Χρήση Vivado για την κατασκευή ενός ενσωματωμένου συστήματος

Εισαγωγή

Το παρών εργαστήριο σας καθοδηγεί στη διαδικασία χρήσης του Vivado ώστε να δημιουργήσετε ένα απλό design βάση του ARM Cortex-A9 επεξεργαστή με στόχο την πλακέτα ZedBoard ή Zybo. Σε περίπτωση που οι οδηγίες αναφέρονται και στα δύο boards, επιλέξτε το board εκείνο που χρησιμοποιείτε. Θα κάνετε χρήση του Vivado ώστε να δημιουργήσετε το hardware σύστημα και το SDK (Software Development Kit) για να δημιουργήσετε μια εφαρμογή ως παράδειγμα ώστε να επαληθεύσετε τη λειτουργικότητα του hardware συστήματος.

Στόχοι

Μετά την ολοκλήρωση αυτού του εργαστηρίου, θα είστε σε θέση να:

- Δημιουργήσετε ένα project στο Vivado για ένα σύστημα Zynq.
- Χρησιμοποιήσετε τον IP Integrator για τη δημιουργία ενός συστήματος hardware
- Χρησιμοποιήσετε το SDK για να δημιουργήσετε ένα τυπικό project για memory test
- Εκτελέσετε την δοκιμαστική εφαρμογή στο board

Διαδικασία

Αυτό το εργαστήριο χωρίζεται σε βήματα που αποτελούνται από τη γενική επισκόπηση καταστάσεων τα οποία παρέχουν πληροφορίες σχετικά με τις λεπτομερείς οδηγίες που ακολουθούν. Ακολουθήστε αυτές τις λεπτομερείς οδηγίες για να διεκπεραιώσετε το εργαστήριο.

Το παρών εργαστήριο περιλαμβάνει 5 πρωταρχικά βήματα: Θα δημιουργήσετε ένα top-level project χρησιμοποιώντας το Vivado, δημιουργήσετε το σύστημα επεξεργαστή χρησιμοποιώντας τον IP Integrator του Vivado, δημιουργήστε την HDL υψηλού επιπέδου και εξάγετε το design στο SDK, έπειτα δημιουργήστε μια εφαρμογή “Memory Test” στο SDK, και, τέλος, δοκιμάστε το σε hardware.

Περιγραφή Design

Σκοπός των εργαστηριακών ασκήσεων είναι να κατανοήσετε (;) το σχεδιασμό ενός ολοκληρωμένου συστήματος υλικού και λογισμικού του επεξεργαστή. Κάθε εργαστηριακή άσκηση θα «χτίζεται» πάνω στα προηγούμενα εργαστήρια. Το παρακάτω διάγραμμα αναπαριστά το ολοκληρωμένο design (**Figure 1**).

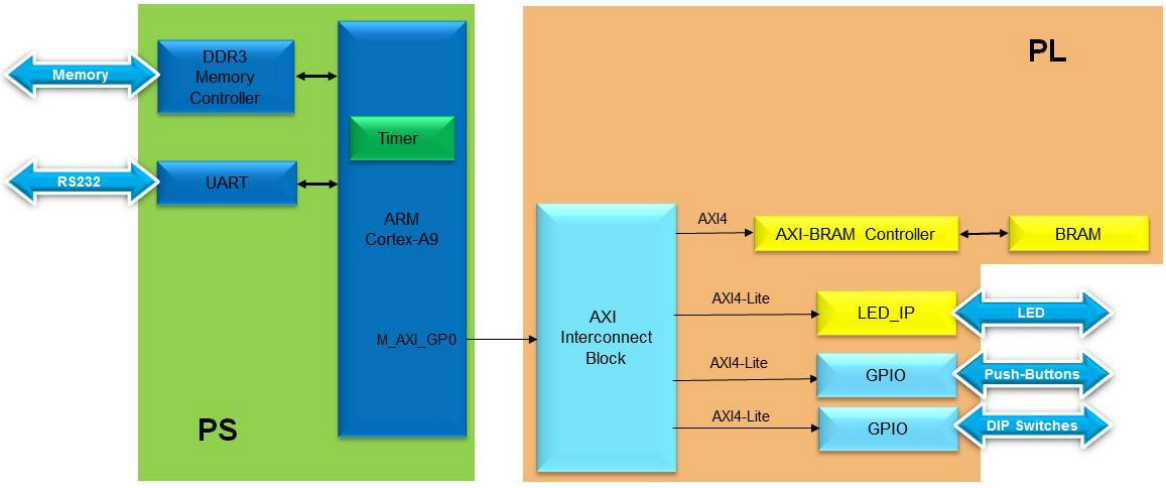


Figure 1. Completed Design

Σε αυτό το εργαστήριο, θα κάνετε χρήση του IP Integrator για να δημιουργήσετε ένα design με βάση το processing system, αποτελούμενο από τα παρακάτω (**Figure 2**):

- ARM Cortex A9 core (PS)
- UART for serial communication
- DDR3 controller for external DDR3_SDRAM memory

