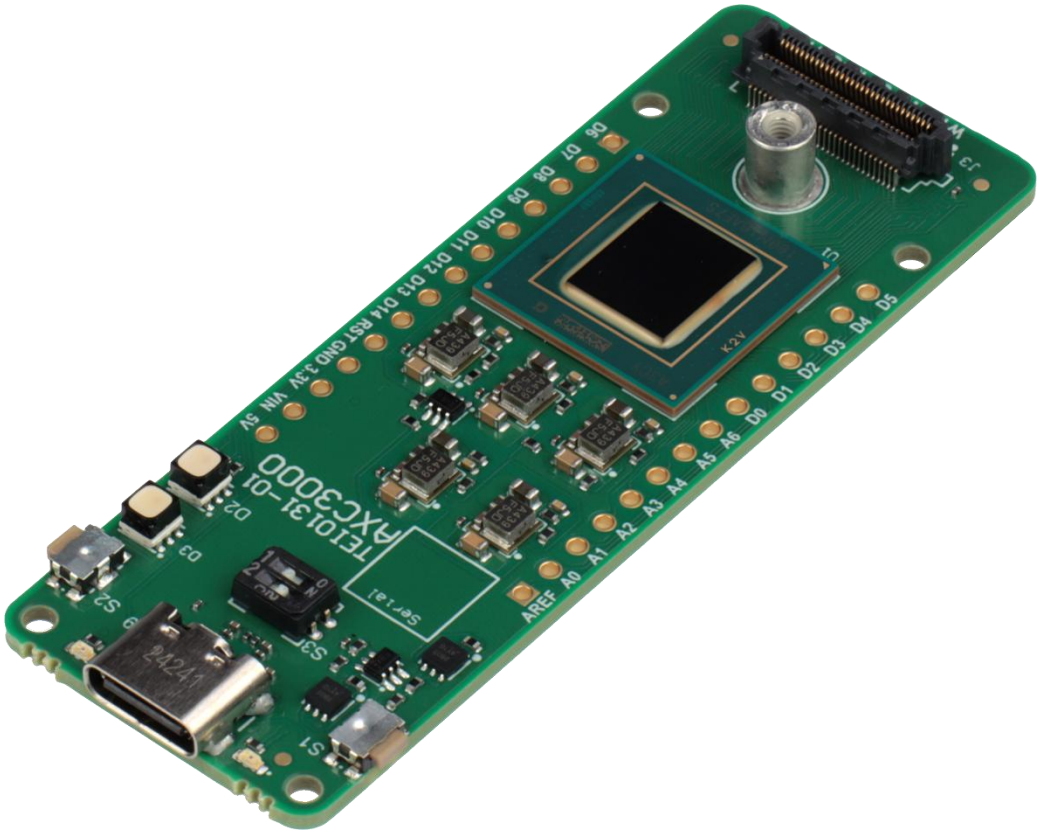
AXC3000 First Design Lab



<u>Software and hardware requirements to complete all exercises.</u>

Software Requirements: Quartus® Prime Pro version 25.1.1

Hardware Requirements: AXC3000 Agilex™ 3C Development Kit

Document Control

Document Version:	Version 1
Document Date:	11/04/2025
Document Author(s):	Steven Kravatsky, Erika Peter, Naji Naufel, ESC Team
Document Classification:	Released
Document Distribution:	This document is still under development. All specifications, procedures, and processes described in this document are subject to change without prior notice
Prior Version History:	Version: 0.1

Please read the legal disclaimer at the end of this document.

Table of Contents

1. Introduction 3
2. Getting Started 4
3. Design Flow 5
4. Project with AXC3000 Board 7
4.1 New Quartus Prime project 7
4.1.1 New project creation 7
4.2 Design entry 12
4.2.1 Add PLL to the Quartus Project 13
4.2.2 Add counter to the Quartus Project 19
4.2.3 Reset Release IP 22
4.2.4 Top-level file 25
4.3 Compile Design 27
4.3.1 Analysis and Synthesis 27
4.3.2 Pin Assignments 27
4.3.3 Timing Constraints 30
4.3.4 Compiling the Design 31
4.3.5 Compilation Report and Timing Results 33
4.3.6 RTL Viewer 36
4.4 Verify the Design Locally 38
4.4.1 File Download 38
Click on the ArrowDTD icon in CloudLabs to launch the tool. 38
4.4.2 Program the FPGA 39
4.5 Verify the Design Remotely 44
4.5.1 Connect to the Board Farm 44
4.5.2 Programming – Configuration 45
4.5.3 Testing the Design 46
4.6 Congratulations 49
5 Legal Disclaimer 50

1. Introduction

This workshop provides comprehensive information to help you

-understand the FPGA flow for design entry to programming

-create and parametrize IP (intellectual property)

-add assignments (such as pin assignments)

-run a complete compile

-view your design in the RTL Viewer

-review the resource and clock resources

and run the design onto the development board.

This lab will not make you an expert, but it will provide information to get you started.

Lab Notes: Many of the names that the lab asks you to choose for files, components, and other objects in this exercise must be spelled exactly as directed. This nomenclature is necessary because the pre-written software application includes variables that use the names of the hardware peripherals. Naming the components differently can cause errors.

This lab can be compiled on a local machine or on a remote machine hosted by CloudLabs. Both options allow for the lab to be tested on a local AXC3000 board.

In addition, when using CloudLabs, testing can also be performed using a remote board in the Arrow board farm. Both options are documented.

2. Getting Started

The first objective is to ensure that you have all the necessary software installed so that the lab can be completed successfully. Below is a list of items required to complete this lab. A number of options are provided.

Option 1: Completing the lab using CloudLabs tools and the board farm.

- Personal computer or laptop running 64-bit Linux / Windows 10 or later with at least an Intel i3 core (or equivalent), 8GB RAM.

Option 2: Completing the lab using CloudLabs tools and local hardware.

- AXC3000 Board
- USB C cable
- Quartus Prime 25.1.1 Pro Standalone Programmer.
- Personal computer or laptop running 64-bit Linux / Windows 10 or later with at least an Intel i3 core (or equivalent), 8GB RAM.
- Terminal emulator (Putty or Tera term)

Option 3: Completing the lab using a laptop and local hardware.

- AXC3000 Board
- USB C cable
- Quartus Prime 25.1.1 Pro
- Ashling RiscFree IDE
- Personal computer or laptop running 64-bit Linux / Windows 10 or later with at least an Intel i3 core (or equivalent), 8GB RAM.
- Terminal emulator (Putty or Tera term)

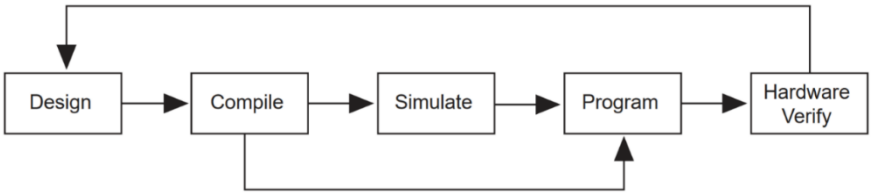
A desire to learn!

