# Direct Memory Access using CDMA

## Introduction

In Zynq, multiple interconnections are available between the PS and PL sections with different performance levels for data transfer between the two subsystems. The General Purpose (GP) Master and Slave AXI interconnect used in the previous labs are intended for peripherals that do not have high bandwidth requirements. E.g. switches, leds, keyboard, mouse. There are four High Performance PS slave to PL master AXI interfaces available for peripherals that need higher bandwidth. E.g. Video and image processing applications. This lab guides you through the process of enabling a High Performance AXI slave port in the PS, adding an AXI central DMA (CDMA) controller, and performing Direct Memory Access (DMA) operations between various memories.

## Objectives

After completing this lab, you will be able to:

- Enable a High Performance (HP) port of the processing system
- Add and connect the CDMA controller in the programmable logic
- Perform DMA operation between various memories

## Procedure

This lab is separated into steps that consist of general overview statements that provide information on the detailed instructions that follow. Follow these detailed instructions to progress through the lab.

## Design Description

In this lab, you will enable the HP port of the PS and add an instance of the Central DMA (CDMA) controller in the PL. You will also add another instance of an AXI-BRAM controller to access the second port of the BRAM via the processor. You will connect the interrupt request line from the CDMA to the input of the GIC of the PS. The following diagram represents the completed design (**Figure 1**).

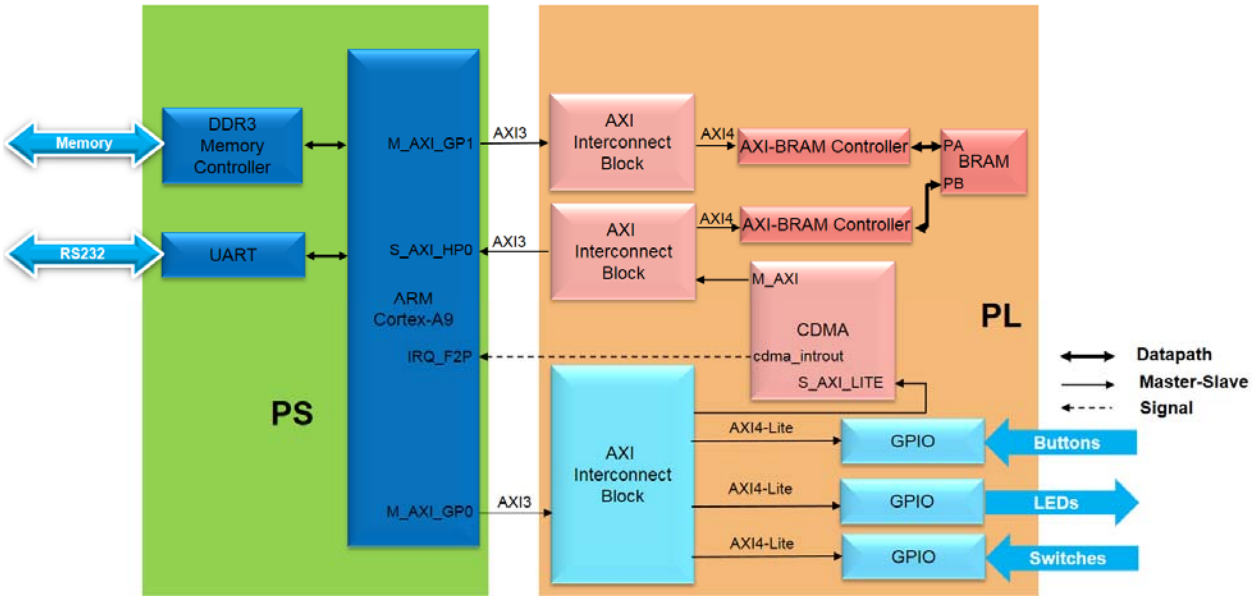
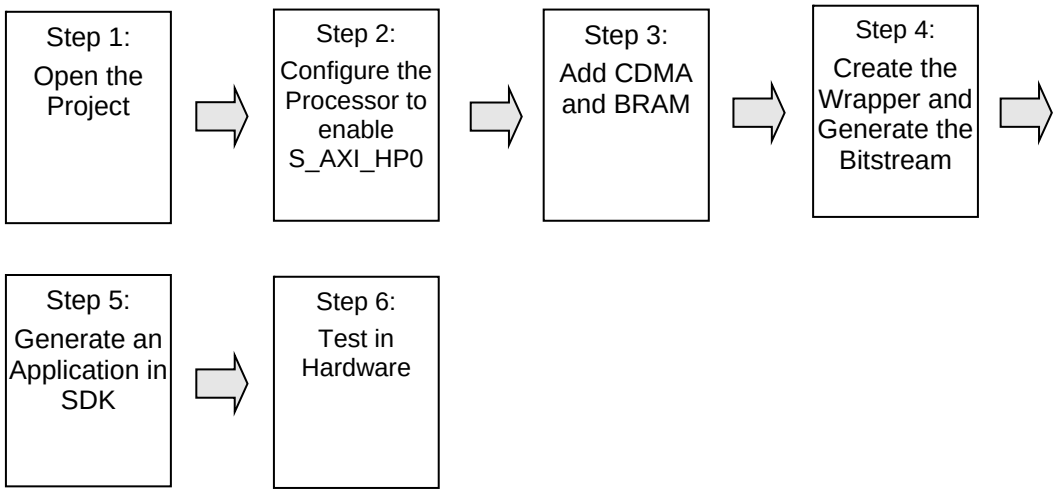



