

Advanced Workshop on Modern FPGA- Based Technology for Scientific Computing



LABORATORY

The Communication Block (ComBlock)

Prepared by

Rodrigo A. Melo, Kasun Mannatunga

ICTP-MLAB

Adding ComBlock IP to the System

Introduction

This lab guides you through the process of adding and connecting the ComBlock IP block to a Processor System (PS) by using Vivado and test it using ramp generators.

Objectives

After completing this lab, you will be able to:

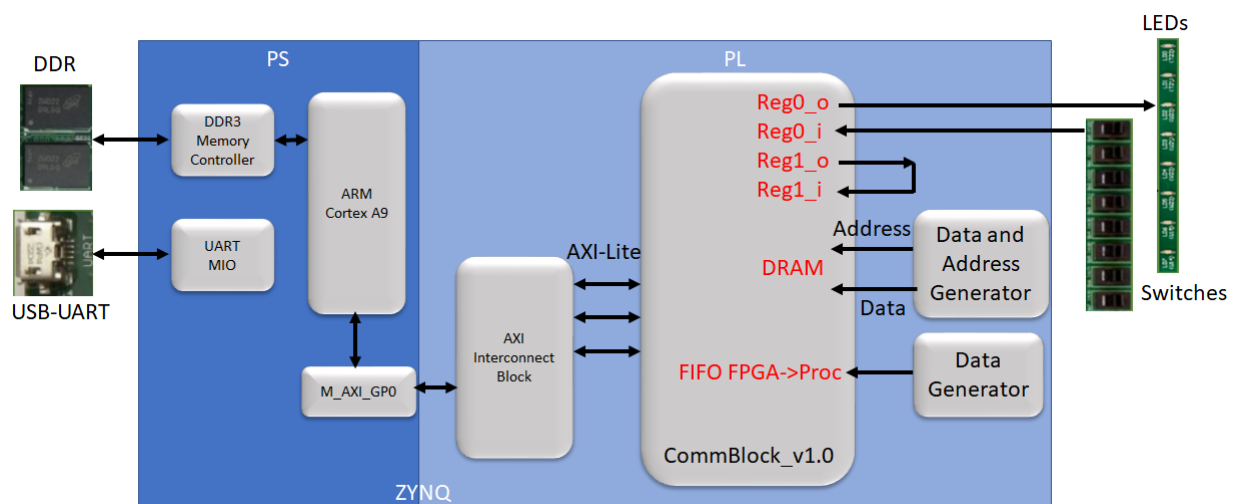
- Instantiate and configure the ComBlock IP in Vivado
- Transfer data between PL and PS through the ComBlock IP.
- Write a baremetal/standalone application to interact with the ComBlock.

Procedure

This lab comprises five primary steps: You will create a project using Vivado with a PS, add a ComBlock IP instance and configure it, create test patterns for the comblock, export the design to the SDK and create an application in the SDK and test the design in hardware (ZedBoard).

Design Description

In this lab, you will design a complete embedded system consisting of the ARM Cortex-A9 processor SoC, comblock IP connected to the eight on-board LEDs, to the eight on board switches and to two data generators. The following block diagram represents the design to be done.



Create a Vivado Project using IDE

Step 1

Objective: launch Vivado and create a project with an embedded processor system targeting the ZedBoard.


1.1. Open Vivado Design Suite.

1.2. Click **Create Project** to start the wizard. Use the information in the table below to configure the different wizard option:

Wizard Option	System Property	Settings
Project Name	Project Name	lab_comblock
	Project Location	C:\...\labs\
	Create Project Subdirectory	Check this option.
Click Next		
Project Type	Specify RTL	Select RTL . Keep <i>do not specify sources at this time</i> box checked
Click Next		
Default Part	Specify	Select Boards
	Board	Select ZedBoard Zynq Evaluation and Development Kit, Rev. D.
Click Next		
New Project Summary	Project Summary	Review the project summary
Click Finish		

