

LAB 1.

Simple Hardware Design

Introduction

This lab guides you through the process of using Xilinx Platform Studio (XPS) to create a simple processor system targeting the Spartan-3E Starter Kit

Objectives

After completing this lab, you will be able to:

- Create an XPS project by using the Base System Builder (BSB)
- Create a simple hardware design by using Xilinx IP cores available in the Embedded Development Kit

Design Description

The purpose of the lab exercises is to walk you through a complete hardware and software processor system design. Each lab will build upon the previous lab. The following diagram represents the completed design ([Figure 1](#)).

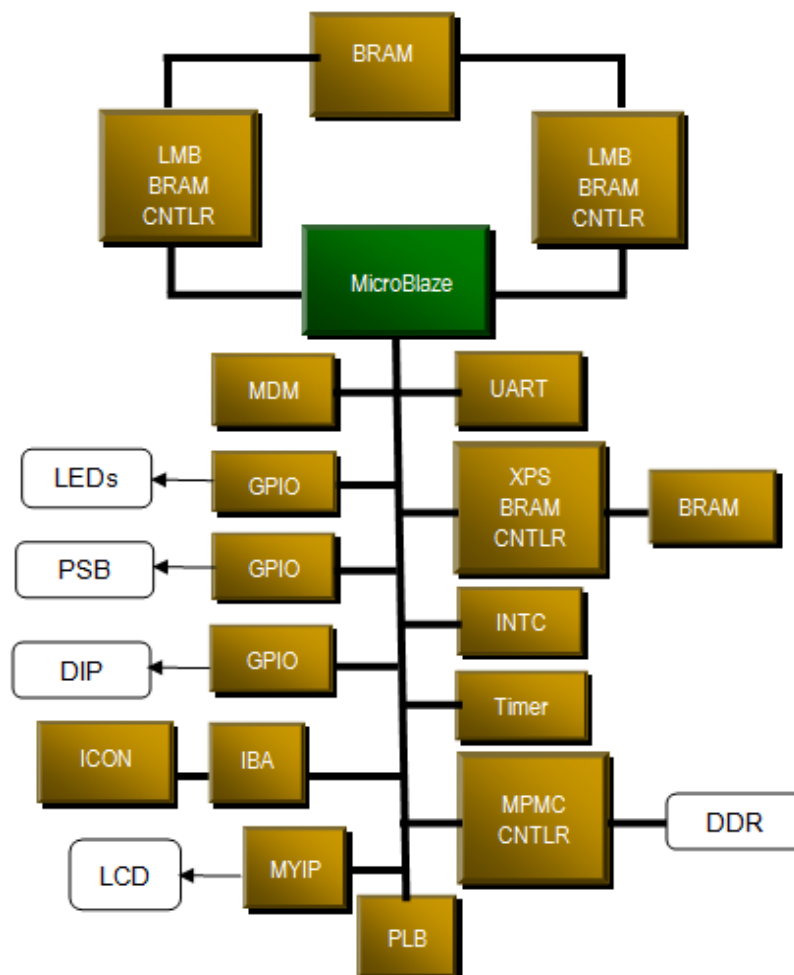


Figure 1. Completed Design

In this lab, you will use the BSB of the XPS system to create a processor system consisting of the following processor IP (**Figure 2**):

- MicroBlaze (version 8.20.a)
- PLB_MDM
- LMB BRAM controllers for BRAM
- BRAM
- UART for serial communication
- GPIO for LEDs
- MPMC controller for external DDR_SDRAM memory

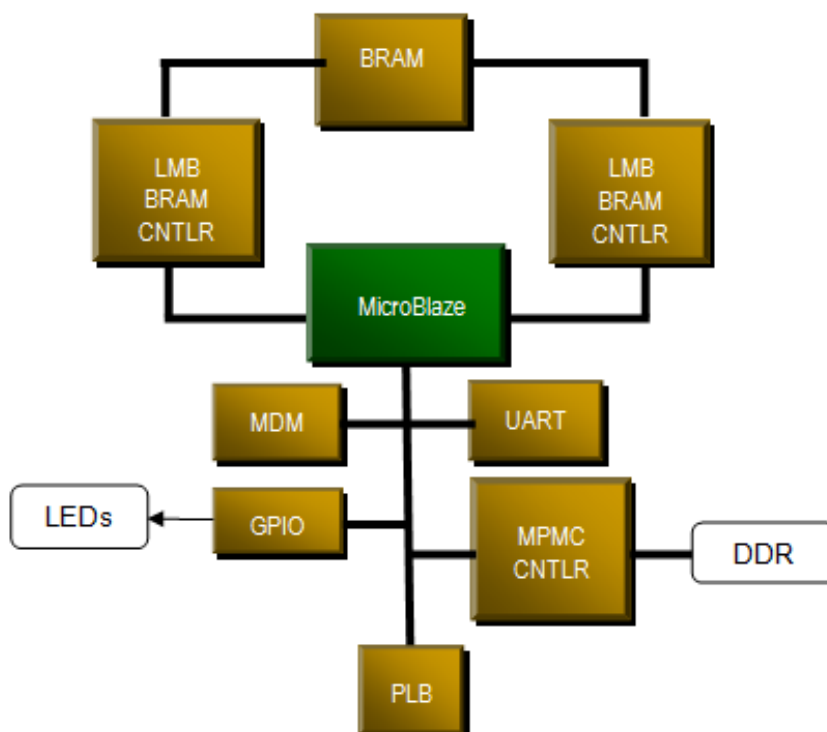


Figure 2. Processor IP

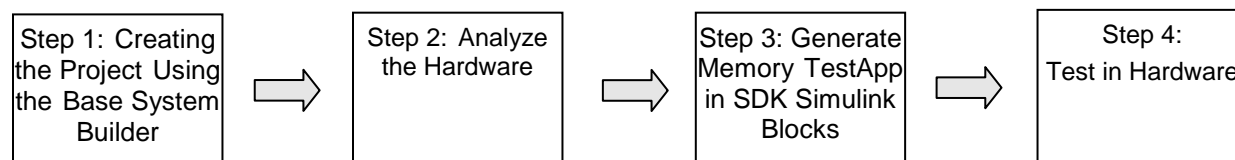
Procedure

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.

This lab comprises 3 primary steps: You will create a project using the base system builder, analyze the hardware, generate memory testApp in SDK and, finally, test in hardware.

