_comp.vhd FIR_Accelerator_top.vhd" FIR_Accelerator.xic hw/FIR_Accelerator_top.vhd
Impulse C HDL Design Generator
Copyright 2002-2007, Impulse Accelerated Technologies, Inc.
All rights reserved.
```

Note: the processing of this example may require a few minutes to complete, depending on the performance of your system.

When processing has completed you will have a number of resulting files created in the **hw** and **sw** subdirectories of your project directory.

See Also

[Exporting Files from CoDeveloper](#)

1.5 Exporting Files from CoDeveloper

Complex FIR Filter Tutorial for PowerPC, Step 5

Recall that in **Step 4** you specified the directory **EDK** as the export target for hardware and software. These export directories specify where the generated hardware and software processes are to be copied when the **Export Software** and **Export Hardware** features of **CoDeveloper** are invoked. Within these target directories (in this case **EDK**), the specific destination (which may be a subdirectory under **EDK**) for each file previously generated is determined from the **Platform Support Package** architecture library files. It is therefore important that the correct **Platform Support Package** (in this case **Xilinx Virtex-5 APU**) is selected prior to starting the export process.

To export the files from the build directories (in this case **hw** and **sw**) to the export directories (in this case the **EDK** directory), select **Project -> Export Generated Hardware (HDL)** and **Project -> Export Generated Software** from the menu. The Build console window will display some processing messages, as shown below:

Export the Hardware Files

```
Build
"C:/Impulse/CoDeveloper3_30/bin/impulse_export" -hardware -srcdirhw "aC:/Impulse/CoDeveloper3_30/Architectures/xilinx_v5_apu.xml" FIR_Accelerator.xic "EDK"Impulse C
Design Exporter
Copyright 2002-2007, Impulse Accelerated Technologies, Inc.
All rights reserved.
Loading C:/Impulse/CoDeveloper3_30/Architectures/xilinx_v5_apu.xml ...
Loading C:/Impulse/CoDeveloper3_30/Architectures/Xilinx/V5/PPC/APU/cpu.xml ...
Loading FIR_Accelerator.xic ...

===== Build of target 'export_hardware' complete =====
```

Export the Software Files

```
Build
chmod -R +rw sw
"C:/Impulse/CoDeveloper3_30/bin/impulse_export" -software -srcdirsw "aC:/Impulse/CoDeveloper3_30/Architectures/xilinx_v5_apu.xml" FIR_Accelerator.xic "EDK"
Impulse C Design Exporter
Copyright 2002-2007, Impulse Accelerated Technologies, Inc.
All rights reserved.Loading C:/Impulse/CoDeveloper3_30/Architectures/xilinx_v5_apu.xml ...
Loading C:/Impulse/CoDeveloper3_30/Architectures/Xilinx/V5/PPC/APU/cpu.xml ...
Loading FIR_Accelerator.xic ...

