Implementing a Virtex-4 FX C-to-HDL Hardware Coprocessor Accelerator in a PowerPC Design

Design Guide

UG096 (v2.0) March 9, 2007

XILINX®

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
12/16/05	1.0	Initial Xilinx release.
03/09/07	2.0	• “About This Guide,” page 7: Changed introductory text. • “PowerPC Coprocessor Accelerator,” page 9: Changed title from “Implementing a Virtex-4 FX PowerPC System with a C-to_HDL Hardware Coprocessor Accelerator.” • Section “Introduction,” page 9: Changed text in the section previously titled “Design Challenge.” • Figure 1 through Figure 11 updated. • Section “Implementing the C-to-HDL Hardware Coprocessor Accelerator,” page 11: Changed heading from “Building and Running a Design”. • Section “Installing Reference Design Files into Target Directories,” page 11: Changed title from “Installing the Design Files and Configuring the Environment” and changed the text in the section. Added bullet for readme.txt file.

Date	Version	Revision
03/09/07	2.0 (*cont'd*)	• Section "Building a Basic XPS Hardware and Software System," page 12: Changed text, figures, and tables. • Table 2: Changed line 8 setting from "Deselect" to "Keep the currently selected setting." • Section "Building the System with the Test Application using Platform Studio (XPS)," page 16: Changed text, figures, and tables. • Section "Adding the Hardware Code Accelerator," page 17: Moved from old location and changed text. • Section "Connecting the Clocks for the Display Controller and Coprocessor," page 18: Changed title from "Configuring the Clocks for the Display Controller and Coprocessor" and updated text and figures. • Former section "Connecting the TFT Display Controller Clock": Deleted. • Former section "Configure the User Constraint File (UCF) for System Timing": Deleted. • Section "Connecting Clock and Reset Signals to the APU Interface Port," page 21: Added. • Section "Adding the Application Code and Configuring for Compilation," page 21: Changed title from "Adding the Application Code" and changed text. • Section "Building and Running the System," page 24: Changed title from "Building and Running the System with a Coprocessor Accelerator" and changed text. • Section "Summary," page 25: Added. • Section "Answers," page 25: Changed text. • Appendix, "Tutorial Configuration": Changed text and deleted figure.

Table of Contents

About This Guide

This guide provides a complete walk-through of the steps necessary to implement a PowerPC® processor system on a Virtex™-4 FX FPGA. The reference design described in this guide serves as an introduction to Xilinx embedded solutions, specifically the PowerPC processor. It also covers the Xilinx Platform Studio™ tool and the included Base System Builder™ wizard. Finally, the reference design illustrates how to add custom or third-party IP to the PowerPC APU interface.

Guide Contents

This guide contains the following sections:

- "PowerPC Coprocessor Accelerator" describes how to build and run the Virtex-4 FX reference design.
- "Tutorial Configuration" summarizes the hardware, software, and configuration requirements for the reference design.

References

These documents provide supplemental material useful to this guide:

1. DeAlmo, Joe. *Applications of Fractal Geometry.* http://hypatia.math.uri.edu/~kulenm/honprsp02/
2. Virtex-4 ML403 Embedded Platform. http://www.xilinx.com/ml403
3. XAPP901, *Accelerating Software Applications Using the APU Controller and C-to-HDL Tools*.
4. Pellerin, David and Scott Thibault. 2005. *Practical FPGA Programming in C*. Prentice Hall. http://www.impulsec.com/practical/index.html
5. UG070, *Virtex-4 User Guide*.
6. Ansari, Ahmad, Peter Ryser, and Dan Isaacs. *Accelerated System Performance with APU-Enhanced Processing.* http://www.xilinx.com/publications/xcellonline/xcell_52/xc_v4acu52.htm
7. UG018, *PowerPC 405 Processor Block Reference Guide*.
8. XAPP717, *Accelerated System Performance with the APU Controller and XtremeDSP Slices*.
9. UG080, *ML40x Evaluation Platform User Guide*.
10. Bodenner, Ralph. *Fixed-Point Arithmetic in Impulse C.* http://www.impulsec.com/IATAPP106_FIXEDPT.pdf

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*
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`** *design_name*
	References to other manuals	See the *Development System Reference Guide* for more information.
	Emphasis in text	If a wire is drawn so that it overlaps the pin of a symbol, the two nets are *not* connected.
Square brackets [ ]	An optional entry or parameter. However, in bus specifications, such as **`bus[7:0]`**, they are required.	**`ngdbuild`** `[`*`option_name`*`]` *`design_name`*

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 *Virtex-II Platform FPGA User Guide.*
Blue, underlined text	Hyperlink to a website (URL)	Go to http://www.xilinx.com for the latest speed files.

PowerPC Coprocessor Accelerator

Introduction

Fractal texturing is a technique used in image rendering to create imagery with an organic appearance. The Mandelbrot image generation algorithm is one example of fractal texturing.

