# EE 3420 Lab Guide: Analog to Digital Converter

*Written by: Grant Seligman, Gabe Garves, and James Starks*

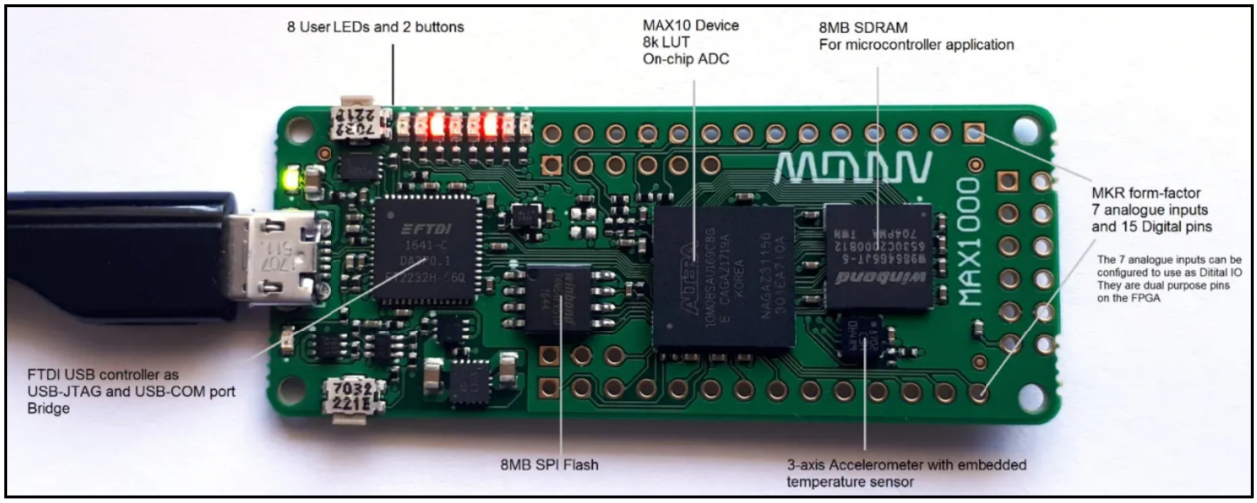


**Figure 1**

## Example Overview:

The MAX10 FPGA comes with an on-chip analog to digital converter (ADC). Our objective is to program the FPGA to utilize the ADC internal IP core. In the past, Verilog modules would have been created to provide this functionality. Fortunately for us, Quartus already has a suite of HDL modules in their IP Catalog including the ADC module. Here we explain the process of implementing this ADC module and how to use Quartus' System Console to not only measure the input waveform, but capture and export the raw data to a .csv file.

## FPGA Implementation:

https://www.youtube.com/watch?v=0oO1RFa-4Xk.

This is a great video showing the step by step process of implementing the ADC module from the IP Catalog. The only difference is the FPGA mentioned in the video is different from ours. This means the two pins used and a few settings are different. Those two pins are the clock (pin H6) and user button (pin E6). The settings and walk through are on the following pages.

Begin by creating a Quartus Project for our FPGA: **10M08SAU169C8G.** Then you will need to open up the **Platform Designer** by pressing this button in the middle top bar menu. 

Go to the **IP Catalog** and type in **“ADC”** and select the **Modular ADC core Intel FPGA IP.**

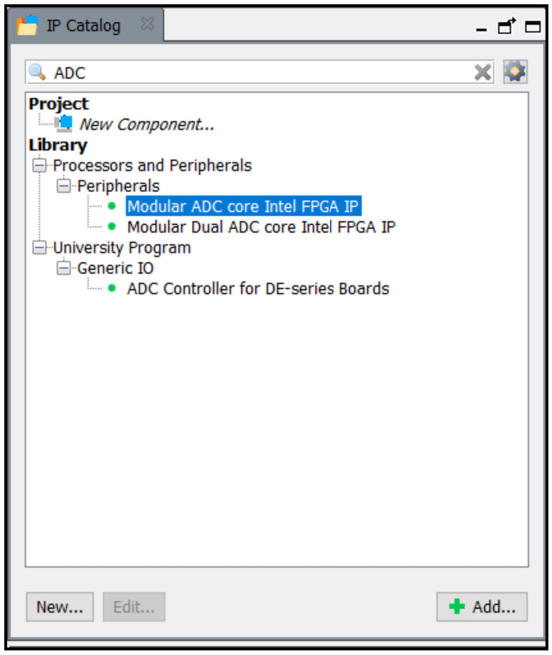


**Figure 2**

The settings for the core that you need are shown in **Figure 3 and 4.** In **Figure 4**, in the **Sequencer** tab, set the **Conversion Sequence Channels** to **CH0.** Then click finish in the window. Make sure to rename the A/D module. We named it **“AD_1.”**

The A/D has a sample rate of 1Mhz and has the option to select the input clock. The A/D will internally step down the clock to the 1MHz clock but we need to supply it with a 10MHz clock. To do this, we need to add a **PLL** from the same IP Catalog menu to drop the 12MHz clock to 10MHz. **See Figure 5.** It is the same process to setup the PLL as we have before in previous labs, except a little different. Don’t uncheck the defaults in **Optional Inputs** and **Lock Outputs** options under **[1]Parameter Settings>>Inputs/Lock**, see **figure 6**.

