Be Boulder.

DE10-Lite ADC Guide

Version 2.1 **09/14/2020** **ECEN 5863**

Table of Contents

OVERVIEW

Building on the PWM FPGA project, complete the design by using the ADC to read the PWM generated voltage and report the results on the 7-segment LEDs.

Module Summary:
The Hardware labs are based on completing the initial portion of the design of a simple voltmeter.
In **Section 1** you will prepare for the project acquiring files and other resources.
In **Section 2** you will examine the system design.
In **Section 3** you will learn how to Create a system design using Quartus Prime and Qsys.
In Section 4 you will place and route the design.
In Section 5 you will create a programming file, program the FPGA, and then test the hardware design.

Goal of this Hardware Lab:
The goal of this hardware lab is develop a working design, using most aspects of the Quartus Prime Design Flow, including Qsys.

Caution:

Do not continue until you have read the following:

The names that this document directs you to choose for files, components, and other objects in this exercise **must be spelled *exactly* as directed.**

1. Getting Started

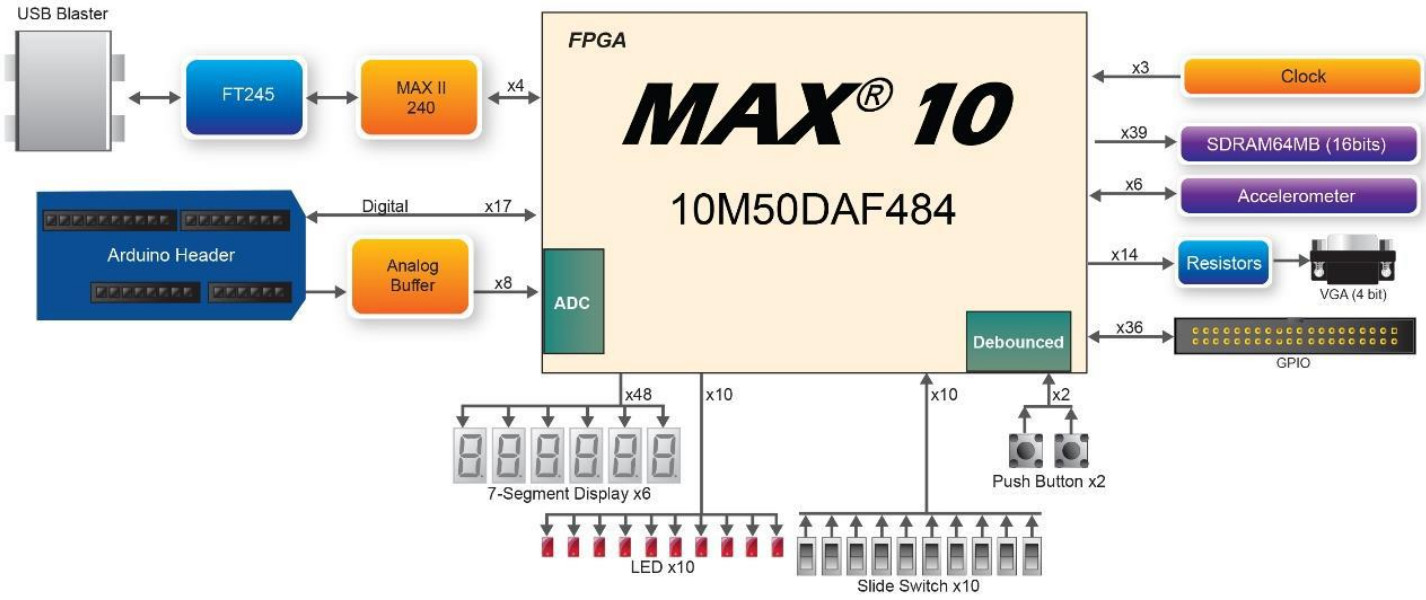
Your first objective is to ensure that you have all of the items needed and to install the tools so that you are ready to create and run your design.

List of Required Items:

- Altera Quartus Prime Version 16.1 FPGA Development Software Tool
- **Module Design Files: ADC_connect.vhd, SEG7_LUT_6.v, etc.**
- Terasic DE10-Lite Development Kit

Before continuing with this Module, ensure that the Altera tools and drivers have been installed.

1.1. DE10-Lite Development Kit



1.2. Extract the Lab Files.

- Create a folder **C:\AlteraPrj\DE10liteADC** on your PC.
- Copy ADC.zip to this directory
- Extract to \ADC

Excellent!!

You have just completed all the setup and installation requirements and are now ready to examine the system-level design.

2. System Design

Section Objective
In this Section you will review the architecture of the design that will be created in Quartus Prime with the DE10-Lite Kit as the target. Typically, a design starts with system requirements. These system requirements become inputs to the system definition.

After completing this Module you will be familiar with the following:

• Using Qsys to create an IP block for use in your design

• Implementing a MAX 10 ADC subsystem.

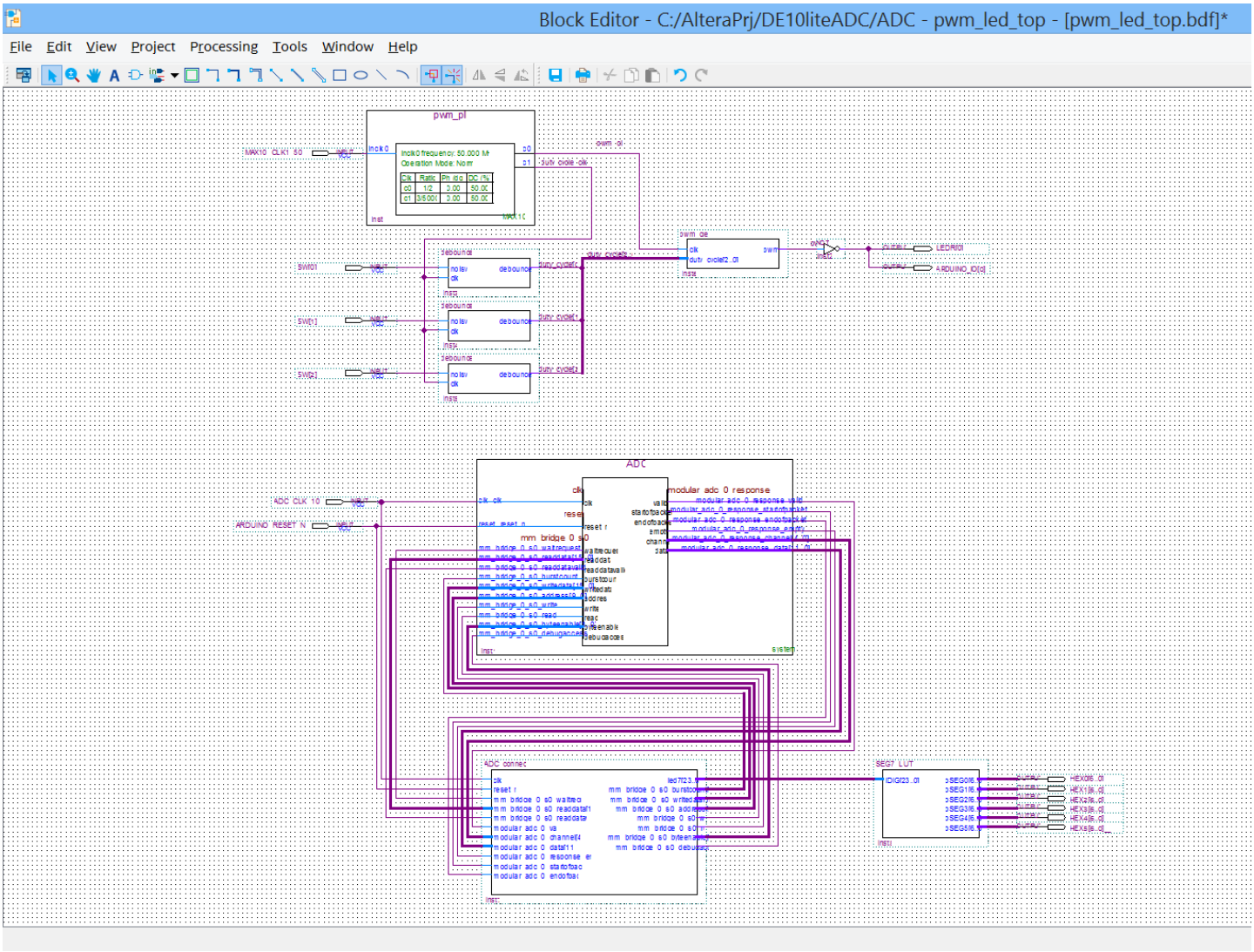
• Incorporating the MAX 10 ADC subsystem into the top level system design, by creating a block symbol from HDL source.

• Implementing the design by running place and route, creating pin assignments and a programming file for the MAX10 silicon, and then analyzing and testing the design.

2.1. System Requirements

During this exercise, you will follow a step-by-step guide to create a simple design. To do this you will create a design in Qsys, the Altera System Design Tool, which includes a PLL block, the ADC module, a JTAG Avalon bus master and Avalon bus bridge. You will then create a block symbol of this Qsys system, and add it to your PWM system design schematic. After connecting the top level blocks together, the design will be compiled and downloaded to the target board for testing. You may be able to use the ADC toolkit to view waveforms as part of the test. The inputs are a System Clock at 50 MHz, 3 toggle switch inputs to encode the PWM duty cycle, a 10 MHz ADC clock, and the ADC input from a pin on the Arduino Analog Connector. The output is an LED, the 7-segment LED displays, and a pin on the Arduino GPIO Connector.

2.1.1 System Block Diagram



The design above includes the previous PWM circuitry from the first part of the module, and adds an ADC module with channel sequencer, and a control module that connects to the ADC, starts the sampling, and generates outputs to a 7-segment LED driver.

Components of MAX10 Device Used

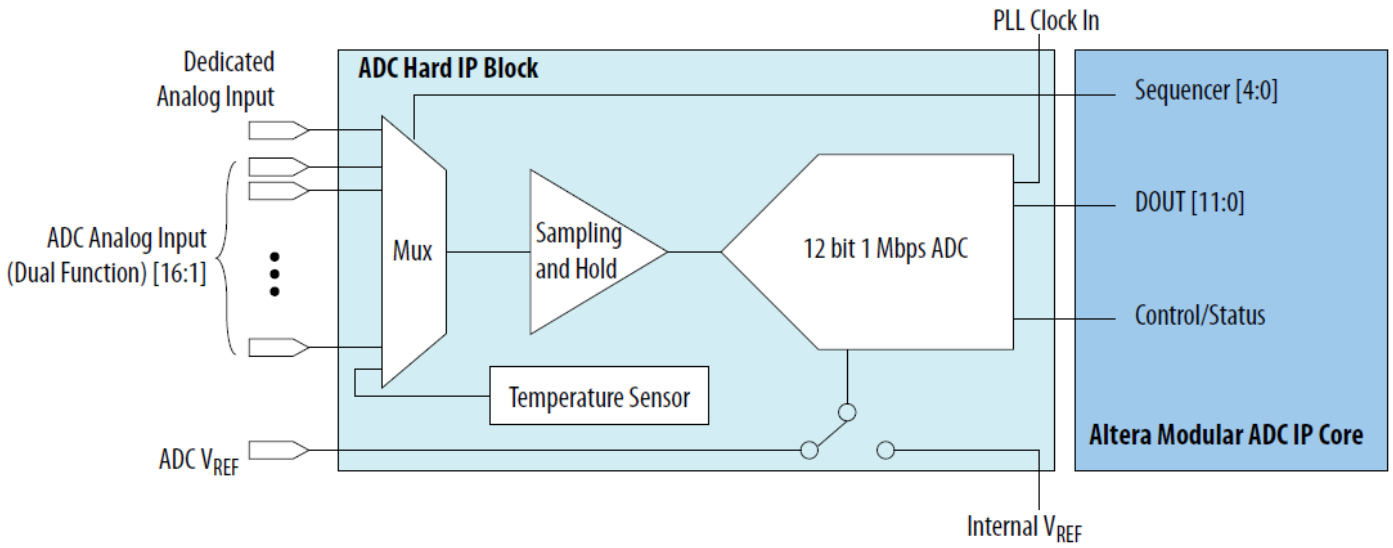
This tutorial uses the MAX10 pll block, ADC module, and logic fabric.

2.2. MAX10 ADC

The MAX 10 ADC is a successive approximation register (SAR) ADC that converts one analog sample in one clock cycle.

Each ADC block supports one dedicated analog input pin and up to 16 channels of dual function pins.

You can use the built-in temperature sensing diode (TSD) to perform on-chip temperature measurement.



Good Work!!

You have just completed the examination of the system-level design

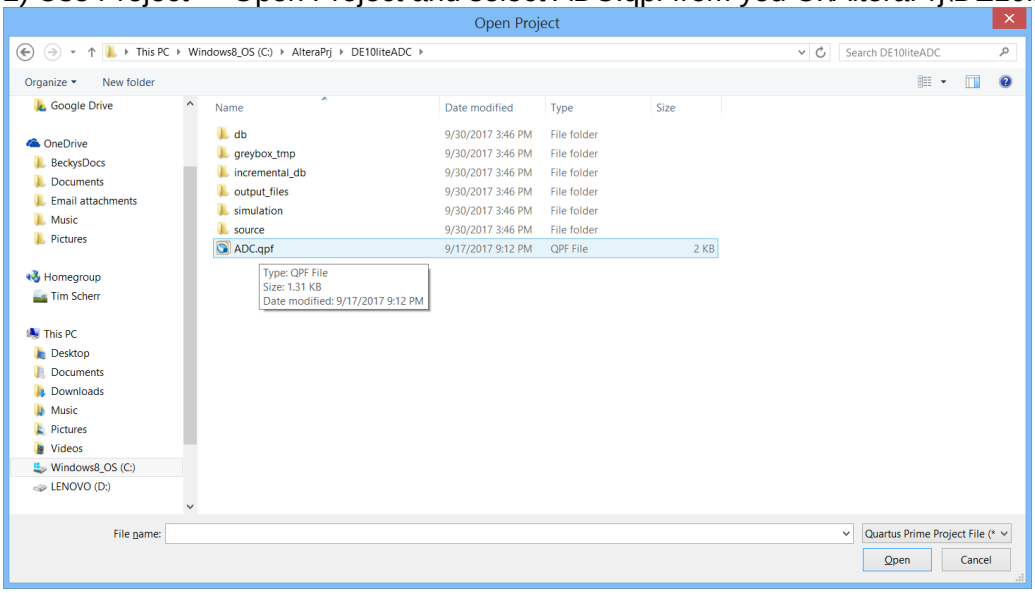
3. Creating the Design

In this section you will create the ADC module design using Quartus Prime. Some source files have been provided in the project files, as the design is a combination of HDL files, IP cores, and Macro Blocks.