===== Build of target 'export_software' complete =====
```

*Note: you must select BOTH **Export Software** and **Export Hardware** before going onto the next step.*

You have now exported all necessary files from **CoDeveloper** to the Xilinx tools environment.

See Also

[Creating the Platform Using the Xilinx Tools](#)

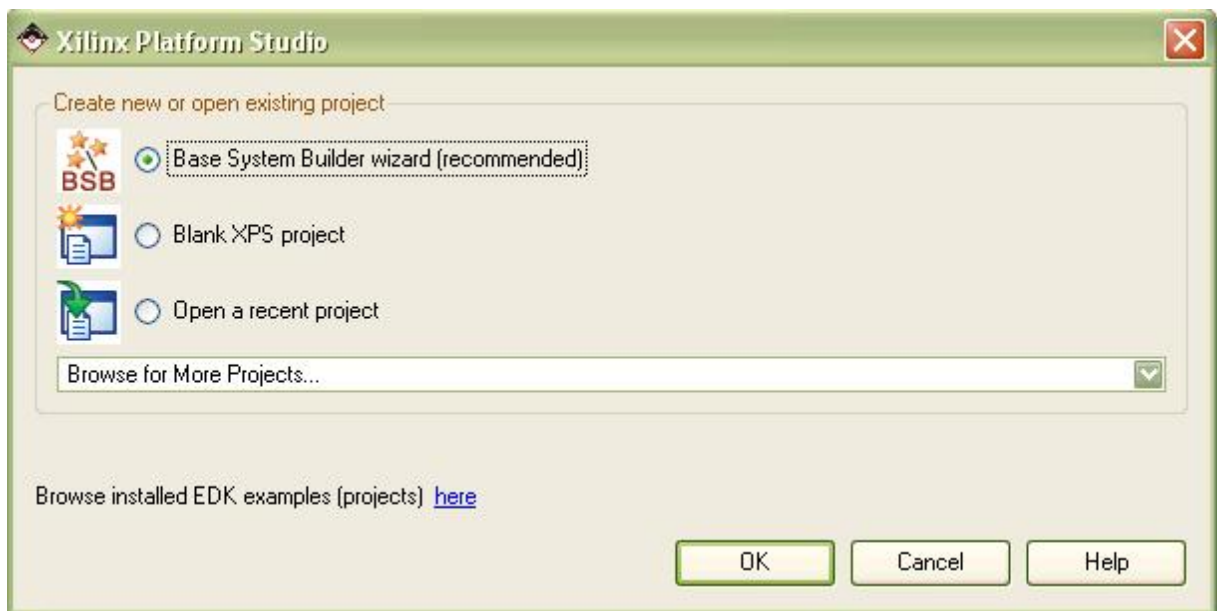
1.6 Creating a Platform Using Xilinx Tools

Complex FIR Filter Tutorial for PowerPC, Step 6

From the previous step, **CoDeveloper** creates a number of hardware and software-related output files that must all be used to create a complete hardware/software application on the target platform (in this case a **Xilinx FPGA** with an embedded **PowerPC** processor). This section will walk you through the file export/import process for this example, using the **Xilinx EDK System Builder**, **Xilinx Platform Studio**.

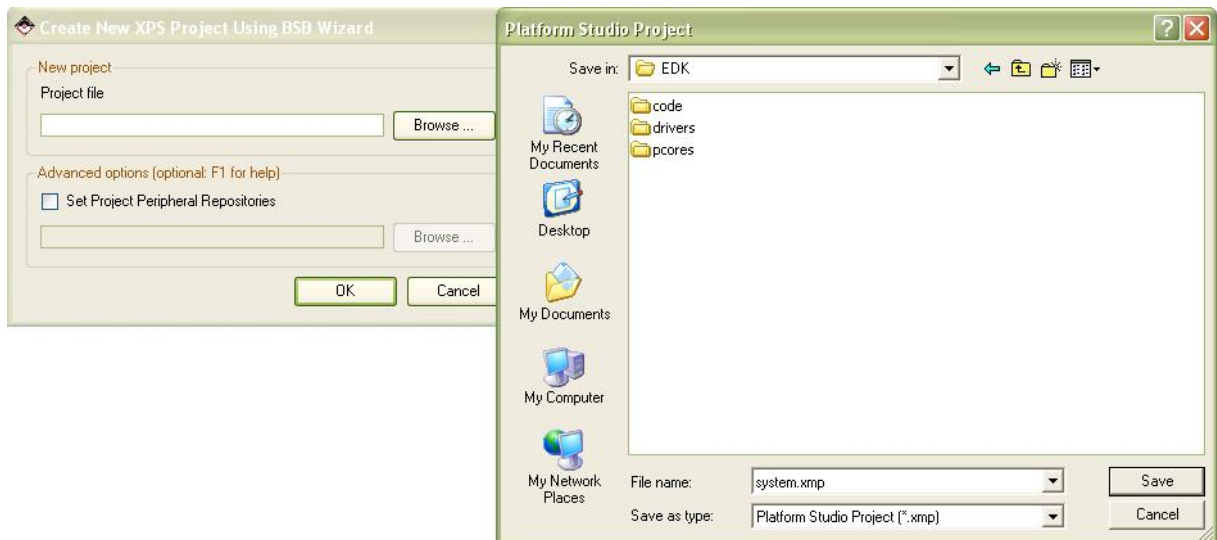
Creating a New Xilinx Platform Studio Project

Now we'll move into the Xilinx tool environment. Begin by launching **Xilinx Platform Studio** from the **Windows Start ->Xilinx ISE Design Suite 10.1 -> EDK -> Xilinx Platform Studio**. The **Xilinx Platform Studio** dialog appears as shown below:

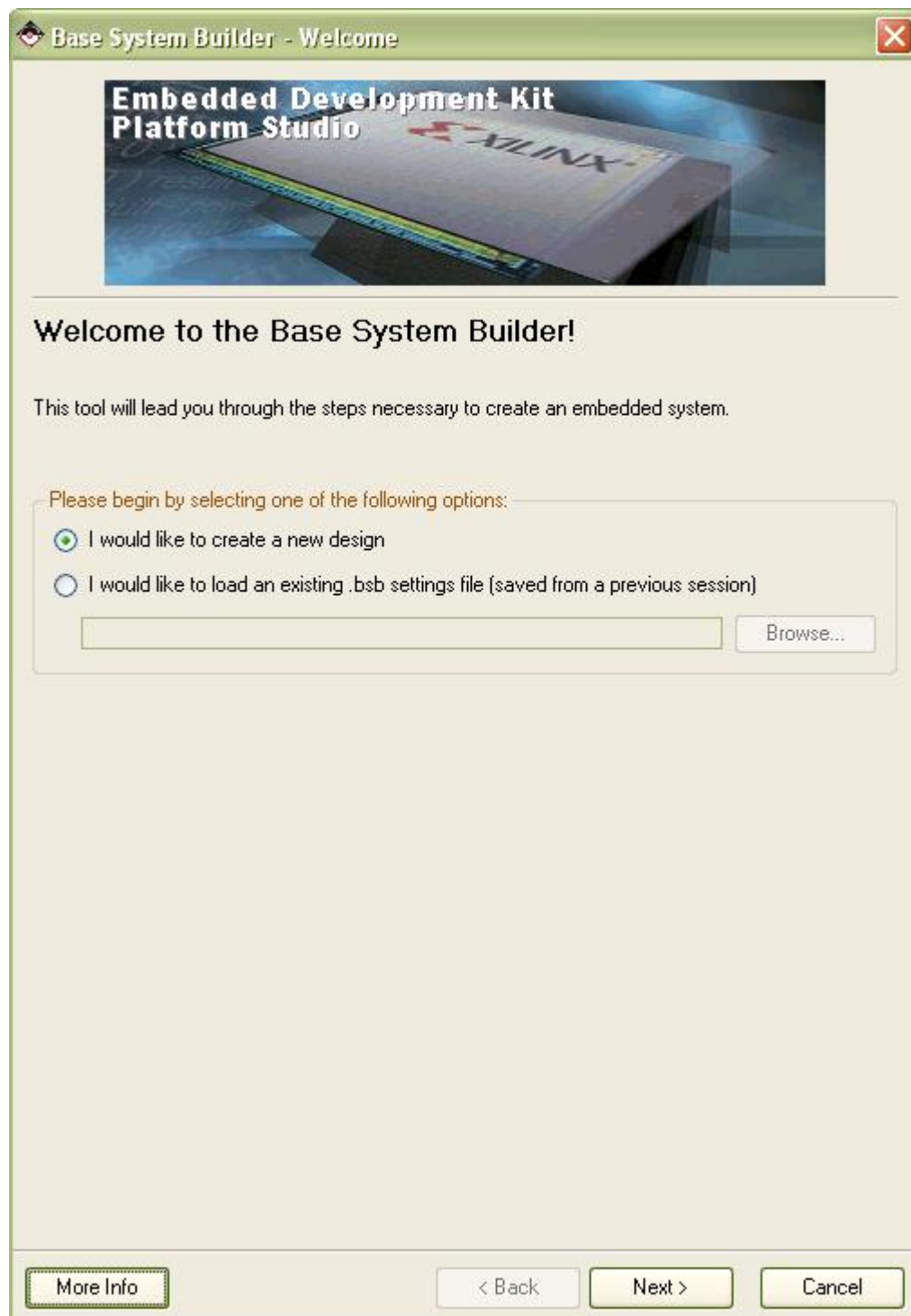


Select the **Base System Builder wizard (recommended)**, and click **OK**.

Next, in the **Create New XPS Project Using BSB Wizard** dialog, click **Browse** and navigate to the directory you created for your **Xilinx EDK** project files. For this tutorial we choose the directory name **EDK**, which is also the directory name we specified earlier in the **Generate Options** dialog. Click **Save** to create a project file called **system.xmp** (you can specify a different project name if desired):



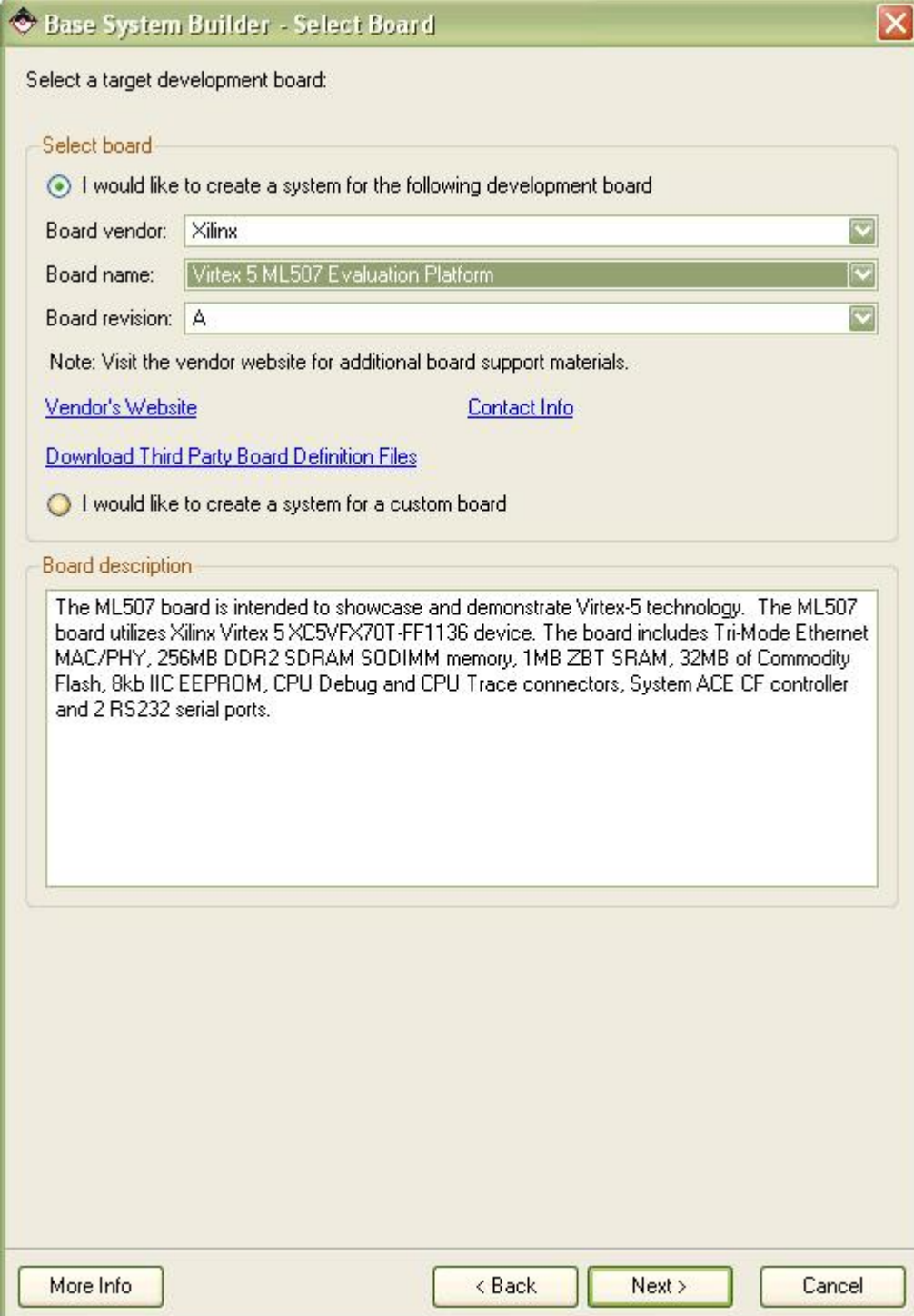
Now click **OK** in the **Create New XPS Project Using BSB Wizard** dialog. The **Base System Builder - Welcome** page will appear as shown below:



Select **I would like to create a new design** (the default), then click **Next** to choose your target board.

Choose your development board from the dropdown boxes. This example will use the following board (you should choose the reference board you have available for this step):

Board Vendor: Xilinx
Board Name: Virtex 5 ML507 Evaluation Platform

Board Revision: A

The image shows a screenshot of the 'Base System Builder - Select Board' dialog box. The dialog has a title bar with a close button. The main content area is divided into two sections: 'Select board' and 'Board description'. In the 'Select board' section, the first radio button is selected, indicating the user wants to create a system for a specific development board. Below this, there are three dropdown menus: 'Board vendor' (set to 'Xilinx'), 'Board name' (set to 'Virtex 5 ML507 Evaluation Platform'), and 'Board revision' (set to 'A'). A note below these fields says 'Note: Visit the vendor website for additional board support materials.' There are three hyperlinks: 'Vendor's Website', 'Contact Info', and 'Download Third Party Board Definition Files'. The second radio button, 'I would like to create a system for a custom board', is unselected. The 'Board description' section contains a text box with the following text: 'The ML507 board is intended to showcase and demonstrate Virtex-5 technology. The ML507 board utilizes Xilinx Virtex 5 XC5VFX70T-FF1136 device. The board includes Tri-Mode Ethernet MAC/PHY, 256MB DDR2 SDRAM SODIMM memory, 1MB ZBT SRAM, 32MB of Commodity Flash, 8kb IIC EEPROM, CPU Debug and CPU Trace connectors, System ACE CF controller and 2 RS232 serial ports.' At the bottom of the dialog, there are four buttons: 'More Info', '< Back', 'Next >', and 'Cancel'.

Select a target development board:

Select board

☒ I would like to create a system for the following development board

Board vendor: Xilinx

Board name: Virtex 5 ML507 Evaluation Platform

Board revision: A

Note: Visit the vendor website for additional board support materials.

[Vendor's Website](#) [Contact Info](#)

[Download Third Party Board Definition Files](#)

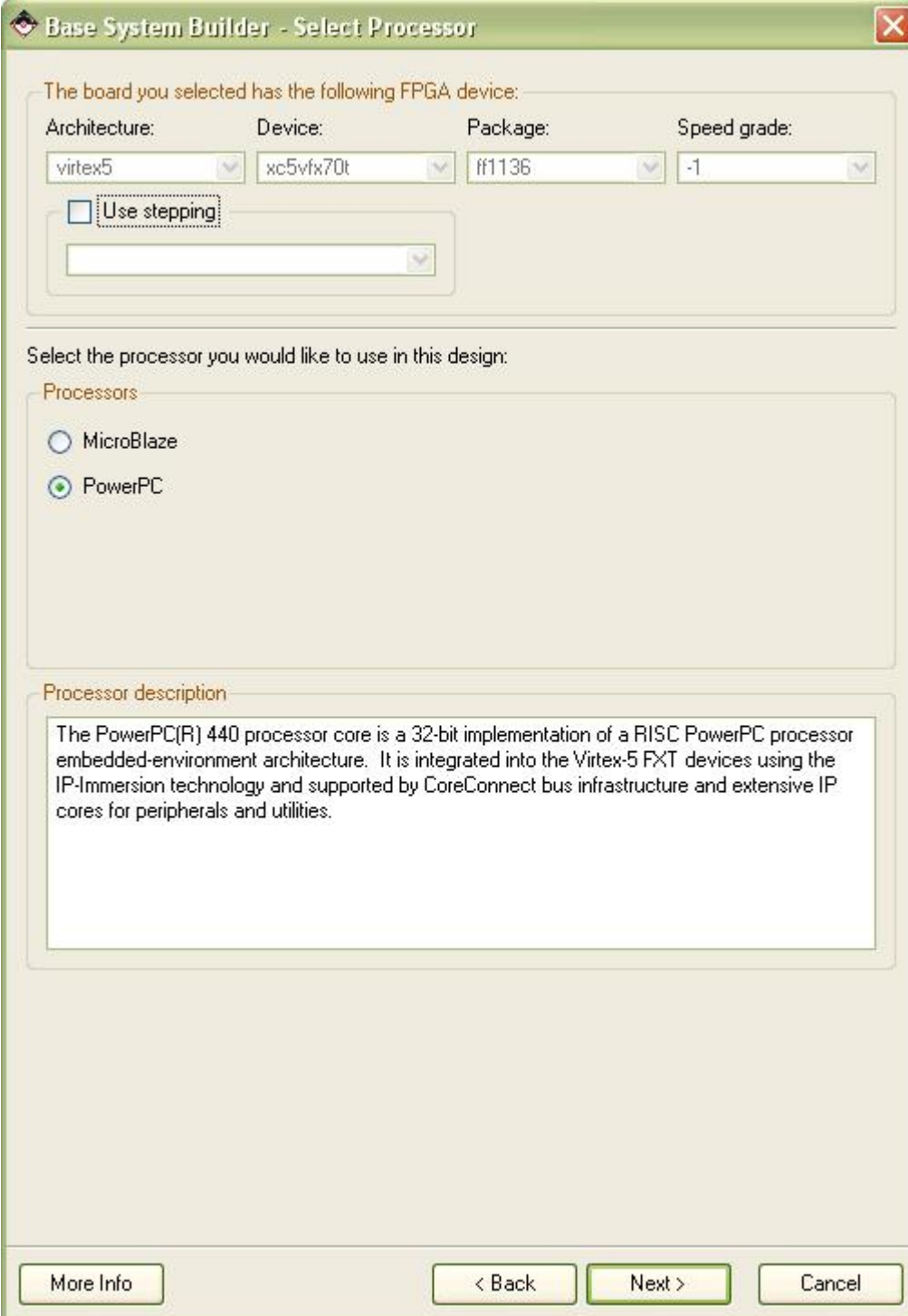
☐ I would like to create a system for a custom board

