Typical PYNQ Development Steps

NOTE: PYNQ library only simplifies HOST (CPU) programming (Python programming instead of low-level C programming) of an SoC / CPU-FPGA system, and it doesn't change the FPGA hardware development flow itself. You will need to create FPGA design as before (with Vivado). (<ad time starts> **OR**, use ***PyLog*** to create your FPGA design! <ad time ends>)

1. Complete FPGA synthesis and generate bitstream, get the following two files:
    a. **Bitstream (*.bit)**, you can search for it with command: find . -name "*.bit". Its typical location is: ./{project_name}/{project_name}.runs/impl_1/design_1_wrapper.bit
    b. **Hardware Handoff File (*.hwh)**: search: find . -name "*.hwh". You may find multiple *.hwh file, the right one is: ./{project_name}/{project_name}.srcs/sources_1/bd/design_1/hw_handoff/design_1.hwh

2. Copy the bitstream file and hardware handoff file to ZedBoard. (You may also want to rename these files so that they have a meaningful name)

3. Create a Python script that loads the bitstream, allocates memory and interacts with FPGA.

4. Run the Python script with "**sudo** python3.6 {your_python_script}.py". Note: pynq module requires sudo to import.

You can find example Python code here:
https://drive.google.com/drive/folders/1fmOanJ8SBiIMNbKRfhSZu8IKwhe1k_9n

- zedboard_test.py: This is the example host code
- pylog_add.bit: bitstream
- pylog_add.hwh: hardware handoff file
- (not needed, only for reference) pylog_add.cpp: C code used for high-level synthesis (generated by PyLog)
- (not needed, only for reference) vecadd.py: PyLog source file that generates C code

REFERENCES

- PYNQ Documents: https://pynq.readthedocs.io/en/v2.5.1/
- PYNQ-Z1 Board Setup Guide: https://pynq.readthedocs.io/en/v2.5.1/getting_started/pynq_z1_setup.html