This guide serves as an introduction on how to build a PowerPC® system with a coprocessor accelerator capable of generating Mandelbrot images using Xilinx Platform Studio™ (XPS). This system demonstrates image generation (Figure 1) in both of the following:

- Software on a 200 MHz PowerPC processor
- Software with a hardware code accelerator

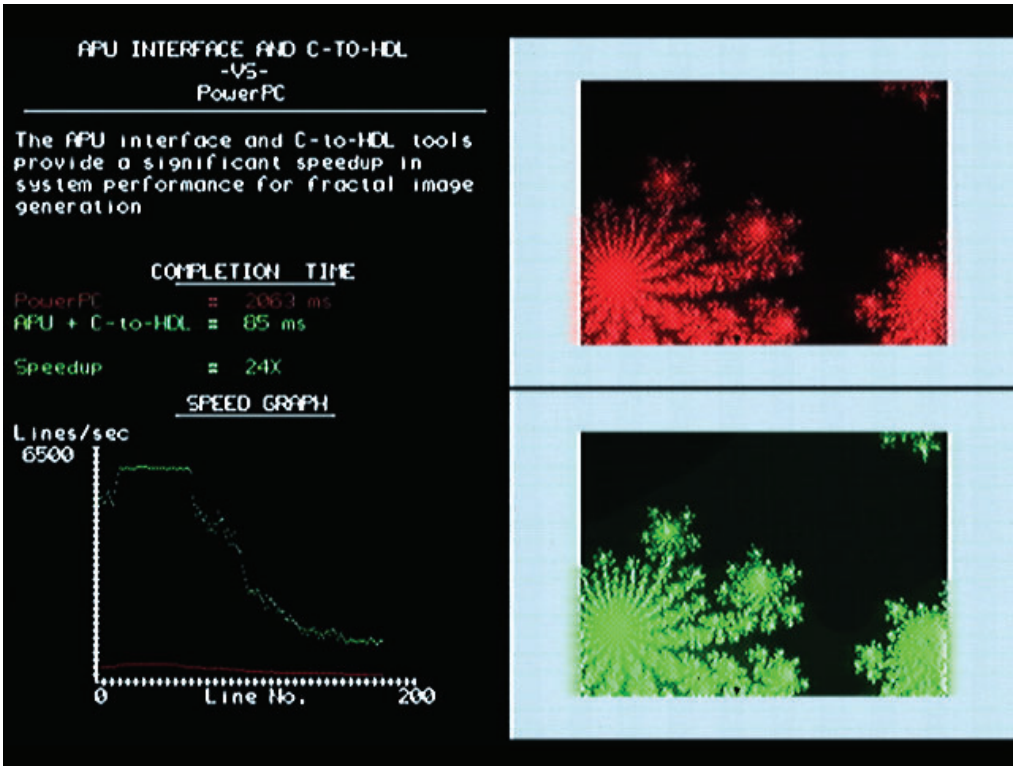
The hardware code accelerator is a type of coprocessor that is tightly coupled to the PowerPC instruction pipeline through the Auxiliary Processor Unit (APU) interface. This coprocessor can accelerate code 24 times faster, which is equivalent to 24 times 200 MHz (that is, 4.8 GHz). As a result, the PowerPC processor with hardware code accelerator can provide performance equal to that of today's multi-GHz processors.

For system implementation requirements and the target board configuration, see "Tutorial Configuration.".

Note: Throughout this guide appear questions such as the following example. Answers to the questions appear in the "Answers" section.

Example –

Question 1: What does this program do?



UG096_01_011707

Figure 1: **Mandelbrot Image Generation**

Implementing the C-to-HDL Hardware Coprocessor Accelerator

This section includes the following subsections:

Installing Reference Design Files into Target Directories

This section explains how to install reference design files into target directories.

1. Set up the design environment and configure the Evaluation Platform according to "Tutorial Configuration.".
2. Copy `PPC_V4_C2HDL_Lab.zip` to the Windows desktop.
3. Double-click on `PPC_V4_C2HDL_Lab.zip` and unzip the following contents to the Windows desktop:
   - `PPC_V4_C2HDL_Design.zip` – The FIR filter design
   - `V4FX12_xbd.zip` – Board description file to be used by EDK readme.txt – an information update file
   - `readme.txt` – Readme file for this design
4. Double-click on the file `PPC_V4_C2HDL_Design.zip` and unzip the file contents to directory `C:\`. The project file are placed in the following directories:

   ```
   C:\Xilinx_Design\V4FX_Labs\C2HDL_Lab _Xilinx\
   C:\Xilinx_Design\V4FX_Labs\C2HDL_Lab_Xilinx_Part2\
   C:\Xilinx_Design\V4FX_Labs\C2HDL_Lab_Xilinx_Part3\
   ```

5. Double-click on the file `V4FX12_xbd.zip` and unzip the file contents to the EDK program directory (typically `C:\EDK`). This step installs the Virtex™- FX evaluation board description files for Base System Builder™.

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`.

Building a Basic XPS Hardware and Software System

This section explains how to build a Basic XPS Hardware and Software System using Base System Builder.

6. Start 'Xilinx Platform Studio' (XPS) by selecting the following, as shown in Figure 2:

   Start → All Programs → Xilinx Platform Studio 8.2i → Xilinx Platform Studio

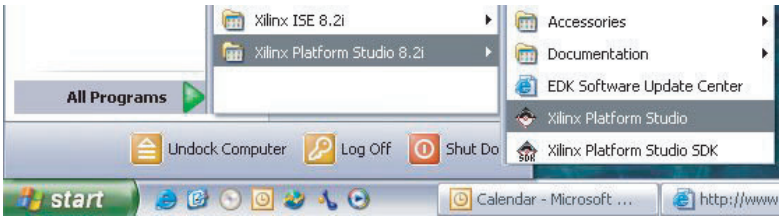


Figure 2: **Starting Xilinx Platform Studio**

7. The Base System Builder is used to create a new basic PowerPC system design. Select **Base System Builder Wizard [recommended]** and click **OK** to start the Base System Builder Wizard (Figure 3).