3. Design Flow

Overall, the flow for a FPGA design is:

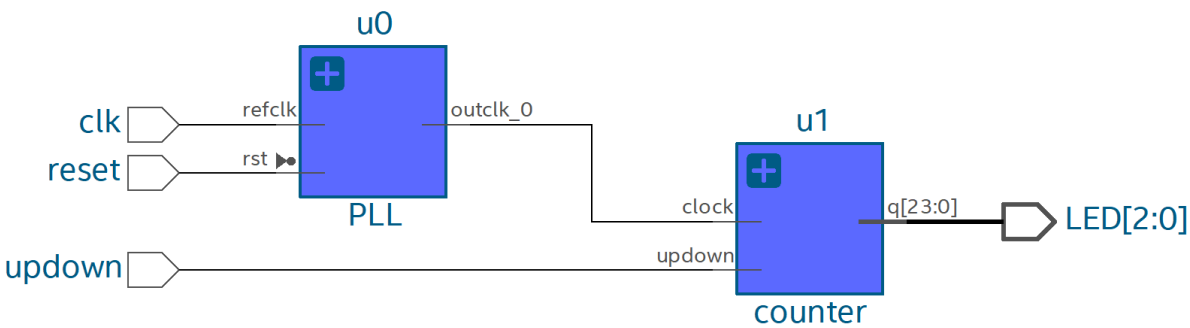
- Design Specification
- Design Entry
- Constraints
- Compilation
- Simulation
- Programming
- Debugging

These steps are iterative with changes as seen as:



Design Specification is important because it defines the FPGA requirements. Having a well-defined specification helps with creating and verifying the design. In this workshop, the design specification is done, but for other designs, it will need to be created.

Design Specification for this design is a PLL connected to a counter (with updown feature) and the counter displays the values on the LEDs on the board.



Design Entry includes hardware description language (HDL), such as Verilog HDL, System Verilog, and VHDL, as well as Platform Designer files that contain IP

(intellectual property) components and/or a combination of the files. In the following step, you create a portion of the digital design with the focus on the IP.

Constraints which is also part of the Design Entry stage is where pin assignments (to correspond to the board that is the design is being programmed to), I/O standards and timing are assigned in Quartus.

Compilation is where the Quartus software checks the design for syntax errors, illegal assignments and/or other issues. It creates and uses an internal database to generate programming files.

Simulation is where the design is functionally checked. Simulation is not covered in this workshop, but simulation is very important to learn and use because the design can be checked for accuracy.

Programming is where an output file is generated and downloaded into the FPGA. Agilex 3 devices are SRAM based devices, which means that when the power is removed, the device will need to be re-programmed. The programming in this workshop is done via JTAG but for production designs, the device can be programmed via an external memory device.

Debugging (Verification) are the steps where the design is checked to see if it works as expected. If the design does not work as expected, investigation of the reason needs to occur i.e. looking at timing, using the built-in logic analysers (Signal Tap), etc.

Quartus Prime design software provides a complete FPGA design environment because it supports the design, constraints, compilation, simulation, programming and debugging.

4. Project with AXC3000 Board

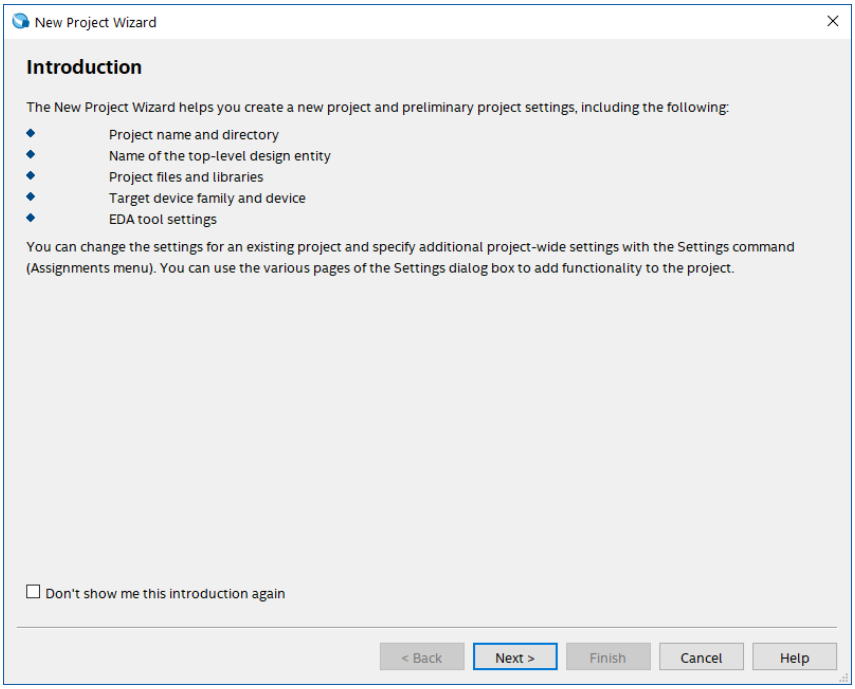
4.1 New Quartus Prime project

A Quartus project defines the location for the files, target family, target device, and initial settings. After the initial creation of the project, changes can be made. Quartus uses the project location for the generated files.

4.1.1 New project creation

4.1.1.1 From the Start menu or the Desktop, **open Quartus Prime Pro 25.1.1** software.

4.1.1.2 Create a new project using the New Project Wizard: **File → New Project Wizard**. This screen shows an introduction window.



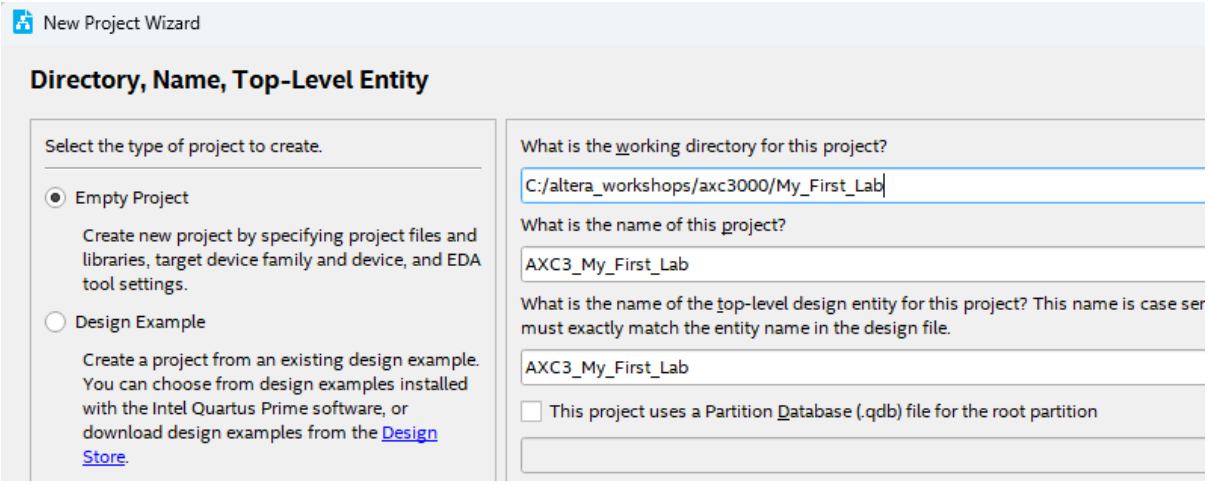
4.1.1.3 Click **Next**.