1.3. After clicking **Finish**, the wizard closes and the project just created opens in the Vivado main GUI.

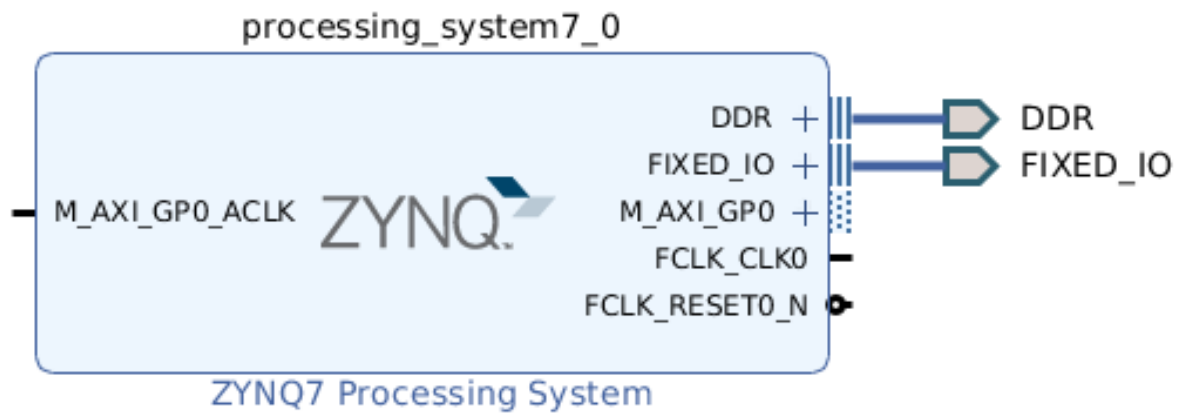
1.4. Click **Create Block Design** in the *Flow Navigator* pane under the *IP Integrator*. Type *comblock* as *Design Name* in the *Create Block Design* window. A new blank *Block Diagram* canvas will be presented.

1.5. To insert a *Processing System (PS)* block either click the *Add IP* icon  or right click on the canvas blank space and select *Add IP* from the available options.

1.6. Scroll down to the very bottom of the list or search using the keyword *zynq*, then double click on the *ZYNQ7 Processing System*.

1.7. The *Zynq7 PS* block is placed in the block diagram canvas. The I/O ports shown in the block diagram are defined by the default settings for this block.

- 1.8. Click Run Block Automation (in the green information bar). Make sure to check the option **Apply Board Preset**. Click OK.
- 1.9. Double click on the ZYNQ7 Processing System block, to open the Re-Customize IP window.
- 1.10. Click on the *Peripheral IO Pins* option under the *Page Navigator*. Uncheck all except Ethernet 0 (will be used in another laboratory) and UART1. Click Ok.
- 1.11. Finish with the Zynq (processing_system7_0) configuration by clicking the OK button on the lower left corner of the Re-Customize IP window.