Note: If you are unable to complete the lab at this time, you can download the original lab files for this module from the Xilinx University Program site at <http://www.xilinx.com/university>

General Flow for this Lab



Creating the Project Using the Base System Builder

Step 1

- 1-1. **Launch Xilinx Platform Studio (XPS) and create a new project. Use Base System Builder to generate a MicroBlaze system and memory test application targeting the Spartan-3E starter kit.**
- 1-1-1. Open XPS by selecting **Start → All Programs → Xilinx → Xilinx ISE Design Suite 13.2 → EDK → Xilinx Platform Studio**.
- 1-1-2. Leave the default **Base System Builder wizard (recommended)** option and click **OK** to start the wizard (**Figure 3**). If you clicked cancel, you can select **File → New Project** and the same dialog box will appear.

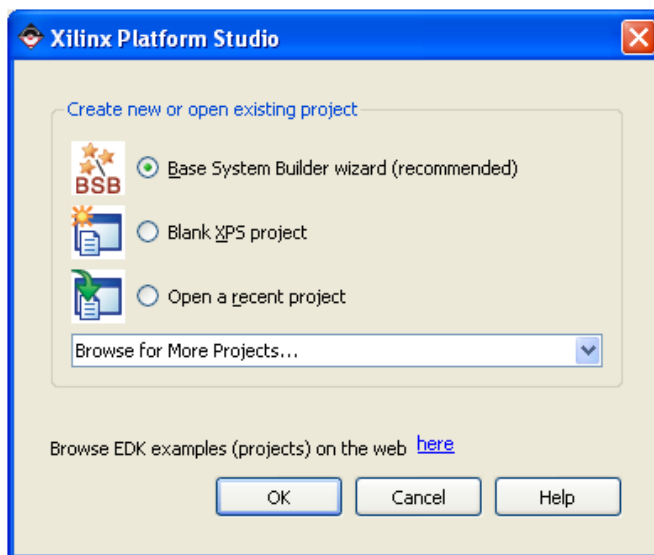


Figure 3. New Project Creation Using Base System Builder

- 1-1-3. Browse to **D:** directory, create a new folder called **lab1** and select it, and click **Open** followed by click **Save (Figure 4)**. Select the **PLB System** option and click **OK**.

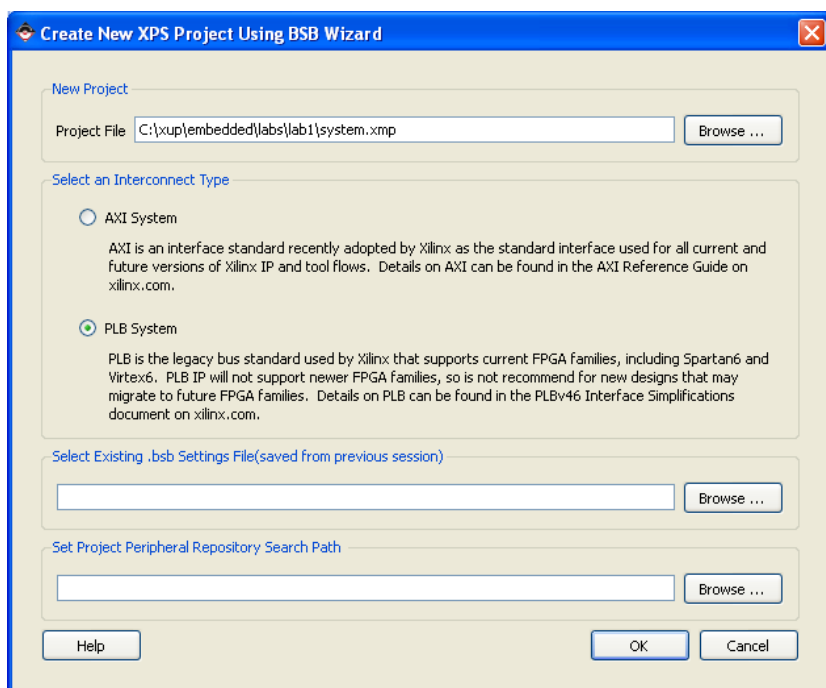


Figure 4. Assigning Project Directory

- 1-1-4.** Select the **I would like to create a new design option** in the Welcome to Base System Builder dialog box and click **Next**.
- 1-1-5.** In the **Board Selection** dialog box, specify the settings below (**Figure 5**) and click **Next** to continue.
- Board Vendor: **Xilinx**
 - Board Name: **Spartan-3E Starter Board**
 - Board Revision (Verify on board): **D**

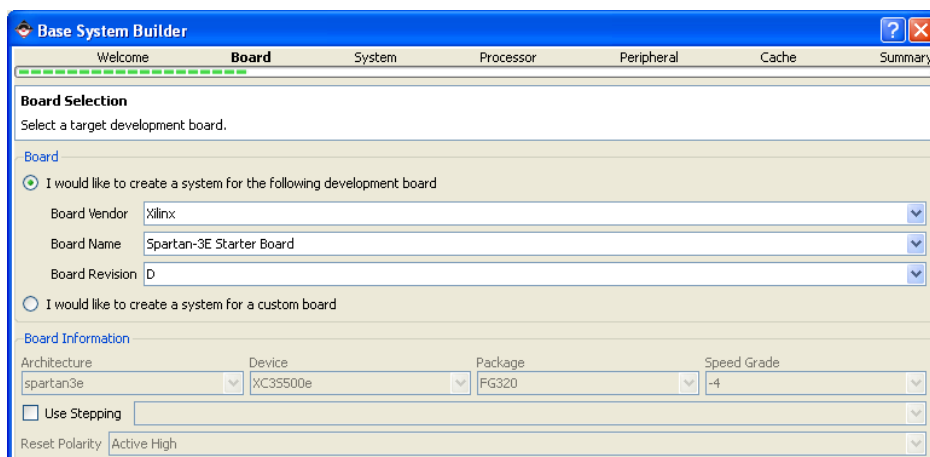


Figure 5. Board Selection Dialog Box

- 1-1-6.** In the **System Configuration** dialog, leave the default **Single-Processor System** option (**Figure 6**) and click **Next**.

