Use Vivado to build an Embedded System

Objectives

After completing this lab, you will be able to:

- Create a Vivado project for a Zynq system.
- Use the IP Integrator to create a hardware system
- Use SDK(Software Development Kit) to create a standard memory test project
- Run the test application on the board and hence verify hardware functionality

Steps

Create a Vivado Project

1. Open Vivado by selecting **Start > All Programs > Xilinx Design Tools > Vivado 2018.1 > Vivado 2018.1**

2. Click **Create New Project** to start the wizard. You will see the Create a New Vivado Project dialog box. Click Next.

3. Click the Browse button of the Project Location field of the New Project and browse to **{labs}**, and click Select.

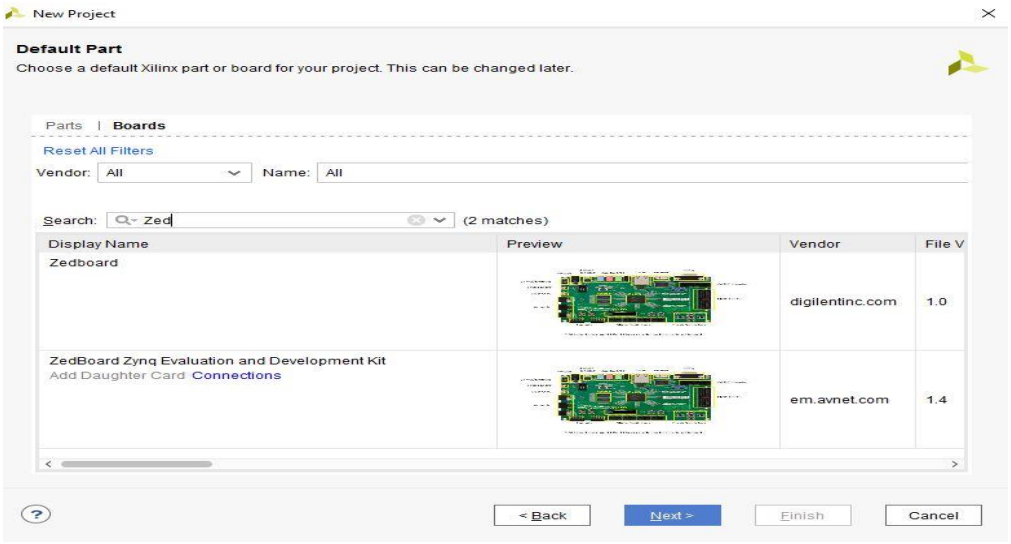
4. Enter **lab1** in the Project Name field. Make sure that the Create Project Subdirectory box is checked. Click Next.

5. In the Project Type form select **RTL Project**, and click Next

6. In the Add Sources form, select Verilog as the Target language and **Mixed** as the Simulator language, and click Next

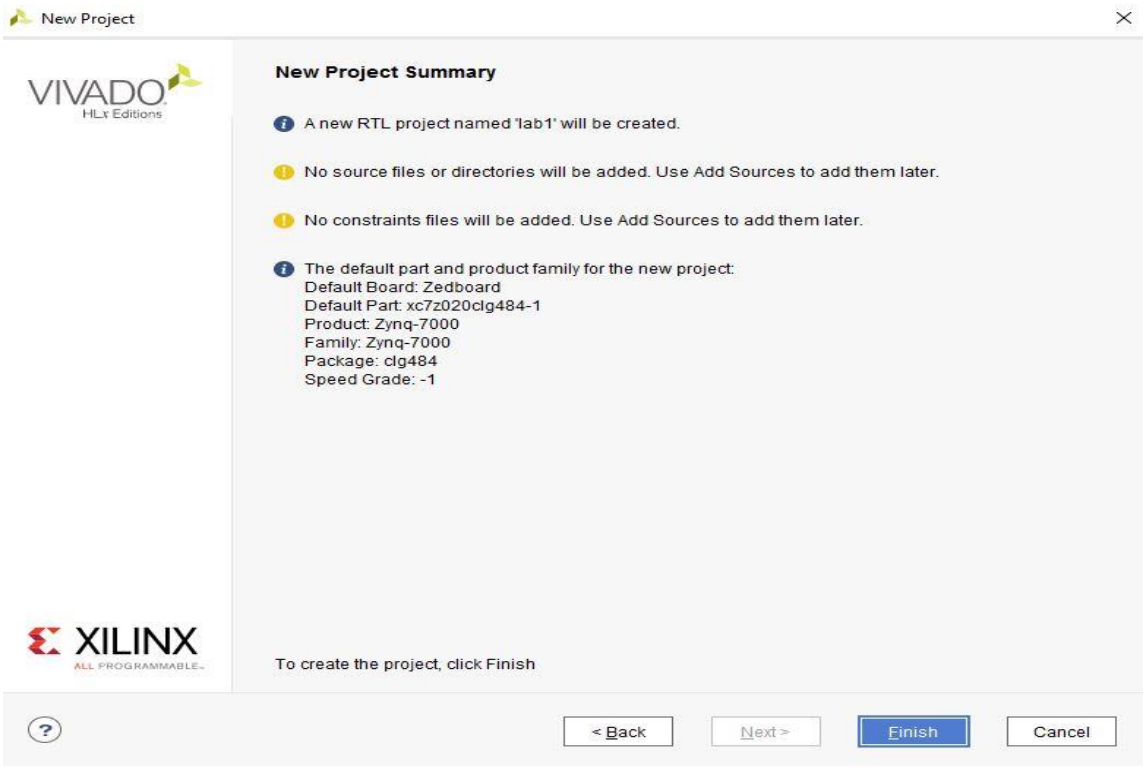
7. Click Next two more times to skip Adding Existing IP and Add Constraints