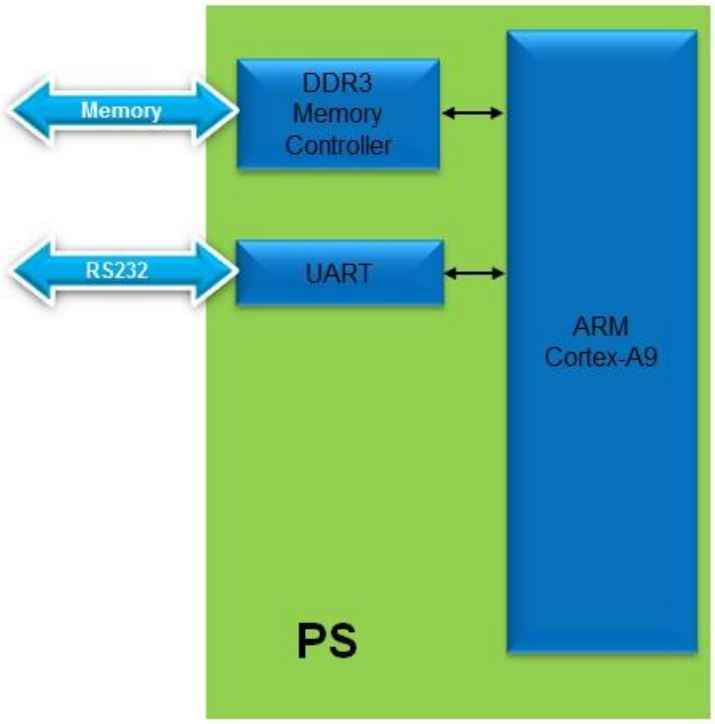
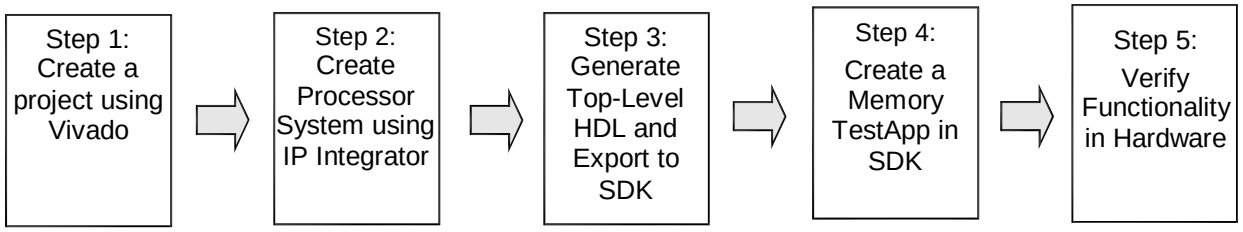


Figure 2. Processor Design of this Lab

General Flow for this Lab



In the instructions below;
{***sources***} refers to: C:\xup\embedded\2015_2_zynq_sources
{***labs***} refers to : C:\xup\embedded\2015_2_zynq_labs

Board support for the Zybo is not included in Vivado 2015.2 by default. The relevant files “zybo.zip” need to be extracted and saved to: {Vivado installation}\data\boards\board_files\zynq

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

1Create a Vivado Project Step 1

1-1. Launch Vivado and create an empty project targeting the ZedBoard or the Zybo, using the VHDL language.

1-1-1. Open Vivado by selecting **Start > All Programs > Xilinx Design Tools > Vivado 2015.2 > Vivado 2015.2**

1-1-2. Click **Create New Project** to start the wizard. You will see the *Create a New Vivado Project* dialog box. Click **Next**.

1-1-3. Click the Browse button of the *Project Location* field of the **New Project** form, browse to **{labs}**, and click **Select**.

1-1-4. Enter **lab1** in the *Project Name* field. Make sure that the *Create Project Subdirectory* box is checked. Click **Next**.

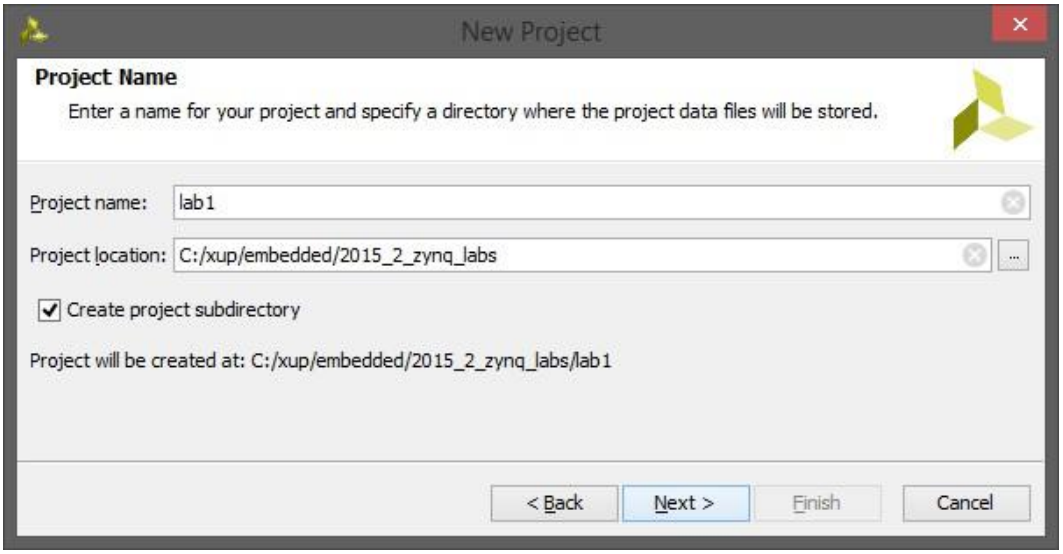


Figure 3. Project Name Entry

1-1-5. In the *Project Type* form select **RTL Project**, and click **Next**

1-1-6. In the *Add Sources* form, select **VHDL** as the *Target language* and **Mixed** as the *Simulator language*, and click **Next**

