Version: 20/12/2012

DE2-115/FGPA README

*For questions email: **jeff.nicholls.63@gmail.com** (do not hesitate!)*

This document serves the purpose of providing additional information to anyone interested in operating the DE2-115 or making changes to the logic on the DE2-115 FPGA. This document will provide the following:

1. An overview of how to operate the DE2-115 during basic operation.
2. A description of the code and project files.
3. Instructions on how to use Quartus to open, edit and compile the code.
4. How to download/program the code onto the FPGA.

1. Running the DE2-115 for basic operation

To run the DE2-115, the following simple procedures should be used.

a) Ensure that the DE2-115 is plugged in to power.
b) Ensure that the RS232 cable is plugged into the RS232 serial port (not the VGA serial port) of the DE2-115 and that it is also connected to the serial port at the back of the computer.
c) Ensure that the 40 pin cable is connected to both the DE2-115 and the BNC Hub with the orientation arrows pointing towards each other and the labels “FPGA” and “BNC” attached to the respective ends.
d) Ensure that the RUN/PROG switch is in the RUN position. The switch is located on the left-hand side just above the set of 17 switches (but is not one of the 17 switches itself).
e) Press the red power button to turn the DE2-115 on.
f) If successful, a blue light beside the power button will illuminate. If everything is working properly, a small green light beside the RS232 port should flash on and off at a rate of 10 times per second.
g) The 17 switches can then be used to adjust pulse widths according to the Table attached just below the FPGA (and also given later in this document).
h) To turn the DE2-115 off, simply press the red power button again.

Note: The DE2-115 can be turned on and off at will without any danger. The only time that it should not be turned off is during programming.

2. The code/project files

Project Files

The DE2-115 Development Board houses a Cyclone IV FPGA which contains all coincidence detection logic, registers for storing count and coincidence information and an RS232 module. The code to program the FPGA is written in VHSIC Hardware Description Language (VHDL). All relevant project

files can be found in the folder “DE2-115 Project Files” on the desktop of the lab computer. The overall project consists of 7 VHDL files in total:

- RS232coincidencecounter.vhd (top level)
- mux32to1.vhd
- coincidence_pulse.vhd
- counter.vhd
- baud_counter.vhd
- data_trigger_counter.vhd
- DataOut.vhd

There is also an additional file included, titled “altera_mf_components.vhd” which contains the modules required for implementing LCELLs (discussed later). The overall project file is titled “RS232coincidencecounter.qpf”. Opening this file will start Quartus and open all relevant project files. Opening this file is the best way to make changes to the code.

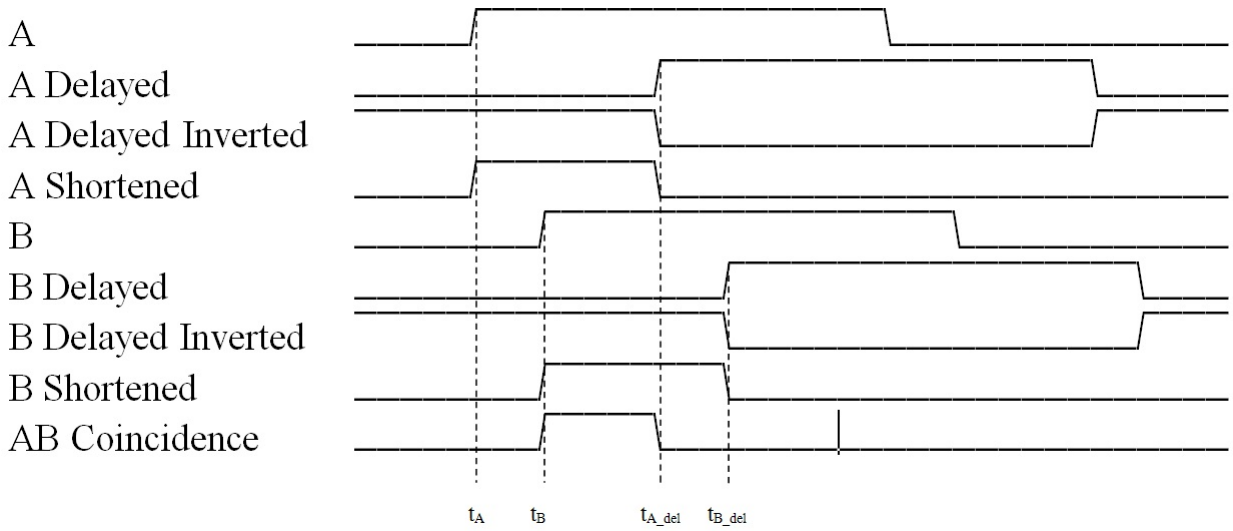
Another important file is the file titled “RS232coincidencecounter.qsf” which contains all of the physical pin assignment information for the DE2-115. DO NOT MAKE CHANGES TO THIS FILE UNLESS YOU ARE ABSOLUTELY SURE YOU KNOW WHAT YOU ARE DOING. To increase the number of GPIO_IN pins, go into this file and change the assignment of GPIO_OUT pins to GPIO_IN pins accordingly. Likewise for changing GPIO_OUT to GPIO_IN. Note that changes can also be made to this file via the assignment editor in Quartus, in fact it is recommended that the latter method is used.

There also exists a folder titled “DE2_Systems_Disc”. This folder contains all of the files that came with the DE2-115 including manuals and tutorials on the DE2-115, Quartus and VHDL. This folder is a great reference for anyone that is confused. Beyond that there is an old readme for the original version of the experiment developed at Whitman College. Most of the information in this document is out-dated but can still be used as a reference if necessary. The rest of the files are mostly files used by Quartus while compiling and synthesizing and should be left alone.

Overview of Code

A basic overview of how the code works along with the design files will be given below. For more information, consult the design report(s) or the documentation within the code itself. There is also a general schematic overview given in the appendix at the end of this document.

The goal of the logic on the FPGA is to determine whether any of the signals from the detectors are coincident within a certain amount of time referred to as the timing window. The FPGA will store both the number of coincidences and the number of counts from each detector for approximately 0.1 s. Every 0.1s, the FPGA will then begin a read-out of these counts and coincidences to the computer via an RS232 serial connection. The coincidence window is chosen by shortening the pulses to shorter pulse widths, where the window essentially becomes twice the pulse width. To shorten each pulse, the pulse is first delayed, then inverted, then ANDed with the original pulse producing a pulse with width equivalent to the delay. Shortened pulses are then ANDed together to determine coincidences. Refer to the following figure for a visual example.