Add the ComBlock to the Block Design

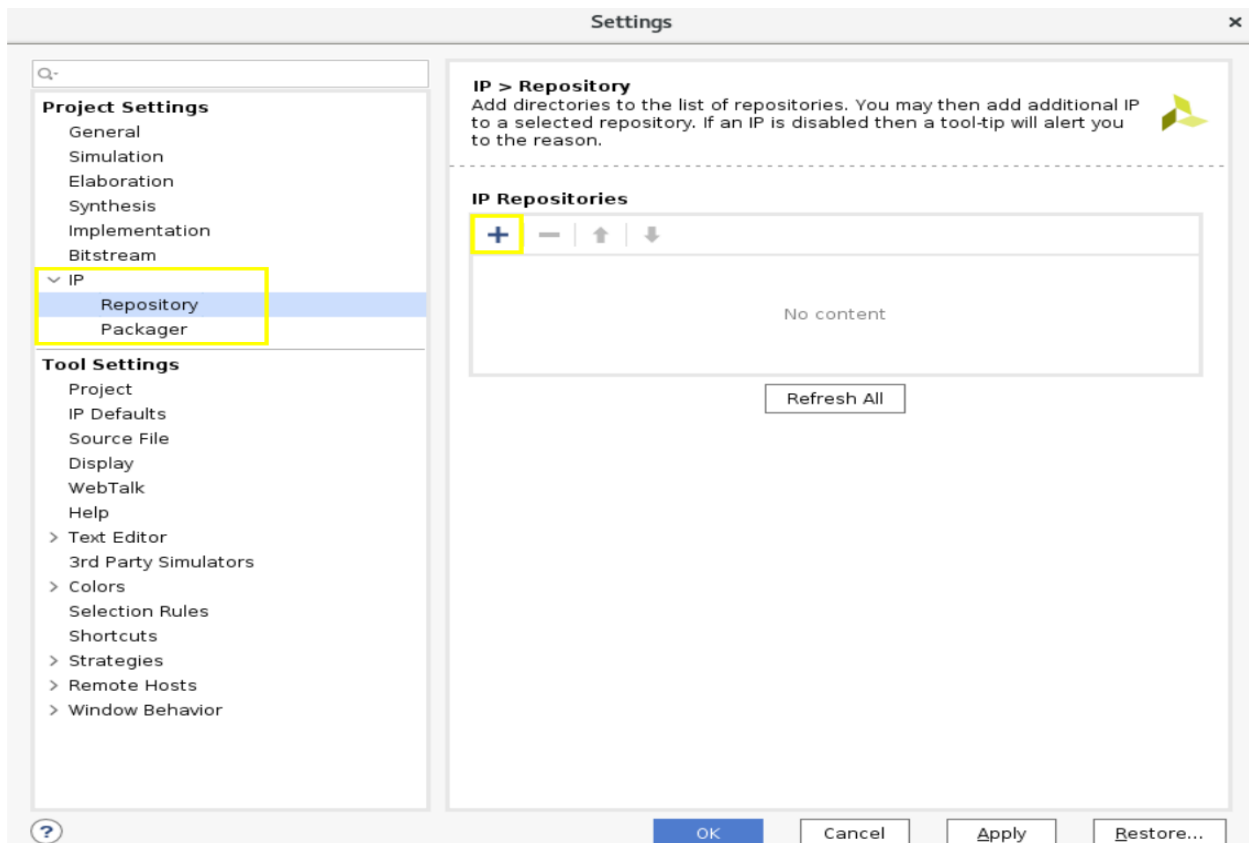
Step 2

Objective: download, unzip and setup the ComBlock IP to be used in Vivado.

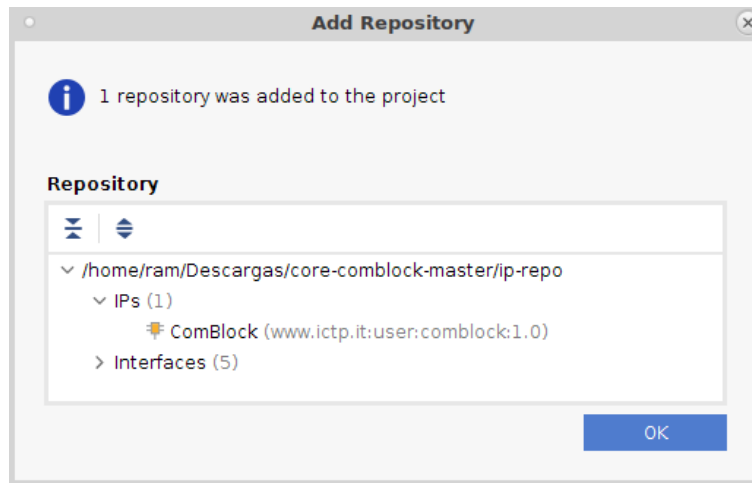
- 2.1. Go to <https://gitlab.com/rodrigomelo9/core-comblock> and download the repository as ZIP file.
- 2.2. Unzip the file under your *labs* directory.