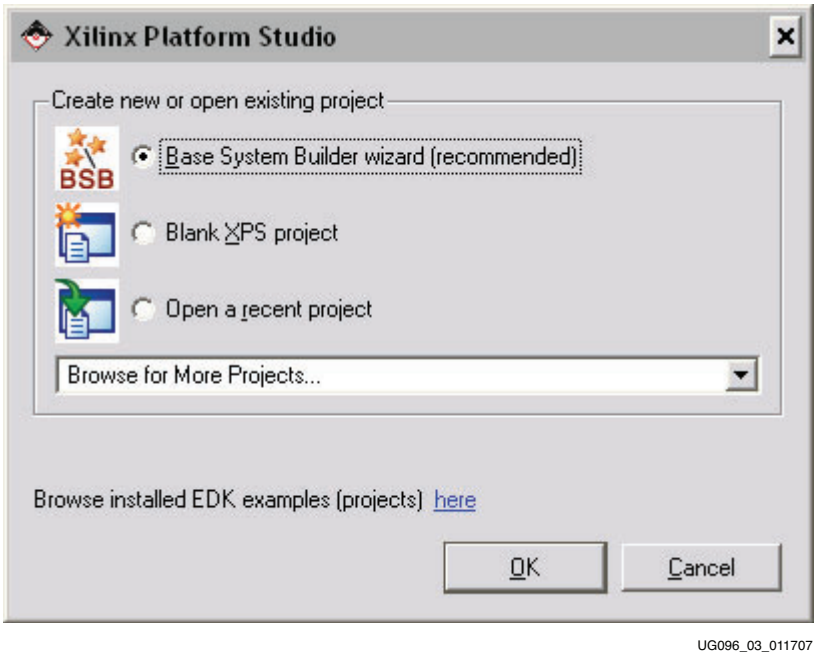


Figure 3: **Xilinx Platform Studio – Base System Builder Wizard**

8. Click the **Browse...** button and browse to the following project directory location (Figure 4):

   `C:/Xilinx_Design/V4FX_Labs/C2HDL_Lab _Xilinx`

   The file name is `system.xmp`. Click **Save** and then click **OK**.

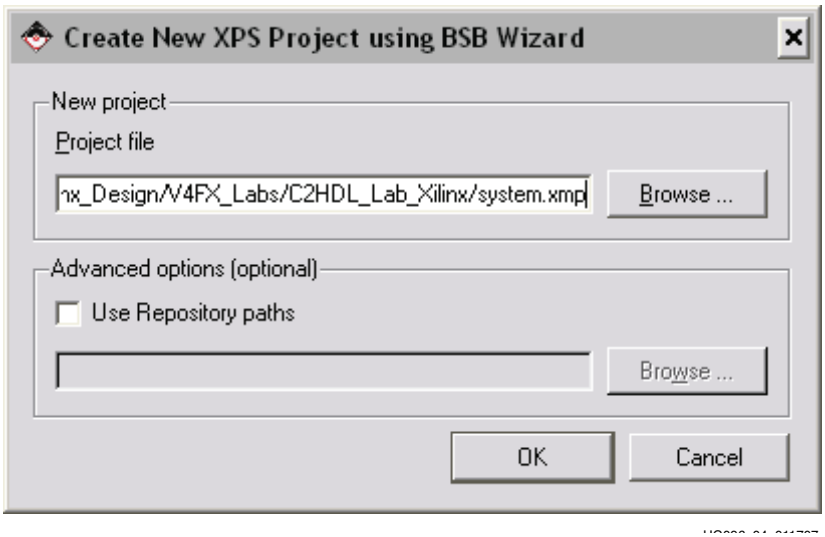


Figure 4: **Create New XPS Project Wizard Using BSB Wizard**

9. From the **Welcome Page**, select **I would like to create a new design** and click **Next**.
10. At the **Select Board Page**, select **We will be using an existing Evaluation Platform**.
11. Select the target Evaluation Platform with the settings listed in Table 1 and click **Next**:.

Table 1: **Target Evaluation Platform Settings**

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

12. At the **Select Processor Page**, select the **Select PowerPC processor** and click **Next**.
13. From the **Configure Processor Page**, do the following:
    a. Confirm that the **Reference clock frequency** is set to **100 MHz**.
    b. Change the **Processor clock frequency** to **200 MHz**.
    c. Confirm that the **Bus clock frequency** is set to **100 MHz**.
    d. Select **Cache Setup ENABLED**.
    e. Leave the remaining options at default and click **Next**.

14. The **Configure I/O Interfaces Page** dialog windows display the peripherals that can be used with the chosen evaluation platform. As appropriate, click **Next** to advance to the next dialog window. Select and deselect devices as listed in Table 2.