The signals from all four detectors A, B, C and D are first input into the FPGA via "GPIO" pins 0, 2, 4 and 6 which are designated as "GPIO_IN". These signals are then each fed into a series of LCELLs. The LCELLs are actually just empty logic cells that do nothing other than add to the propagation time of the pulses, effectively delaying them. A chain of LCELLs is used in order to create a wide range of delays output on a bus titled "X_internal" (where X represents A, B, C or D). Note that there is one delay chain per channel, and the chain for each is created using a GENERATE statement. Since the LCELLs are actually empty cells, Quartus will attempt to optimize them away during compilation. To avoid this, the "X_internal" signals are given the attribute "syn_keep" which prevents the compiler from removing both the signals and the LCELLs.

The delayed pulses for each channel along with the associated original pulse are then input into a mux32to1 module (MA, MB, MC, MD). Note that there is one multiplexer per channel. The 32to1 mux is not only a multiplexer, but also includes all of the shortening logic as discussed before. To determine which delayed pulse to use for the shortening, the switches 17 through 13 on the DE2-115 are used as selectors. A table giving the rough pulse widths for each switch selection can be seen below. Note that the widths are only just approximate and are measurements of the FWHM of each pulse created using the actual APDs. The actual pulse width varies due to temperature and power fluctuations, and thus it is best to use a coincidence window determined from calibration via LabVIEW.

Table 1: Estimate of shortened pulse widths with switch selections.

Switch Setting ("17;16;15;14;13")	Shortened Pulse Width [ns ± 0.1] (measured @ 2.5 V crossings)	Pulse Peak Voltage [V ± 0.05]
0: "00000"	14.45 (original pulse, no shortening)	5.2
1: "00001"	13.7	5.2
2: "00010"	13.35	5.1
3: "00011"	13.3	5.1
4: "00100"	12.9	5.1
5: "00101"	12.5	5.1
6: "00110"	12.25	5.1
7: "00111"	12.1	5.05
8: "01000"	11.8	5.05

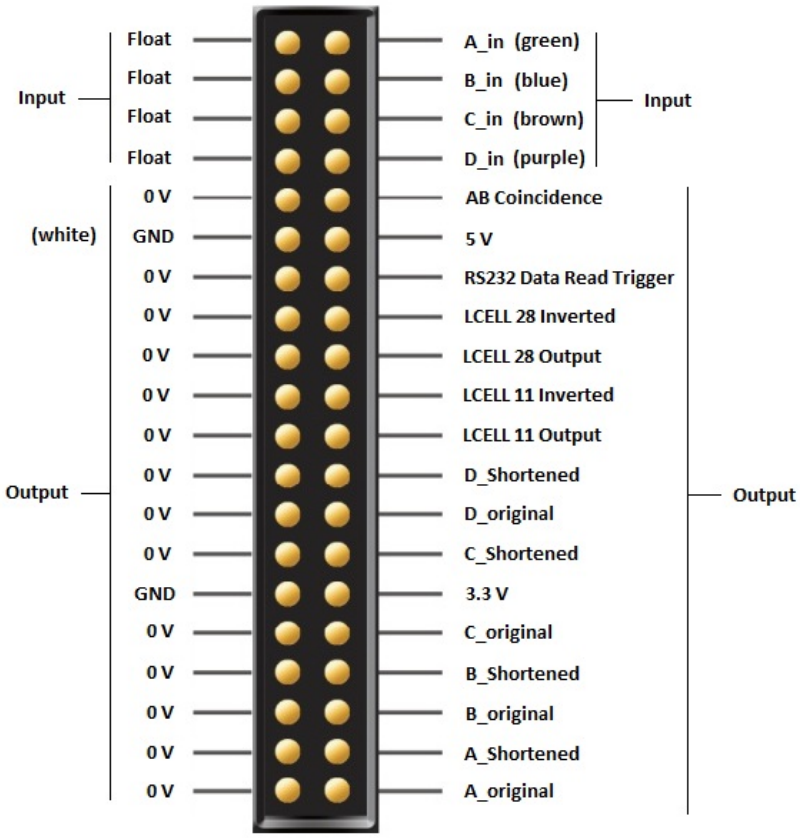
9: “01001”	11.35	5.1
10: “01010”	11.2	5.1
11: “01011”	11.0	5.1
12: “01100”	10.85	5.1
13: “01101”	10.25	5.1
14: “01110”	9.9	5.1
15: “01111”	9.75	5.1
16: “10000”	9.45	5.05
17: “10001”	9.0	5.0
18: “10010”	8.7	4.9
19: “10011”	8.7	4.95
20: “10100”	8.2	4.85
21: “10101”	7.4	4.55
22: “10110”	6.95	4.35
23: “10111”	6.8	4.25
24: “11000”	6.65	4.15
25: “11001”	5.85	3.8
26: “11010”	5.4	3.55
27: “11011”	5.1	3.35
28: “11100”	4.45	3.15
29: “11101”	4.3 @ 2.2 V	2.75
30: “11110”	3.6 @ 2.2 V	2.55
31: “11111”	<2.9 @ 2.2 V	2.4

Output from the 32to1 muxes are the signals “X_s” which represent the shortened versions of the pulses for each channel. These are then fed into the coincidence_pulse modules (CP0 to CP10) to determine each coincidence. There are 11 different coincidence_pulse modules, each one corresponding to one of the 11 different two, three and four-fold coincidences. Each module also takes 4 inputs. For the two and three-fold coincidences, a dummy signal of “1” is fed into the port corresponding to the channel which is not taking part in the coincidence. This ensures that the output will not depend on the value of that channel. The output from each coincidence module, “Coincidence(0…10)”,is then used to increment a counter for each specific coincidence (C0 through C10). There are also 4 counters (CA, CB, CC, CD) which are incremented by the original pulses themselves, which act to hold the count for each channel. The counters are 32 bits wide, which is sufficient for both the read-out rate of 10 Hz and the maximum number of pulses which one would ever see from the APDs.