4.1.1.4 Configure the New Project Wizard directory, name, and top-level entity information:

- **Type the directory** name in which you will store your Quartus project files for this design C:/altera_workshops/axc3000/My_First_Lab
  This will be the directory where Quartus will place the files.
- **Specify the name of the project**: AXC3_My_First_Lab
- **Specify the name of the top-level entity**: **AXC3_My_First_Lab**

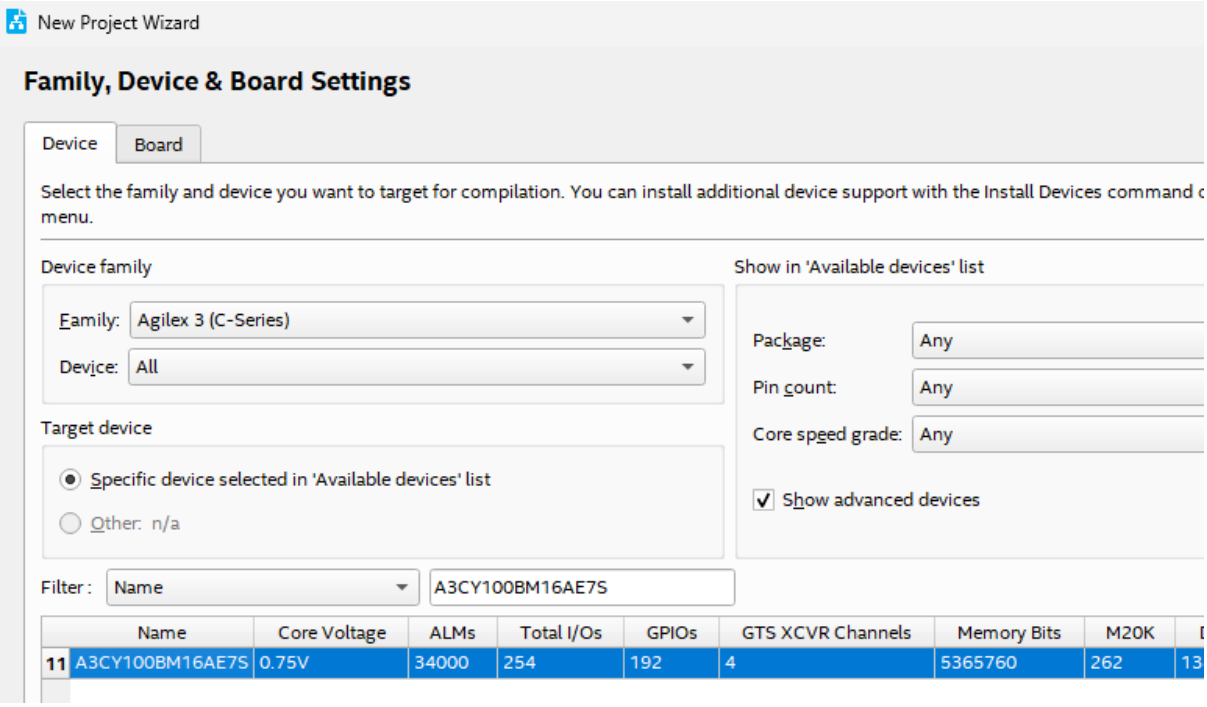
Quartus uses this name to associate files.
There should be no space in the file name nor a space in the directory path to avoid possible compile issues.
The name of the top-level entity can be the same name as the project or a different name. The top-level entity tells Quartus the file that is the top-level design file when it compiles. The name can be changed later.

- On the Project Type page, select **"Empty project".**
  Empty Project means that you can start from scratch or use files that were created rather than basing this design on a previous based design.



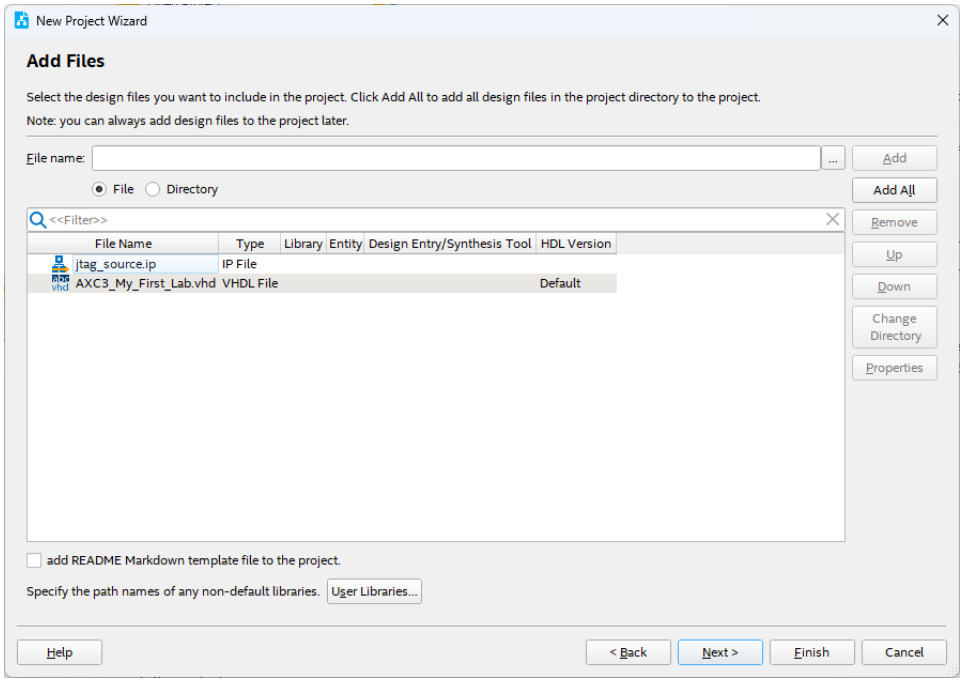
4.1.1.5 Click **Next** (at the bottom of the page)

4.1.1.6 Specify Family and Device Settings. Select **Family** to select Agilex 3 family and enter the part number in the Name Filter text box. The part number is **A3CY100BM16AE7S**. The family and part number selected needs to correspond to the device on the **AXC3000 Board** to avoid incompatibility when programming. It is very important to ensure that part number is highlighted or else another (default) part number will be used.



4.1.1.7 Click **Next** (on the bottom right of the page)

4.1.1.8 To avoid possible typing issues, a top level VHDL file was created in advance for this workshop. Select the … by the file name, change to the **c:/altera_workshops/axc3000/AXC3_My_First_Lab** directory and select the files called **AXC3_My_First_Lab.vhd** and **jtag_source.ip.** The files will be added in the list as seen below.