Table 2: **Evaluation Platform Device Settings**

Item Number	Item	Setting
1	RS232_Uart	Keep the default baud rate.
2	LEDs_4Bit	Keep the currently selected setting.
3	LEDs_Positions	Deselect
4	Push_Buttons_Position	Deselect
5	LCD_7Bit_GPIO	Keep the currently selected setting.
6	IIC_EEPROM	Deselect
7	SysACE_CompactFlash	Deselect
8	DDR_SDRAM_64Mx32	Keep the currently selected setting.
9	plb_tft_cntlr_ref_0 (Video Interface)	Keep the currently selected setting.
10	Ethernet_MAC	Deselect
11	SRAM_256Kx32	Deselect
12	FLASH_@Mx32	Deselect

15. 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 PLB BRAM IF CNTLR from **16KB** to **64KB** and click **Next**.
16. **Cache Setup Page**: Instructions and data can be cached or not cached for each type of memory. Enable Cache for all memory (including by checking all six boxes and then Click **Next**.

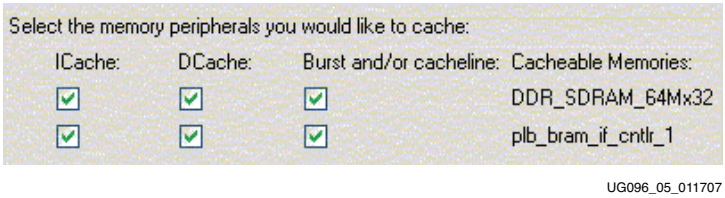


Figure 5: **Cache Setup Page**

17. The **Software Setup Page** dialog window does two things – it shows what devices are used for standard input and standard output, and it enables sample application selection. Keep the default setting, which uses the RS-232 interface for standard input and output. Generate code for a sample Peripheral self test as follows:
    a. Uncheck **Memory Test**.
    b. Leave **Peripheral selftest** checked and click **Next**.

18. On the **Configure Peripheral Test Application Page,** use the drop-down menus for Instruction, Data, and Stack/Heap (Figure 6) to change the location for each to `plb_bram_if_cntlr_1`, and click **Next**.

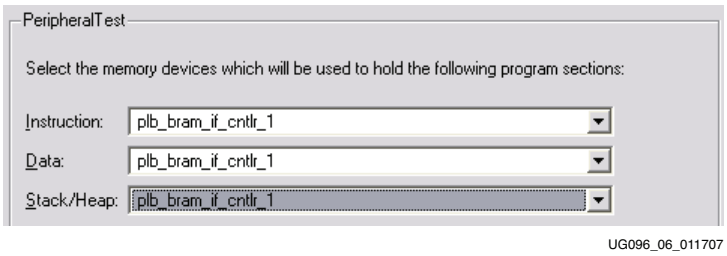


Figure 6: **Peripheral Test Settings**

19. 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.
20. The **Finish Page** summarizes all of the files automatically created. Click **Finish** to complete Base System Builder.
21. If the **Next Step** dialog box appears, click **OK** to start using Platform Studio and continue with the next step.
22. To build the complete system, proceed to "Building the System with the Test Application using Platform Studio (XPS)". This process takes approximately 8 minutes on a Pentium T2500, 2 GHz laptop.

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

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

       ```
       C:\Xilinx_Design\V4FX_Labs\C2HDL_Lab_Xilinx_Part2\
       ```

    d. Click on `system.xmp` and click **Open**.
    e. If the **Block Diagram** view displays, click on the **System Assembly Tab**. Otherwise, continue at "Building the System with the Test Application using Platform Studio (XPS)", step 26.

Building the System with the Test Application using Platform Studio (XPS)

This section explains how to build a Basic XPS Hardware and Software System with the test application by using Platform Studio (XPS).

23. From the menu bar, build the libraries and board support packages by selecting the following:

    Software → Generate Libraries and BSPs

    This process takes less than 30 seconds.

24. From the menu bar, build the complete hardware and software system by selecting the following:

    Hardware → Generate Bitstream

    This process takes approximately 8 minutes on a Pentium T2500, 2 GHz laptop.

    While the project builds, use the next steps to examine the test code and learn about what it does.

25. Open the test program `TestAPP_Peripheral.c` as follows:
    a. Click on the **Applications** tab in the main XPS window.
    b. Under **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 47.

       Question 1: What does the program do?

       Question 2: What is the program unable to do?

    d. The test program `TestAPP_Peripheral.c` calls functions in `xgpio_tapp_example.c`. Open this module by navigating to the **Applications** window and double-clicking on `xgpio_tapp_example.c`. 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 will be executed to enable the LED to be visible? (Hint: Look below line 188).

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

26. After the bitstream is built, download the design and peripheral test software to the ML403 evaluation platform by selecting the following from the menu bar.

    Device Configuration → Download Bitstream

    In less than a minute, the set of LEDs in the lower right corner of the board light in sequence.

27. To restart the program, push the CPU Reset button **SW10** located at the lower left corner of the board.

    At this time, the hardware and software systems build is complete.

Adding the Hardware Code Accelerator

This section explains how to add the hardware code accelerator.

28. To add the hardware code accelerator and enable the APU interface, do the following:
    a. Near the center bottom of the XPS window, click on the **System Assembly View1** tab.
    b. Click on the **IP Catalog** tab.
    c. To the left of **Project Repository**, look down the list and click on the **+** symbol, which displays the two custom devices created for this design.

       Note: The Base System Builder wizard added the display controller IP earlier when the wizard was run.

    d. Click on and drag the coprocessor IP, `apu_mand`, to the right and drop it in the open space in the **System Assembly** view.
    e. To the left of `apu_mand` core, click on the **+** symbol to expand and view its bus connections.
    f. Under `apu_mand_0`, click on the bus **SFCB** to highlight the line.
    g. To the right of **SFCB** connection label, click on **No Connection** and use the drop-down menu to change the Net name to **New Connection**. The connection name changes to **fcb v10_0**, and the new FCB bus in blue appears to the left of the IP list.

