

EE 2420 Lab Guide 1: 3-Bit Adder

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

Example Overview:

Addition is a basis for all computer design applications. Addition takes up roughly 30 to 40 percent of the instructions in your average CPU. Thus, it is important to understand how addition works. In this lab, you will use logic gates to design a circuit that will add two 3-bit numbers.

Figure 1 and Figure 2 are examples of the gate logic you will be creating in Verilog.

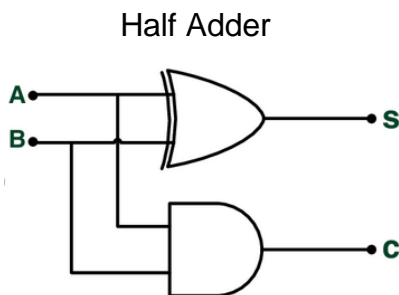


Figure 1

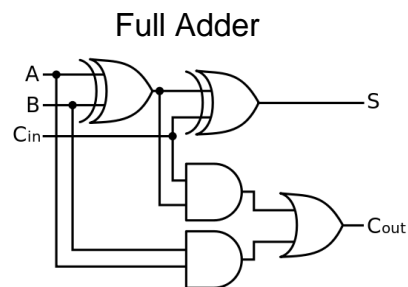


Figure 2

Verilog Overview:

For this course all Verilog will be presented as structural level. This is simply gate level design of the logic circuits. You can create all your gates using the gate function calls as seen **Figure 3** below. Take note that every Verilog logic circuit created is called a **module**. Think of a module as a logic block representing your code. This will be crucial in understanding how Verilog is imported into a hardware system such as the Arrow Max1000 FPGA. Each module will have inputs and outputs. These can either be declared as a **wire** or a **reg**. Wires can be internal or external connections made to other modules. Regs or registers are connections that are “driven” meaning they will carry data or be active to other modules. There are a lot of resources to help you better visualize this. Below, are three sites that can help you get started:

<http://www.asic-world.com/>

Provides information on Verilog language itself and code tutorials.

<https://software.intel.com/en-us/fpga-academic/learn/tutorials>

For in-depth documentation and tutorials on Intel Quartus and ModelSim.

<https://www.fpga4student.com/>

Provides Verilog information and hardware projects.

Verilog Code Breakdown:

We start by putting the first two bits through a half adder. The half adder has two inputs and the sum and carry as outputs. The carry will be sent to the next adder and added with the middle two bits. Another carry will be tied to the second full adder and is summed with the final two bits. The output will be the sums from each adder and a carry from the last adder.

```
1 //3-bit Adder in Verilog
2 /*
3  AUTHOR: GABE GARVES
4  DATE: 10/30/2019
5  FROM: TXST SENIOR DESIGN PROJECT FALL 2019
6  FOR: TEXAS STATE UNIVERSITY STUDENT AND INSTRUCTOR USE
7  */
8 //Structural Code
9
10 module adder(sum, cout, a, b); //Port list
11     output wire[2:0] sum; //array of length 3 for sum
12     output wire cout; //Final carry
13
14     input [2:0] a, b; //array of length 3 for inputs
15
16     wire w[5:0]; //declaring 6 wire connections
17     wire c[1:0]; //declaring intermediate carries
18
19     //gate var_name(output, inputs);
20
21     xor xor_1(sum[0], a[0], b[0]); //First adder module
22     and and_1(c[0], a[0], b[0]); //Half Adder
23
24     xor xor_2(w[0], a[1], b[1]); //Second adder module
25     xor xor_3(sum[1], c[0], w[0]); //Full Adder
26     and and_2(w[1], a[1], b[1]);
27     and and_3(w[2], c[0], w[0]);
28     or or_1(c[1], w[1], w[2]);
29
30     xor xor_4(w[3], a[2], b[2]); //Third adder module
31     xor xor_5(sum[2], c[1], w[3]); //Full Adder
32     and and_4(w[4], a[2], b[2]);
33     and and_5(w[5], c[1], w[3]);
34     or or_2(cout, w[4], w[5]);
35
36 endmodule
```

Figure 3

FPGA Implementation:

Intel Quartus is the final step into programming the FPGA. Quartus along with ModelSim will allow one to simulate, verify and transfer verilog into the FPGA. This guide will run through the entire process to get the 3-bit adder working on the Arrow Max1000.

Creating a Quartus Project:

Before opening a Quartus make sure to **create a new folder for each project**. This will keep the files Quartus creates organized and is recommended for ease of use. Then open up Quartus and you should see a window similar to **Figure 4**.

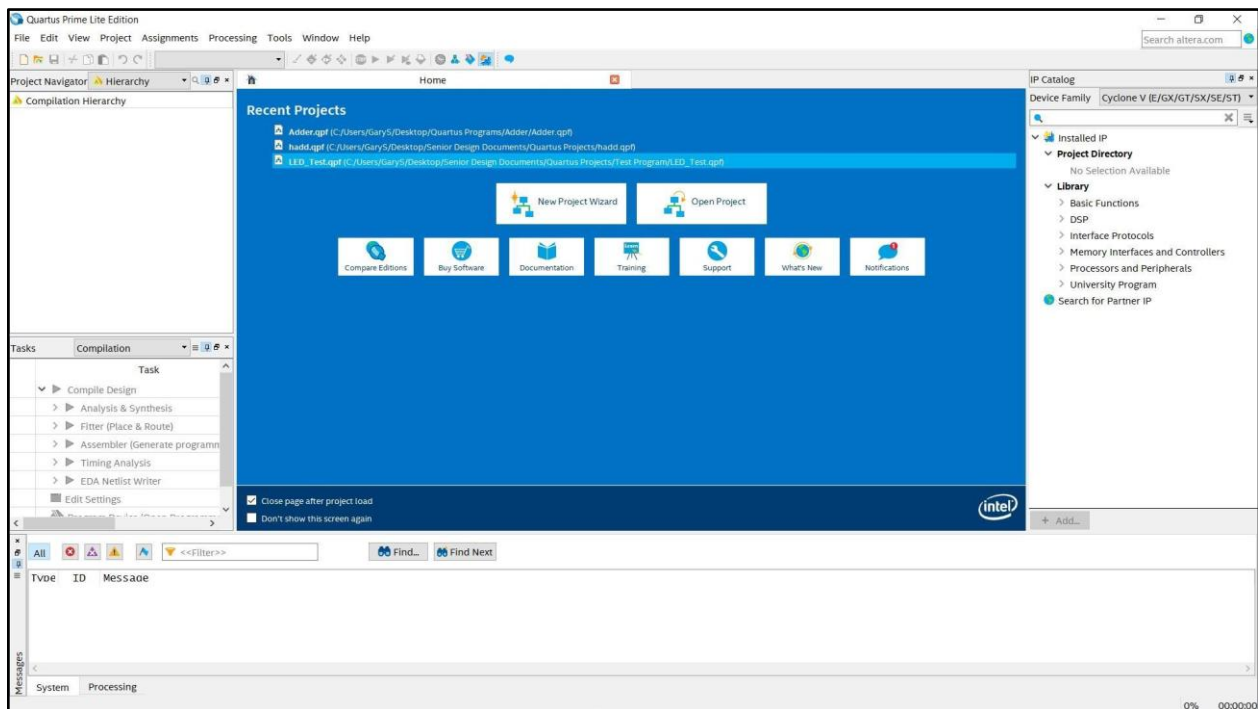


Figure 4

Next, click **New Project Wizard**, click next until you get to **Directory, Name, Top-Level Entity** window (**Figure 5**). Name the project **Adder** and then select the directory folder you created earlier. Click **Next** until you reach **Family, Device & Board Settings** window.

