**FPGA Lab Python Interface Guide**

**Introduction**

A Python interface infrastructure is provided for advanced interaction with your FPFA Lab design. The infrastructure enables easy interaction with the design and in fact utilizing your programmed FPGA hardware design as a hardware extension of your Python software program.

**Prerequisite**

In order to utilize the python interface, you should have basic knowledge and experience with Python programming. Additionally you should have a Python prompt available on your hosting computer, in case you don’t have such serval installation sources are available, one popular recommended installation is [ANACONDA](https://www.anaconda.com/distribution/) , if this is your choice go to its windows download section and get **Python 3.7 version.** Once installed you can either run Python from the anaconda prompt (change directory to the fpga\_project folder) or from a “PowerShell window” (SHIFT + right-click at the folder from where you want to run Python)

You should also have the Python serial package installed, to test if it exists enter python and type **import serial ,** in case it is missingyou should install it typically by running ***pip install serial***

**The Python Interface**

A local package named **fpga\_lab\_access.py** is provided in the FPGA Lab **py** folder, this folder interface to the serial COM port and provide following access routines:

***write\_reg*** *(reg\_index, data )*

Writes 32 bit value to the FPGA -lab input port indicated by address.

***read\_reg*** *(reg\_index) # returns signed integer*

Reads 32-bit value from the FPGA -lab output port indicated by address.

***read\_reg\_unsigned*** *(reg\_index)*

Reads 32-bit value from the FPGA -lab output port indicated by address.

Returns UNSIGNED value

**Usage:**

Create a new .py file for example , usage\_example.py , and use following template

*import sys*

*import random*

*import fpga\_lab\_access as fla*

*# Main*

*serPort = fla.configPort(sys.argv[1])*

*…. Your Design specific code , utilizing write\_reg and read\_reg , read\_reg\_signed functions*

**Example:**

See and understand py/usage\_example.py for interfacing with the lab reference example.

It randomly adds or subtract values from the accumulator and prints them out.

Run the program from the prompt by:

check on the Device Manager which COM is your serial COM as follow, right click “This PC” icon on your desktop and go to properties -> device manager, scroll to ports and check your indicated USB serial port.

***python usage\_example.py COM***

(use your specifically indicated port name)

It should respond as follow for example:

py>python .\usage\_example.py COM

Configured and opened serial port COM

*-2 : acmltr = 94*

*-1 : acmltr = 93*

*-7 : acmltr = 86*

*-9 : acmltr = 77*

*+3 : acmltr = 80*

*+7 : acmltr = 87*

*+0 : acmltr = 87*

*+9 : acmltr = 96*

*+3 : acmltr = 99*

*+7 : acmltr = 106*

Now create your own version of usage\_example.py to specifically interact with your design.