XAPP1038 (v1.0) February 8, 2008

Reference System: PLBv46 PCI Using the Avnet Spartan-3 FPGA Evaluation Board

Author: Lester Sanders

Summary

This application note describes how to build a reference system for the Processor Local Bus Peripheral Component Interconnect (PLBv46 PCI) core using the MicroBlaze™ processor-based embedded system in the Avnet Spartan™-3 Evaluation Board.

A set of files containing Xilinx Microprocessor Debugger (XMD) commands is provided for writing to the Configuration Space Header and for verifying that the PLBv46 PCI core is operating correctly. Several software projects illustrate how to configure the PLBv46 PCI cores, set up interrupts, scan configuration registers, and set up and use DMA operations. The procedure for using ChipScope™ Pro Analyzer to analyze PLBv46 PCI functionality is provided.

Included Systems

This application note includes one reference system:

www.xilinx.com/support/documentation/application_notes/xapp1038.zip

The project name used in xapp1038.zip is s3_mb_plbv46_pci.

Required Hardware and Tools

Users must have the following tools, cables, peripherals, and licenses available and installed. EDK provides an evaluation license for PLBv46 PCI.

- Xilinx EDK 9.2.02i
- Xilinx ISE™ 9.2.04i
- Xilinx Download Cable (Board Cable USB or Parallel Cable IV)
- Model Technology ModelSim v6.1e
- ChipScope 9.2.01
- PLBv46 PCI License
- Avnet Spartan-3 PCI Evaluation Board

Introduction

PCI transactions are done between an initiator and a target. This reference design is for the Avnet Spartan-3 Evaluation Board. To be useful, it must be inserted into a PCI slot. In the examples provided in this application note, the Avnet Spartan-3 Evaluation Board is inserted into PCI slot P3 of the Xilinx ML410 Evaluation Platform. This allows both configuration and memory transactions to be done on the PCI bus between an initiator and a target. The examples use the ML410 PLBv46 PCI as the initiator and the Avnet Spartan-3 Evaluation Board PLBv46 PCI as the target. It is relatively easy to modify the examples so that the initiator and target functions are swapped.

Figure 1 is a functional diagram of the Avnet Spartan-3 PCI Board interfacing to the ML410 Evaluation Platform.

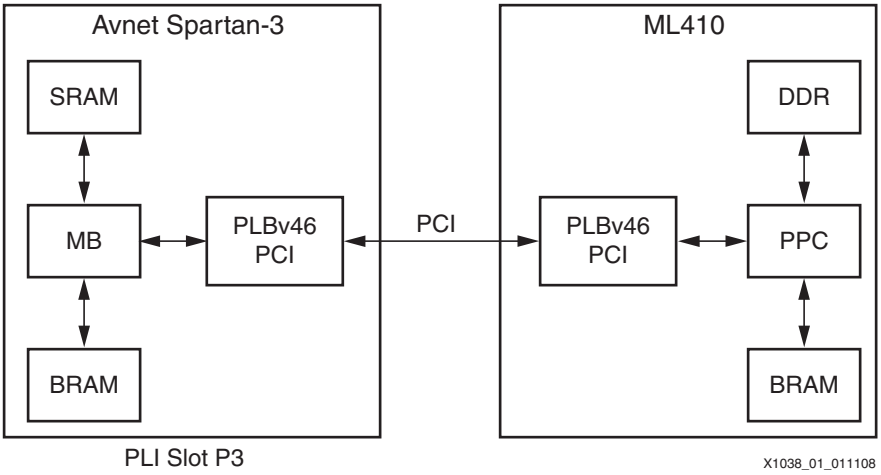


Figure 1: **Interfacing Avnet Spartan-3 PLBv46 PCI with ML410 PLBv46 PCI**

Figure 2 is the Avnet Spartan-3 PCI Evaluation Board.