8. In the Default Part window, select the Boards tab, and depending on the board you are using, (if you can't find the board you are looking for, refer to README.md for setup) and click Next.



Boards and Parts Selection

9. Check the Project Summary (should be similar to what you see below) and click Finish to create an empty Vivado project.



Project Summary

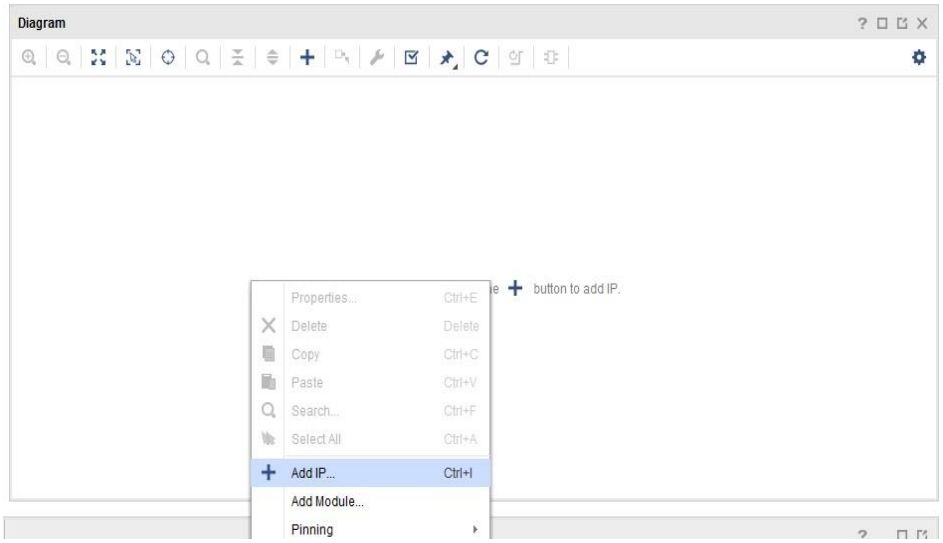
Creating the System Using the IP Integrator