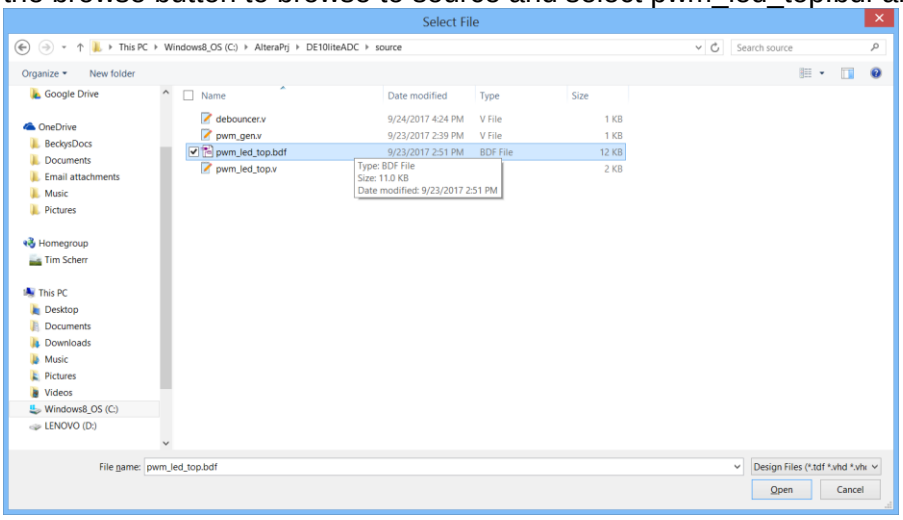
3.1. Create the Quartus Project

1) Launch Quartus Prime 16.1 (64-bit)

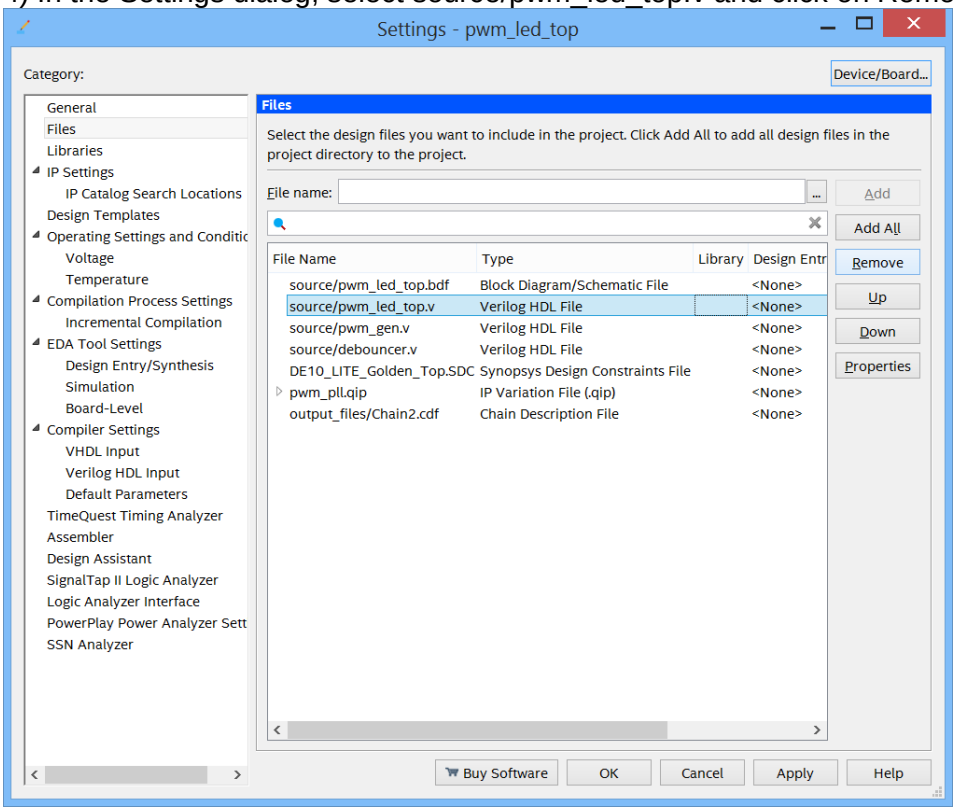
2) Use Project -> Open Project and select ADC.qpf from you C:\AlteraPrj\DE10liteADC directory.



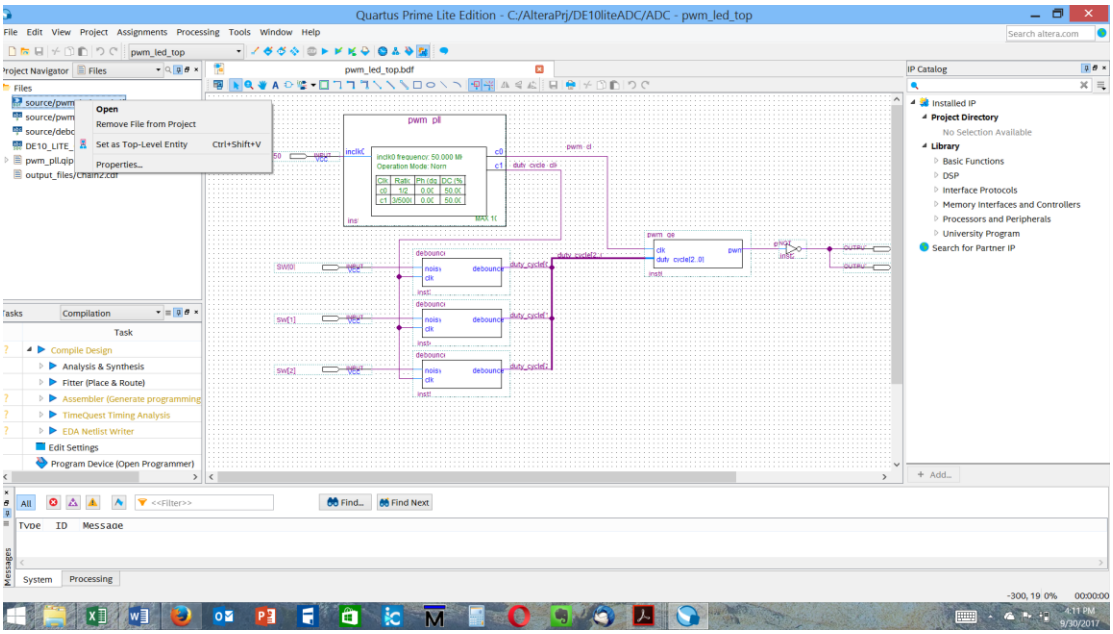
3) You may notice the top level is still pwm_led_top. This is OK, as you will be adding the ADC module to this design. If you double click on this and see pwm_led_top.v, then you will need to add the pwm_led_top.bdf and make it the top level entity in the design. To do this, select Add/Remove Files from the Project menu, and use the browse button to browse to source and select pwm_led_top.bdf and Click Open.



4) In the Settings dialog, select source/pwm_led_top.v and click on Remove, and then OK.

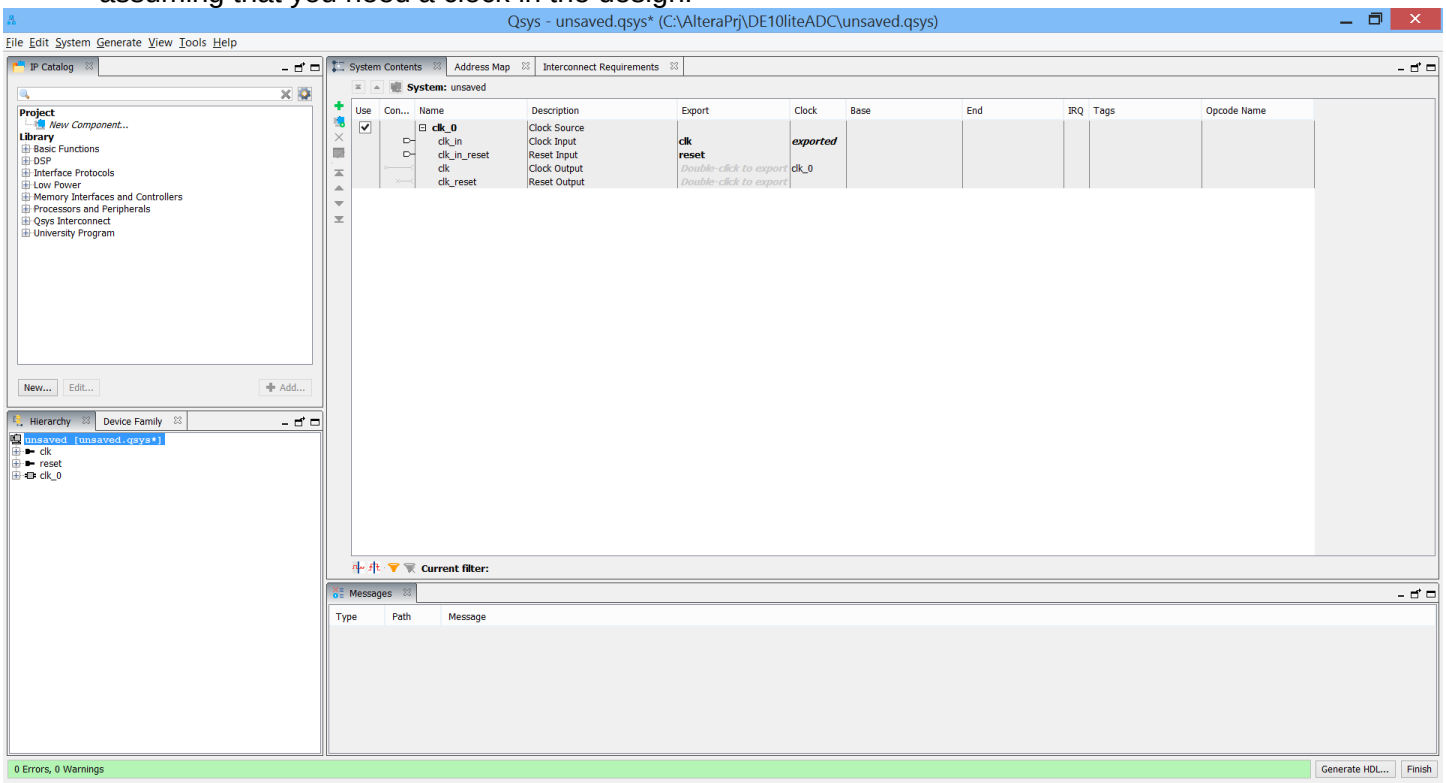


5) In the Project Navigator on the files tab, right click on source/pwm_led_top.bdf and select Set as Top-Level Entity. Double click on this file to see the block diagram of the design.

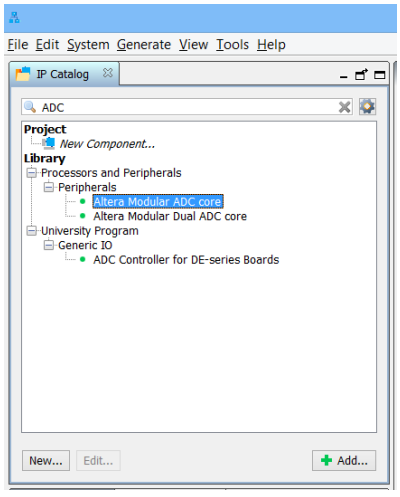


3.2. Create the Qsys System

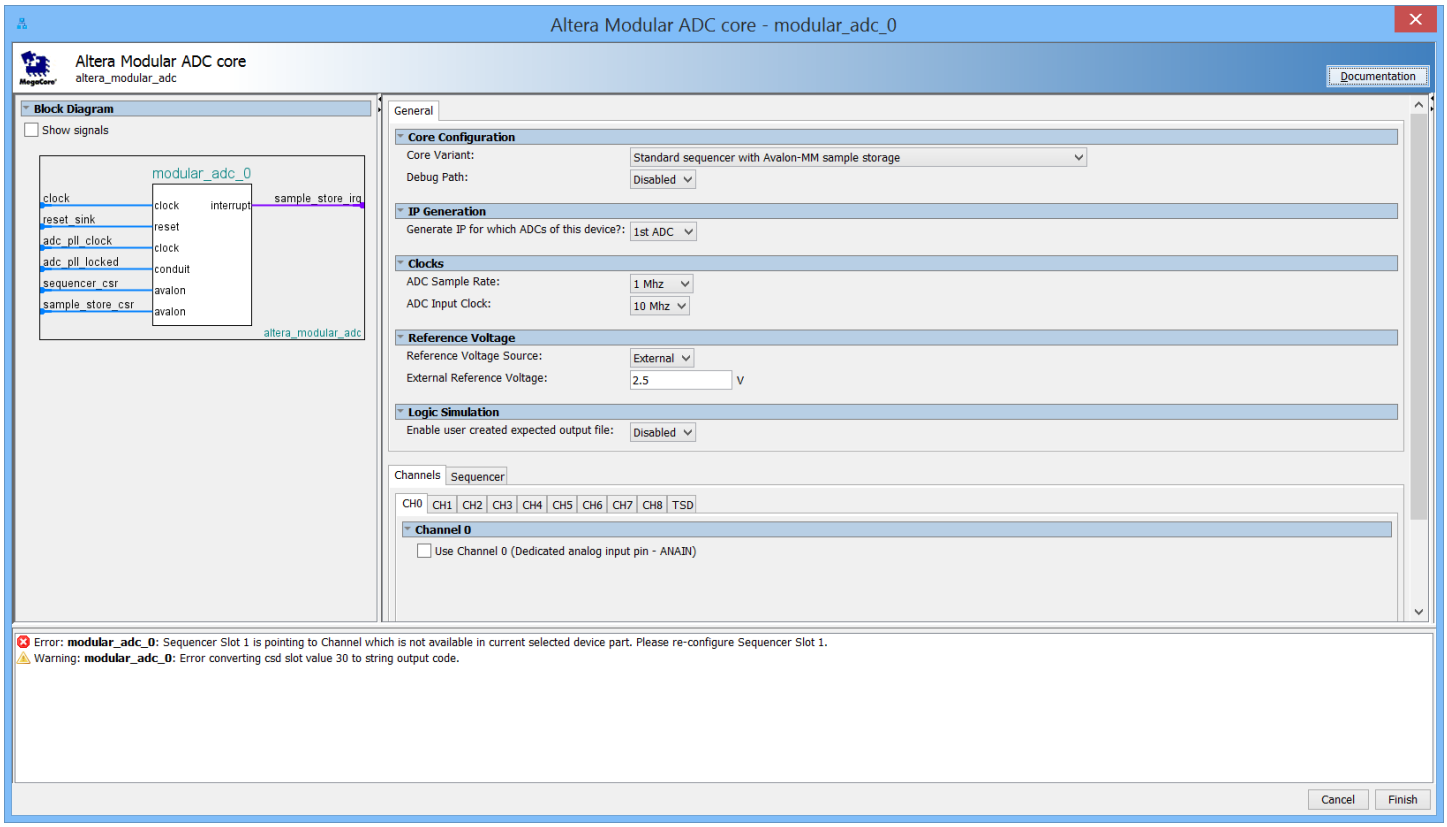
1) From the Tools menu, select Qsys. After a moment or two, the Qsys design tool will open. It begins assuming that you need a clock in the design.



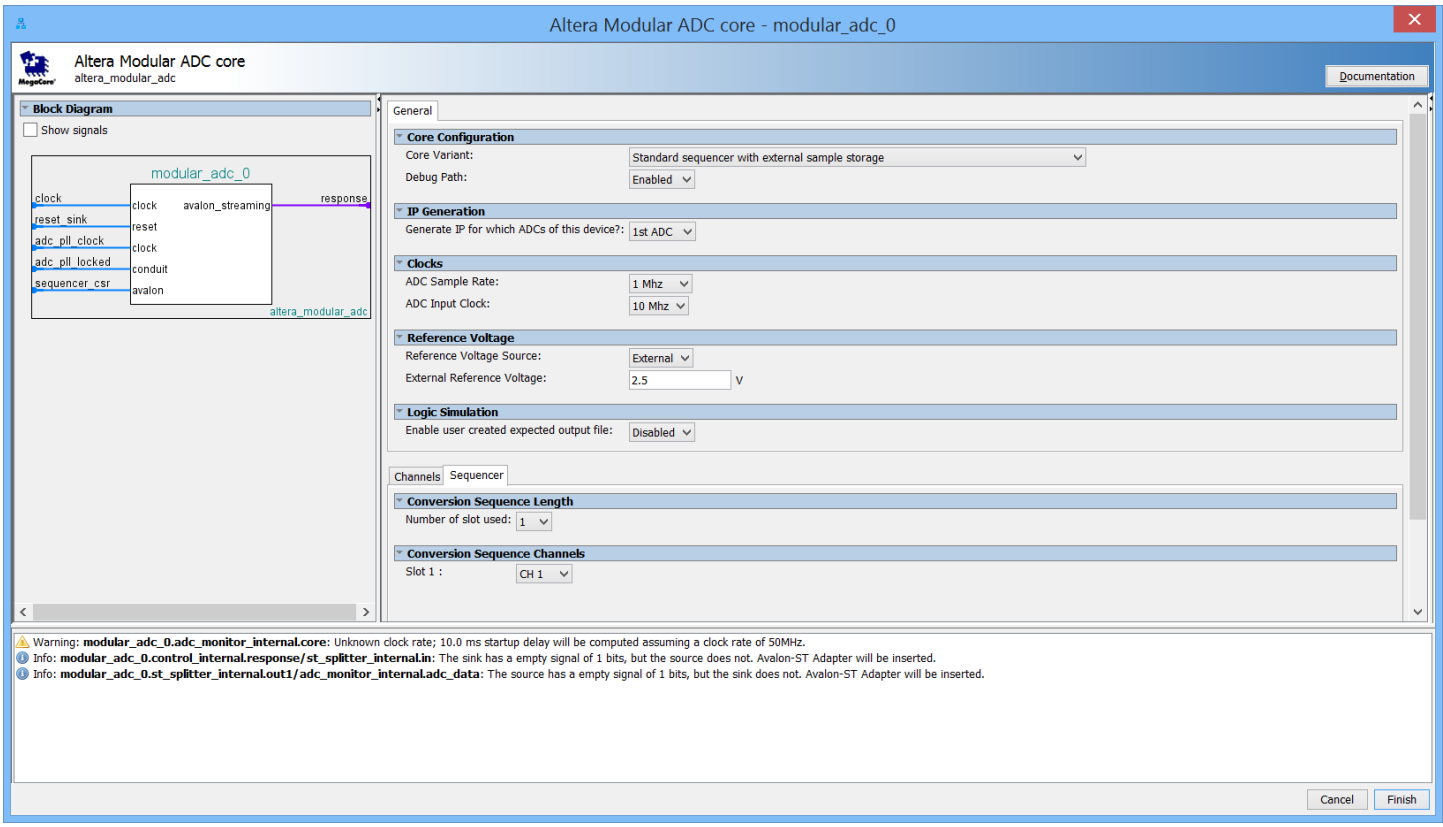
2) In the IP catalog in the upper left hand corner, type ADC in the search box. Select the Altera Modular ADC core and then click add in the lower right corner of this window to add the ADC module.