Multiple counters are used to create both the Baud clock used for the RS232 connection and the data readout trigger. The baud_counter (CY) outputs a tick for every 2604 50 MHz DE2-115 clock pulses. This baud clock is then used as the clock for the RS232 connection which operates at 19200 Hz corresponding to approximately 9600 bits per second baud rate. The data_trigger_counter (CX) outputs a tick for every 1920 baud ticks. This tick thus occurs once every tenth of a second and does two things: It first saves the values of the all the registers to signals “X_out”, while at the same time resetting the registers so that they can continue counting while the data is being read out. The trigger then triggers the read-out of the “X_out” signals using the module DataOut (D0).

The DataOut module takes as input all of the saved values of the registers and begins to output them over RS232 by slowly iterating through each bit of "X_out" signal. RS232 works by first sending out a start bit, then 7 data bits, then a parity bit, then a stop bit. Since the registers have 32 bits of data, it thus requires 5 cycles to output one register. After it outputs one register it moves onto the next, repeating the cycle until all registers have been read-out. The data is read out on the signal "UART_TXD" which corresponds to the RS232 pin of the FPGA.

Finally, the red LEDs are assigned to the value of the switches such that it is easy to see which switches are activated in lowlight conditions. Unused output pins are then grounded (set to 0) and the rest of the output pins are assigned signals which allow for debugging. A pin-out description of the pins corresponding to the 40-pin cable in the BNC Hub is given in the following diagram. The colors refer to the wires attached to those pins inside the Hub, and allow for easy orientation of the diagram with respect to the Hub. A general pin-out description matching each pin to its pin label is given in the project files folder, and also can be found in the DE2-115 User Manual.

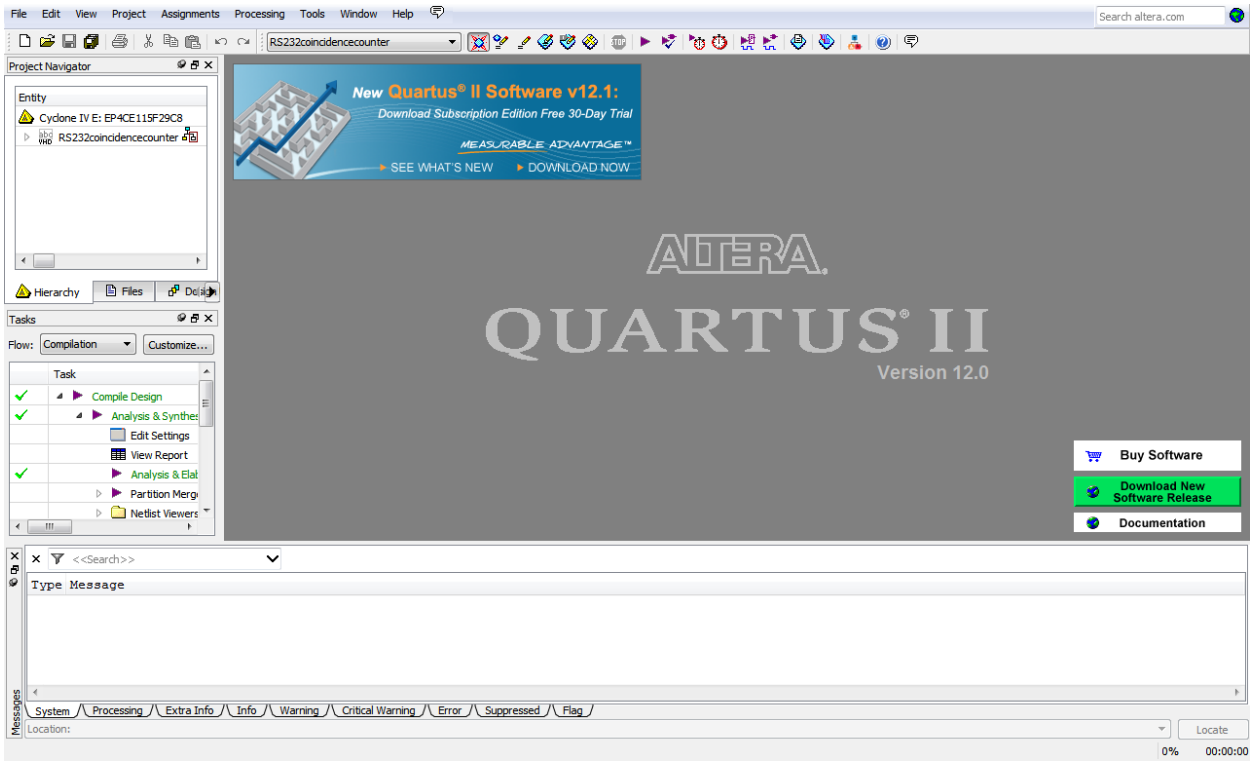


3. Using Quartus

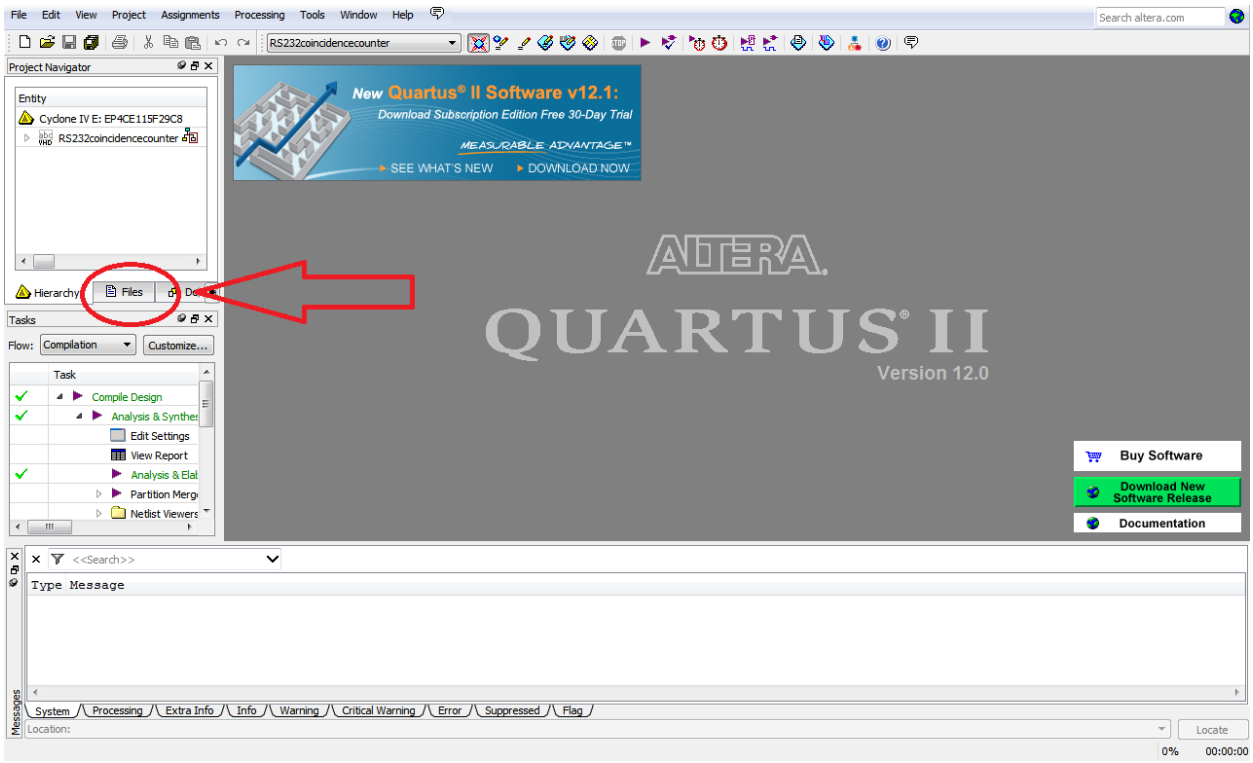
To open the project files, navigate to the folder titled "UofT_Project_Files" which is found on the desktop of the lab computer. Inside this folder are all of the VHDL (.vhd) files, the Quartus project file (.qpf) along with other various files that Quartus uses when compiling. Additionally there is a folder titled "DE2_Systems_Disc" which contains all the basic information needed to run Quartus and use the