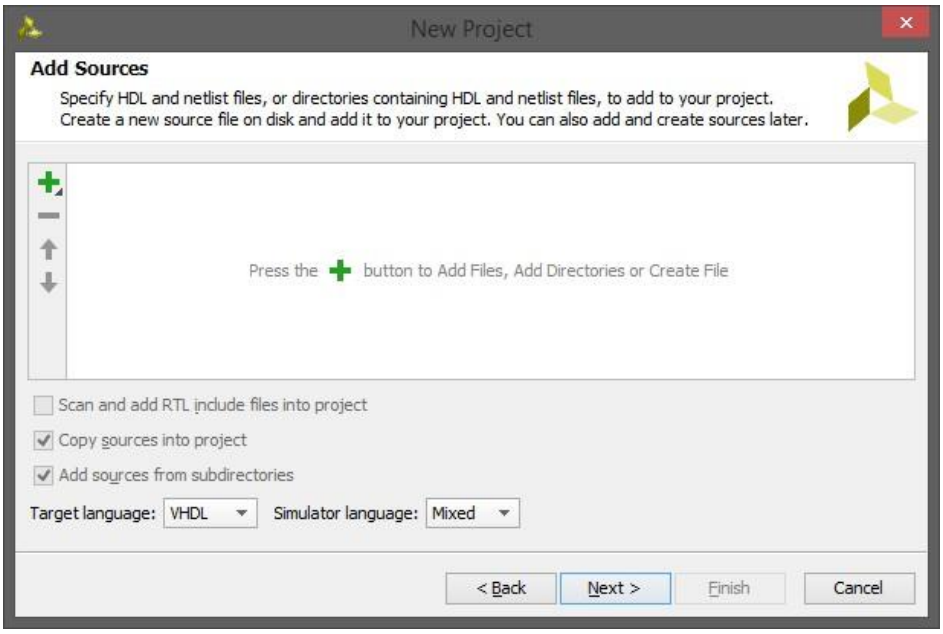


Figure 4. Add sources to new project

1-1-7. Click **Next** two more times to skip *Adding Existing IP* and *Add Constraints*

1-1-8. In the *Default Part* form, select *Boards*, and depending on the board you are using, select *ZedBoard or Zybo* and click **Next**.

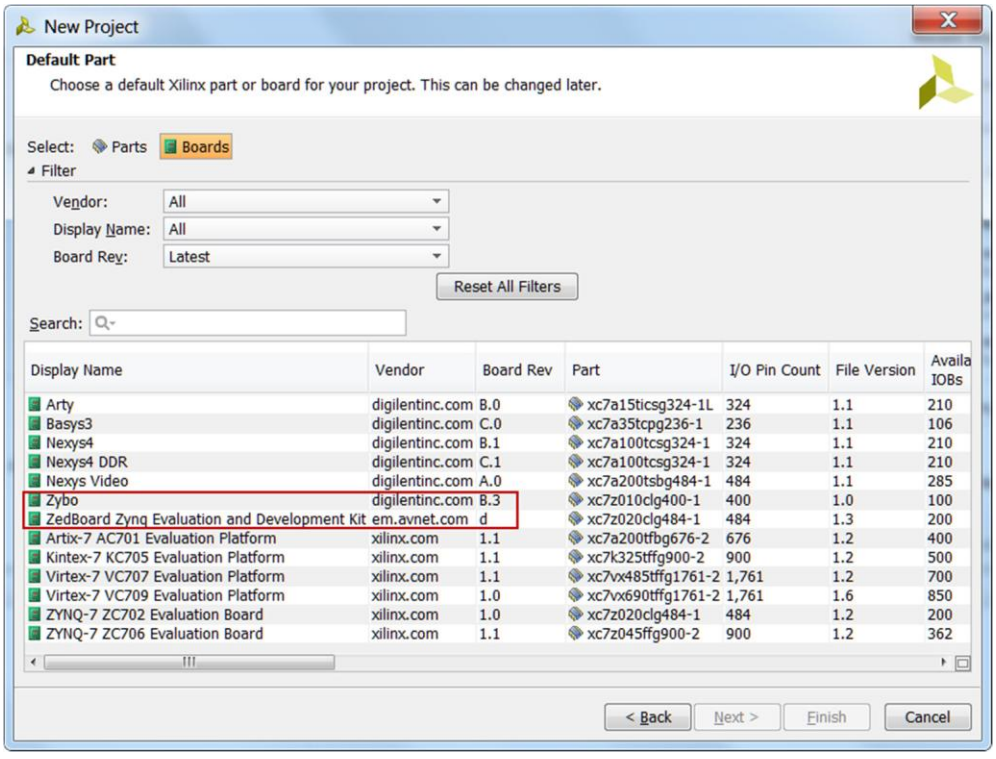


Figure 5. Boards and Parts Selection

1-1-9. Check the *Project Summary* and click **Finish** to create an empty Vivado project.

2Creating the System Using the IP Integrator Step 2

2-1. Use the IP Integrator to create a new Block Design, add the ZYNQ processing system block, and import the provided xml file for the board.

2-1-1. In the Flow Navigator, click **Create Block Design** under IP Integrator

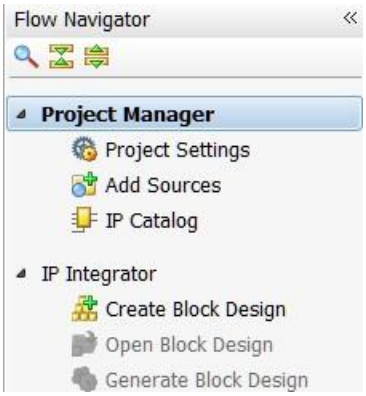


Figure 6. Create IP Integrator Block Diagram

2-1-2. Enter **system** for the design name and click **OK**

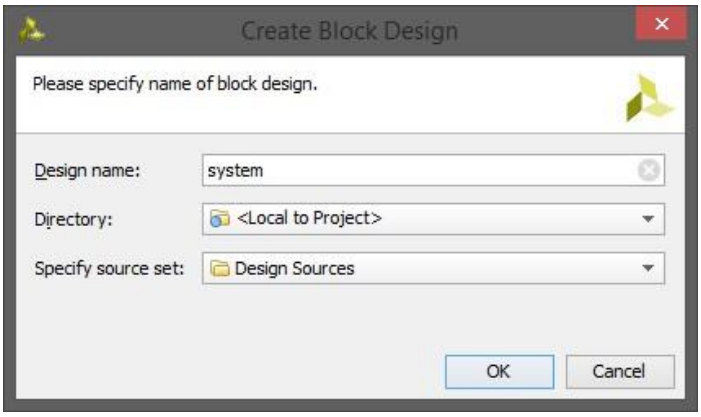


Figure 7. Create New Block Diagram

2-1-3. IP from the catalog can be added in different ways. Click the *Add IP icon*  in the block diagram side bar, press Ctrl + I, click the *Add IP icon* in the empty Diagram workspace, or right-click anywhere in the Diagram workspace and select Add IP.