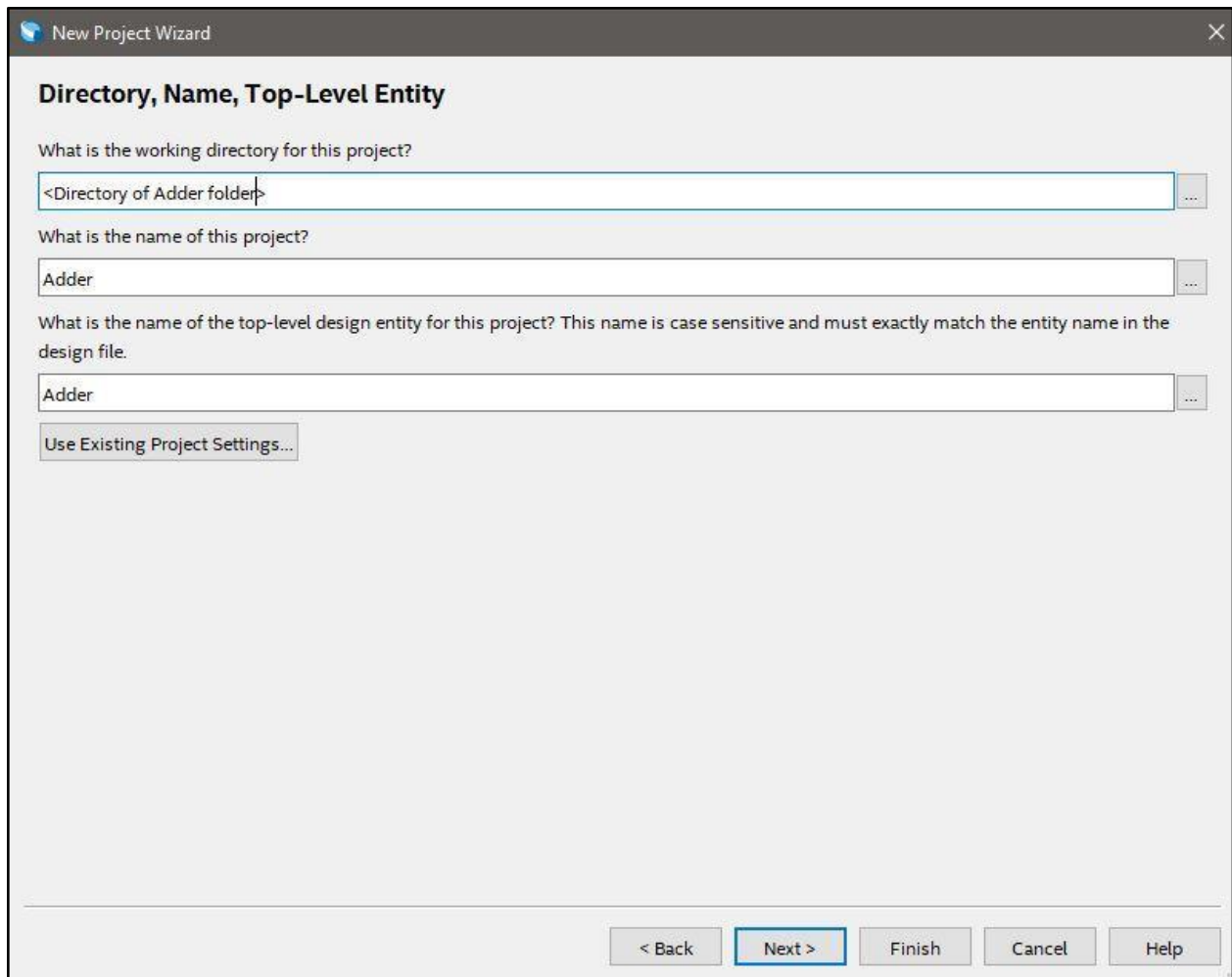
The image shows a 'New Project Wizard' dialog box with a title bar containing a blue icon and the text 'New Project Wizard' and a close button. The main area is titled 'Directory, Name, Top-Level Entity'. It contains three text input fields with labels: 'What is the working directory for this project?' (containing '<Directory of Adder folder>'), 'What is the name of this project?' (containing 'Adder'), and 'What is the name of the top-level design entity for this project? This name is case sensitive and must exactly match the entity name in the design file.' (containing 'Adder'). Each field has a browse button (three dots) to its right. Below these fields is a button labeled 'Use Existing Project Settings...'. At the bottom right are five buttons: '< Back', 'Next >' (highlighted with a blue border), 'Finish', 'Cancel', and 'Help'.

Figure 5

Here you are setting up the file path for your project. For each lab, make sure to make a new project so you don't risk working in a previous project or having directory path location issues later.

Look at **Figure 6**, first in the **Name filter** box type in "**10M08SAU169C8G.**" Second click on first entry in the table below, then click next until finish.

New Project Wizard

Family, Device & Board Settings

Device Board

Select the family and device you want to target for compilation.
You can install additional device support with the Install Devices command on the Tools menu.

To determine the version of the Quartus Prime software in which your target device is supported, refer to the [Device Support List](#) webpage.

Device family

Family: MAX 10 (DA/DF/DC/SA/SC)

Device: All

Target device

☐ Auto device selected by the Fitter

☒ Specific device selected in 'Available devices' list

☐ Other: n/a

Show in 'Available devices' list

Package: Any

Pin count: Any

Core speed grade: Any

Name filter: 1. 10M08SAU169C8G

☐ Show advanced devices

Available devices:

Name 2.	Core Voltage	LEs	Total I/Os	GPIOs	Memory Bits	Embedded multiplier 9-b
10M08SAU169C8G	3.3V	8064	130	130	387072	48
10M08SAU169C8GES	3.3V	8064	130	130	387072	48

< >

< Back Next > Finish Cancel Help

Figure 6

This window is where you tell Quartus what hardware you will be working with. If you installed Quartus Lite correctly, The Max 10 device family should be present within Quartus' internal directory files.