Board description

The ML507 board is intended to showcase and demonstrate Virtex-5 technology. The ML507 board utilizes Xilinx Virtex 5 XC5VFX70T-FF1136 device. The board includes Tri-Mode Ethernet MAC/PHY, 256MB DDR2 SDRAM SODIMM memory, 1MB ZBT SRAM, 32MB of Commodity Flash, 8kb IIC EEPROM, CPU Debug and CPU Trace connectors, System ACE CF controller and 2 RS232 serial ports.

More Info < Back Next > Cancel

Click **Next** to continue with the **Base System Builder Wizard**. In the next wizard page, make sure that **PowerPC** is selected as the processor:



The dialog box is titled "Base System Builder - Select Processor". It contains the following sections:

- FPGA Device Information:** A section titled "The board you selected has the following FPGA device:" containing four dropdown menus: Architecture (set to "virtex5"), Device (set to "xc5vfx70t"), Package (set to "ff1136"), and Speed grade (set to "-1"). Below these is a checkbox labeled "Use stepping" which is currently unchecked, followed by an empty dropdown menu.
- Processor Selection:** A section titled "Select the processor you would like to use in this design:" containing a sub-section "Processors" with two radio buttons: "MicroBlaze" (unselected) and "PowerPC" (selected).
- Processor Description:** A section titled "Processor description" containing a text box with the following text: "The PowerPC(R) 440 processor core is a 32-bit implementation of a RISC PowerPC processor embedded-environment architecture. It is integrated into the Virtex-5 FXT devices using the IP-Immersion technology and supported by CoreConnect bus infrastructure and extensive IP cores for peripherals and utilities."
- Navigation:** At the bottom, there are four buttons: "More Info", "< Back", "Next >" (which is highlighted with a green border), and "Cancel".

Click **Next** to continue with the **Base System Builder Wizard**.

*Note: the **Base System Builder** options that follow may be different depending on the development board you are using.*

The next steps will demonstrate how to configure the **PowerPC** processor and create the necessary I/O interfaces for our sample application.

See Also

[Configuring the New Platform](#)

1.7 Configuring the New Platform

Complex FIR Filter Tutorial for PowerPC, Step 7

Now that you have created a basic PowerPC project in the **Base System Builder Wizard**, you will need to specify additional information about your platform in order to support the requirements of your software/hardware application. Continuing with the steps provided in the **Base System Builder Wizard**, specify the following information in the Configure processor page, making sure to increase the local data and instruction memory as shown:

System Wide Setting

Reference Clock Frequency: 125 MHz

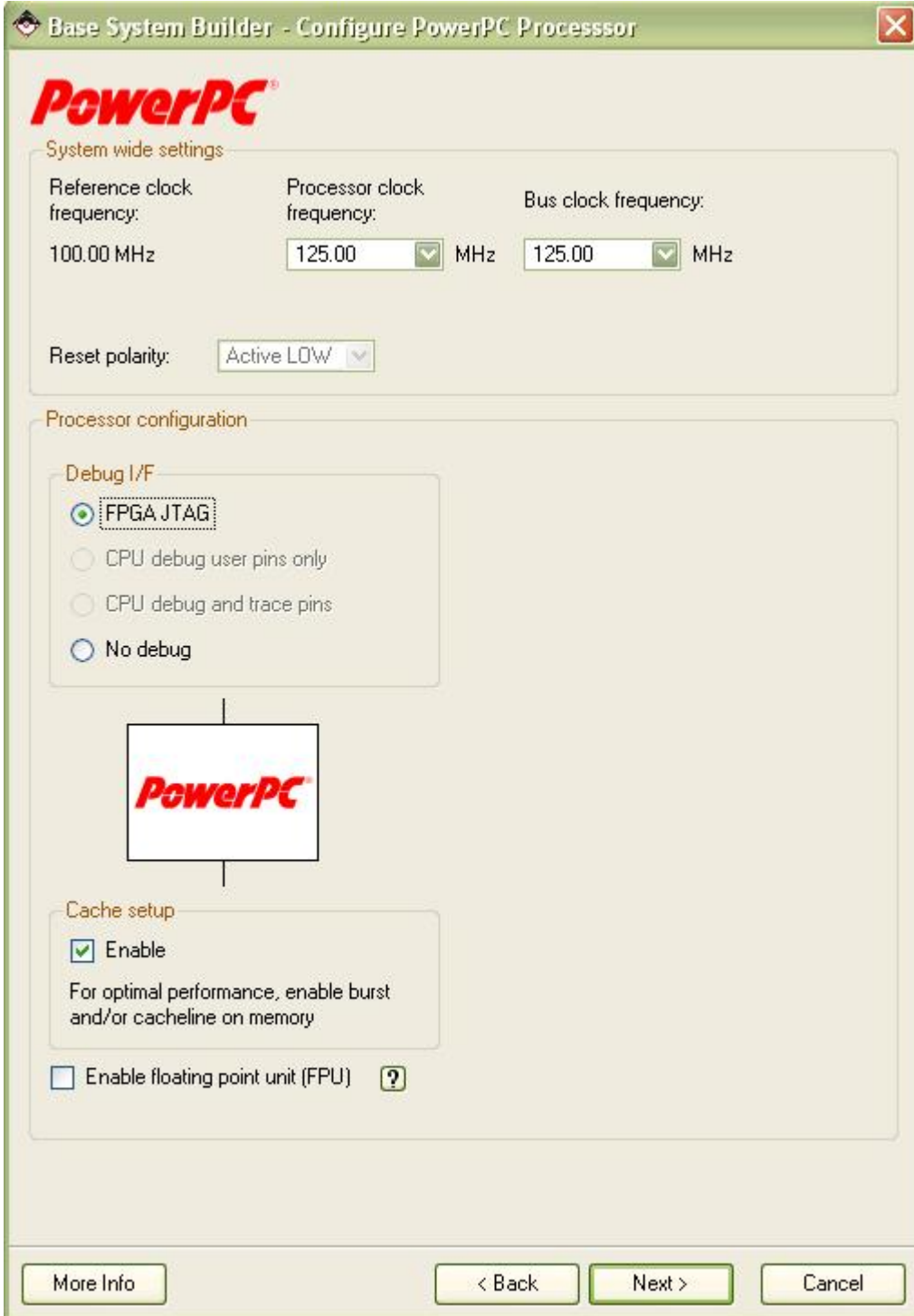
Bus Clock Frequency: 125 MHz

Processor Configuration

Debug I/F: FPGA JTAG (default setting)

Cache setup: Enable

Unselect Enable floating point unit (FPU)



The image shows a Windows-style dialog box titled "Base System Builder - Configure PowerPC Processor". It features the PowerPC logo at the top left. The dialog is divided into two main sections: "System wide settings" and "Processor configuration".

System wide settings:

- Reference clock frequency: 100.00 MHz
- Processor clock frequency: 125.00 MHz (with a dropdown arrow)
- Bus clock frequency: 125.00 MHz (with a dropdown arrow)
- Reset polarity: Active LOW (with a dropdown arrow)

Processor configuration:

Debug I/F:

- ☒ FPGA JTAG
- ☐ CPU debug user pins only
- ☐ CPU debug and trace pins
- ☐ No debug

A diagram shows a box with the PowerPC logo connected by a vertical line to the "Cache setup" section below.

Cache setup:

- ☒ Enable
- For optimal performance, enable burst and/or cacheline on memory
- ☐ Enable floating point unit (FPU) [?]

At the bottom, there are four buttons: "More Info", "< Back", "Next >", and "Cancel". The "Next >" button is highlighted with a green border.

Click **Next** to continue with the wizard. You will now be presented with a series of pages specifying the I/O peripherals to be included with your processor. (The actual layout of these pages will depend on your screen resolution.) Select one **RS232** device peripheral by setting the following options:

I/O Device: RS232_Uart_1
Peripheral: XPS UARTLITE
Baudrate: 9600

Data Bits: 8
Parity: NONE
Use Interrupt: disabled

Unselect the **RS232_Uart_2** and **LED_8Bit** devices.



The following external memory and IO devices were found on your board:
Xilinx Virtex 5 ML507 Evaluation Platform Revision A

Please select the IO devices which you would like to use:

IO devices

☒ RS232_Uart_1 Data Sheet

Peripheral: XPS UARTLITE

Baudrate (bits per seconds): 9600

Data bits: 8

Parity: NONE

☐ Use interrupt

☐ RS232_Uart_2 Data Sheet

☐ LED_8Bit Data Sheet

More Info < Back Next > Cancel

Click **Next** to continue. In the next wizard page, unselect all the **IO devices** as shown below:



Click **Next** to continue. Again, in the next wizard page, unselect all the **IO devices** as shown below:



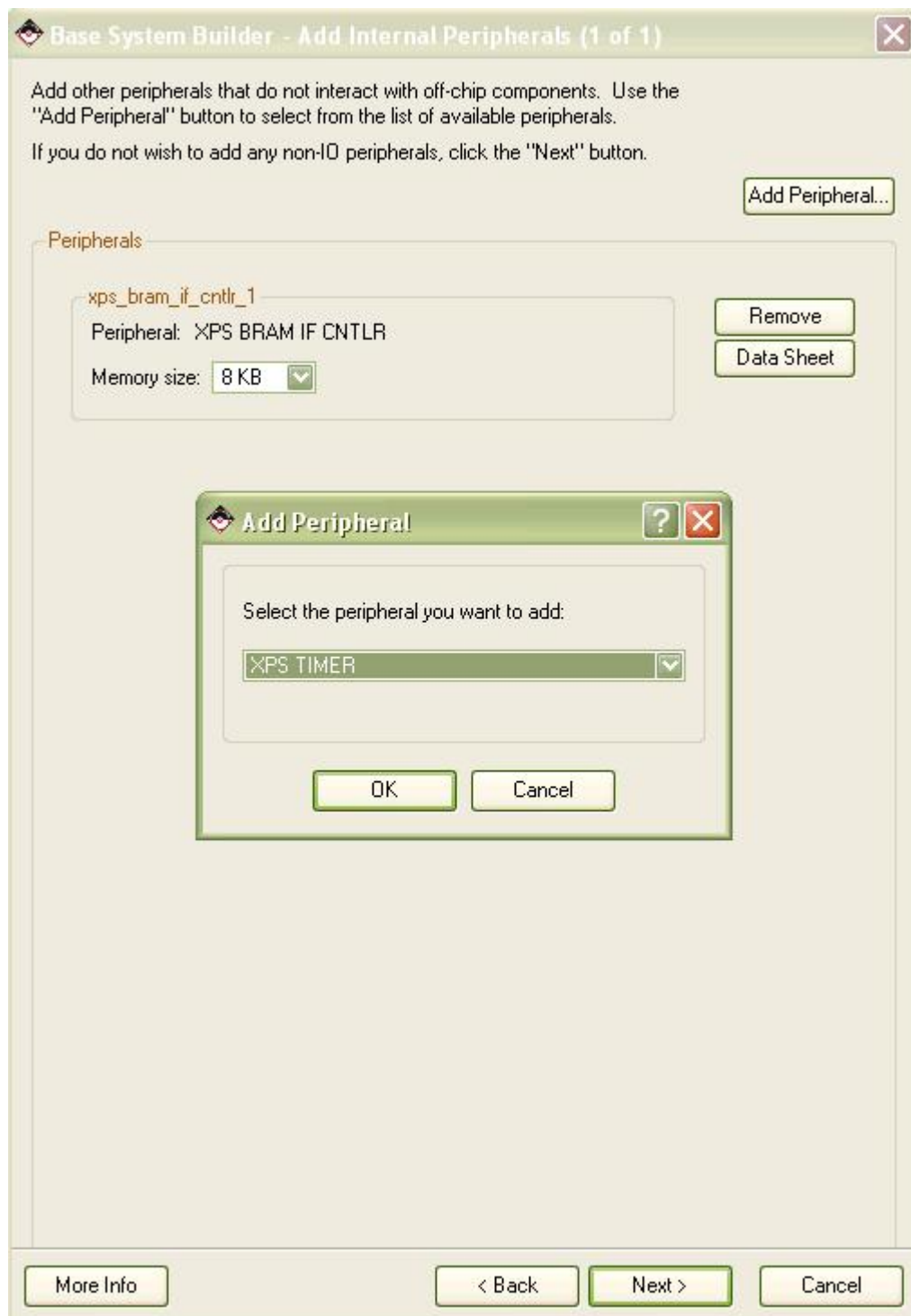
Click **Next** to continue. In the next wizard page, disable all the **IO interfaces** except the **DDR2_SDRAM**:

I/O Devices: DDR2_SDRAM
Peripheral: MPMC

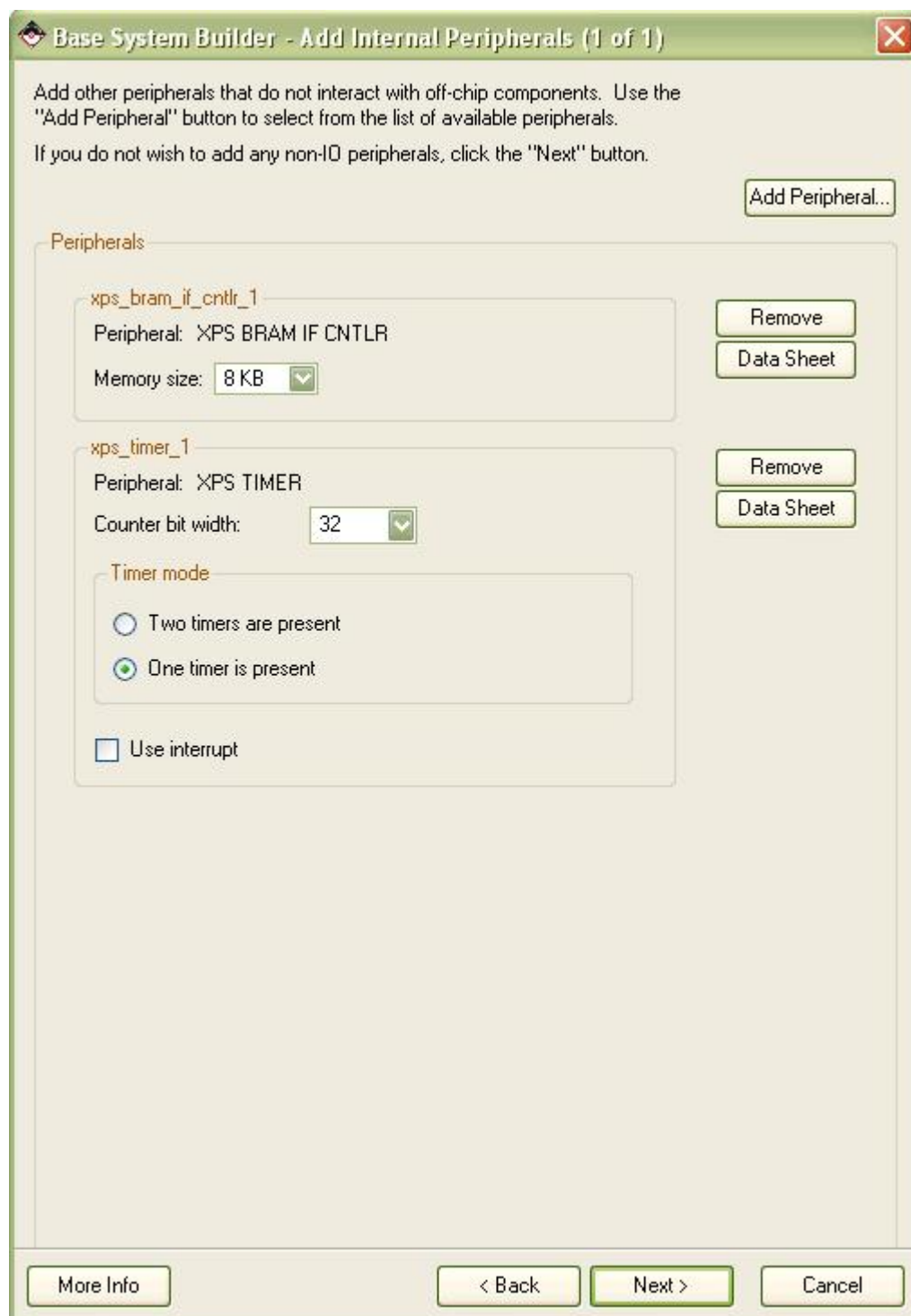


Click **Next** to continue. In the **Add Internal Peripherals** page, the **xps_bram_if_cntlr** is being added with **Memory size** of **8KB**.

Click the **Add Peripheral...** button, and select the **XPS TIMER** peripheral from the dropdown list as shown below:



Click **Next**, and the **xps_timer_1** appears in the **Peripherals** list. Choose to the **Timer mode** as **One timer is present**, and do not **Use interrupt**.

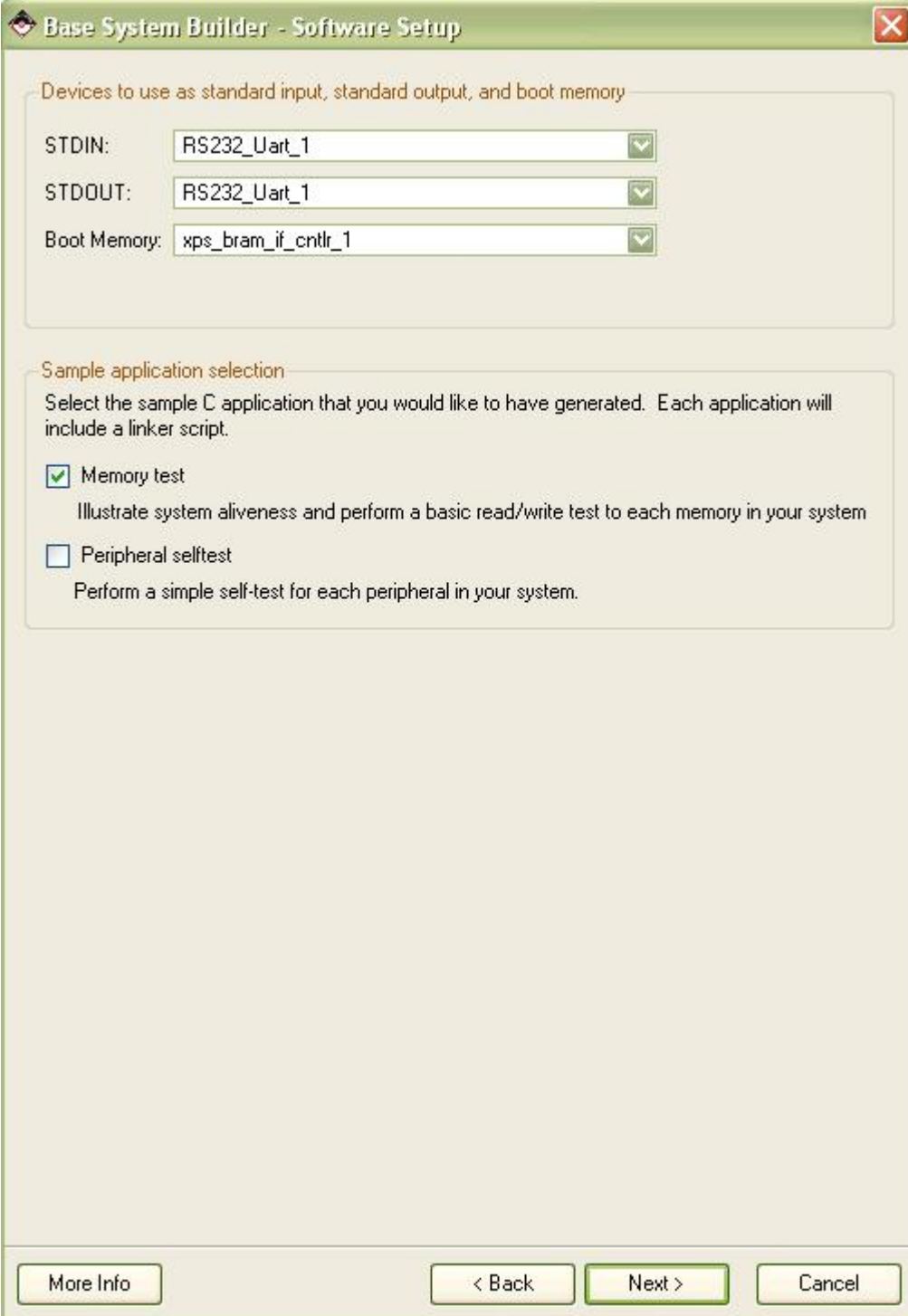


Click **OK** to add the above two peripherals.

In the **Cache Setup** page, choose the cache settings as follows:



On the **Software Setup** dialog that appears, select the **Memory test** option, and unselect the **Peripheral selftest** option:



The image shows a Windows-style dialog box titled "Base System Builder - Software Setup". It has a green title bar with a close button (X) in the top right corner. The dialog is divided into two main sections. The first section, titled "Devices to use as standard input, standard output, and boot memory", contains three dropdown menus: "STDIN:" set to "RS232_Uart_1", "STDOUT:" set to "RS232_Uart_1", and "Boot Memory:" set to "xps_bram_if_cntlr_1". The second section, titled "Sample application selection", contains a paragraph of text: "Select the sample C application that you would like to have generated. Each application will include a linker script." Below this text are two options: "Memory test" (checked with a green checkmark) and "Peripheral selftest" (unchecked). Each option has a descriptive sentence below it. The "Memory test" option says "Illustrate system aliveness and perform a basic read/write test to each memory in your system." The "Peripheral selftest" option says "Perform a simple self-test for each peripheral in your system." At the bottom of the dialog, there are four buttons: "More Info", "< Back", "Next >" (which is highlighted with a green border), and "Cancel".

Base System Builder - Software Setup

Devices to use as standard input, standard output, and boot memory

STDIN: RS232_Uart_1

STDOUT: RS232_Uart_1

Boot Memory: xps_bram_if_cntlr_1

Sample application selection

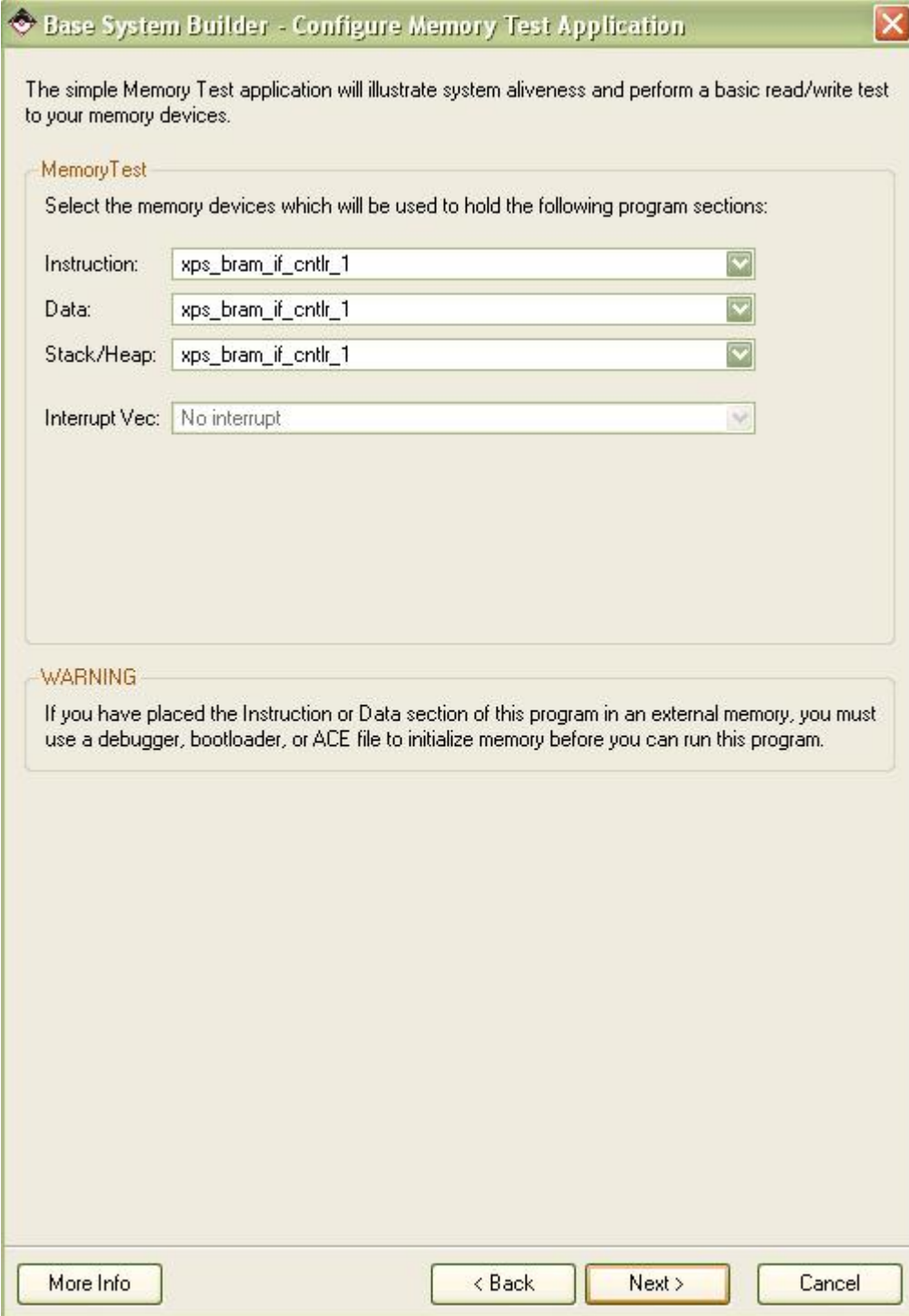
Select the sample C application that you would like to have generated. Each application will include a linker script.

☒ Memory test
Illustrate system aliveness and perform a basic read/write test to each memory in your system

☐ Peripheral selftest
Perform a simple self-test for each peripheral in your system.

More Info < Back Next > Cancel

Click **Next** to view the **Memory Test** software settings as shown below:



Base System Builder - Configure Memory Test Application

The simple Memory Test application will illustrate system aliveness and perform a basic read/write test to your memory devices.

MemoryTest

Select the memory devices which will be used to hold the following program sections:

Instruction:

Data:

Stack/Heap:

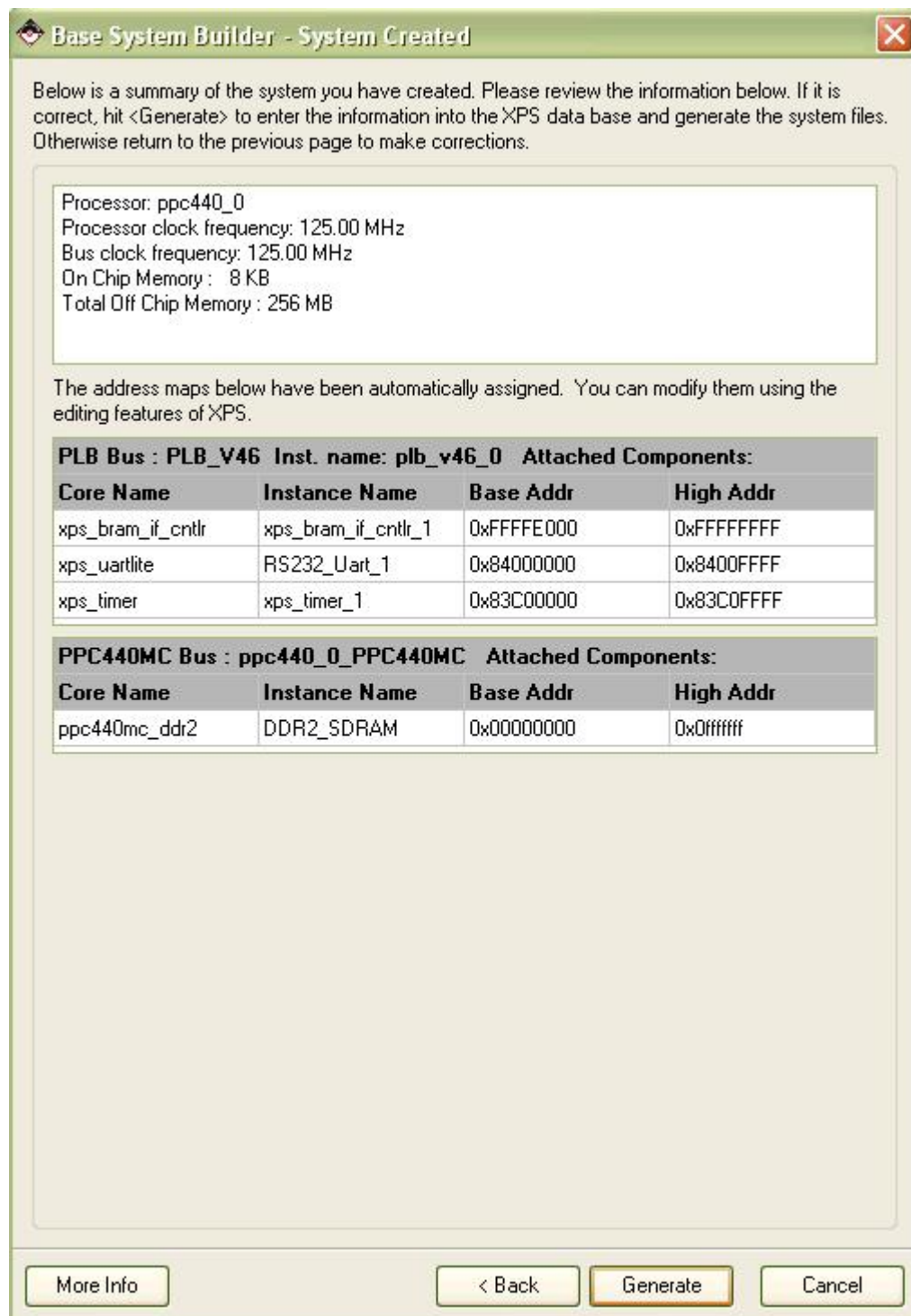
Interrupt Vec:

WARNING

