

Virtex-4 FX PowerPC System with FPU

***Implementing the Design
Using Platform Studio***

UG243 (v1.2) October 31, 2007





Xilinx is disclosing this Document and Intellectual Property (hereinafter “the Design”) to you for use in the development of designs to operate on, or interface with Xilinx FPGAs. Except as stated herein, none of the Design may be copied, reproduced, distributed, republished, downloaded, displayed, posted, or transmitted in any form or by any means including, but not limited to, electronic, mechanical, photocopying, recording, or otherwise, without the prior written consent of Xilinx. Any unauthorized use of the Design may violate copyright laws, trademark laws, the laws of privacy and publicity, and communications regulations and statutes.

Xilinx does not assume any liability arising out of the application or use of the Design; nor does Xilinx convey any license under its patents, copyrights, or any rights of others. You are responsible for obtaining any rights you may require for your use or implementation of the Design. Xilinx reserves the right to make changes, at any time, to the Design as deemed desirable in the sole discretion of Xilinx. Xilinx assumes no obligation to correct any errors contained herein or to advise you of any correction if such be made. Xilinx will not assume any liability for the accuracy or correctness of any engineering or technical support or assistance provided to you in connection with the Design.

THE DESIGN IS PROVIDED “AS IS” WITH ALL FAULTS, AND THE ENTIRE RISK AS TO ITS FUNCTION AND IMPLEMENTATION IS WITH YOU. YOU ACKNOWLEDGE AND AGREE THAT YOU HAVE NOT RELIED ON ANY ORAL OR WRITTEN INFORMATION OR ADVICE, WHETHER GIVEN BY XILINX, OR ITS AGENTS OR EMPLOYEES. XILINX MAKES NO OTHER WARRANTIES, WHETHER EXPRESS, IMPLIED, OR STATUTORY, REGARDING THE DESIGN, INCLUDING ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE, AND NONINFRINGEMENT OF THIRD-PARTY RIGHTS.

IN NO EVENT WILL XILINX BE LIABLE FOR ANY CONSEQUENTIAL, INDIRECT, EXEMPLARY, SPECIAL, OR INCIDENTAL DAMAGES, INCLUDING ANY LOST DATA AND LOST PROFITS, ARISING FROM OR RELATING TO YOUR USE OF THE DESIGN, EVEN IF YOU HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. THE TOTAL CUMULATIVE LIABILITY OF XILINX IN CONNECTION WITH YOUR USE OF THE DESIGN, WHETHER IN CONTRACT OR TORT OR OTHERWISE, WILL IN NO EVENT EXCEED THE AMOUNT OF FEES PAID BY YOU TO XILINX HEREUNDER FOR USE OF THE DESIGN. YOU ACKNOWLEDGE THAT THE FEES, IF ANY, REFLECT THE ALLOCATION OF RISK SET FORTH IN THIS AGREEMENT AND THAT XILINX WOULD NOT MAKE AVAILABLE THE DESIGN TO YOU WITHOUT THESE LIMITATIONS OF LIABILITY.

The Design is not designed or intended for use in the development of on-line control equipment in hazardous environments requiring fail-safe controls, such as in the operation of nuclear facilities, aircraft navigation or communications systems, air traffic control, life support, or weapons systems (“High-Risk Applications”). Xilinx specifically disclaims any express or implied warranties of fitness for such High-Risk Applications. You represent that use of the Design in such High-Risk Applications is fully at your risk.

©2006–2007 Xilinx, Inc. All rights reserved. XILINX, the Xilinx logo, and other designated brands included herein are trademarks of Xilinx, Inc. PowerPC is a trademark of IBM, Inc. All other trademarks are the property of their respective owners.

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
11/17/06	1.0	Initial Xilinx release.
03/22/07	1.1	Updated for functionality with ISE 9.1i.
10/31/07	1.2	Updated for functionality with EDK 9.2i and ISE 9.2i.

Table of Contents

Preface: About This Guide

Guide Contents	5
Additional Resources	5
Conventions	5
Typographical	5
Online Document	6

Chapter 1: Tutorial Overview

The Tutorial Design Goal	7
--------------------------------	---

Chapter 2: Building and Running the Design

Install Reference Design Files into Target Directories	9
Build a Basic XPS Hardware and Software System Using Base System Builder	10
Build the System with the Test Application Using Platform Studio (XPS)	13
Adding the Application Code and Configuring for Compilation	14
Running the Software-Only Design	15
Using the New Floating Point Performance Library	16
Adding and Enabling the Hardware-Based Floating Point Unit	16
Building and Running the System	18
Answers	19

Appendix A: Tutorial Configuration

Hardware Requirements	21
Software Requirements	21
System Configuration	21

About This Guide

This tutorial provides a complete walk-through of the steps necessary to implement a Virtex™-4 FX PowerPC™ 405 system with floating point coprocessor using Xilinx Platform Studio software.