Importing and Checking the Verilog File

To create a new project: **File>>New** then a window will open (**Figure 7**) which will allow you to create a variety of programs that are linked with Quartus. For now, select the **Verilog HDL File**. This will create a new (.v) file within Quartus or the 3-Bit Adder code file can be imported by **File>>Open>>3-Bit Adder.v**

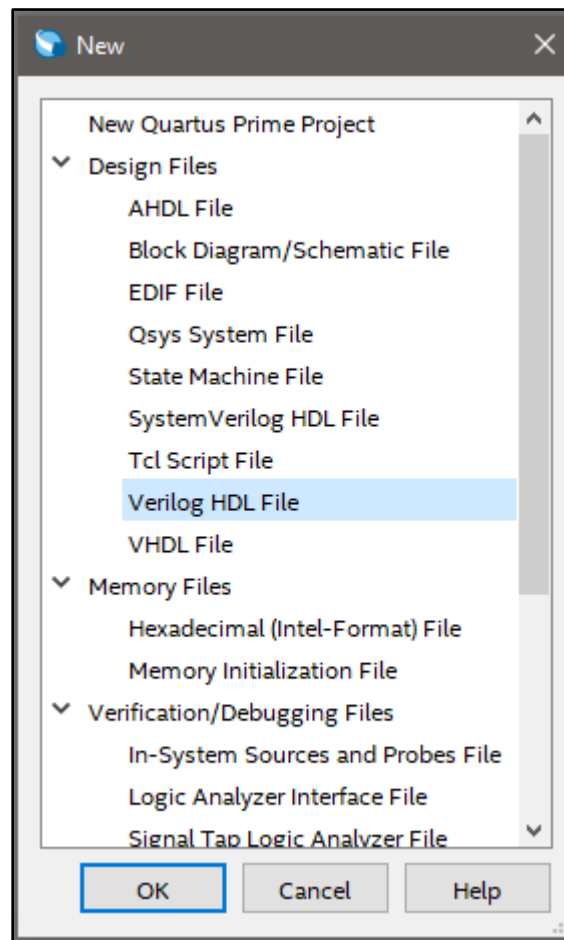


Figure 7

Quartus is basically a set of programs that can do a wide range of design analysis. You can potentially create all your programs, compile, simulate, and export hardware programs all within Quartus.

Once you either have imported or copied over the 3-bit adder file the window should look similar to **Figure 8**.

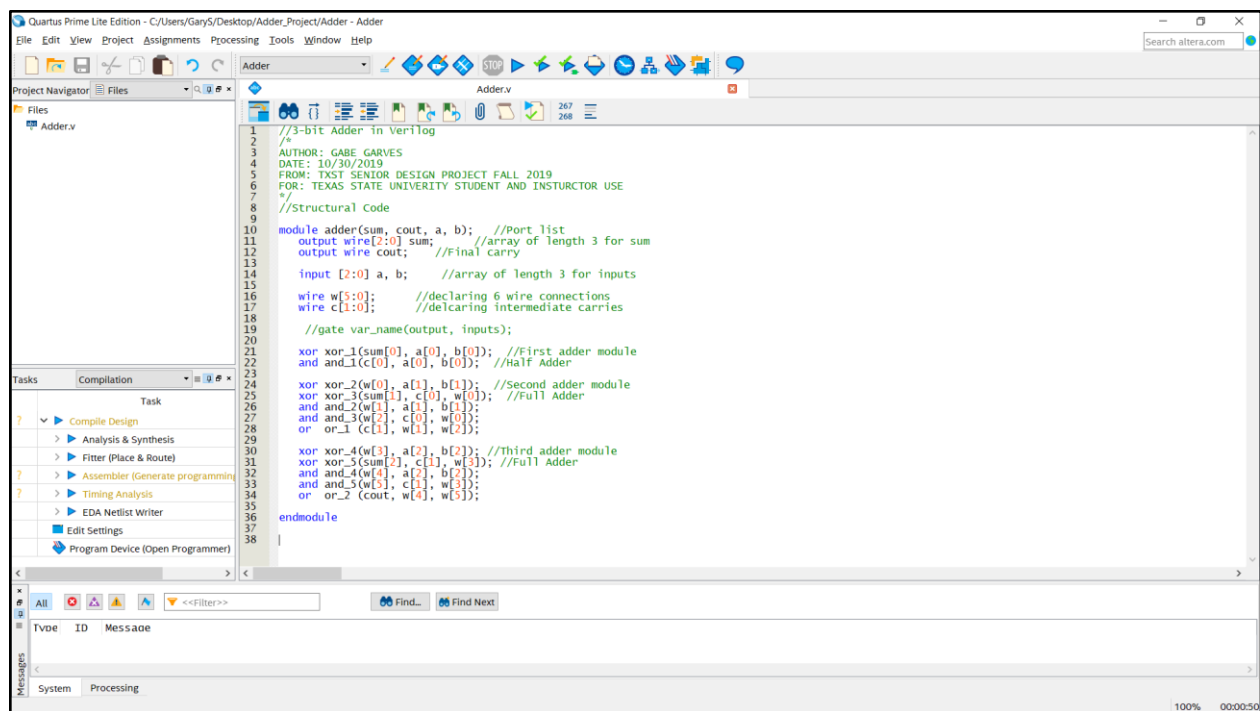


Figure 8

Creating a Symbol File

A symbol file is the hardware block representation of a module. Similar to if you were to create a new model of a circuit or component in either MultiSim or LTSpice. While Adder.v is open click **File>>Create / Update>>Create Symbol Files from Current File**. A message window will notify you if the process was successful (Alt+3 brings up the message window if not already showing).

When a **Symbol File** is created, Quartus checks for errors in the verilog file in question. So if there are any errors that are present they will show up in the **Messages** tab. This is usually located at the bottom of the screen like in **Figure 8**.

Creating and Wiring the Schematic File

In order to connect a symbol file to specific I/O pins or other symbol files, you will need to create a Block Diagram file (**.bdf**). To do this, click **File>>New>>Block Diagram/Schematic File**. This opens a new **.bdf** tab in Quartus and is like a virtual breadboard where you can add and interconnect symbol files. **Right click** on the Block Diagram window and click **Insert>>Symbol**, inside the Libraries window click **Project>>Adder**. Press okay and place the Adder symbol on the schematic. As you can see the Adder symbol you created from the Verilog file has the inputs and outputs defined from the code. Inputs will be on the left and outputs on the right. **Note:** When creating a Verilog file, make sure to label your I/O with relevant names. (**Figure 9**) This will make it easier for you to connect all the I/O.