If you have placed the Instruction or Data section of this program in an external memory, you must use a debugger, bootloader, or ACE file to initialize memory before you can run this program.

Click **Next** to accept the default **Memory Test** memory settings.

You have now configured the platform and processor features. The **Base System Builder Wizard** displays a summary of the system you have created:

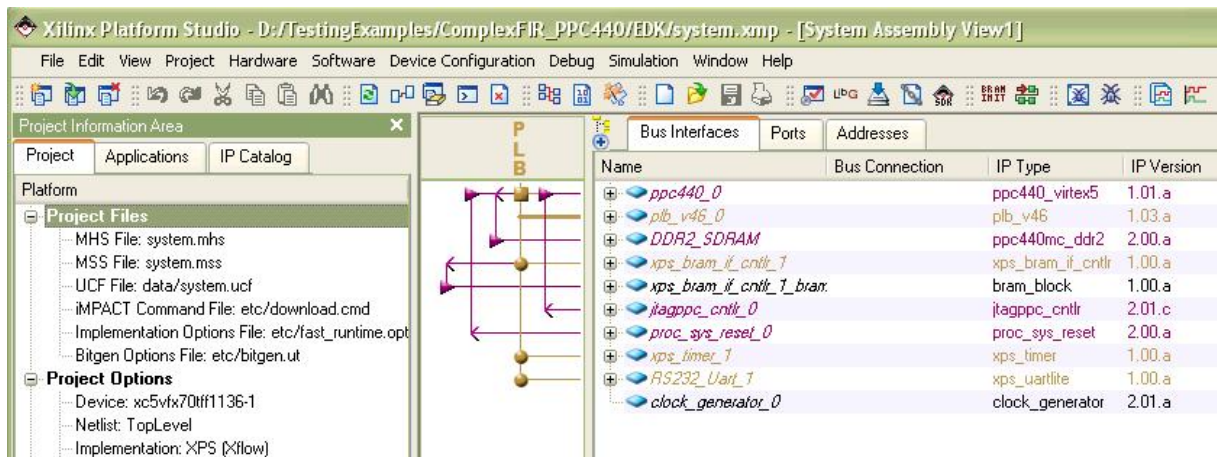


Click **Generate** to generate the system and project files. After this is done, the **Base System Builder** will display the **Finish** page as shown below:



Click **Finish** to close the wizard.

The **System Assembly View** of the **Platform Studio** should look like this:



See Also

[Importing the Generated Hardware](#)

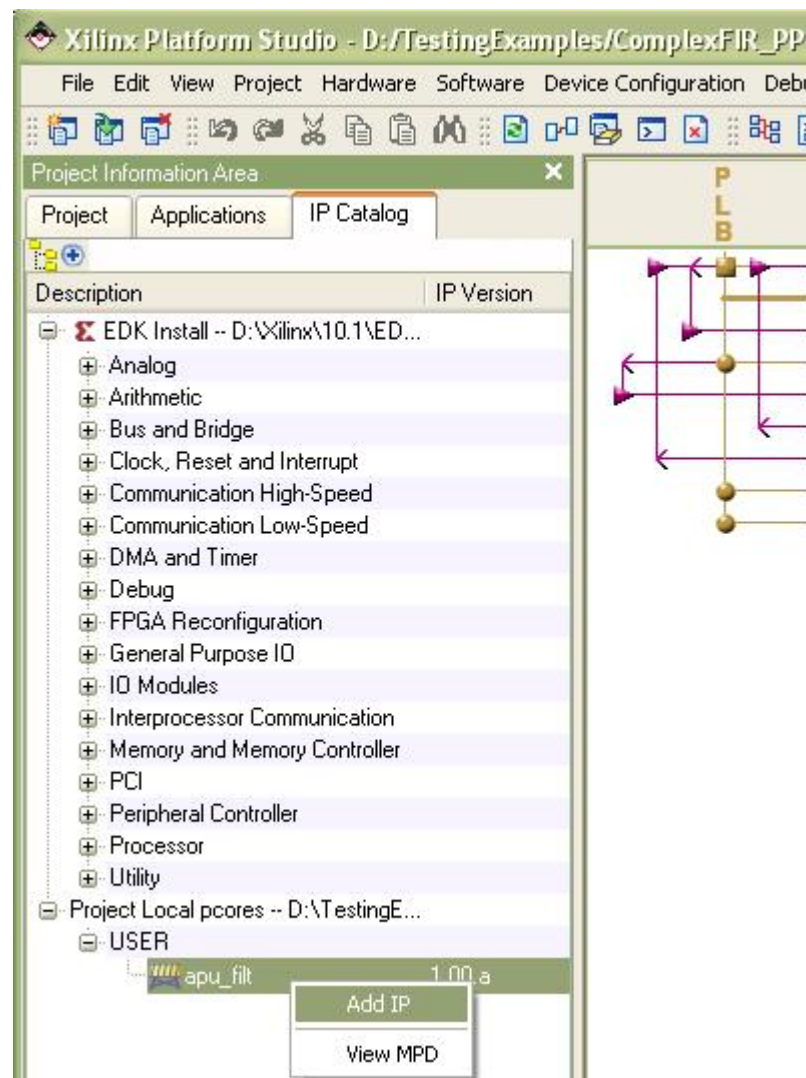
1.8 Importing the Generated Hardware

Complex FIR Filter Tutorial for PowerPC, Step 8

You will now create the target platform in the Xilinx Platform Studio. This procedure is somewhat lengthy but will only need to be done once for any new project.

Add the ComplexFIR Hardware IP Core

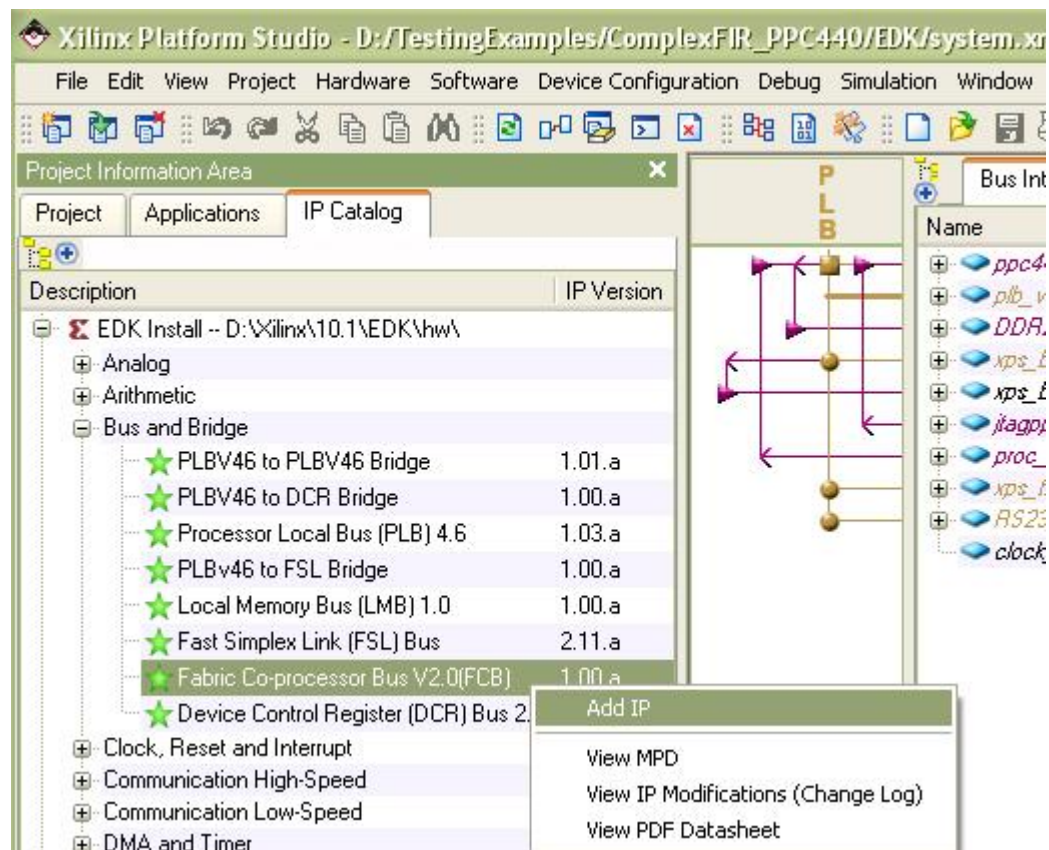
First, add the module representing the **ComplexFIR** hardware process to your development system. Switch to the **IP Catalog** tab in the **Project Information Area**. Expand **Project Local pcores -> USER** to reveal the generated **apu_filt** IP core. Right-click the **apu_filt** and select **Add IP** as shown below:



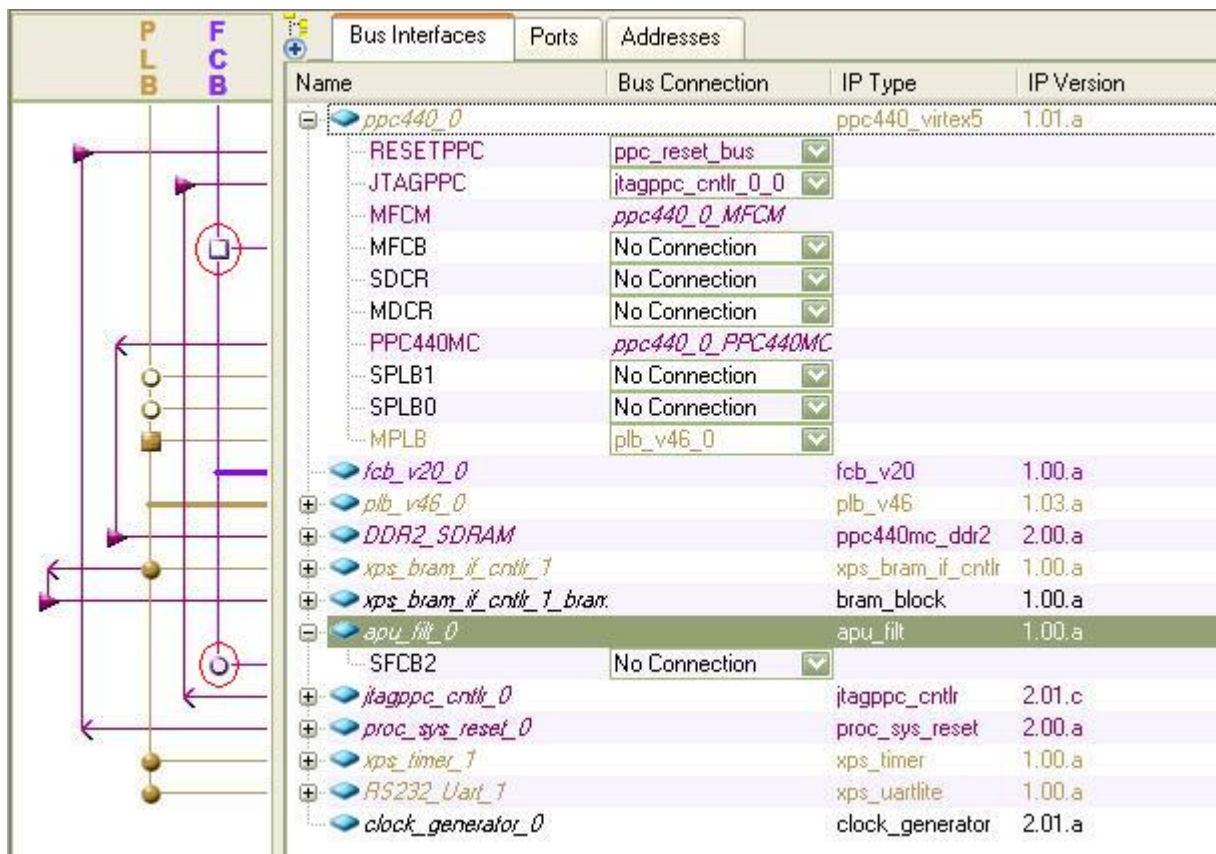
The `apu_filt` module will appear in the list of peripherals in the **System Assembly View** on the right.

Connect the IP Core to PowerPC

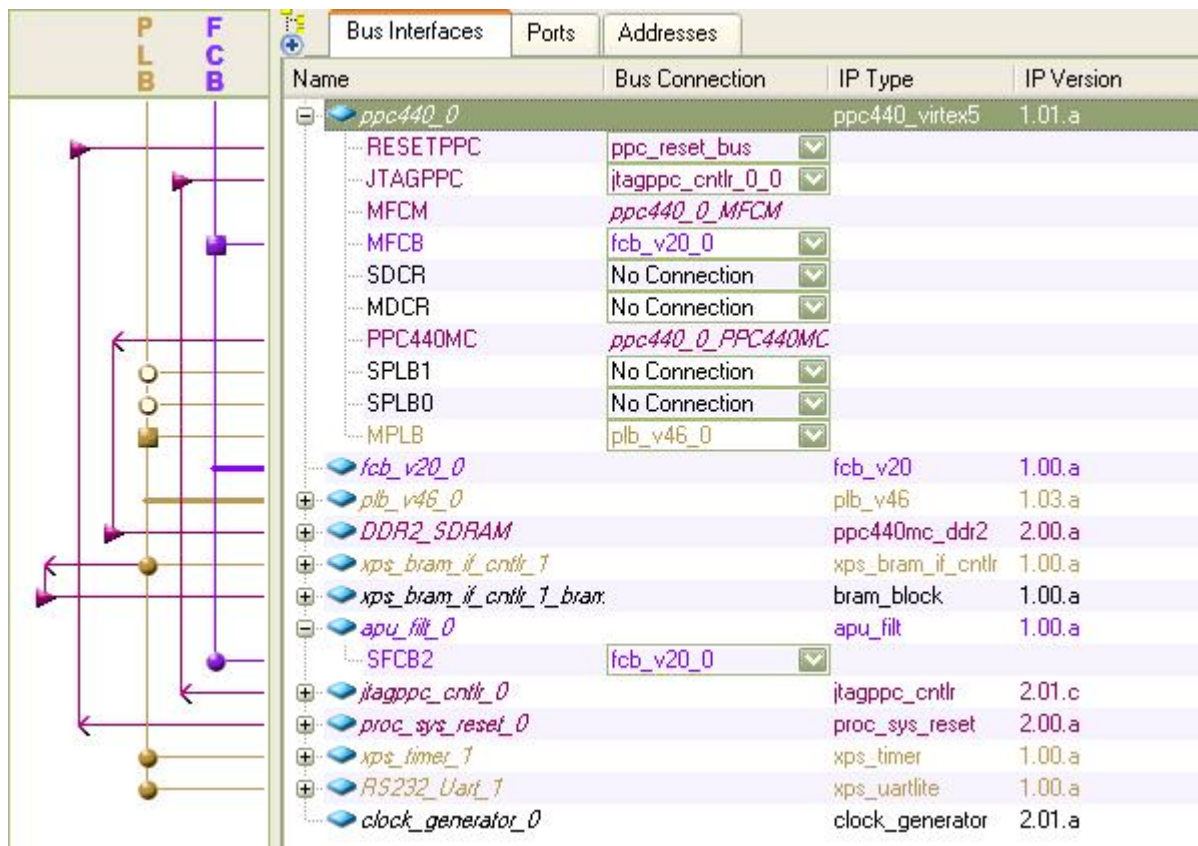
In order to connect the `apu_filt` IP core to the APU of the processor, a **Fabric Co-processor Bus** is needed for interfacing purposes. Click on the **IP Catalog** tab and select the **Fabric Co-processor Bus V2.0 (FCB)** IP core from the **Bus and Bridge** category. Right-click it and select **Add IP** as shown:



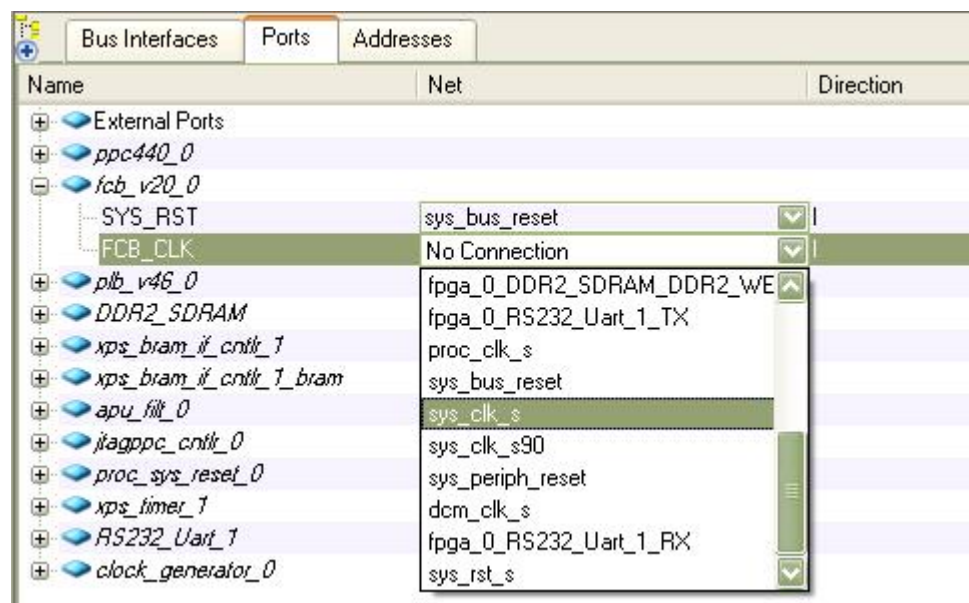
When you have added the **FCB** IP core, your project should look like this:



Notice that there are an empty square and an empty circle on the **FCB** bus. Click the empty square to connect the **ppc440_0**'s **MFCB** interface to the **fcb_v20_0** bus. Click the empty circle to connect the **apu_filt_0**'s **SFCB2** interface to the **fcb_v20_0** bus. They should be connected as shown below:



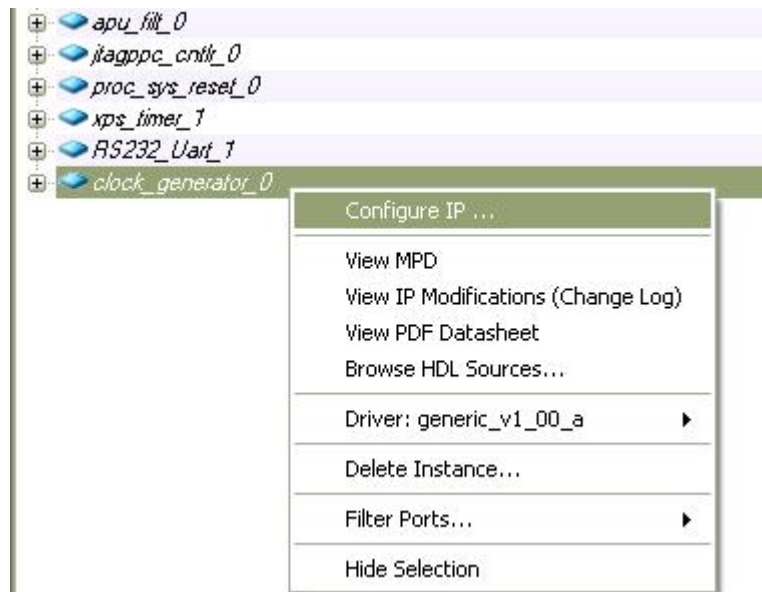
Two ports have to be configured for the **FCB**. Click on the **Ports** tab in the **System Assembly View** and expand **fcb_v20_0** IP core. Set **SYS_Rst** to **sys_bus_reset** and set **FCB_Clk** to **sys_clk_s** as shown.



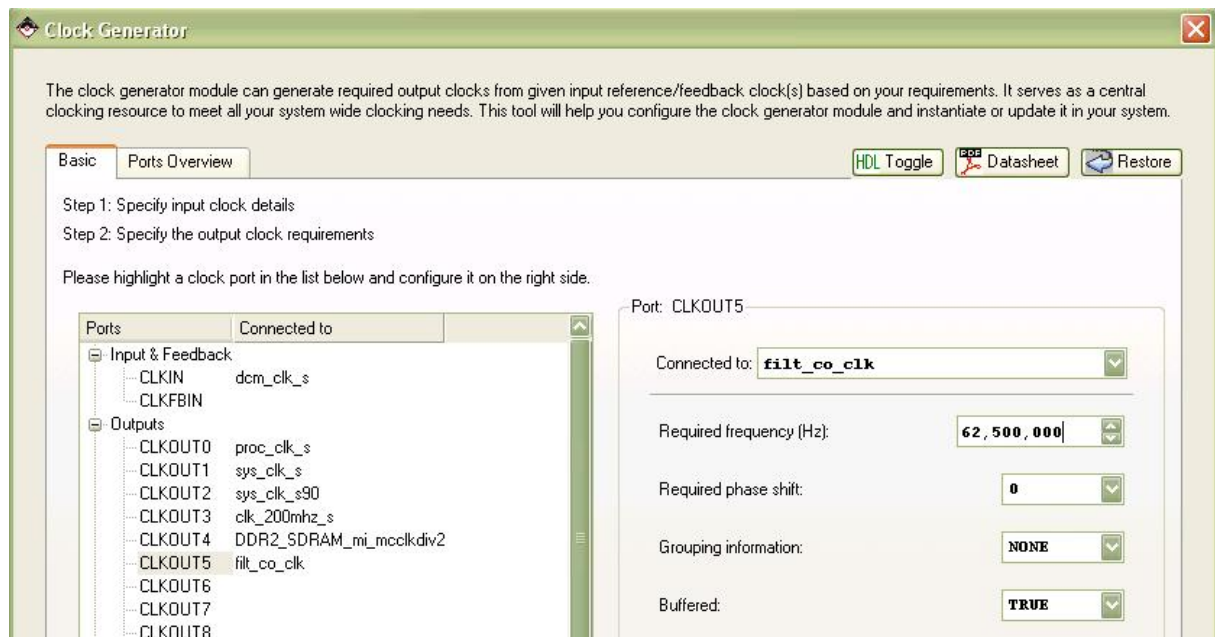
Configuring the Clock

The ComplexFIR hardware requires a separate clock source. For this purpose, we configure the

clock_generator_0 settings by right-clicking and selecting **Configure IP...** as shown:

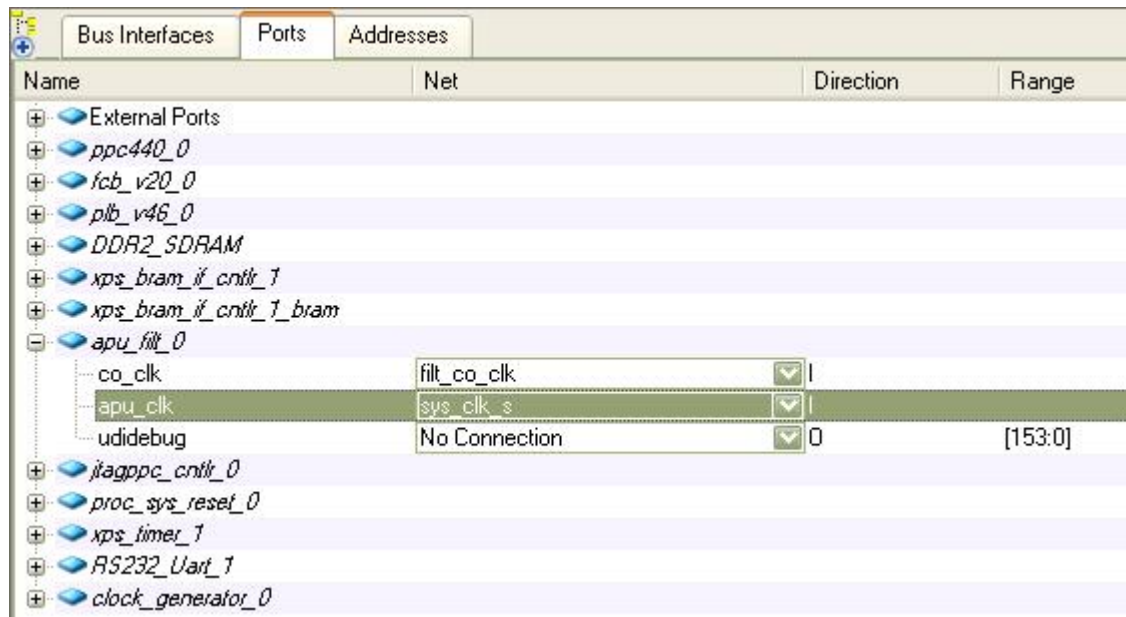


Here we add a new clock output from CLKOUT5 named **filt_co_clk**. The frequency is set to be **62,500,000 Hz**, which is half of the **125,000,000 Hz** system bus frequency.



Click **OK** to save the settings and exit the configure IP dialog.

Select the **Ports** tab in the **System Assembly View** and expand **apu_filt_0**. This should reveal ports **co_clk** and **apu_clk**. The **co_clk** has to be connected to the **filt_co_clk** clock that we configured in the previous steps. The **apu_clk** should be connected to **sys_clk_s**, as shown below:



Note: if **co_clk** is missing from the **apb_filt_0** section, then will need to return to [step 3](#) of this tutorial and specify the **Dual Clock** option in the **CoDeveloper Generate Options** page.

Generate the Addresses

Now you will need to set the addresses for each of the peripherals specified for the platform. This can be done simply by clicking on the **Generate Addresses** button in the **Addresses** tab and . The addresses will be assigned for you automatically:

Bus Interfaces Ports Addresses Generate Addresses							
Instance	Name	Base Address	High Address	Size	Bus Interface(s)	Bus Connection	