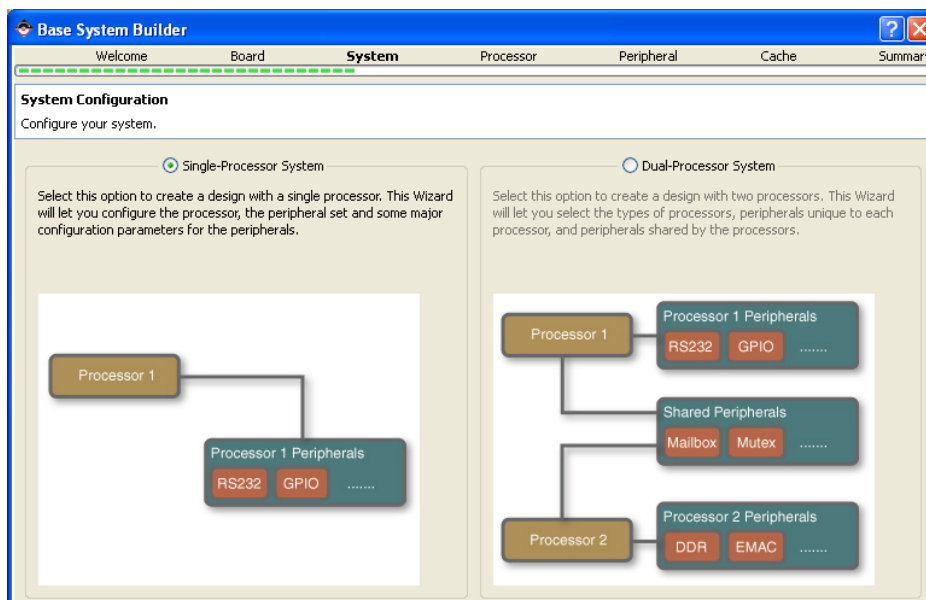


Figure 6. System Configuration Dialog Box

1-1-7. In the **Processor Configuration** dialog box (**Figure 7**), leave the default settings (see below) and click **Next**.

- Reference Clock Frequency: **50 MHz**
- This is the external clock source on the board you are using. This clock will be used to generate the processor and bus clocks.
- Processor type: **MicroBlaze**
- System Clock Frequency - bus Clock Frequency: **50 MHz**
- Local Memory: **8 KB**
- Debug Interface: **On-Chip H/W debug module**

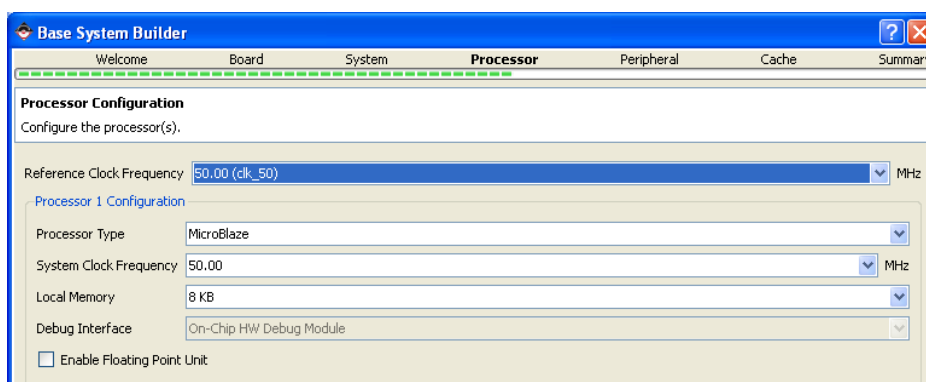


Figure 7. Processor Configuration Dialog Box

1-2. Select and configure the LEDs_8Bit, RS232_DCE, and DDR_SDRAM as the only external devices, and dlmb and ilmb controllers as internal to be used. Generate the memory and peripheral test sample applications and linker script.

- 1-2-1.** In the **Peripheral Configuration** dialog, select and configure the **RS232_DCE**, **LEDs_8Bit** and **DDR_SDRAM** as the external peripherals as shown below, removing the rest of the Processor 1 (MicroBlaze) Peripherals. You won't be able to remove dlmb and ilmb controllers as they are the required internal controllers.
- RS232_DCE: XPS UARTLITE, 115200 baud rate, 8 Data bits, no interrupt, no parity (**Figure 8**)
 - LEDs_8Bit: XPS GPIO. No interrupt (**Figure 9**)
 - DDR_SDRAM: MPMC Controller (Multi-Port Memory Controller) (**Figure 10**)

At this point you could click **Add** to add additional IO peripherals and internal peripherals, but you will see an alternative method in the next lab for adding internal peripherals to an existing project.

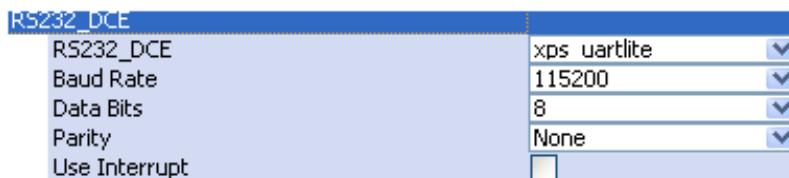


Figure 8. Configure RS-232 DCE

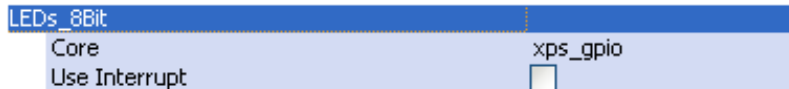


Figure 9. Configure XPS GPIO



Figure 10. Configure DDR_SDRAM with MPMC Controller

Warning: Note that the number of peripherals that appear on each window will depend on the resolution of your monitor.

- 1-2-2.** Click **Next** to display the **Cache Configuration** dialog, make sure nothing is checked. Click **Next** again.

- 1-2-3.** Verify the system summary in the **Summary** dialog (**Figure 11**) and click **Finish**.

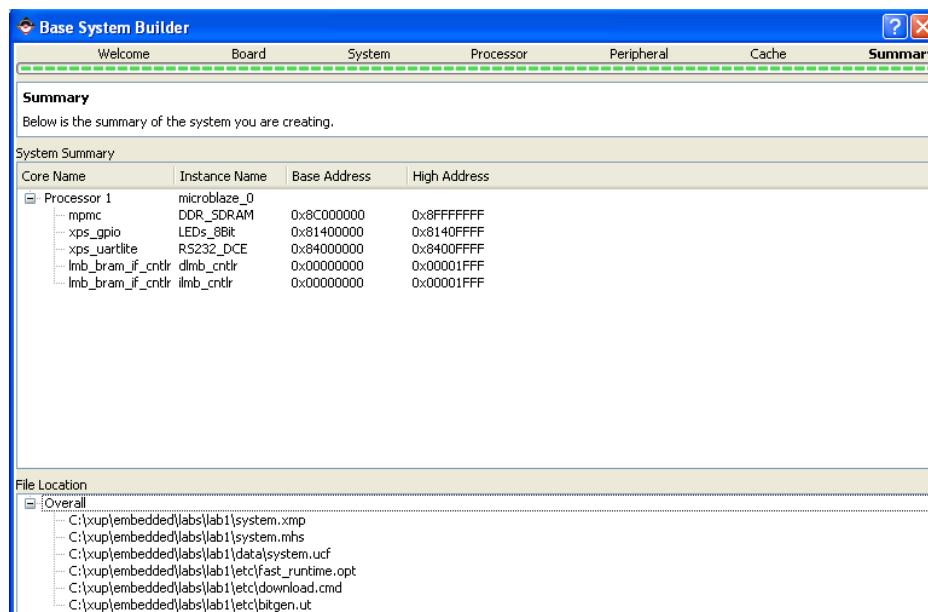


Figure 11. System Summary

- 1-2-4. A System Assembly View will be displayed (**Figure 12**) showing peripherals and busses in the system, and the system connectivity.