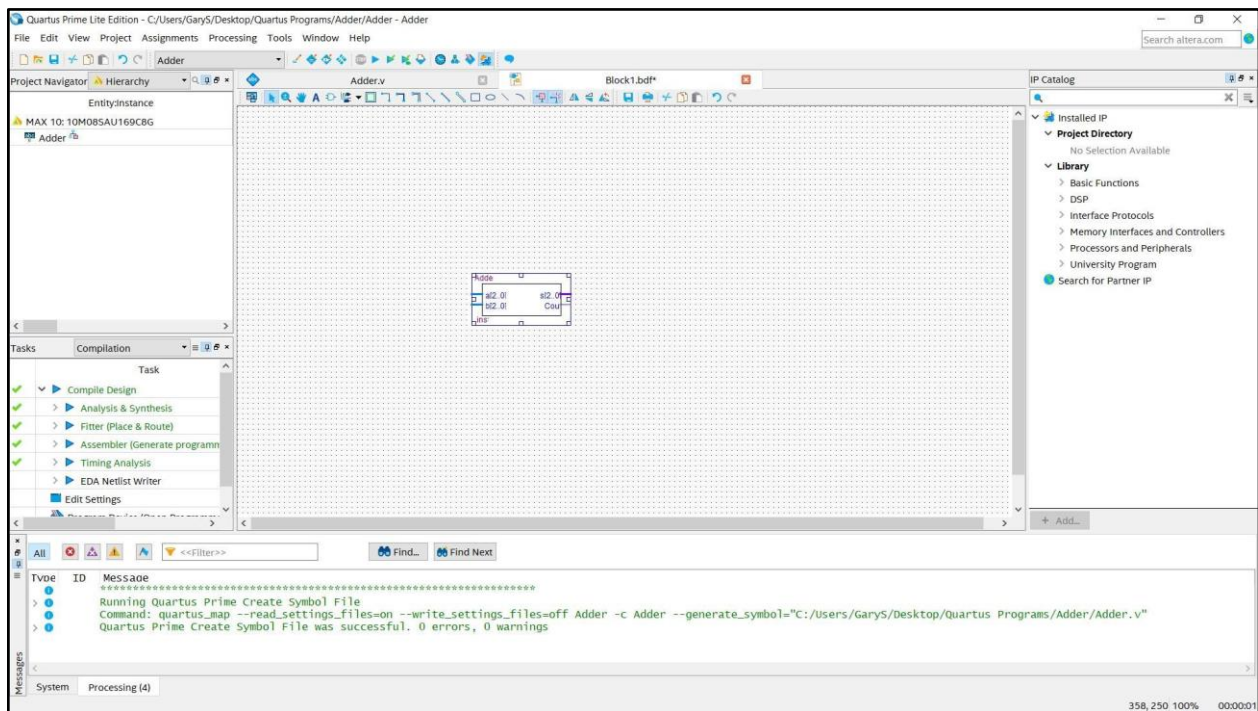


Figure 9

*The .bdf file will allow you to configure all your I/O for your symbols. Quartus also has built in symbols for basic functions, clocks, etc. To add them into your grid, just view the drop-down menus on the **Library** section to the right in **Figure 9**.*

On the toolbar click on the drop-down arrow next to the **Pin Tool (Figure 10, Blue Box)**. Place two inputs and two outputs on their respective sides. You can rename the I/O pins by **right clicking** the **Pin>>Properties** then typing in the desired name into the **Pin name(s)** text box. Press **Ok** to save changes.



Figure 10

*As a note, to exit/deselect out of the current design tool, press the **ESC** key.*

Now you need to connect I/O pins to the I/O terminals on the Adder symbol. Input A, B, and output Sum will all need the **Bus Tool (Figure 10, Green Box)** because they are composed of multiple wires. Since Cout is composed of only one wire, you will use the **Orthogonal Node Tool (Figure 10, Red Box)**. You can use the Diagonal version of the Bus and Node tool if you wish. After all I/O has been wired, the schematic file should look similar to the one in **Figure 11**.

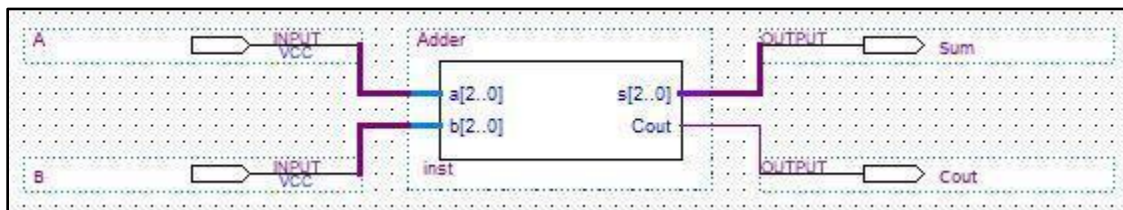



Figure 11

Compiling the Project

Once you have created your circuit, you need to compile the project. Compiling will check all the files for final errors and then set the initial pin I/O for the **Pin Planner** which we will get to in the next section. To compile: Go to the top menu bar and click:

Processing >> Start Compilation. Or you can press  button. If this initial compilation is successful you will see something similar to **Figure 12**.

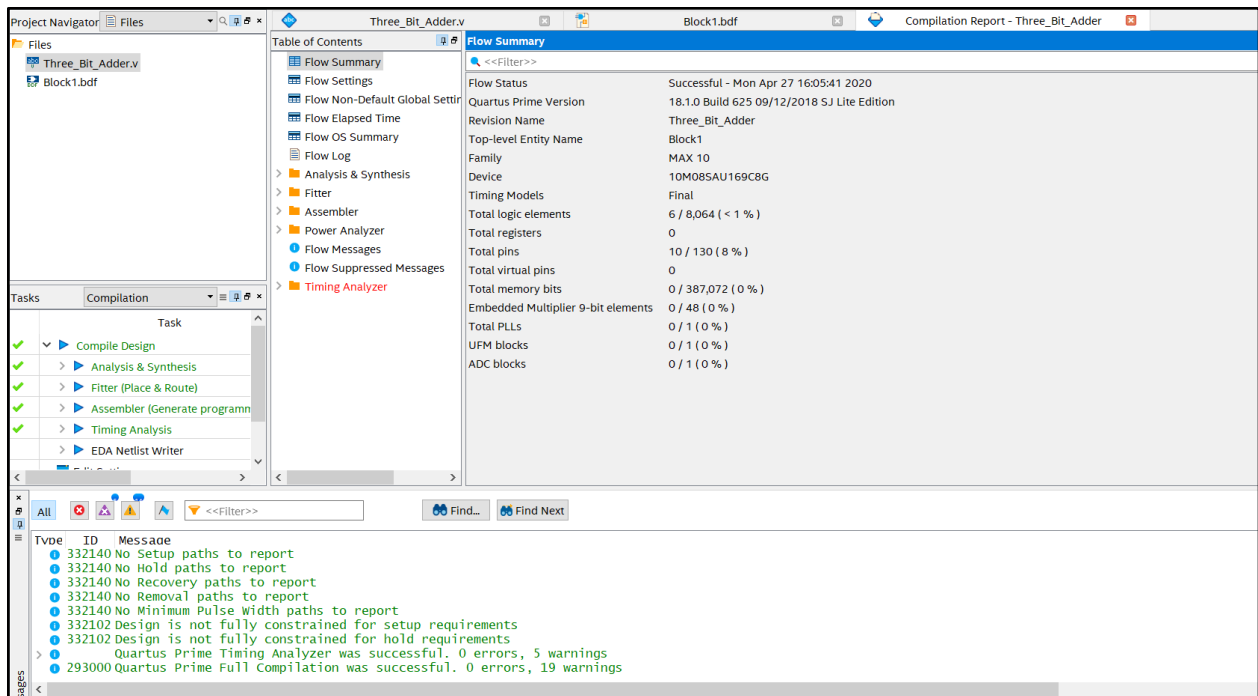


Figure 12

Common Issue: Top Level Entity Layer is Undefined

During compilation at some point you will get an error in the console similar to **Figure 13**.

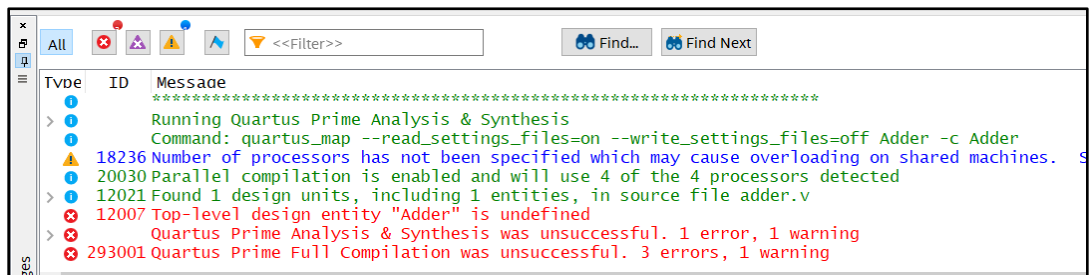


Figure 13

When Quartus compiles, it looks for a specific file to compile. Usually, most Quartus Projects have a main **Block Diagram/Schematic file** and Quartus gets confused sometimes on which file in the hierarchy to pick. To fix this issue, Go to the **Project Navigator** window and use the drop down menu and select **Files**. This will show you all the current files within your Quartus Project. **See Figures 14-16**.