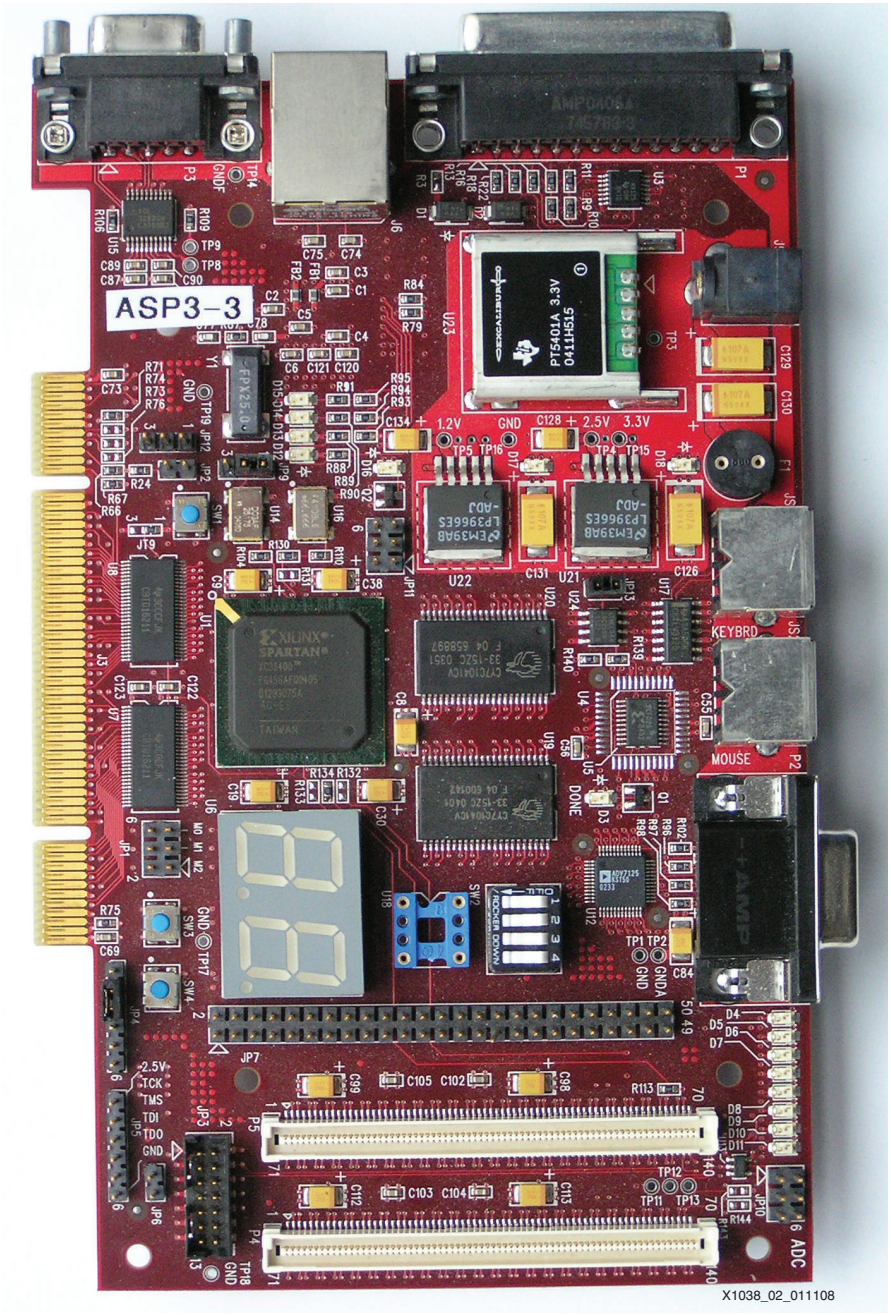


Figure 2: **Avnet Spartan-3 PCI Evaluation Board**

Figure 3 is a block diagram of the reference system.

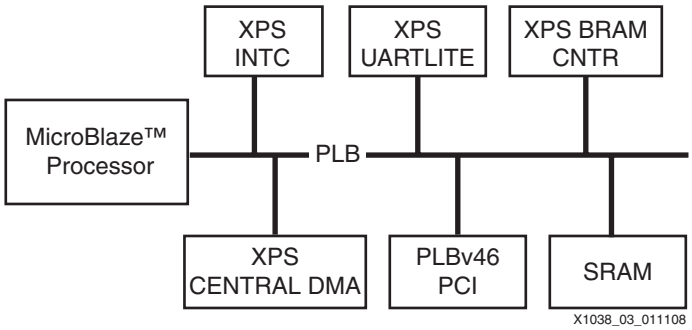


Figure 3: **Avnet Spartan-3 PLBv46 PCI Reference System Block Diagram**

The Avnet Spartan-3 Evaluation Board has a PCI edge connector on one side of the board. Three variations of the Avnet Spartan-3 Evaluation Board differ in the use of the XC3S400, XC3S1000, or XC3S1500, each in a 456 pin package.

The functions, devices, and buses in this PLBv46 PCI reference design are addressed using the Configuration Address Port format shown in Figure 4.

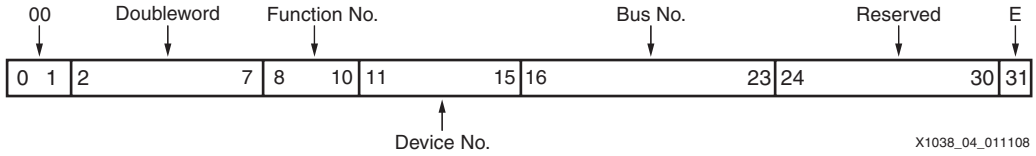


Figure 4: **Configuration Address Port Format**

The Configuration Address Port and Configuration Data Port registers in the host bridge PLBv46 PCI Bridge are used to configure multiple PCI bridges when host bridge configuration is enabled. The bit definitions of the Configuration Address Port in the big endian format used by the PLB are given in Table 1.

Table 1: **Configuration Address Port Register Definitions**

Bit	Definition
0-5	Target word address in configuration space
6-7	Hardwired to 0
8-12	Device
13-15	Function
16-23	Bus Number
24	Enable
25-31	Hardwired to 0

Reference System Specifics

In addition to the MicroBlaze processor and PLBv46 PCI, this system includes SRAM and BRAM memory, UART, MDM, XPS Central DMA, and interrupt controller. The PCI Arbiter core is included in the FPGA.

Avnet Spartan-3 FPGA PCI Evaluation Board

In the reference design, the PLBv46 PCI in the XC3S1500 on the Avnet Spartan-3 Evaluation Board interfaces to the PLBv46 PCI in the Virtex-4 ML410 Evaluation Board. The Avnet Spartan-3 board uses the Xilinx XC3S1500 device in the 456 pin package.

Table 2 provides the address map for the XC3S1500.

Table 2: **Avnet Spartan-3 Address Map**

Peripheral	Instance	Base Address	High Address
LMB_BRAM	DLMB_CNTLR/ILMB_CNTLR	0x00000000	0x00001FFF
XPS UART Lite	RS232_Uart_1	0x84000000	0x8400FFFF
PLBv46 PCI	plbv46_pci_0	0x42600000	0x4260FFFF
XPS MCH EMC	SRAM	0x80300000	0x803FFFFF
MDM	debug_module	0x84400000	0x8440FFFF
XPS INTC	xps_intc_0	0x81800000	0x8180FFFF
XPS CENTRAL DMA	xps_central_dma_0	0x81810000	0x8181FFFF
XPS BRAM	xps_bram_if_cntlr_1	0x8A208000	0x8A20FFFF
XPS GPIO	LEDs_8Bit	0x81400000	0x8140FFFF

The Avnet Spartan-3 1500 Evaluation Board includes a 64-bit PCI edge connector, 1 MB Cypress SRAM memory, RS232C port, LED displays, Atmel Serial EEPROM, XCF04S Board Flash configuration PROM, and a JTAG port. The MicroBlaze microprocessor runs at 66 MHz.

The application note XAPP1001 Reference System: PLBv46 PCI in a ML410 Embedded Development Platform provides a link to the hardware design files used for the ML410.

Configuration of the PLBv46 PCI on the ML410 Board

The PLBv46 PCI bridge uses the 32-bit Xilinx LogiCORE™ IP Version 3 (v3.0) core. For the PLBv46 PCI bridge to perform transactions on the PCI bus, the v3.0 core must be configured using configuration transactions from either the PCI-side or the PLBv46 side. In this reference design, the ML410 PLBv46 PCI is the host bridge, and it configures itself and the Avnet Spartan-3 Evaluation Board PLBv46 PCI. The C_INCLUDE_PCI_CONFIG parameter is set to 1.The IDSEL input of the v3.0 is connected to address ports, and the PLBv46 PCI IDSEL port is unused.

