Lab #1
Linux and the ZedBoard

Introduction

This lab is an introduction to the Zedboard, an embedded computing device based on an ARM processor and a Linux operating system. We will be using this platform throughout this course to get a hands-on experience working with an embedded system. The platform is a very rich environment with abundant hardware features, though we will primarily focus on the ARM processor (CPU) and the Xilinx FPGA, both of which are part of the Zynq System-on-Chip, the central chip on the ZedBoard. The ZedBoard also contains a wide set of input/output (I/O) devices, among which we will use LED lights, switches, buttons, and Pmod interfaces connected to a Wiimote and a robotic arm.

The Zedboard is running a version of the Ubuntu distribution of the Linux operating system, called Xillinux (Xilinx + Linux). This is a slightly scaled-down version of Ubuntu, but has a lot of the functionality of a full-fledged distribution of this operating system. The operating system runs on the ARM processor, and supports the execution of custom C or C++ programs.

Submission of lab reports

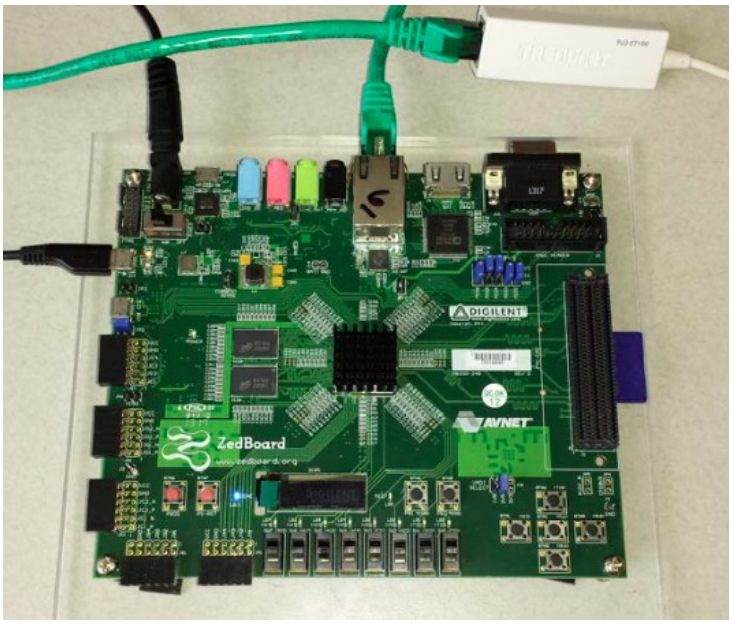
Each lab manual, including this document itself, is structured as a small tutorial on how to use certain parts of the ZedBoard, plus a set of assignments. You will find each numbered assignment enclosed in a text box. You will need to get familiar with lab assignments before coming to class. For this purpose, lab manuals will usually be available on Blackboard around one week before the corresponding lab session. You will realize that part of the assignments, including C/C++ programs or hardware designs, can be sketched on paper before the lab session itself. If you do so, you will probably need no more time that one single lab session in order to complete any lab assignment. However, the lab room will be open and proctored by at least one teaching assistant at designated time slots every day during the week, in order for you to complete any unfinished lab assignment.

For each lab session, you need to submit a lab report as one single Word or PDF document, together with additional attachments as requested in each specific lab manual. The lab report should contain the answers to each of the assignments found within the text boxes.

Lab reports must be submitted on Blackboard (section *Assignments*) by the due dates specified on the Syllabus document, usually six days after the corresponding lab session. Notice that these due dates are strict, as specified on the course syllabus. Before submitting each lab report, please verify that it is the final version, since you will have only one possible submission attempt. If you are working with a lab partner, one one person in the team must submit the report, making sure that you specify the name of both team members in the header of the document.

Connecting to the ZedBoard

The first step to familiarize yourself with our lab equipment consists in physically connecting the ZedBoard and creating an SSH (*Secure Shell*) connection between your computer and the ZedBoard:



- Log in to the Windows desktop PC. We will refer to this system as the host. Use your MyNEU credentials.
- Make sure your ZedBoard is powered, that is, plugged in and with the power switch on the ON position. Give it some time to finish the booting process, usually around one minute. Then connect the ZedBoard to the host via an Ethernet cable (green cable on the right figure). Connect one end of the Ethernet cable to the Ethernet connector on the board, and the other end to the Ethernet-to-USB adapter (white accessory on the right figure). Connect the USB end of the adapter to the host.
- Then find the *MobaXterm* program that will be used to run a secure shell (*ssh*) to connect to the ZedBoard. Select menu *Sessions → New session → SSH*, and fill in the basic SSH settings. The ZedBoard has been configured to be accessible at the local IP address 192.168.1.10, and the default SSH port is 22.