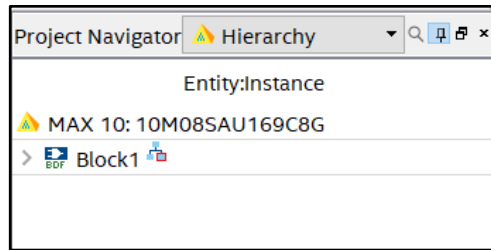


Figure 14

Right-click on the **.bdf** file you are using for your project and select **Set as Top-Level Entity**.

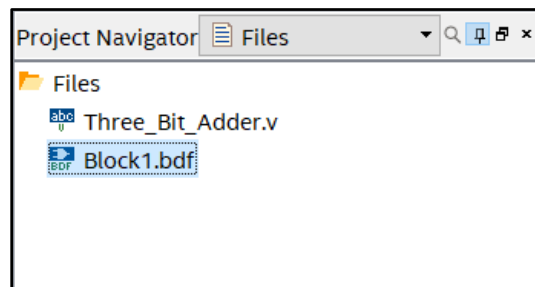


Figure 15

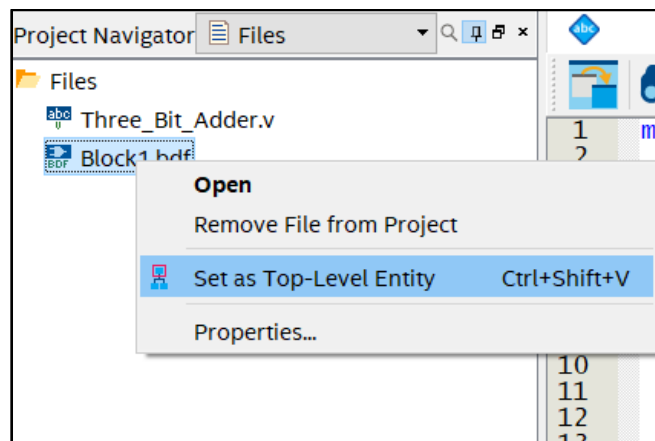


Figure 16

Once you have set the **Top Level Entity**, you can recompile the project and that error should go away.

Pin Allocation with the Pin Planner

Once your project successfully compiles, then launch the Quartus Pin Planner by clicking **Assignments** on the menu bar then **Pin Planner (Figure 17)**. In the **All Pins List** in the **Red Boxed** window, you can see what Nodes can be assigned a pin on the Max1000.

You can either click and drag the nodes to the desired pin location on the **Top View** or type the pin location in the **Location** field. The **Top View** grid uses letters for rows and numbers for columns. Use **Figure 18** as a guide to set the pins. Any of the orange pins can be used as inputs and the LEDs will be the outputs. **The orange pins names do not associate with the Pin Planner layout. The actual pin names are the smaller letters in Figure 18.** Insert the pin name in the **Locations** column in the **All Pins** window. After all pins have been set, exit out of the pin planner, all data saves automatically.