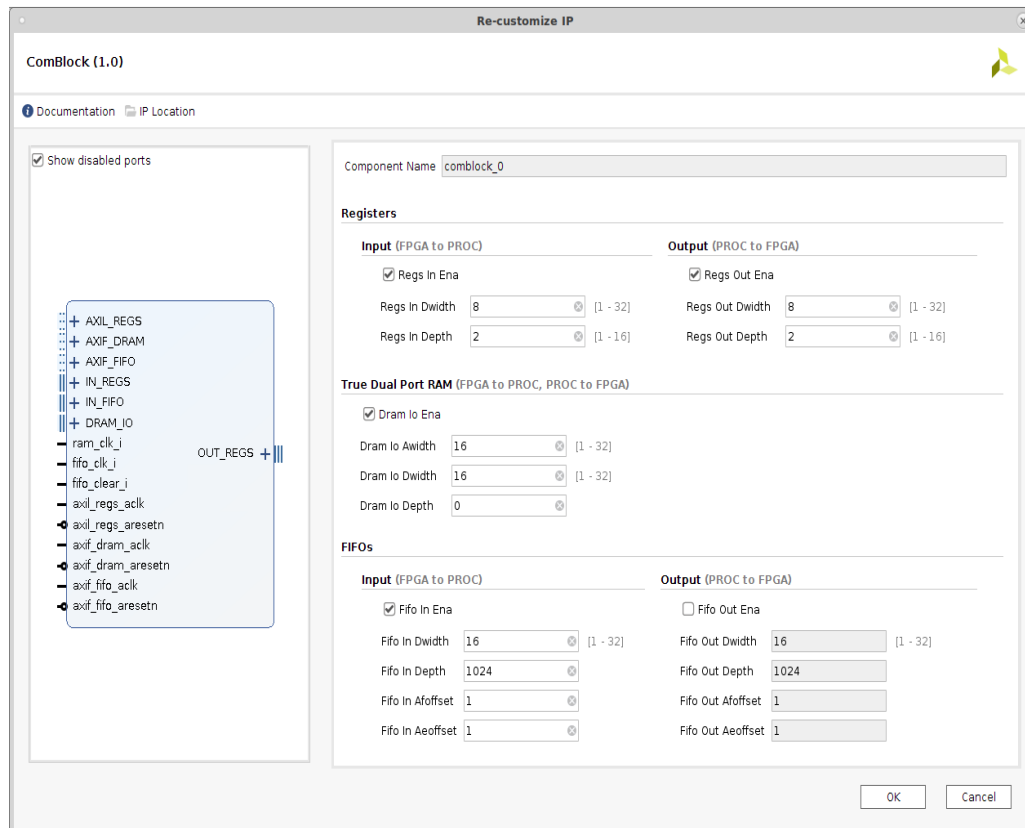
- 2.3. Click *Project Settings* in the *Flow Navigator*. Select *IP* in the left pane of the *Project Settings* form.
- 2.4. Click on the *Add Repository...* button.



- 2.5. Browse to ~\labs\core-comblock-master\ip-repo directory and select it.
- 2.6. The ComBlock IP should appear under IPs in the Add Repository window.

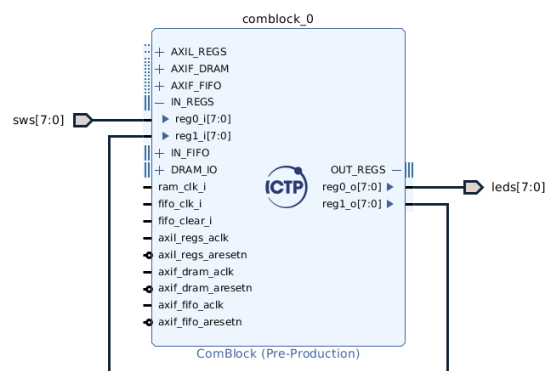


- 2.7. Click OK to finish the process.
- 2.8. Click the Add IP icon  and search the ComBlock IP in the catalog by typing "comblock" in the search field.
- 2.9. Double-click ComBlock to add the core in the design.
- 2.10. Double-click on the ComBlock block to open the configuration properties.
- 2.11. Set the configuration options according to the following image.



2.12. Expand *IN_REGS* and *OUT_REGS*, select the *reg0_o* and *reg0_i* ports by clicking on its pin, then right-click and select *Make External*. Rename the output port (*reg0_o_0*) to *leds* and input port (*reg0_i_0*) to *sws*.