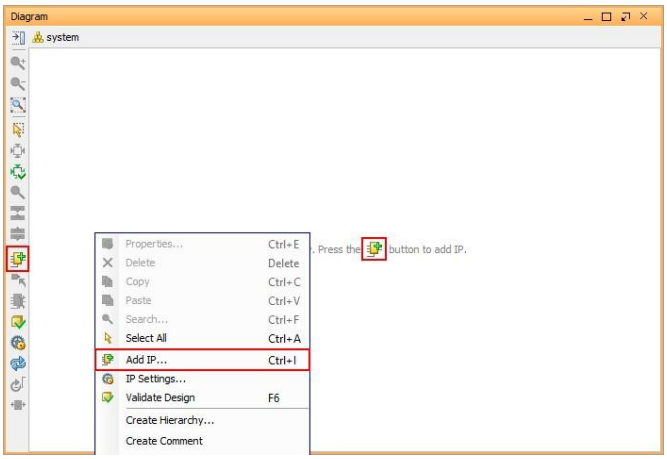


Figure 8. Add IP to Block Diagram

2-1-4. Once the IP Catalog is open, type “z” 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.

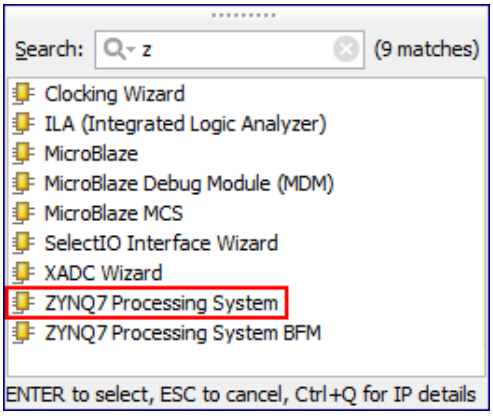


Figure 9. Add Zynq block to the design

2-1-5. Notice the message at the top of the Diagram window that *Designer Assistance* available. Click *Run Block Automation* and select **/processing_system7_0**

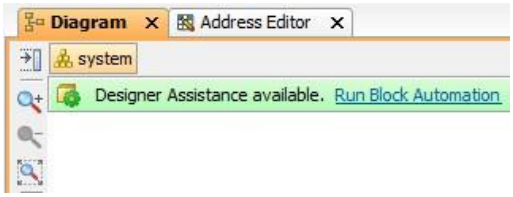


Figure 10. Run block automation

2-1-6. In the *Run Block Automation* window, leave the default settings, including *Apply Board Preset* checked, and click **OK**

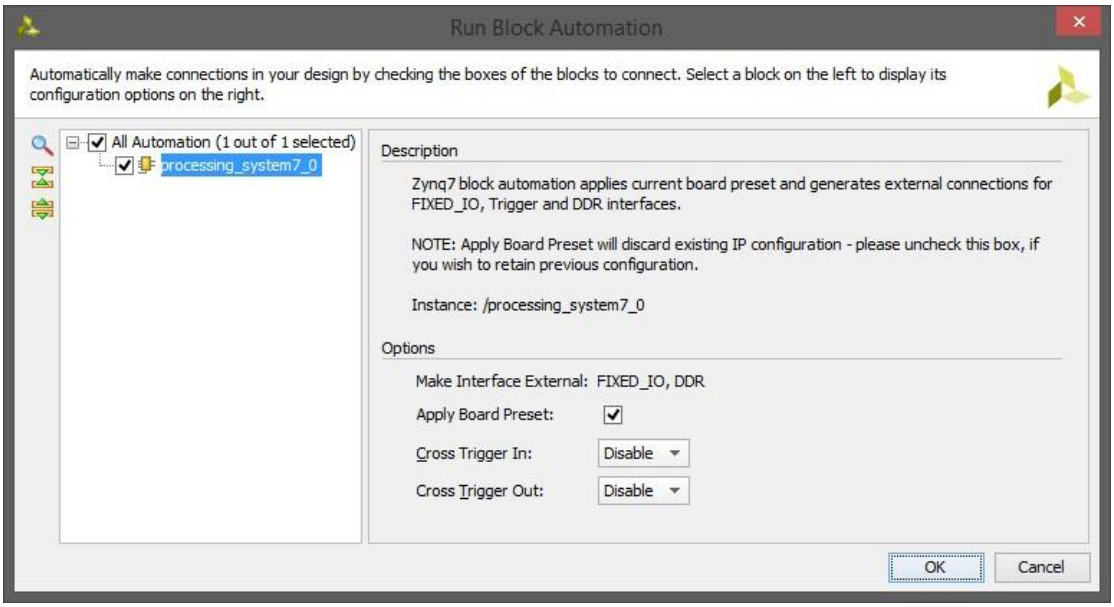


Figure 11. Block Automation settings

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 Zybo board has been applied which will now be modified.

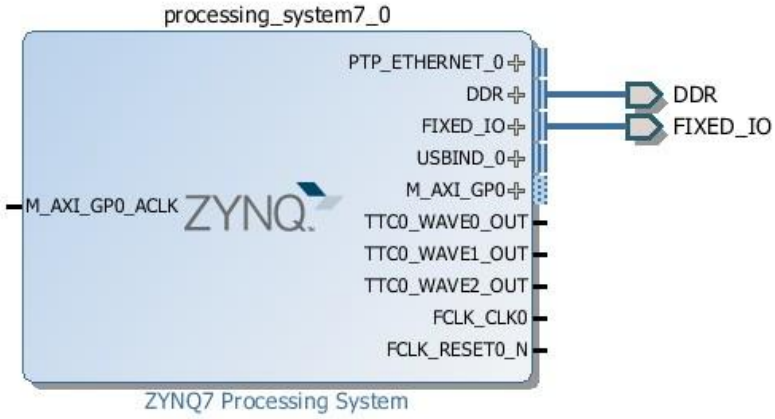


Figure 12. Zynq Block with DDR and Fixed IO ports

2-1-7. 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.