**Figure 1. Completed Design**

## General Flow for this Lab




In the instructions below;
{***sources***} refers to: C:\xup\adv_embedded\2018_2_zynq_sources
{***labs***} refers to : C:\xup\adv_embedded\2018_2_zynq_labs

Board support for PYNQ-Z1 and PYNQ-Z2 are not included in Vivado 2018.2 by default. The relevant zip file need to be extracted and saved to: {Vivado installation}\data\boards\board_files\.

These files can be downloaded from the XUP webpage (http://www.xilinx.com/support/university/vivado/vivado-workshops/Vivado-adv-embedded-design-zynq.html ) where this material is also hosted.

## Open the Project — Step 1

### 1-1. Open the Vivado program. Open the *lab3* project you created earlier or use the *lab3* project from the labsolutions directory, and save the project as *lab4*.

**1-1-1.** Start Vivado if necessary and open either the lab3 project (lab3.xpr) you created earlier or the lab3 project in the labsolution directory using the **Open Project** link in the Getting Started page.

**1-1-2.** Select **File > Project > Save As …** to open the *Save Project As* dialog box. Enter **lab4** as the project name. Make sure that the *Create Project Subdirectory* option is checked, the project directory path is **{labs}** and click **OK**.

**1-1-3.** This will create the lab4 directory and save the project and associated directory with lab4 name.

## Configure the Processor to Enable S_AXI_HP0 — Step 2

### 2-1. Open the Block Design and enable the S_AXI_HP0 interface

**2-1-1.** Click **Open Block Design** in the *Flow Navigator* pane

**2-1-2.** Double-click on the *Zynq processing system* instance to open its configuration form.

**2-1-3.** Select *PS-PL Configuration* in the Page Navigator window in the left pane, expand *HP Slave AXI Interface* on the right, and click on the check-box of the **S AXI HP0 Interface** to enable it, and click **OK** to close the Configuration window.

# Add CDMA and BRAM Step 3

## 3-1. Instantiate the AXI central DMA controller.

**3-1-1.** Click the + button and search for **Central** in the catalog.

**3-1-2.** Double-click the **AXI Central Direct Memory Access** to add an instance to the design.

**3-1-3.** Double-click on the *axi_cdma_0* instance and uncheck the *Enable Scatter Gather* option.

**3-1-4.** Change the *Write/Read Data Width* to **64** and click **OK**.

Note the burst size changes from 16 to 8. You can increase this up to 256 to improve the performance. Here we are using smallest number since the application allows small number of words transfer.

## 3-2. Run connection automation

Connection automation could be run on all unconnected ports simultaneously. For the purposes of this lab, each port will be connected separately so that the changes made by the automation process are easier to follow.