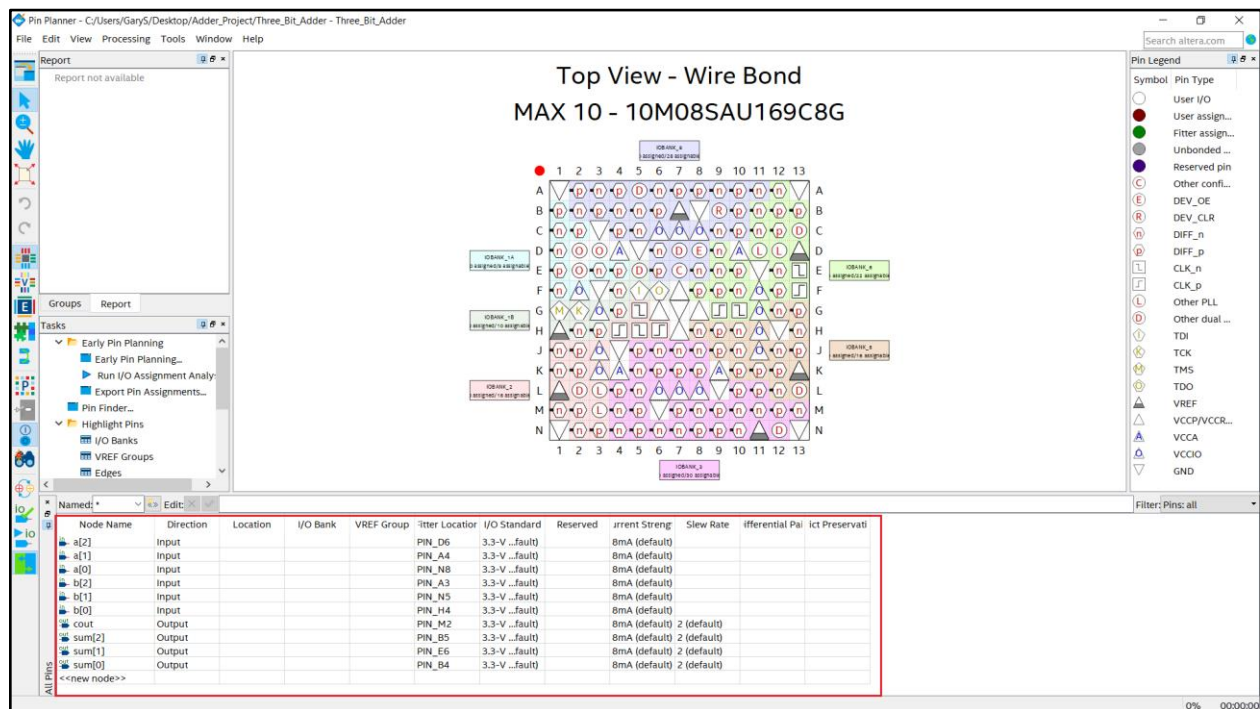


Figure 17

TEI0001-02 MAX1000

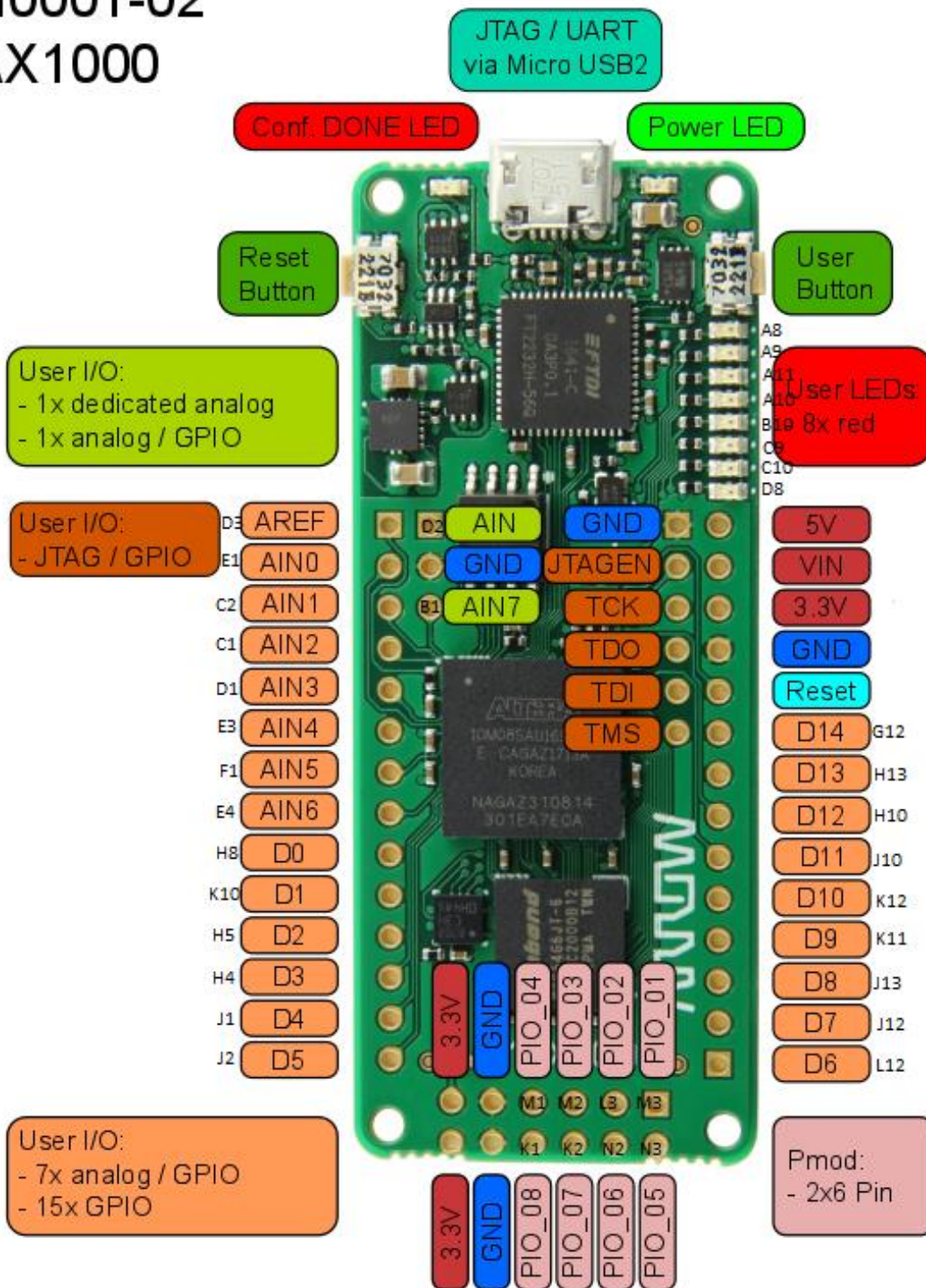


Figure 18

Lastly make sure operating conditions of the device are set correctly. Go to **Assignments>> Settings>>Operating Settings and Conditions>>Voltage** and set **VCCA** to **3.3V** then click Apply and Ok (Figure 19).

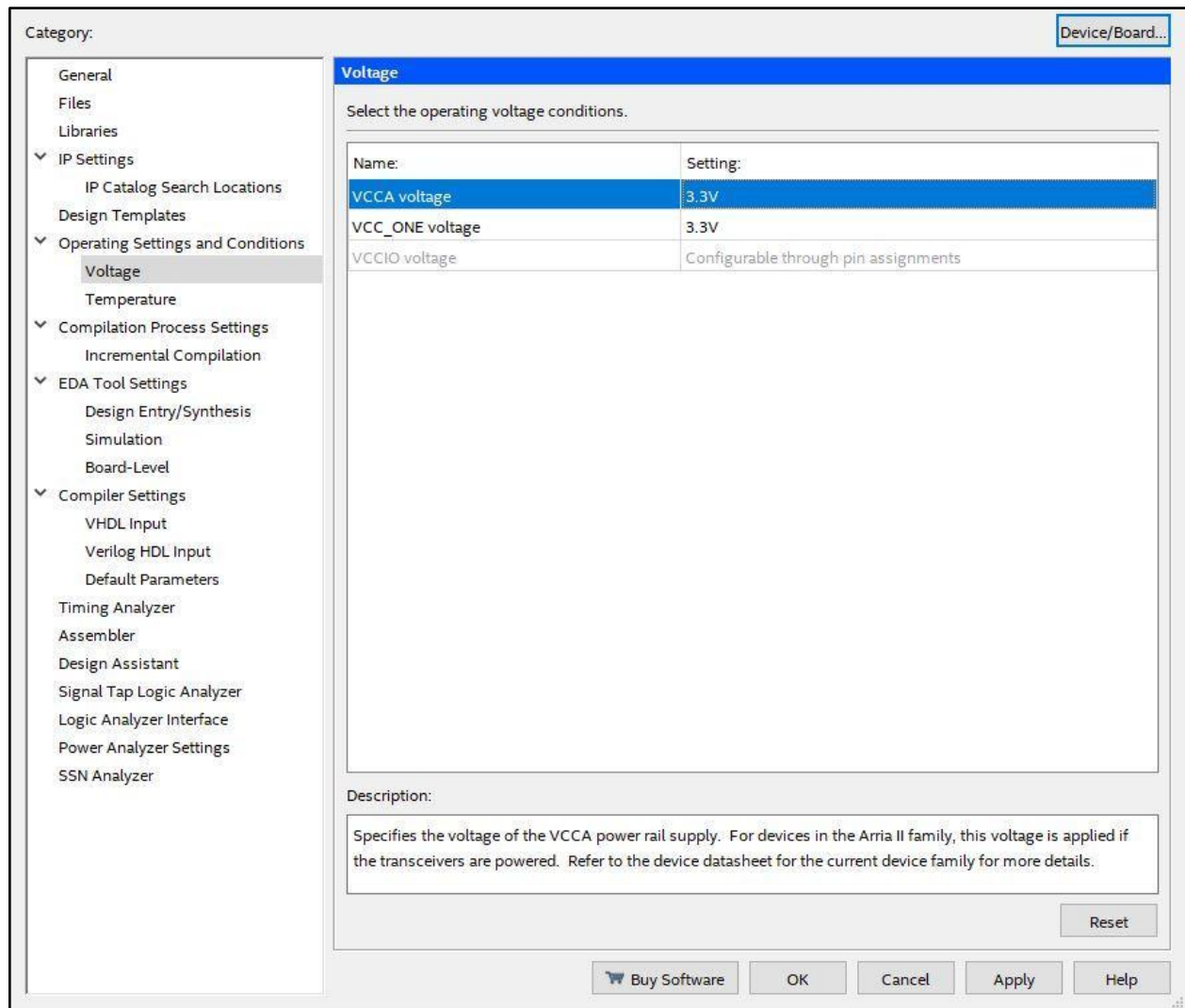


Figure 19

The Max1000 operates around 3.3 volts like many similar project boards. It does have a 5V output pin that comes off of the USB hub but there is no use for it in this application.