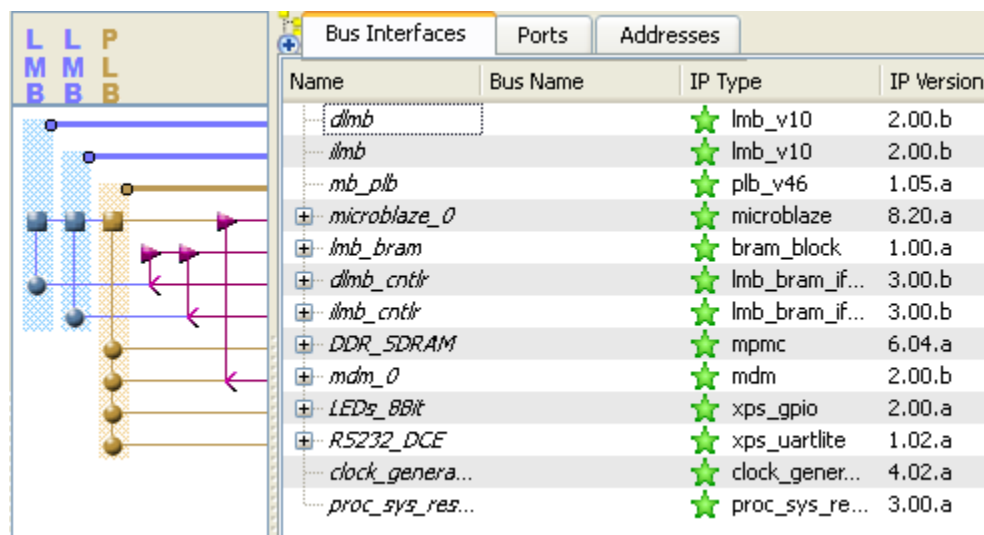


Figure 12. System Assembly View

Analyze the Hardware

Step 2

- 2-1. **Generate a block diagram of the system and study the system components and interconnections. Look in the System Assembly View and analyze the bus and port connections. Run PlatGen to generate the system netlists (NGC) and look in Design Summary. Review the generated files.**
- 2-1-1. Click on the **Generate Block Diagram Image in project** to open a block diagram view (**Figure 13**) and observe the various components that are used in the design.

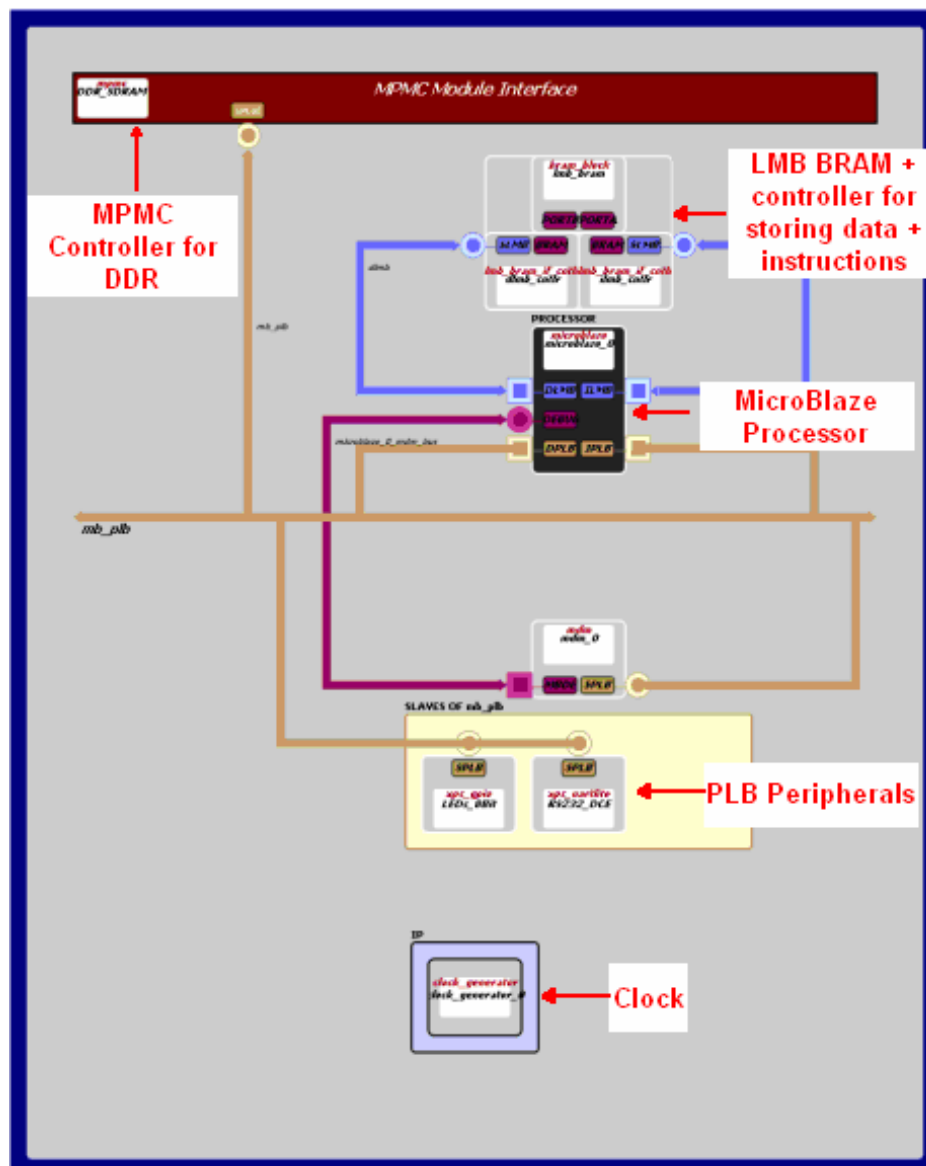


Figure 13. Block Diagram View of the Generated Project

You can zoom in and out and use the scroll bars to navigate around the block diagram. You will see the MicroBlaze™ processor, LMB controller and PLB bus connected to the MicroBlaze processor.