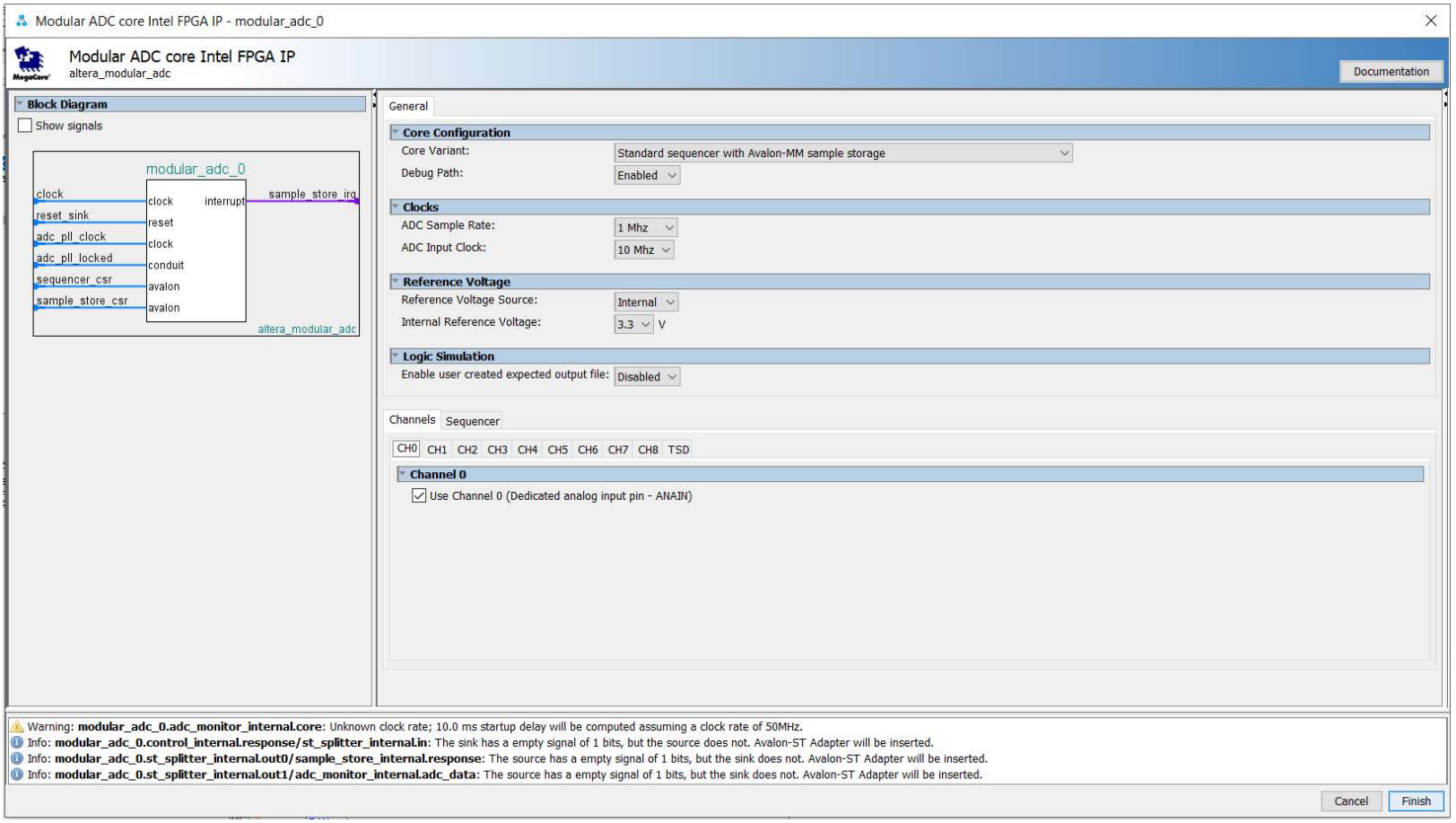


**Figure 3**

*These channels represent the amount of possible analog inputs that the A/D can take in and measure. For this lab we will only work with Channel 0.*