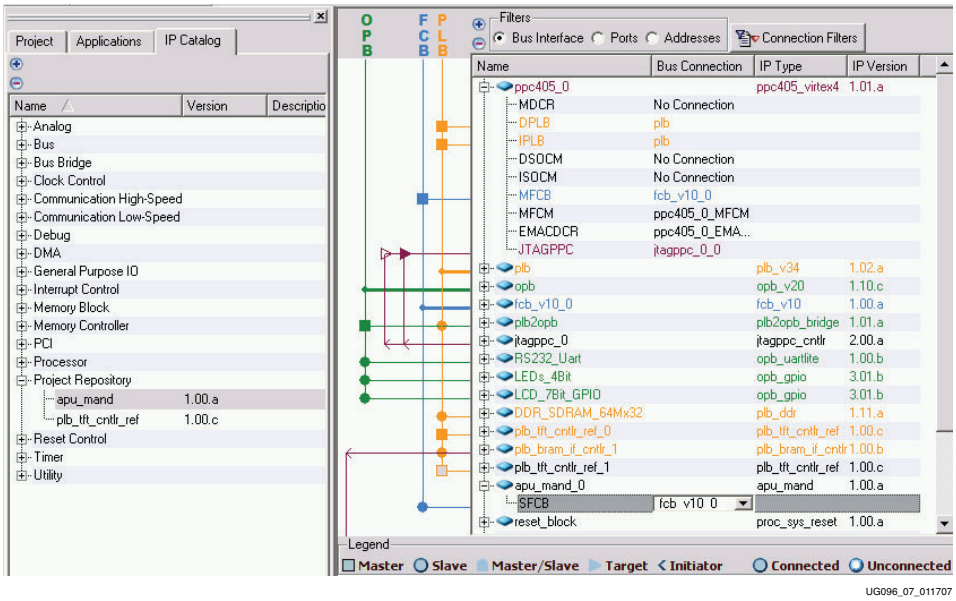


Figure 7: **System Assembly View**

h. To the left of the **ppc405_0 core**, click on the **+** symbol to expand and view its bus connections.

i. To the left of the **ppc405_0 core** at the MFCB bus label, click on the non-solid blue rectangle, after which the blue rectangle turns solid, indicating completion of the bus connection.

j. Ensure the completed connections match those in Figure 7.

k. Right click on **ppc405_0** and select **Configure IP...** to open the PowerPC processor properties dialog.

l. Select the **APU** tab.

m. Change the **APU Controller Configuration Register Initial Value** from the large number to `0b1` to enable the APU interface and click **OK**.

Connecting the Clocks for the Display Controller and Coprocessor

This section explains how to connect the clocks for the display controller and coprocessor.

29. To configure the Digital Clock Manager (DCM) for the display controller, implement a 25 MHz display buffered clock by dividing the input clock of 100 MHz by 4 as follows.

    a. In the System Assembly Window, at **dcm_1**, double-click on **dcm_module** (Figure 8).

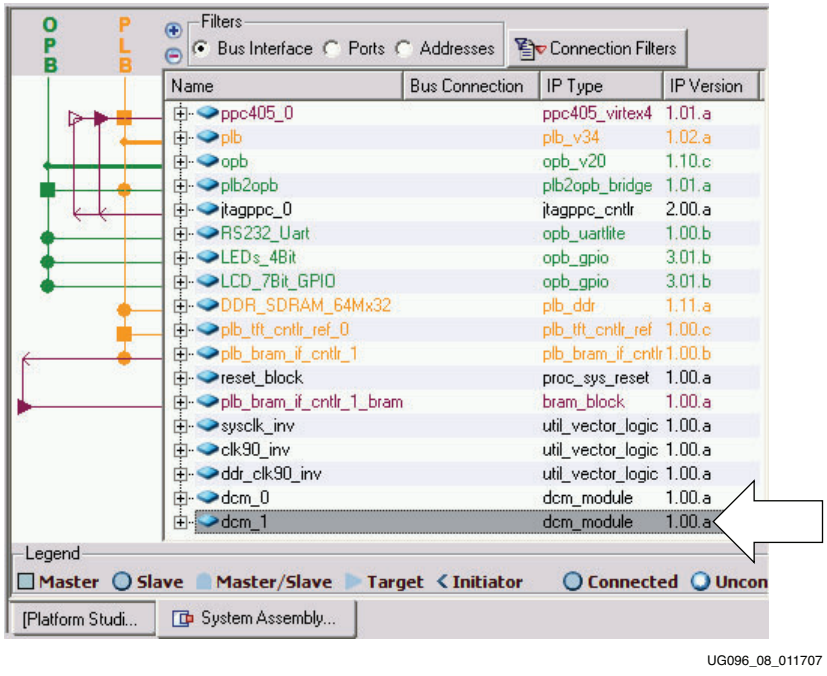


Figure 8: **dcm_1: Digital Clock Manager**

b. In the configuration window that appears, scroll down to **CLKDV Divisor,** click on the value, and use the drop-down menu to change the value to **4.0** (Figure 9).

