Implementing a Virtex-4 FX PowerPC System with a C-to-HDL Hardware Coprocessor Accelerator

Design Guide

UG096 (v1.0) December 16, 2005

XILINX®

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
12/16/05	1.0	Initial Xilinx release.

Table of Contents

About This Guide

The reference design described in this document 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, this reference design illustrates how to add custom or third-party IP to the PowerPC APU interface.

Guide Contents

This manual contains the following chapters:

- "Implementing a Virtex-4 FX PowerPC System with a C-to-HDL Hardware Coprocessor Accelerator," describes how to build and run the Virtex™-4 FX reference design.
- "Lab Configuration," summarizes the hardware, software, and configuration requirements for the reference design.

References

These documents provide supplemental material useful to this user 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.

Online Document

The following conventions are used in this document:

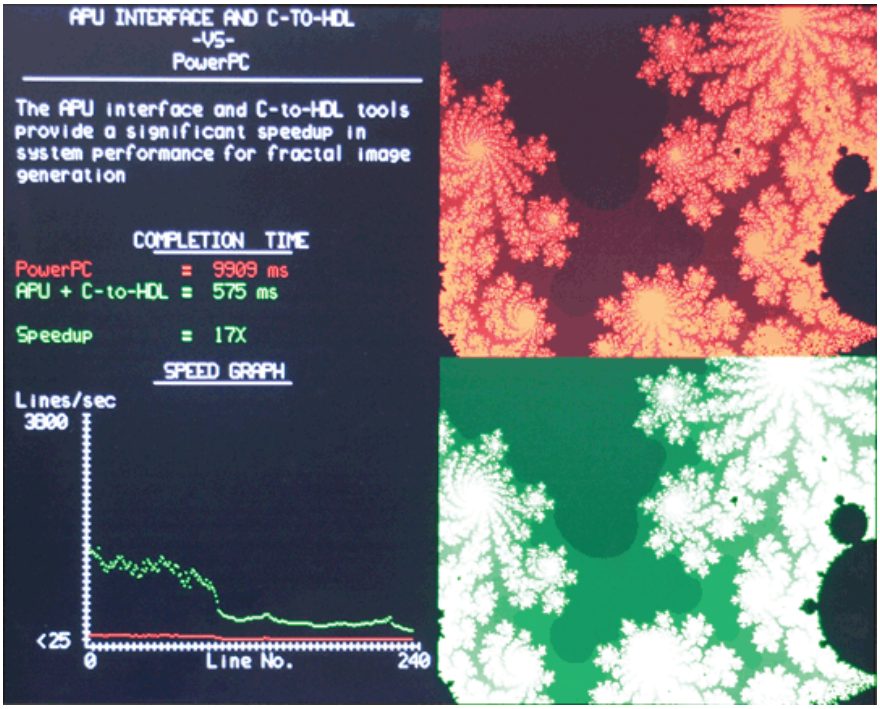
Convention	Meaning or Use	Example
Blue text	Cross-reference link to a location in the current document	See the section "Additional Resources" for details. Refer to "Title Formats" in Chapter 1 for details.
Red text	Cross-reference link to a location in another document	See Figure 2-5 in the *Virtex-II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

Implementing a Virtex-4 FX PowerPC System with a C-to-HDL Hardware Coprocessor Accelerator

Design Challenge

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.

Using the Xilinx Platform Studio™ toolset, or XPS, a Mandelbrot image generator hardware and software system can be built. This system demonstrates image generation first in software on a 300 MHz PowerPC™ system, and then in software with a hardware code accelerator. The hardware code accelerator is tightly coupled to the PowerPC instruction pipeline via the Auxiliary Processor Unit (APU) interface. There is an opportunity to see over 15 times code acceleration via the coprocessor. This performance is equivalent to that of 15 times 300 MHz, or 4.5 GHz!



UG096_01_120605

This workshop walks through the steps of creating a PowerPC design and attaching the hardware code accelerator via the APU interface.

The steps to create this design are:

1. The design is created using XPS Base System Builder tool targeting a Virtex™-4 FX Evaluation board.
2. A previously created code accelerator and the application code are added.
3. The design is downloaded and run.

Building and Running the Design

This section provides step-by-step instructions to build and run a PowerPC design. It contains these sections:

- "Installing the Design Files and Configuring the Environment"
- "Building a Basic XPS Hardware and Software System"
- "Building the System with the Test Application using XPS"

Installing the Design Files and Configuring the Environment

Installing Reference Design Files into Target Directories

1. Set up the design environment and configure the evaluation platform as described in Appendix A, "Lab Configuration."
2. Download the reference design file associated with XAPP901. Copy `PPC_V4_C2HDL_Lab.zip` to your windows desktop.
3. Double-click on `PPC_V4_C2HDL_Lab.zip` and unzip the contents to your desktop.
   - The file `PPC_V4_C2HDL_Design.zip` is copied to your desktop.
   - The file `V4FX12_xbd.zip` is copied to your desktop.
4. Double-click on the file `PPC_V4_C2HDL_Design.zip` and unzip the contents of the file to `C:\`.

   The project files are placed in the following directories:

   ```
   C:\Xilinx_Design\V4FX_Labs\C2HDL_BitFiles\
   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\
   C:\Xilinx_Design\V4FX_Labs\C2HDL_Lab _Avnet\
   C:\Xilinx_Design\V4FX_Labs\C2HDL_Lab_Avnet_Part2\
   C:\Xilinx_Design\V4FX_Labs\C2HDL_Lab_Avnet_Part3\
   C:\Xilinx_Design\V4FX_Labs\C2HDL_CoDeveloper_project\
   ```

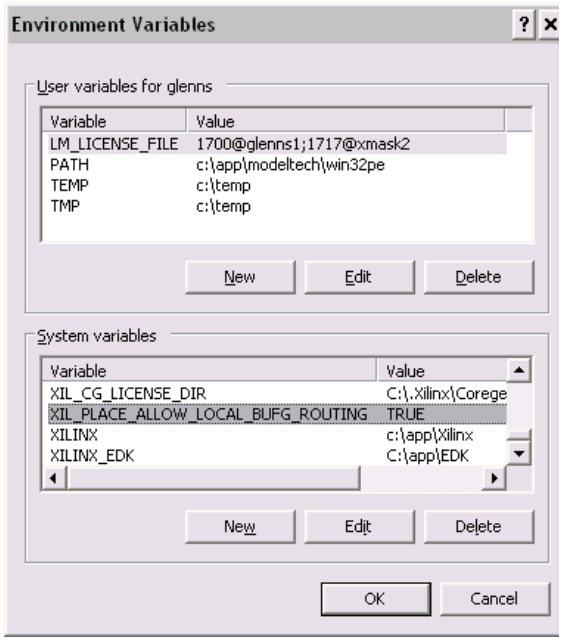
5. Double-click on the file `V4FX12_xbd.zip` and unzip the contents of the file to your EDK program directory (most likely, this directory is `C:\EDK`). This step installs the Virtex-4 FX Evaluation Board description files for the Base System Builder tool.