DE2-115. If ever you are confused, all of the documentation for the DE2-115 is contained within that folder, and should help answer your questions.

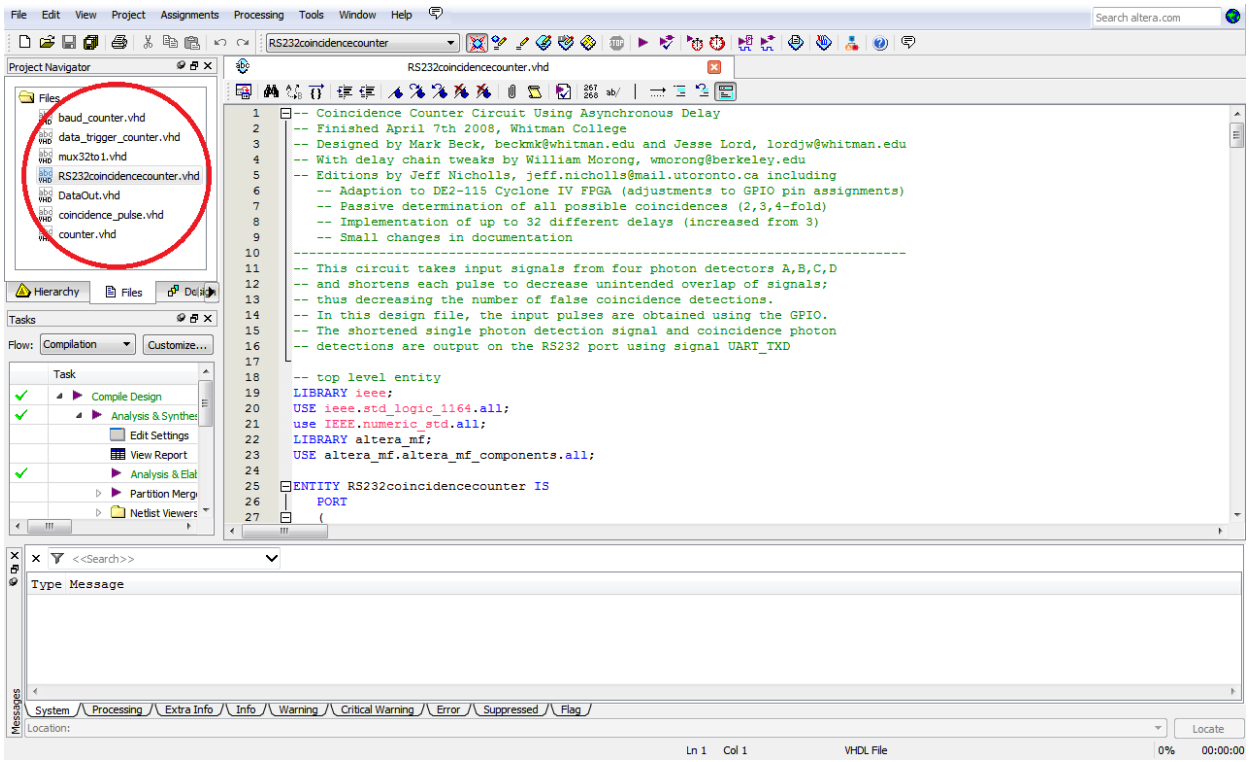
To open the project, double click the Quartus project file "RS232coincidencecounter.qpf". This will launch Quartus and open the entire project allowing access to all of the design files including pin assignments and device assignments. You should get a window which looks like the following.