Do the following steps to write to the configuration header.

1. Configure the Command and Status Register. The minimum that must be set is the Bus Master Enable bit in the command register. For memory transactions, set the memory space bit. For I/O transactions, set the I/O space bit.
2. Configure the Latency Timer to a non-zero value.
3. Configure at least one BAR. Configure subsequent BARs as needed for other memory/IO address ranges.

The v3.0 core configures itself only after the Bus Master Enable bit is set and the latency timer is set to avoid time-outs. If the v3.0 core latency timer remains at the default 0 value, configuration writes to remote PCI devices do not complete, and configuration reads of remote PCI devices terminate due to the latency timer expiration. Configuration reads of remote PCI devices with the latency timer set to 0 return 0xFFFFFFFF.

ML410 XC4VFX60 Address Map

The address map of the ML410 XC4VFX60 is shown in Table 3.

Table 3: **ML410 XC4VFX60 System Address Map**

Peripheral	Instance	Base Address	High Address
MPMC3	DDR_SDRAM_32Mx64	0x00000000	0x03FFFFFF
XPS UART16550	RS232_Uart_1	0x83E00000	0x83E0FFFF
XPS INTC	XPS_INTC_0	0x81800000	0x8180FFFF
PLBv46_PCI	PCI32_Bridge	0x85E00000	0x85E0FFFF
XPS Central DMA	xps_central_dma_0	0x80200000	0x8020FFF
XPS BRAM	xps_bram_if_cntlr_0	0xFFFF0000	0xFFFFFFFF
XPS GPIO	LEDs_8Bit	0x81400000	0x8140FFFF
XPS SysAce	SysACE_CompactFlash	0x83600000	0x8360FFFF
XPS IIC	IIC_Bus	0x81600000	0x8160FFFF

Avnet Spartan-3 PLBv46 PCI Generics

The reference design contains the following settings for Spartan 3 PLBv46 PCI generics:

C_FAMILY = spartan3

C_INCLUDE_PCI_CONFIG = 0

C_INCLUDE_BAROFFSET = 0

C_IPIFBAR_NUM = 2

C_PCIBAR_NUM = 2

C_IPIFBAR_0 = 0x20000000

C_IPIFBAR2PCIBAR_0 = 0x80000000

C_IPIFBAR_1 = 0xE8000000

C_IPIFBAR2PCIBAR_1 = 0x90000000

When C_FAMILY is defined as Virtex4 or Spartan3, the PLBv46 PCI uses the v3.0 PCI LogiCORE IP. When C_FAMILY is defined as Virtex5, the PLBv46 PCI uses the v4.0 PCI LogiCORE IP.

Figure 5 shows how to specify the values of Base Address Registers (BARs) in EDK.

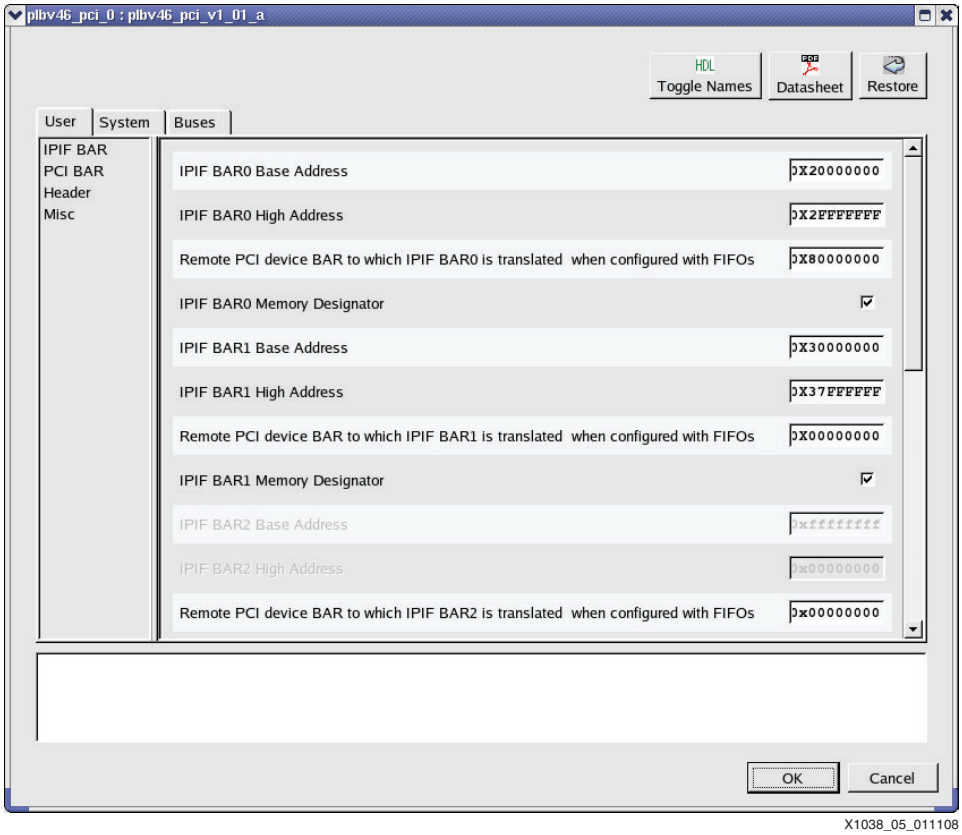


Figure 5: **Specifying the Values of Base Address Register (BAR) Generics in EDK**

Setting C_INCLUDE_PCI_CONFIG = 1 configures the bridge as a host bridge, and since the ML410 does the configuration in this setup, C_INCLUDE_PCI_CONFIG is set to 0. When C_INCLUDE_BAR_OFFSET = 0, the C_IPIFBAR2PCIBAR_* generic(s) are used in address translation instead of IPIFBAR2PCIBAR_* registers. Setting C_IPIFBAR_NUM = 2 specifies that there are two address ranges for PLB to PCI transactions. Setting C_PCIBAR_NUM = 2 specifies that two address ranges are used for PCI to PLBv46 transactions.

Figure 6 provides a functional diagram of the PLBv46 PCI core. The functions in the PLBv46 PCI are the PLBv46 Master, PLBv46 Slave, v3.0 PCI Core, and the IPIF/v3.0 Bridge.