Guide Contents

This manual contains the following chapters:

- [Chapter 1, “Tutorial Overview,”](#) contains background material for completing the tutorial.
- [Chapter 2, “Building and Running the Design,”](#) provides a step-by-step walk-through of the tutorial.
- [Appendix A, “Tutorial Configuration,”](#) lists the necessary software and hardware configuration for the tutorial.

Additional Resources

To find additional documentation, see the Xilinx website at:

<http://www.xilinx.com/literature>.

To search the Answer Database of silicon, software, and IP questions and answers, or to create a technical support WebCase, see the Xilinx website at:

<http://www.xilinx.com/support>.

Conventions

This document uses the following conventions. An example illustrates each convention.

Typographical

The following typographical conventions are used in this document:

Convention	Meaning or Use	Example
Courier font	Messages, prompts, and program files that the system displays	speed grade: - 100
Courier bold	Literal commands that you enter in a syntactical statement	ngdbuild design_name

Convention	Meaning or Use	Example
Helvetica bold	Commands that you select from a menu	File → Open
	Keyboard shortcuts	Ctrl+C
Italic font	Variables in a syntax statement for which you must supply values	ngdbuild <i>design_name</i>
	References to other manuals	See the <i>Development System Reference Guide</i> for more information.
	Emphasis in text	If a wire is drawn so that it overlaps the pin of a symbol, the two nets are <i>not</i> connected.
Square brackets []	An optional entry or parameter. However, in bus specifications, such as bus [7:0] , they are required.	ngdbuild [<i>option_name</i>] <i>design_name</i>
Braces { }	A list of items from which you must choose one or more	lowpwr = { on off }
Vertical bar	Separates items in a list of choices	lowpwr = { on off }
Vertical ellipsis . . .	Repetitive material that has been omitted	IOB #1: Name = QOUT' IOB #2: Name = CLKIN' . . .
Horizontal ellipsis ...	Repetitive material that has been omitted	allow block <i>block_name loc1 loc2 ... locn</i> ;

Online Document

The following conventions are used in this document:

Convention	Meaning or Use	Example
Blue text	Cross-reference link to a location in the current document	See the section “ Additional Resources ” for details. Refer to “ Title Formats ” in Chapter 1 for details.
Red text	Cross-reference link to a location in another document	See Figure 2-5 in the <i>Virtex-II Platform FPGA User Guide</i> .
Blue, underlined text	Hyperlink to a website (URL)	Go to http://www.xilinx.com for the latest speed files.

Tutorial Overview

This tutorial serves as an introduction to Xilinx embedded solutions, specifically the PowerPC™ 405 processor. In addition, the tutorial also covers using Xilinx Platform Studio (XPS) and Base System Builder. Finally, the design illustrates the enhanced floating-point library, the floating-point coprocessor, and the technique for adding custom or third-party IP to a PowerPC system.

This tutorial provides the steps necessary to create a PowerPC design using the Xilinx APU floating point coprocessor, downloading and the running the design on target hardware. The design is created using Xilinx Platform Studio and Base System Builder for the ML403 evaluation platform, using a software-based finite impulse response (FIR) filter. The floating point coprocessor is added to achieve code acceleration.

See [Appendix A, "Tutorial Configuration,"](#) for system implementation requirements and the ML403 board configuration.

The Tutorial Design Goal

The tutorial constructs a FIR filter to recover the source signal shown in [Figure 1-1](#), from the noisy signal shown in [Figure 1-2](#).