1. In the Flow Navigator, click **Create Block Design** under IP Integrator

Create IP Integrator Block Diagram

2. Enter **system** for the design name and click OK

3. Right-click anywhere in the Diagram workspace and select **Add IP**.



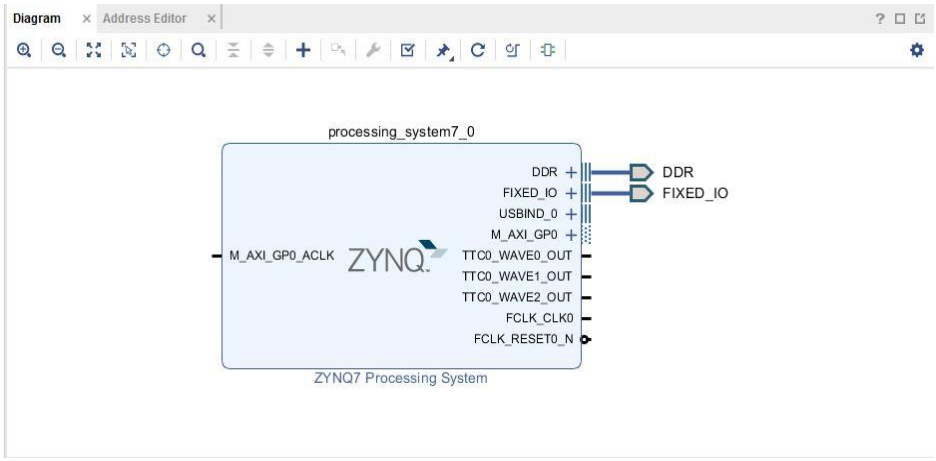
Add IP to Block Diagram

4. Once the **IP Catalog** opens, type "zyn" into the Search bar, find and double click on **ZYNQ7 Processing System** entry, or click on the entry and hit the Enter key to add it to the design.

5. Notice the message at the top of the Diagram window in a green label saying that Designer Assistance available. Click **Run Block Automation**.

6. A new window pops up called the Run Block Automation window. In it, select /processing_system7_0, leave the default settings and click OK

7. Once Block Automation has been complete, notice that ports have been automatically added for the DDR and Fixed IO, and some additional ports are now visible. The imported configuration for the Zynq related to the board has been applied which will now be modified. The block should finally look like this:



Zynq Block with DDR and Fixed IO ports

8. Double-click on the added block to open its **Customization** window. Notice now the Customization window shows selected peripherals (with tick marks). This is the default configuration for the board applied by the block automation.

Configure the processing block with just UART 1 peripheral enabled.

1. A block diagram of the Zynq should now be open again, showing various configurable blocks of the **Processing System**.

2. At this stage, the designer can click on various configurable blocks (highlighted in green) and change the system configuration.

3. Click on one of the peripherals (in green) in the **IOP Peripherals** block of the Zynq Block Design, or select the MIO Configuration tab on the left to open the configuration form