**3-2-1.** Click on **Run Connection Automation** and select **processing_system7_0/S_AXI_HP0** only.

**3-2-2.** Check that this port will be connected to the */axi_cdma_0/M_AXI* port and click **OK**.

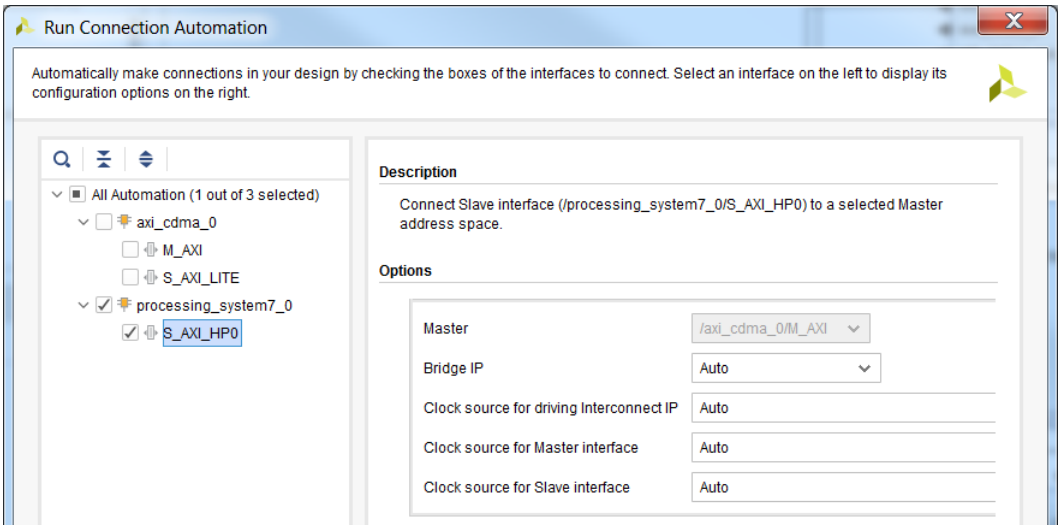



**Figure 2. Connection automation**

**3-2-3.** Verify the CDMA connection through the AXI SmartConnect to the HP0 port

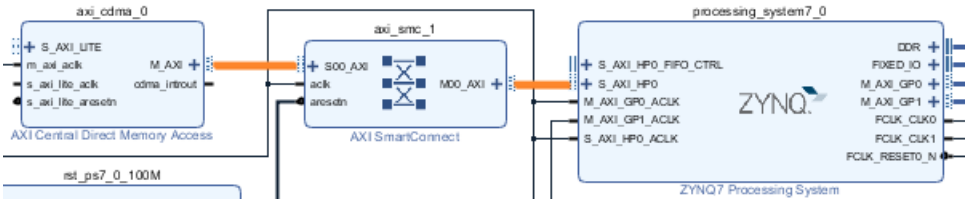



**Figure 3. Connecting AXI Central DMA controller to S_AXI_HP0**

Notice that an instance of AXI SmartConnect (axi_smc_1) is added, S_AXI_HP0 of the processing_system7_0 is connected to M00_AXI of the axi_smc_1, S00_AXI of the axi_smc_1 is connected to the m_axi of the axi_cdm_0 instance. Also, m_axi_aclk of the axi_cdma_0 is connected to the net originating from FCLK_CLK0 of the processing_system7_0.

**3-2-4.** Click on **Run Connection Automation** again, and select **/axi_cdma_0** (which includes **S_AXI_LITE**).

Notice that the axi_cdma_0/M_AXI port is no longer available to select. This is because it was connected to the processing system in the previous step.