- 2-1-2. In the **System Assembly View** click on plus button and observe the expanded (detailed) bus connection view of the system (**Figure 14**).

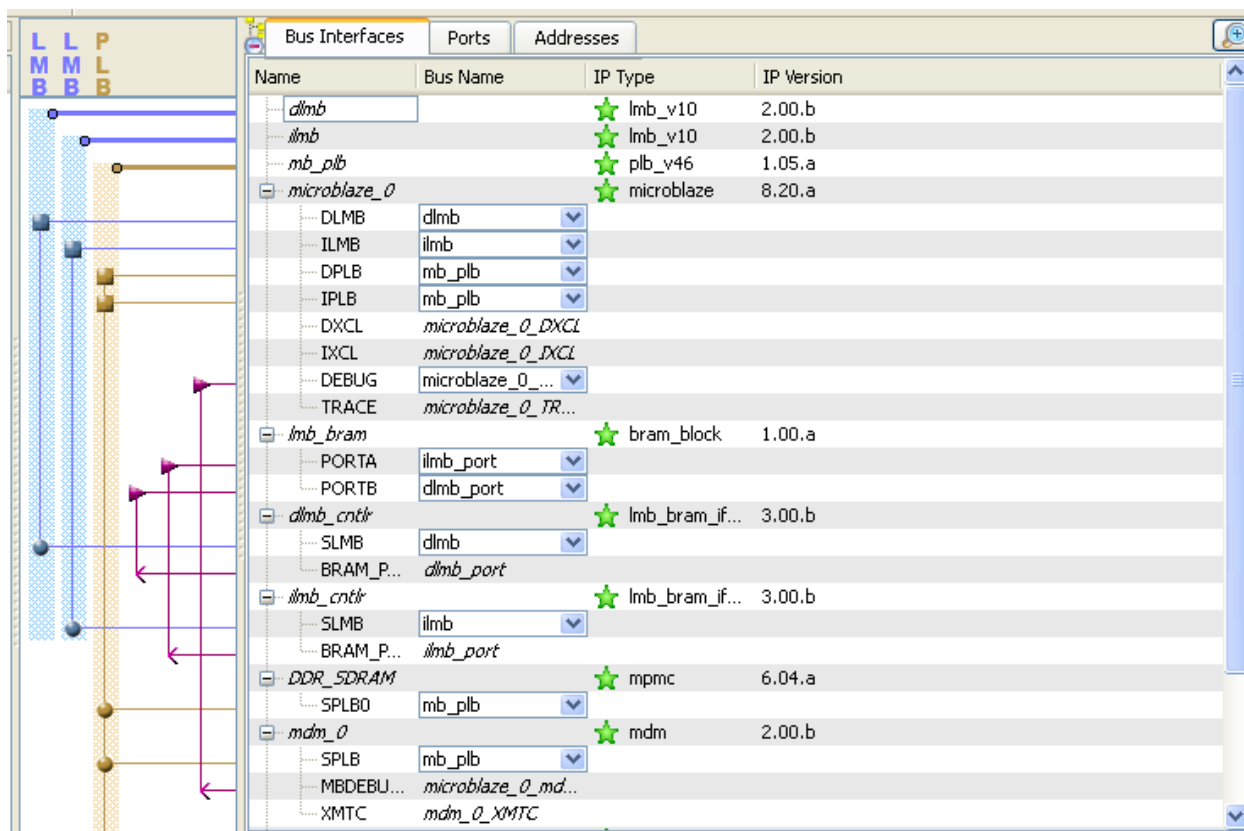


Figure 14. Detailed Bus Connections



Question 1

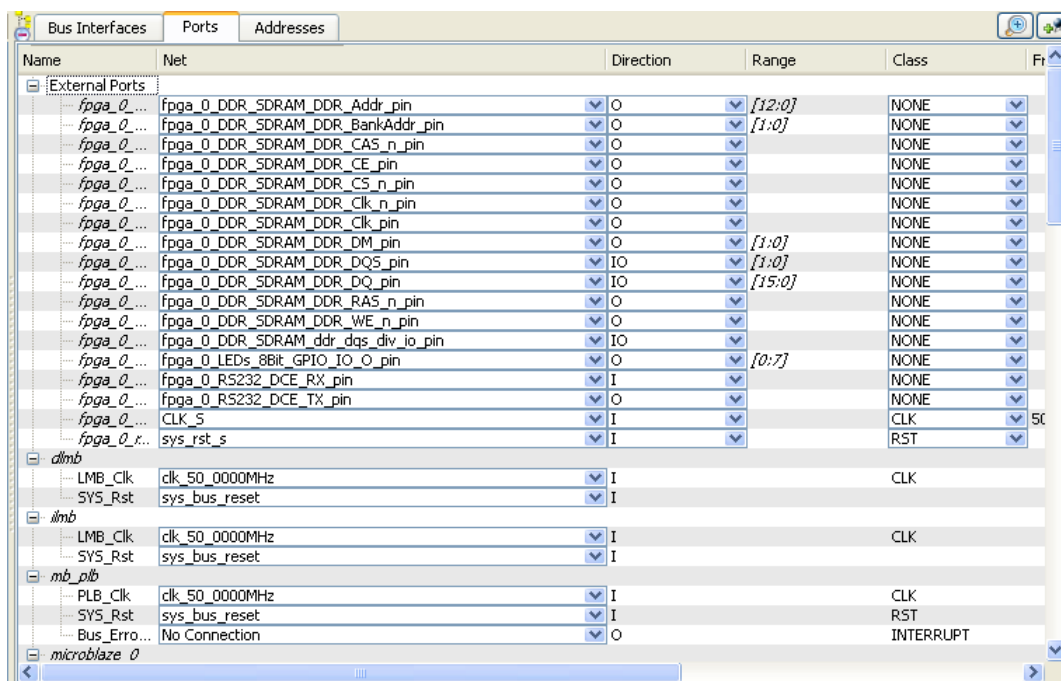
List the name of the bus connection to the following peripherals:

mdm_0: _____

dlmb_cntrl: _____

RS232_DCE: _____

- 2-1-3.** Click on the **Ports** filter and have an expanded view similar to **Figure 15**. This is where you can make internal and external net connections.