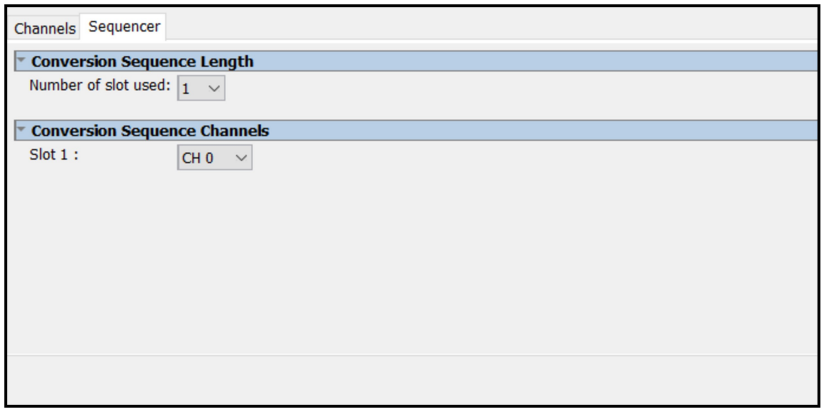


**Figure 4**

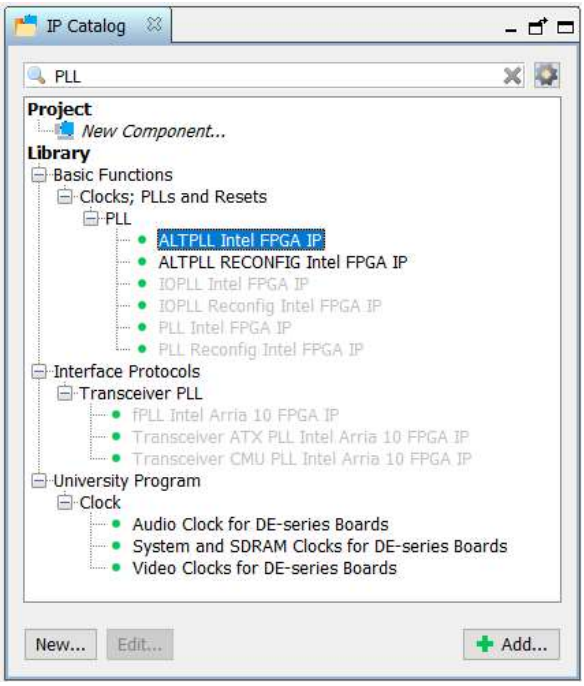


**Figure 5**

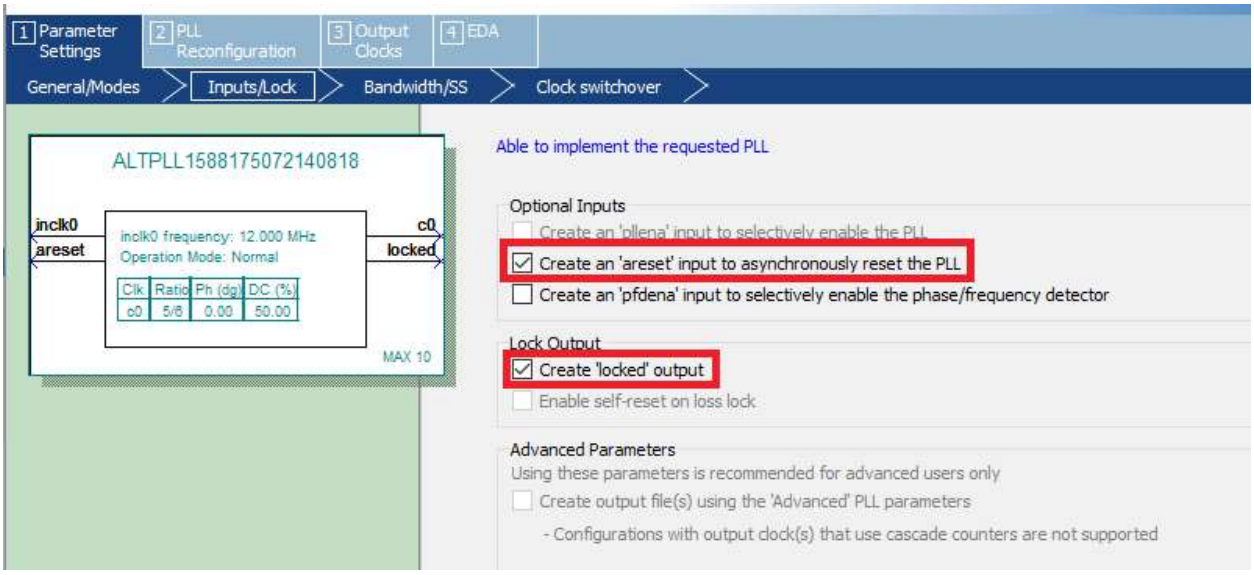


**Figure 6**

The last thing you need for the A/D core is the **JTAG Master Bridge.** Type in **"Jtag"** in the IP Catalog menu as shown in **Figure 7.** You don't have to adjust any setting in the Jtag menu. Just select finish when the window pops up.

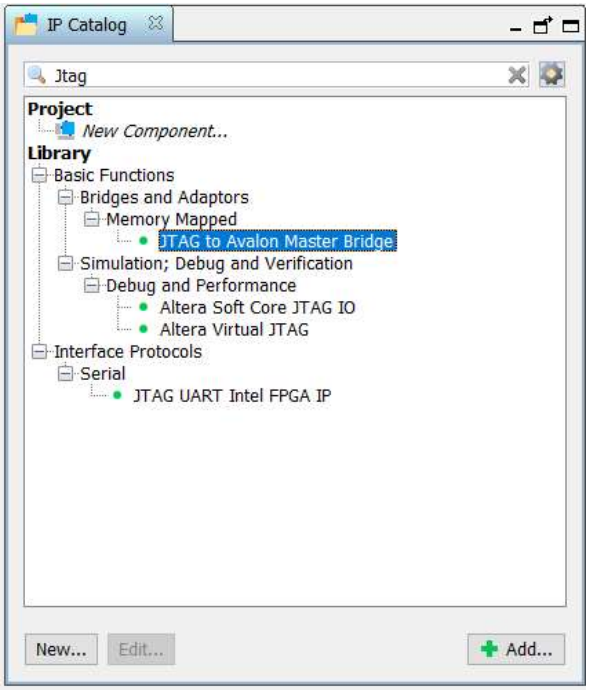


**Figure 7**

Use **Figure 7** as a guide to connect all modules with the Avalon bus connections. Once you have made these connections, set the base address of the **sequencer_csr** in the A/D module to 0x0000_0200. Then you will need to set all other base addresses by going to the top bar menu and select **System>>Assign Base Addresses.**