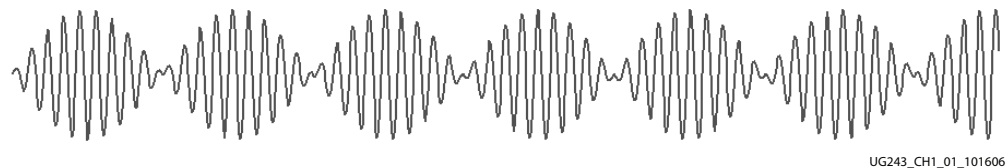


Figure 1-1: Source AM Signal

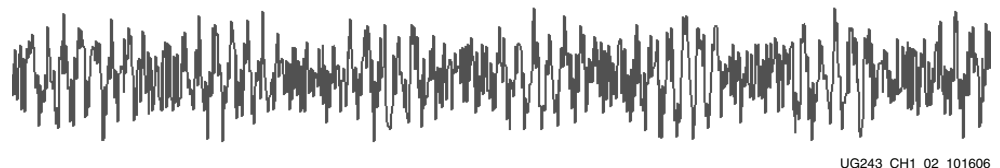


Figure 1-2: Source Signal Masked by Noise

Building and Running the Design

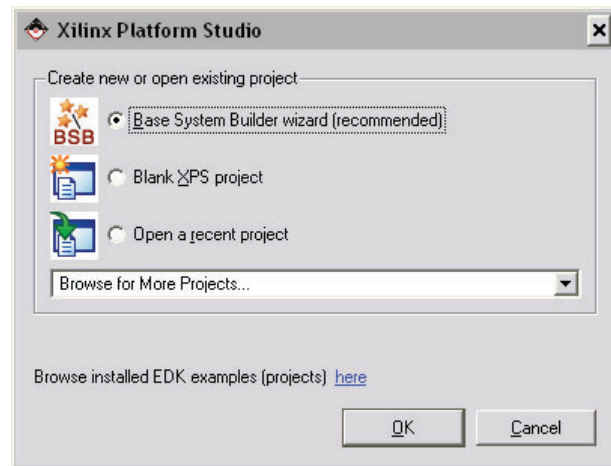
Install Reference Design Files into Target Directories

1. Set up the design environment and configure evaluation platform per [Appendix A, "Tutorial Configuration."](#)
2. Copy `xapp547.zip` to the Windows desktop. (This file can be downloaded from <http://www.xilinx.com/bvdocs/desfiles/xapp547.zip>.)
3. Double-click on `xapp547.zip` and unzip the contents to the Windows desktop:
 - ◆ `MathGraphX.exe` – A utility to display the lab data output.
 - ◆ `PPC_V4_FPU_FIR_Design.zip` – The FIR filter design.
 - ◆ `V4FX12_xbd.zip` – Board description file used for the lab.
 - ◆ `readme.txt`
4. Double-click on the file `PPC_V4_FPU_FIR_Design.zip` and unzip the contents of the file to directory `C:\`. The project files are placed in the following directories:
 - `C:\Xilinx_Design\V4FX_Labs\FPU_Lab_FIR_Filter\`
 - `C:\Xilinx_Design\V4FX_Labs\FPU_Lab_FIR_Filter_Part2\`
 - `C:\Xilinx_Design\V4FX_Labs\FPU_Lab_FIR_Filter_Part3\`
5. Double-click on the file `V4FX12_xbd.zip` and unzip the contents of the file to the EDK program directory (most likely `C:\EDK`). Unzipping the file installs the Virtex™-4 FX evaluation board description files for Base System Builder used by this tutorial.

Note: This design uses Base System Builder, which allows a design to be created specifically for a known evaluation platform. These board definition files are found in the install directory, typically located at `C:\EDK\boards`.

Build a Basic XPS Hardware and Software System Using Base System Builder

6. Start Xilinx Platform Studio (XPS) by selecting:
Start → **All Programs** → **Xilinx Platform Studio 9.2i** → **Xilinx Platform Studio**
7. The Base System Builder is used to create a new basic PowerPC design. Click **OK** to start the Base System Builder Wizard ([Figure 2-1](#)).



UG243_CH2_02_101606

Figure 2-1: Base System Builder Wizard

8. Click the Browse button and browse to the following location ([Figure 2-2](#)):
C:\Xilinx_Design\V4FX_Labs\FPU_Lab_FIR_Filter
The file name should be `system.xmp`. Click **Save**, then **OK**.



UG243_CH2_03_101606

Figure 2-2: Create New XPS Project Wizard

9. From the **Welcome Page**, select **I would like to create a new design** and click **Next**.
10. At the **Select Board Page**, select the target evaluation platform with the settings listed in [Table 2-1](#) and then click **Next**.