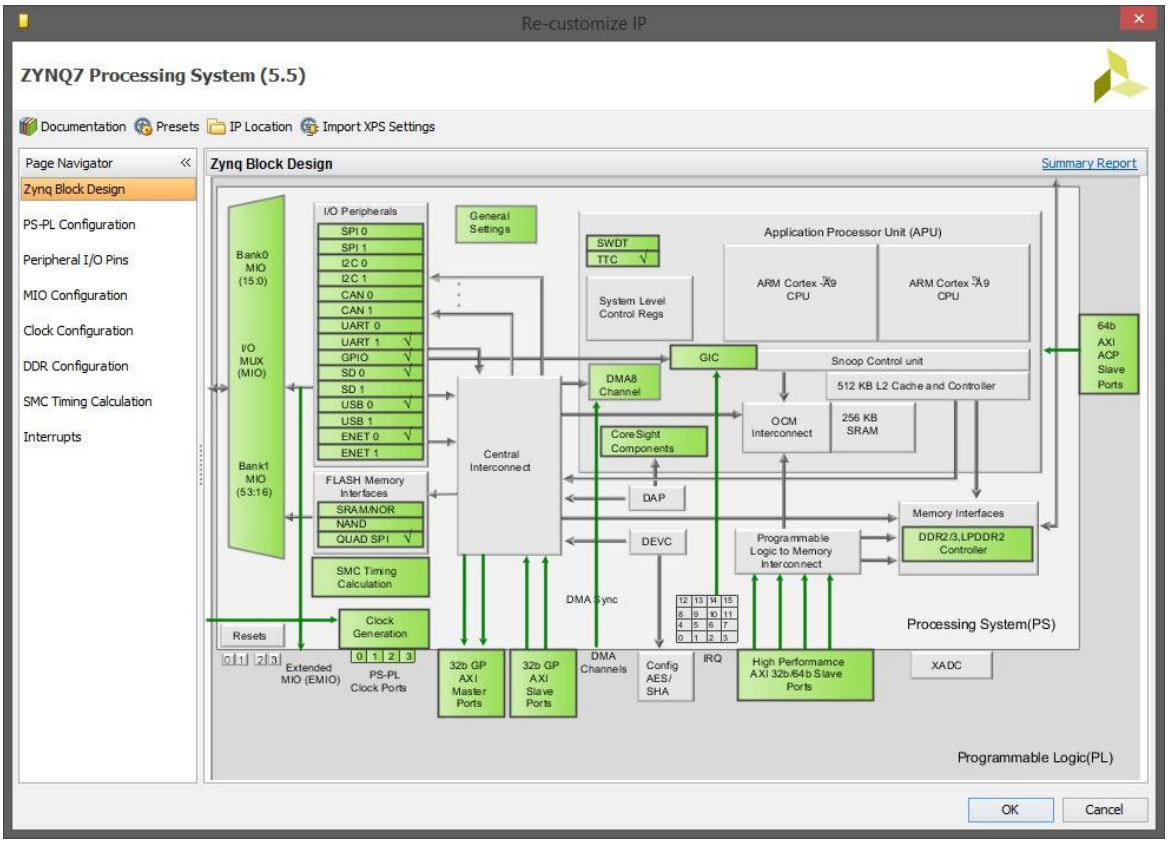


Figure 13. Imported peripherals settings

2-2. Configure the processing block with just UART 1 peripheral enabled.

2-2-1. A block diagram of the Zynq should now be open again, showing various configurable blocks of the Processing System.

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

Only the UART is required for this lab, so all other peripherals will be deselected.

2-2-2. Click on one of the peripherals (in green) in the *IOP Peripherals* block, or select the *MIO Configuration* tab on the left to open the configuration form

2-2-3. Expand I/O peripherals if necessary, and ensure all the following *I/O peripherals are deselected* except *UART 1.*

i.e. Remove: *ENET 0*
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*.