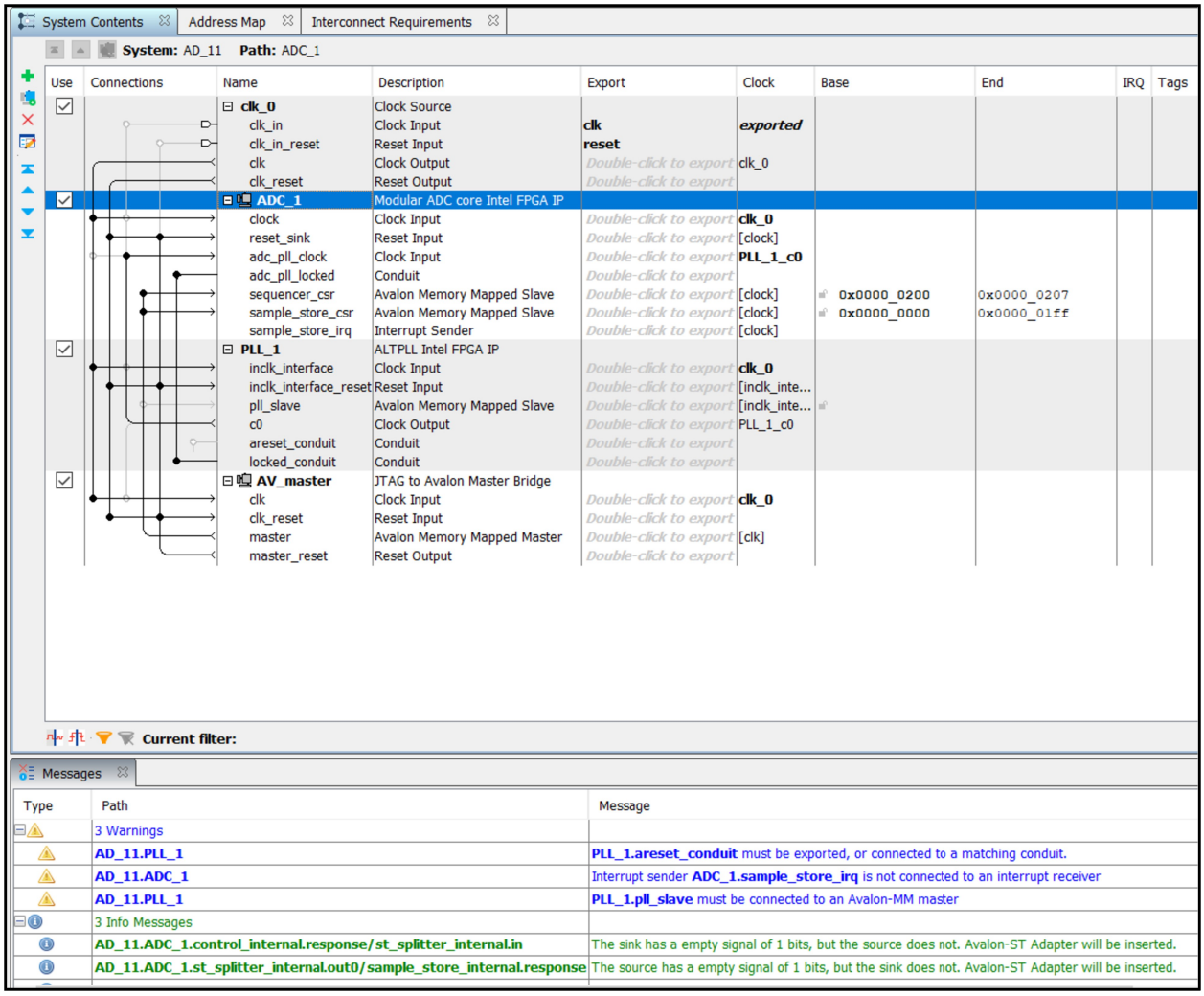


**Figure 8**

The three warnings in blue that you should see once you have set everything are fine and just leave those be. They will not affect the system at all. Now go ahead and **Generate the HDL** file and then click **Finish.**

Now that you have generated the A/D core fully. You will need to add it to your project. Go back to the Quartus main page. In the top bar menu click **Project>>Add/Remove Files in Project....** Then go to your directory/folder where your project is located and add the **.qsys** file to your project.

**Setting Pins and Programming the FPGA**

When you create the A/D core in the Platform designer, the HDL files and a **symbol file** is created for you to use. Now create a new **Block Diagram/Schematic** file. **Right click** on the open schematic and search for the A/D core. It should appear as a **.bsf** file. Use

the **Name:** search bar below the **Libraries** window and search for your file as shown in **Figure 9.**