4. Expand **I/O peripherals** if necessary, and ensure all the following I/O peripherals are deselected except UART 1.

   Note : Select UART 0 for PYNQ-Z2 instead of UART 1

   i.e. Remove: *ENET*

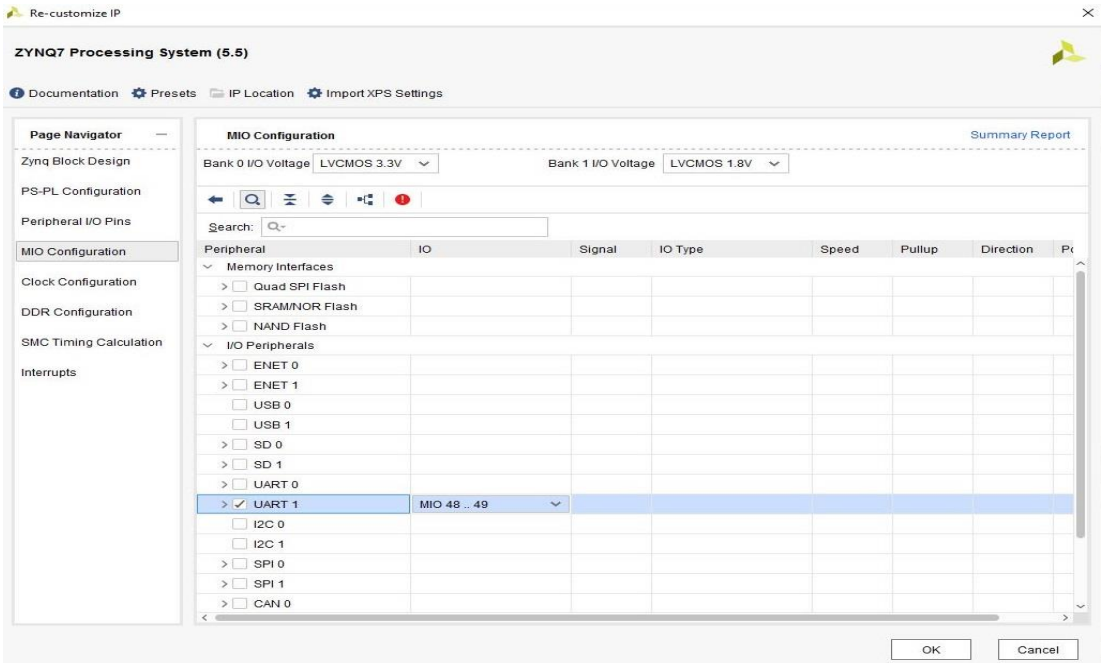
   USB 0

   SD 0

   *Expand **GPIO** to deselect GPIO MIO*

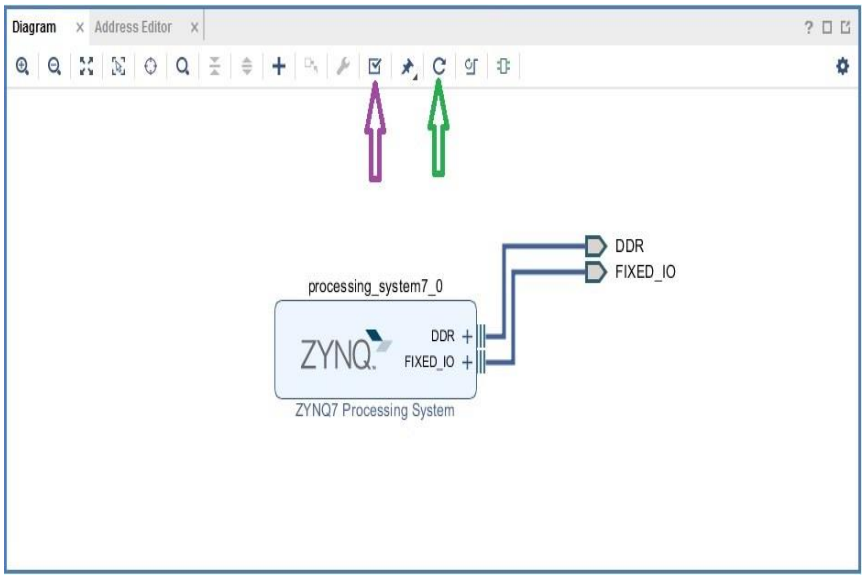
   *Expand **Memory Interfaces** to deselect Quad SPI Flash*

*Expand **Application Processor Unit** to disable Timer 0.*



Selecting only UART 1

5. Select the **PS-PL Configuration** tab on the left.
6. Expand **AXI Non Secure Enablement > GP Master AXI interface** and deselect M AXI GP0 interface.
7. Expand **General > Enable Clock Resets** and deselect the FCLK_RESET0_N option.
8. Select the **Clock Configuration** tab on the left. Expand the PL Fabric Clocks and deselect the FCLK_CLK0 option and click OK.
9. Click on the **Regenerate Layout button** (green arrow) shown below:

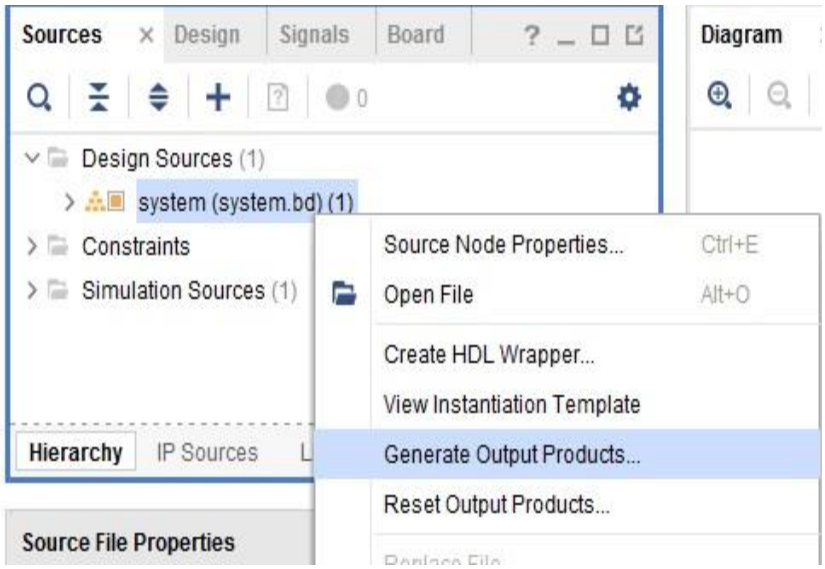


Regenerating and Validating Design

10. Click on the **Validate Design button** (purple arrow) and make sure that there are no errors.

Generate Top-Level and Export to SDK

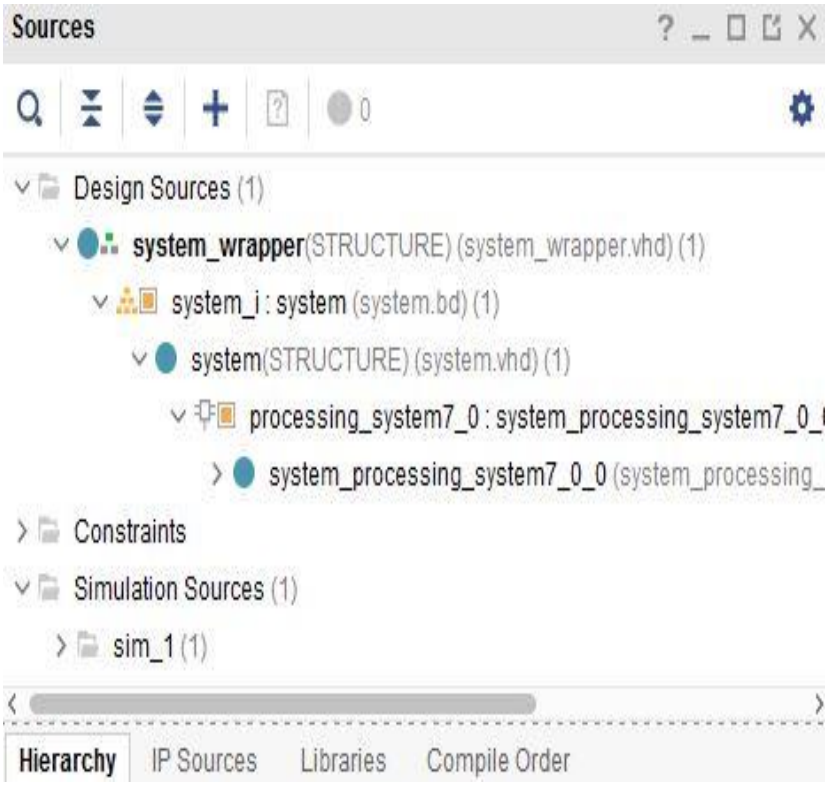
1. In the sources panel, right-click on **system.bd**, and select **Generate Output Products…** and click Generate to generate the Implementation, Simulation and Synthesis files for the design (You can also click on **Generate Block Design** in the Flow Navigator pane to do the same)



Generating output products

2. Right-click again on system.bd, and select Create **HDL Wrapper…** to generate the top-level VHDL model. Leave the Let Vivado manager wrapper and auto-update option selected, and click OK

   The system_wrapper.vhd file will be created and added to the project. Double-click on the file to see the content in the Auxiliary pane.



The HDL Wrapper file generated and added to the project

3. Notice that the VHDL file is already Set As the Top module in the design, indicated by the icon

4. Select **File > Export > Export hardware** and click OK. (Save the project if prompted) Note: Since we do not have any hardware in Programmable Logic (PL) there is no bitstream to generate, hence the Include bitstream option is not necessary at this time.

5. Select **File > Launch SDK** leaving the default settings, and click OK

   SDK should now be open. If only the Welcome panel is visible, close or minimize this panel to view the Project Explorer and Preview panel. A Hardware platform project should have been automatically created, and the system_wrapper_hw_platform_0 folder should exist in the Project Explorer panel.

The system.hdf file **(Hardware Description File)** for the Hardware platform should open in the preview pane. Double click system.hdf to open it if it is not.

Basic information about the hardware configuration of the project can be found in the .hdf file, along with the Address maps for the PS systems, and driver information. The .hdf file is used in the software environment to determine the peripherals available in the system, and their location in the address map.

