

Getting First Measurements and Simulations through Demonstration Blocks

In the previous document, you learned how to make the resulting tools work, including a simple example reading and writing through the FPAA fabric using the remote tool set. In this document, one will learn how to connect to the board and get first level measurements working through using the demonstration system.

A major focus of this discussion will be connecting to the physical device, which will include

- getting the USB communication working
- identifying the I/O pins to connect outside the IC
- understanding the code base for using the Digilent system
- utilizing of the ADC block element for in-hand and remote board usage.

Further, we will have a discussion showing the implemented simulation capabilities for this system; each block should have both a simulation model as well as compile to a working implementation in FPAA hardware. We assume that everyone has gone through the material in VM_FPAA_setup01.pdf.

Working USB communication

The first case is making sure the FPAA board is communicating with the virtual machine. Once you plug in the USB cable connected to the FPAA device, you can check whether the virtual machine sees the board by pressing the USB picture at the lower corner of your VM. You will want to make sure that “Ga. Tech CADSP Lab RASP 3.0 Control Board” is selected, indicated by a check in front of those words as illustrated here.

External I/O to the FPAA board

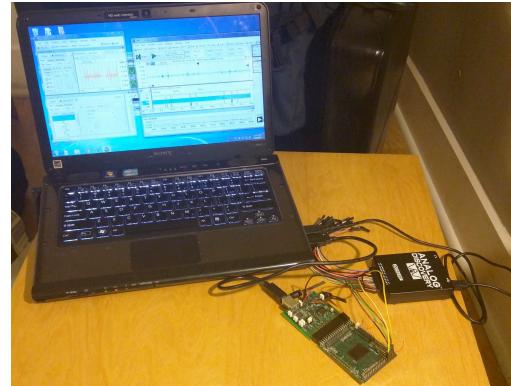


This section focuses on the basic I/O pins possible to connect to the FPAA board for this experiment in addition to the USB connection described in the previous section. The diagram shows that we have 8 I/O pins labeled, labeled between 1 and 8, that correspond to the I/O pins that are used in the high-level Xcos tools. These I/O pins are ones we will use with the Digilent board as well as other external interfaces; when we need more than 8 I/O pins we will identify others on the resulting board.

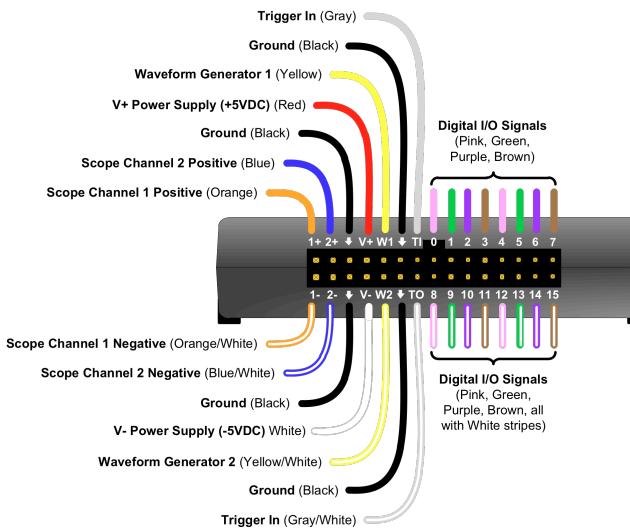
For Experts, the pins (1) and (2) actually have a more complicated definition in the FPAA device (9 0 0 and 11 0 0 respectively); non experts almost certainly appreciate this abstraction. Further, there is a hidden file describing the pads (often called “pads file”) that translates pins in Xcos to the resulting I/O pins given.

Using the Digilent Board through tools for Data Acquisition

The focus of this section is to illustrate the Digilent board, Analog Discovery, partially developed by Analog Devices, as the primary additional bench tool for using these FPAAs devices. Effectively the combination of these two elements as well as a functioning laptop computer constitutes a fully working (and portable) testbench for most classroom experiences as well as development for many application environments. We show such an illustration of a test system operating in San Francisco (2015) as a fully working test infrastructure.



One might first ask why use the digilent board at all since we can compile all sorts of analog and digital blocks directly on the fabric? Two typical views are in adding capability for fully utilized FPAAs devices, as well as adding known debugging capability for building up a particular hardware interface. More typically, the board allows implementation of some data functions that, although can be implemented on the FPAAs device and resulting board, might take some time to fully implement. (e.g. continuous streaming data).



on those pins). Other GND pins as on the board document are also possible.