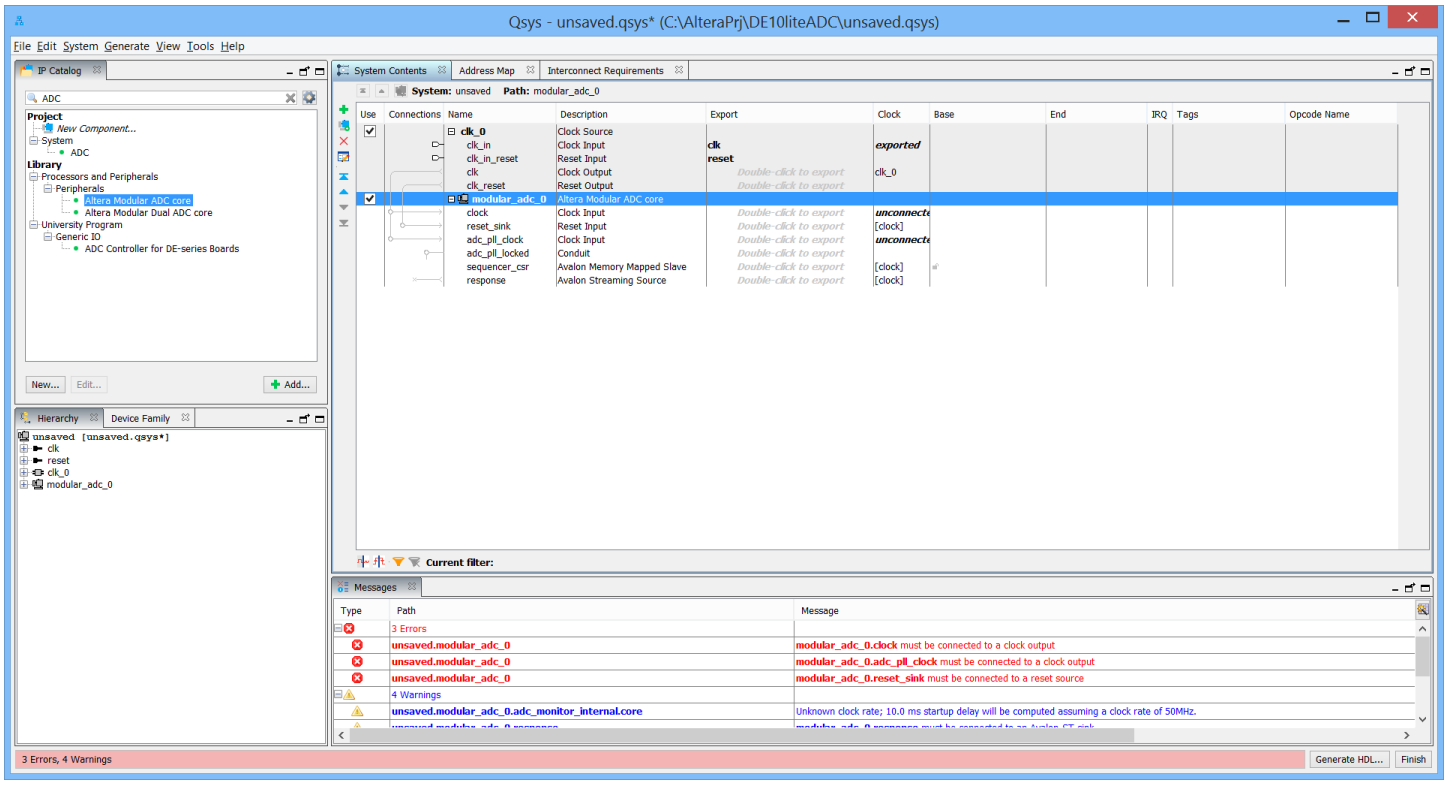
The ADC core configuration window should appear:



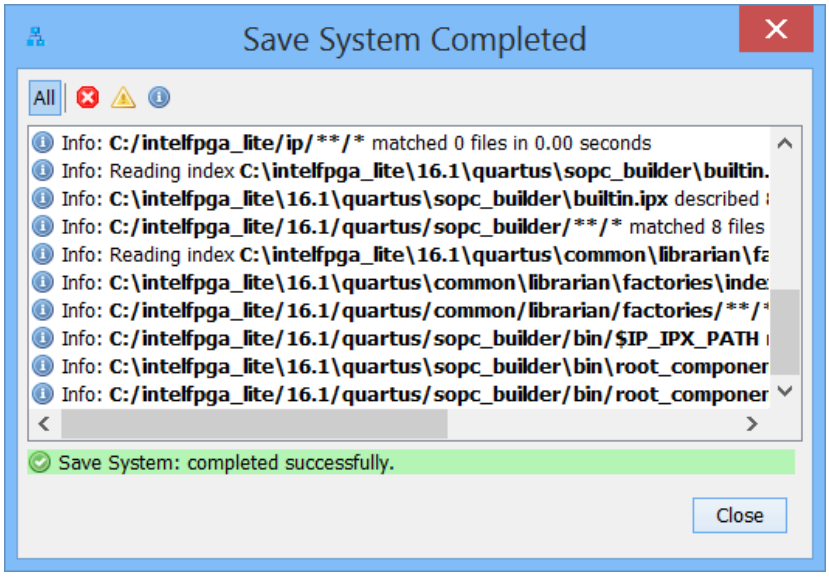
2) For the Core Configuration, select the Standard sequencer with external sample storage (we need to be able to get the sample data in the top level) and Enable the Debug Path. Accept the other defaults. In Channels, select CH1 and check the box to use Channel 1 In the Sequencer, set the number of slots used to 1, and set Slot 1 to CH 1. Click Finish.



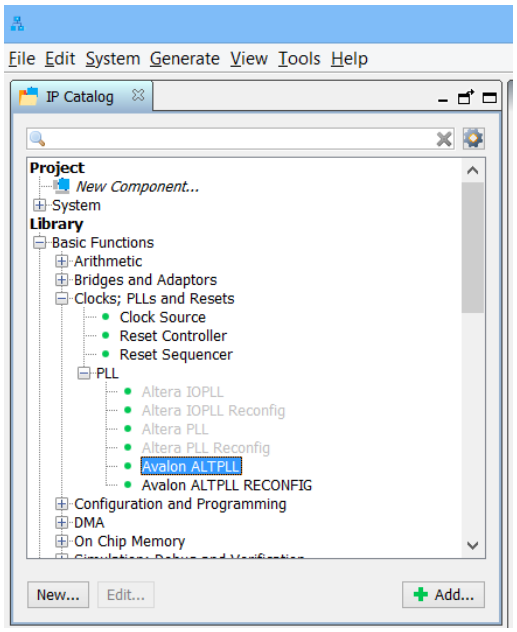
You should see an addition to the Qsys system like this:



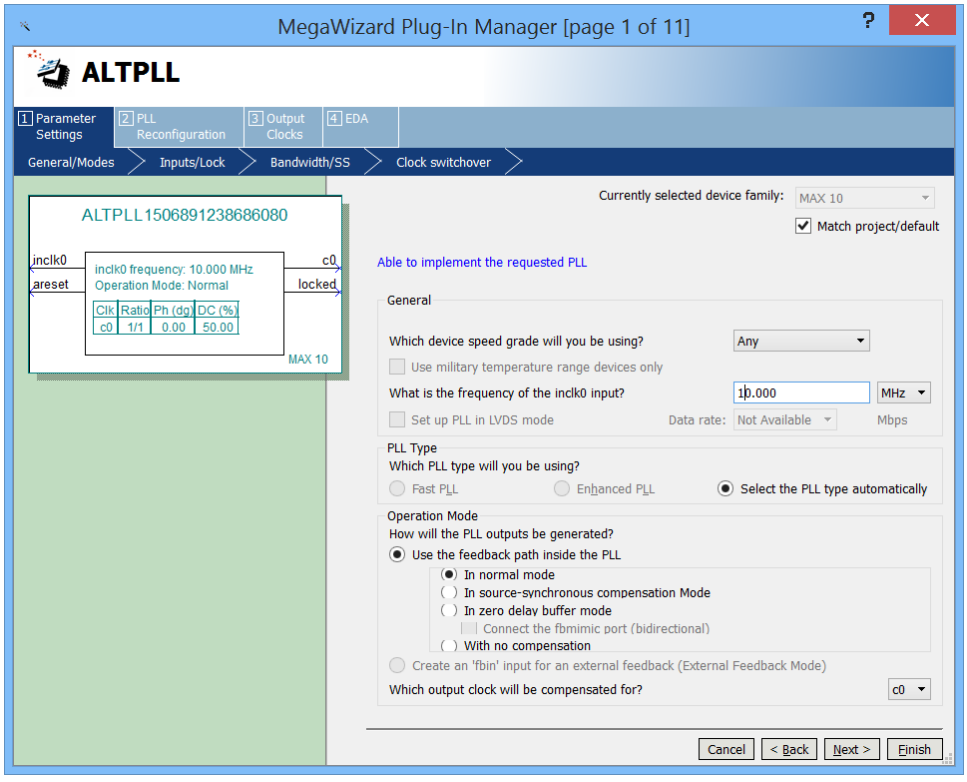
Note that there are errors. We will address these as we continue the design. At this point, we might want to save our work. Click File and Save As, save the qsys file as ADC.qsys.



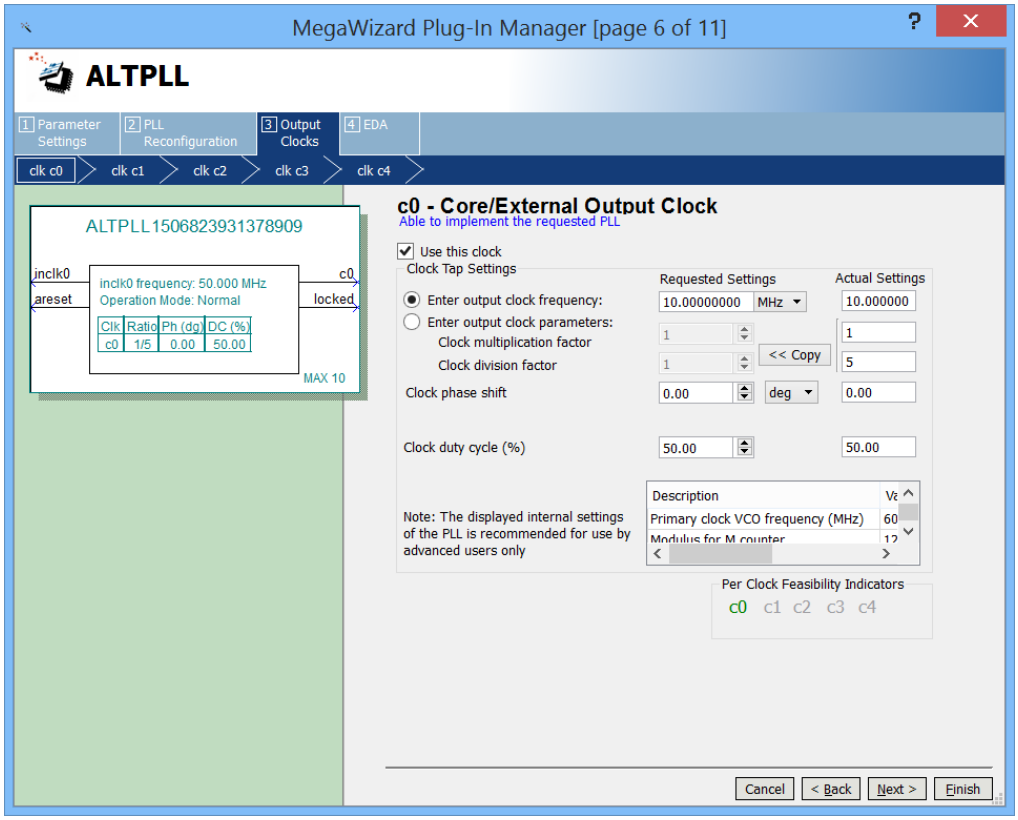
4) Next you need to add an alt_pll to provide a clock for the ADC module. In the IP catalog, click the x on the search box if it still has the ADC present. Under Basic Functions, then Clocks, then PLL select Avalon ALTPLL:



Double click on the ALTPLL, the ALTPLL MegaWizard Plug-In Manager should appear. Enter 10.000 MHz for the inclk0 input frequency and accept all other defaults:

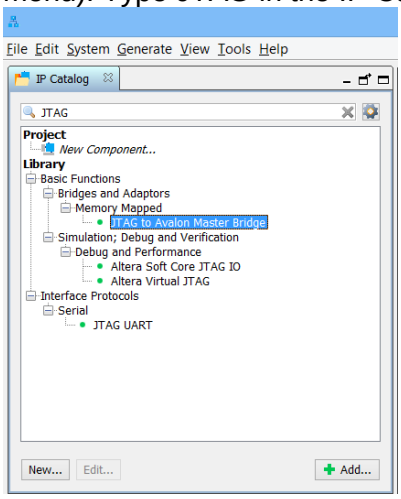


5) Next click on the Output Clocks tab and set the output clock frequency to 10 MHz. This will be used to drive the ADC module.