Now go to **Assignments>>Device>>Device and pin Options>>Voltage** and set **Default I/O standard to 3.3-V LVTTL** then click Ok (Figure 20 and 21).

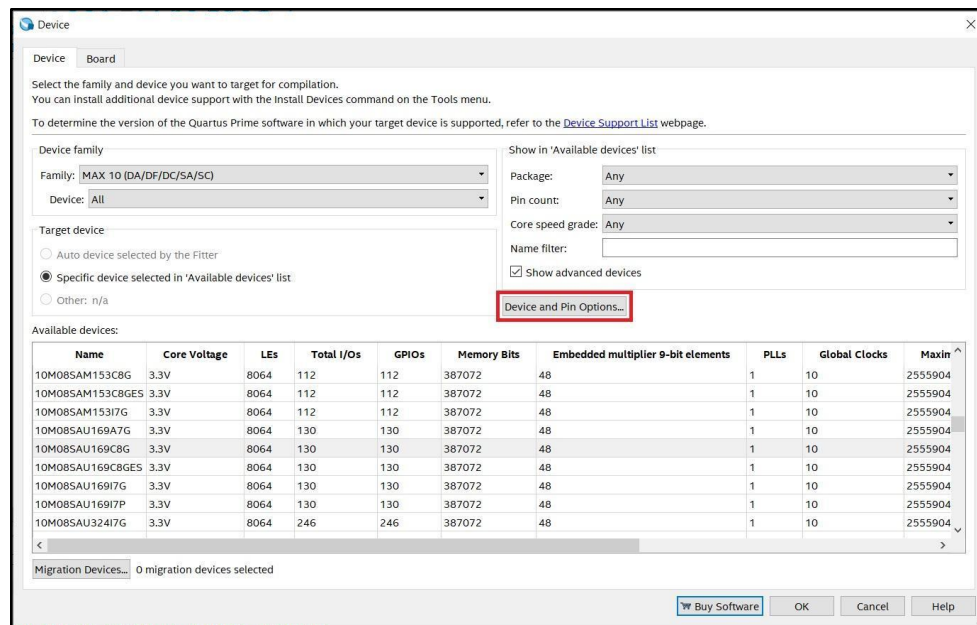


Figure 20

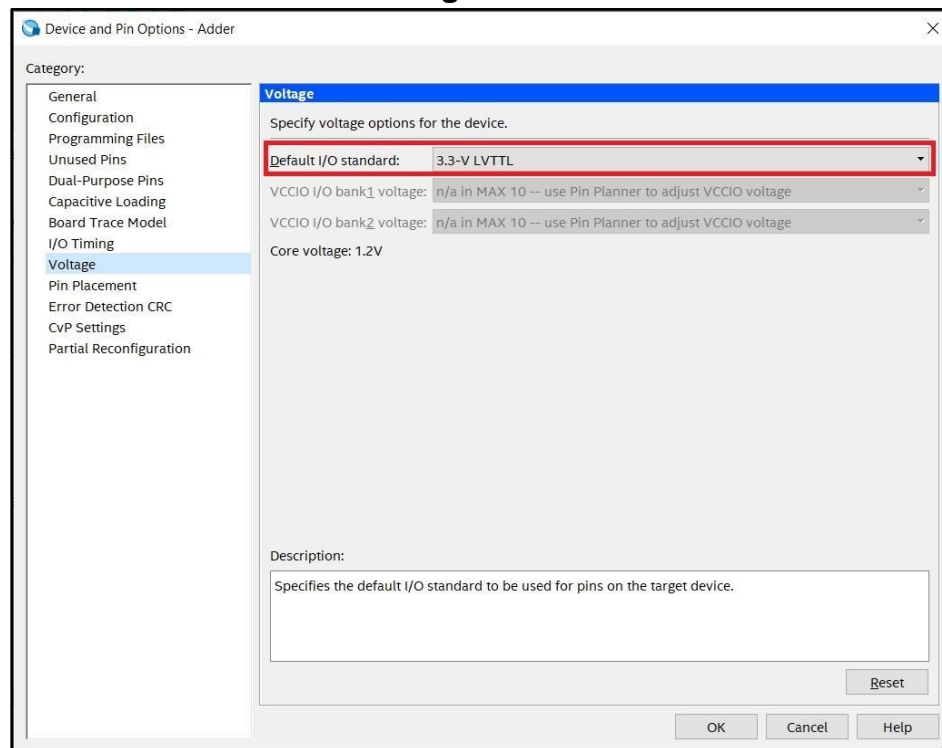


Figure 21

It's important to set the default I/O to 3.3-V LVTTL. If not, you may have power issues or damage the board or peripherals.

Now go to **Device and pin Options>>Unused Pins** and set **Reserve all unused pins** to **As output driving ground** then click **Ok (Figure 22)**.