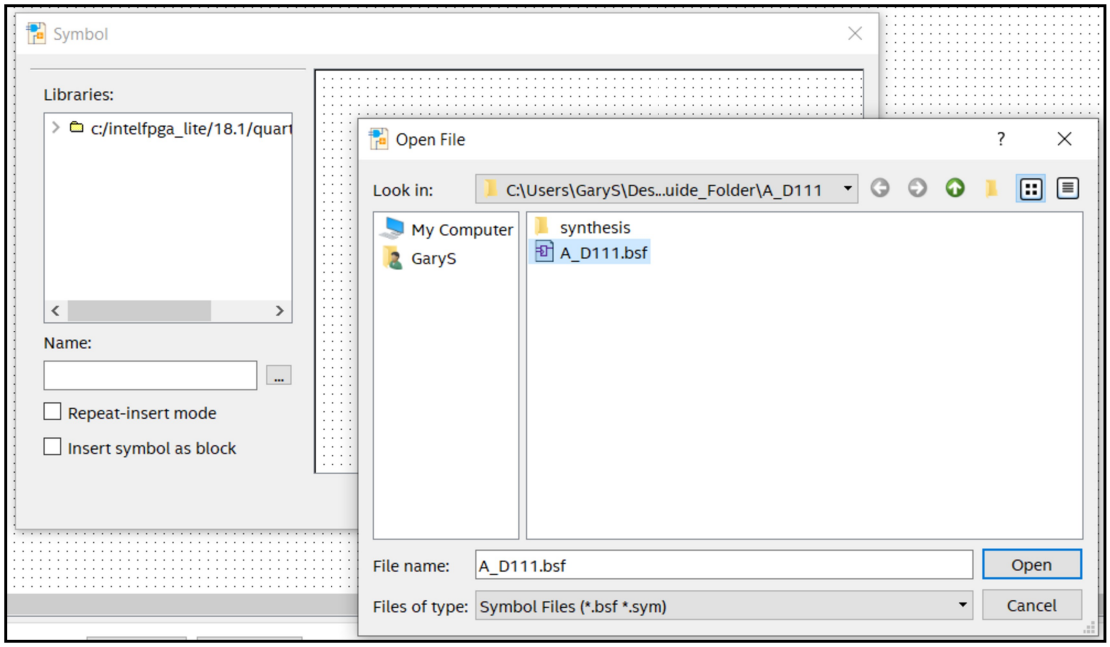


**Figure 9**

The A/D connections should look like **Figure 10.**

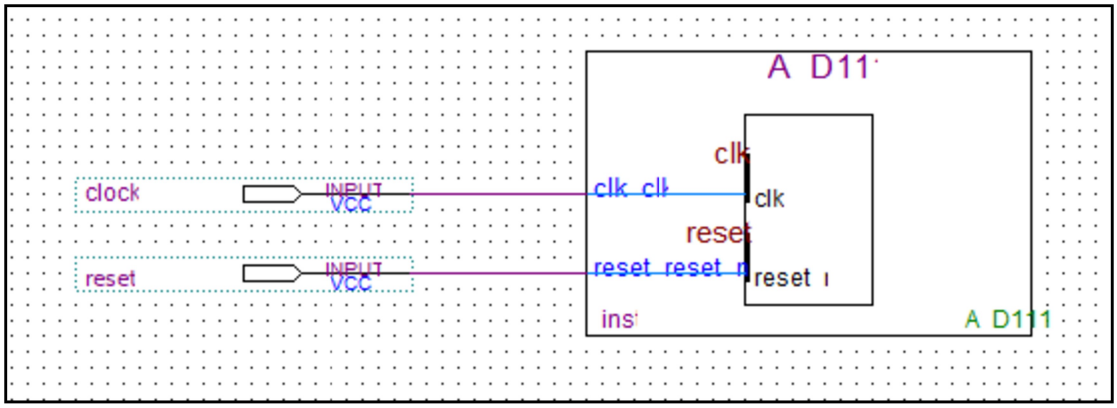


**Figure 10**

After setting up your inputs to the A/D, compile the project. Once the compilation is done and successful, go the **Pin Planner.** The clock pin will be set to **PIN_H6** and the reset will be set to **PIN_E6.** Once the pins are set, recompile the project.

Next plug in the Max1000 to your computer and program the **.sof** file to it. Because we will be using Quartus' built in System Console next to take measurements you will need to keep the FPGA plugged in, have a function generator and an oscilloscope near you,

and solder male header pins to the **AIN and GND.** Use **Figure 11** to find these pins on the Max1000. Since we are only using Channel 0, pin **AIN** is the dedicated analog pin and is only used for analog inputs. When we set the channel in the A/D core settings, Quartus automatically sets the right pin mapping. That is why you only had to set the 12MHz clock and the User Buttons as inputs.

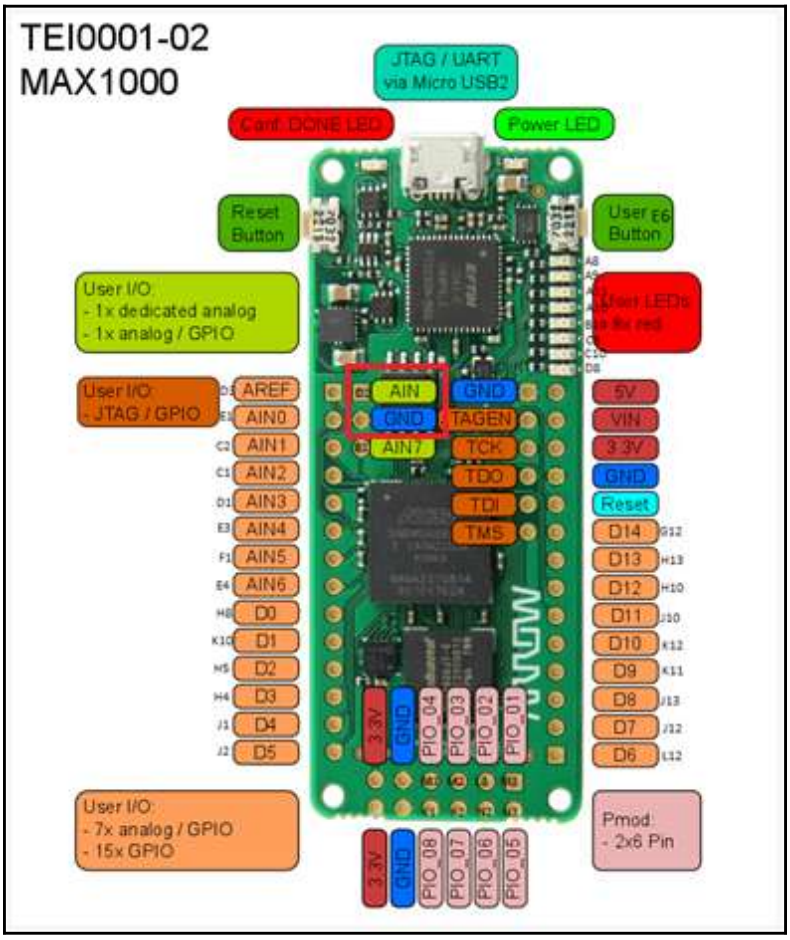


**Figure 11**

**Running the System Console with the A/D**

Now that you have the A/D programmed on the FPGA, go back to the **Platform Designer,** open up the A/D **.qsys** file. Then go to the top bar menu and click on **Tools>>System Console.** You should see the window as shown in **Figure 12.**