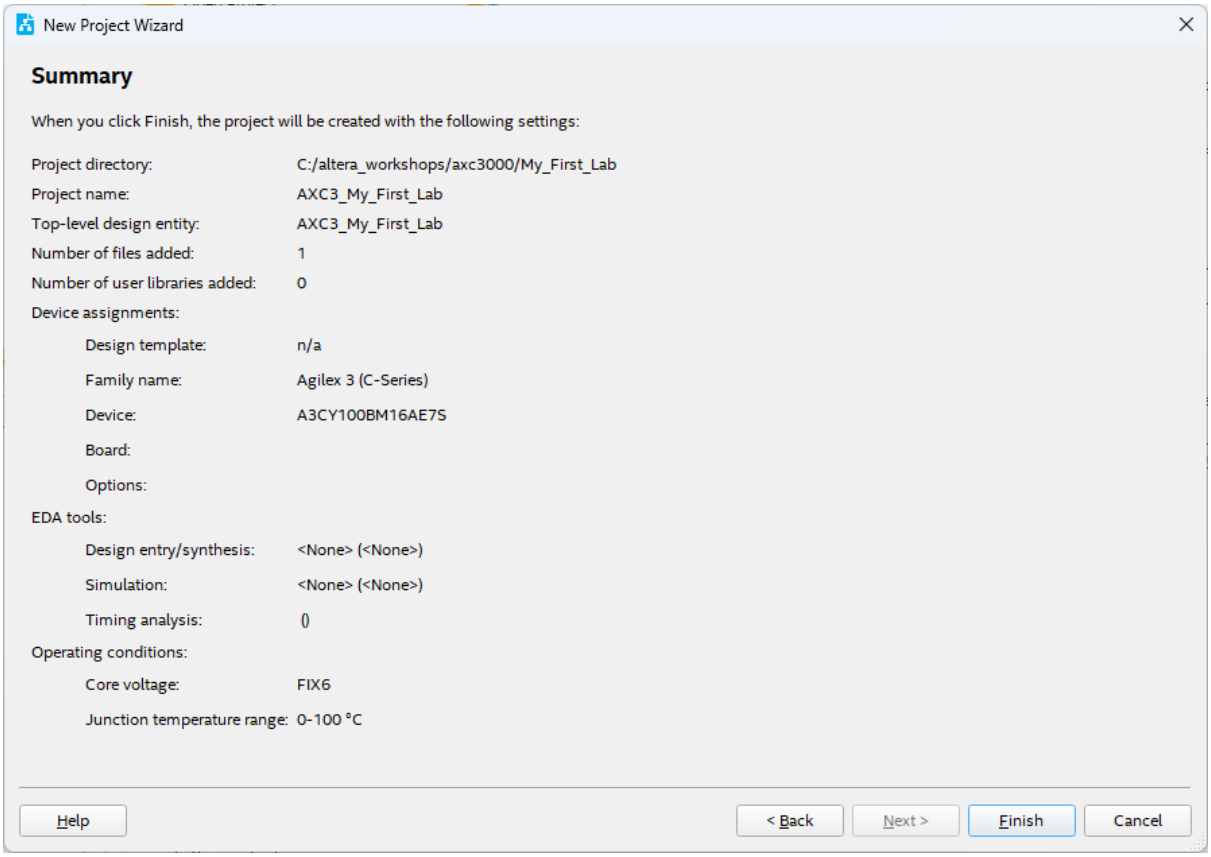


4.1.1.9 Click **Next**

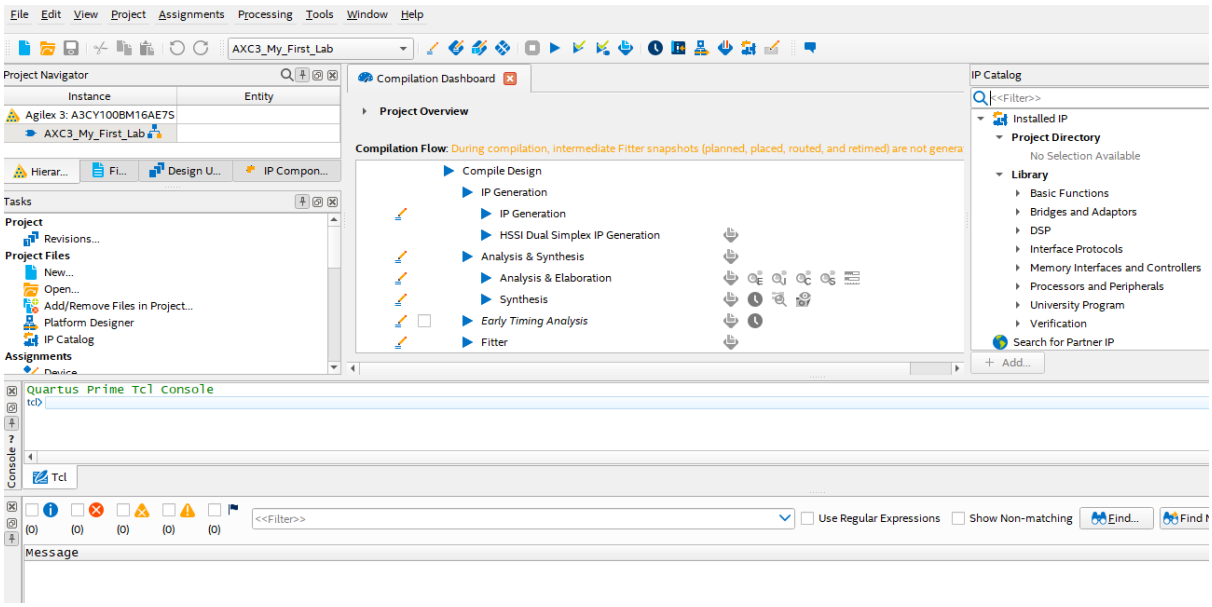
4.1.1.10 On the EDA tool setting, the settings will be left as default. EDA tools are third party software such as simulation, board level, etc and they will not be used here.