  We have created different user accounts on the ZedBoards' SD cards. Your user name will depend on the course section that you're registered in, as well as your board number, as follows:

  `user<SectionNumber><GroupNumber>`

  Here, `<SectionNumber>` is the number of your EECE 2160 class section, and `<GroupNumber>` is the 2-digit number of your group, printed on your computers and on your board, using a leading “0” if necessary. For example, the group in Section 6 getting the ZedBoard labeled “3” will have user name `user603`. The password is the same as the user name.
- Once you are logged in, type command `pwd` and make sure that the initial working directory is set to `/home/user<Section><Group>` (for example `/home/user603`).

Hello-world on the ZedBoards

As a first lab assignment, we will run a simple *hello world* program on the ARM processor of the ZedBoard.

- Create a directory named `lab1` in your home folder. In the future, make sure that you have a separate directory for each lab session, where you will place the corresponding files.
- With `lab1` as your current working directory, use the `vi` editor to create a C++ source file named `hello.cc`. Write the *hello world* program in this file.
- Use `g++` to compile the program. You should generate an executable file named `hello`, also located in directory `lab1`.
- Execute the program and verify that the output is as expected.
- Open a new SFTP session in MobaXterm, and transfer the C++ source file back to the PC. You will need to find a writeable place on the PC, and a good option for this is the system desktop or a flash drive. For the SFTP session, use IP address 192.168.1.10 as the ZedBoard address, and type your user name. Notice that you cannot rely on the host system's desktop to keep your files across session, so make sure that you keep all your files in a USB flash drive at the end of each lab session. You cannot rely on the ZedBoards to hold your files across lab sessions either, since SD cards are periodically re-formated.

Assignment 1 (3 pt.)

a) (1 pt.) Show the content of the C++ program that you wrote on the ZedBoard and then transferred back into the host (file `hello.cc`). You can just copy and paste the code in your report.

b) (2 pt.) Take a screenshot of the MobaXterm terminal after the execution of the program, and add it to your report. Make sure that it contains the list of all commands you ran on the ZedBoard in order to edit the source code, compile it, and run it.

Sorting an array of integers

As a more complex programming exercise on the ZedBoard, write a C++ program that generates an array of 10 random integer numbers between 0 and 99, prints the original array, sorts the array in ascending order, and prints the sorted array. Here are some suggestions on how to proceed:

- Edit the program source on the ZedBoard in a file named `array.cc`, also placed in directory `lab1`. You should also use the *gcc* compiler to build your program into an executable file named *array*. This file should run correctly on the ZedBoard and provide the expected output.

- You can start by writing a function named `PrintArray` with the following prototype:

```
void PrintArray(int v[], int size)
{
   ...
}
```

  This function takes an array `v` of integers as the first argument, and its `size` in number of elements as the second argument. The function traverses the array with a loop and prints all elements between 0 and `size` – 1.

- Write another function named `RandomArray` with the following prototype:

```
void RandomArray(int v[], int size)
{
   ...
}
```

  This function should initialize `size` elements of array `v` to random values between 0 and 99. You can use function `rand()` to generate random numbers. You can obtain information on how to use this function from the manual pages with shell command `man 3 rand`. You can limit the range of the generated random number with a modulo operation (operator `%` in C++).

- Write a function named `SortArray` with the following prototype:

```
void SortArray(int v[], int size)
{
   ...
}
```

  As its name suggests, this function sorts array `v` composed of `size` integer elements in ascending order. Use the *selection sort* algorithm to sort the array. This algorithm is based on finding the minimum element on the right of the array, and placing it on the left in an iterative manner. In each iteration, the array will be split in two parts: one that is sorted on the left, and one with the remaining unsorted elements on the right. As the algorithm progresses, the sorted part grows, and the unsorted part shrinks. You can find more information about selection sort in the following Wikipedia page, which includes a basic implementation of the algorithm and an animation illustrating its behavior:

  https://en.wikipedia.org/wiki/Selection_sort

- Finally, write a `main()` program that instantiates an array of 10 elements and invokes the previous functions in the appropriate order.

Assignment 2 (7 pt.)

a) (5 pt.) Upload a file named `array.cc` containing the source code of the program described above. The code should compile correctly without any modifications using the `g++` command on the ZedBoard.

b) (2 pt.) Take a screenshot of the MobaXterm terminal after executing your program and include it in your report. The screenshot should show the output of the execution, as well as the sequence of commands you ran to create the source file, compile it, and run it.