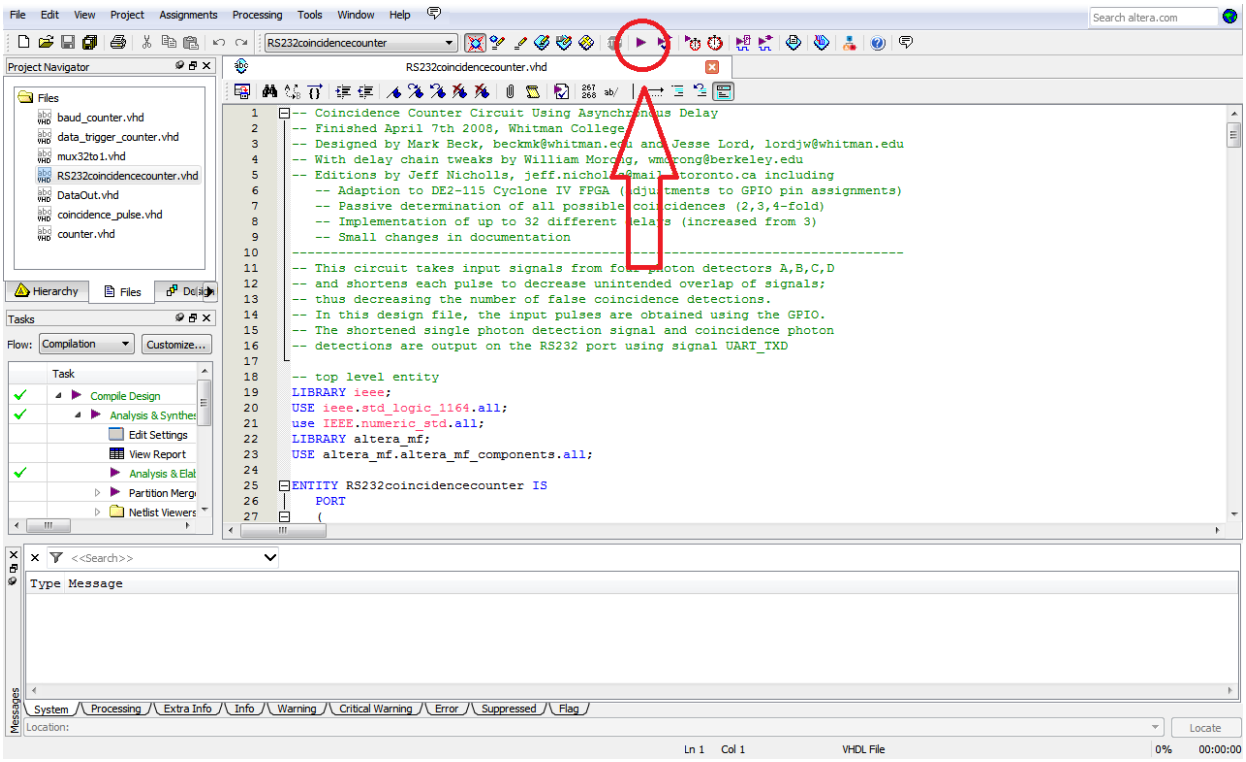
To open the files for editing, click the files button on the left side of the screen as indicated in the following picture.



A list of files will appear above the tab you clicked. By double clicking on one of the files you can edit them using Quartus' built in editor. The top level file is titled "RS232coincidencecounter.vhd" and would be the best place to start if you are familiarizing yourself with the code.



After making any changes, you need to first save, then compile the code. To compile, click the purple "play" button on the top menu bar of Quartus as shown below.



The code will take a while to compile as it not only compiles the code but synthesizes it and routes it onto the FPGA (virtually). If any errors were made during the changes that were applied, Quartus will halt compilation and give an error. Errors can be viewed in the bottom message panel. Quartus will also give many warnings which can be ignored. Some common warnings which can be safely ignored are:

- Warning (20028): Parallel compilation is not licensed and has been disabled
- Warning (19016): Clock multiplexers are found and protected
- Warning (13024): Output pins are stuck at VCC or GND
- Warning (21074): Design contains 4 input pin(s) that do not drive logic
- Warning (20028): Parallel compilation is not licensed and has been disabled
- Warning (292013): Feature LogicLock is only available with a valid subscription license. You can purchase a software subscription to gain full access to this feature.
- Warning (15714): Some pins have incomplete I/O assignments. Refer to the I/O Assignment Warnings report for details
- Warning (15705): Ignored locations or region assignments to the following nodes
- Warning (171167): Found invalid Fitter assignments. See the Ignored Assignments panel in the Fitter Compilation Report for more information.
- Critical Warning (332012): Synopsys Design Constraints File file not found: 'RS232coincidencecounter.sdc'. A Synopsys Design Constraints File is required by the TimeQuest Timing Analyzer to get proper timing constraints. Without it, the Compiler will not properly optimize the design.

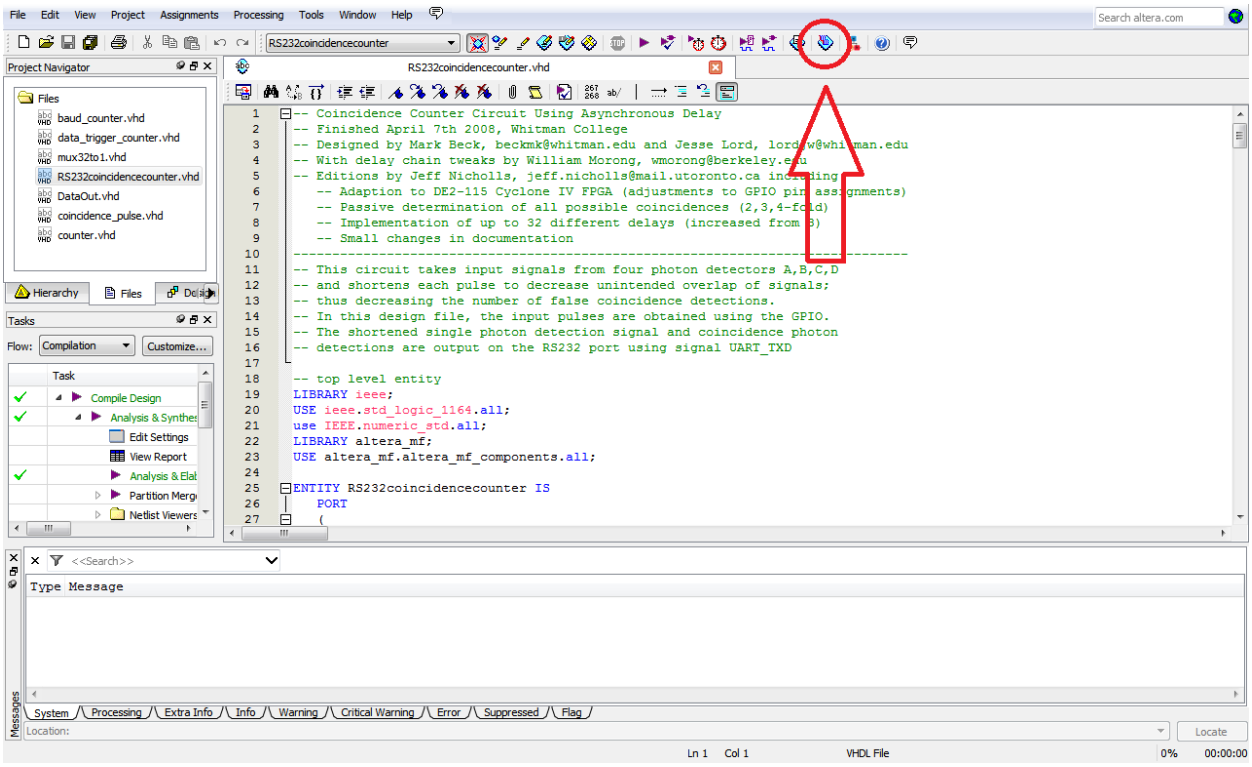
- Critical Warning (332148): Timing requirements not met

Note: Although timing requirements may seem to be important, none of the logic performed for this lab is synchronized to the clock, and hence there is no need to meet timing requirements.