**3-2-5.** Ensure /processing_system7_0/*M_AXI_GP0* is selected in the drop-down button and click **OK**.

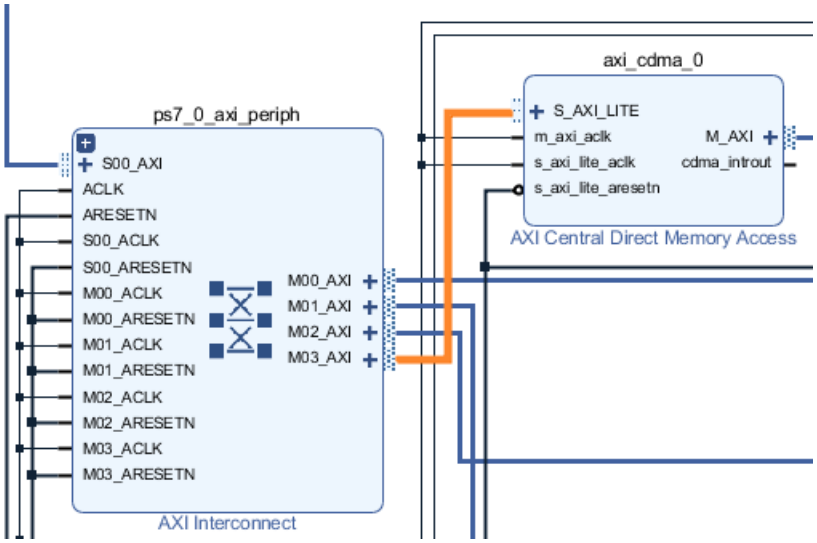



**Figure 4. CDMA connected**

## 3-3. Instantiate another BRAM Controller and a BRAM.

**3-3-1.** Click the + button and search for **BRAM** in the catalog.

**3-3-2.** Double-click the **AXI BRAM Controller** to add an instance to the design.

**3-3-3.** Click on **Run Connection Automation**, and select **/axi_bram_ctrl_1/S_AXI** only.

**3-3-4.** For the *Master* connection, select **axi_cdma_0/M_AXI** from the dropdown box.

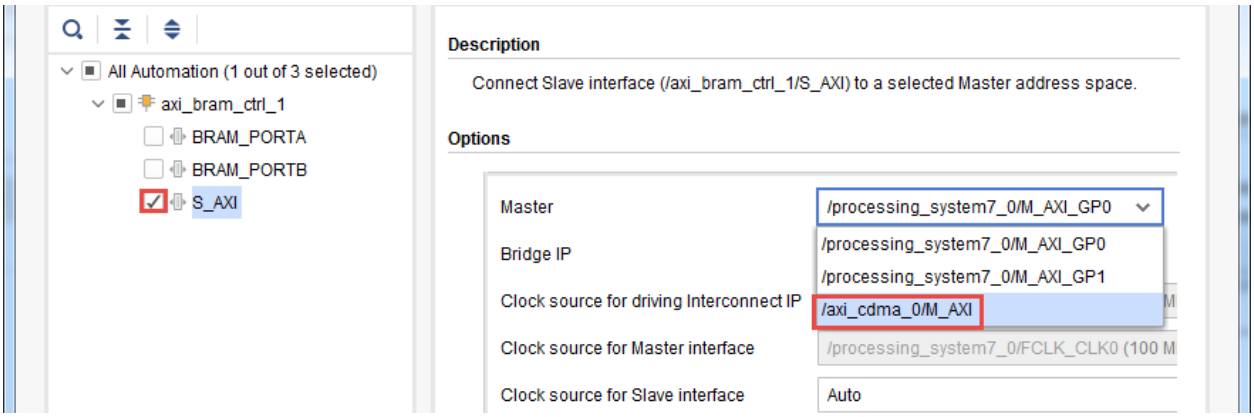



**Figure 5. BRAM connection automation**

**3-3-5.** Click **OK** to make the connection.

Notice that another axi interface (M01_AXI) is added to the axi_smc_1 instance and is connected to the S_AXI interface of the axi_bram_ctrl_1 instance.

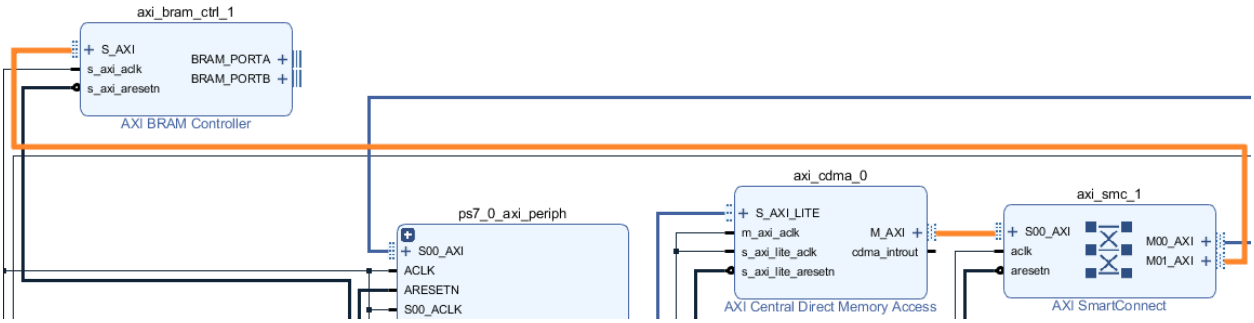



**Figure 6. Connection between the new BRAM controller to the CDMA**

**3-3-6.** Double-click the *axi_bram_ctrl_1* instance and change the *Number of BRAM Interface* to **1**.

**3-3-7.** Change the *Data Width* to **64** and click **OK**.

**3-3-8.** Double-click the *axi_bram_ctrl_0* instance and also change the *Number of BRAM Interface* to **1**. Click **OK**.

**3-3-9.** Using the wire tool, connect the **BRAM_PORTA** of the *axi_bram_ctrl_1* instance to the **BRAM_PORTB** of the Block Memory Generator *axi_bram_ctrl_0_bram* instance.