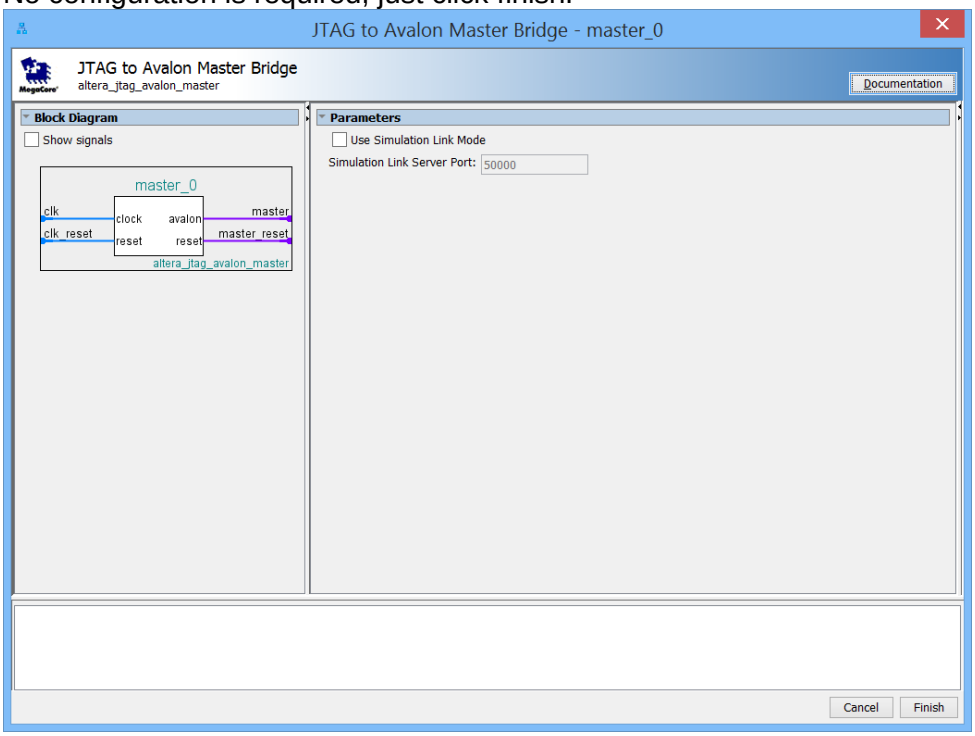


Click **Finish.**

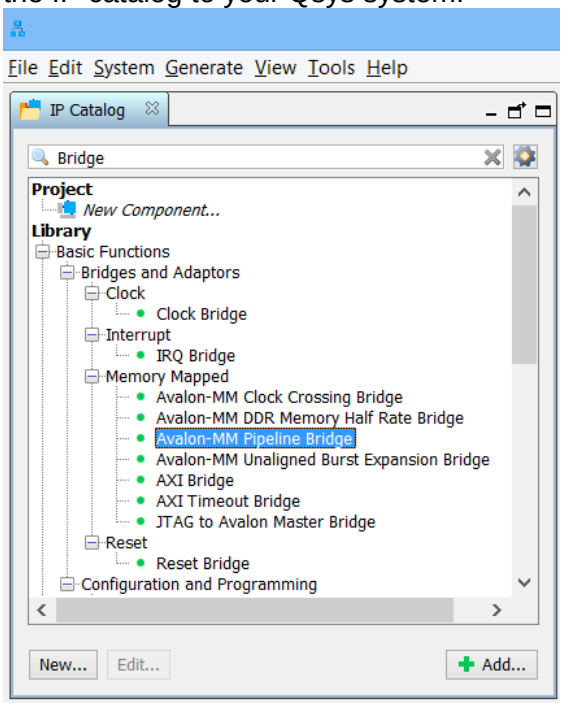
6) To use the built-in debugging features, you need to add a JTAG to Avalon Master Bridge. This gives you a master interface to talk through with the "System Console" tool which is a part of the Quartus installation (Tools menu). Type JTAG in the IP Catalog search box. Select and add the JTAG to Avalon Master Bridge:



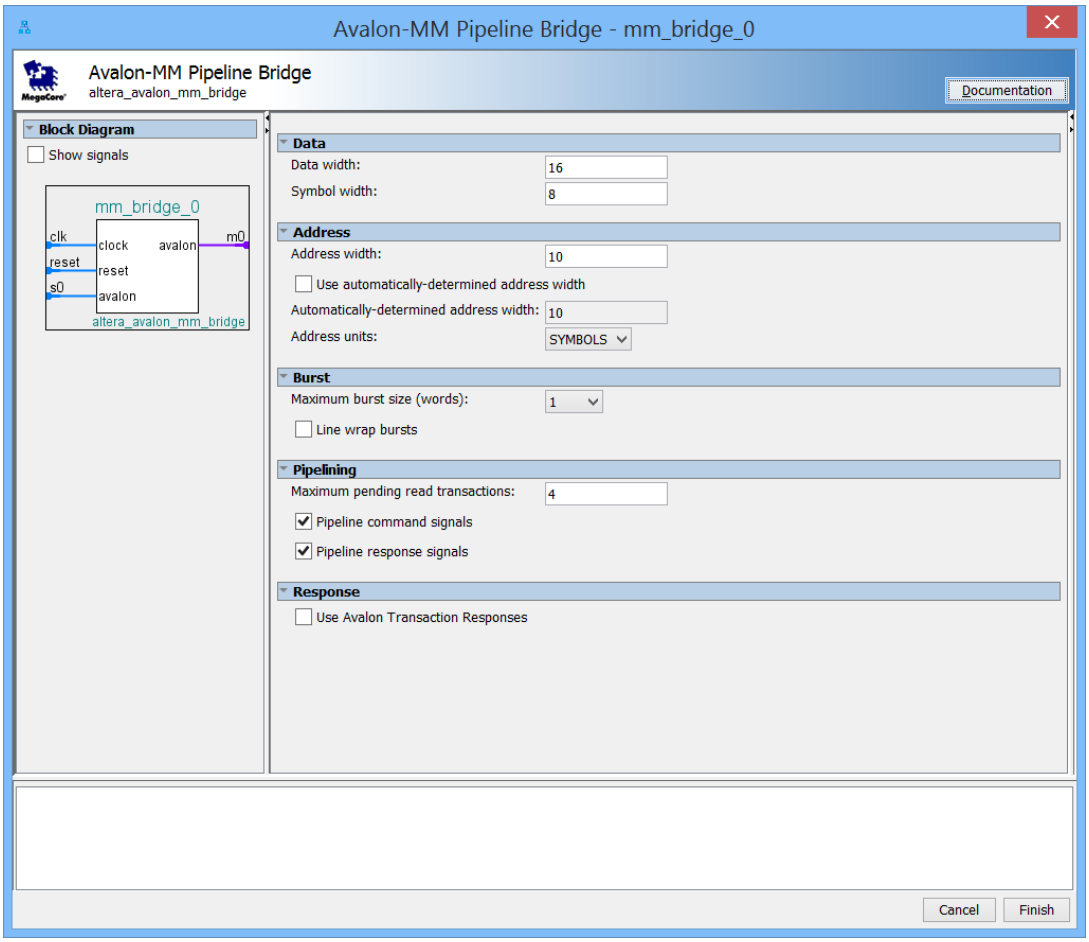
No configuration is required, just click finish.



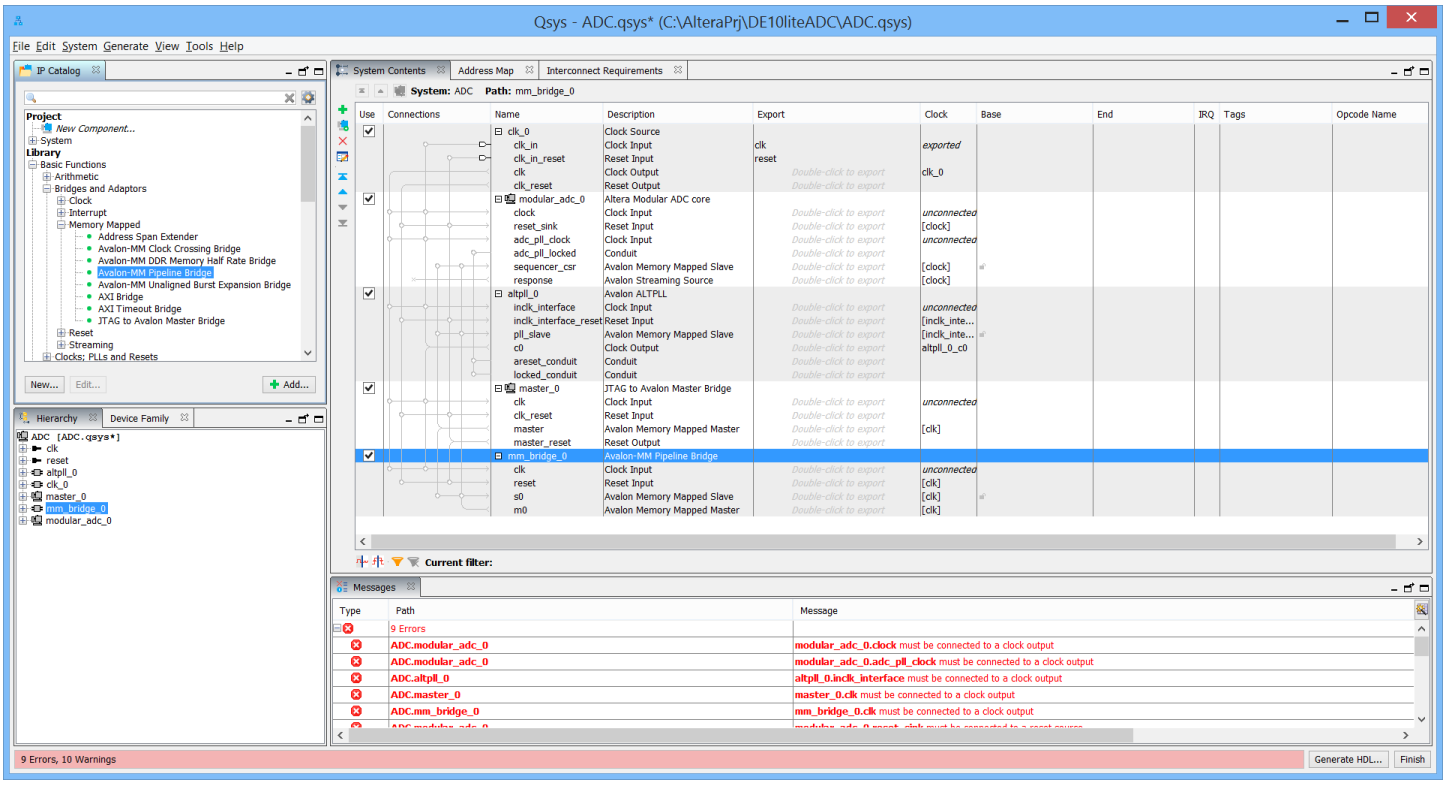
7) To communicate to the rest of the design elements, you need to add an “Avalon-MM Pipeline Bridge” from the IP catalog to your Qsys system:



Set the data width to 16 and hit Finish:

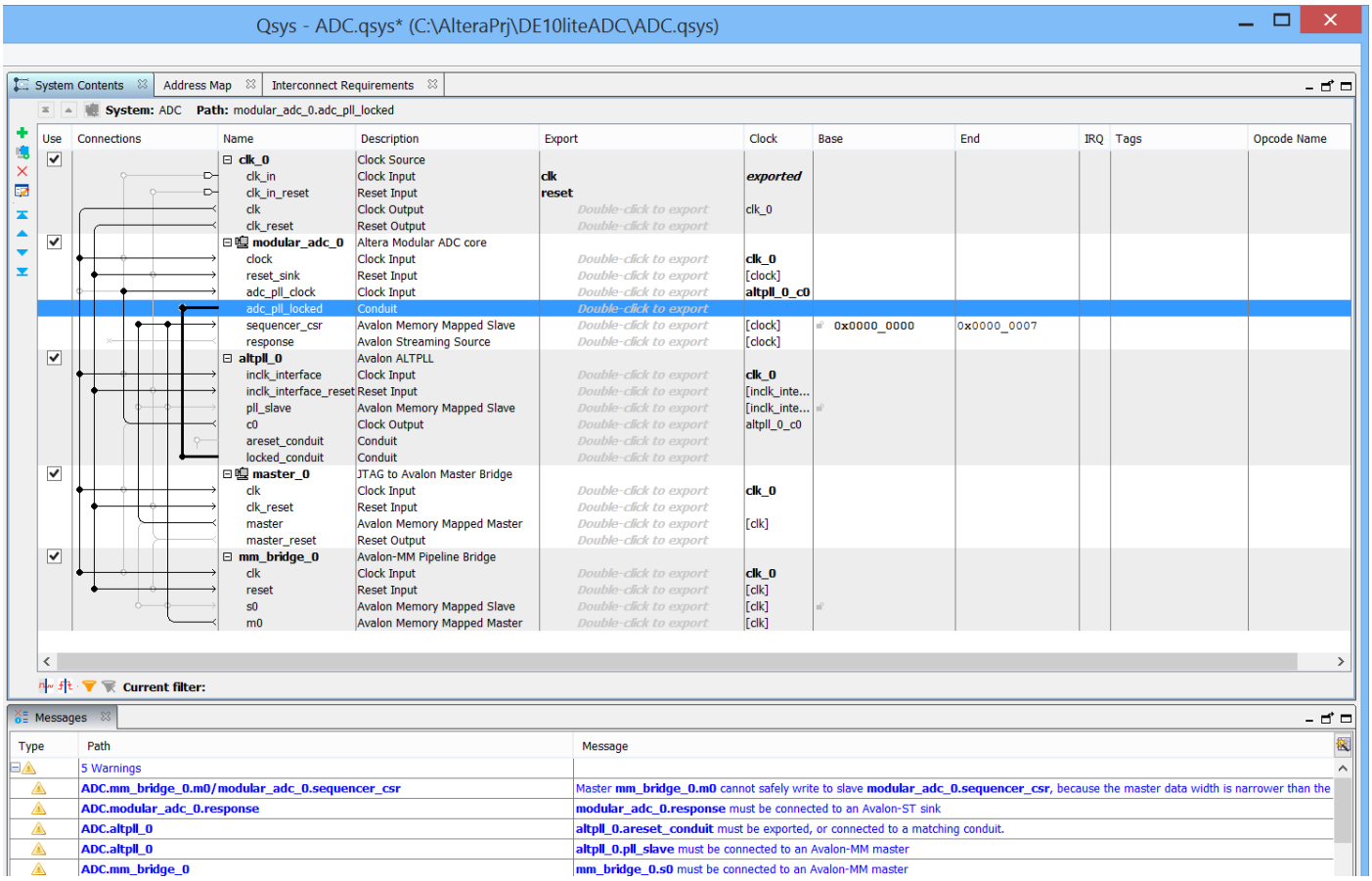


This finishes the component insertion portion of the Qsys design. You will likely see 9 errors listed:

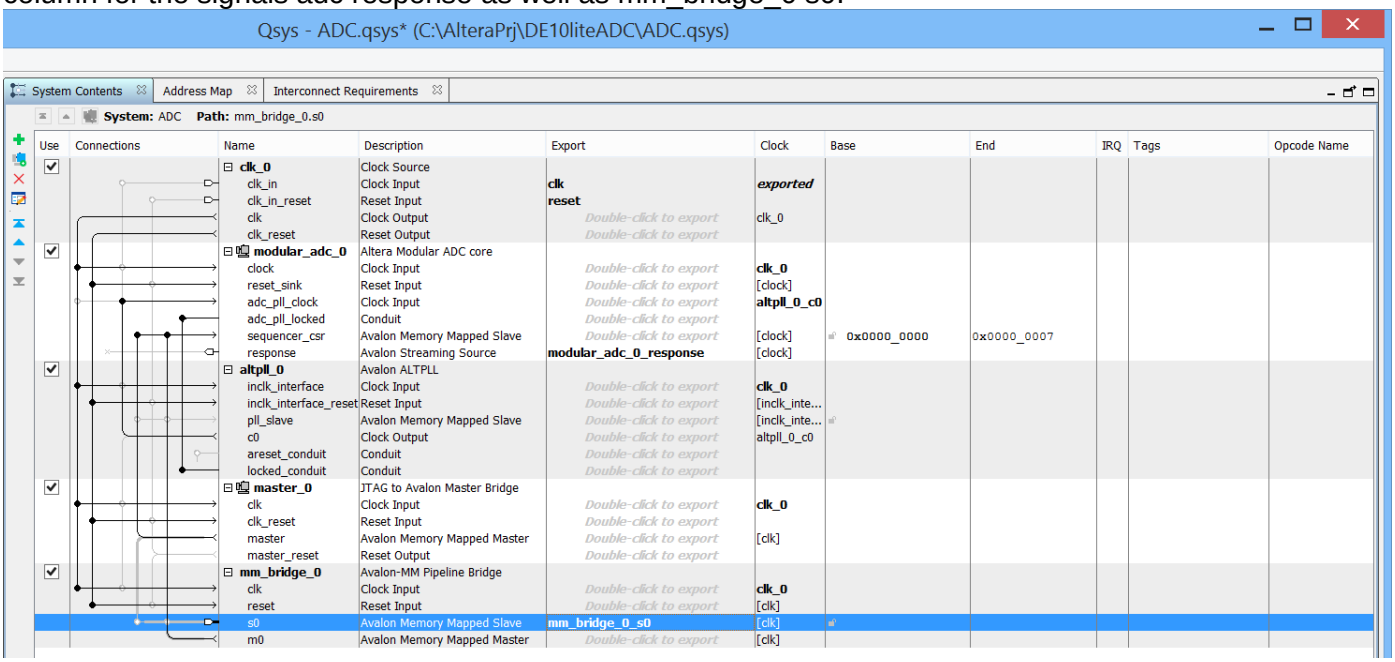


8) Now to interconnect all the Qsys components. By clicking on the small circles to the left of the signal names, the interconnections are made. Connect the clk_0 Clock Output to the modular_adc_clock input, the altpll_0 clock input, the master_0 clock input, and the mm_bridge_0 clock input. This should have eliminated several of the listed errors. Connect the clk_0 clk_reset to modular_adc_0 reset sink, altpll_0 inclk_interface_reset, master_0 clk_reset, and mm_bridge_0 reset. Only one error should be left. Connect the alt_pll_0 c0 output to the adc_pll_clock input, and the altpll_0 locked conduit to the adc_pll_locked Conduit. Now there should be no errors, and your design should look something like this:

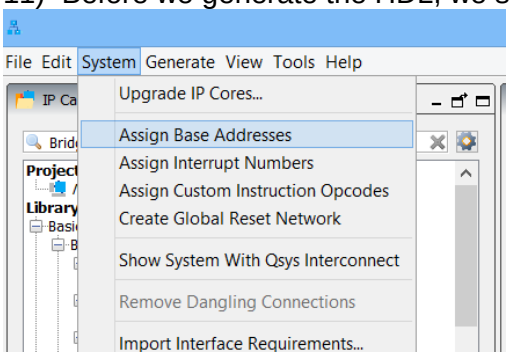
9) However, we are still not done. Connect the sequencer_csr slave to the master_0 master and the mm_bridge_0 master. Connect the modular_adc_0 adc_pll_locked to the atl_pll_0 locked conduit. At this point, you should see:



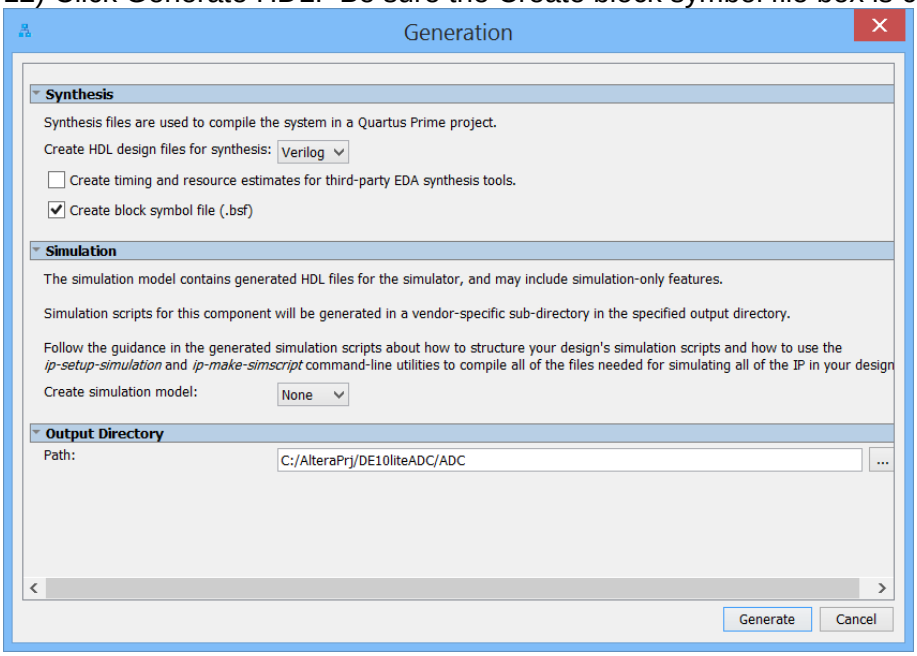
10) To expose signals to the top level, they need to be exported. Export them by double clicking in the export column for the signals adc response as well as mm_bridge_0 s0.

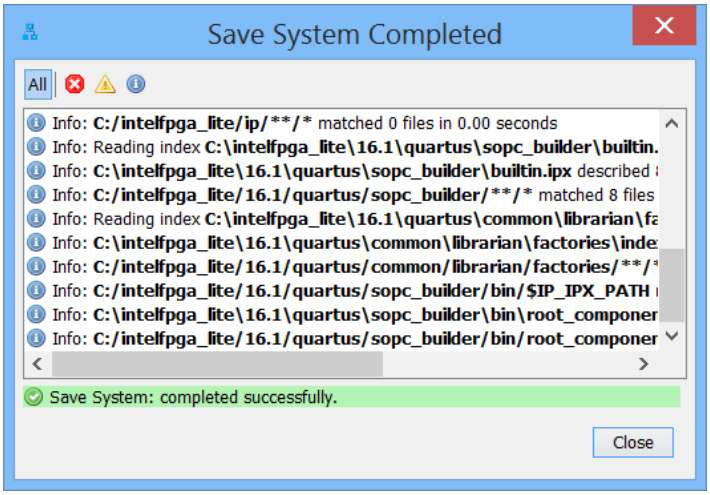


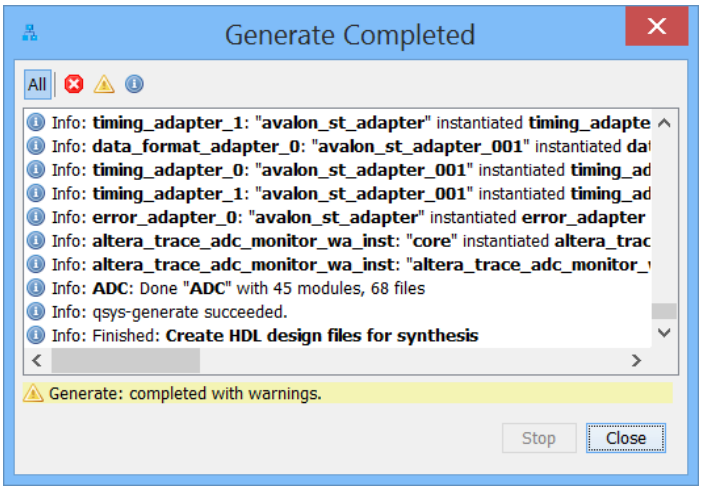
11) Before we generate the HDL, we select System and then Assign Base Addresses:



12) Click Generate HDL. Be sure the Create block symbol file box is checked.

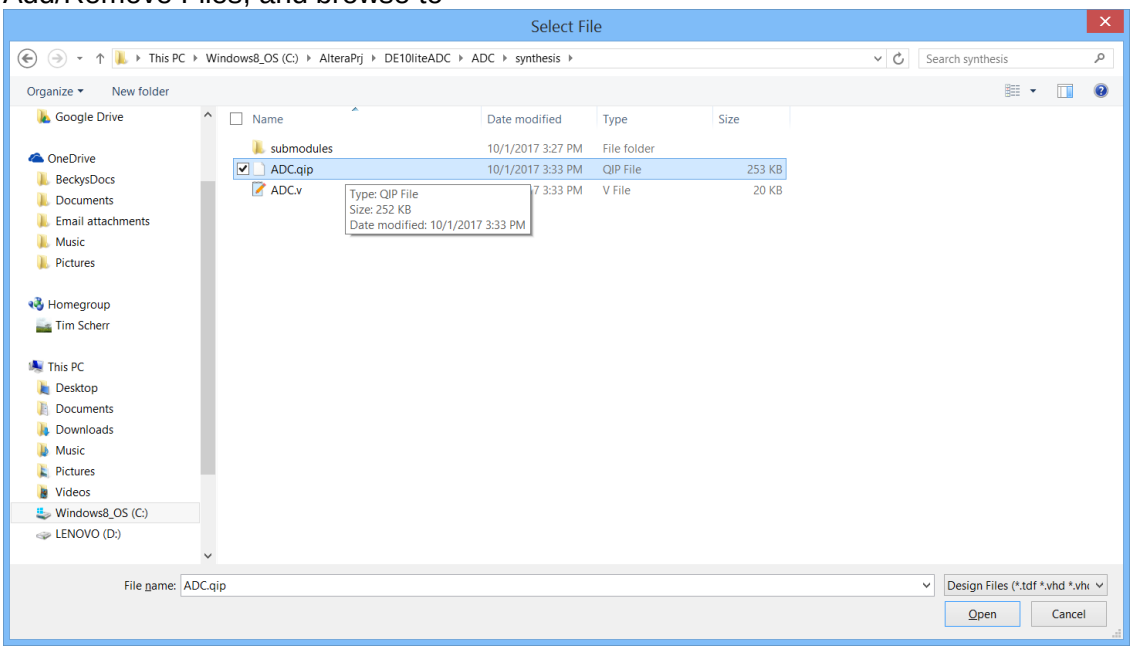






13) Save the Qsys system.

14) Back in Quartus, you may see a message to add the ADC.qip file. Do so by selecting Project – Add/Remove Files, and browse to



Click open, then apply and OK to add the .qip file.

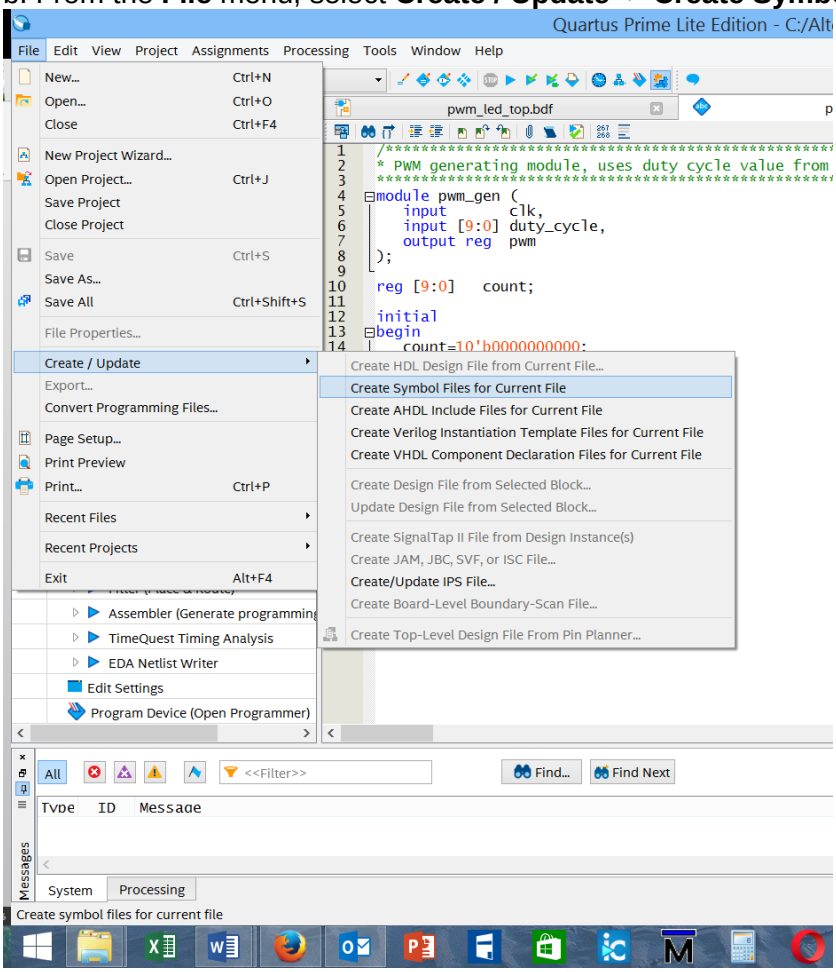
3.3. Create and Add Symbols to the top-level schematic.

The design uses several Verilog files which each define a design entity. The different design entities will need to be connected together to create our FPGA design. While the Verilog design entities can be connected together with Verilog code, another option exists in the Quartus software. It is possible to create symbols for Verilog files and then the design entities can be placed onto a Quartus block diagram file and can be connected together using the Quartus schematic editor.

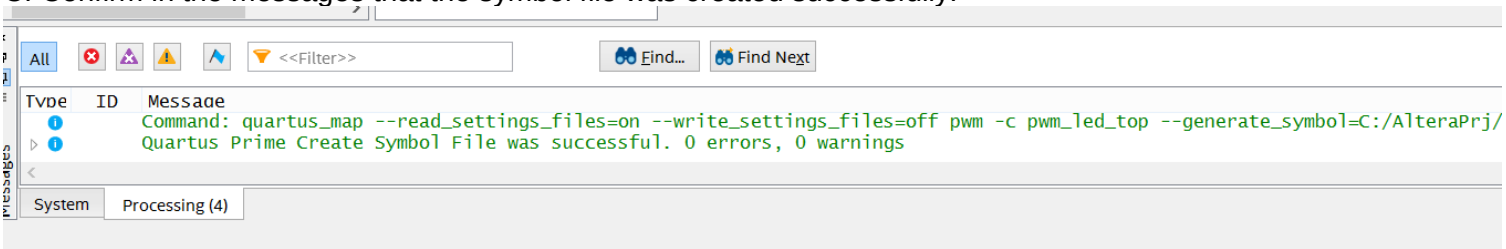
1) Add the ADC module. Right-click in the schematic, select insert symbol, under the project library select the ADC and place it below the debounce switches
2) In Project – Add/Remove files, add files from the source directory, including ADC_connect.vhd, adc_led7.vhd, adc_sequencer.vhd, SEG7_LUT.v and SEG7_LUT_6.v. Note that we are mixing both Verilog and VHDL files in the same design.
3) For the new HDL files, ADC_connect.vhd and SEG7_LUT_6.v, create a block symbol for each and add to the schematic. For each file, do the following:

a. Open the HDL source file.