4.1.1.11 Select **Next**

4.1.1.12 The Summary screen will show the project and file locations. These settings can be changed later, if needed.

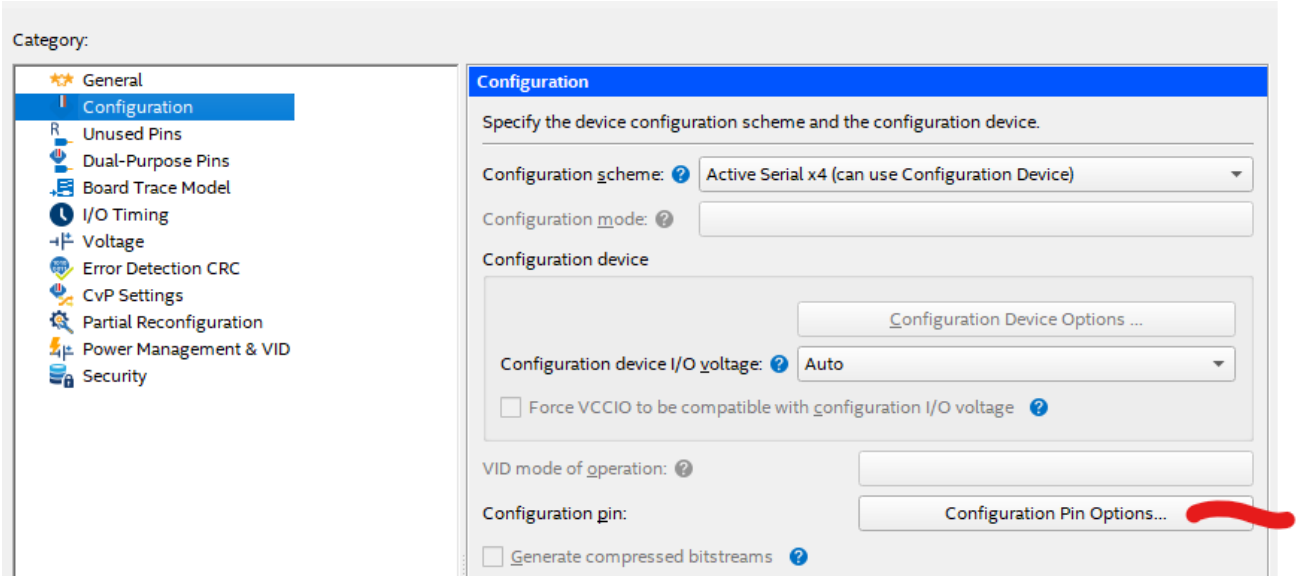


4.1.1.13 Select **Finish.** The Quartus project will now be set up as

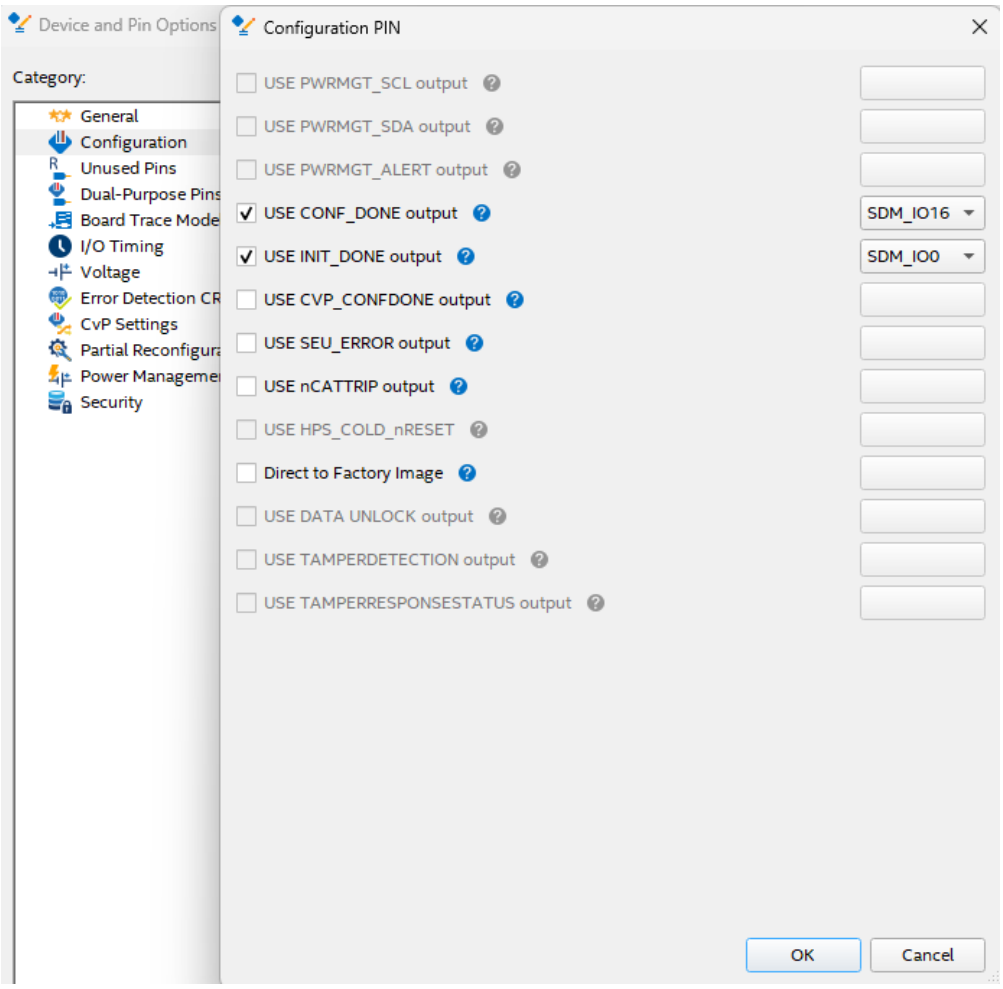


The top-level file is seen in the top left corner. The middle section (compilation flow section) shows the FPGA steps which will be covered in the following sections. These steps include analysis and synthesis, fitter, etc. The far right of the Quartus window shows the IP Catalog. The IP Catalog contains parameterizable functions which will be covered in the next steps.

4.1.1.14 **Select Assignments -> Device -> Device and Pin Options-> Configuration -> Configuration Pin Options as**



4.1.1.15 **Enable the Conf_done** and **set the pin to the SDM_IO16 and enable the Init_done output and set to to SDM_IO0.** The Conf_done pin can be used to tell that the configuration is complete and after the configuration, it can be used as general purpose I/O. The init_done pin indicates that the initialization after the boot up of the FPGA is complete. The Reset Release IP that will be described and created in this workshop will be connected to the Init_done pin.



4.1.1.16 Select **OK**.
4.1.1.17 Select **OK** again to close the Device and Pin Option window.
4.1.1.18 Select **OK** again to close the Device window.

4.2 Design entry

Overview: In this module you will create the IP components for the design. You will set pin assignments and compile the design.

There are different ways to create a design in Quartus including Verilog, System Verilog, VHDL, and IP (intellectual property) blocks. Quartus will accept a mixture of these file formats.

4.2.1 Add PLL to the Quartus Project

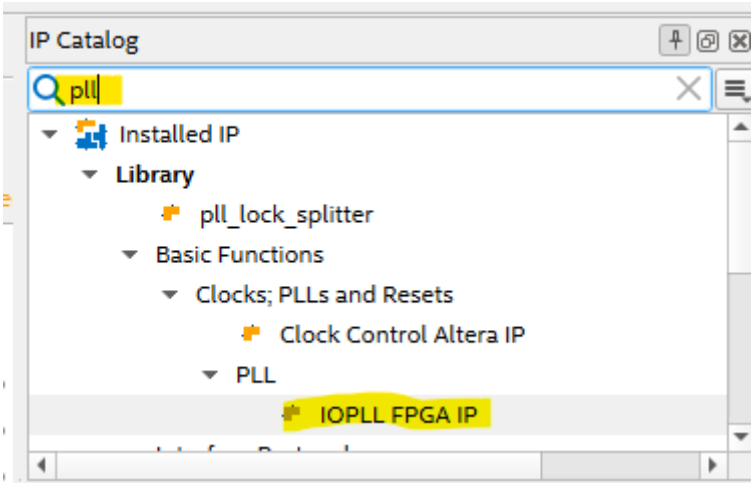
The IP catalogue in Quartus contains many parameterizable functions including a PLL (phase lock loop).

All FPGA designs need at least one clock associated with the data. Synchronous designs are more reliable, have better performance and are more verifiable than a combinational design. Combinational design generally is not a good design approach.

A PLL (phase lock loop) provides flexibility with the clock settings because with an input clock, there can be multiple output clocks with different rates and phases.

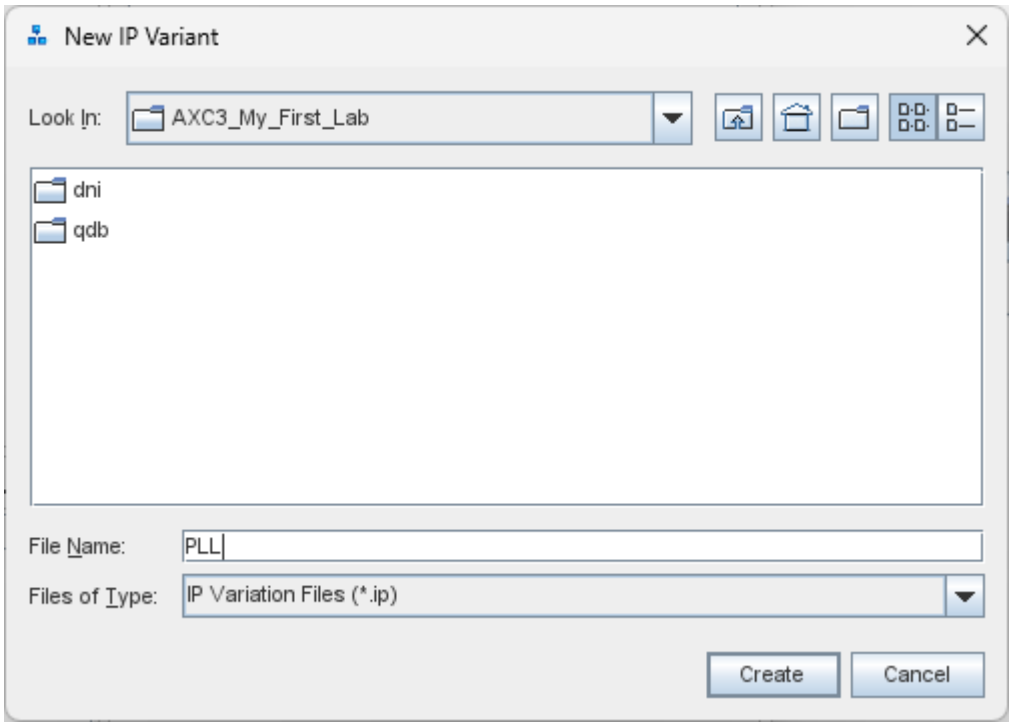
4.2.1.1 From the IP Catalog panel on the right side, expand the menus for the **Basic Functions** → **Clocks; PLLs and Resets** → **PLL** and double click on **IOPLL FPGA IP**.