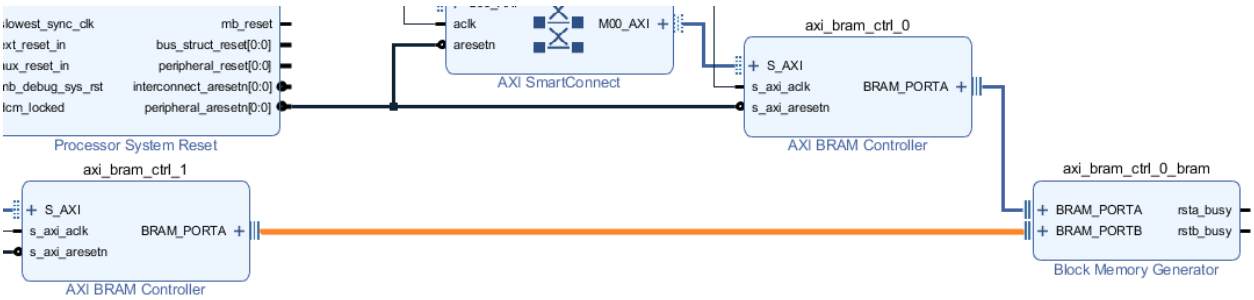



**Figure 7. Connect the second BRAM controller**

## 3-4. Connect the CDMA interrupt out port to the port of the processor.

**3-4-1.** Double-click on the *processing_system7_0 instance* to open its configuration form.

**3-4-2.** Select *Interrupts* in the Page Navigator window in the left pane, check the *Fabric Interrupts* box.

**3-4-3.** Expand *Fabric Interrupts* > *PL-PS Interrupts Ports*, and click on the check-box of the **IRQ_F2P**.

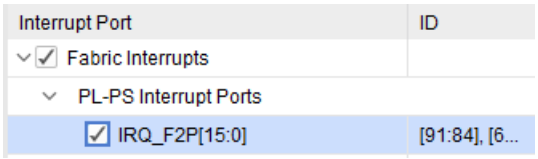



**Figure 8. Enabling the processor interrupt**

**3-4-4.** Click **OK**.

**3-4-5.** Using wiring tool, connect the **cdma_introut** to the **IRQ_F2P** port. (Click on the *cdma_introut* port and drag to the *IRQ_F2P* port)

## 3-5. Using the Address Editor tab, set the BRAM controller size to 64KB. Validate the design.

**3-5-1.** Select the **Address Editor** tab.

**3-5-2.** Expand the *axi_cdma_0*> *Data* section, and change the memory size of *axi_bram_ctrl_1* to **64K**.

Diagram | Address Editor

| Cell | Slave Interface | Base Name | Offset Address | Range | High Address |
|---|---|---|---|---|---|
| processing_system7_0 | | | | | |
| Data (32 address bits : 0x40000000 [ 1G ],0x80000000 [ 1G ]) | | | | | |
| axi_bram_ctrl_0 | S_AXI | Mem0 | 0x8000_0000 | 64K | 0x8000_FFFF |
| buttons | S_AXI | Reg | 0x4120_0000 | 64K | 0x4120_FFFF |
| leds | S_AXI | Reg | 0x4121_0000 | 64K | 0x4121_FFFF |
| switches | S_AXI | Reg | 0x4122_0000 | 64K | 0x4122_FFFF |
| axi_cdma_0 | S_AXI_LITE | Reg | 0x7E20_0000 | 64K | 0x7E20_FFFF |
| axi_cdma_0 | | | | | |
| Data (32 address bits : 4G) | | | | | |
| processing_system7_0 | S_AXI_HP0 | HP0_DDR_LOWOCM | 0x0000_0000 | 512M | 0x1FFF_FFFF |
| axi_bram_ctrl_1 | S_AXI | Mem0 | 0xC000_0000 | 64K | 0xC000_FFFF |