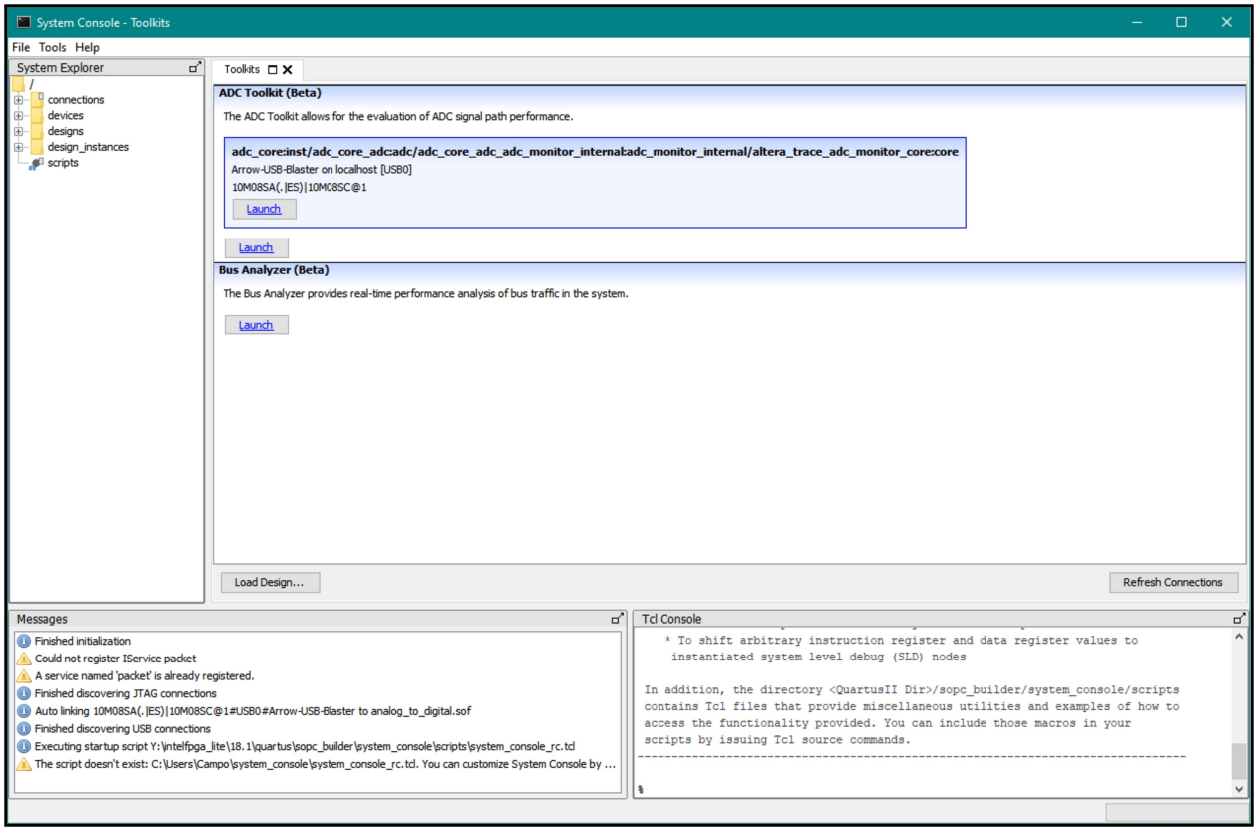


**Figure 12**

Select **Launch** in the **ADC Toolkit blue box** and you should see a window similar to **Figure 13.**