Configuring the Environment Variable for the Avnet Virtex-4 Development Kit

When targeting the Avnet Virtex-4 Development Kit, the environment variable must be configured to enable buffered clock routing on local clock lines:

1. Right-click on the My Computer icon and select **Properties**.

2. In the Systems Properties window, select the Advance tab.
3. Click on the “Environment Variables” button.
4. Under “System variables”, click on the New button.
5. Enter the variable name: **`XIL_PLACE_ALLOW_LOCAL_BUFG_ROUTING`**.
6. Enter the variable value: **`TRUE`**.
7. Click **OK** on each of the three open windows.



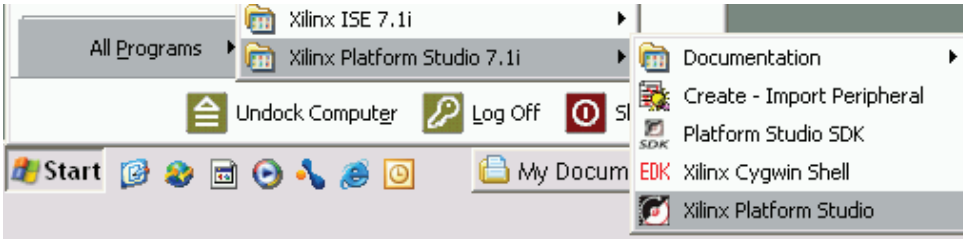
UG096_02_120705

Building a Basic XPS Hardware and Software System

Follow these steps to build a basic XPS hardware and software system.

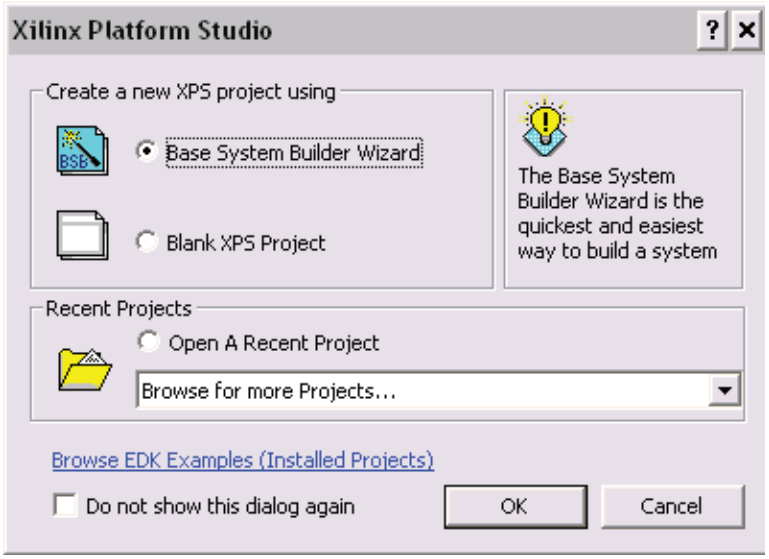
1. Start XPS.

   `Start →All Programs →Xilinx Platform Studio 7.1i →Xilinx Platform Studio`



UG096_03_120705

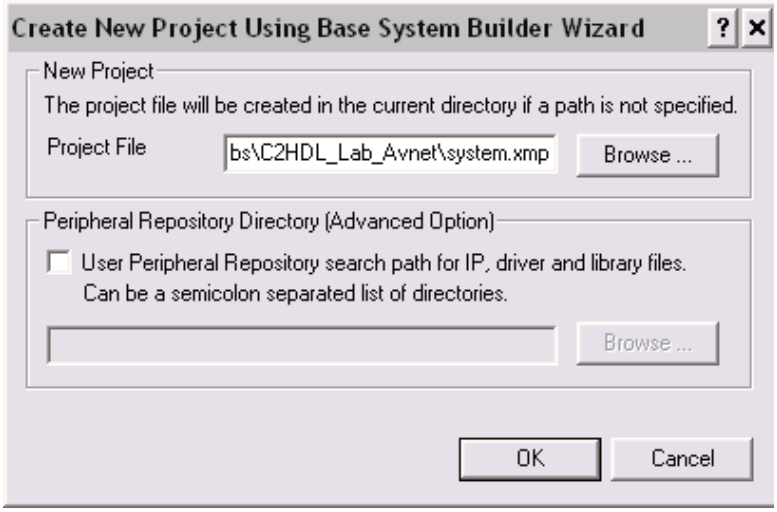
2. Use the Base System Builder tool to create a new basic PowerPC design. Click **OK** to start the Base System Builder Wizard.



UG096_04_120705

3. Click **Browse** and browse to one of the following locations:
   - Xilinx board: `C:\Xilinx_Design\V4FX_Labs\C2HDL_Lab _Xilinx\`
   - Avnet board: `C:\Xilinx_Design\V4FX_Labs\C2HDL_Lab _Avnet\`

   Click **Open** and then click **OK**.



4. **Welcome Page**: Select "I would like to create a new design" and click **Next**.
5. **Select Board Page**: This example uses an existing evaluation platform, either the ML403 or the FX12. Select the target evaluation platform with the following settings and then click **Next**:

Board Vendor	Xilinx	Avnet
Board Name	Virtex-4 ML403 Evaluation Platform with TFT	Virtex-4 FX12 Evaluation Board
Board Revision	1	1.0

6. **Select Processor Page:** Select **PowerPC** and click **Next**.
7. **Configure Processor Page**:
   - Do *not* change the reference clock frequency.
   - Change the processor clock frequency to **`300 MHz`**.
   - Confirm that the bus clock frequency is set to **`100 MHz`**.
   - Select **Cache ENABLED.**
   - Leave the remaining options at their defaults and click **Next.**
8. **Configure I/O Interfaces Page**: The next few dialog windows display the peripherals that can be used with the chosen evaluation platform. Click **Next** as appropriate to advance to the next dialog window.
   a. Deselect the following devices for the applicable evaluation platform:

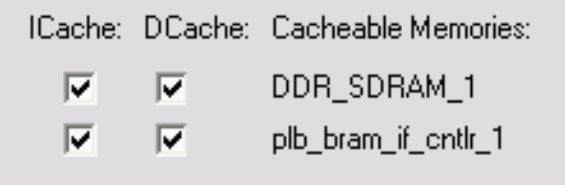
Xilinx	Avnet
• LEDs_Positions • Push_Buttons_Position • IIC_EEPROM • SysACE_CompactFlash • Ethernet_MAC • SRAM_256Kx32 • FLASH_2Mx32	• DIP_Switches_8Bit • FLASH_2Mx16 • Ethernet_MAC

b. Leave the following devices selected for the applicable evaluation platform:

Xilinx	Avnet
• RS232_Uart • LEDs_4Bit • LCD_7Bit_GPIO (Xilinx) • DDR_SDRAM_64Mx32 • plb_tft_cntlr_ref_0 (Video Interface)	• RS232 • LEDs_8Bit • DDR_SDRAM_1 • plb_tft_cntlr_ref_0 (Video Interface)

9. **Add Internal Peripherals Page**: The code and data are stored using memory connected to the IBM CoreConnect™ Peripheral Local Bus (PLB).
   a. Change the memory size of the PLB BRAM IF CNTLR from 16KB to **64KB.**
   b. Click **Next**.
10. **Cache Setup Page**: Code and data can be cached or not for data or instructions and for each type of memory.

    To enable cache for all memories, check all four boxes and then click **Next**.

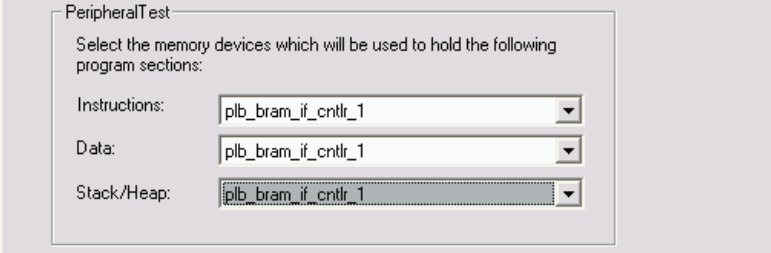


UG096_06_120705

11. **Software Setup Page**: This dialog shows what devices are used for Standard Input and Standard Output and enables Sample Application Selection. For this design,
    a. Keep the default settings.
    b. Have the Base System Builder tool use the RS232 interface for Standard Input and Output.
    c. Create a sample application and linker script.
    d. Click **Next.**
12. **Configure Memory Test Application Page**: This page selects where memory test code, data, and stack/heap reside.

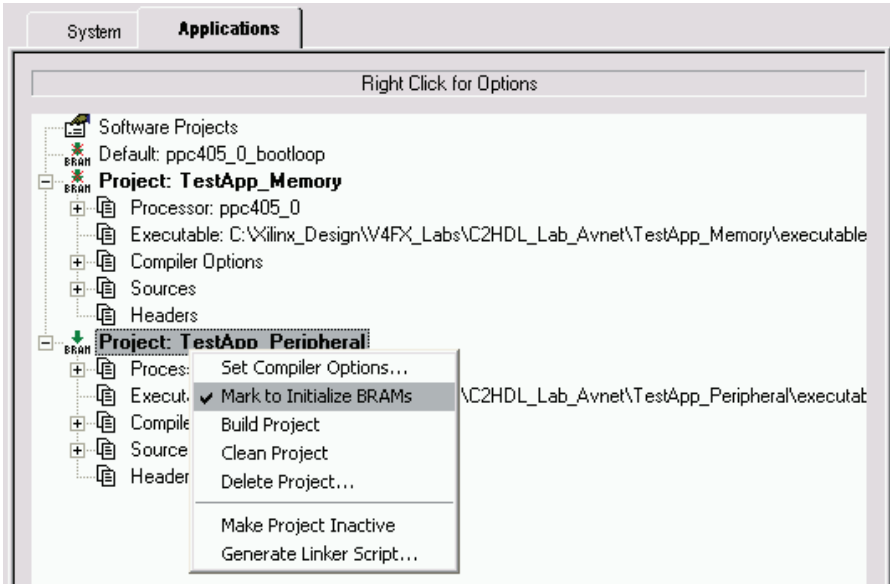
    Leave all settings at Plb_bram_ifcntlr_1 and click **Next**.
13. **Configure Peripheral Test Application Page**: This page selects where peripheral code, data, and stack/heap reside.

    Change the location to **plb_bram_if_cntlr_1** using the drop-down lists for instructions, data, and stack/heap and click **Next**.



UG096_07_120705

14. **System Created Page**: This page shows a summary of the system created. Peripherals and memory were automatically assigned addresses. Click **Generate**.
15. **Congratulations Page**: This page summarizes all of the automatically created files. Click **Finish** to complete the Base System Builder Wizard.
16. If "The Next Step" dialog box appears, click **OK** to start using XPS.
17. Target the peripheral test application for Block RAM execution. These steps deselect the memory test and select the peripheral test:
    a. Click on the Applications tab in the main XPS window.
    b. Right-click on the green down arrow to the left of the Project:TestApp_Memory label and deselect **Mark to Initialize BRAMs**.
    c. Right-click on the red X to the left of the Project:TestApp_Peripheral label and select **Mark to Initialize BRAMs**.



UG096_08_120705

The hardware design for the system is now complete!

To build the complete system, go to "Building the System with the Test Application using XPS." This process takes about 16 minutes on a 1.6 GHz Pentium mobile laptop.

To continue with a prebuilt system, perform these steps:

1. Select **File →Close Project**.
2. Select **File →Open Project**.
3. Browse to the applicable directory:
   Xilinx board: `C:\Xilinx_Design\V4FX_Labs\C2HDL_Lab_Xilinx_Part2\`
   Avnet board: `C:\Xilinx_Design\V4FX_Labs\C2HDL_Lab_Avnet_Part2\`
4. Click on `system.xmp` and click **Open**.
5. Go to step 3 in "Building the System with the Test Application using XPS," to download the design to the FPGA.

Building the System with the Test Application using XPS

Follow these steps when using XPS to build a system with the test application:

1. To build the libraries and board support packages, select:

   Tools →Generate Libraries and BSPs

   This step takes about 30 seconds.

2. Build the complete hardware and software system by selecting:

   Tools →Generate Bitstream

   This step takes about 16 minutes on a 1.6 GHz Pentium mobile laptop.

While the project is building, look at some of the test code and answer a few questions:

3. Open the test program `TestAPP_Memory.c` as follows:
   a. Click on the Applications tab in the main XPS window.
   b. Underneath Project:TestApp_Memory, expand "Sources" by clicking on the "+" to the left of "Sources".
   c. Double-click on the test program `TestAPP_Memory.c`.
4. The main program starts near line 43.
   - What does this program do?