2.13. Connect *reg1_o* to *reg1_i*.

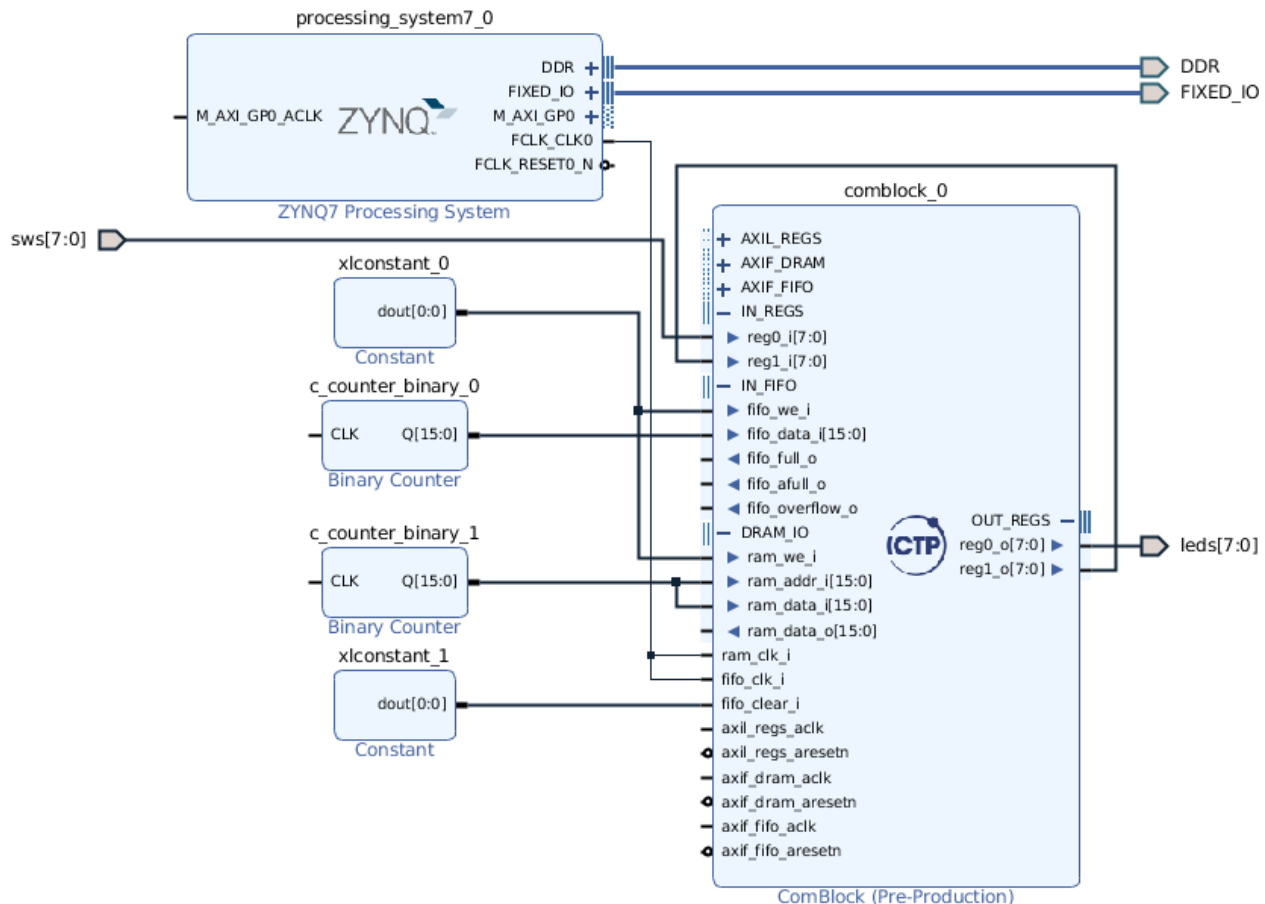


Add test pattern generators to the Comblock

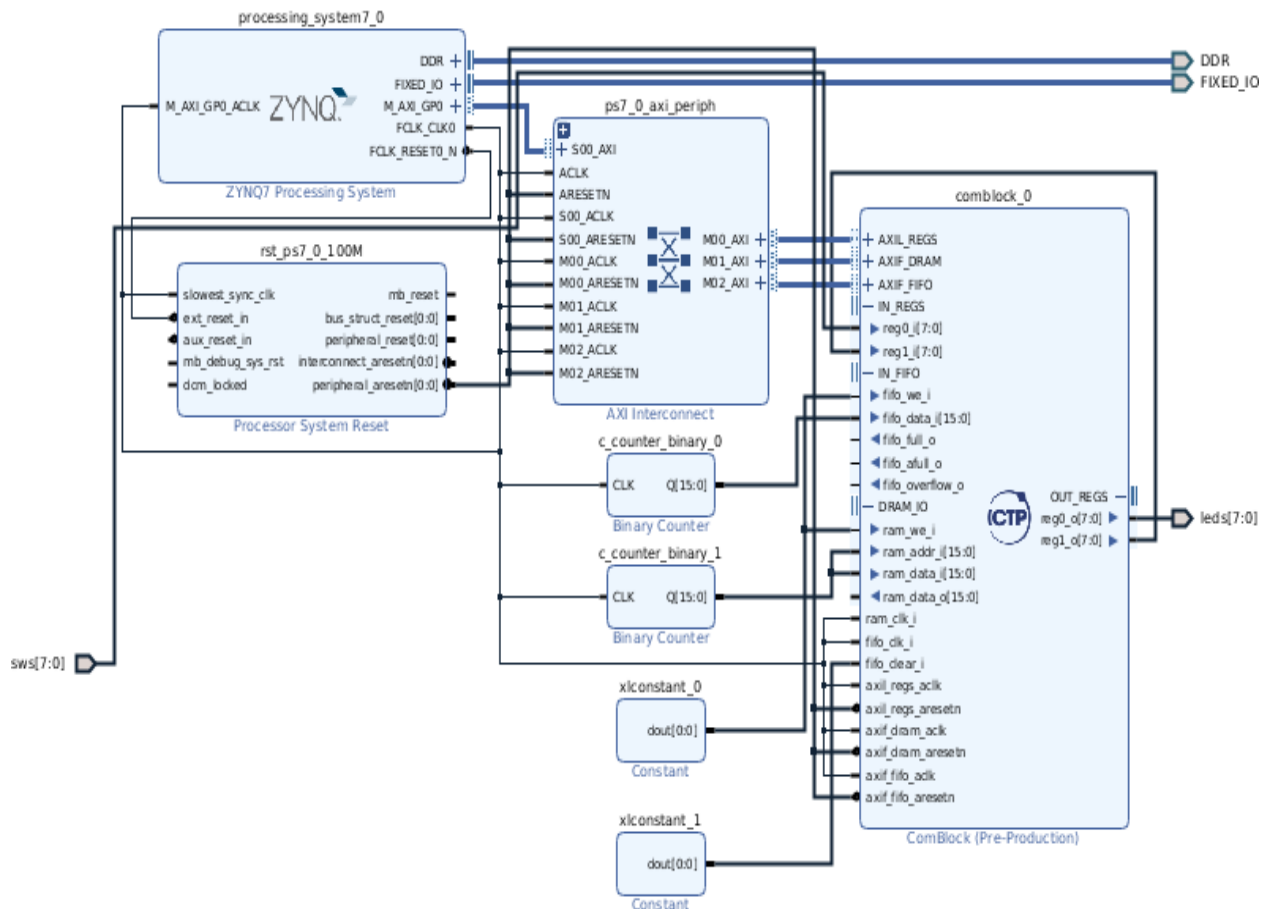
Step 3

Objective: add binary counters from the IP catalog in order to generate a stream of data for the DRAM_IO and IN_FIFO interfaces of the ComBlock. This will mimic a continuous stream of data from the PL to the PS.

- 3.1. In the Block Diagram, click the *Add IP* icon  and search for *Binary Counter* and add two instances to the canvas.
- 3.2. Expand *IN_FIFO* and connect the output of the *c_counter_binary_0* block (*Q[15:0]*) to *fifo_data_i*.
- 3.3. In order to write data to FIFO continuously, we need to tie *fifo_we_i* to 1. Add a *Constant* IP block from the IP catalog. Connect the output of the *Constant* block (*dout[0:0]*) to *fifo_we_i*.
- 3.4. We must also tie *fifo_clear_i* to 0. Add another *Constant* IP block from the IP catalog and configure the block by setting *const width* field to 1 and *const val* field to 0. Connect the output of the *Constant* block (*dout[0:0]*) to *fifo_clear_i*.
- 3.5. Connect the output of *c_counter_binary_1* (*Q[15:0]*) to *ram_addr_i* and *ram_data_i*.
- 3.6. Expand *DRAM_IO* and interconnect *ram_we_i* with *fifo_we_i*.
- 3.7. Connect *fifo_clk_i* and *ram_clk_i* with the *FCLK_CLK0* pin of the PS.



- 3.8. Click on *Run Connection Automation*, select *All Automation* and click OK to automatically make the connections.
- 3.9. Validate the design to ensure there are no errors (F6), and click the regenerate button (🔄) to redraw the diagram. The design should look similar to the figure below.