Table 2-1: Target Evaluation Platform Settings

Option	Settings
Board Vendor	Xilinx
Board Name	Virtex-4 ML403 with TFT
Board Revision	1

11. At the **Select Processor Page**, select **PowerPC** and click **Next**.
12. From the **Configure Processor Page**:
 - a. Do not change the reference clock frequency.
 - b. Change the processor clock frequency to **200 MHz**.
 - c. Confirm that the bus clock frequency is set to **100 MHz**.
 - d. Leave the remaining options at default and click **Next**.

Note: On this page, it is possible to add the soft floating-point unit to the design. However, to enable demonstration of software floating point operation, the FPU will be added at a later step.

13. The **Configure I/O Interfaces Page** dialogue windows display the peripherals that can be used with the chosen evaluation platform. Click **Next** as appropriate to advance to the next dialogue window. Select and deselect devices as shown in [Table 2-2](#).

Table 2-2: Proper Evaluation Platform Device Settings

Item No.	Item	Setting
1	RS232_Uart	Change baud rate to 115200
2	LEDs_4Bit	–
3	LEDs_Positions	Deselect
4	Push_Buttons_Position	Deselect
5	LCD_7Bit_GPIO	–
6	IIC_EEPROM	Deselect
7	SysACE_CompactFlash	Deselect
8	Ethernet_MAC	Deselect
9	SRAM_256Kx32	Deselect
10	FLASH_2Mx32	Deselect

14. Code and data are stored using memory connected to the IBM CoreConnect Peripheral Local Bus (PLB). At the **Add Internal Peripherals Page**, change the memory size of the XPS BRAM IF CNTLR from **8 KB** to **64 KB** and click **Next**.

15. The **Software Setup Page** dialog window shows what devices are used for standard input and standard output and enables sample application selection. Keep the default setting and have Base System Builder use the RS-232 interface for standard input and output.

Generate code for a sample peripheral self test:

- a. Uncheck **Memory Test**.
 - b. Leave **Peripheral Self Test** checked and click **Next**.
16. On the **Configure Peripheral Test Application Page**, the user can select where peripheral code, data, and stack/heap reside. The tutorial design only uses FPGA block RAM. Leave the current setting unchanged and click **Next**.
 17. The **System Created Page** displays a summary of the system created. The peripherals and memory are automatically assigned addresses. Click **Generate** to produce the design.
 18. The **Finish Page** summarizes all of the files automatically created. Click **Finish** to complete Base System Builder.
 19. If the **Next Step** dialog box appears, click **OK** to start using Platform Studio.
 20. To build the complete system, proceed to [“Build the System with the Test Application Using Platform Studio \(XPS\).”](#) This process takes approximately twelve minutes on a Pentium T2500, 2 GHz laptop.

Otherwise, to continue with a pre-built system, perform the following:

- a. From the menu, click **File** and select **Close Project**.
- b. From the menu, click **File** and select **Open Project**.
- c. Browse to the directory:

```
C:\Xilinx_Design\V4FX_Labs\FPU_Lab_FIR_Filter_Part2\
```

Click on `system.xmp` and click **Open**.

- d. Go to [Step 23](#) to download design to FPGA.

The hardware design for the system has been completed.

Build the System with the Test Application Using Platform Studio (XPS)

21. From the menu bar, build the libraries and board support packages by selecting:
Software → **Generate Libraries and BSPs**
This process takes about four minutes on a Pentium T2500, 2 GHz laptop.
22. Again from the menu bar, build the complete hardware and software system by selecting: **Hardware** → **Generate Bitstream**
This process takes about eight minutes on a Pentium T2500, 2 GHz laptop.
23. While the project is building, take a look at some of the test code and answer a few questions. Open the test program `TestAPP_Peripheral.c` as follows:
 - a. Click on the **Applications** tab in the main XPS window.
 - b. Underneath Project:TestApp_Peripheral, expand **Sources** by clicking on the **+** symbol to the left of **Sources**.
 - c. Double-click on the test program, `TestAPP_Peripheral.c`.

The main program starts near line 45.

Question 1: What is the program able to test?

Question 2: What is the program unable to test?

`TestAPP_Peripheral.c` calls functions in `xgpio_tapp_example.c`. Open this module by double-clicking on `xgpio_tapp_example.c` in the **Applications** window. The `XGpio_DiscreteWrite` function near line 193 is used to write out data to the LEDs on the board.

Question 3: What is the constant name defining the number of times a wait loop is executed to enable the LED to be visible? (Hint: Look below line 193).