     ANSWER 1:

   - What is the program unable to do?

     ANSWER 2:
5. Open the test program `TestAPP_Peripheral.c` as follows:
   a. Underneath Project:TestApp_Peripheral, expand "Sources" by clicking on the "+" to the left of "Sources".
   b. Double-click on the test program `TestAPP_Peripheral.c`.
6. The main program starts near line 77.
   - What does this program do?

     ANSWER 3:
7. Determine the I/O Port address of the XPAR_LEDS_8BIT or XPAR_LEDS_4BIT.
   - Looking near line 90, what is the name of the constant defining the LED port base address?

     ANSWER 4:

   - The above constant was generated by Base System Builder and is defined in an include file called `xparameters.h`. Where is this file located (look near line 34)?

     ANSWER 5:
8. Open the `xparameters.h` file by selecting **File →Open** and browsing to the file following the path identified in the previous step.
   - Looking at the code in the `xparameters.h` file, what is the base address of XPAR_LEDS_8BIT or XPAR_LEDS_4BIT?

     ANSWER 6:

If the design is still building, check out some of the other automatically defined parameters.

9. Once the bitstream is built, download the design and program the Virtex-4 FX Evaluation Board by selecting:

   Tools →Download

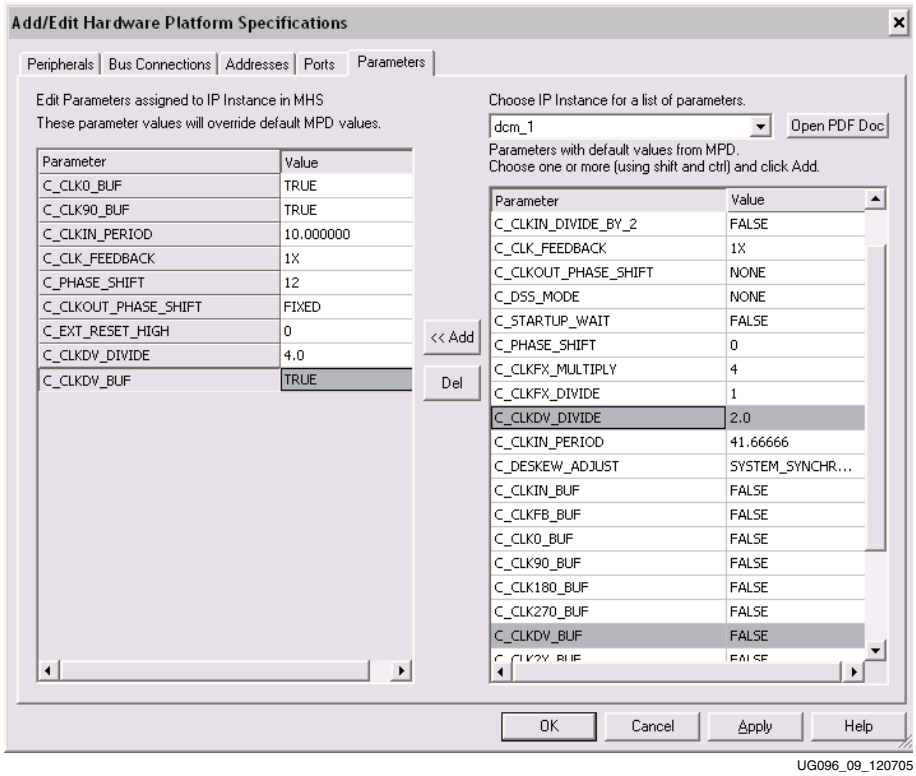
   In less than a minute, a set of LEDs should light up in sequence. Push the CPU Reset button SW11 (Xilinx)/SW1 (Avnet) located at the left corner to restart the program.

This completes building the hardware and software systems!

Configuring the Clocks for the Display Controller and Coprocessor

Follow these steps to create the 25 MHz TFT clock by dividing the 100 MHz input clock by 4:

1. Select **Project →Add/Edit Cores**…
2. Click on the Parameters tab.
3. On the right using the "Choose IP Instance" drop-down list, select **dcm_1**.
4. At the right side of the window, select **C_CLKDV_DIVIDE** and click **<<Add**.
5. Double-click on the value of the C_CLKDV_DIVIDE parameter in the left window and change the value to **`4.0`** (the ".0" is needed).
6. At the right side of the window, select **C_CLKDV_BUF** and click **<<Add**.
7. To enable a clock distribution buffer, double-click on the value of the C_CLKDV_BUF parameter in the left window and change the value to **`TRUE`**.



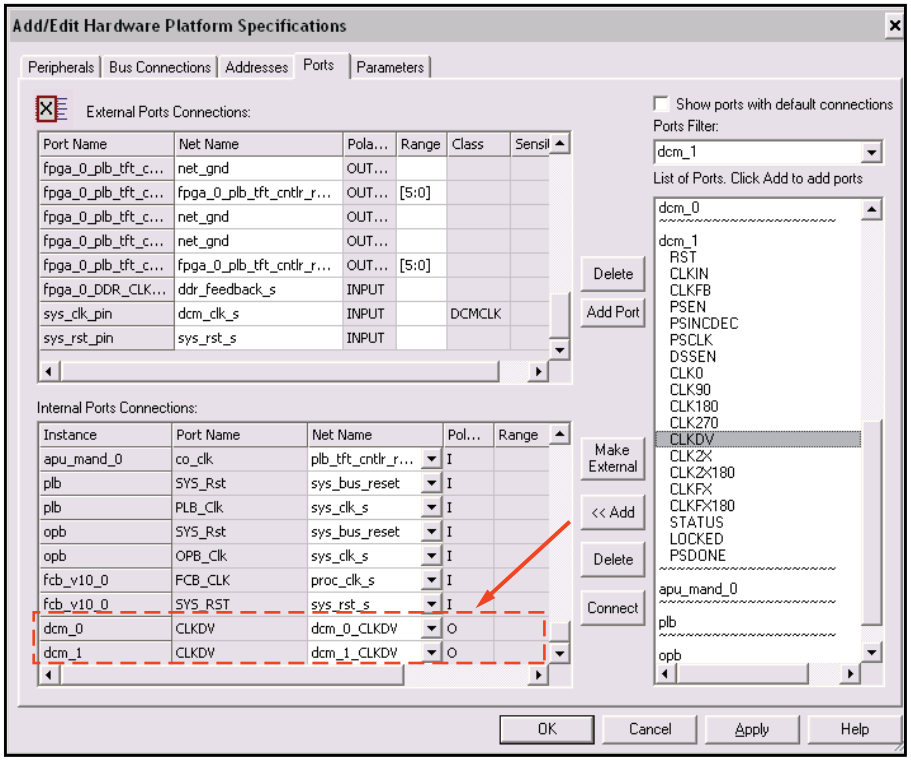
UG096_09_120705

Follow these steps to create a 50 MHz coprocessor clock by dividing the 100 MHz input clock by 2:

1. On the right using the "Choose IP Instance" drop-down list, select **dcm_0**.
2. At the right side of the window, select **C_CLKDV_DIVIDE** and click **<<Add**.
3. Leave unchanged the default value of 2.0 for the C_CLKDV_DIVIDE parameter.
4. At the right side of the window, select **C_CLKDV_BUF** and click **<<Add**.
5. To add a clock distribution buffer, double-click on the value of the C_CLKDV_BUF parameter in the left window and change the value to **TRUE**.

Follow these steps to add the clock ports to the design:

1. Click on the Ports tab.
2. On the right using the "Ports Filter" drop-down list, select **dcm_0**.
3. Scroll down and find the **dcm_0** peripheral instance.
4. Select the **CLKDV** port and click **<<Add**.
5. On the right using the "Ports Filter" drop-down list, select **dcm_1**.
6. Scroll down and find the **dcm_1** peripheral instance.
7. Select the **CLKDV** port and click **<<Add**.

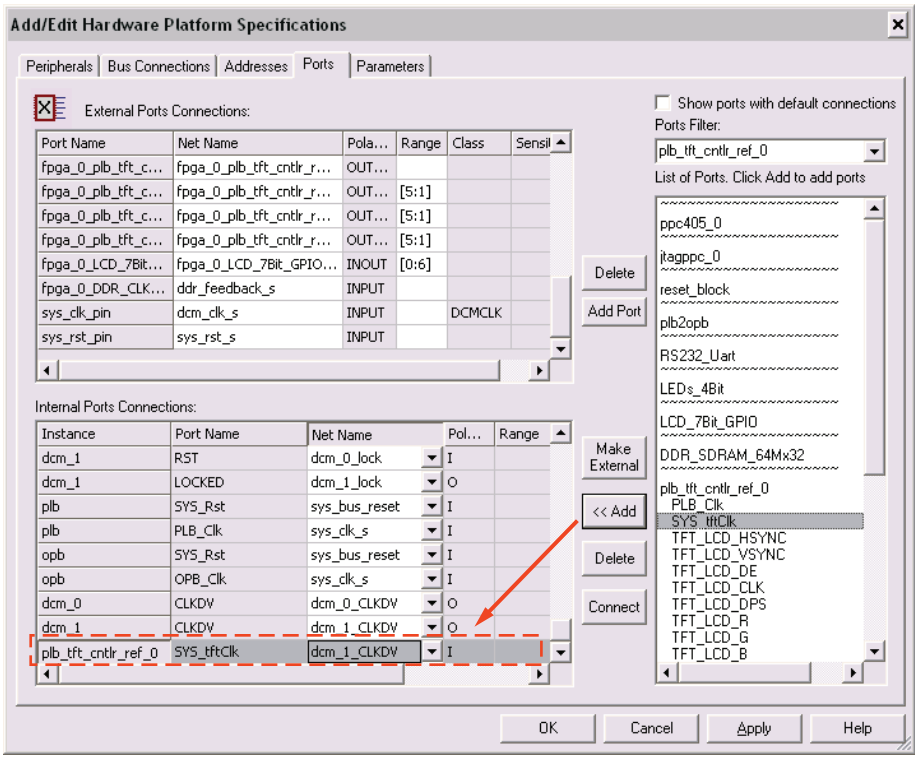


UG096_10_120705

Connecting the TFT Display Controller Clock

Follow these steps to connect the TFT display controller clock:

1. If not already editing cores, select **Project →Add/Edit Cores**…
2. Click on the Ports tab.
3. On the right using the "Ports Filter" drop-down list, select **plb_tft_cntlr_ref_0**.
4. Scroll down and find the plb_tft_cntlr_ref_0 peripheral instance.
5. Select the **SYS_tftCLK** port and click **<<Add**.
6. In the lower left side of the window, for this peripheral, double-click on the drop-down list next to "plb_tft_cntlr_ref_0 SYS_tftCLK" and select **dcm_1_CLKDV**. (Hint: the signal is near the bottom of the list.)
7. Click **OK**.



UG096_11_120805

Configure the User Constraint File (UCF) for System Timing

Follow these steps to configure the UCF:

1. Click on the System tab in the main XPS window.
2. Under "Project Files", double-click on the `data\system.ucf` file to open the UCF file.
3. Copy and paste these lines of code into the beginning of the UCF file (line 5) to correctly set timing constraints for the design:

```
############### Timing Constraints Added Manually###############
Net C405RSTCHIPRESETREQ TIG;
Net C405RSTCORERESETREQ TIG;
Net C405RSTSYSRESETREQ  TIG;
##############################################################
```

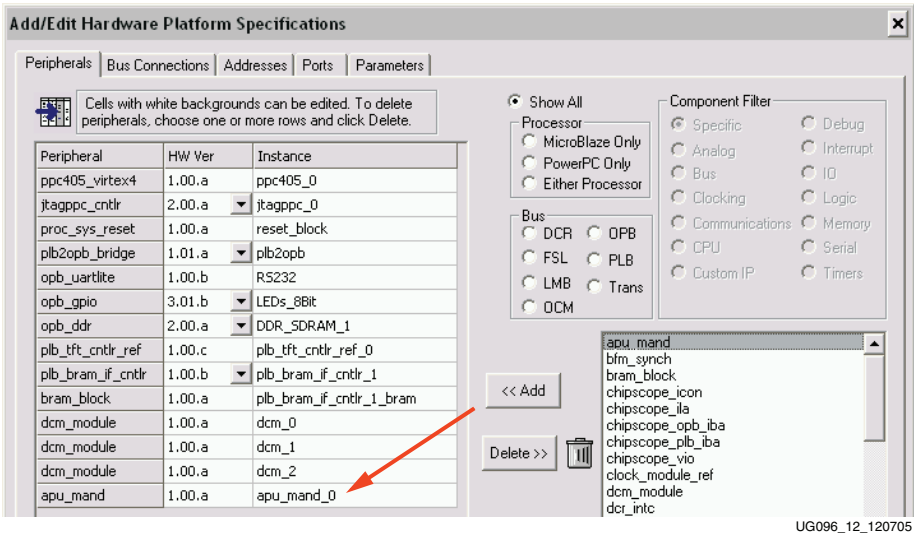
4. Select **File** →**Save** to update the file.