If the IP Catalog is not visible, then right click on the toolbar and select IP Catalog.



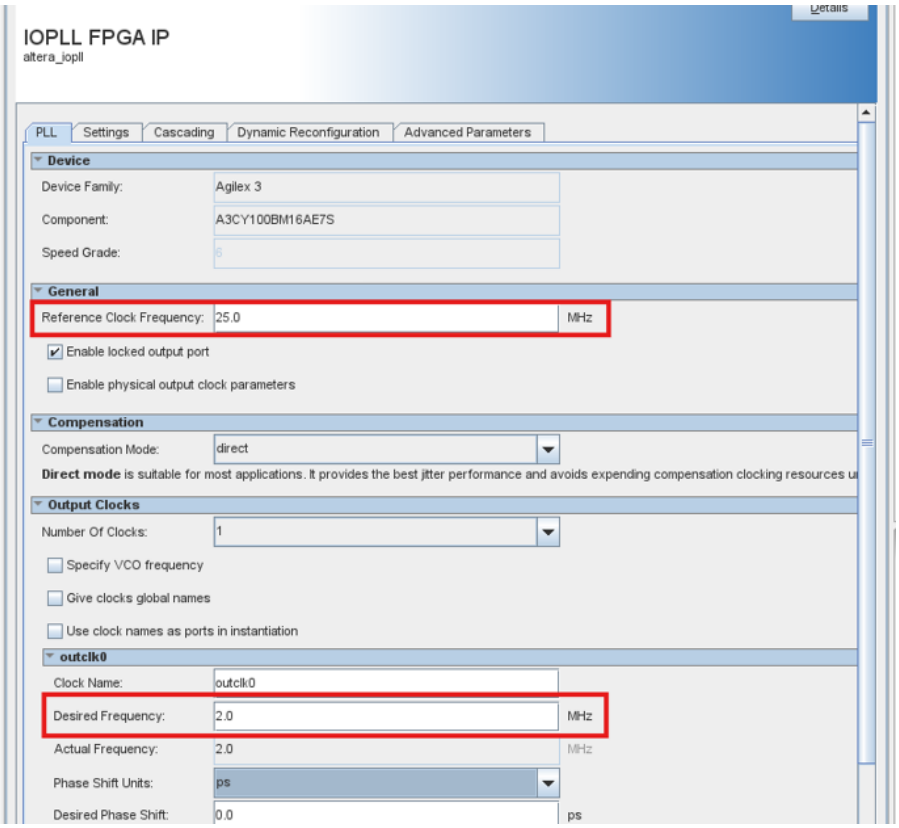
4.2.1.2 On the New IP Variation window, enter the following information.

- IP variation file name: **<project_directory>/PLL** as shown:



4.2.1.3 Select **Create**

4.2.1.4 **Change the default Reference Clock Frequency to 25 MHz and Desired output frequency to 2MHz** as shown:

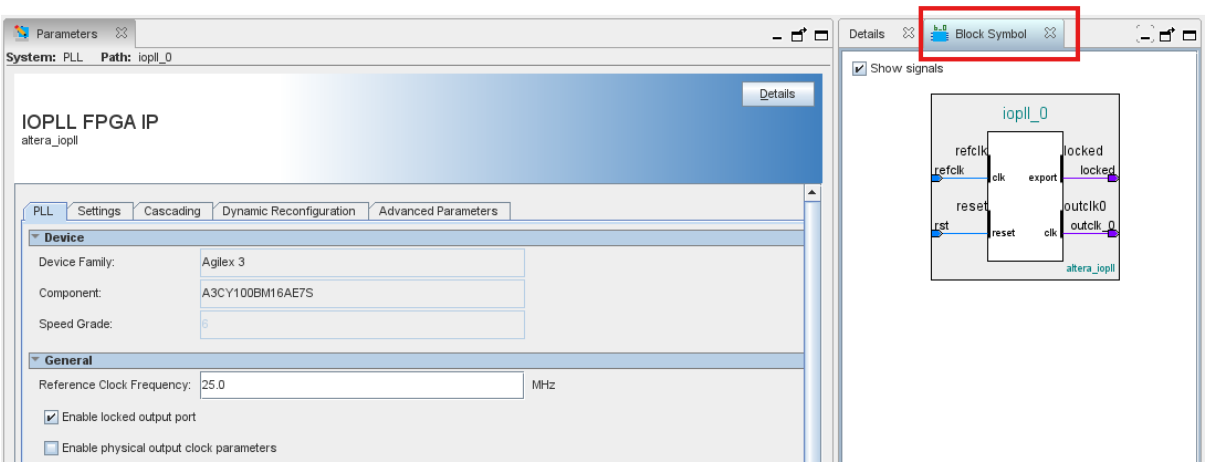


The reference clock frequency needs to be changed to be 25 MHz so that it corresponds to the clock device on the board.

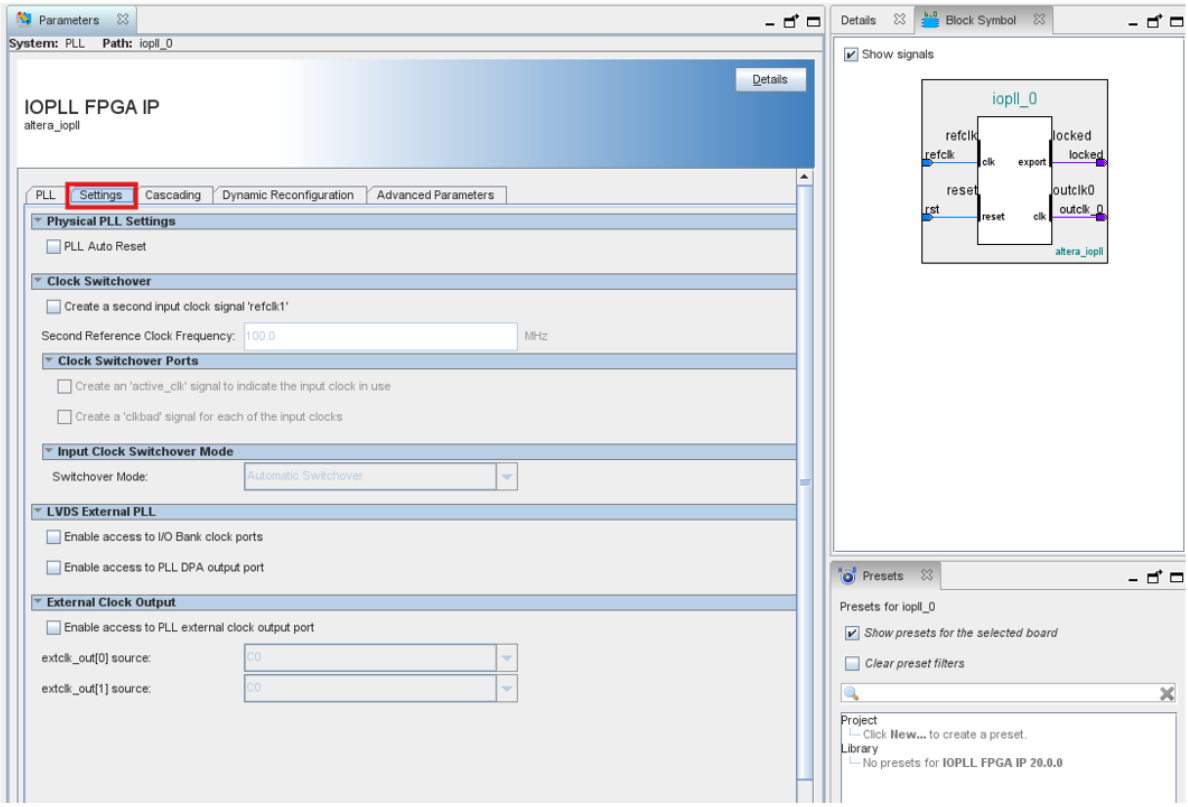
In the outclkc0, the desired frequency needs to be changed to be 2 MHz. A slower frequency is needed so that the changing LEDs could be seen on the board. If the clock is too fast, the toggling LEDs cannot be seen.