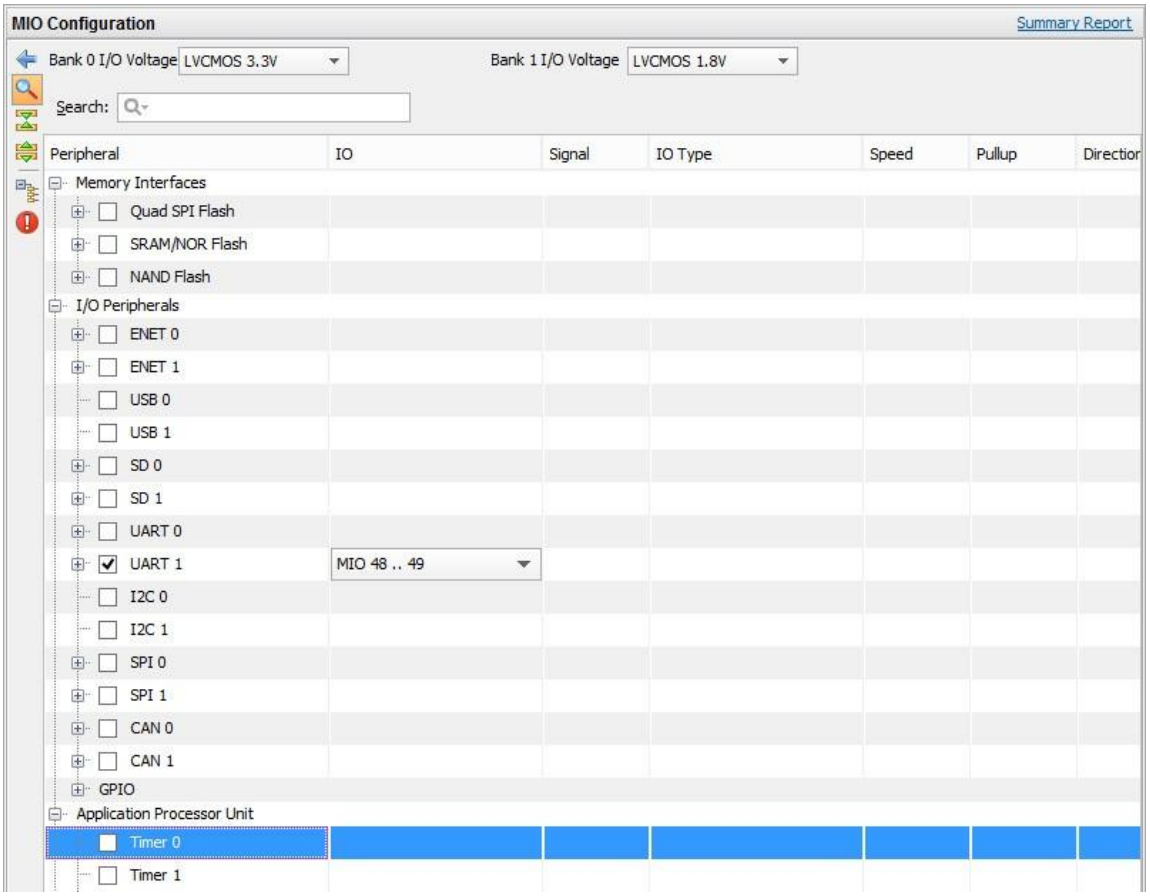


Figure 14. Selecting only UART 1

2-2-4. Select the **PS-PL Configuration** tab on the left.

2-2-5. Expand *AXI Non Secure Enablement* > *GP Master AXI interface* and deselect **M AXI GP0** interface.

2-2-6. Expand **General > Enable Clock Resets** and deselect the **FCLK_RESET0_N** option.

2-2-7. Select the **Clock Configuration** tab on the left. Expand the **PL Fabric Clocks** and deselect the **FCLK_CLK0** option and click **OK.**

Click on the (Regenerate Layout) button and see the following block diagram.

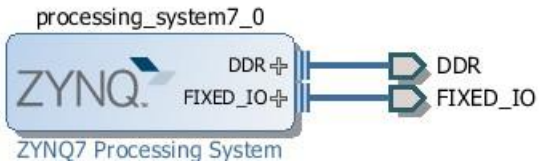


Figure 15. Updated Zynq Block

2-2-8. Click on the (Validate Design) button and make sure that there are no errors.

3Generate Top-Level and Export to SDK Step 3

3-1. Generate IP Integrator Outputs, the top-level HDL, and start SDK by exporting the hardware.

3-1-1. 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)

3-1-2. 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.

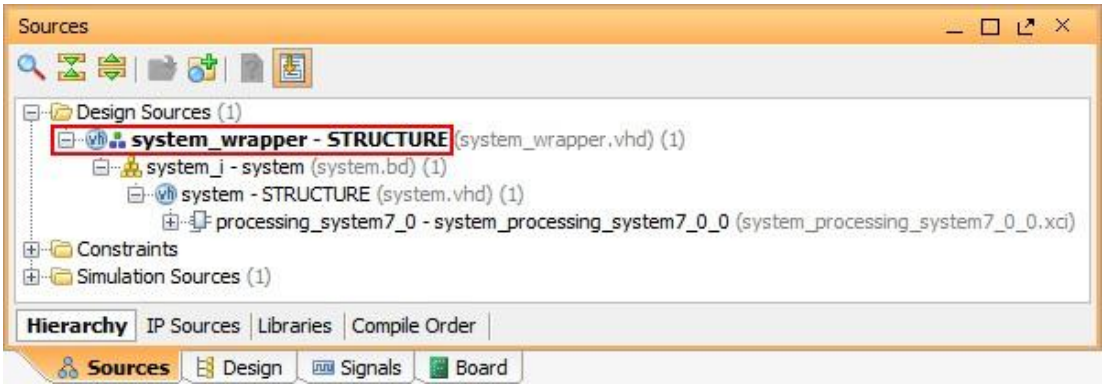


Figure 16. The HDL Wrapper file generated and added to the project

3-1-3. Notice that the VHDL file is already *Set As the Top* module in the design, indicated by the icon

3-1-4. 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.

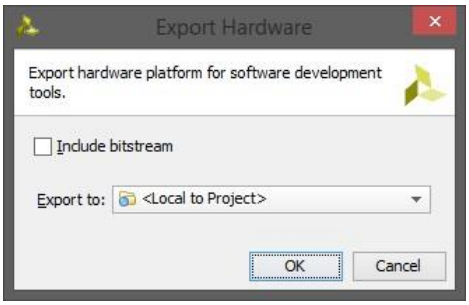


Figure 17. Exporting hardware

3-1-5. Select **File > Launch SDK** leaving the default settings, and click **OK**

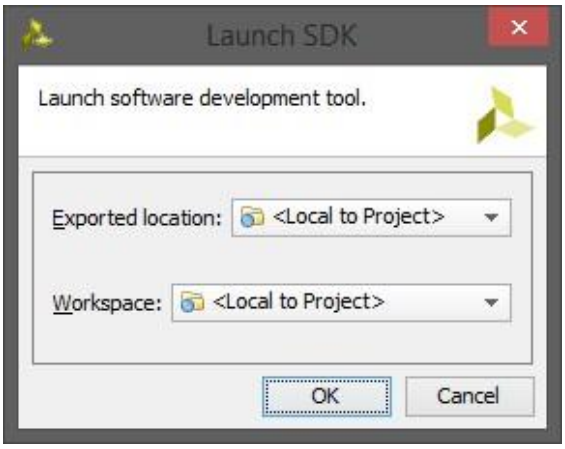


Figure 18. Launch SDK

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.