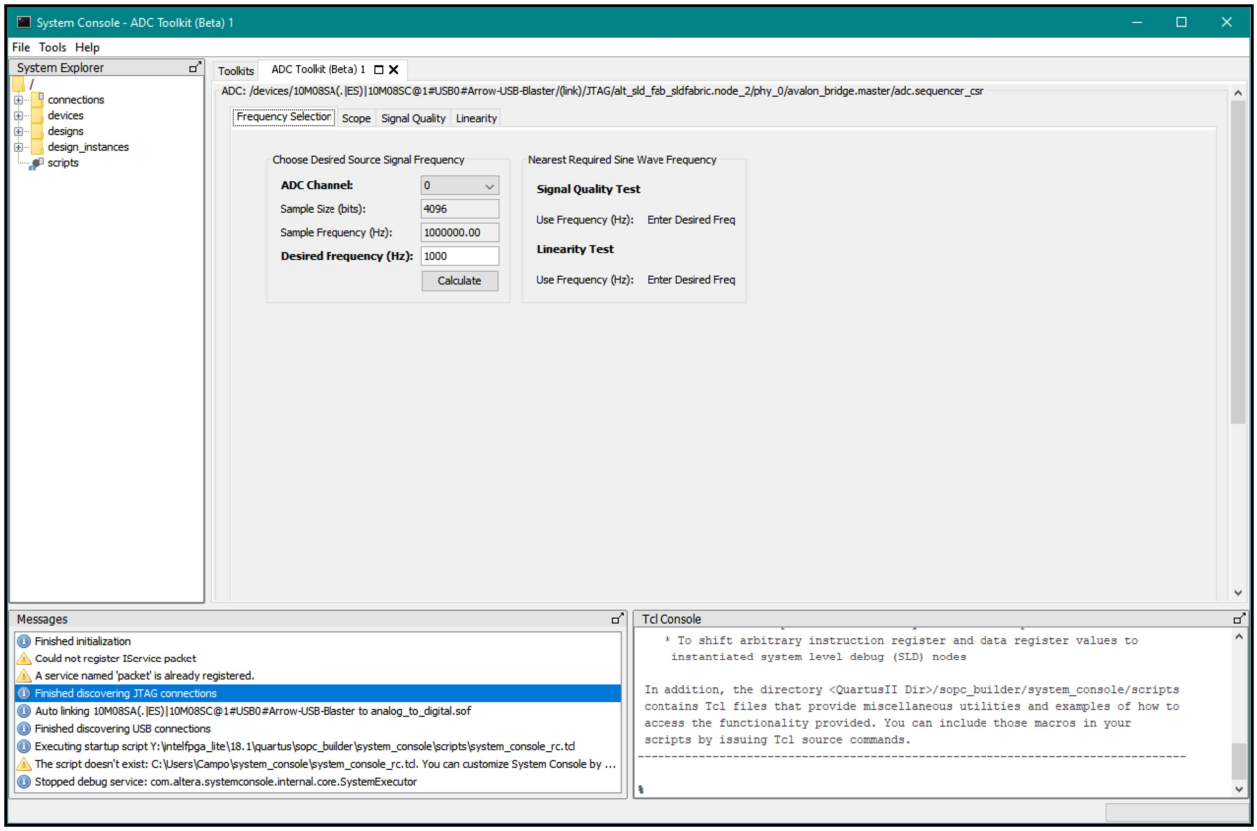


**Figure 13**

Use the options to select what frequency you want to read in on the **AIN** pin, set the sample rate if it doesn't show 1MHz, turn on your function generator, set the signal to a **1kHz square wave with a max peak to peak value of no more than 3.63 V. Make sure to measure this output first on an oscilloscope. If you send a signal that is greater than 3.63 volts you will damage the Max1000.** Once you set up your waveform, connect the signal to the **AIN** pin. **Then click Calculate.** Next click on the **Scope tab** and click **Run** as shown in **Figure 14.** Now you should see the waveform on your screen in real time. The waveform shape can be of your choosing but for this guide, a square wave was used as shown in **Figure 15.**