There are various options that can be selected as seen in the various tabs such as settings (clock switchover, LVDS External PLL), cascading (connecting multiple PLLs together so that different PLL combinations can be used), dynamic reconfiguration, and other parameters. However, to make this workshop easier, the default settings are selected.

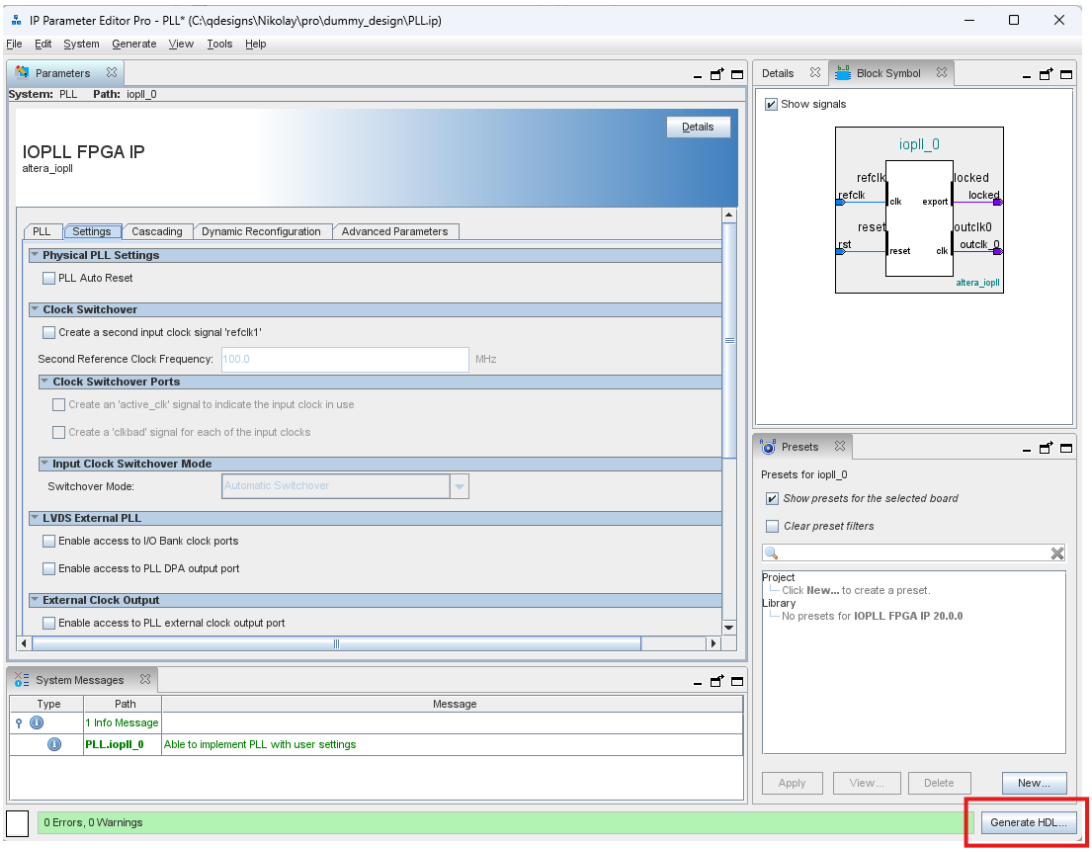
As settings are changed, the Block Symbol (as below) shows the input and output ports. However, Quartus Prime Pro, which supports Agilex 3, does not support a way to create a top-level design with schematic editor.



4.2.1.5 **Select the Settings tab in the IOPLL** and review the settings. No changes are needed.



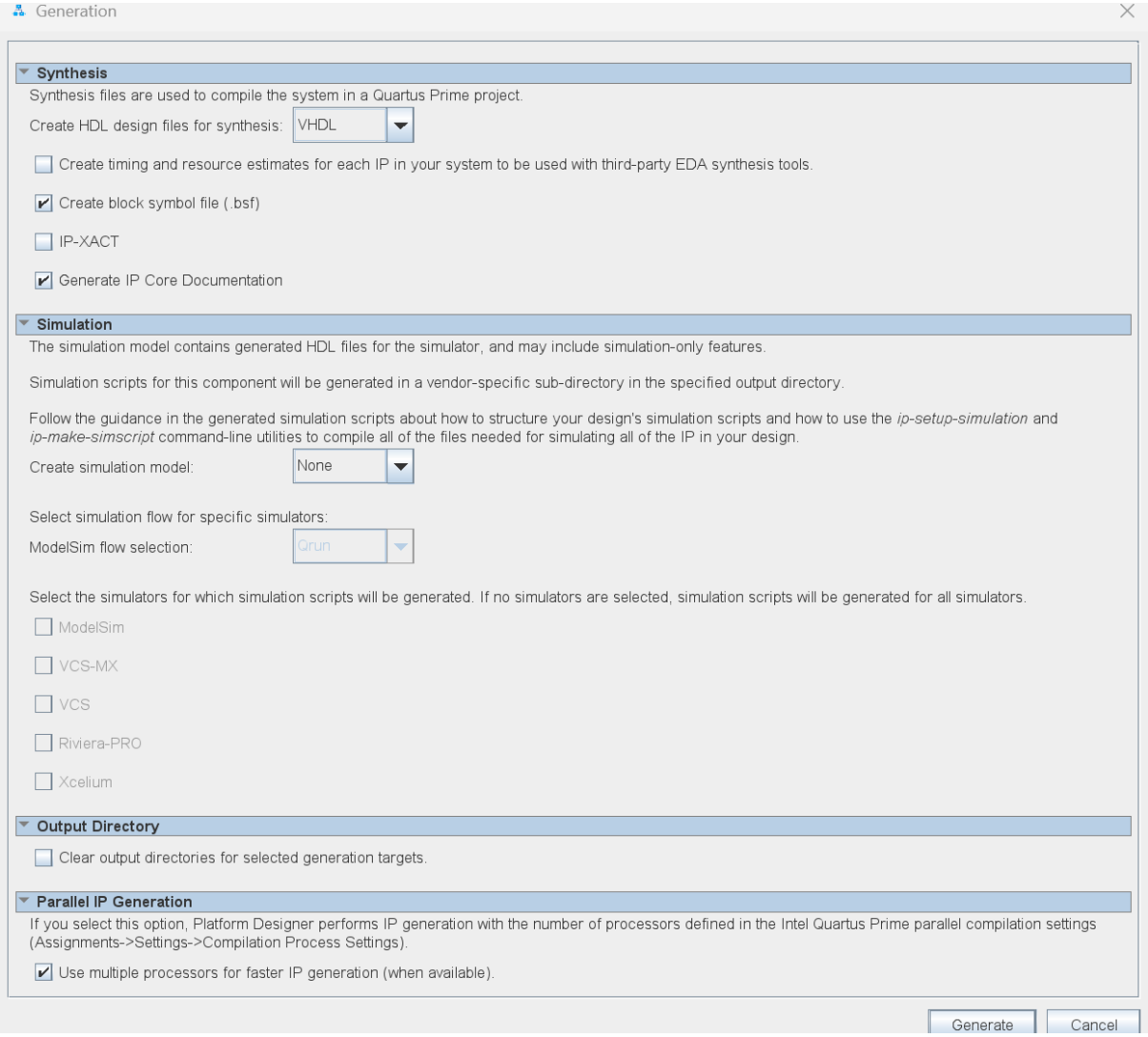
4.2.1.6 To generate the IP, select the **Generate HDL** button (bottom right-hand corner).