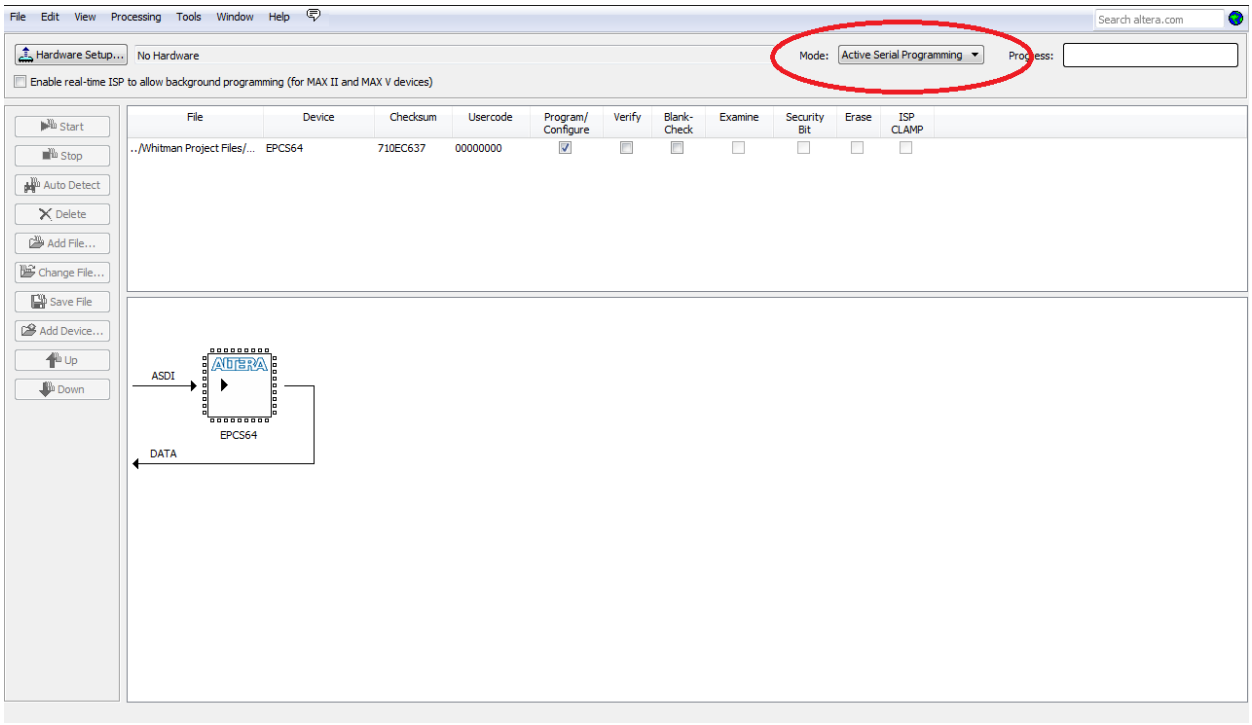
4. Program the FPGA

Assuming compilation was successful, the next step is to program the FPGA itself. Connect the FPGA to the computer via the USB cable. Make sure that the cable is plugged into the USB connector on the FPGA titled "USB Blaster" which is the leftmost connector when looking at the board from above with the switches at the bottom. Once it is connected, set the RUN/PROG switch on the FPGA to the PROG position. The switch is located on the left-hand side just above the set of 17 switches (but is not one of the 17 switches itself). Once the switch is in the proper position, turn the FPGA on via the red power button. The red LEDs should give off a dull red glow which shows that it is indeed set to the PROG mode. The RS232 light should also no longer flicker.

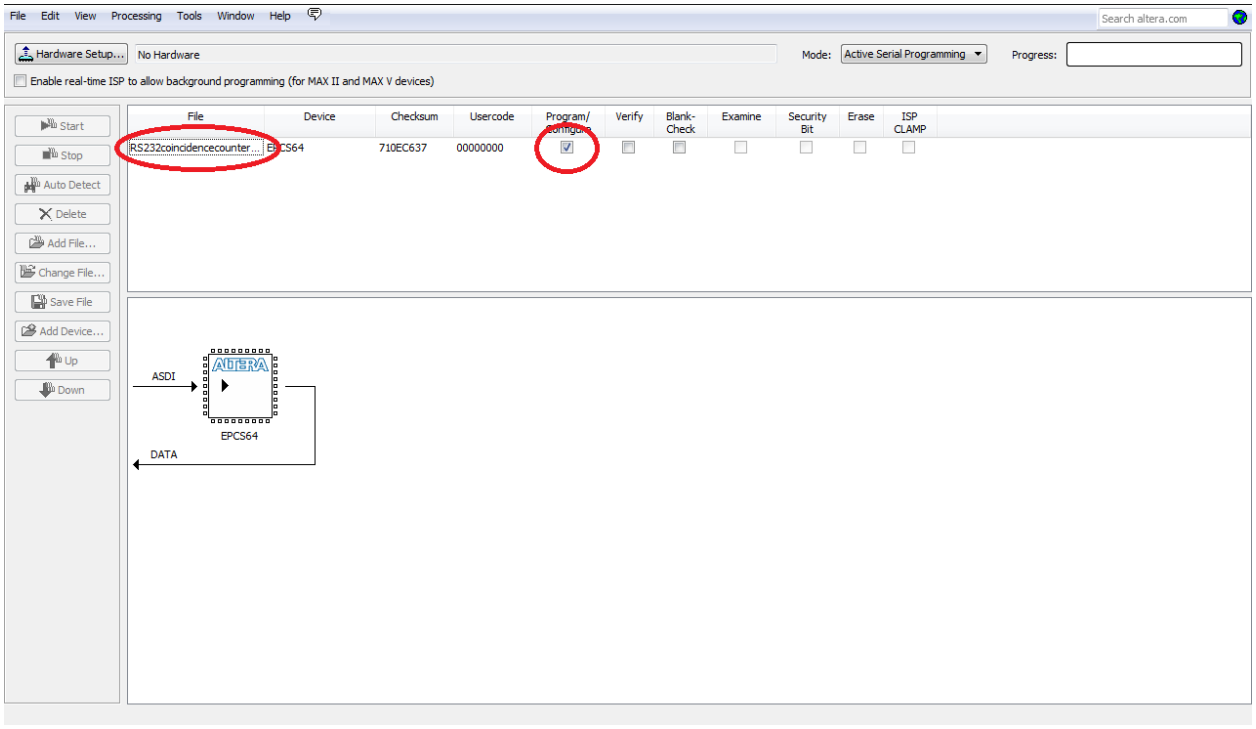
Next, click the Programmer button on Quartus found to the right of the compilation button as shown in the following picture.



