Lab 0: Introduction to Integrated Systems Design Labs

1 Overview

1.1 Introduction

PYNQ is an open-source project from Xilinx which integrates software and hardware components for faster development using Zynq devices. PYNQ combines the Python language with FPGA-based Programmable Logic (PL) and an Arm-based Processing System (PS) for building electronic systems. We'll be using PYNQ-Z2 boards for this lab, however the PYNQ software framework can be run on other development boards as well.

The PYNQ framework allows for user interaction through Jupyter Notebooks. 'Notebooks' are interactive documents containing live executable code organized into cells. PYNQ's Jupyter Notebooks are hosted on the Zynq's Arm processor and we can access them using a web browser (so long as the PYNQ board and the PC with the web browser are connected to the same network).

Vivado is used for creating the desired hardware system design (known as an *overlay*) and then generating the bitstream (*.bit) file. After transferring the bit file to a memory card inserted in the PYNQ-Z2 board, the hardware system can be programmed into the PL using a Jupyter Notebook.

For the initial 4C1 labs, we'll be using the PYNQ-Z2 simply as an FPGA board. Verilog HDL and Vivado will be used to implement various hardware design challenges. In later labs, we'll get to use more of the PYNQ functionality (like Jupyter Notebooks, building overlays and using more of the PYNQ's peripherals).

1.2 Definitions

Below are some of the definitions used throughout the 4C1 lab sheets:

- Field Programmable Gate Array (FPGA): Devices which implement hardware functionality using reconfigurable circuit elements.

- Hardware Description Language (HDL): HDLs are used to describe the behaviour or structure of digital circuits. They provide a formal description of a circuit which allows for automated analysis and simulated testing of the circuit. Verilog and VHDL are the most widely used HDLs and in this course, we will be using Verilog HDL.

- Overlay: Overlays are hardware system designs on PYNQ. It's a configurable and reusable class of Programmable Logic Design and can be downloaded into the Programmable Logic at runtime to provide functionality required by the software application. The PYNQ overlay has a Python interface, meaning it can be used like a Python package.

1.3 Resources

For more background reading, you can refer to:

- Vivado Design Suite User Guide: https://china.xilinx.com/content/dam/xilinx/support/documents/sw_manuals/xilinx2019_1/ug908-vivado-programming-debugging.pdf

- Exploring Zynq MPSoC textbook: https://www.zynq-mpsoc-book.com (Chapter 22 provides further explanation of the PYNQ framework and its applications)

- PYNQ Documentation: https://pynq.readthedocs.io/en/latest/

2 Vivado

2.1 Creating a Vivado Project with the PYNQ-Z2 board

In order to use the PYNQ-Z2 as the target board in Vivado, the board files must be added to the Vivado file on your local drive.

I. The folder called *'A.0'* provided on blackboard contains all the necessary files. Download the folder, unzip it and move it to the directory C:Xilinx/vivado/data/boards/board_files.

II. Open Vivado and create a new project. Give your project a name using the format AWalsh_lab0 and save it within your local directory (e.g. C:/Users/<your username>/Desktop/4C1_labs). Tick the box to create a project subdirectory.

III. Select RTL Project as the project type. Untick the "Do not specify sources at this time" box so we can add project files now. Alternatively, you can tick this box and add your sources after project creation is complete.

IV. When adding sources, ensure the target language and simulator language are both set to Verilog. Select the Verilog file provided on Blackboard *'lab0_top.sv'*. **Make sure you select "Copy sources into project". If you do not, you're project will be incomplete for submission.**

V. Add the provided *'PYNQ-Z2 v1.0.xdc'* as the constraints file. This allows for the mapping of your top module ports to the hardware on the board. **Make sure you select "Copy constraint files into project". If you do not, you're project will be incomplete for submission.**

VI. You must now specify the PYNQ-Z2 as the target board. Click the '*boards'* tab and select the PYNQ-Z2 board. If you cannot see the PYNQ-Z2 board, try re-starting Vivado or logging-out and logging back in. If you still cannot see the board, the board files were not correctly added.

VII. Select Finish to complete your project creation.

3 Controlling the PYNQ Peripherals

3.1 Understanding the Constraints File

As previously mentioned, the Constraints File connects the ports of your top level module to hardware on the PYNQ-Z2 board, allowing you to use the board's peripherals (e.g. switches, buttons, LEDs).

The constraints file provided has a list of example set_property commands which tell Vivado to create a bit file that connects a top level port to the PYNQ's FPGA pins. By default, each line of the constraints file is commented out. In order for you to use a peripheral, you must uncomment the appropriate line and write Verilog code in your *lab0_top.sv* file to describe the desired behaviour.

```
set_property -dict { PACKAGE_PIN D19   IOSTANDARD LVCMOS33 } [get_ports {
btn[0] }];
```

This constraint assumes you have an input port on your top module called btn[0], this will be connected to pin D19, which is connected to the first button on the PYNQ board.

3.2 Exercise: Control the PYNQ-Z2 LEDs

Your task is to write Verilog code and alter the Constraints File so that the PYNQ-Z2's LEDs light up when the respective button is pressed. Open the *lab0_top.sv* file in your project to get started.

I. Declare the module's input and output port signals in the module statement. For example, you should define four input buttons using the following line:

```
input [3:0] btn
```

This declares four buttons grouped together in a single bus named "btn". You can access an individual button using the root name and the signal index, e.g. btn[0]. Declare four output LEDs in a similar manner. **Ensure the root name you select for the buttons and LEDs matches the name in the constraints file.**

II. Assign statements are used to define combinational logic circuits. Verilog uses assign statements to assign a value to a wire. The assignment

```
assign led = btn;
```

will map the btn signal to led, i.e. if the button is pressed, the LED will light up. As led and btn are declared as buses, the assign statement is applied multiple times for each respective button and led. It starts with the least significant bus signal (assign led[0] = btn[0]) and ends with the most significant (assign led[3] = btn[4]).

III. Uncomment the lines in the constraints file which correspond to the peripherals you are using, in this case the LED and button peripherals.

IV. Generate the bitstream to test your design. To do this, go to Program and Debug section of the Vivado Flow Navigator as shown in figure 1. Synthesize the design, run the implementation then click generate bitstream.

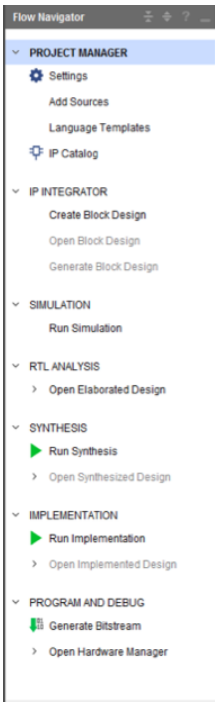

Figure 1: Vivado Flow Navigator Program and Debug Section

V. You can then download your circuit to your board using the Hardware Manager. Open the *Hardware Manager* in the Program and Debug section of the Vivado flow navigator.

VI. Click *Open Target* and select *Auto Connect* to automatically connect to your PYNQ-Z2 board. Click *Program Device*

4 Submission

Please submit the following for this lab:

- Your project as a zipped file using your name and lab0 as the file name (e.g. AWalsh_lab0).