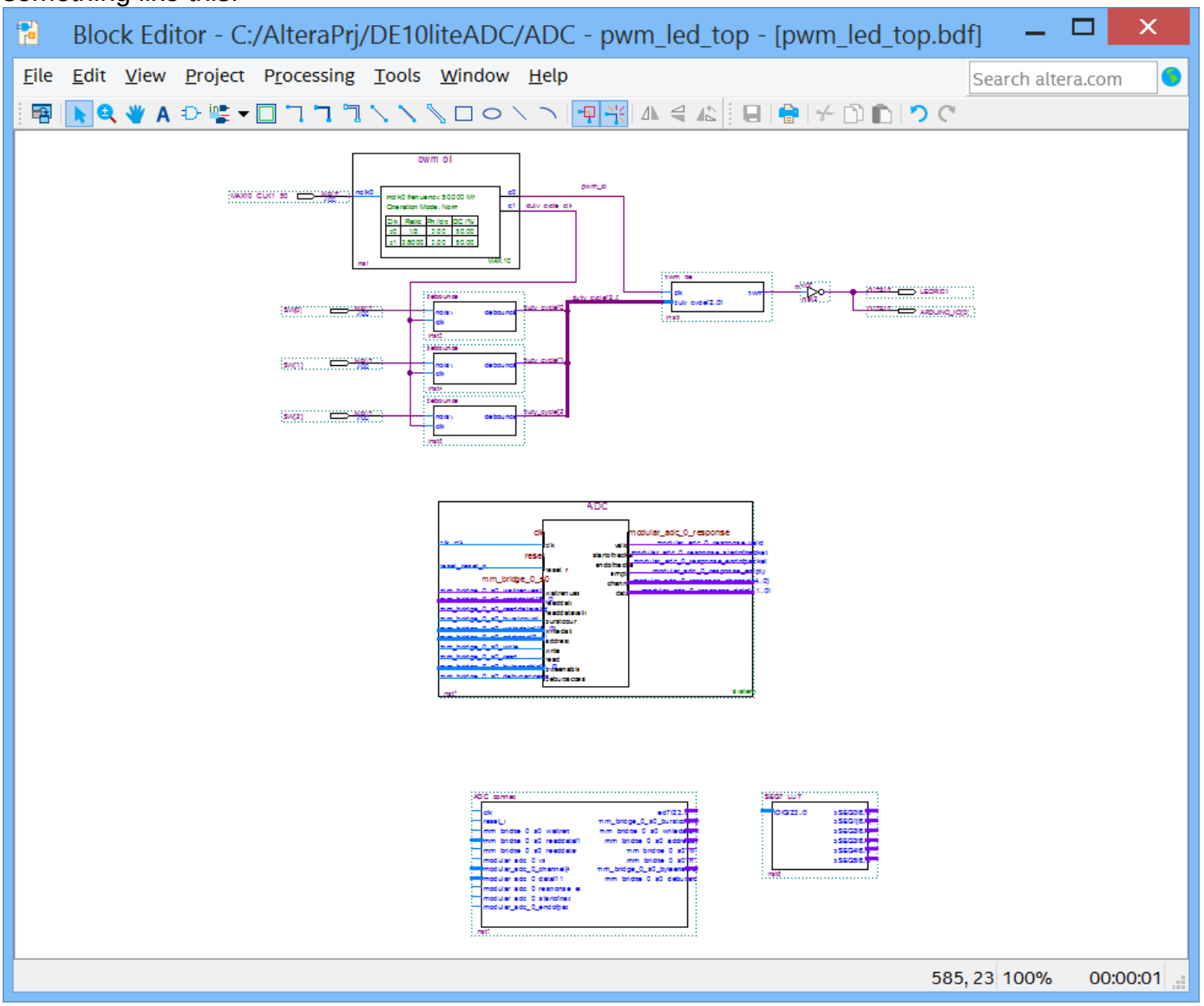
b. From the **File** menu, select **Create / Update** -> **Create Symbol Files for Current File**



C. Confirm in the messages that the symbol file was created successfully:



4) Insert the ADC_connect block below the ADC module.
5) Insert the SEG7 block to the right of the ADC_connect block. Your schematic should then look something like this:

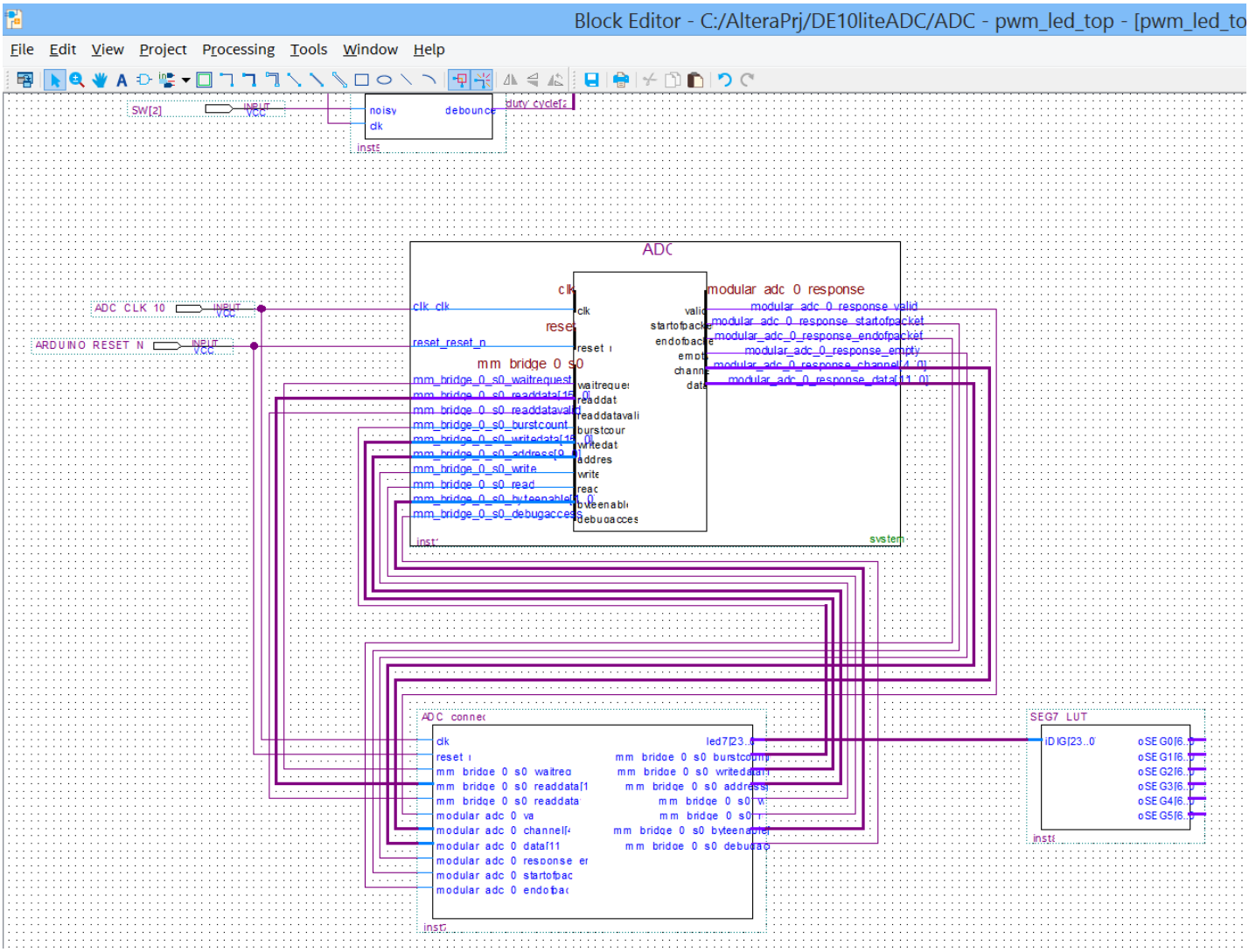


3.4. Complete the Schematic

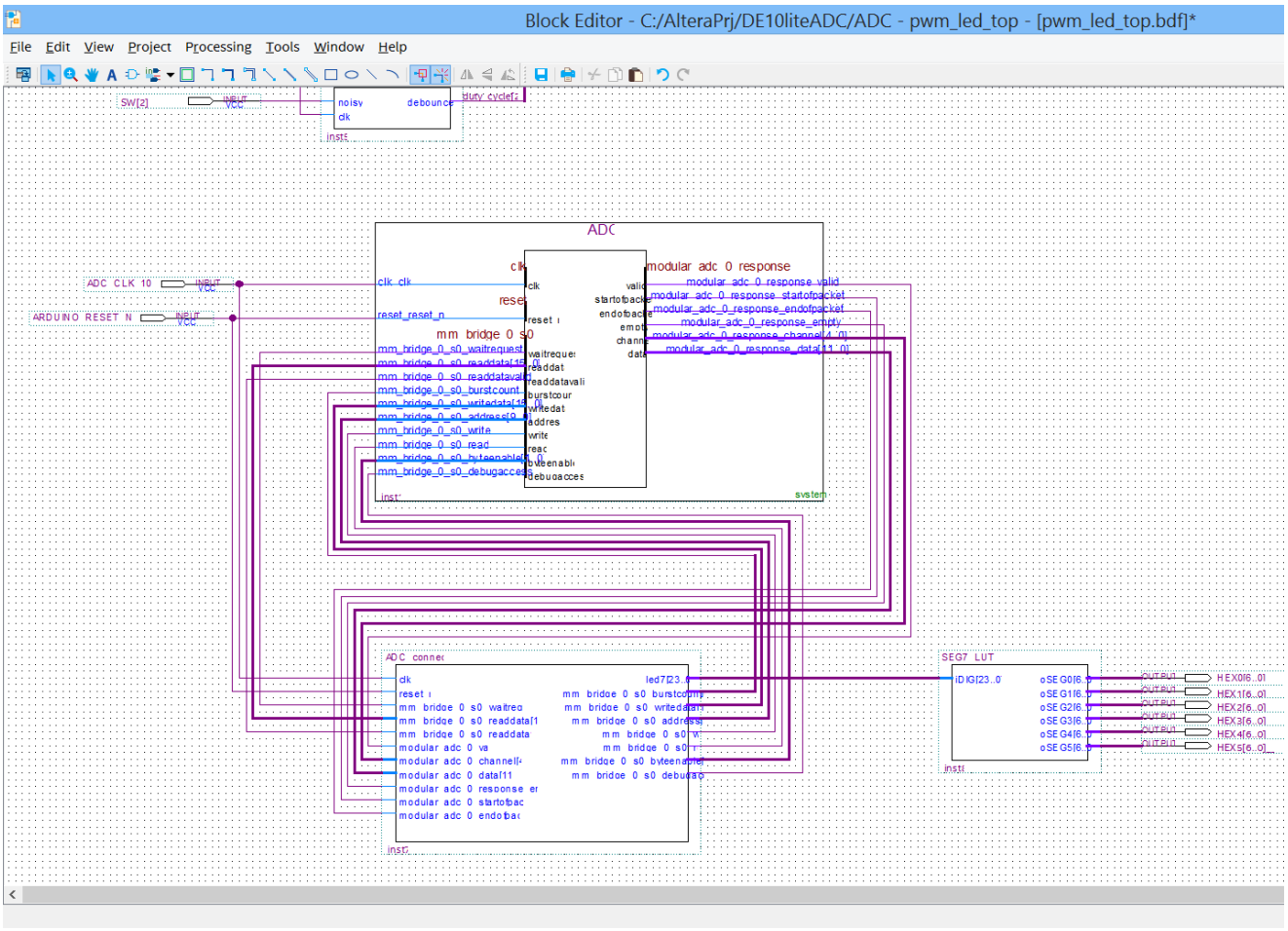
Now that we have all the components, we need to wire them together.

1) For the ADC clock input, create an input port and name it ADC_CLK_10. This signal name comes from the .qsf file and the DE10lite schematic. Connect this clock to the ADC clk_clk input and the ADC_connect clk input. Create another input port and name it ARDUINO_RESET_N, and connect this to the ADC reset and ADC_connect reset.

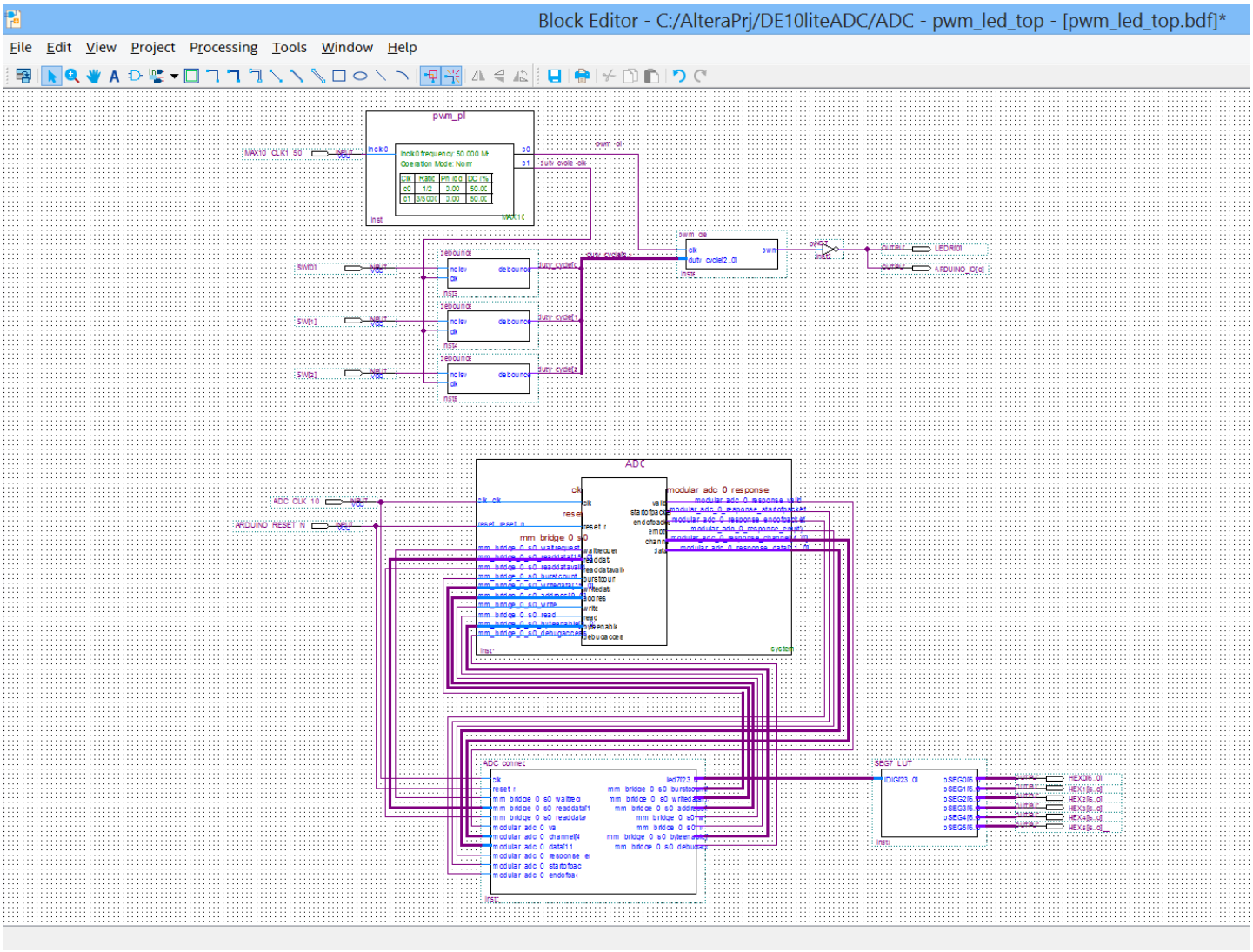
2) Connect all the mm_bridge signals from the ADC to ADC_connect.
3) Connect all the modular_adc signals from the ADC to ADC_connect.
4) Connect ADC_connect led7 to SEG7_LUT iDIG[23..0]. At this point the schematic will look something like this:



5) Next, wire up the 7-segment displays. Create an output port, right-click on it and select properties. Change the pin name to HEX0[6..0]. You must use this exact name from the .qsf file. Do this for HEX1 through HEX5. Connect the corresponding outputs from SEG7_LUT to the HEX output ports. Your schematic is now complete, and should look like this:


Block Editor - C:/AlteraPrj/DE10liteADC/ADC - pwm_led_top - [pwm_led_top.bdf]*
File Edit View Project Processing Tools Window Help
SW[2]
INPUT
VCC
noisy
clk
debounce
duty_cycle[7..
inst5
ADC
clk
clk_clk
reset
reset_reset_n
mm_bridge_0_s0
modular_adc_0_response
modular_adc_0_response_valid
modular_adc_0_response_startofpacket
modular_adc_0_response_endofpacket
modular_adc_0_response_empty
modular_adc_0_response_channel[4..0]
modular_adc_0_response_data[11..0]
ADC_CLK_10
ARDUINO_RESET_N
inst
system
ADC_conne
led7[23..0]
SEG7_LUT
iDIG[23..0
oSEG0[6..
oSEG1[6..
oSEG2[6..
oSEG3[6..
oSEG4[6..
oSEG5[6..
HEX0[6..0]
HEX1[6..0]
HEX2[6..0]
HEX3[6..0]
HEX4[6..0]
HEX5[6..0]
inst1

Final Schematic:



6) Go to the **File** menu and select **Save** to save the changes you have made to the schematic block diagram file. Run an Analysis and Elaboration (or Analysis & Synthesis).