plb_v46_0	C_BASEADDR			U	Not Applicable		
xps_bram_if_cntlr_1	C_BASEADDR	0xffffe000	0xffffffff	8K	SPLB	plb_v46_0	
xps_timer_1	C_BASEADDR	0x83c00000	0x83c0ffff	64K	SPLB	plb_v46_0	
RS232_Uart_1	C_BASEADDR	0x84000000	0x8400ffff	64K	SPLB	plb_v46_0	
ppc440_0	C_IDCR_BASEADDR	0b0000000000	0b0011111111	256	Not Connected		
DDR2_SDRAM	C_MEM_BASEADDR	0x00000000	0x0ffffff	256M	PPC440MC	ppc440_0_PPC440MC	
ppc440_0	C_SPLB0_RNG_MC_BASEADDR			U	Not Connected		
ppc440_0	C_SPLB1_RNG_MC_BASEADDR			U	Not Connected		

You have now exported all necessary hardware files from **CoDeveloper** to the Xilinx tools environment and have configured your new platform. The next step will be to compile the HDL files and generate downloadable FPGA bitstream.

See Also

[Generating the FPGA Bitmap](#)

1.9 Generating the FPGA Bitmap

ComplexFIR Filter Tutorial for PowerPC, Step 9

At this point, if you have followed the tutorial steps carefully you have successfully:

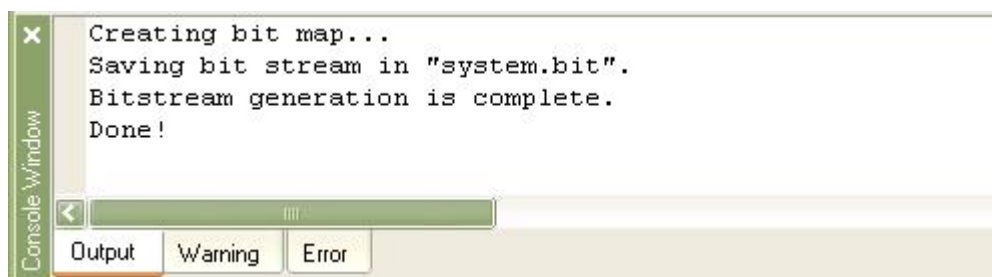
- Generated hardware and software files from the **CoDeveloper** environment.
- Created a new **Xilinx Platform Studio** project and created a new **PowerPC**-based platform.
- Imported your **CoDeveloper**-generated files to the **Xilinx Platform Studio** environment.
- Connected and configured the **Impulse C** hardware process to the **PowerPC** processor via the **FCB** bus.

You are now ready to generate the bitmap.

From within **Xilinx Platform Studio**, select the menu **Hardware -> Generate Bitstream**.

Note: this process may require 10 minutes or more to complete, depending on the speed and memory size of your development system.

After the generation process is done, the message in the **Output Console Window** will be like shown below:



Now we can move on to add our own software application.

See Also

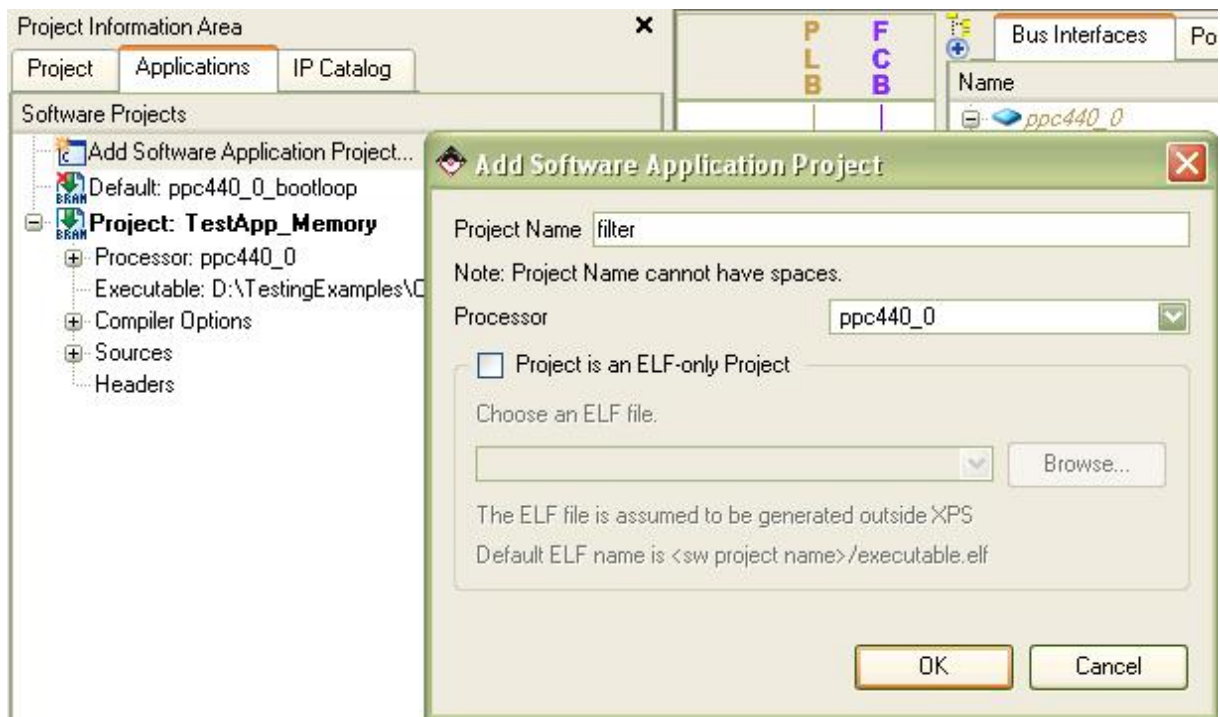
[Importing the Application Software](#)

1.10 Importing the Application Software

Complex FIR Filter Tutorial for PowerPC, Step 10

You will now import the relevant software source files to your new **Platform Studio** project.

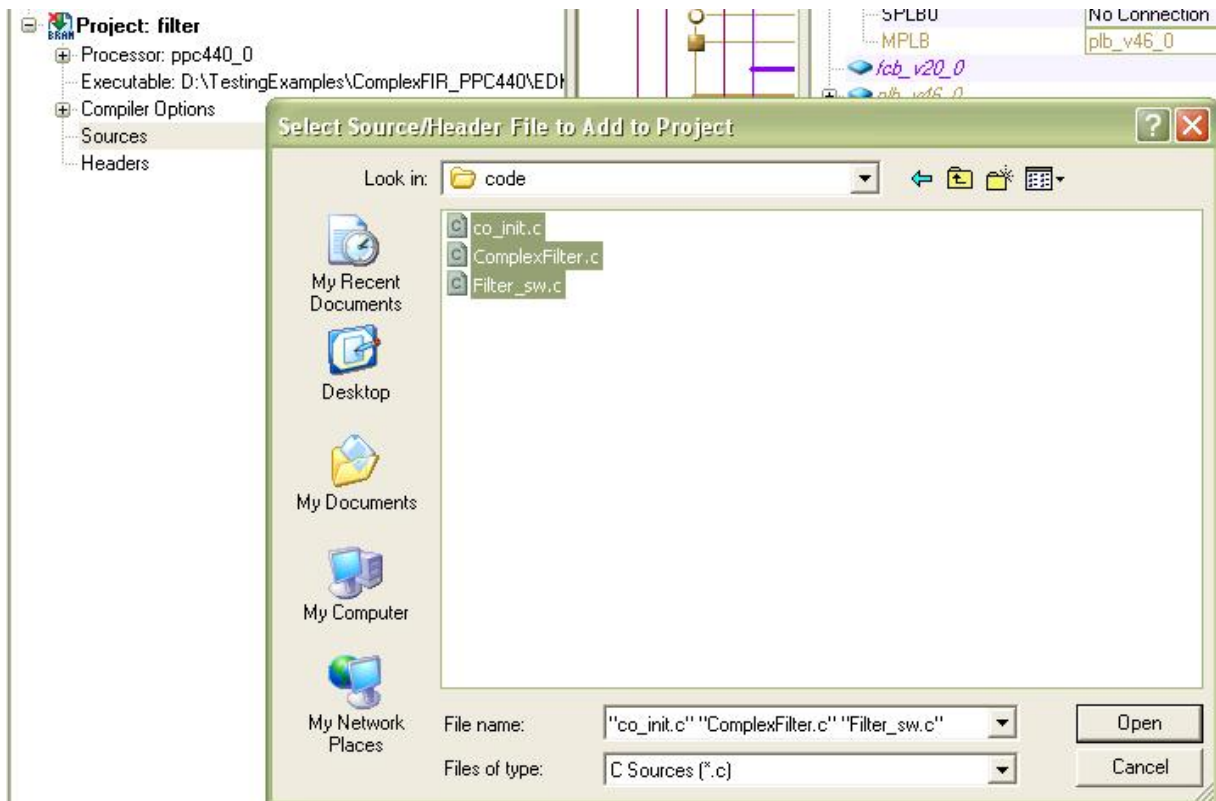
On the **Applications** tab of the **Project Information Area**, create a new software project by double-clicking the **Add Software Application Project...** item. An **Add Software Application Project** dialog will appear. Type in the project name: **filter**.



Click **OK** to continue.

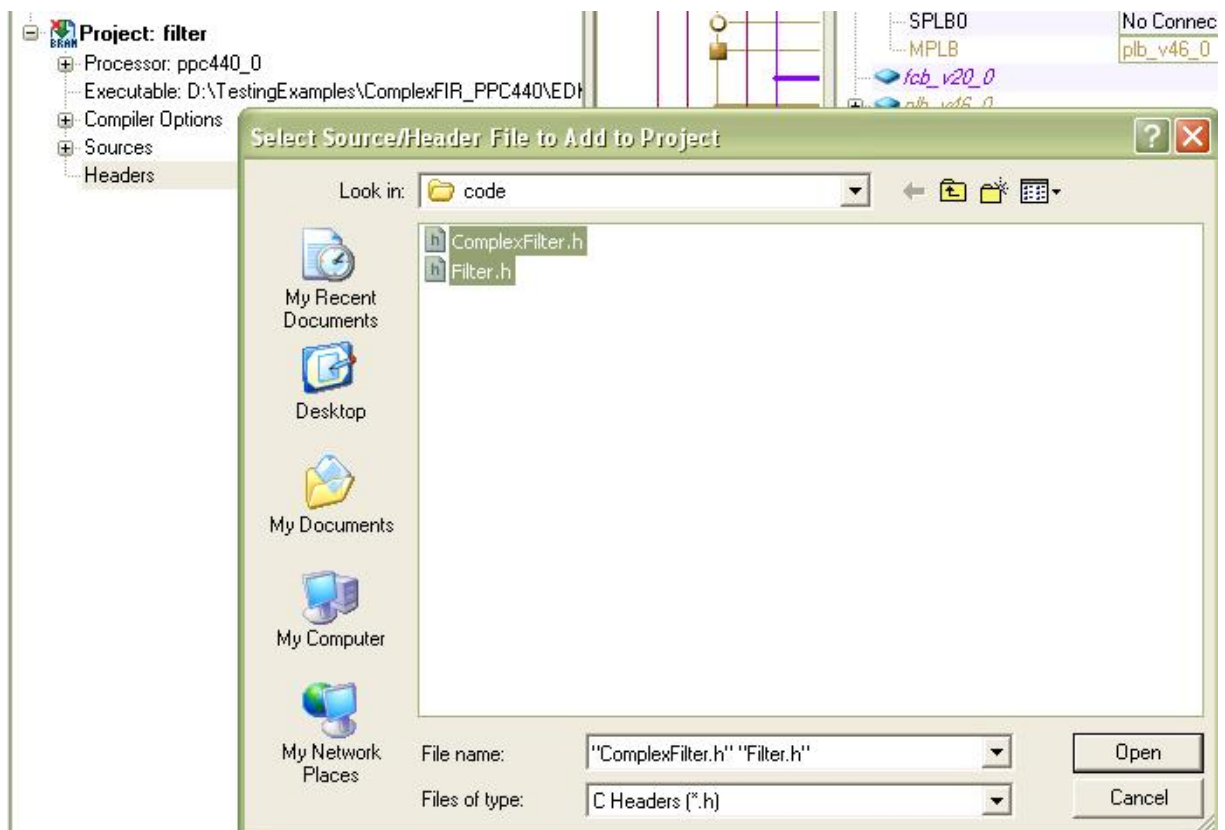
A new **Project: filter** appears in the **Software Projects** list. Double-click the **Sources** item under the **Project: filter** to add source files.

In the **Select Source/Header File to Add to Project** dialog, enter the **code** subdirectory and select all the three C files are shown below:



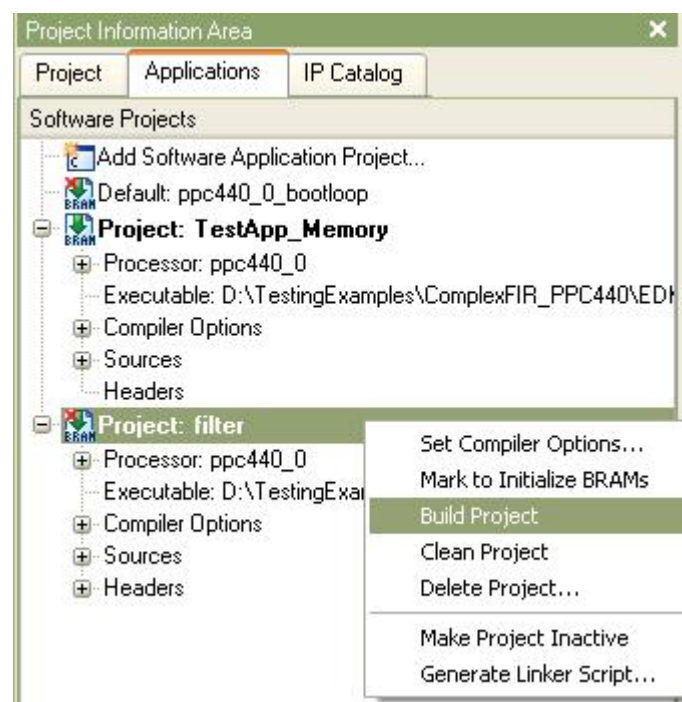
Click **Open** to add the source files shown to your project. These files comprise the software application that will run on the PowerPC CPU.

Next, double-click **Headers** item under the **Project: filter** to add header files. A file selection dialog appears. Select both the header files in the **code** directory shown below.

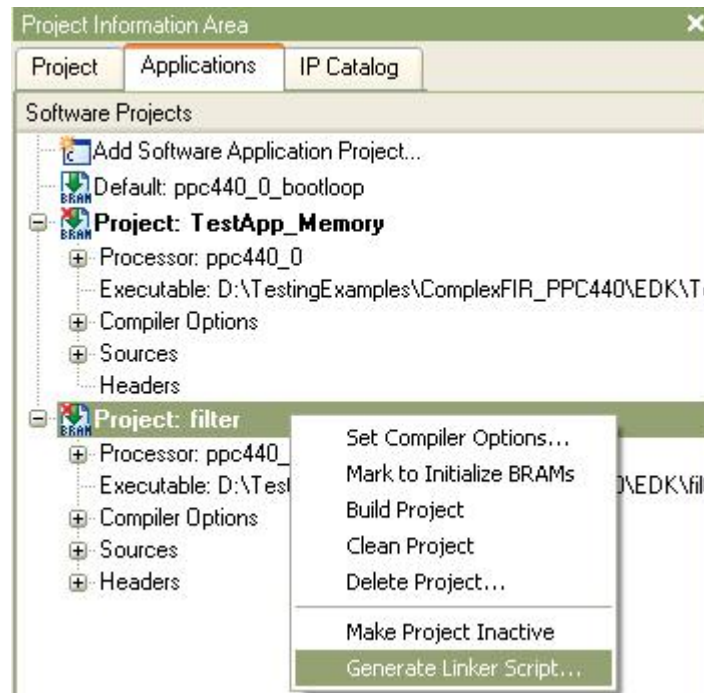


Click **Open** to add the header files to your project.

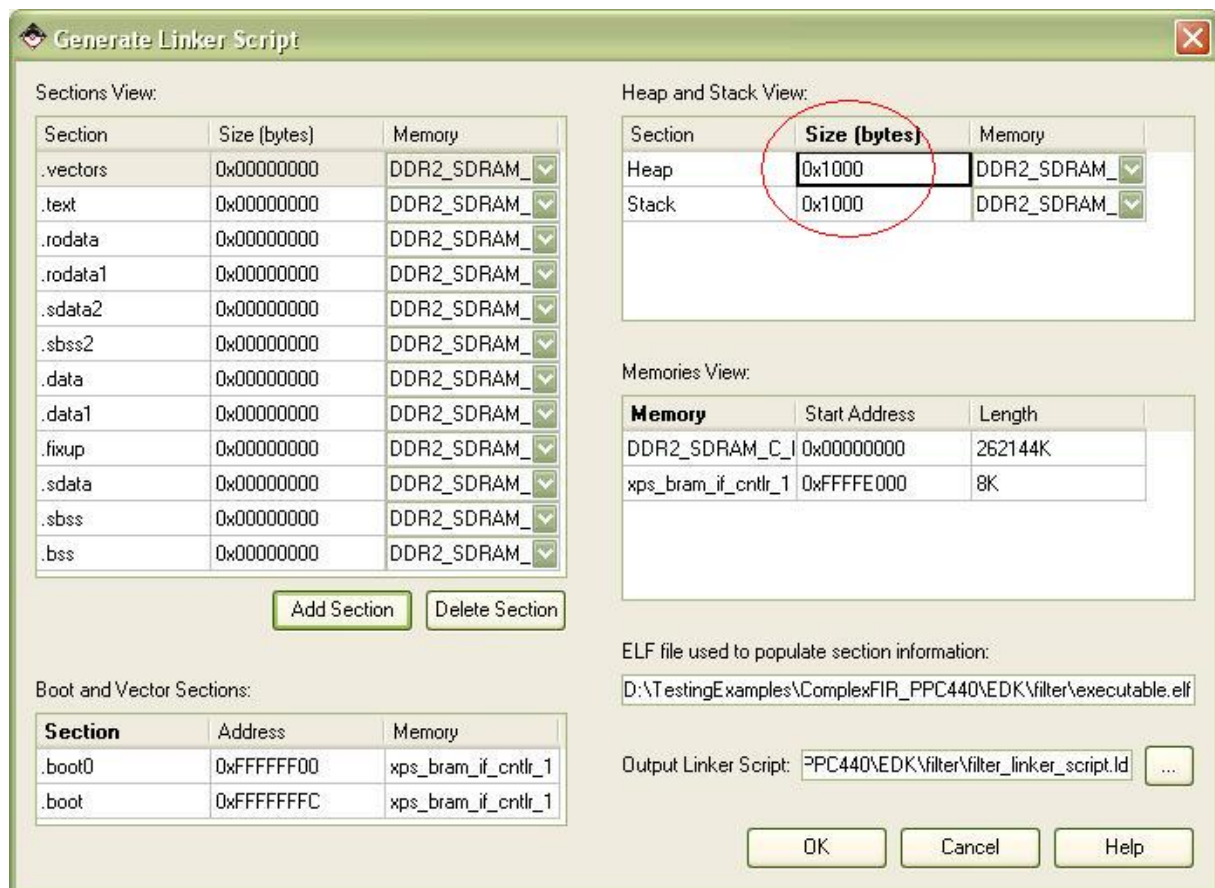
Next, right-click the **Project: filter** and select **Build Project**.



The on-chip memory is not enough for the project. The instruction and data sessions need to be put into external **DDR2 SDRAM**, and also for the heap and stack. To do this, right-click the **Project: filter**, and select **Generate Linker Script...**

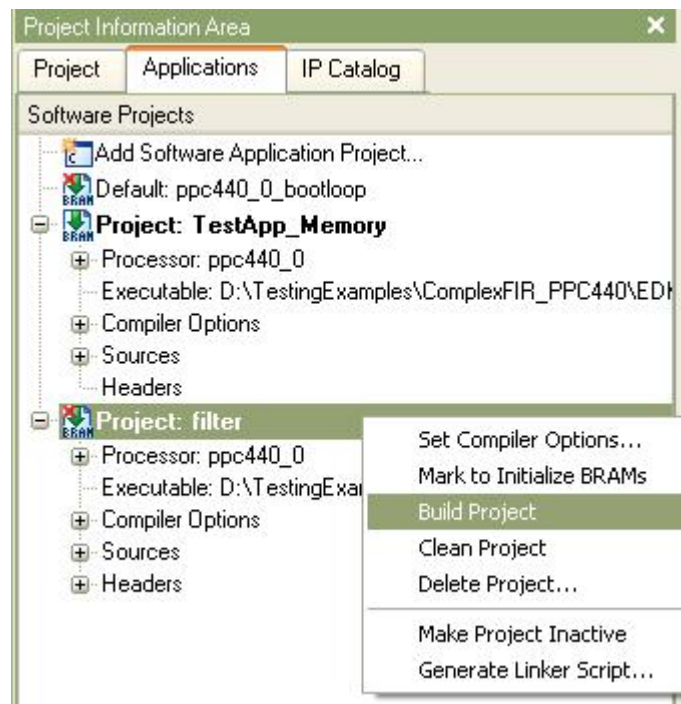


The **Generate Linker Script** dialog will appear. Make sure that all the sections in the **Sections View** are using **DDR2_SDRAM** as memory. For the **Heap and Stack**, enlarge the size to **0x1000 bytes**, and also use **DDR2_SDRAM** as memory.



Click **OK** to generate the **Linker Script** file.

Next, right-click the **Project: filter** and select **Build Project**.



The following messages shown in the **Console Window Output** indicate the software project is built.

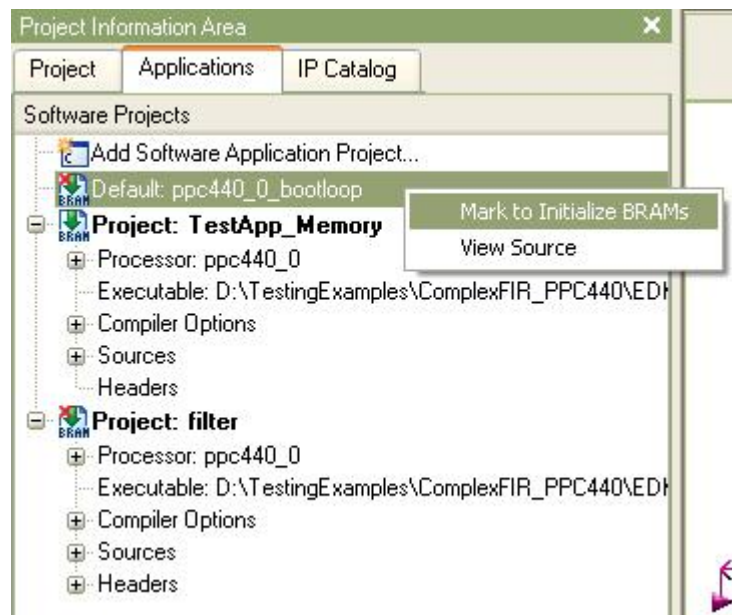
```