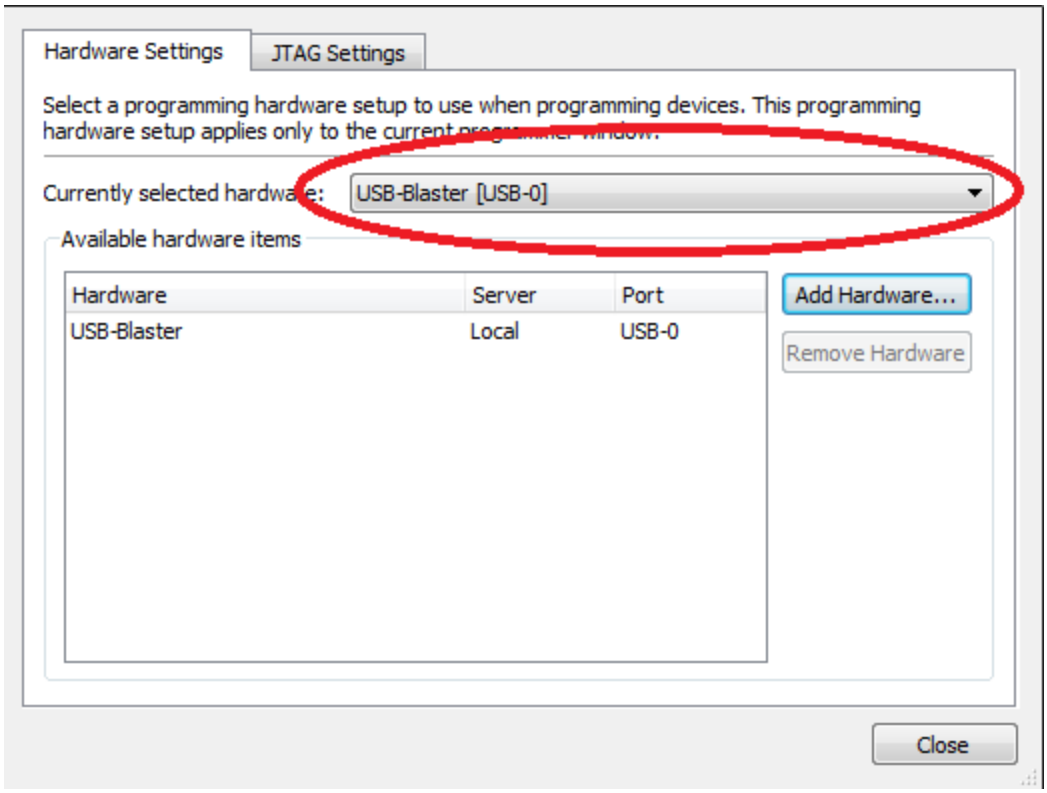
The following window will appear. Ensure that the drop down menu titled "Mode" indicates "Active Serial Programming".



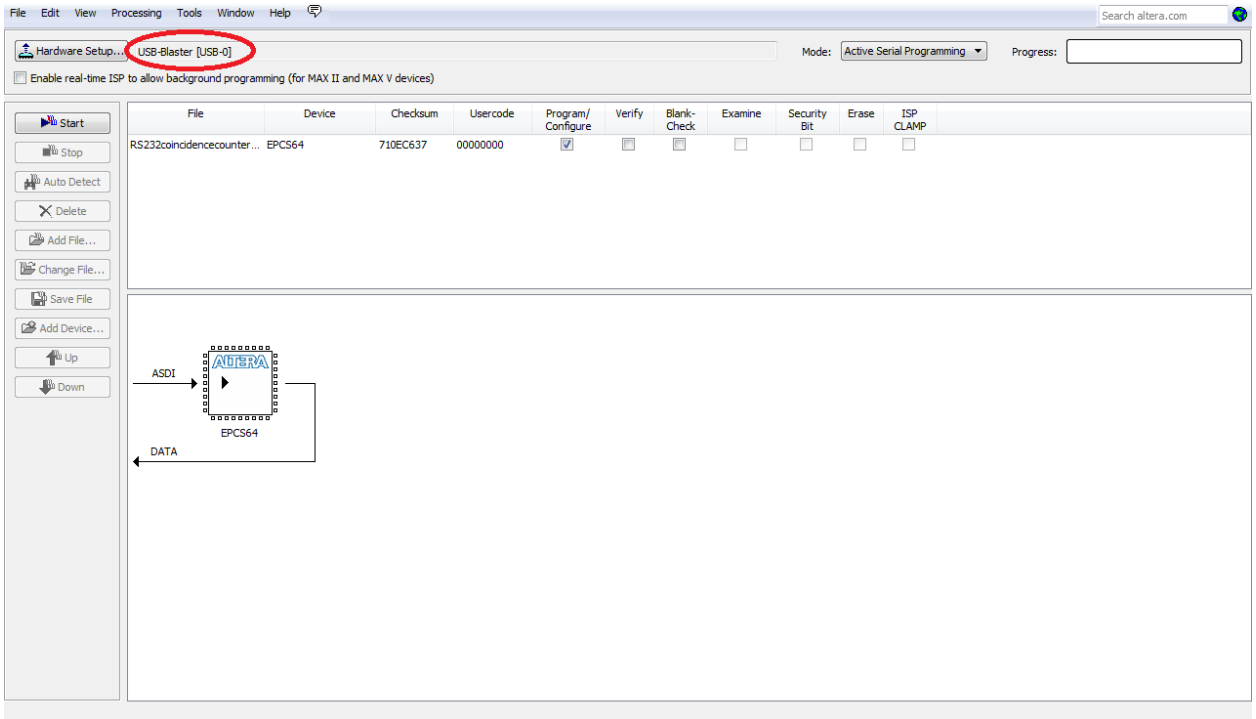
Ensure that the file selected is the RS232coincidencecounter.pof and that the "Program/Configure" box is checked as in the following picture.