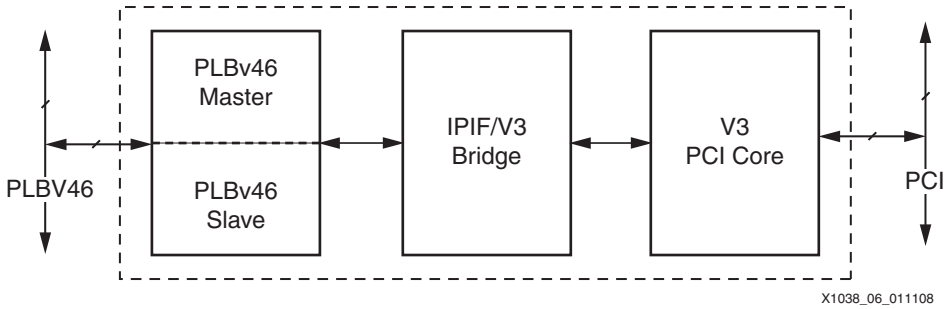


Figure 6: **PLBv46 PCI Functional Diagram**

Configuration of PLBv46 PCI on the Avnet Spartan-3 PCI Evaluation Board

When the Avnet Spartan-3 FPGA Board is inserted into the ML410 PCI slot P3, the PLBv46 PCI Bridge in the XC4VFX60 FPGA interfaces to an PLBv46 PCI Bridge in the XC3S1500 FPGA on the Avnet Spartan-3 Board. To configure the XC3S1500, connect the Xilinx download cable to the Avnet Spartan-3 FPGA Board JTAG port, and use Impact to download the `s3_mb_plbv46_pci/ready_for_download/download.bit` file.

To configure the XC4VFX60, connect the Xilinx download cable to the ML410 JTAG port, and use Impact to download the `ML410 download.bit` file.

After downloading the XC3S1500 FPGA bit file, the PCI functionality in the XC3S1500 PLBv46 PCI is configured using Configuration write PCI transactions from the ML410 PLBv46 PCI.

Executing the Reference System using the Pre-Built Bitstream and the Compiled Software Applications

Use the steps below to execute the system using files inside the s3_mb_plbv46_pci/ready_for_download directory.

1. Change to the `s3_mb_plbv46_pci/ready_for_download` directory.
2. Use iMPACT to download the bitstream.
   `impact -batch xapp1038.cmd`
3. Invoke XMD and connect to the MicroBlaze processor.
   `xmd -opt xapp1038.opt`
4. While the PLBv46 PCI in the Avnet Spartan-3 PCI board can act as the host bridge, this reference design uses the PLBv46 PCI in the ML410 as the host bridge. Connect the JTAG cable to the ML410. Download the bitstream. Download the executable.
   `dow ml410_ppc_plbv46_pci/ready_for_download/hello_pci.elf`

Executing the Reference System from EDK

To execute the system using EDK, follow these steps:

1. Select **File Open Project** system.xmp.

2. The Xplorer flow in EDK must be used to meet timing. Figure 7 shows how to specify the Xplorer flow after selecting Project Options.

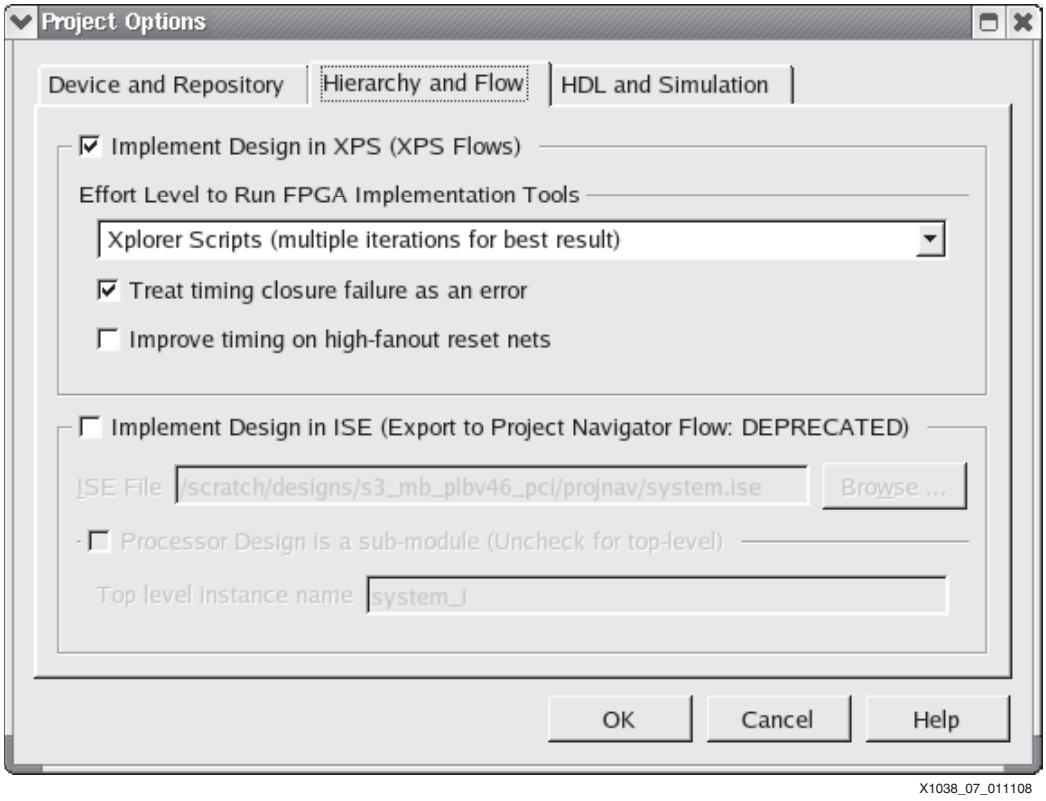


Figure 7: **Specifying Xplorer**

3. Select **Hardware** → **Generate Bitstream** to generate a bitstream
4. Download the bitstream to the board using **Device Configuration** → **Download Bitstream**.
5. Invoke XMD with **Debug** → **Launch XMD**.
6. While the PLBv46 PCI in the Avnet Spartan-3 PCI board can act as the host bridge, this reference design uses the PLBv46 PCI in the ML410 as the host bridge. Connect the JTAG cable to the ML410. Download the bitstream. Download the executable.
   `dow ml410_ppc_plbv46_pci/ready_for_download/hello_pci.elf`

Verifying the Reference Design with the Xilinx Microprocessor Debugger

After downloading the bitstream file and writing to the configuration header, verify that the Avnet Spartan-3 reference design is set up correctly.