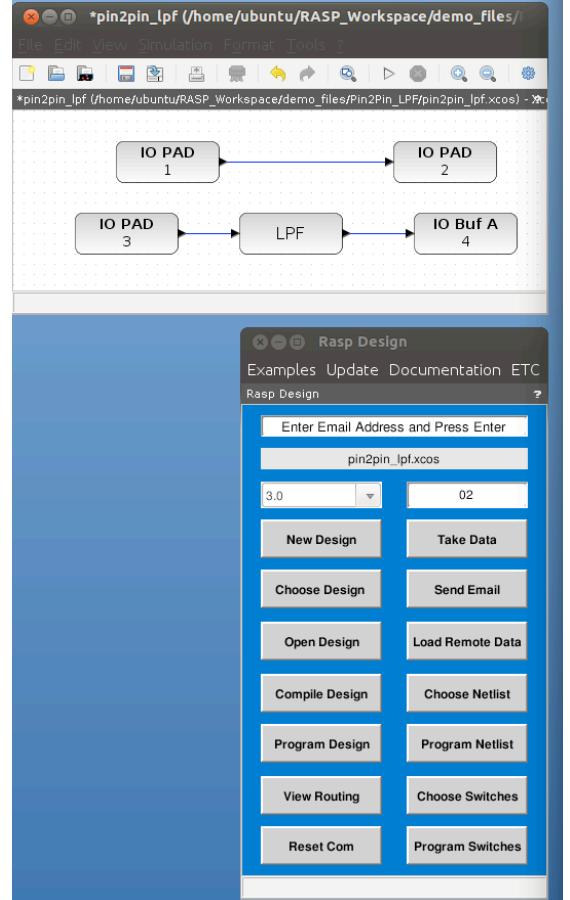
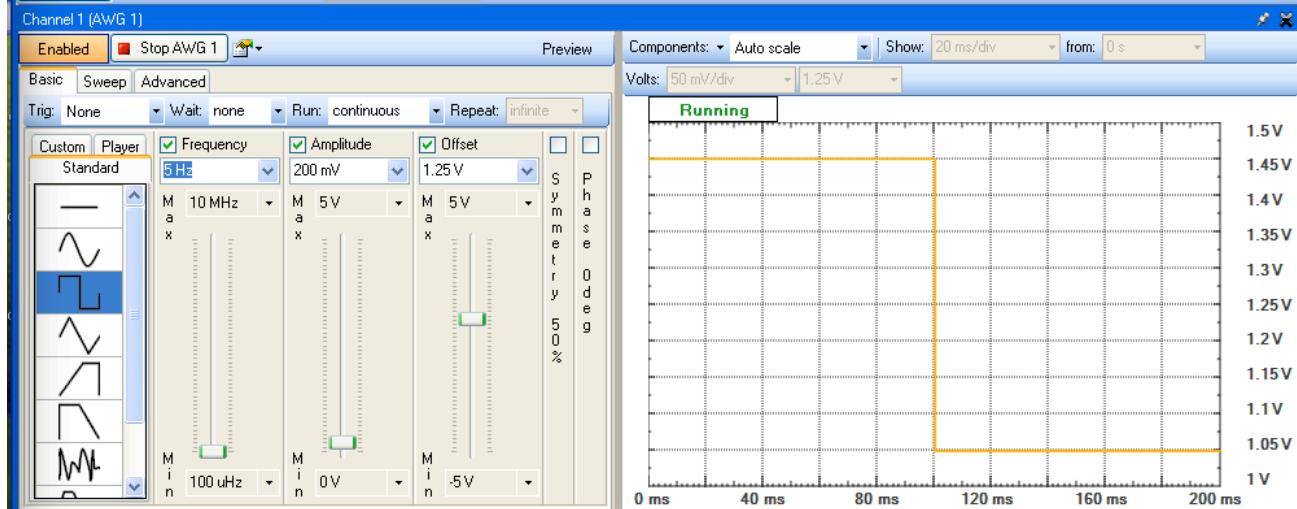
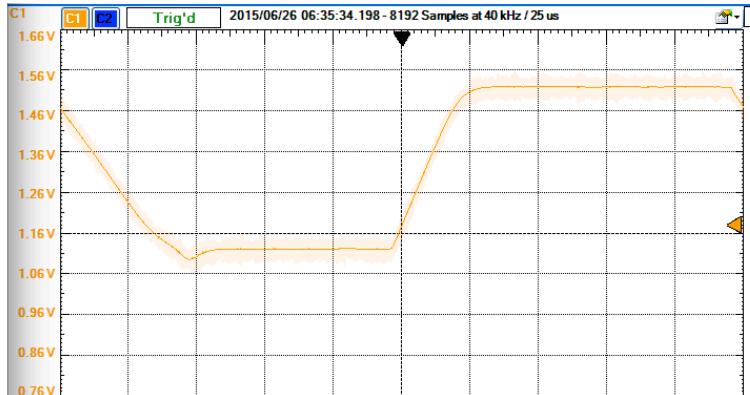
The orange and blue wires at the one end are the differential ends of the two scope channels. One can place the negative end of the scope channel to GND as well as use the channels in differential mode. The devices are capable of doing a single-ended measurement as well.