**Figure 9. Address space**

**3-5-3.** The design should look similar to the figure below.

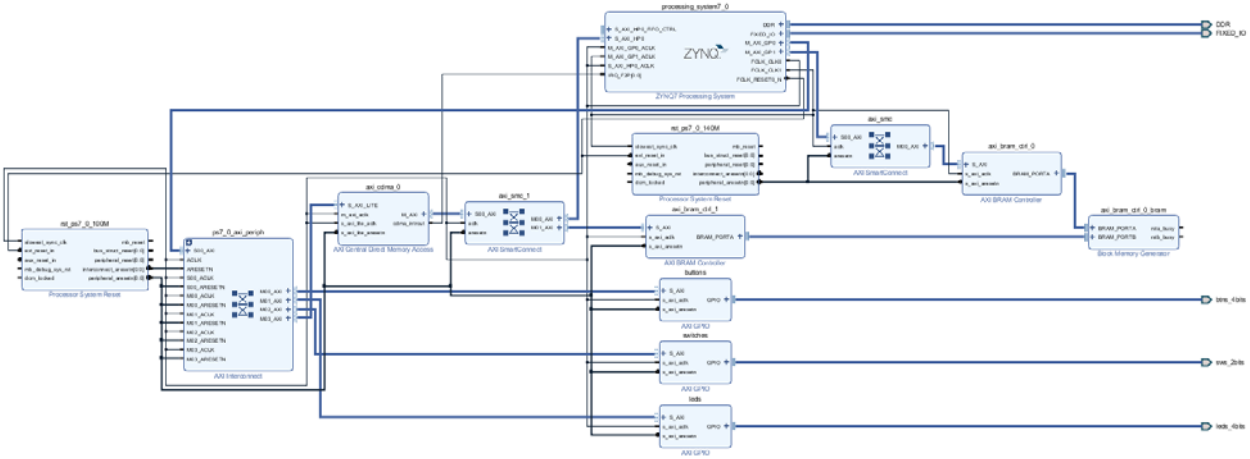


**Figure 10. Completed design**

**3-5-4.** Select the *Diagram* tab, and click on the (Validate Design) button to make sure that there are no errors.

## Generate the Bitstream Step 4

**4-1-1.** Click on the **Generate Bitstream** to run the synthesis, implementation, and bit generation processes.

**4-1-2.** Click **Save** to save the project, and **Yes** if prompted to run the processes. Click **OK** to launch the runs.

**4-1-3.** When the bitstream generation process has completed successfully, click **Cancel**.

## Generate an Application in the SDK Step 5

### 5-1. Export the implemented design, and start SDK

**5-1-1.** Export the hardware configuration by clicking **File > Export > Export Hardware…**

**5-1-2.** Click the box to *Include Bitstream* and click **OK** (Click *Yes* if prompted to overwrite a previous module)

**5-1-3.** Launch SDK by clicking **File > Launch SDK** and click **OK**

**5-1-4.** To clean the workspace, right-click on each open project except *system_wrapper_hw_platform_2* and select close project.

### 5-2. Create an empty application project, named lab4, and import the provided lab4.c file.

**5-2-1.** Select **File > New** > **Application Project.**

**5-2-2.** In the *Project Name* field, enter **lab4** as the project name.

**5-2-3.** Leave the default settings to create a new *Board Support Package* and click **Next.**

**5-2-4.** Select the **Empty Application** template and click **Finish.**

The lab4 project will be created in the Project Explorer window of SDK.

**5-2-5.** Select **lab4 > src** in the project view, right-click, and select **Import.**

**5-2-6.** Expand the **General** category and double-click on **File System.**

**5-2-7.** Browse to the **{sources}\lab4** folder.

**5-2-8.** Select **lab4.c** and click **Finish.**

## Test in Hardware Step 6

### 6-1. Connect and power up the board. Download the bitstream and program the FPGA.

**6-1-1.** Connect and power up the board.

**6-1-2.** In SDK, select **Xilinx > Program FPGA.**

**6-1-3.** Click the **Program** button to program the FPGA.

### 6-2. Establish serial communication, and run the lab4 application from the DDR3 memory.

**6-2-1.** Connect the terminal by selecting the appropriate COM port and setting the Baud Rate to **115200**.

On ZedBoard you will see that the terminal window closes when you try to input your selection. In this case you can either use SDK Terminal (you must hit Enter key after your selection) or use third party comminucation programs such as TeraTerm, Putty etc.

On Zybo, you can continue using the Terminal window.

**6-2-2.** Run the **lab4** application**.**

Follow the menu in the terminal emulator window and test transfers between various memories.

**6-2-3.** Select option 4 in the menu to complete the execution.

**6-2-4.** Close the SDK and Vivado programs by selecting **File > Exit** in each program.

**6-2-5.** Turn OFF the power on the board.

## Conclusion

This lab led you through adding a CDMA controller to the PS so that you can perform DMA transfers between various memories. You used the high-performance port so DMA could be done between the BRAM residing in the PL section and DDR3 connected to the PS. You verified the design functionality by creating an application and executing it from the DDR3 memory.