Reconfigurable Arbitrary Waveform Generator with PYNQ-Z1 Board

Advisor: Muham

Timothy Nunez **Trenton Condrin** **Anthony Ho** **Carlos Ramos**

Mahan Bastani

Abstract

In this project we took advantage of the PYNQ-Z1 board to create a reconfigurable waveform generator. The PYNQ board is a derivative of the Z1 board that uses the PYNQ open-source framework. The PYNQ framework allows FPGA developers to import programmable logic circuits as software libraries into Python code. Python "drivers" can then be written to create a simple interface to control the logic circuits. In our project we have a couple hundred lines of Python code that calls a programmable logic circuit we designed in the C programming language and compiled to a binary file using the Xilinx SDK. With this design, we can easily generate any waveform saved to a CSV file through the Python driver we designed.

Keywords

PYNQ (Python Productivity for Zynq): An open source framework that enables engineers to design IPs without using conventional ASIC-style design tools.

Python Driver: Python class that creates an easy to use interface (i.e. a set of class methods) for using programmable logic circuits (in our case a binary file)

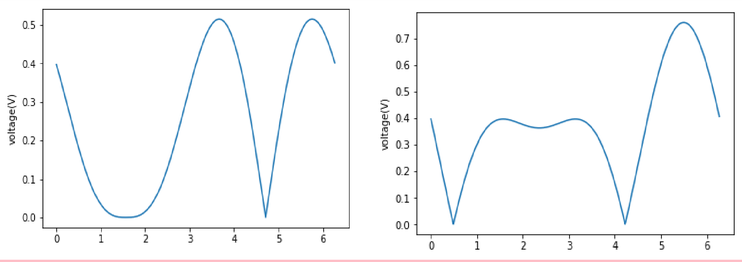
.bin File: A binary file that represents a block of logic we can use to program the FPGA portion of the board. We compile these from C code using the Xilinx SDK.

Overlay: Premade Python driver made by Xilinx that is on the PYNQ image we used.

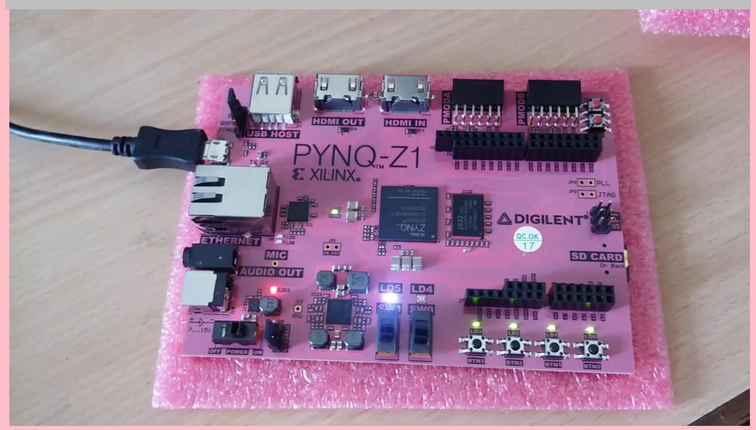
PMOD DA4: Digital to analog converter that we used in this project.

MicroBlaze Processor: A 'soft' microprocessor we utilize to send commands and data to our programmable logic circuit.

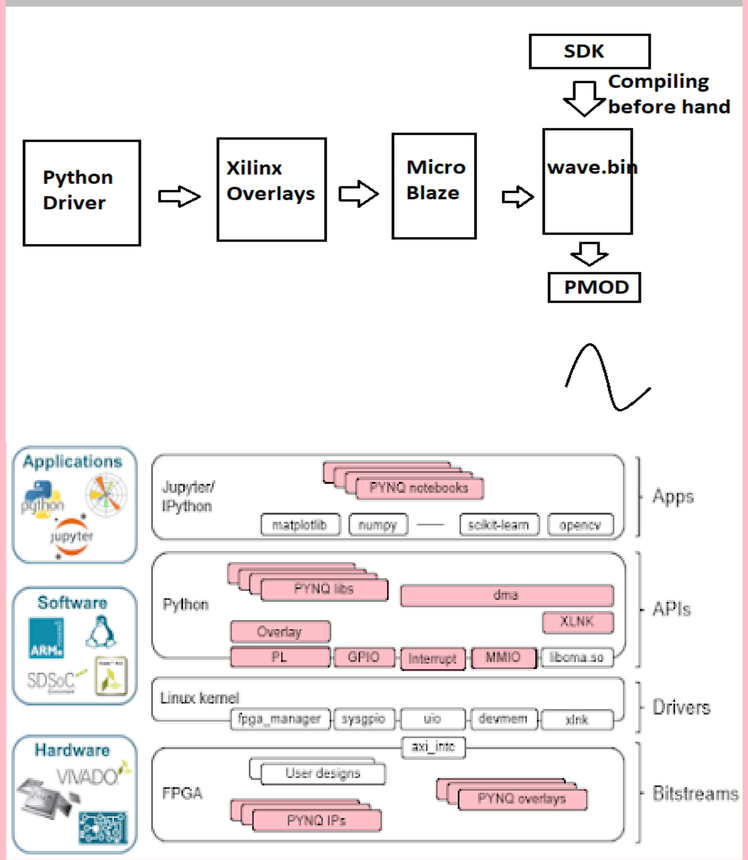
Controlled Results



Board



Control Diagram



Methodology

The motivation of this project comes from the need to be able to generate waveforms of arbitrary shape. This might be used for signal processing purposes or for verification and debugging of hardware devices. For example, an engineer might want to test communication devices by generating specific waveforms and seeing how the device reacts. Using our project, the engineer would only have to create the waveform desired in a CSV file and then run our Python driver.

In addition to the code we wrote for this project, we also utilize the overlays written by Xilinx. For example, in order to interface with the PMOD DA4 chip we used for digital to analog conversion, we used the PMOD overlay on the PYNQ image.

Results