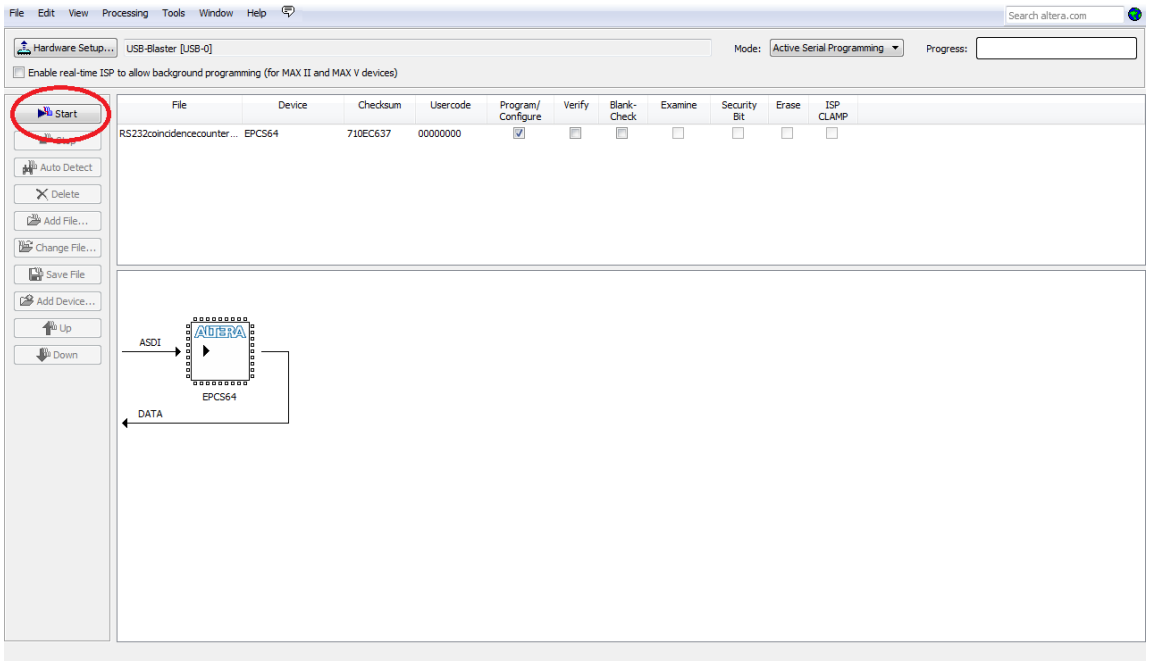
Finally, check to make sure that beside the "Hardware Setup" button, the "USB-Blaster" is labeled. If not, click the "Hardware Setup" button. The following window will appear.



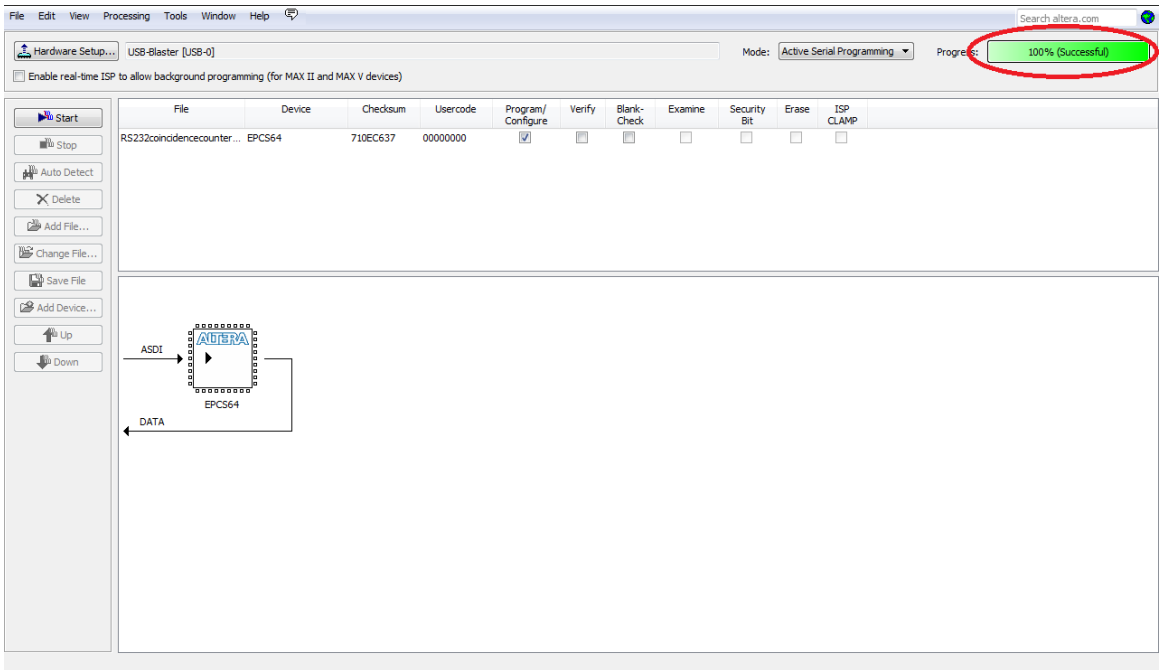
Select the "USB-Blaster" from the "Currently selected hardware" dropdown menu and click close. If the "USB-Blaster" option is not available, ensure that the USB cable is connected to both the computer and the DE2-115 USB –Blaster port. Once the "USB-Blaster" is selected, close the menu. The "USB-Blaster" should now appear beside the "Hardware Setup" button.



Once that is completed, click the “Start” button on the left side of the screen.



A green light will appear next to the USB-Blaster on the DE2-115 and a progress bar will start making progress at the top right of the screen. When it is finished, the progress bar will read “100% (Successful)”.



Now simply turn the DE2-115 off. Put the RUN/PROG switch in the RUN mode. Then turn the DE2-115 back on according to the procedure given at the start of the document. If the programming was successful, the green RS232 light should again be blinking. Now try it out!

Appendix: Schematic Overview of Logic