The yellow wires are the single ended waveform generators (referenced to GND).

Note: be gentle on the USB connector for these boards. Apparently plugging and unplugging this connector in *rough* ways results in the connector no longer working and becomes difficult to fix the resulting board afterwards. Most find this one aspect is, by far, the weakest link on a very powerful tool.

As a first measurement, we recommend you start up your VM and use the pin2pin_lpf demonstration on the FPAA GUI block. You will see the block diagram which has a set of switches between two I/O pads as well as a single LPF block between two I/O pads (shown in figure). The particular pin shown on the block is the pin shown in the board photo (with pins 1-8) above. You will want to select that demonstration, and then press “Compile Design” (in the FPAA GUI shown) to compile the design to a switch list that can be used to program the FPAA device. Make sure that your USB port is activated as mentioned above. Then you will want to press “Program Design” to send the program to the FPAA device; you will see multiple messages during programming, but they are only interesting for the seasoned expert. When finished, remember to press “Program Design” to enable the IC to be turned onto its normal operational mode and sets the resulting DC values. “Take data” is only when update the DAC values with compilation from the graphical description, enabling to compile and setting the new DAC voltages using “Take data” (instead of programming the board again).

Then we begin with the exciting part of measuring our FPAA circuit, in this case using I/O pins to a Digilent device. We will show the output for measuring the LPF block; one can see a similar



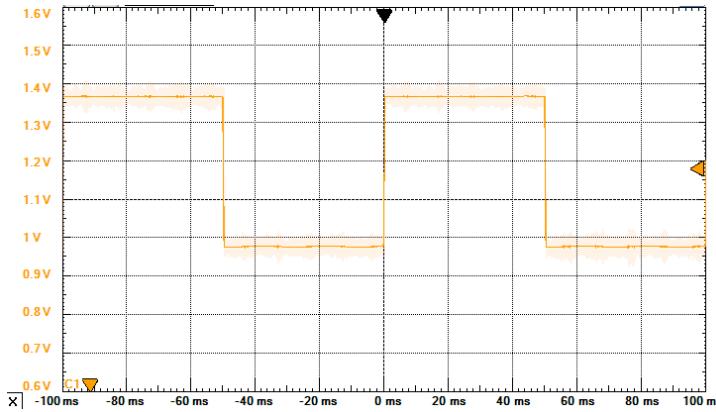
measurement for the route through the FPAA block.

In this example, we use the first Arbitrary Waveform Generator (AWB) channel for the input (pad 3) and use scope channel 1 (positive) for the output (pad 4). The negative terminal of the scope channel is connected to GND. The

waveform generator block allows for a wide range of functions, where we show a step function input being given. The measured output trace is a typical LPF response, including an expected slew-rate regime given the low linear range ($2U_T / \kappa$) for this particular OTA device. The speed shown is typical for the measurement relating to the resulting capacitance on the line and the low bias current used.

We show a second version of the measured data for an input square wave (5Hz, same voltages) to show the different behavior for the resulting the compiled wire measurement through the LPF.