Adding the Hardware Code Accelerator

Follow these steps to add the Hardware Code Accelerator:

1. In XPS, select **Project** →**Add/Edit Cores**…

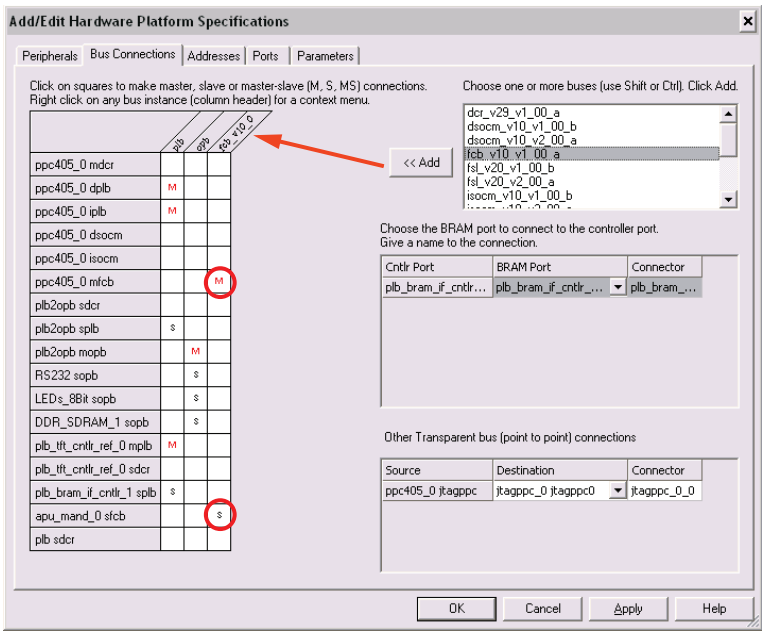
   The Peripherals tab shows all the hardware components that have been added to the system. On the right, are all available hardware components that come with the XPS install or are found in a peripheral repository. To add the coprocessor to the design, select the **apu_mand** peripheral and click on the **<<Add** button.



UG096_12_120705

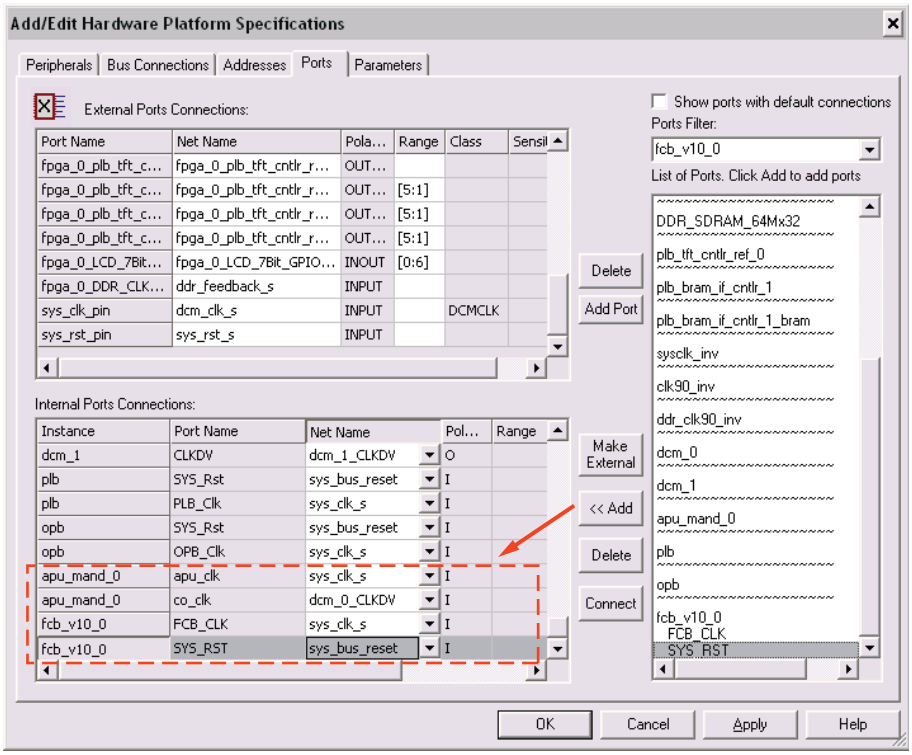
2. Click on the "Bus Connections" tab to see how the peripherals are connected in the system. First the processor must be connected to the APU interface bus, and then the coprocessor must be connected to the bus:
   a. To add the APU bus, select **fcb_v10_v1_00_a** and click **Add**.
   b. To set the PowerPC processor as the master on the APU interface, click on the white square next to "ppc405_0 mfcb" in the fcb_v... column. An "M" should appear indicating the PowerPC processor is the master device.

c. To add the hardware coprocessor to the APU interface, click on the white square next to “apu_mand_0 sfcb” in the fcb_v... column. An “S” should appear indicating the FPU is a slave device.



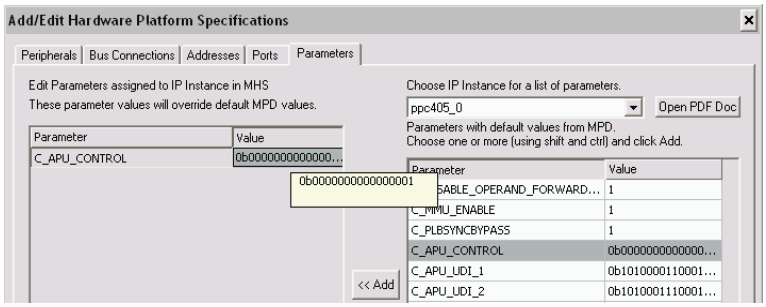
UG096_13_120705

3. To connect the hardware coprocessor clocks:
   a. Click on the Ports tab.
   b. On the right using the “Ports Filter” drop-down list, select **apu_mand_0**.
   c. Scroll down to the apu_mand_0 peripheral instance.
   d. To add the APU interface clock, select the **apu_clk** port and click **<<Add**.
   e. To add the coprocessor clock, select the **co_clk** port and click **<<Add**.
   f. In the lower left side of the window, for this peripheral, double-click on the drop-down list next to apu_clk and select **sys_clk_s** to connect the interface clock.
   g. For this peripheral, double-click on the drop-down list next to co_clk and select **dcm_0_CLKDV** to connect the coprocessor clock. (Hint: the signal is near the bottom of the list.)
4. To connect the APU interface port clock and reset:
   a. On the right using the “Ports Filter” drop-down list, select **fcb_v10_0**.
   b. Scroll down and find the fcb_v10_0 peripheral instance.
   c. Select the **FCB_CLK** port and click **<<Add**.
   d. Select the **SYS_Rst** port and click **<<Add**.
   e. For this peripheral, in the lower left side of the window, double-click on the drop-down list next to FCB_CLK and select **sys_clk_s**.
   f. For this peripheral, double-click on the drop-down list next to SYS_Rst and select **sys_bus_reset**.