Name	Net	Direction	Range	Class	Fr
External Ports					
fpga_0...	fpga_0_DDR_SDRAM_DDR_Addr_pin	O	[12:0]	NONE	
fpga_0...	fpga_0_DDR_SDRAM_DDR_BankAddr_pin	O	[1:0]	NONE	
fpga_0...	fpga_0_DDR_SDRAM_DDR_CAS_n_pin	O		NONE	
fpga_0...	fpga_0_DDR_SDRAM_DDR_CE_pin	O		NONE	
fpga_0...	fpga_0_DDR_SDRAM_DDR_CS_n_pin	O		NONE	
fpga_0...	fpga_0_DDR_SDRAM_DDR_Clk_n_pin	O		NONE	
fpga_0...	fpga_0_DDR_SDRAM_DDR_Clk_pin	O		NONE	
fpga_0...	fpga_0_DDR_SDRAM_DDR_DM_pin	O	[1:0]	NONE	
fpga_0...	fpga_0_DDR_SDRAM_DDR_DQS_pin	IO	[1:0]	NONE	
fpga_0...	fpga_0_DDR_SDRAM_DDR_DQ_pin	IO	[15:0]	NONE	
fpga_0...	fpga_0_DDR_SDRAM_DDR_RAS_n_pin	O		NONE	
fpga_0...	fpga_0_DDR_SDRAM_DDR_WE_n_pin	O		NONE	
fpga_0...	fpga_0_DDR_SDRAM_ddr_dqs_div_io_pin	IO		NONE	
fpga_0...	fpga_0_LEDs_8Bit_GPIO_IO_O_pin	O	[0:7]	NONE	
fpga_0...	fpga_0_RS232_DCE_RX_pin	I		NONE	
fpga_0...	fpga_0_RS232_DCE_TX_pin	O		NONE	
fpga_0...	CLK_S	I		CLK	5C
fpga_0...	sys_rst_s	I		RST	
dmb					
LMB_Clk	clk_50_0000MHz	I		CLK	
SYS_Rst	sys_bus_reset	I			
ilmb					
LMB_Clk	clk_50_0000MHz	I		CLK	
SYS_Rst	sys_bus_reset	I			
mb_plb					
PLB_Clk	clk_50_0000MHz	I		CLK	
SYS_Rst	sys_bus_reset	I		RST	
Bus_Erro...	No Connection	O		INTERRUPT	
microblaze_0					

Figure 15. Ports Filter



Question 2

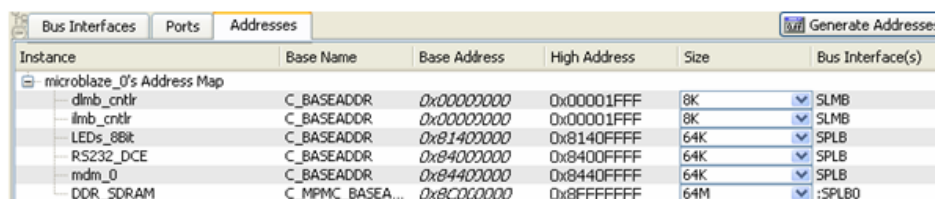
List the nets which are connected to the following ports:

RS232_DCE - RX: _____

RS232_DCE - TX: _____

LEDs_8Bit - GPIO_IO: _____

- 2-1-4.** Click on the **Addresses** tab and have an expanded view similar to **Figure 16**. This is where you can assign base/high addresses to the peripherals in the system.



Instance	Base Name	Base Address	High Address	Size	Bus Interface(s)
microblaze_0's Address Map					
dmb_cntrlr	C_BASEADDR	0x00000000	0x00001FFF	8K	SLMB
ilmb_cntrlr	C_BASEADDR	0x00000000	0x00001FFF	8K	SLMB
LEDs_8Bit	C_BASEADDR	0x81400000	0x8140FFFF	64K	SPLB
RS232_DCE	C_BASEADDR	0x84000000	0x8400FFFF	64K	SPLB
mdm_0	C_BASEADDR	0x84400000	0x8440FFFF	64K	SPLB
DDR_SDRAM	C_MPMC_BASEA...	0x8C000000	0x8FFFFFFF	64M	:SPLB0

Figure 16. Assign Base/High Addresses



Question 3

Select Addresses filter and list the address for the following instances:

RS232_DCE - Base address: _____
RS232_DCE - High address: _____
LEDs_8Bit - Base address: _____
LEDs_8Bit - High address: _____
dlmb_cntlr - Base address: _____
dlmb_cntlr - High address: _____
ilmb_cntlr - Base address: _____
ilmb_cntlr - High address: _____
DDR_SDRAM - Base address: _____
DDR_SDRAM - High address: _____

2-1-5. Run PlatGen by selecting **Hardware Generate Netlist** or click  in the toolbar.

2-1-6. Click on the **Design Summary** tab. The Design Summary allows you to quickly access design overview information, reports, and messages. It displays information specific to your targeted device and software tools. The panes on the left side of the Design Summary allow you to control the information displayed in the right pane.

2-1-7. Browse to the **Lab1** project directory using Windows Explorer.

Several directories containing VHDL wrappers and implementation netlists have been created.



Question 4

List the directories that were created.

Generate Memory TestApp in SDK

Step 3

3-1. Start SDK from XPS, generate software platform project with default settings and default software project name.

3-1-1. Start SDK by clicking **Project → Export Hardware Design to SDK**.

3-1-2. Click on **Export & Launch SDK** button with default settings and browse to **SDK_Export** folder of your project.

3-1-3. In SDK, select **File → New → Xilinx C Project**, select Memory tests from the *Select Project Template* window and then click **Next**.