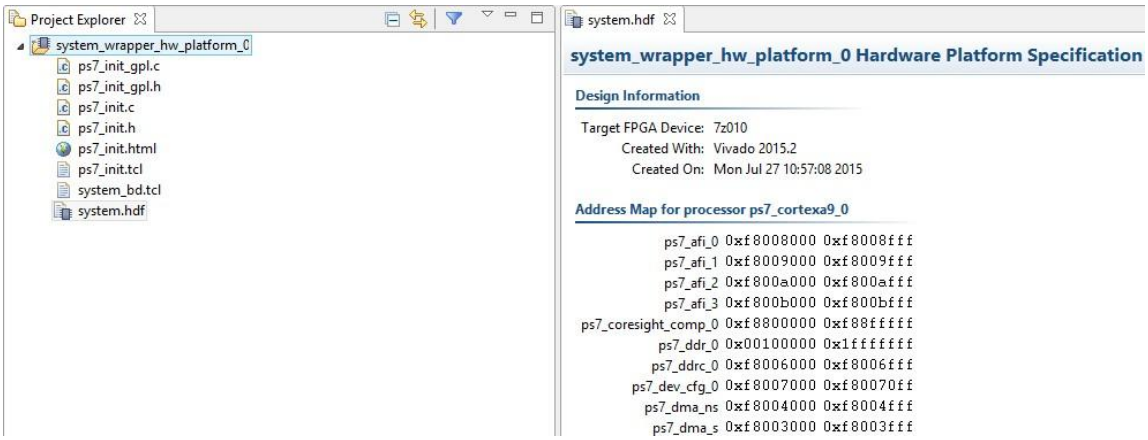


Figure 19. SDK C/C++ development view

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.

4-1. Generate memory test application using one of the standard projects template.

4-1-1. In SDK, select **File** > **New** > **Application Project**

4-1-2. 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.** (Note that this application will run on ps7_cortexa9_0 i.e. core 0 of the two processor cores available.)

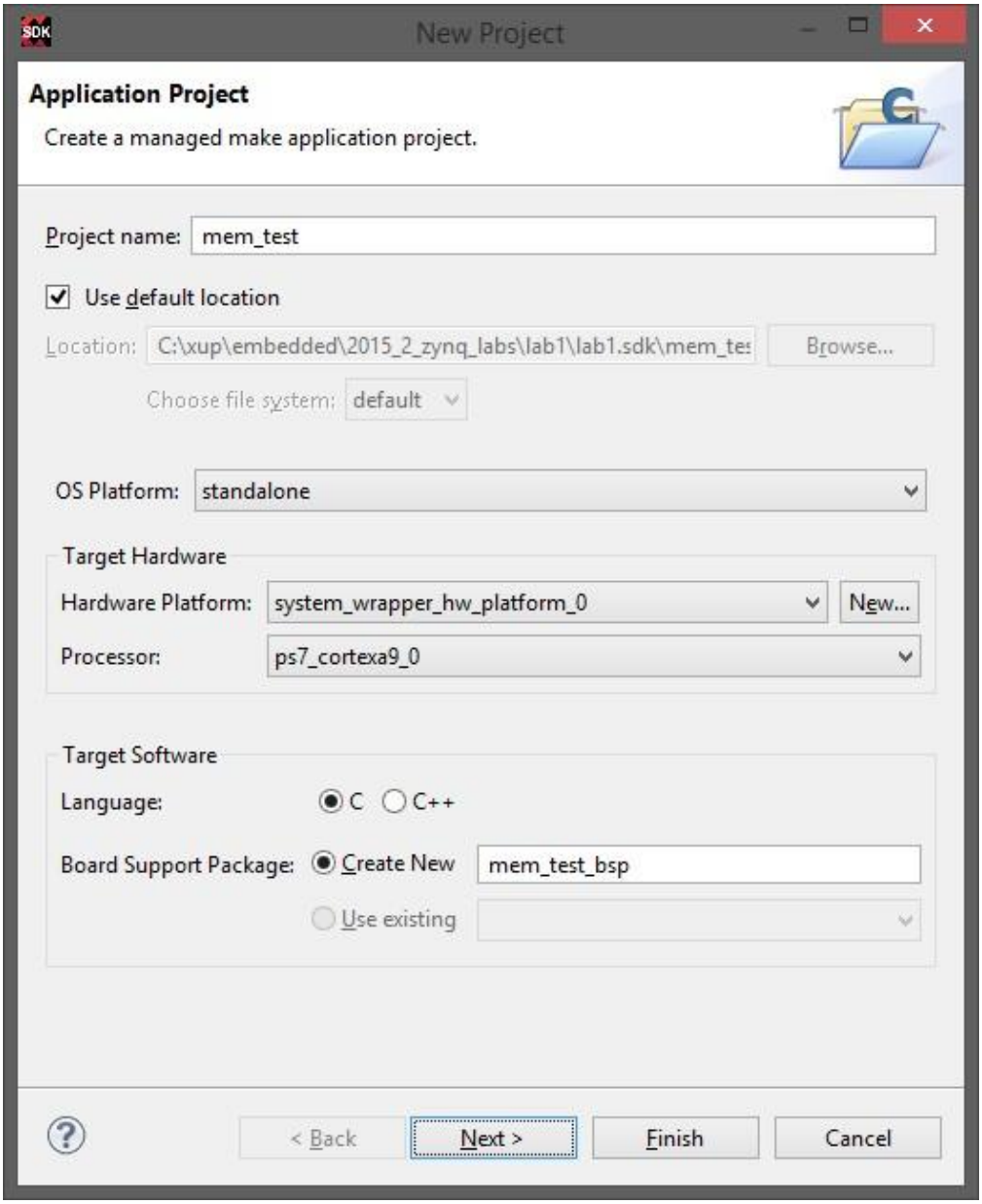


Figure 20. Create new SDK application project

4-1-3. Select **Memory Tests** from the *Available Templates* window, and click **Finish.**

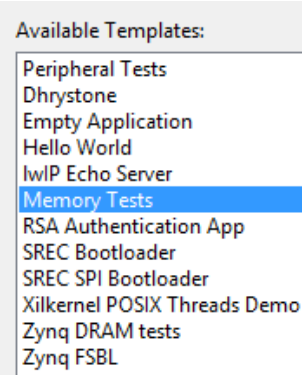


Figure 21. Creating Memory Tests C Project

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. You can monitor the progress in the Console panel.

4-1-4. Expand folders in the Project Explorer view, 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.