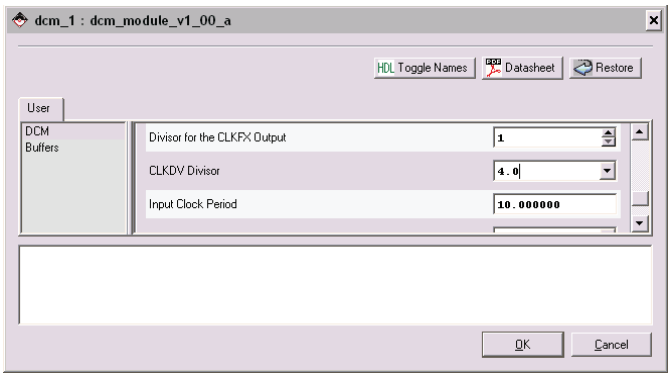


Figure 9: **Configuration Window**

c. Under the **User** tab in the same window, click on **Buffers**.

d. Scroll down to **Insert a BUFG for CLKDV**, and use the drop-down menu to change the value to **TRUE**. (Figure 10).

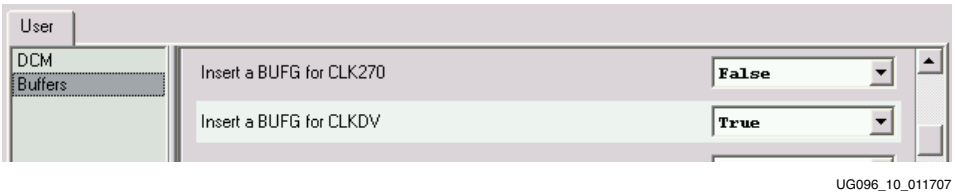


Figure 10: **Configuration Window: Insert a Clock Buffer**

e. Click **OK** to complete the changes.

30. To configure the DCM for the coprocessor, implement a 50 MHz buffered clock by dividing the input clock of 100 MHz by 2 as follows:

a. In **System Assembly View1** at **dcm_0** double-click on **dcm_module**.

b. In the **Configuration** window that appears, scroll down to **CLKDV Divisor** and confirm that the value is set to **2.0**.

c. Under the **User** tab in the same window, click on **Buffers**.

d. Scroll down to **Insert a BUFG for CLKDV**, and use the drop-down menu to change the value to **TRUE**.

e. Click **OK** to complete the changes.

31. To assign net names to the new clock ports, do the following:
    a. In **System Assembly View1**, under **Filters**, select **Ports**.
    b. In **System Assembly View1/Ports**, locate the second column (labeled **Net)**. Drag the column divider line to the right to provide a column width of approximately 2 cm.
    c. Click on the **+** symbol to the left of **dcm_1**.
    d. Click on the port named **CLKDV** to highlight the line and click on **No Connection**.
    e. Use the drop-down menu to assign the Net name to **dcm_1_CLKDV**.
    f. Press **Enter** on the keyboard to complete the connection.
    g. Click on the **+** symbol to the left of **dcm_0**.
    h. Click on the port named **CLKDV** to highlight the line and click on **No Connection**.
    i. Use the drop-down menu to assign the Net name to **dcm_0_CLKDV**.
    j. Press **Enter** on the keyboard to complete the connection.
32. To connect the coprocessor clocks, do the following:
    a. Scroll to and click on the **+** symbol to the left of **apu_mand_0**.
    b. Click on the port named **co_clk** to highlight the line, and click on **No Connection**.
    c. Use the drop-down menu to connect the clock by selecting the Net name **dcm_0_CLKDV**.
    d. Press **Enter** on the keyboard to complete the connection.
    e. With the same instance **apu_mand_0**, click on the port apu_clk to highlight the line, and click on **No Connection**.
    f. Use the drop-down menu to connect the clock by selecting the Net name **sys_clk_s**.
    g. Press **Enter** on the keyboard to complete the connection.
33. To connect the TFT Display controller clock, do the following:
    a. Scroll to and click on the **+** symbol to the left of **plb_tft_cntlr_ref_0**.
    b. Click on the port named **SYS_tftClk** to highlight the line, and click on **No Connection**.
    c. Use the drop-down menu to connect the clock by selecting the Net name **dcm_1_CLKDV.**
    d. Press **Enter** on the keyboard to complete the connection.

Connecting Clock and Reset Signals to the APU Interface Port

This section explains how to connect the clock and reset signals to the APU interface port.

34. To connect the APU interface port clock and reset signals, do the following:
    a. Scroll to and click on the **+** symbol to the left of **fcb_v10_0**.
    b. Click on the port named **FCB_CLK** to highlight the line, and click on **No Connection**.
    c. Use the drop-down menu to connect the clock by selecting the Net name **sys_clk_s**.
    d. Press **Enter** on the keyboard to complete the connection.
    e. Click on the port **SYS_RST** to highlight the line, and click on **No Connection**.
    f. Use the drop-down menu to connect the clock by selecting the Net name **sys_bus_reset**.
    g. Press **Enter** on the keyboard to complete the connection.
35. Before going to the next section, carefully check the work performed in this section.

Adding the Application Code and Configuring for Compilation

This section explains how to remove the test application code, add the application code, and set compiler options.