1. Configure the v3.0 Command Register, Latency Timer, and BAR(s).
2. Read the configuration header.
3. Configure the Command Register, Latency Timer, and BAR(s) of the other devices in the system.
4. Read the configuration headers of the other devices in the system.
5. Perform a memory read of one of the IPIF BARs.
6. Perform a memory write of one of the IPIF BARs.

Verification is done using either Xilinx Microprocessor Debugger (XMD) or the software projects discussed later. TCL scripts using XMD commands are provided in the `s3_mb_plbv46_pci/xmd_commands` directory. The wr_410_s3.tcl script configures and verifies the ML410 and Spartan-3 PLBv46 PCI cores. To run this script, enter **`xmd -tcl xmd_commands/410_s3pci.tcl`** at the command prompt.

The XMD commands in the `410_s3pci.tcl` file, partially listed in Figure 8, write to the Configuration Address Port and to the Configuration Data Port to program the Configuration Space Header. The Command/Status Register, Latency Timer, and Base Address Registers are written and read.

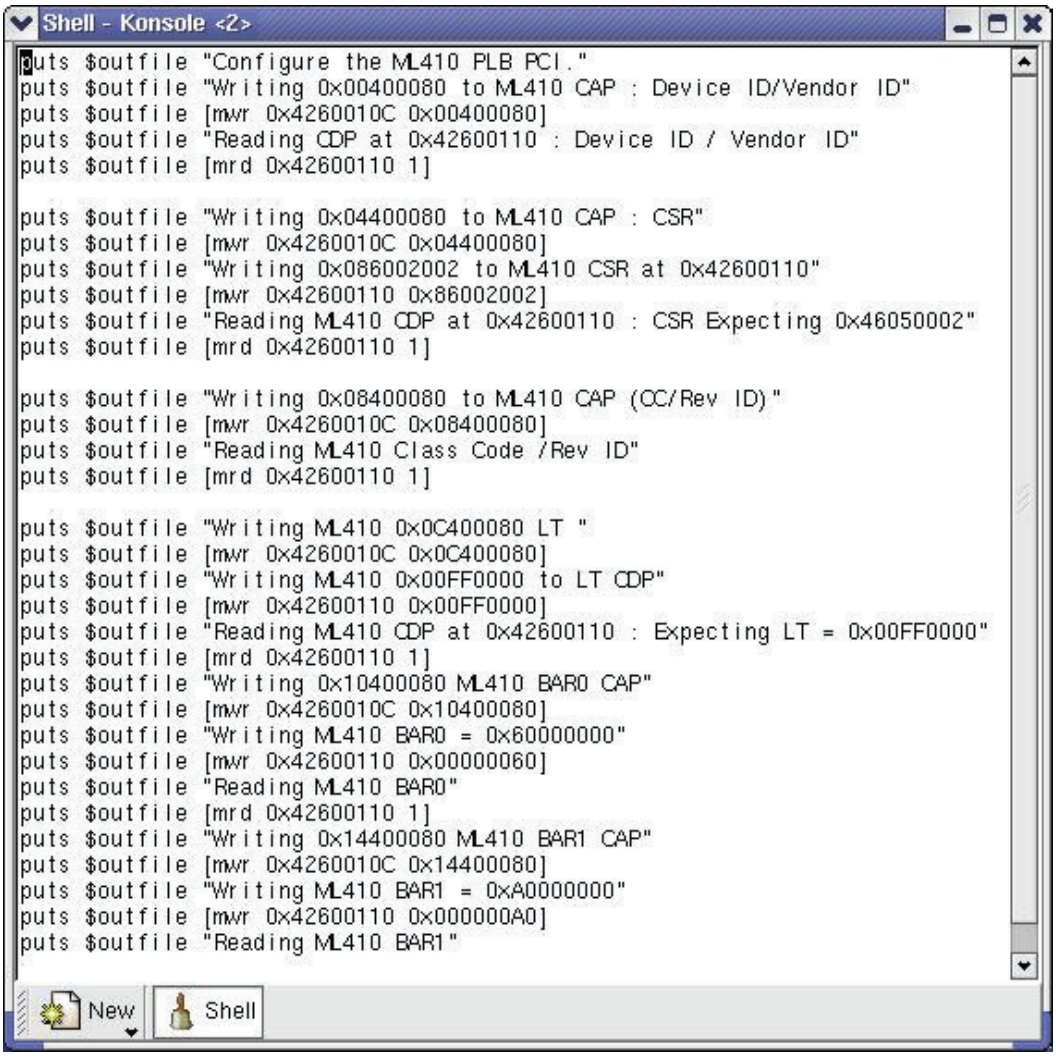


Figure 8: **XMD Commands Used in Configuration**

Software Projects

The reference system contains the following software projects.

hello_pci. This project enables master transactions, sets the latency timer, defines the bus number/subordinate bus number, and scans the PCI bus configuration space headers.

pci_dma. This project runs DMA operations. The user sets the source address, destination address, and DMA length. This code is used for DMA operations between a variety of source and destination addresses. Figure 9 shows the parameters in `pci_dma.c` which can be edited to run DMA transactions between different memory regions.

```
define MEM_0_BASEADDR 0x20000000
define MEM_1_BASEADDR 0x20002000

..

DMALength = 1024
```

X1038_09_011108

Figure 9: **Defining Source and Destination Addresses, Length in pci_dma.c**

DMA Transactions

Many of the XMD scripts and C code examples generate Direct Memory Access (DMA) operations. DMA transactions are initiated by writing to the Control, Source Address, Destination Address, and Length registers of the DMA controller. Table 4 provides these register locations of the XPS Central DMA controller.