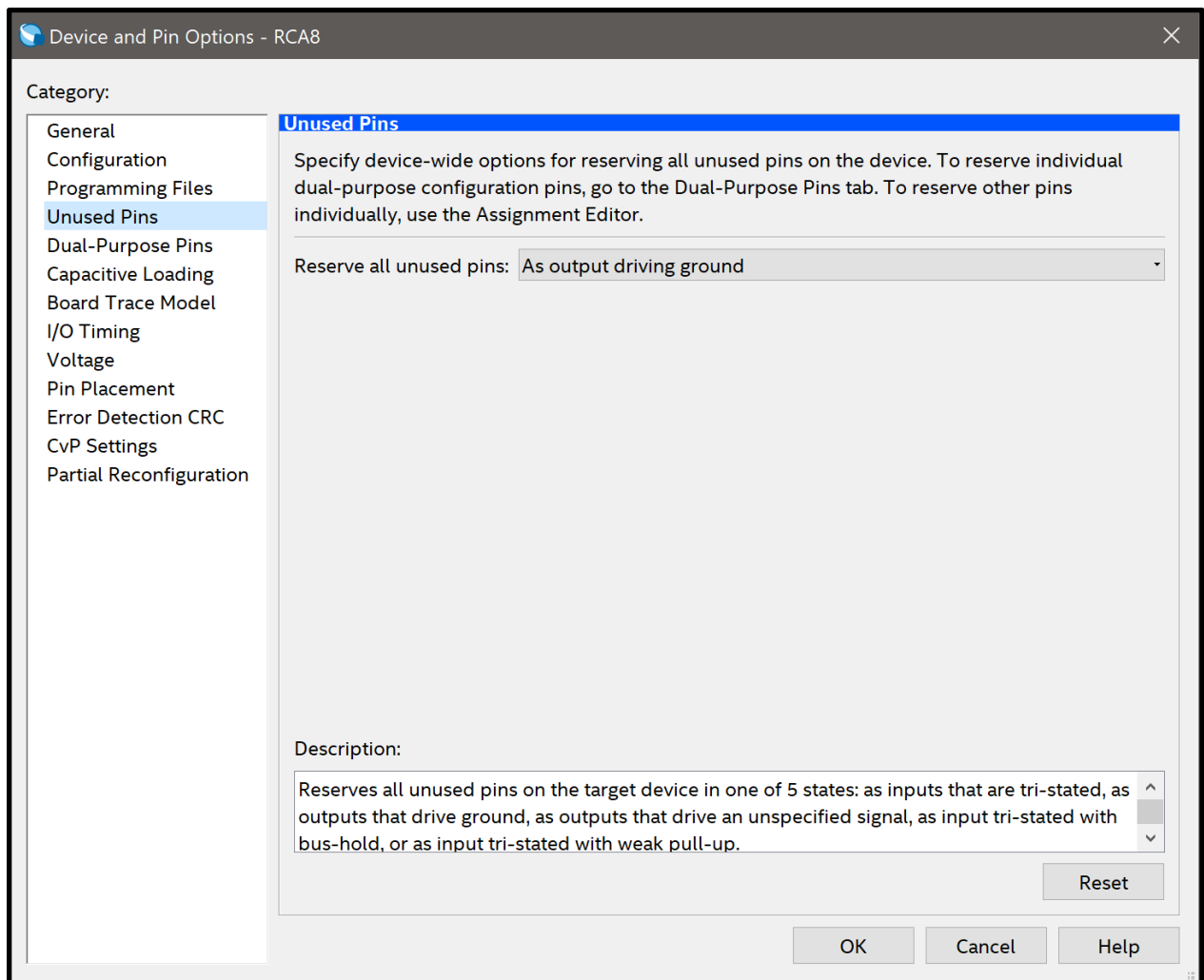


Figure 22

All unused pins are now grounded and will be off. If this is left undone, the LEDs on the Max1000 that are not in use will stay on. The pins can also be set to the tri-state option, which should ground all unused pins.

Programming the FPGA

Once all is compiled successfully, you can view the report which is full of interesting information. Flow Summary will show how many logic elements were needed, along with pins, registers, etc. You can look through the table of contents for loads of information regarding the synthesis of your design.

Now open the **Quartus Prime Programmer** by selecting **Tools>>Programmer**. Connect your Max1000 board to the PCB Peripheral provided and then to your computer using a USB cable. A green LED on the Max1000 should light up indicating power, along with a brief flash of a red LED. Click **Hardware Setup** and you should see **Arrow USB Blaster** listed in the **Available Hardware Items** table. If the Arrow USB Blaster isn't shown in the table contact the lab instructor. Click Ok to close this window.

Click **Auto Detect** and make sure the **Mode** is set to **JTAG**. Select **10M08SA** then click Ok. You will see an entry in the file select table generate. Click on the entry then select **Add File**. Navigate to **<Project Dir>/output_files/** and select **Adder.sof** or **Adder.pof**. SOF stands for SRAM Object Files, these are programmed directly to the FPGA fabric. They are volatile and will be erased when power is cycled. Select the checkbox under **Program/Configure**, then click **Start** to program the Max 10.

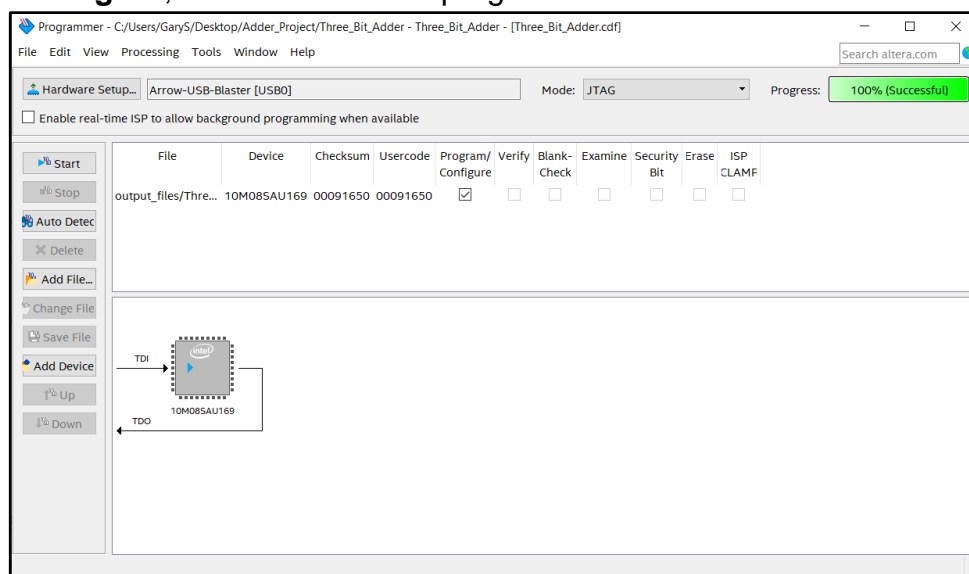


Figure 23

POF stands for Programming Object Files, these are programmed to the Max1000's Flash memory unit. They are non-volatile and will remain on the device even after a power cycle.

Working with the Peripheral Board

If you set your pins correctly and upload the program the FPGA LEDs should be working. 3 of the dip switches will represent the A inputs and 3 other switches will represent the B inputs. The sum of any combination of these switches should appear. On the 3 rightmost LEDs. Where the rightmost LED is the LSB. The fourth LED is your carry.

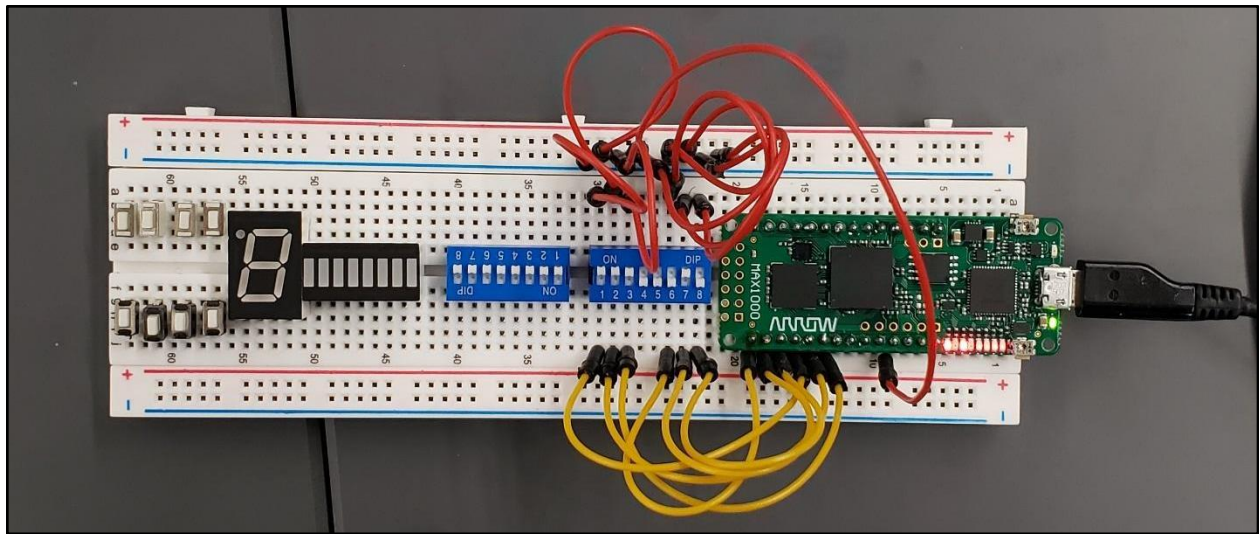


Figure 18

The pin layout on the FPGA can be changed to one's liking but make sure to reassign pins in Quartus and go through the steps to reprogram it. Also be sure to set the pins correctly depending on the type of I/O that is being used.