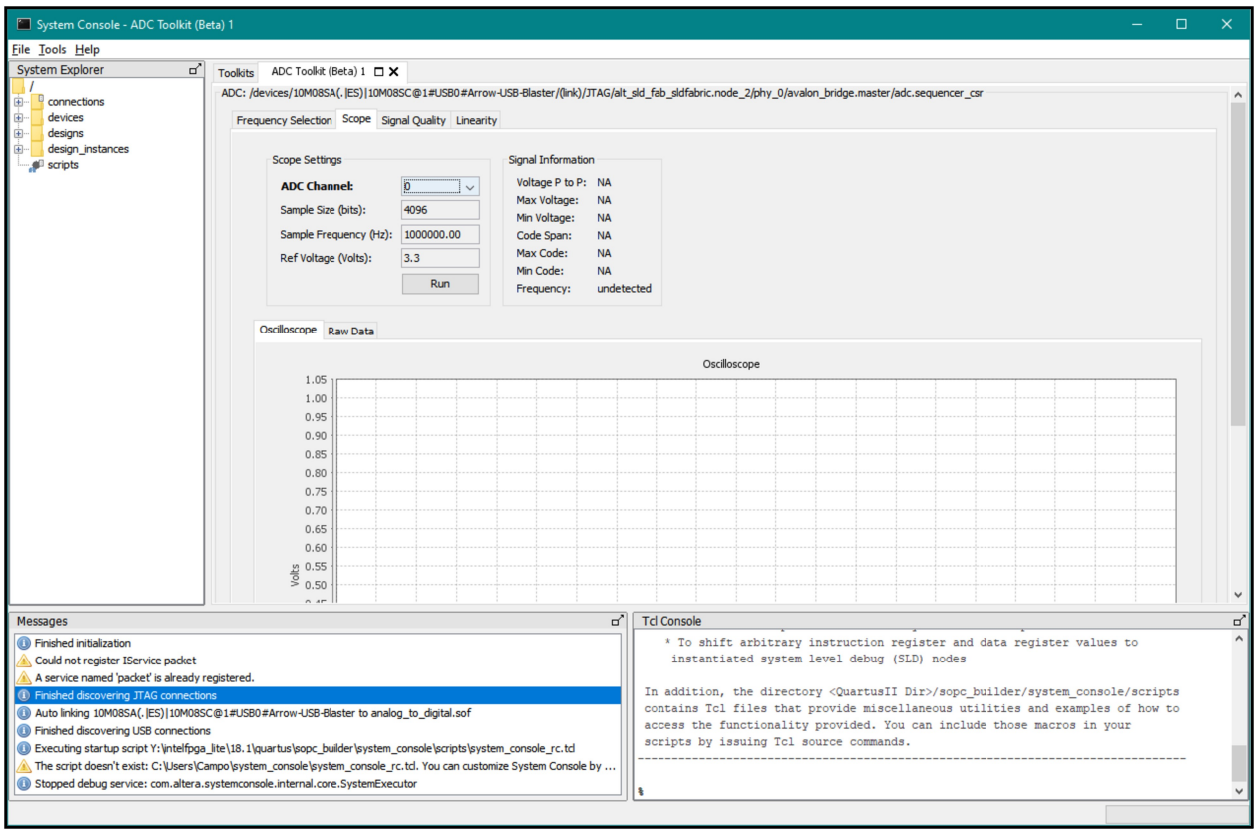


**Figure 14**

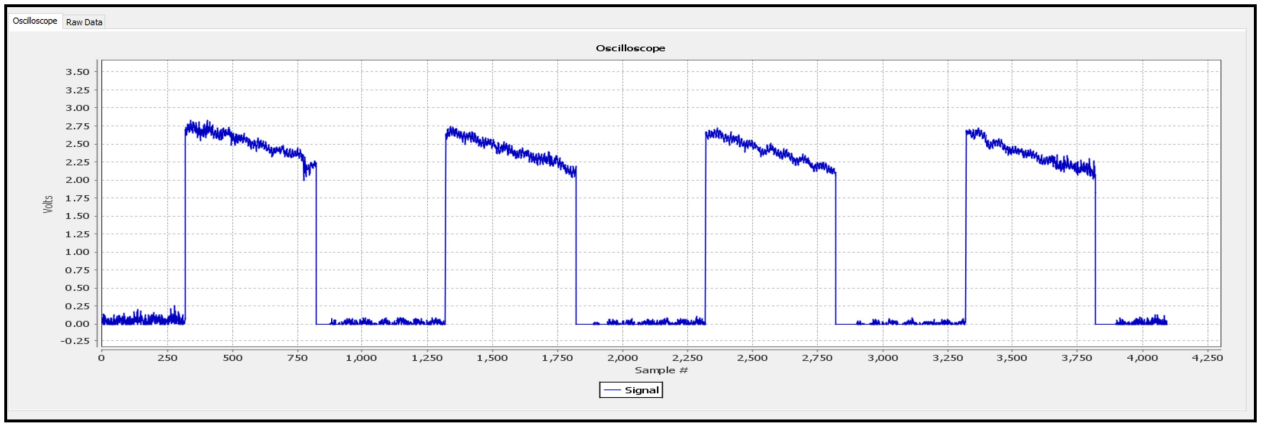


**Figure 15**

## Exporting Raw Data

So the ADC Toolkit offers the option to export the raw data it pulls from the input into a **.csv file.** To do this, stop your running the scope in the Toolkit first. Make sure to always do this or the data will not be correct. Next, Click on the **Raw Data tab** and you should see a window similar to **Figure 16.**

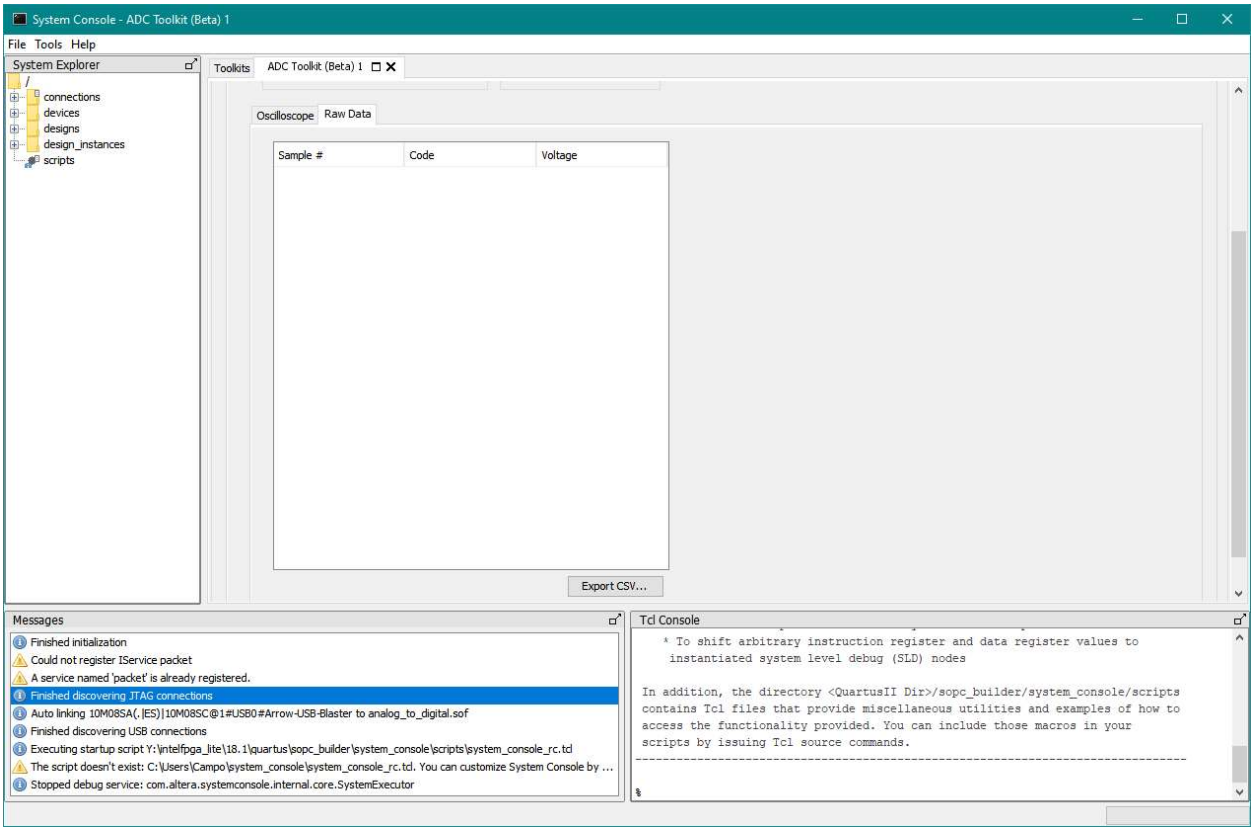


**Figure 16**

Unlike **Figure 16,** you should see samples with voltage values. All you have to do is click **Export CSV** and Quartus will let you save the file anywhere. Then you can open the **.csv** file and parse out the data to your liking.

*We would recommend playing with different inputs such as adjusting the waveform or the frequency, as long as you watch the analog input voltages. You basically now have a $30 oscilloscope which is really cool.*