36. Remove the test application as follows:
    a. Click on the **Applications** tab in the main XPS window.
    b. Under **Project: TestApp_Peripheral**, expand **Sources** by clicking on the **+** symbol to the left of **Sources**.
    c. Remove the files `TestAPP_Peripheral.c` and `xgio_tapp_example.c` by right clicking on each and selecting **Remove**.
    d. Under **Project: TestApp_Peripheral**, expand **Headers** by clicking on the **+** symbol to the left of **Headers**.
    e. Remove the file `gpio_header.h` by right clicking on it and selecting **Remove**.

37. Add source and header files (Figure 11) as follows:
    a. Double-click on **Sources** and browse to the directory `.\code`.
    b. Add all of the files found there by selecting them and clicking **Open**.
    c. Double-click on **Headers** and browse to the directory `.\code`.
    d. Add all of the files found there by selecting them and clicking **Open**.

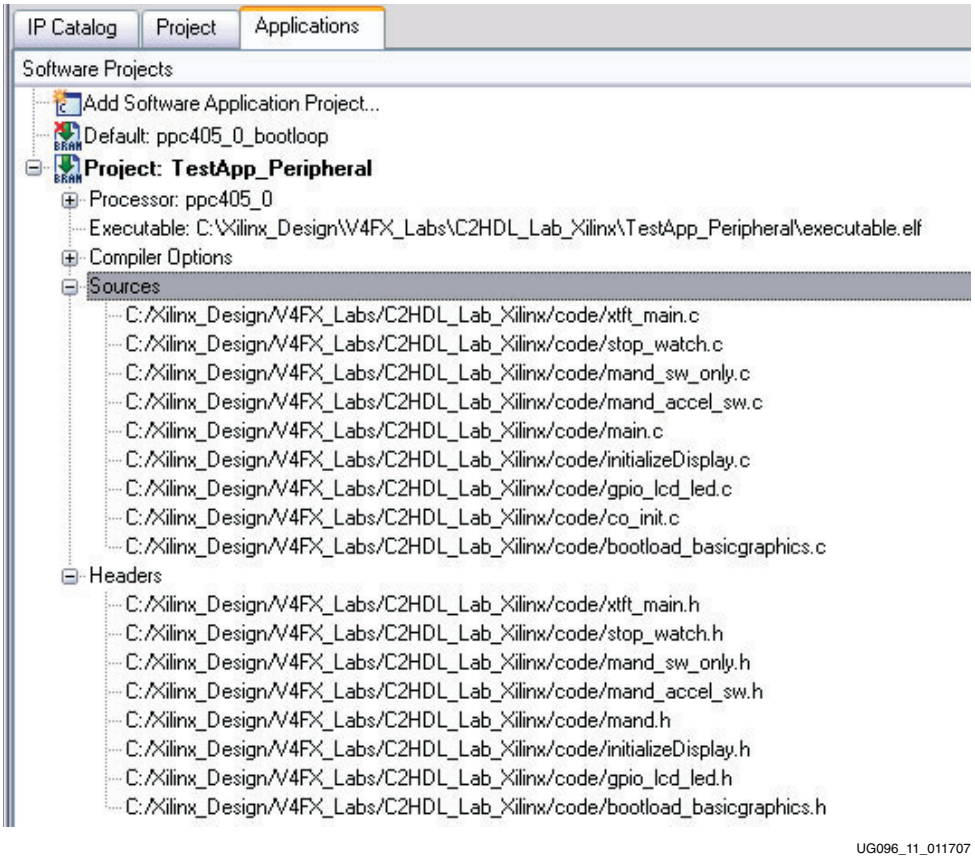


Figure 11: **Add Source and Header Files**

38. Change the compiler options as follows:
    a. Under **Project: TestApp_Peripheral**, double-click on **Compiler Options**, after which the dialog box shown in Figure 12 displays.
    b. Check the box **Use Default Linker Script**.
    c. Set Stack Size to **1024**.
    d. Set Heap Size to **2048**.