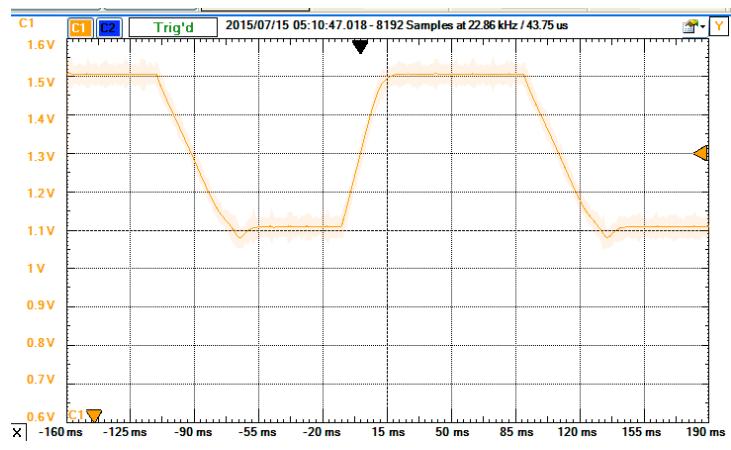
We show a version of the data for a different input



on your system. Make sure you should compile this file (pin2pin_lpf, from the demo file), and make sure it is turned to run mode. Then follow the following sequence:

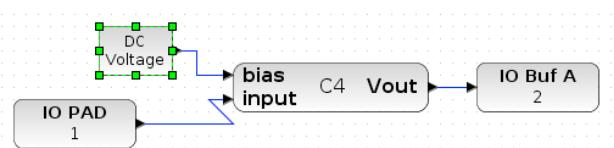
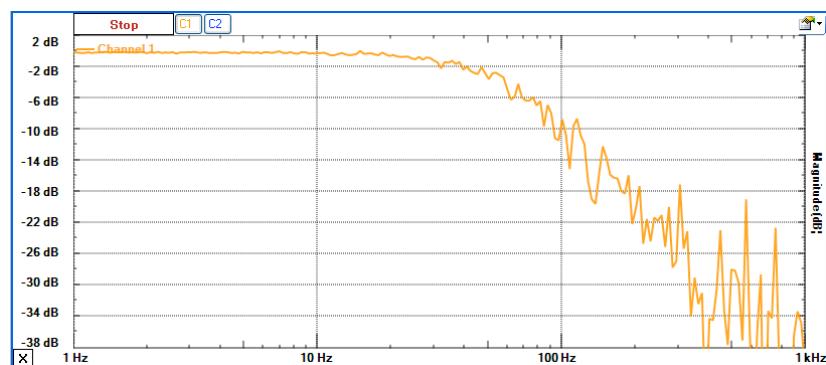
- Measure the transfer characteristic between two switches (I/O Pad 1 and I/O Pad 2)
- Simulate the behavior of the LPF
- Measure the behavior of single LPF,
 - Low Frequency Transfer Function
 - Step Response for LPF
 - Measure the Frequency Response for LPF (we show a plot for the first case) using Digilent's built in function
- Add a second LPF block at a different corner frequency and measure the resulting block

Next, lets look at a second circuit block, also found in the demonstration circuits, the C4 Bandpass filter block. The block diagram for the circuit is shown, where the options can be input directly by double clicking on the block.

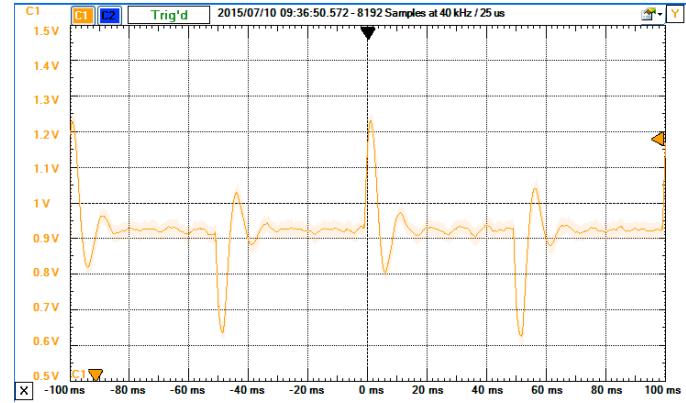


step response to show the different behavior for the resulting the compiled wire measurement using the measurement I/O pads as shown in the original experiment diagram for a 10Hz input square wave voltage waveform (step from 1.05 to 1.45V). We expect that the data should effectively pass through the resulting single FG switch elements given the roughly uniform conductance for the signal range.