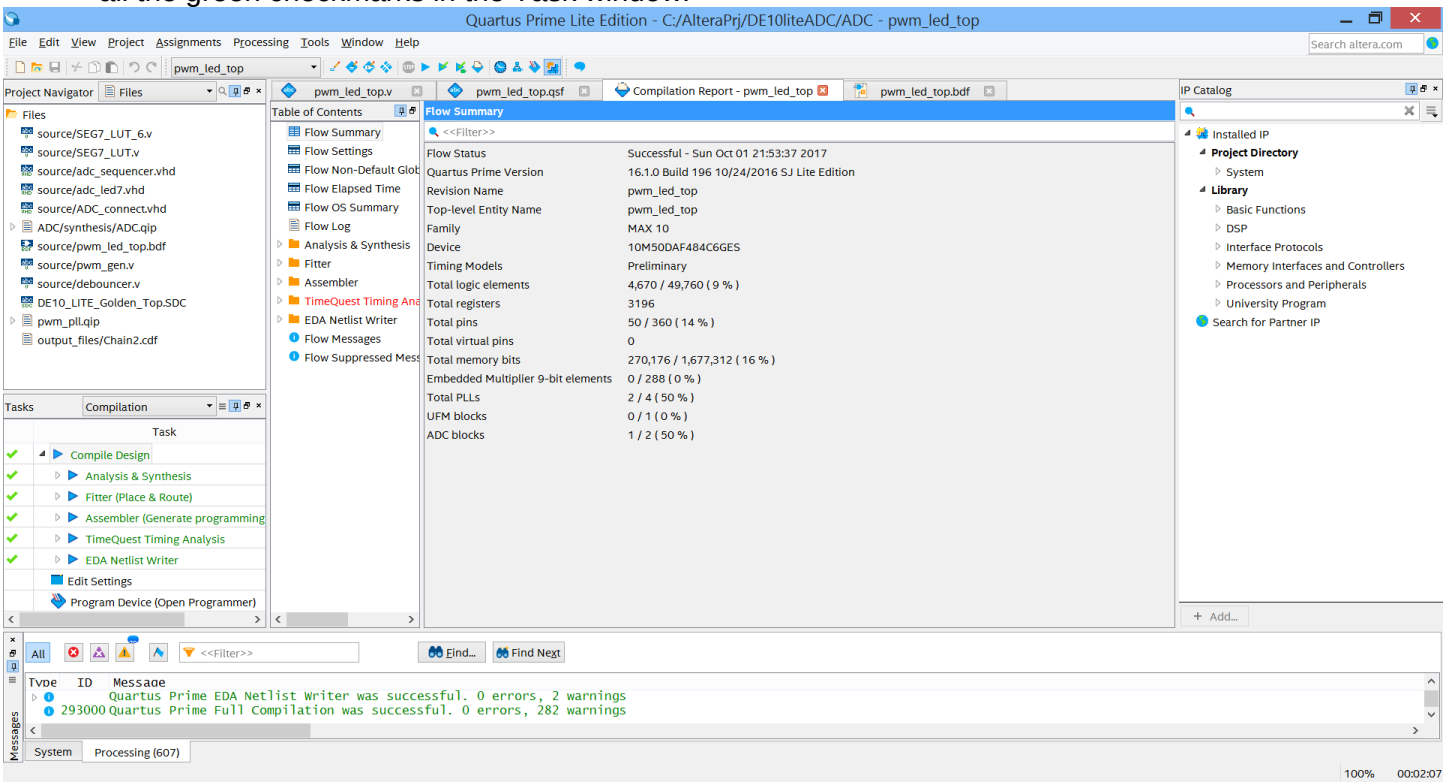
CONGRATULATIONS!!

You've completed the design entry of the ADC circuit!

4. Placing and Routing the Design

Section Objective: In this section you will do a full compilation on the ADC + PWM design. This design runs with the fastest clock at 50 MHz, so we have not entered timing constraints as one normally would.

1. Double click on Compile Design in the Task window. You should see a result something like this, with all the green checkmarks in the Task window:



2. When compilation complete, look at the Flow Messages. Note that there are tabs at the top of the messages window that allow you to filter by message type.
3. Look at the TimeQuest Timing Analyzer, Slow 1200mV 85C Model, Fmax Summary to determine the Fmax. Not that the TimeQuest result is in red because we have not yet constrained all the ports. This is not a concern for now.
4. Look at the Flow summary to determine the total registers (Flip-Flops) and % utilization of logic elements.

CONGRATULATIONS!!

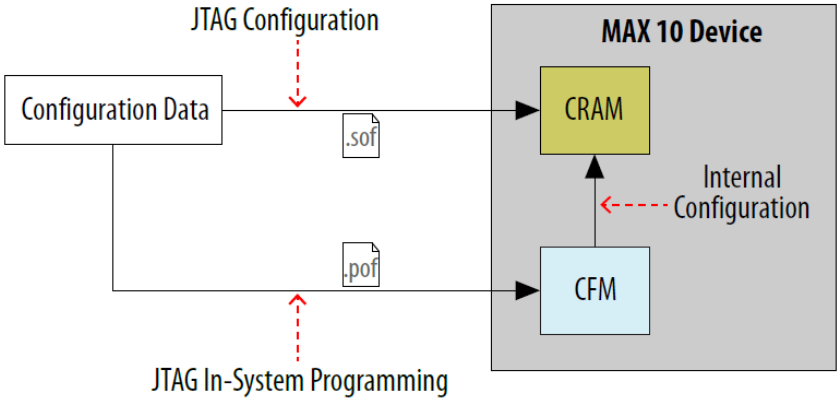
You have just placed and routed your FPGA design.

5. Programming the Hardware and Testing the Design

Section Objective: In this section you will learn how to generate a programming file that can be handed off to production.

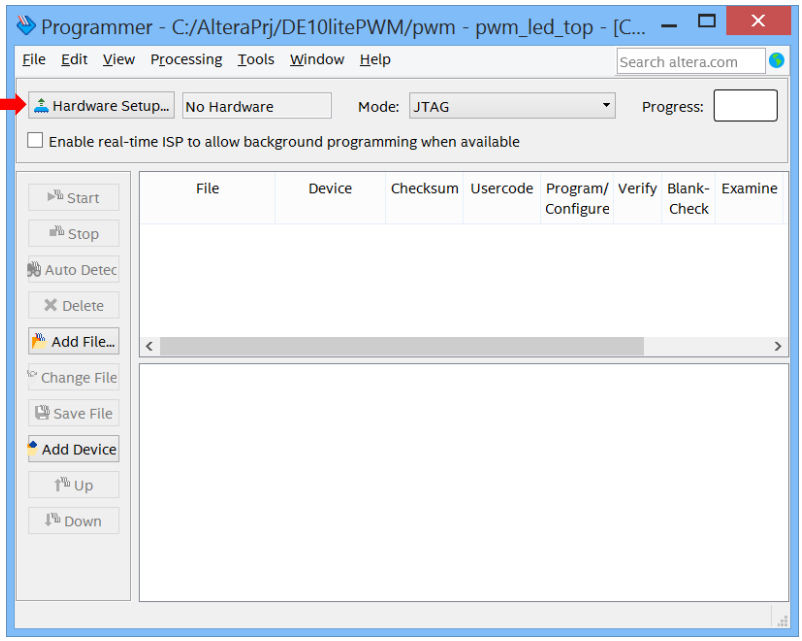
The MAX10 is unique to Altera in that it has internal FLASH memory for configuration, and so there are 2 ways to program it. On is with JTAG as with other FPGAs using a .sof file directly to the SRAM configuration cells, and the other also uses JTAG but programs the configuration flash memory, which is transferred to the SRAM configuration cells on power up. JTAG programming requires a programming cable, like a USB Blaster II or Ethernet Blaster II.

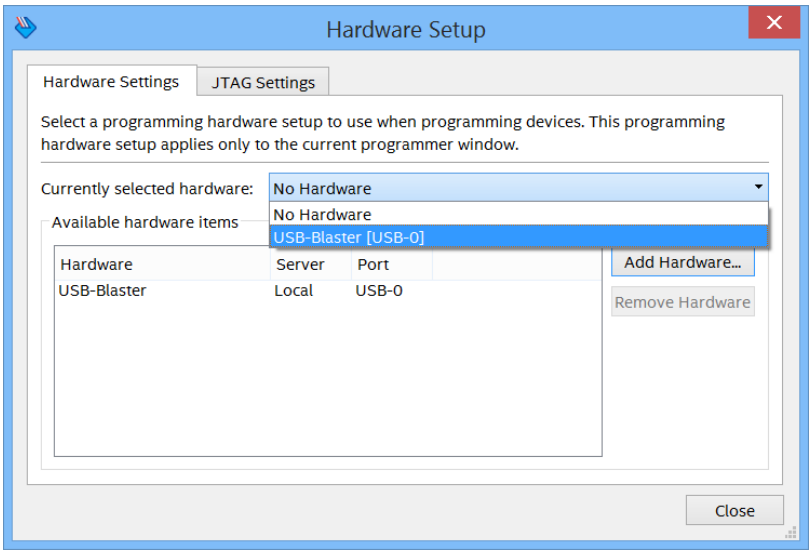
Figure 29: Overview of JTAG Configuration and Internal Configuration for MAX 10 Devices



5.1. Programming the DE10-lite

After Compilation, do the following to program the board:
1) Connect the USB cable to your DE10-Lite kit.
2) Plug the other end of the USB cable to a USB port of your computer.
3) Launch the Quartus Programmer, via the icon or through the Tools menu (**Tools -> Programmer**)
4) Setup the programming hardware. To do this, click the hardware setup button in the upper left corner of the programmer, and select the hardware you want to use. Choose USB Blaster in the Hardware Setup Dialog.

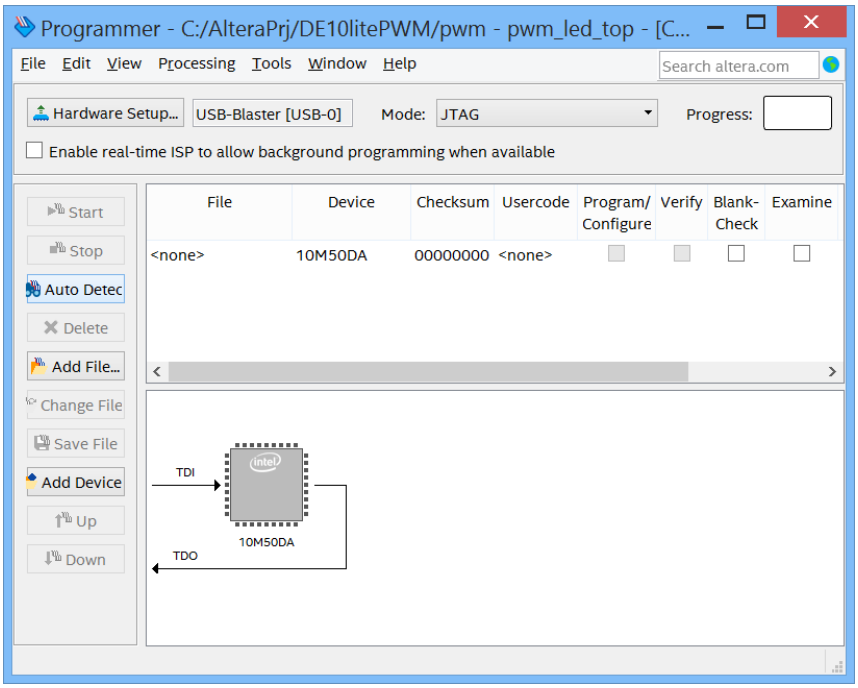




Close the Dialog Box and your setup should appear as this:

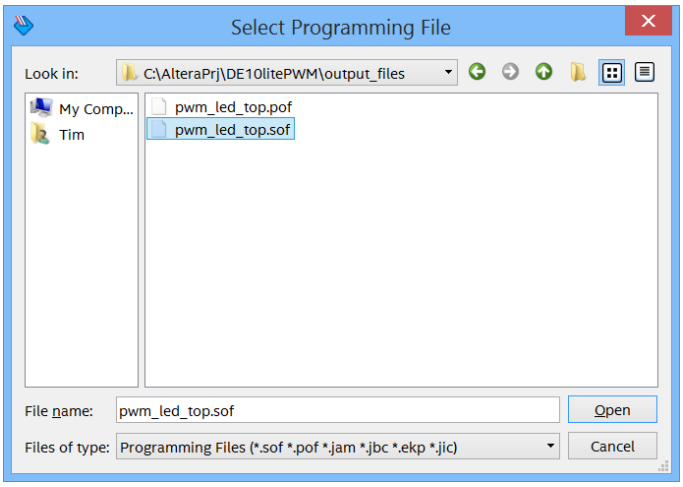
A programming device must have drivers installed and be detected correctly to be listed. Next select the programming mode – the most common is JTAG. The JTAG chain can consist of both non-Altera and Altera devices.

Once the hardware is setup, a toolbar in the programmer provides all the commands needed to control the programming of devices. For example, the order of programming devices on the chain can be arranged. Other common operations include Auto detect, in which the chain is scanned and devices found is reported, and change file, which selects a new file to program into the selected target device. Clicking on Auto Detect makes the programmer show the device chain:

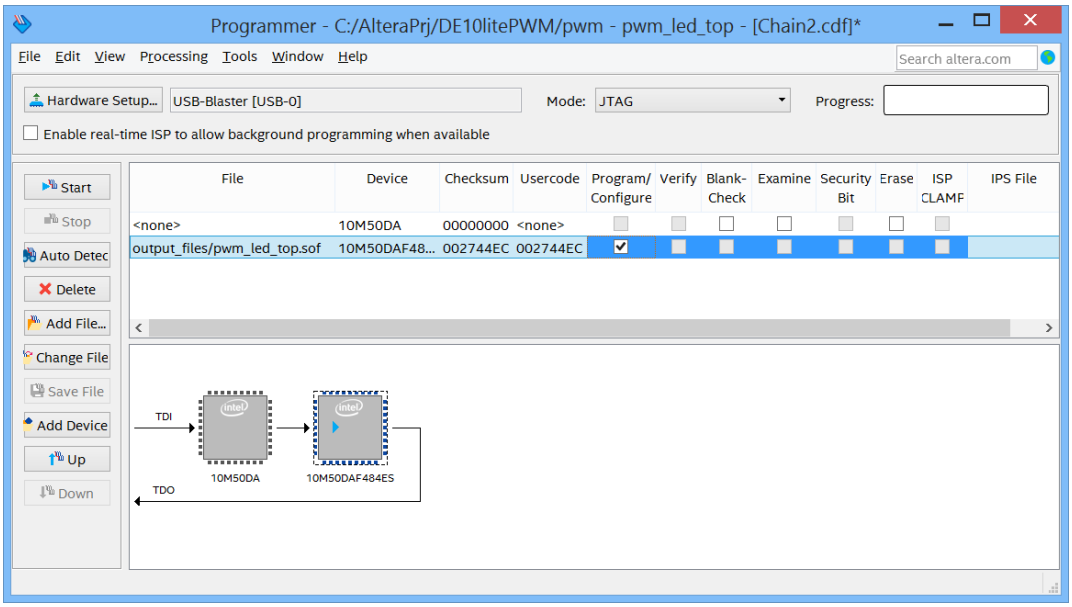


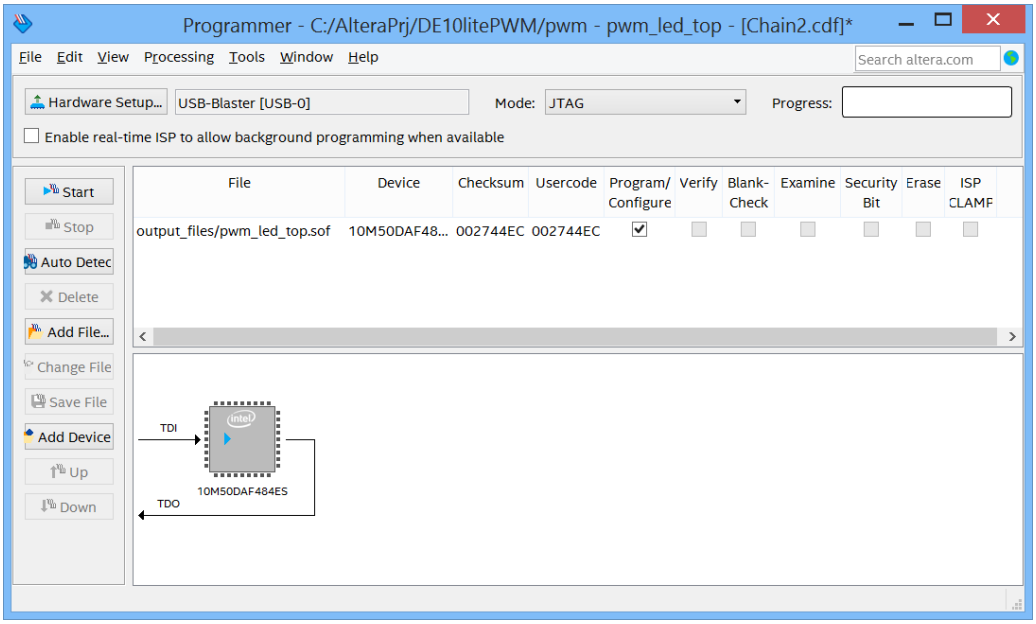
You can also verify a device after it has been programmed, or blank check a device, erase a device, or set a security bit if available.