Table 4: **XPS Central DMA Registers**

Control Register	C_BASEADDR + 0x04
Source Address Register	C_BASEADDR + 0x08
Destination Address Register	C_BASEADDR + 0x0C
Length Register	C_BASEADDR + 0x10

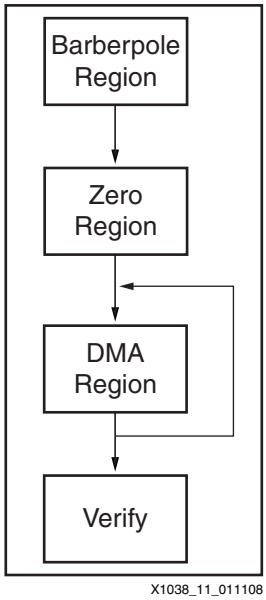
An example of XMD code which generates DMA transactions is given in Figure 10.

```
# Write DMA Control Register
mwr 0x80200004 0xC0000004
# Write DMA Source Address Register
mwr 0x80200008 0x20000000
# Write DMA Destination Address Register
mwr 0x8020000C 0x20002000
# Write DMA Length
mwr 0x80200010 64
```

X1038_10_011108

Figure 10: **Generating DMA Transactions**

The `pci_dma.c` code consists of the four functions in the functional diagram in Figure 11. The Barberpole Region function provides a rotating data pattern on the memory located at the source address. The Zero Region function sets the memory located at the destination address to all zeroes. The DMA Region function performs a DMA transaction of data located at the source address to the memory at the destination address. The Verify function verifies that data at the source address and destination address are equal.



X1038_11_011108

Figure 11: **Functional Diagram of pci_dma.c**

Figure 12 show the Hyperterminal output when running the `pci_dma/executable.elf`.

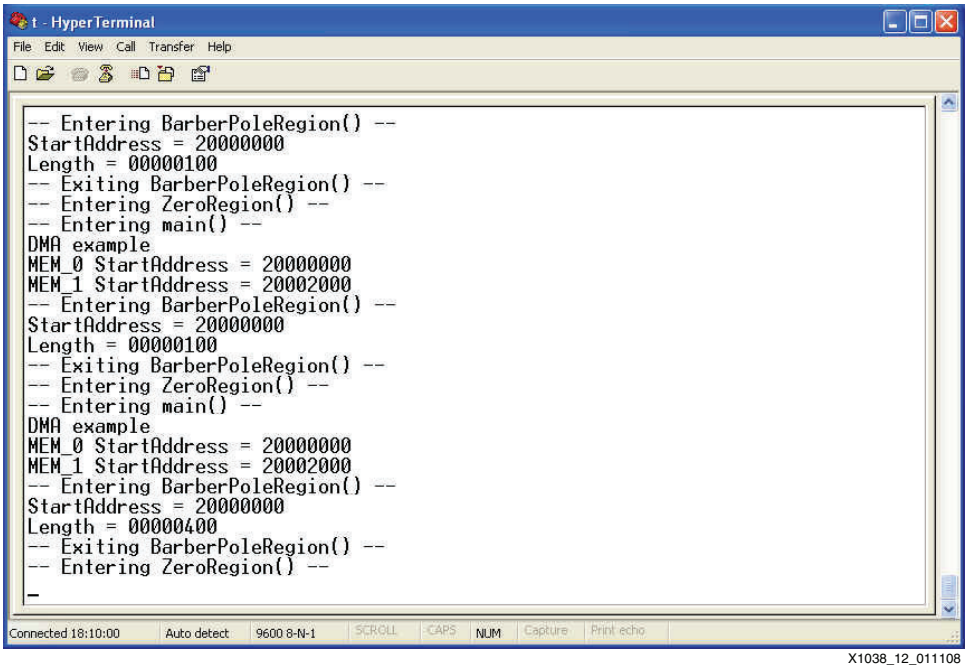


Figure 12: **pci_dma.c output**

Running the Applications

The selection of the hello_pci project is shown in Figure 13. Make the hello_pci project active and the remaining software projects inactive.

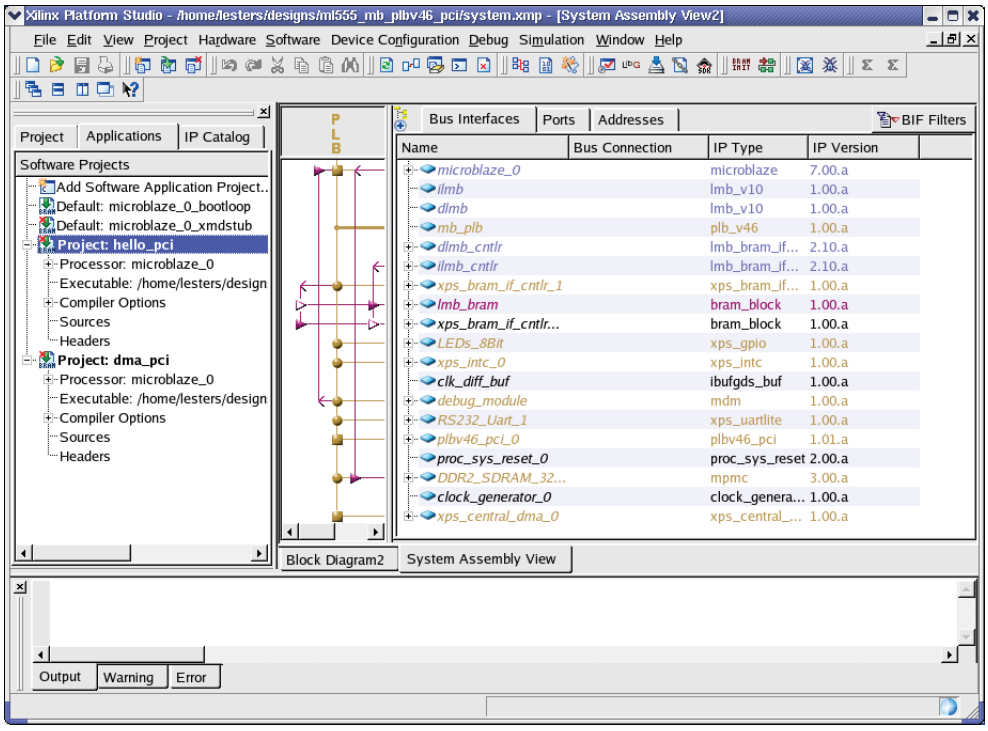


Figure 13: **Selecting the hello_pci Software Project**

With hello_pci selected, right click to build the project. Connect a serial cable to the RS232C port on the ML410 board. Start a HyperTerminal session. Set the baud rate to **9600**, number of data bits to **8**, no parity, and no flow control, as shown in Figure 14.