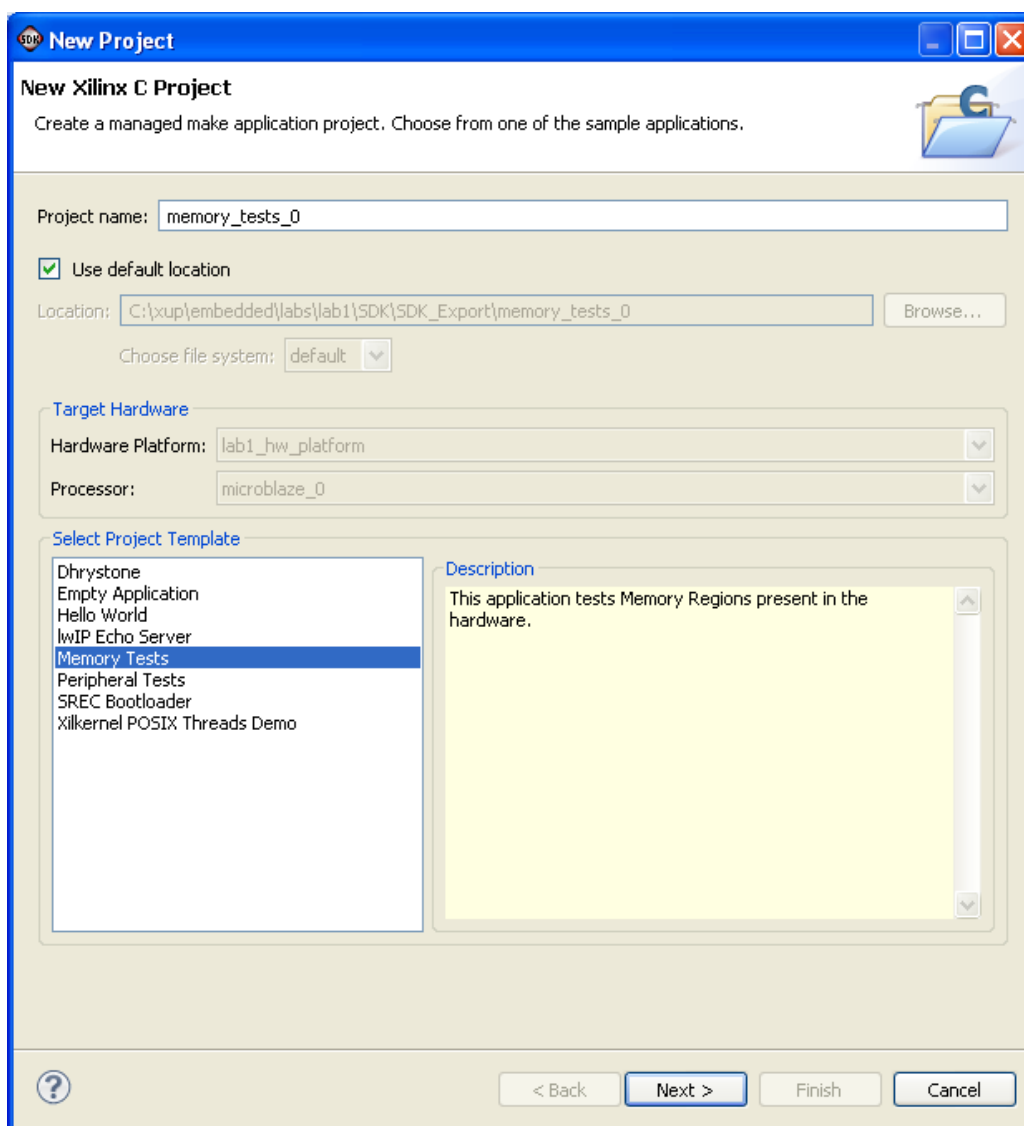


Figure 17. Creating Memory Tests C Project

- 3-1-4.** Click **Finish** with default settings (with `memory_tests_bsp_0` as the default Board Support Package).

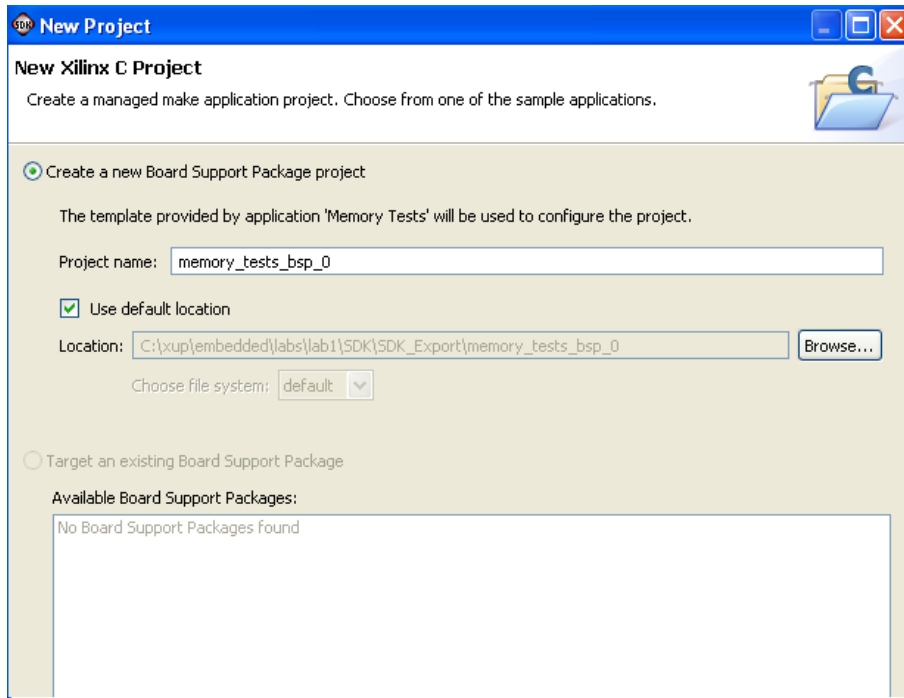


Figure 18. Creating Default Board Support Package

The `memory_tests_0` project will be created in the Project Explorer window of SDK.

Test in Hardware

Step 4

4-1. Generate bitstream and download to the board. Prior to download, the instruction memory (FPGA Block RAM) will be updated in the bitstream with the executable generated using the GNU compiler.

4-1-1. Connect and power up the Spartan-3E starter kit (**Figure 19**).

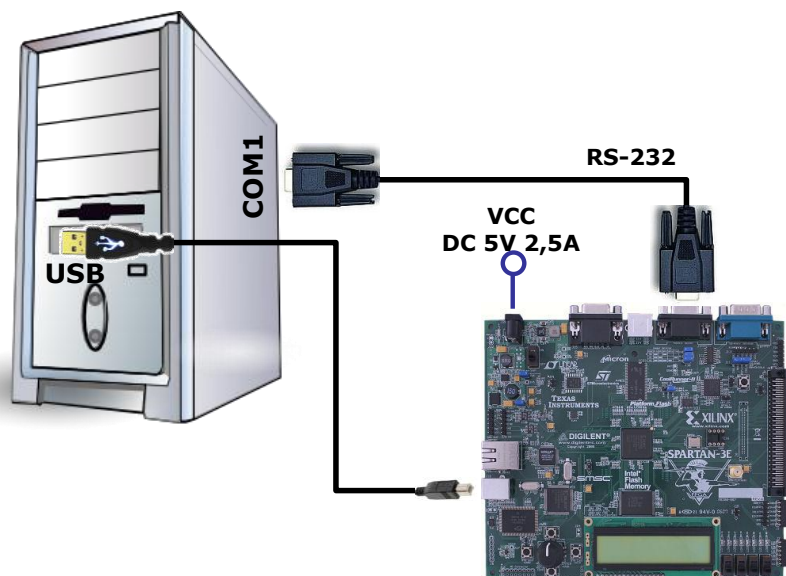


Figure 19. Connection diagram

4-1-2. Open a PuTTY serial session and configure it with the parameters as shown (**Figure 20**).