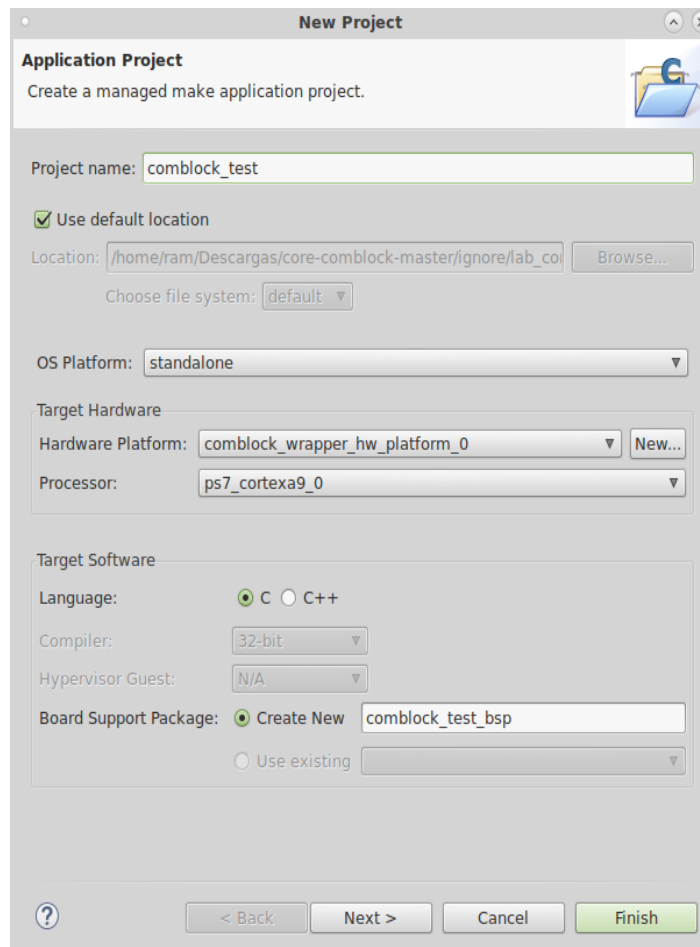


- 3.10. Click *Add Sources* in the *Flow Navigator* pane, select *Add or Create Constraints*, and click *Next*.
- 3.11. Click the *Add Files* button, browse to *P:/shared/3289/ComBlock_labs/comblock/* and select *comblock_test.xdc* (check **Copy constraints files into project**). Click *Finish* to add the file.
- 3.12. Expand *Constraints* folder in the *Sources* pane, and double click the *comblock_test.xdc* file entry to see its content. This file contains the pin locations and IO standards for the LEDs and Switches on the ZedBoard.
- 3.13. *Create HDL wrapper* of the block design.
- 3.14. In the *Flow Navigator* pane Click on *Generate Bitstream* and click *Yes* if prompted to save the Block Diagram. Also click *Yes* when prompted to launch synthesis and implementation. Click *Cancel* when prompted to *Open the Implemented Design*.

Export to SDK and create an Application Project Step 4

Objective: export the hardware along with the generated bitstream to the SDK and create a new application project.

- 4.1. Click *File > Export > Export Hardware* (include the bitstream). If prompted, click *Ok* to overwrite existing exported file. Then click *File > Launch SDK* and click *OK*.
- 4.2. Select *File > New > Application Project*.
- 4.3. Enter **comblock_test** as the *Project Name*, select *standalone* for *OS platform* and Click *Next*.



- 4.4. Select *Hello World* application and Click *Finish* to generate the app.
- 4.5. Add *#include "comblock.h"* and observe its content.
- 4.6. Modify the hello world example to:
 - 4.6.1. Write 0x99 to the output *REG1* and read from the input *REG1* (check if equals).
 - 4.6.2. Read 1KB from the *DRAM_IO* (you can use the *memcpy* function). Check that the values from 0 to 1023 were received.

- 4.6.3. Read 1KB from the *FIFO_IN*. Check that the 1024 values were consecutive.
- 4.6.4. Read the switches and write their value to the leds (in a loop which finish when all the LEDs are ON).
- 4.6.5. Inform the Start and the End of the test. Inform when an ERROR is found.

Test in Hardware

Step 5

Objective: test the application in hardware.

- 5.1. Connect the two micro-usb cables as well as the power supply correctly.
- 5.2. Turn on the ZedBoard.
- 5.3. Open and configure the serial terminal.
- 5.4. Click the *Program* button to download the hardware bitstream to the FPGA.
- 5.5. Configure a serial terminal.
- 5.6. Select *comblock_test project* in *Project Explorer* pane, right-click and select *Run As > Launch on Hardware (GDB)* to download the application.
- 5.7. See the result in the serial terminal.
- 5.8. What happens with the FIFO values if you run again the PS program without a reset in the PL? Why? Explain/ask to an instructor.

Optional

In the downloaded repository, go to the *examples* directory and try to reproduce the instructions in *README.md*. See and analyze the *block design* and the content of the *test.c*.