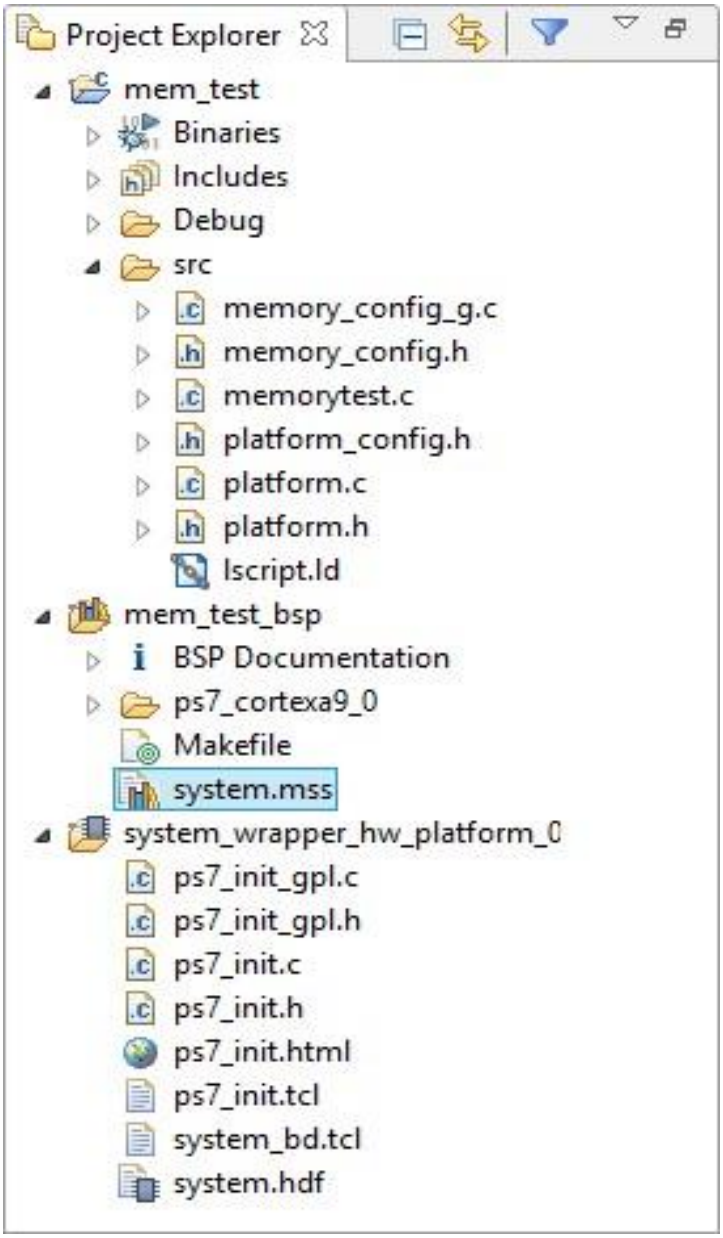
Generate Memory TestApp in SDK

1. Generate memory test application using one of the standard projects template.
2. In SDK, select **File > New > Application Project**
3. Name the project **mem_test**, and in the Board Support Package section, leave Create New selected and leave the default name mem_test_bsp and click Next. Then click on Next

1. Select **Memory Tests** from the Available Templates window, and click Finish.

   The mem_test project and the board support project mem_test_bsp will be created and will be visible in the Project Explorer window of SDK, and the two projects will be automatically built.

2. Expand folders in the Project Explorer view on the left, and observe that there are three projects – system_wrapper_hw_platform_0, mem_test_bsp, and mem_test. The **mem_test** project is the application that we will use to verify the functionality of the design. The **hw_platform** includes the ps7_init function which initializes the PS as part of the first stage bootloader, and **mem_test_bsp** is the board support package. The Explorer view should look something like this:

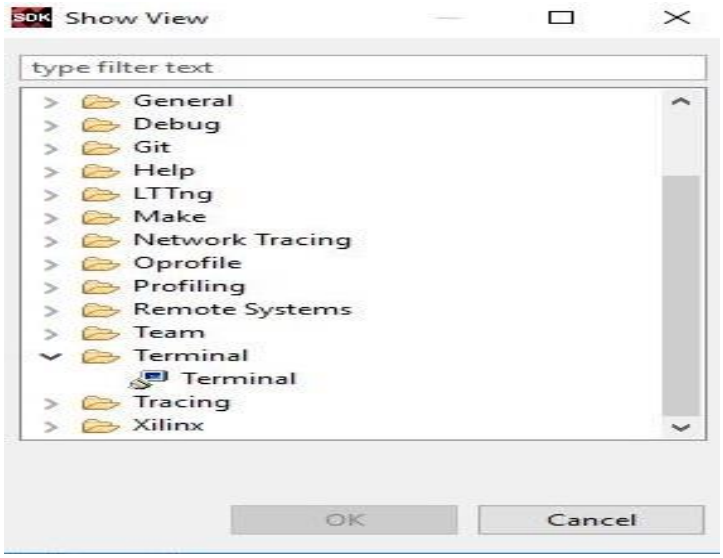


The Project Explorer view

3. Open the memorytest.c file in the mem_test project (under src), and examine the contents. This file calls the functions to test the memory.