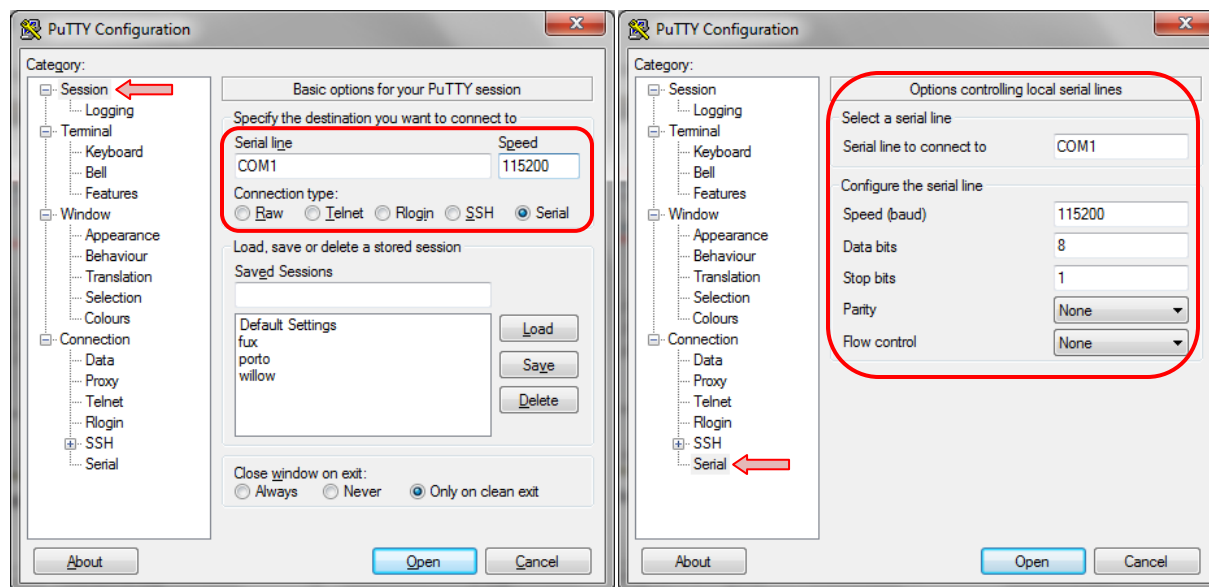


Figure 20. PuTTY Serial Connection Settings

4-1-3. Select **Xilinx Tools** → **Program FPGA** in SDK.

4-1-4. Click on **drop-down** button and select **memory_tests_0.elf** file.

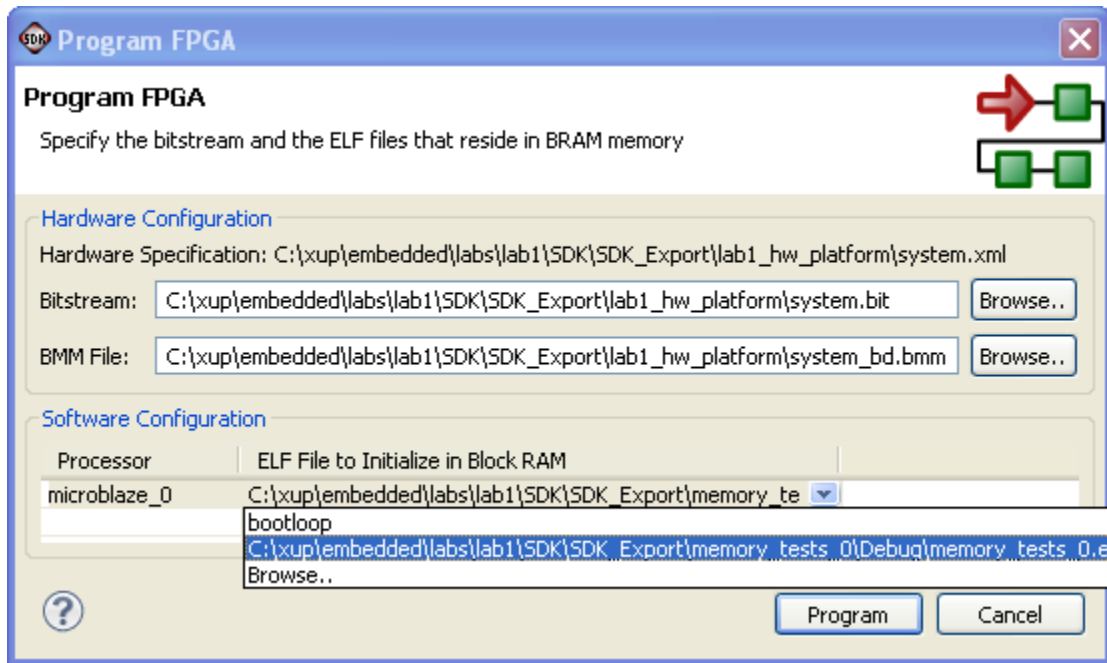


Figure 21. Selecting Application

4-1-5. Click **Program**.

The elf file and the system.bit file will be combined into download.bit file which will then be downloaded to program the FPGA.

You should see the following output on PuTTY

```
--Starting Memory Test Application--
NOTE: This application runs with D-Cache disabled.As a result, cacheline request
s will not be generated
Testing memory region: DDR_SDRAM
Memory Controller: mpmc
Base Address: 0x8c000000
Size: 0x04000000 bytes
32-bit test: PASSED!
16-bit test: PASSED!
8-bit test: PASSED!
--Memory Test Application Complete--
```

Figure 22. PuTTY Terminal Output

Conclusion

The Base System Builder can be used in XPS to quickly generate a MicroBlaze system and software test application. Several files-including an MHS file representing the processor system are created. A System Assembly View, representing the hardware system, provides hardware system parameters information. After the system has been defined, the netlist of the processor system can be created. You can verify hardware operation by downloading a bitstream (configured with test application) to the FPGA.