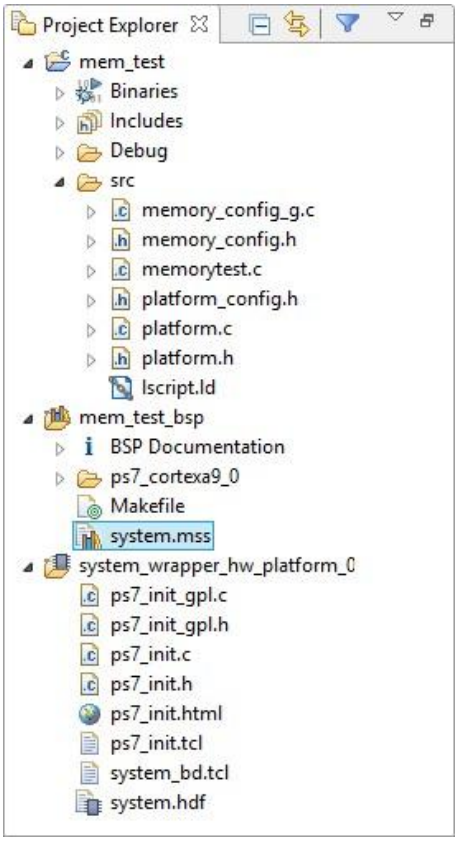


Figure 22. The Project Explore view

4-1-5. 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.

5-1. Zybo: Make sure that the JP7 is set to select USB power and JP5 is set to JTAG mode. Connect the board with a micro-usb cable and power it ON.

ZedBoard: Make sure that two micro-usb cables are used between the PC and the PROG and the UART connectors of the board and that the board is placed in the JTAG mode (MIO6-MIO2 jumpers are in the Dn position).

Establish the serial communication using SDK's Terminal tab.

5-1-1. Zybo: Make sure that the JP7 is set to select USB power, and JP5 is set to JTAG. Make sure that a micro-USB cable is connected to the JTAG PROG connector (next to the power supply connector). Turn ON the power.

ZedBoard: Make sure that two micro-usb cables are used between the PC and the PROG and the UART connectors of the board and that the board is placed in the JTAG mode (MIO6-MIO2 jumpers are in the Dn position). Turn ON the power

5-1-2. Select the Terminal tab. If it is not visible then select **Window > Show view > Terminal**.

5-1-3. Click on and if required, select appropriate COM port (depends on your computer), and configure it with the parameters as shown.

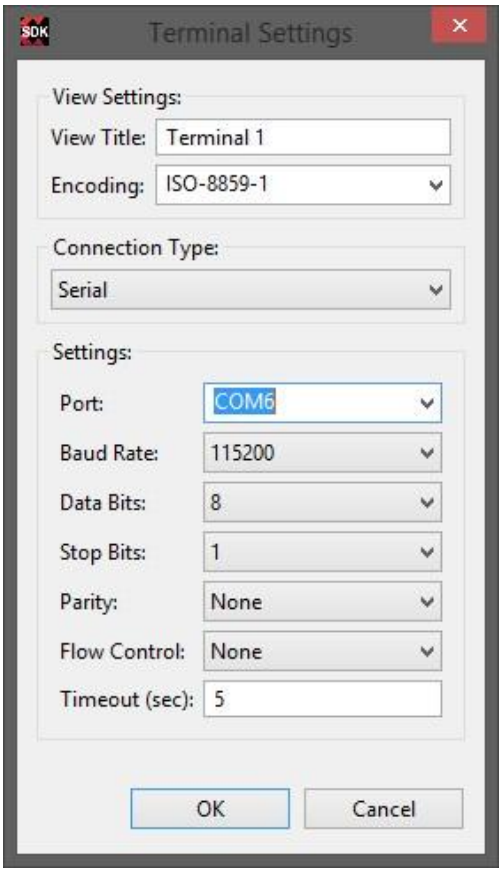


Figure 23. SDK Terminal Settings

You can find the COM port from the Windows Device Manager, in this case, COM6:

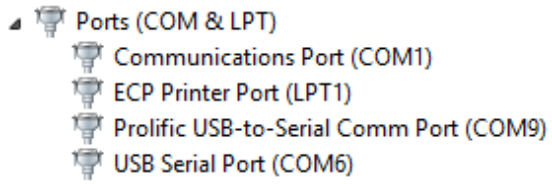


Figure 24. COM ports in Windows control panel

5-2. Run the mem_test application and verify the functionality.

5-2-1. In SDK, select the **mem_test** project in *Project Explorer*, right-click and select **Run As > Launch on Hardware (GDB)** to download the application, and will execute ps7_init, and then execute mem_test.elf (user application).

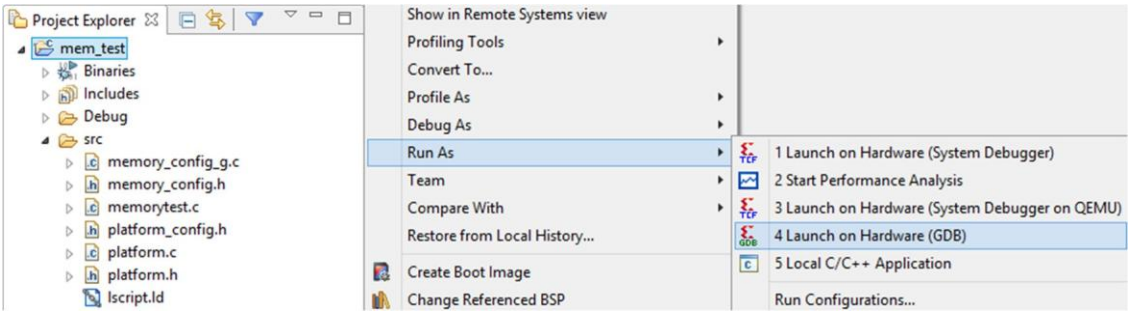


Figure 25. Launch Application

5-2-2. 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--
```

Figure 26. SDK Terminal Output

5-2-3. 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 downloading a test application, executing on the processor, and observing the output in the serial terminal window.