UG096_14_120705

5. To enable the APU interface:
   a. Click on the Parameters tab.
   b. On the right, using the "Choose IP Instance" drop-down list, select **ppc405_0**.
   c. On the right, select **C_APU_CONTROL** and click **<<Add**.
   d. On the left, double-click on the value of the C_APU_CONTROL parameter and change the value to **0b0000000000000001** or **0x1**.



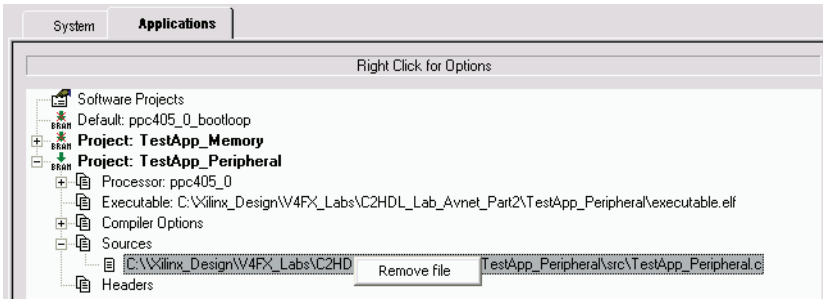
UG096_15_120705

6. Carefully check your work in step 1 through step 5.
7. Complete the changes by clicking **OK**.

Adding the Application Code

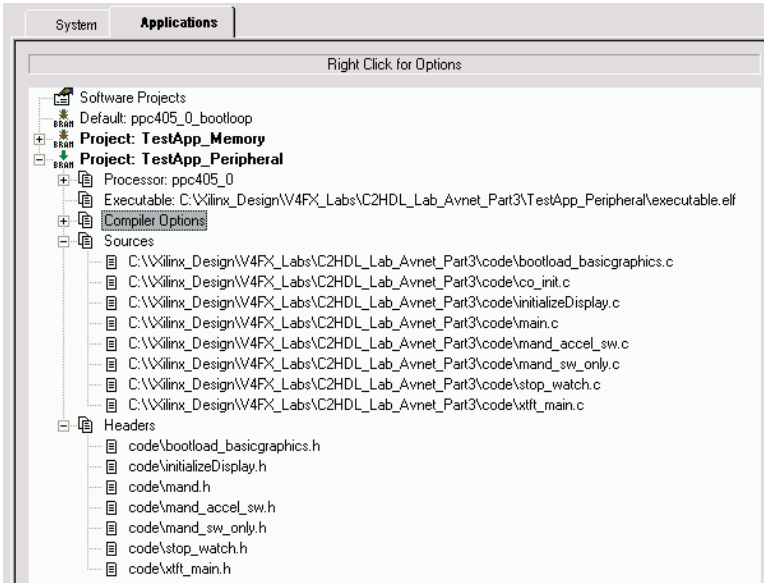
Follow these steps to add the application code:

1. Remove the test application:
   a. Click on the Applications tab in the main XPS window.
   b. Under Project: TestApp_Peripheral, expand "Sources" by clicking on the "+" in front of "Sources".
   c. To remove `TestAPP_Peripheral.c`, right-click and select **Remove File**.



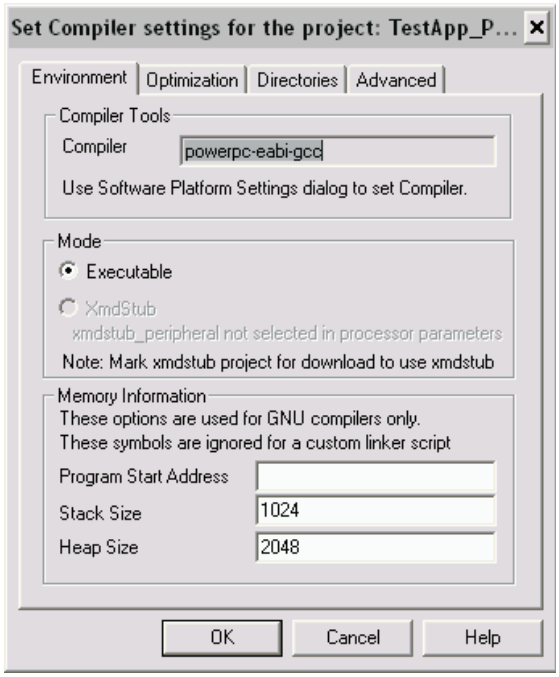
UG096_16_120705

2. Double-click on **Sources** and browse to the `\code` subdirectory. To add all of the files found in that location, select them and then click **Open**.
3. On the left side, under Project: TestApp_Peripheral, double-click on **Compiler Options**.



UG096_17_120705

   a. Set the stack size to **1024**.
   b. Set the heap size to **2048**.



4. Select the Optimization tab and set the Optimization Level to **Level 3**. Click **OK**.

Building and Running the System with a Coprocessor Accelerator

To build the complete system, go to step 2, otherwise continue at step 1. Building the complete system takes about 11 minutes on a 1.6 GHz Pentium mobile laptop.

1. To continue with a prebuilt system, perform the following steps:
   a. Select **File →Close Project**.
   b. Select **File →Open Project**.
   c. Browse to the applicable directory:
   Xilinx Board: `C:\Xilinx_Design\V4FX_Labs\C2HDL_Lab_Xilinx_Part3\`
   Avnet Board: `C:\Xilinx_Design\V4FX_Labs\C2HDL_Lab_Avnet_Part3\`
   d. Click on `system.xmp` and click **Open**.
2. To build the new hardware and software systems, update the bitstream, and download to the board, select **Tools →Download**.
3. Observe your software only working design on the CRT or LCD display. The software-only solution is displayed in red.
4. Enable the hardware coprocessor as follows:
   a. If the Applications tab is not already selected, click on it in the main XPS window.
   b. Under Project: TestApp_Peripheral, expand “Sources” by clicking on the “+” in front of “Sources”.
   c. Double-click on the file …`\code\main.c` to open the main program file.
   d. In the right-hand window, scroll down near line 268 and enable the coprocessor by uncommenting the following line:

```
co_execute(my_arch);
```

e. Save your changes. Rebuild the software, update the bitstream and download to the board by clicking on **Tools →Download**.