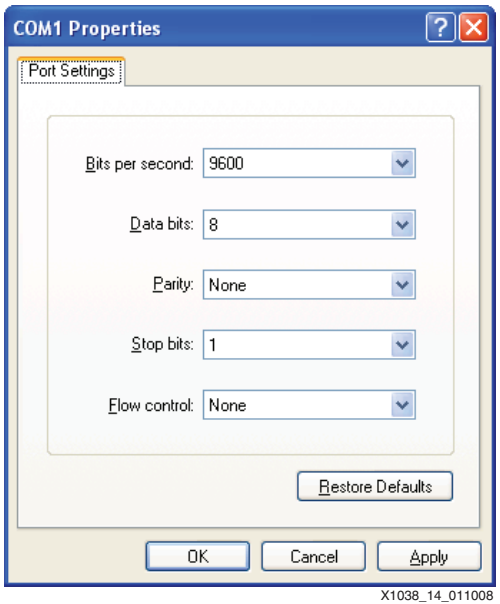


Figure 14: **HyperTerminal Parameters**

From XPS, start **XMD** and enter **`connect ppc hw`** and **`rst`** at the XMD prompt. Invoke GDB and select Run to start the application as shown in Figure 15. The `hello_pci.c` code, originally written for the ML310 Embedded Development Platform, runs without any modifications on this reference system.

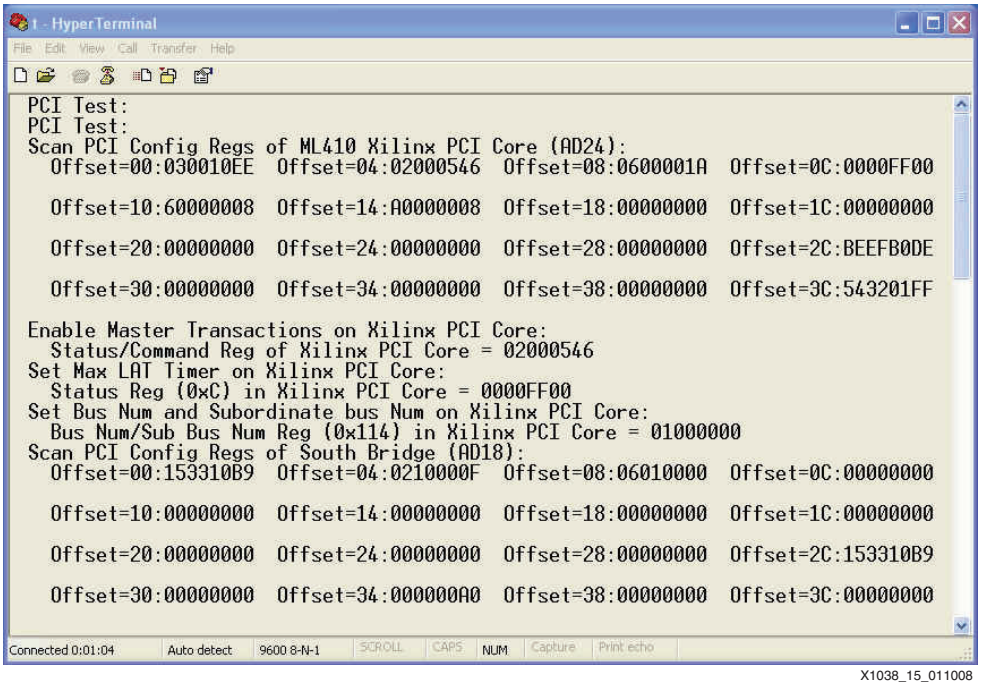


Figure 15: **Running hello_pci**

Using ChipScope with PLBv46 PCI

Because of limited JTAG BSCAN resources, using ChipScope Inserter in a Spartan-3 FPGA usually results in overmapping errors. To avoid these errors, the ICON and the IBA and ILA cores are included in the mhs file, and the design is implemented just as when the ChipScope cores are not included.

Figure 16 shows the use of ChipScope cores in the mhs file.

```
BEGIN chipscope_icon
 PARAMETER INSTANCE = chipscope_icon_0
 PARAMETER HW_VER = 1.01.a
 PARAMETER C_NUM_CONTROL_PORTS = 1
 PARAMETER C_SYSTEM_CONTAINS_MDM = 1
 PORT control0 = chipscope_plbv46_iba_0_icon_ctrl
 ORT control1 = chipscope_plbv46_iba_0_icon_ctrl
END

BEGIN chipscope_ila
 PARAMETER INSTANCE = chipscope_ila_0
 PARAMETER HW_VER = 1.01.a
 PARAMETER C_TRIG0_UNITS = 1
 PARAMETER C_TRIG0_TRIGGER_IN_WIDTH = 1
 PARAMETER C_NUM_DATA_SAMPLES = 512
 PARAMETER C_DATA_SAME_AS_TRIGGER = 0
 PARAMETER C_DATA_IN_WIDTH = 50
 PARAMETER C_ENABLE_TRIGGER_OUT = 1
 PARAMETER C_ENABLE_TRIGGER_OUT = 0
 PORT chipscope_ila_control = chipscope_ila_0_icon_control
 PORT DATA = PCI_monitor & RS232_Interrupt & CDMA_Interrupt
 PORT CLK = sys_clk_s
 PORT TRIG0 = PCI_monitor[0]  & CDMA_Interrupt & mb_plb_
 PLB_PAValid & mb_plb_PLB_MAddrAck
 PORT TRIG0 = PCI_monitor[0]
END


BEGIN chipscope_plbv46_iba
 PARAMETER INSTANCE = chipscope_plbv46_iba_0
 PARAMETER HW_VER = 1.00.a
 PARAMETER C_NUM_DATA_SAMPLES = 512
 PARAMETER C_USE_MU_5_RD_DBUS = 1
 PARAMETER C_USE_MU_4_WR_DBUS = 1
 PARAMETER C_MU_1_TRIG_IN_WIDTH = 1
 BUS_INTERFACE MON_PLB = mb_plb
 PORT chipscope_icon_control = chipscope_plbv46_iba_0_icon_ctrl
 PORT PLB_Clk = sys_clk_s
END
```

X1038_16_011108

Figure 16: **Instantiating ChipScope cores in mhs file**

The PCI_Monitor signals are the PCI bus signals: AD, CBE, and the remaining PCI Bus signals. Table 5 defines the functionality of the PCI_Monitor signals. The Filter Pattern *PCI_Monitor* is used to locate the PCI bus signals.