Question 4: How many times will the wait loop be executed? (Hint: Look for a `#define` statement near the beginning of the program.)

24. After the bitstream has been built, download the design and software to the ML403 Virtex-4 FX evaluation platform by selecting from the menu bar:
Device Configuration → **Download Bitstream**
In about 30 seconds, the set of LEDs light in sequence. Push the CPU Reset button SW10 located at the lower left corner to start the program again.
The hardware and software systems build is complete.

Adding the Application Code and Configuring for Compilation

Remove the test application as follows:

25. Click on the **Applications** tab in the main XPS window.
Under **Project: TestApp_Peripheral**, expand **Sources** by clicking on the + symbol in front of **Sources**.
26. Remove the two files `TestAPP_Peripheral.c` and `xgio_tapp_example.c` by right-clicking on each file and selecting **Remove**.
27. Under **Project: TestApp_Peripheral**, expand **Headers** by clicking on the + symbol in front of **Headers**.
28. Remove the file `gpio_header.h` by right-clicking and selecting **Remove**.
29. Add source and header files by first double-clicking on **Sources** and browse to the directory `.\code`. Add all of the files found there by selecting them and clicking **Open**.

Next, double-click on **Headers** and browse to the directory `.\code`. Add all of the files found there by selecting them and clicking **Open**. Figure 2-3 shows the added source and header files.

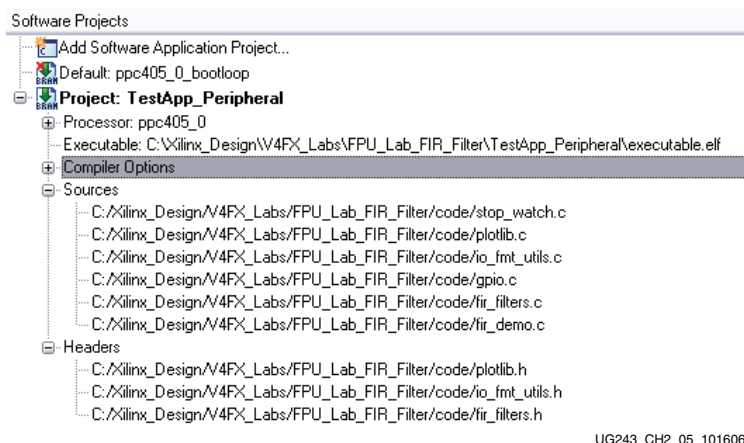


Figure 2-3: Added Source and Header Files

Running the Software-Only Design

30. Compile and link the software, update the bitstream, and download to the board by selecting from the menu bar **Device Configuration** → **Download Bitstream**.
31. Start the MathGraphX utility by clicking on the MathGraph icon found on the desktop. If the system uses a port other than COM-1 for the serial port, select the correct COM port in MathGraphX via **File Tools** → **Options**.

Look at the MathGraphX window (Figure 2-4) and note how the software-based FIR filter cleans the noisy input signal.

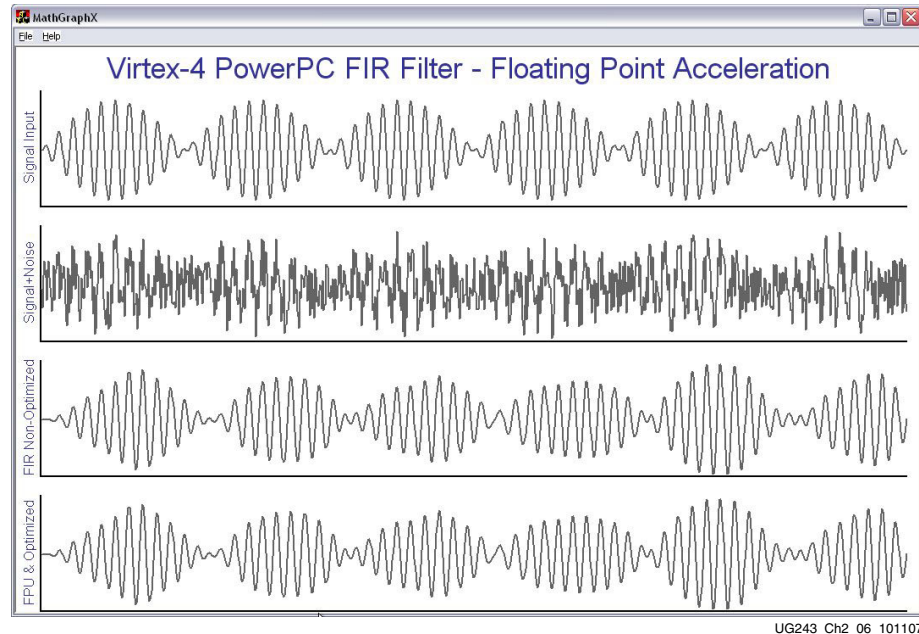


Figure 2-4: MathGraphX Window

Question 5: What is the SW Megaflops value displayed on the board LCD?