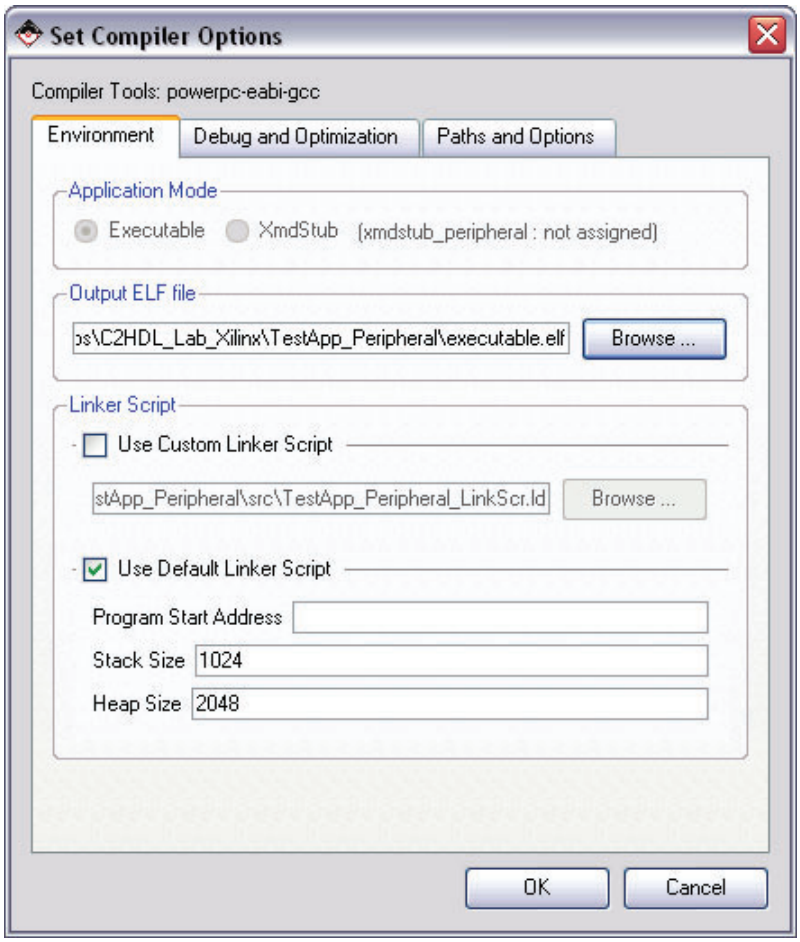


Figure 12: **Set Compiler Options**

39. In the same dialog box, select the **Debug and Optimization** tab.
40. Change the **Optimization Level** drop-down menu to **High (-03)**, and click **OK**.
41. At this time, choose between building a complete system or continuing with a pre-built system.
    a. To build a complete system, go to "Building and Running the System". The process of building a complete system takes approximately 10 minutes on a 2.1 GHz Pentium-Mobile laptop.
    b. To continue with a pre-built system, go to the next step.

42. 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\C2HDL_Lab_Xilinx_Part3\
       ```

    c. Select the file system `.xmp` and click **Open**.
    d. If the **Block Diagram** view appears, click on the **System Assembly View1 Tab** and continue with the next step.
    e. Continue with "Building and Running the System".

Note: XPS uses the GNU Make tool and rebuilds a design only as changes necessitate. When using a pre-built design, XPS bypasses the hardware and software build steps and only a bitstream download occurs.

Building and Running the System

This section explains how to build and run the design.

43. Build the new hardware and software systems, update the bitstream, and download the hardware and software designs to the board by selecting from the menu bar the following:

    Device Configuration → Download Bitstream

44. On the system CRT or LCD display, observe the software-only solution, which is displayed in red.
45. Enable the hardware coprocessor as follows:
    a. If the **Applications** tab is 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 to the left of **Sources**.
    c. To open the main program file, double-click on the following file:

       ```
       ..\code\main.c
       ```

    d. In the right-hand window, scroll down to near line 299, and enable the coprocessor by un-commenting the following line:

       ```
       co_execute(my_arch);
       ```

    e. Save the file by selecting the following from the menu bar:

       File → Save

    f. Re-build the software, update the bitstream, and download the hardware and software designs to the board by selecting from the menu bar the following:

       Device Configuration → Download Bitstream

Note: XPS uses the GNU Make tool and rebuilds modules only as changes necessitate. The process of re-compiling, linking, and downloading takes a few seconds.

46. On the CRT or LCD display, observe both the software-only solution and the hardware-accelerated solution.
    - The software-only solution displays in red.
    - The hardware-accelerated solution with the APU attached coprocessor displays in green.

    Question 5: How much faster is the hardware accelerator solution as compared to the software solution?

    Question 6: The software solution is running on a 200 Hz PowerPC processor. With the acceleration noted in the previous step, what is the required equivalent processor speed to achieve the same accelerated performance in software?

    At this time, the PowerPC system with a Hardware Code Accelerator build is complete.

Summary

This guide explains how to complete a PowerPC processor system design with APU attached hardware code accelerator by performing the following steps:

1. Implement a PowerPC processor system design using Xilinx Platform Studio (XPS) Base System Builder that targets a Virtex-4 FX Evaluation board.
2. Test the design with the automatically generated test code.
3. Run a software Mandelbrot program by replacing the test code with software to generate and display Mandelbrot images.
4. Add a coprocessor accelerator to the PowerPC processor APU interface, enabling the coprocessor calls.
5. Run both the software-only Mandelbrot program and the accelerated Mandelbrot program.
6. Measure and observe the 24 times acceleration of the software algorithm.

Critical software code segments can be replaced with APU attached hardware accelerators. Accelerators can be created by hand or by C-to-HDL tools. As demonstrated in this guide, it is possible to accelerate code by up to 24 times. A 200 MHz processor with code acceleration of 24 times provides performance equivalent to a 4.8 GHz processor.

Answers

Answer 1: Test the I/O interfaces driving the LEDs and LCD on the board.

Answer 2: The RS232_Uart test does not run because it is selected as the STDOUT device.

Answer 3: LED_DELAY

Answer 4: 1,000,000 times

Answer 5: 24 times faster

Answer 6: Required processor speed: 200 MHz x 24 (that is, 4.8 GHz).

Appendix

Tutorial Configuration

Hardware Requirements

- Xilinx ML403 Virtex™-4FX Evaluation Platform
- Xilinx JTAG Platform Cable USB or Parallel IV Cable
- LCD or CRT with VGA connector

Software Requirements

- ISE 8.2 Service Pack 2
- EDK 8.2 Service Pack 1

System Configuration

- ML403 Evaluation Platform, configured for factory default settings as described in the evaluation platform documentation.
- ISE is installed.
- EDK is installed.
- CRT or LCD is connected to the VGA connector on the evaluation board.
- JTAG Platform Cable USB or Parallel IV cable is installed and connected to the evaluation board
- Power cable is connected to the evaluation board.
- Evaluation board is turned ON.