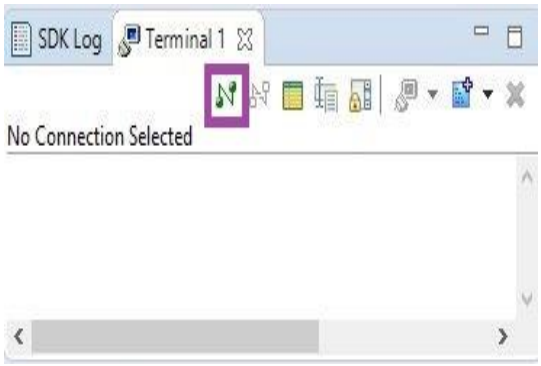
Test in Hardware

1. Setup the hardware as shown in README.md
2. Select the tab. If it is not visible then select Window > Show view > Other...
3. Select Terminal>Terminal and click OK

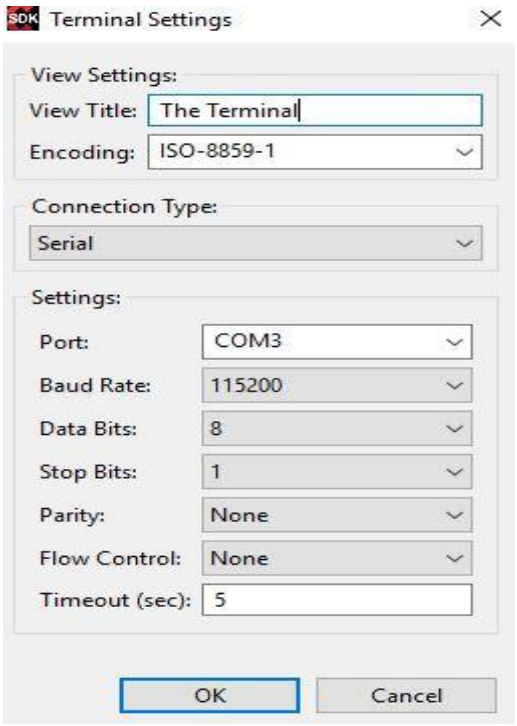


Finding Terminal window

4. Click on the **Connect** button (shown below in a violet box) and if required, select appropriate COM port (depends on your computer), and configure it with the parameters as shown in the next figure.



Terminal window



SDK Terminal Settings

You can find the COM port from the Windows Device Manager

5. Run the mem_test application and verify the functionality.

6. In SDK, select the mem_test project in Project Explorer, right-click and select **Run As > Launch on Hardware (System Debugger)** to download the application, and will execute ps7_init, and then execute mem_test.elf (user application).

7. You should see the following output on the Terminal tab.

```
NOTE: This application runs with D-Cache disabled.As a result, cacheline request
s will not be generated
Testing memory region: ps7_ddr_0
    Memory Controller: ps7_ddr
        Base Address: 0x00100000
                Size: 0x1ff00000 bytes
         32-bit test: PASSED!
         16-bit test: PASSED!
          8-bit test: PASSED!
Testing memory region: ps7_ram_1
    Memory Controller: ps7_ram
        Base Address: 0xffff0000
                Size: 0x0000fe00 bytes
         32-bit test: PASSED!
         16-bit test: PASSED!
          8-bit test: PASSED!
--Memory Test Application Complete--
```

SDK Terminal Output

1. Close SDK and Vivado by selecting **File > Exit** in each program.

Conclusion

Vivado and the IP Integrator allow base embedded processor systems and applications to be generated very quickly. After the system has been defined, the hardware can be exported and SDK can be invoked from Vivado.

Software development is done in SDK which provides several application templates including memory tests. You verified the operation of the hardware by using a test application, executing on the processor, and observing the output in the serial terminal window.