32. Accelerate the system by enabling caches:
 - a. In Platform Studio, if not already selected, click on the **Applications** tab in the main XPS window.
 - b. Under **Project: TestApp_Peripheral**, expand **Sources** by clicking on the **+** symbol in front of **Sources**.
 - c. Double-click on the file `..\code\fir_demo.c` to open the main program file.
 - d. In the right-hand window, scroll down to near line 307 and enable the caches by uncommenting the following lines:


```
XCache_EnableDCache(0x00000001);
XCache_EnableICache(0x00000001);
```
 - e. Save the file by selecting: **File** → **Save**.
 - f. Compile and link the software, update the bitstream, and download to the board by selecting from the menu bar: **Device Configuration** → **Download Bitstream**
 - g. Observe the MathGraphX window to see the performance improvement achieved by enabling the cache.

Question 6: What is the software performance in MFLOPS after enabling cache?

Question 7: How much faster is the software?

Using the New Floating Point Performance Library

The new floating point performance library included with EDK 8.2 and subsequent versions enables additional software acceleration.

33. Enable the floating point performance library as follows:
 - a. Click on the **Applications** tab in the main XPS window.
 - b. On the left side, under Project: TestApp_Peripheral, double-click on **Compiler Options**.
 - c. Select the **Paths and Options** tab. In the box **Other Compiler Options to Append**, insert the compiler option to enable the floating-point performance library:

```
-mppcperflib
```

Click **OK**.

- d. Compile and link the software, update the bitstream, and download to the board by selecting **Device Configuration** → **Download Bitstream** on the Menu Bar.

Question 8: What is the software Megaflops with the performance library?

Question 9: What is the performance improvement over the original library?

34. Remove the floating point performance library compiler option as follows:
 - a. Click on the **Applications** tab in the main XPS window.
 - b. On the left side, under **Project: TestApp_Peripheral**, double-click on **Compiler Options**.
 - c. Select the **Paths and Options** tab. In the box **Other Compiler Options to Append**, remove the option:

```
-mppcperflib
```

- d. Click **OK**.

Building the software FIR filter is complete.

Adding and Enabling the Hardware-Based Floating Point Unit

35. Add the hardware-code accelerator and enabling the APU Interface:
 - a. Near the center bottom of the Platform Studio Window, click on the **System Assembly** tab to select the **System Assembly** window.
 - b. On the left side, click on the **IP Catalog** tab.
 - c. Look down the list and click on the + symbol to the left of **Arithmetic**. The apu_fpu floating-point unit appears.
 - d. Right-click on **apu_fpu** and select **ADD IP**.
 - e. In the **System Assembly View**, click on the + symbol to the left of the **apu_fpu_0** core to expand and view its bus connections.
 - f. Click on **No Connection** to the right of the **SFCB** connection label, and using the drop down list, select **New Connection**. The connection name changes to **fcv_v10_0**, and the new FCB bridge appears as a vertical line to the left of the IP list.
 - g. Click on the + symbol to the left of the **ppc405_0 core** to expand and view its bus connections.