5) To select the programming file, click Add File, in this case pwm_led_top.sof.

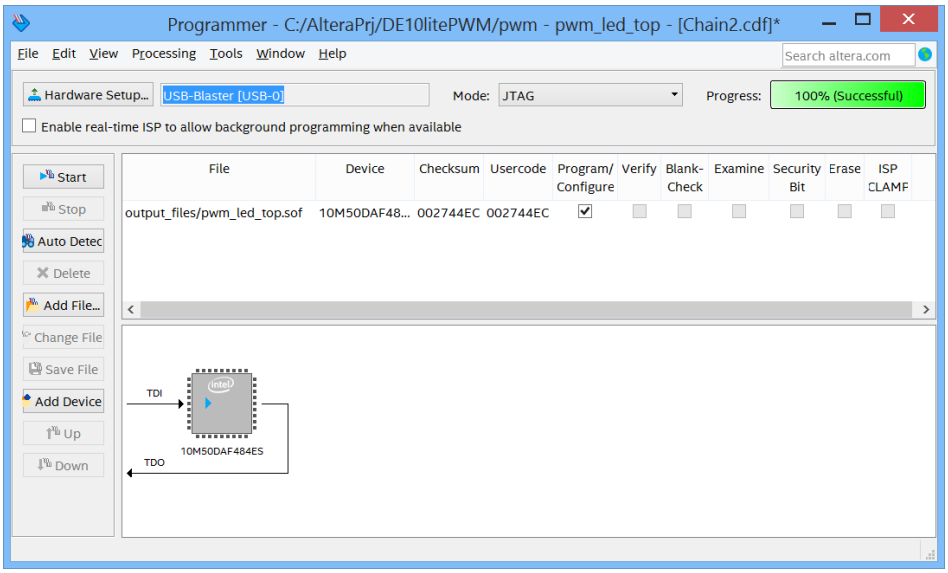


Click Open. Back in the programming window, you may have to delete any other entries that are listed.

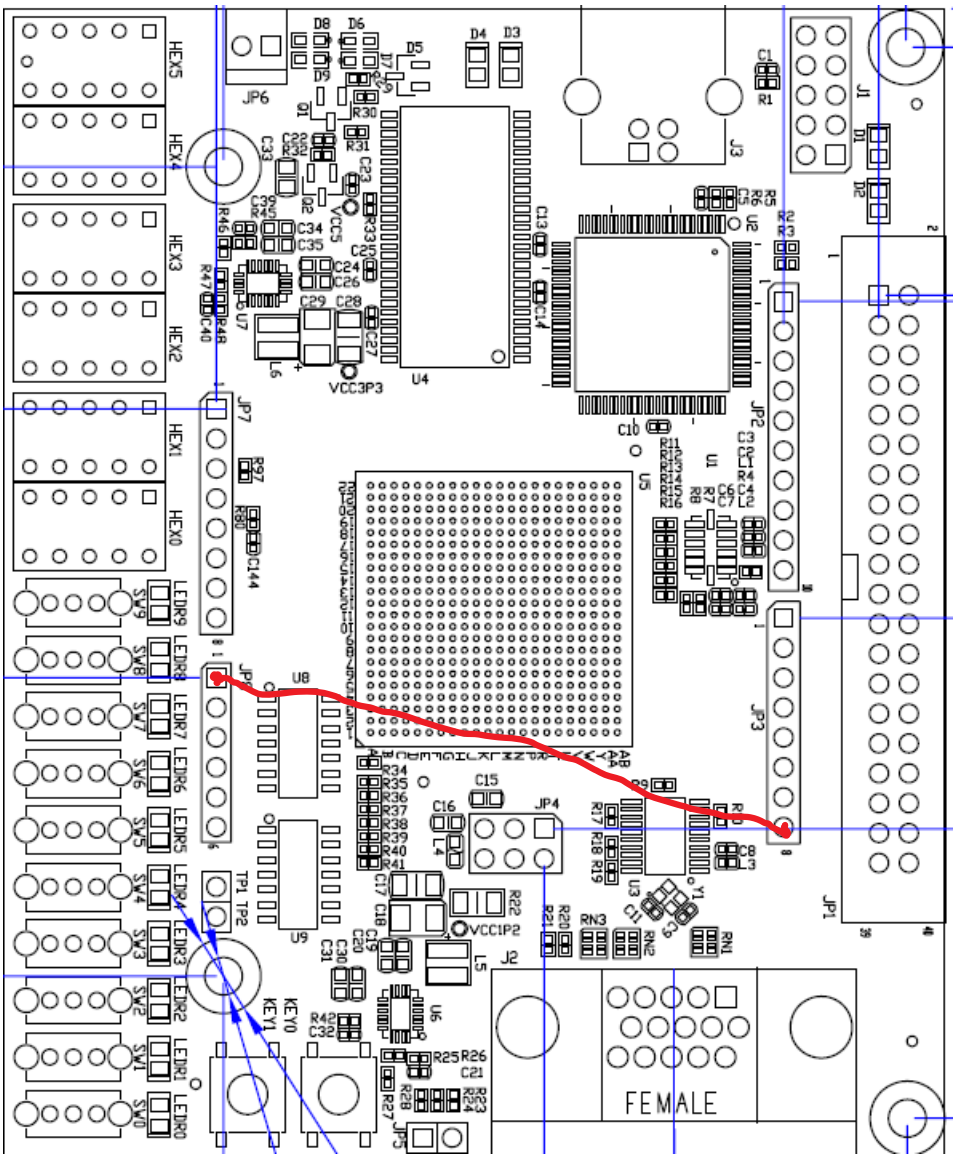




6) When all the devices are defined and options set correctly, click on Start in the upper left. The progress bar shows the status of the programming and the messages windows provides detailed status, and information about any errors that may occur. When the progress bar reaches 100%, the programming is complete.

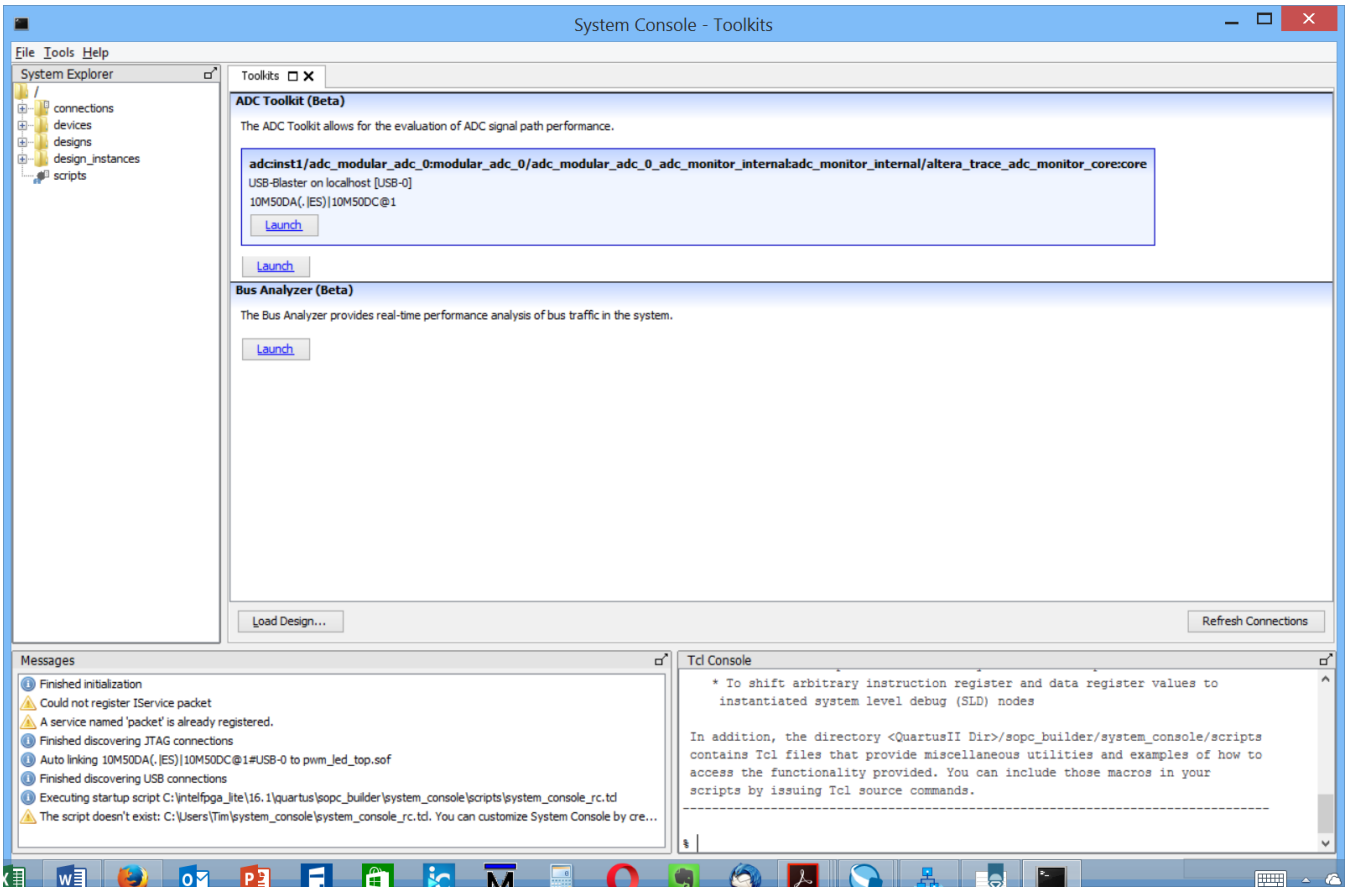


7) You should see the 7 segment LED display change. The last 3 switches should still control the brightness of LED0. If not, you may need to use the RTL viewer, etc. to debug the design.
8) Connect a jumper wire between pin 8 on JP3 and pin 1 of JP8. See the picture below:

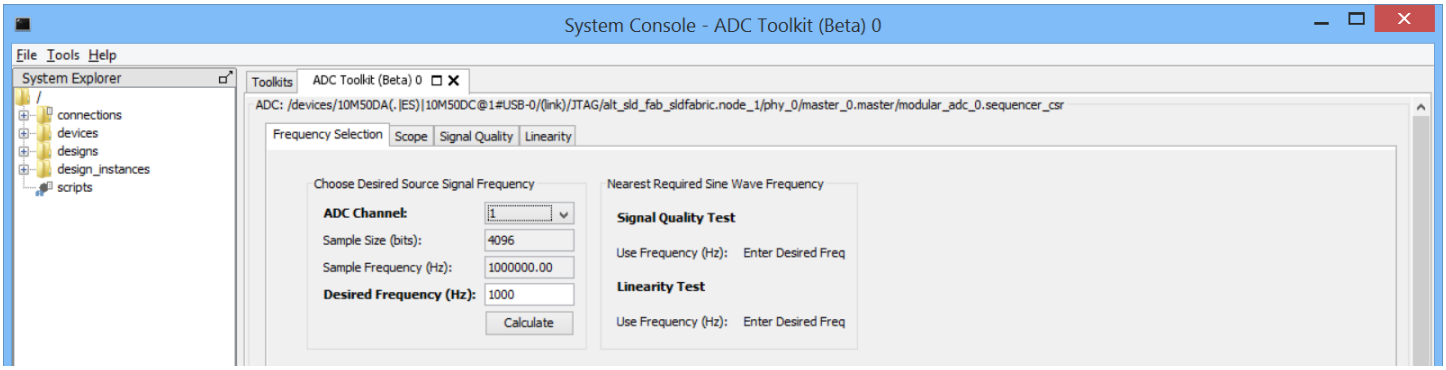


9) Now when you change switches 0, 1, and/or 2 you should different values on the 7-segment display, proportional to the binary sum of your switch selections. These are raw ADC values in 12 bits represented as 3 Hex numbers.
   Optional BONUS points: rewrite the added files to display the voltage instead of the raw ADC values.

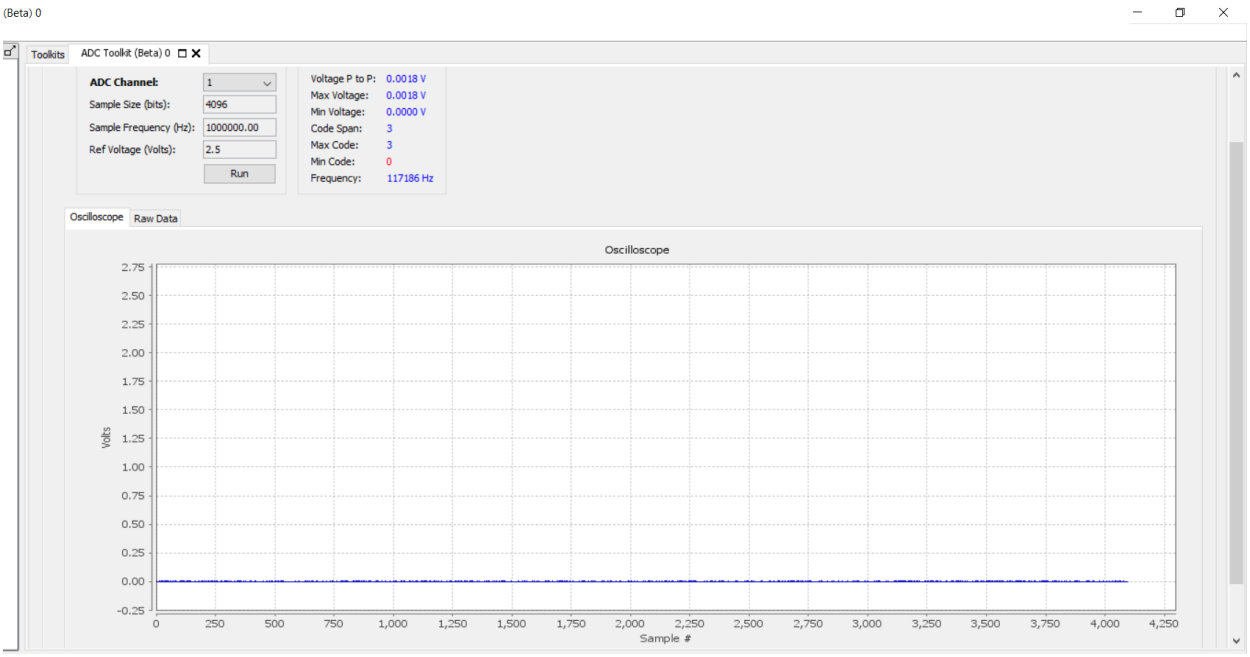
10) Additional analysis and debugging can be done using the ADC toolkit. To use this, go back to Qsys, or start Qsys and load it with the ADC.qsys design. In Qsys, select tools and then System Console. If your board is still connected, you should see:



11) Click Launch on the ADC toolkit. Retain the defaults as shown:



12) Select Scope, hit Run and record what you see. It may take a long time before the results appear. There is a limit as to the signal that can be displayed, if the frequency desired is not seen it will be undetected. The toolkit may also not be able to acquire data because of the presence of another master on the ADC control bus. If it works, you may see something like this:



CONGRATULATIONS!!

You have just completed the ADC FPGA Design!

References

[1] Intel Altera. (2016). *Cyclone V Device Handbook*. [Online]. Available: https://www.intel.com/content/www/us/en/programmable/products/fpga/cyclone-series/cyclone-v/support.html

[2] Intel Altera. (2016). *Max 10 Device Handbook*. [Online]. Available: https://www.intel.com/content/www/us/en/programmable/products/fpga/max-series/max-10/support.html

[3] Intel Altera. (2016). *Qsys System Design Tutorial.* [Online]. Available: https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/tt/tt_qsys_intro.pdf