Table 5: **PCI Monitor Signals**

Bit Position	PCI Signal
0	FRAME_N
1	DEVSEL_N
2	TRDY_N
3	IRDY_N
4	STOP_N
5	IDSEL_int
6	INTA
7	PERR_N
8	SERR_N
9	Req_N_toArb
10	PAR
11	REQ_N
12:43	AD
44:47	CBE

To begin debugging, invoke ChipScope Pro Analyzer by selecting

Start → Programs → ChipScope Pro → ChipScope Pro Analyzer

Click on the Chain icon located at the top left of Analyzer GUI. Verify that the message in the transcript window indicates that an ICON is found.

The ChipScope Analyzer waveform viewer displays signals named `DATA*`. To replace the `DATA*` signal names with the signal names, select **File → Import** and enter the ila and iba cdc files from the implementation directory in the dialog box.

The Analyzer waveform viewer is more readable when buses rather than discrete signals are displayed. Select the **32 PLB_ABus<*>** signals, click the right mouse button, and select **Add to Bus → New Bus**. With PLB_ABus<0:31> in the waveform viewer, select and remove the 32 discrete **PLB_ABus<*>** signals.

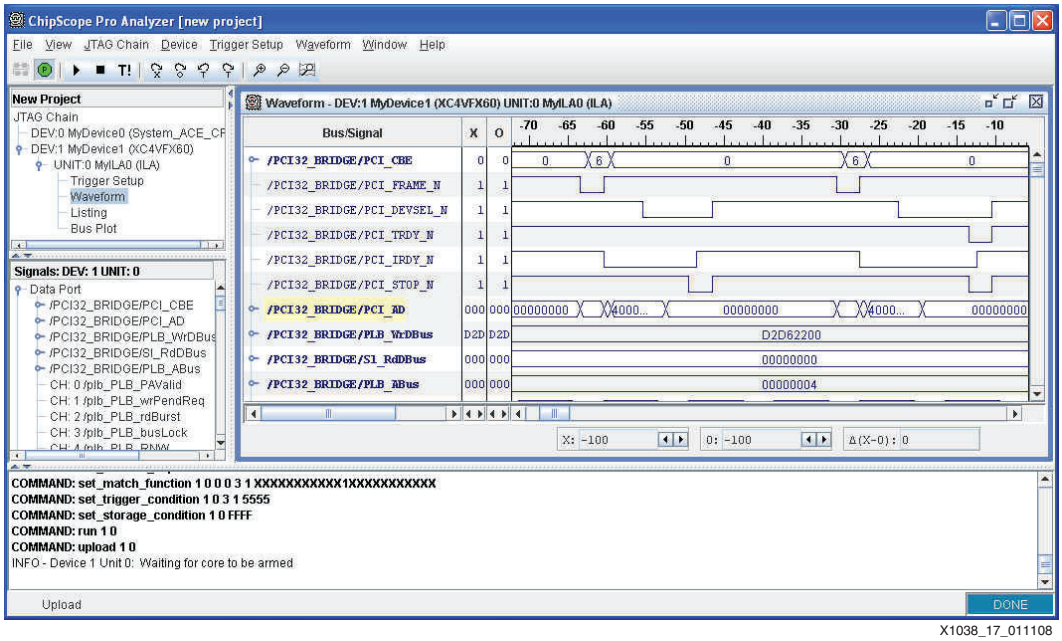
Repeat this for the PLB data buses. Make PCI Bus signals by creating a new bus for PCI_Monitor(44:47). Rename PCI_Monitor(44:47) PCI_CBE. Make PCI Bus signals for PCI_Monitor(12:43) and renaming it PCI AD.

Note: The Reverse Bus Order operation is useful for analyzing bus signals.

1. Set the trigger in the Trigger Setup window. The trigger used depends on the problem being debugged. For example, if debugging a configuration transaction from the ML410 PLB, trigger on an PLB address of C_BASEADDR + 0x10C. If debugging a problem configuring from the PCI side, trigger on the PCI_Monitor(44:47) for a configuration write on PCI_CBE. Change the Windows to N samples to a setting of **500**. Arm the trigger by selecting **Trigger Setup → Arm**, or clicking on the **Arm** icon.
2. Run **XMD** or **GDB** to activate trigger patterns which cause ChipScope to display meaningful output. For example, set the trigger to PA_Valid = **1**, and run `xmd -tcl wr_410_s3.tcl` at the command prompt.

3. ChipScope results are analyzed in the waveform window as shown in Figure 17. This figure shows the original PCI_monitor<*> signals and the PCI_monitor_ad signal generated. The waveforms may be easier to read if the discrete PCI_monitor<*> signals are removed after they are renamed.

   To share the results with remote colleagues, save the results in the waveform window as a Value Change Dump (vcd) file. The vcd files can be translated and viewed in most simulators. The `vcd2wlf` translator in ModelSim reads a vcd file and generates a ModelSim waveform log file (wlf) file for viewing in the ModelSim waveform viewer. The vcd file can be opened in the Cadence Design System, Inc. Simvision design tool by selecting **File → Open Database**.



X1038_17_011108

Figure 17: **ChipScope Analyzer Results**

References

1. DS207 *PCI 64/32 Interface v3.0 Data Sheet*
2. UG159 *LogiCORE IP Initiator/Target v3.1 for PCI*
3. UG262 *LogiCORE IP Initiator/Target v4.5 for PCI*
4. UG085 *ML410 Embedded Development Platform User Guide*
5. UG044 *ChipScope ILA Tools Tutorial*
6. UG241 *OPB PCI v1.02a User Manual*
7. XAPP1001 *PLBv46 PCI Using the ML410 Embedded Development Platform*
8. XAPP998 *PCI Bus Performance Measurements using the Vmetro Bus Analyzer*

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
2/8/08	1.0	Initial Xilinx release.

Notice of Disclaimer