   XPS has the "make" capability and only rebuilds modified modules. The recompile, link, and download only take a few seconds.

5. Observe your software-only and hardware-accelerated solution on the LCD display. The software-only solution is displayed in red. The solution with the hardware coprocessor attached to the APU interface is displayed in green.

   a. Approximately how much faster is the hardware accelerator solution as compared to the software solution?

      ANSWER 7:

   b. The software solution is running on a 300 MHz PowerPC system. With the acceleration noted in step a what would the required equivalent processor speed to achieve the same accelerated performance in software?

      ANSWER 8:

Building a PowerPC system with a Hardware Code Accelerator is now complete!

Preparing to Use the Base System Builder Tool

This design uses the Base System Builder tool, 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`.

This lab includes a customized Base System Builder configuration file which includes the video TFT controller.

Answers

This section provides the answers to the questions located throughout this chapter.

- ANSWER 1: Runs a memory test on the external DDR memory.
- ANSWER 2: The memory test is not run at `0xffff0000` (plb_bram_if_cntlr_1), the block RAM memory, due to the code being in the same memory bank to be tested.
- ANSWER 3: Test the I/O interface driving the LEDs on the board.
- ANSWER 4: Xilinx Board: XPAR_LEDS_4BIT_BASEADDR
  Avnet Board: XPAR_LEDS_8BIT_BASEADDR
- ANSWER 5: `ppc405_0/include/xparameters.h`
- ANSWER 6: `0x40000000`
- ANSWER 7: Typically 17 times faster.
- ANSWER 8: Required processor speed is 300 MHz x 17, or 5.1 GHz!

Appendix A

Lab Configuration

Hardware Requirements

The reference design requires the following hardware:

- Xilinx ML403 Virtex-4 FX Evaluation Platform
  or
  Avnet Xilinx Virtex-4 FX Evaluation Kit (ADS-XLX-V4FX-EVL12) with Avnet Audio/Video Module (ADS-AV-DAU)
- Xilinx JTAG Platform Cable USB or Parallel IV Cable
- LCD or CRT with VGA connector

Software Requirements

The reference design requires the following software:

- ISE 7.1 Service Pack 4
- XPS 7.1 Service Pack 2

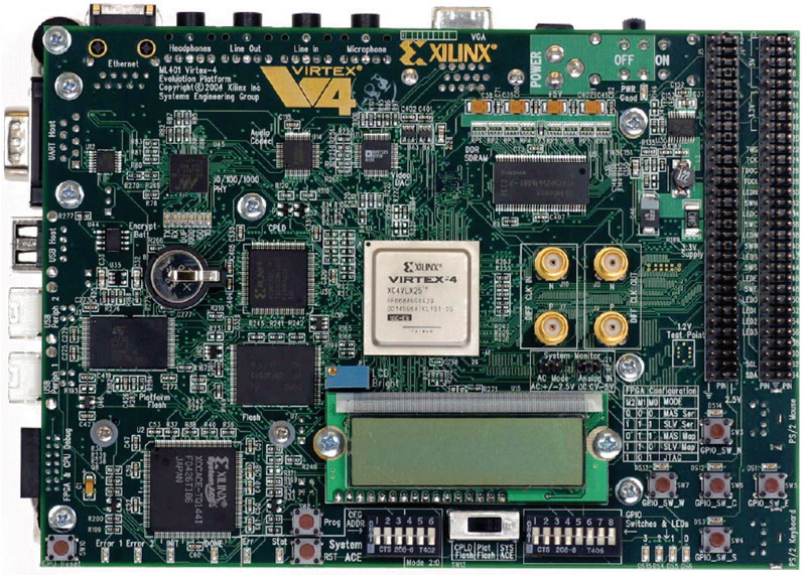
System Configuration

The system configuration process is summarized in six steps:

1. ISE is installed.
2. XPS is installed.
3. The CRT or the LCD is connected to the VGA connector on the evaluation board.
4. The JTAG platform cable USB or Parallel IV cable is installed and connected to the evaluation board.
5. The power cable is connected to the evaluation board.
6. The evaluation board is turned on.

Xilinx ML403 Evaluation Platform Configuration

Figure A-1 shows the ML403 evaluation platform. Set the jumpers and switches on the ML403 as delivered from the factory.

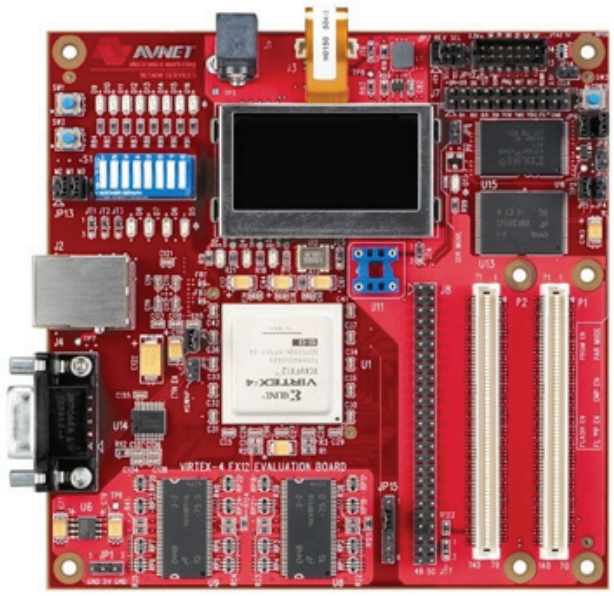


UG096_19_120705

Figure A-1: **ML403 Evaluation Platform**

Avnet Virtex-4 FX Evaluation Board Configuration

Figure A-2 shows the Avnet Virtex-4 FX evaluation board.



UG096_20_120705

Figure A-2: **Avnet Virtex-4 FX Evaluation Board**

Table A-1 shows the jumper settings for the evaluation board. Table A-2 shows the jumper settings for the audio/video module.

Table A-1: **Virtex-4 FX Evaluation Board Jumper Settings**

Jumper	Setting
JP3	ON
JP6	ON
JP7	1-2
JP9	ON
JP11	ON
JP13	1-2 and 5-6 (Boundary-Scan mode)
JP15	2-3
All others OFF	

Table A-2: **Audio/Video Module Jumper Settings**

Jumper	Setting
JP1	**1-2**
JP2	2-3
JP3	2-3
JP7	1-2