- h. Click on the non-solid rectangle to the left of the **ppc405_0** core at the **MFCB** bus label. The rectangle turns solid indicating completion of the bus connection.
 - i. Right-click on **ppc405_0** and select **Configure IP...** to open the PowerPC properties dialog.
 - j. Select the **APU** tab.
 - k. Change the **APU Controller Configuration Register Initial Value** from the large number to 0b1 to enable the APU interface, and click **OK**.
36. Connect the APU interface port clock and reset:
 - a. In the **System Assembly** page, towards the top-center, select the **Ports** tab.
 - b. Click on the **+** symbol located in front of **fcv_v10_0**. At **FCB_CLK**, click on **No Connection**, and using the drop-down list, connect the clock by selecting the net name **proc_clk_s**.
 - c. At **SYS_RST**, click on **No Connection**, and using the drop down list, connect the reset by selecting the net name **sys_bus_reset**.
37. Connect the FPU clock:
 - a. Click on the **+** symbol located in front of **apu_fpu_0**. At **FPU_CLK**, click on **No Connection**, and using the drop-down list, connect the clock by selecting the net name **sys_clk_s**.
38. Set the FPU type to single precision and the lite version to provide a smaller design (the lite version has hardware add/subtract, multiply):
 - a. To select the hardware version, right-click on **apu_fpu_0**, and select **Configure IP...** to open the FPU properties dialog.
 - b. At the right side of the window, change the **FPU Features** from **FULL** to **Lite**, and click **OK**.
39. To build the complete system, proceed to "[Building and Running the System.](#)" This process takes about fourteen minutes on a Pentium T2500, 2 GHz laptop.
40. To continue with a pre-built system, perform the following:
 - a. From the menu bar, click on **File** and select **Close Project**.
 - b. From the menu bar click, on **File** and select **Open Project**, and browse to the directory:

```
C:\Xilinx_Design\V4FX_Labs\FPU_Lab_FIR_Filter_Part3\
```
 - c. Select the file `system.xmp`, and click **Open**, and continue to [Step 41](#).

Building and Running the System

41. Build the new hardware and software systems, update the bitstream, and download to the board by selecting on the Menu Bar:

Device Configuration → **Download Bitstream**

Note: The `-mfpu=sp_lite` compiler switch is automatically generated and tells the compiler to generate code targeting the floating point unit. `#define HAVE_XFPU` is generated by the compiler in response to the `-mfpu` compiler switch. This switch is used by the code in `fir_demo.c` to enable an additional test case demonstrating optimized floating-point code.

42. Answer the following (refer to the source code in `fir_demo.c` near line 65):

Question 10: What is the size of the input data array?

Question 11: How many filter taps are there?

Question 12: A FIR filter computation consists of multiplying and accumulating all of the input values across all of the filter taps. How many multiplies and accumulates are there?

There are two FIR computation routines. The first, `fir_basic`, is the FIR filter written in its standard form. The second, `fir_8reg`, is optimized to use the floating-point registers and maximize utilization of the FPU pipeline. To enable efficient utilization of the pipeline in the FPU, it exploits the following techniques:

- ◆ Loop unrolling
- ◆ Utilization of array pointers instead of indexes
- ◆ Reorganization of the code to eliminate FPU pipeline dependencies
- ◆ Holding of a partial set of coefficients in the FPU register file

43. If the Design License Status dialog window appears, click **OK**.

44. Once the design has been downloaded and run, answer the following:

Question 13: What is the performance of the non-optimized FIR in MFLOPS?

Question 14: What is the performance of the optimized FIR in MFLOPS?

Question 15: How much faster is the hardware FPU than the software performance library (Question 8)?

Building a FIR filter with floating point coprocessor is complete.

Answers

Question 1 – The program tests the I/O interfaces driving the LEDs and LCD on the board.

Question 2 – The program is unable to test the RS232_Uart because it has been selected as the STDOUT device.

Question 3 – LED_DELAY.

Question 4 – 1,000,000 times.

Question 5 – 0.12 MFLOPS.

Question 6 – 1.06 MFLOPS.

Question 7 – 8.3 times faster.

Question 8 – 2.81 MFLOPS.

Question 9 – 2.7 times faster.

Question 10 – Line 67: #define IMP_SIZE 984: There are 984 data input elements.

Question 11 – Line 66: #define NTAPS 64: There are 64 FIR filter taps.

Question 12 – There are $984 \times 64 = 62976$ multiplies and 62976 adds. There are a total of 125952 floating point operations.

Question 13 – 9.85 MFLOPS.

Question 14 – 36.4 MFLOPS.

Question 15 – The HW FPU is 13 times faster than the software floating point execution.

Tutorial Configuration

Hardware Requirements

- Xilinx ML403 Virtex™-4 FX Evaluation Platform
- Xilinx JTAG Platform Cable USB or Parallel IV Cable
- Female-to-Female Serial Data Cable

Software Requirements

- ISE 9.2 Service Pack 3
- XPS 9.2

System Configuration

- ISE installed
- Xilinx Platform installed
- JTAG Platform Cable USB or Parallel IV cable is installed and connected to the evaluation board
- Power cable connected to the evaluation board
- Serial data cable connected between a PC and the evaluation board
- Evaluation board is turned on