At Local date and time: Wed Mar 11 09:28:37 2009
make -f system.make filter_program started...
powerpc-eabi-gcc -O2 /cygdrive/d/TestingExamples/ComplexFIR_PPC440/EDK/code/co_init.c /cygdrive/d/TestingExamples/
-mcpu=440 -Wl,-T -Wl,/cygdrive/d/TestingExamples/ComplexFIR_PPC440/EDK/filter/filter_linker_script.ld -g

powerpc-eabi-size filter/executable.elf
   text    data    bss     dec     hex filename
  17174    1372    33128   51674   c9da filter/executable.elf
Done!

```

Now you will need to change the BRAM initialization application, which is currently the **TestApp_Memory** project. Right-click the **Default: PowerPC_0_bootloop** and select **Mark to Initialize BRAMs**. This will let the bootloop reside in the **BRAMs**.



Next, you will run the application.

See Also

[Running the Application](#)

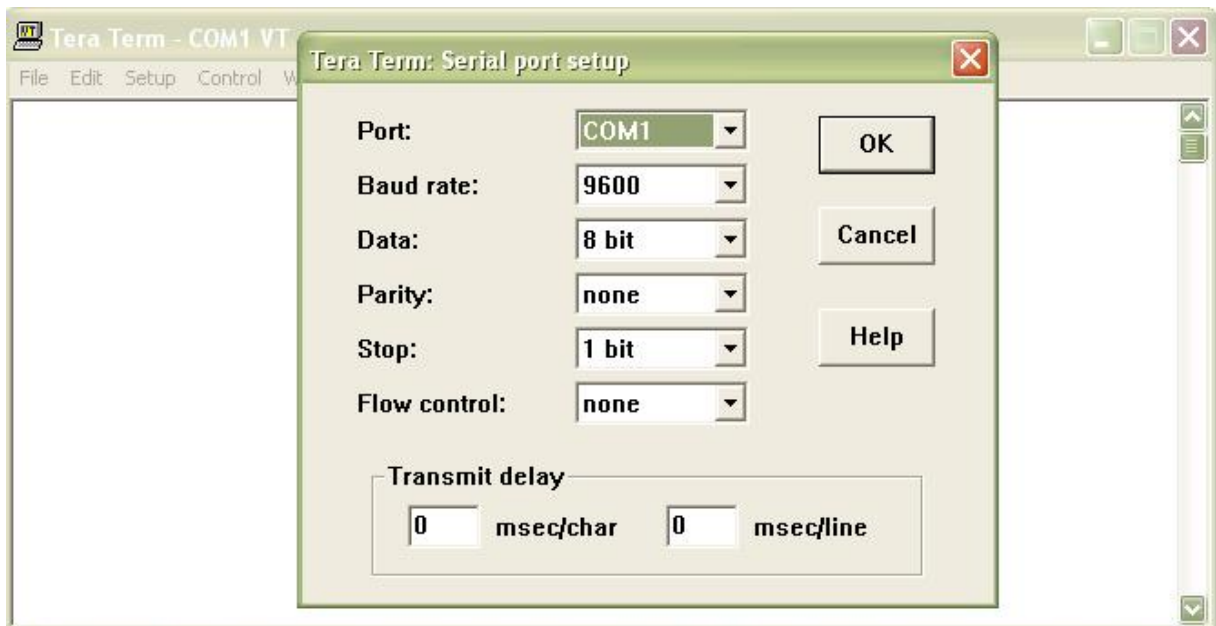
1.11 Running the Application

Complex FIR Filter Tutorial for PowerPC, Step 11

Now let's run the application on the development board.

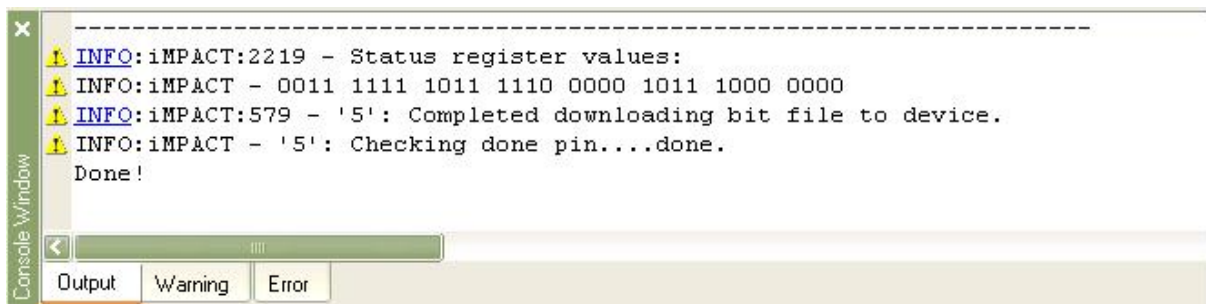
Connect the serial port of your development machine to that of your development board via a **RS232** cable. Make sure the **JTAG** download cable is connected on the development board. Also ensure that the board is configured to be programmed. Turn on the power to the board.

Open **Tera Term** or **Windows HyperTerminal** application to display the UART output message. Use the same communication settings you chose when defining the **RS232_Uart_1** peripheral in **Base System Builder** (9600 baud, 8-N-1). Turn off **flow control**, if available.



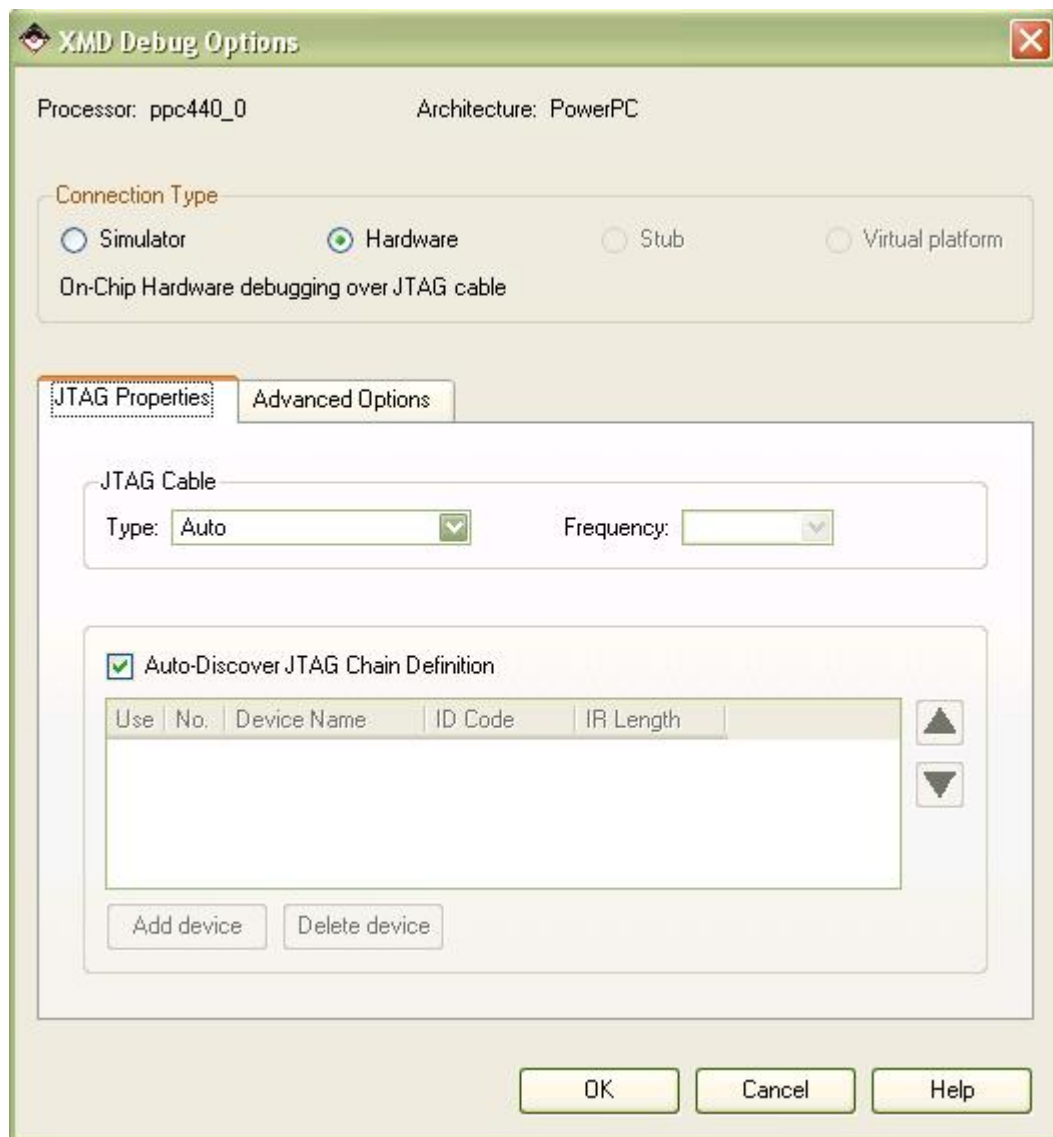
Now, download the bitstream to the device by selecting **Device Configuration -> Download Bitstream**.

When downloading is done, the **Console Window Output** will be like this:



Select from menu **Debug -> Launch XMD...**

The **XMD Debug Options** dialog will appear for the first time opening **XMD**.



Click **OK** to accept the default settings.

A Cygwin bash shell will come up. This runs a script, connecting to the **PowerPC** processor and the debugger inside the FPGA, as shown below:

```

C:\D:\Xilinx\10.1\EDK\bin\nt\xbash.exe
-----
Device    ID Code      IR Length  Part Name
1         f5059093      16        XCF32P
2         f5059093      16        XCF32P
3         59608093      8         xc95144x1
4         0a001093      8         System_ACE
5         632c6093     10        XC5VFX70T_U

PowerPC440 Processor Configuration
-----
Version.....0x7ff21912
User ID.....0x00f00002
No of PC Breakpoints.....4
No of Addr/Data Watchpoints.....2
User Defined Address Map to access Special PowerPC Features using XMD:
  I-Cache <Data>.....0x70000000 - 0x70007fff
  I-Cache <TAG>.....0x70008000 - 0x7000ffff
  D-Cache <Data>.....0x78000000 - 0x78007fff
  D-Cache <TAG>.....0x78008000 - 0x7800ffff
  DCR.....0x78020000 - 0x78020fff
  TLB.....0x70020000 - 0x70023fff

Connected to "ppc" target. id = 0
Starting GDB server for "ppc" target (id = 0) at TCP port no 1234
XMD%

```

Now you can download the **filter** project **ELF** file to the target board and run the application with the following **XMD** command:

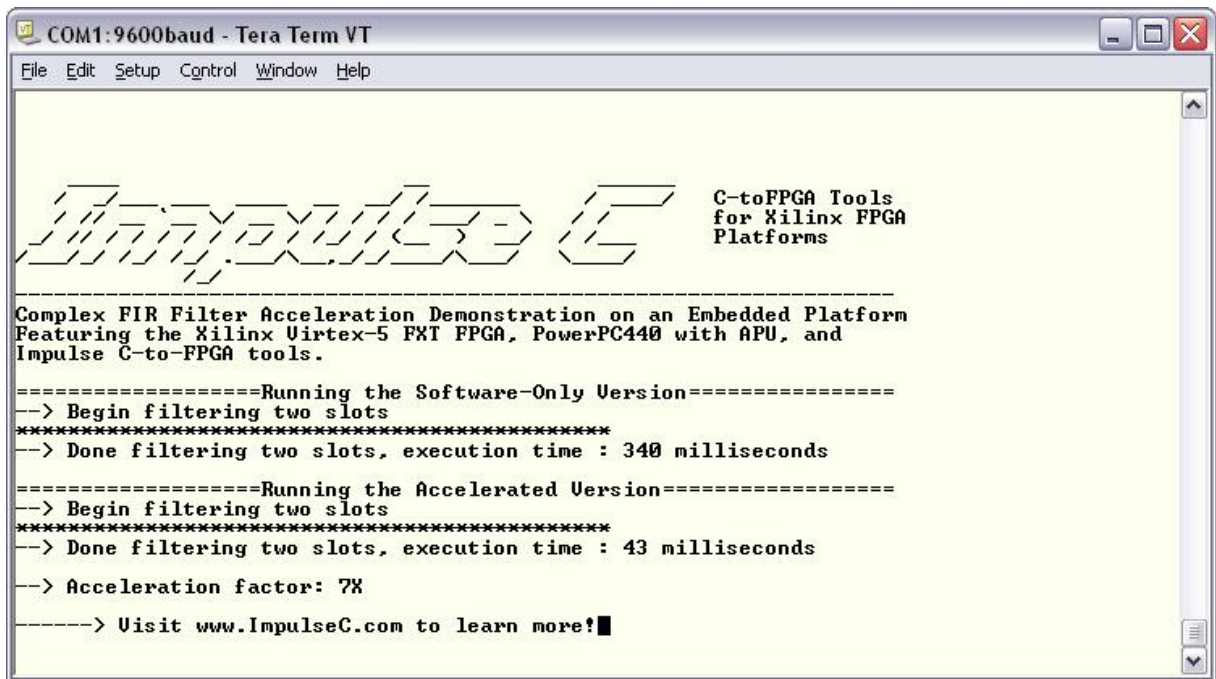
```
dow filter/executable.elf
con
```

```

C:\D:\Xilinx\10.1\EDK\bin\nt\xbash.exe
XMD% dow filter/executable.elf
System Reset .... DONE
Downloading Program -- filter/executable.elf
section, .text: 0x00000000-0x00003b1b
section, .init: 0x00003b1c-0x00003b3f
section, .fini: 0x00003b40-0x00003b5f
section, .boot0: 0xffffffff00-0xffffffffa7
section, .boot: 0xfffffffffc-0xffffffffff
section, .rodata: 0x00003b60-0x00004269
section, .sdata2: 0x0000426c-0x0000426b
section, .sbss2: 0x0000426c-0x0000426b
section, .data: 0x0000426c-0x0000476b
section, .got: 0x0000476c-0x0000476b
section, .got1: 0x0000476c-0x0000476b
section, .got2: 0x0000476c-0x00004787
section, .ctors: 0x00004788-0x0000478f
section, .dtors: 0x00004790-0x00004797
section, .fixup: 0x00004798-0x00004797
section, .eh_frame: 0x00004798-0x0000479f
section, .jcr: 0x000047a0-0x000047a3
section, .gcc_except_table: 0x000047a4-0x000047a3
section, .sdata: 0x000047a4-0x000047c7
section, .sbss: 0x000047c8-0x0000480b
section, .bss: 0x0000480c-0x0000a923
section, .stack: 0x0000a924-0x0000b92f
section, .heap: 0x0000b930-0x0000c92f
Setting PC with Program Start Address 0xfffffffffc
XMD% con
Info:Processor started. Type "stop" to stop processor
RUNNING> XMD%

```

Watch the Tera Term window again. You should see the messages generated by the software process indicating that the test data has been successfully filtered. The execution with hardware acceleration is 7 times faster than the software-only version running on **PowerPC** microprocessor.



The screenshot shows a Tera Term VT window titled "COM1:9600baud - Tera Term VT". The window contains the following text:

```
Impulse C-toFPGA Tools
for Xilinx FPGA
Platforms

-----
Complex FIR Filter Acceleration Demonstration on an Embedded Platform
Featuring the Xilinx Virtex-5 FXT FPGA, PowerPC440 with APU, and
Impulse C-to-FPGA tools.

=====Running the Software-Only Version=====
--> Begin filtering two slots
*****
--> Done filtering two slots, execution time : 340 milliseconds

=====Running the Accelerated Version=====
--> Begin filtering two slots
*****
--> Done filtering two slots, execution time : 43 milliseconds
--> Acceleration factor: 7X

-----> Visit www.ImpulseC.com to learn more!■
```

Congratulations! You have successfully completed this tutorial and run the generated hardware on the development board.