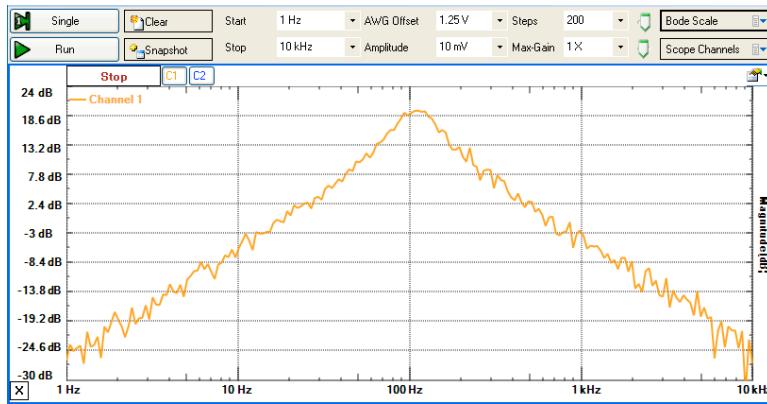
We would recommend one starting with the board to begin a set of initial measurements after finding that the board and the digilent device is working



We show the input signal as well as the step response output for the second order section in the demonstration block, which gives a small Q (roughly 2) for the resulting circuit. The core circuit is a C4 bandpass filter block built using two programmable (FG) OTA devices. Looking at the block diagram from the user's perspective, one does not see circuits, but rather a simple system block. This block is capable of a vector of inputs and outputs represented by their input and output lines. We show the circuit response for an input square wave (10Hz, 40mV amplitude, 1.25 offset) showing the characteristic overshoot seen



for a $Q > 0.5$.

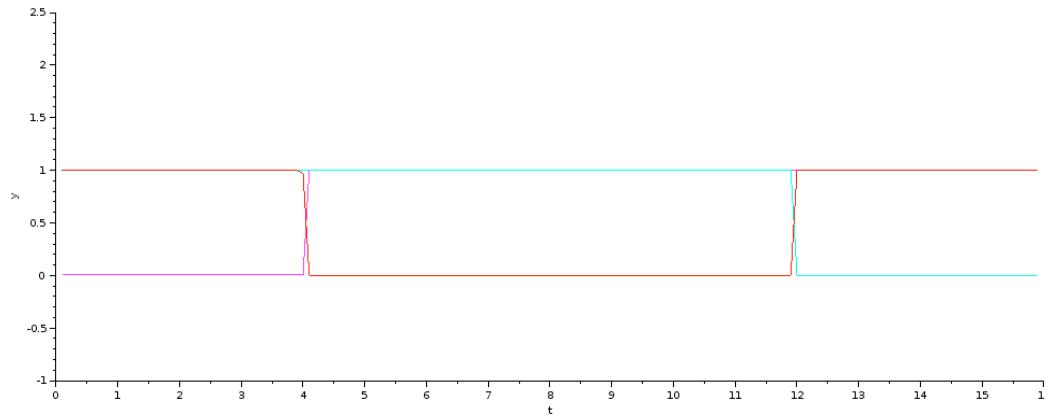


We also demonstrate the frequency response curve for the second-order section using the built-in function for the analog discovery block showing the characteristic 1st order rolloff for high and low frequencies and the gain amplification at the center frequency corresponding to $Q=2$.

Next sequence is to measure a digital block using the resulting demonstration elements.

The VMM + WTA classifier simulation demonstration computes a 2 input XOR case, showing the computational power for a single one-layer network. The input vector has three inputs at digital levels where 0.2V is a low signal (0) and 1.2V is high signal (1); two inputs go through the truth table sweep (00,01,10,11) with the third signal, the control value, held at 1.2V (1) signal. The weight matrix is [2 0 1.5; 0 2 1; 0 2 1] in simulation, weight matrix, biased around a current level for these weights as built in the simulation model and consistent in the experimental framework.

The VMM + WTA classifier experimental demonstration computes a 2 input XOR case, where the source-coupled input signals are 2.1V (0) and 2.3V (1) for the low and high input values, respectively.



The VMM + WTA classifiers compile with a hard coded 4 input, 4x4 VMM into the 4 input WTA block. The first signal is the

control (held at 2.3V (1)), the second and third signals are the 2-input XOR signals, with the last signal to be ignored. We have a weight normalized to ()nA (W = 1), where we program a weight to 1nA for a near 0 value (W = 0.01) for this computation. The two inputs go through the truth table sweep (00,01,10,11) for the resulting input signals.