The Generate HDL converts the settings into a RTL file that can be instantiated and "connected" with other RTL files

4.2.1.7 Change the **Default settings to be VHDL**. Verilog could have been used as well but this workshop was created with VHDL files. The focus on this workshop is on synthesis (not on simulation).

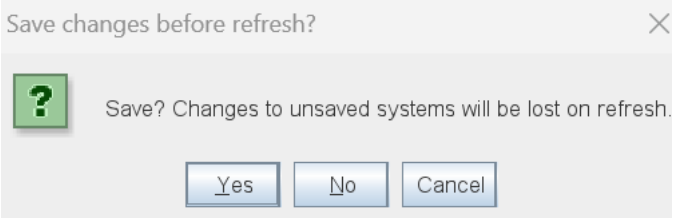
The settings should look as:



One thing to note is above there is an option to create block symbol file (.bsf). Quartus Prime Pro, which supports Agilex 3, does not support a way to create a design with schematic editor.

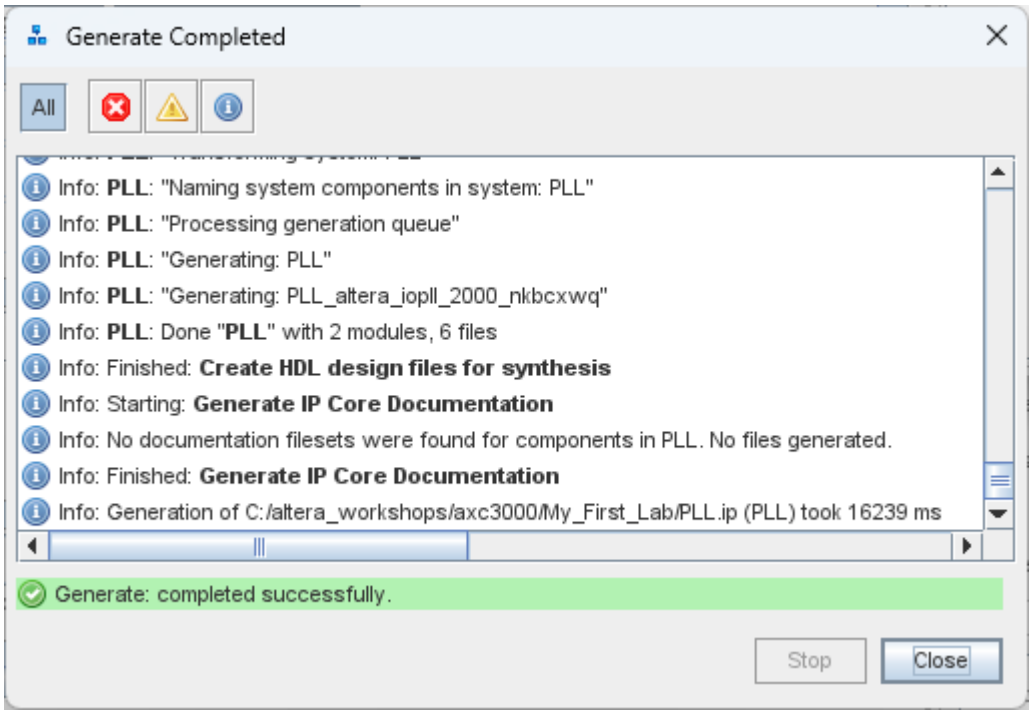
4.2.1.8 Select **Generate**

4.2.1.9 A window will appear that states **Save? Changes to unsaved system will be lost on refresh.**



4.2.1.10 Select **Yes**. The file is saved and generated.

4.2.1.11 A window will appear that shows when the RTL files (and other files) are generated. The green comment states when it is completed successfully.



4.2.1.12 Select **Close**.

4.2.1.13 Close the IP Parameter Editor by selecting **File** → **Exit**

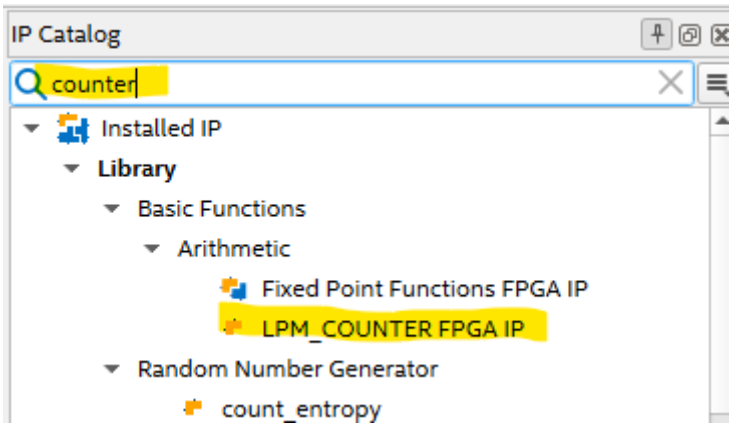
There are different files that are generated by Quartus including:

.qip (a single file that contains list of files and assignments for the IP core rather than having to add these individually),
.rpt (Quartus report file),
.ip file (intellectual parameterizable file)
.vhd (RTL file) and many other files.

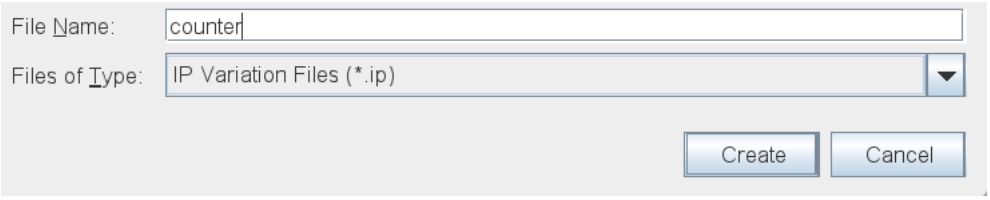
4.2.2 Add counter to the Quartus Project

A counter is used so it can increment up and down and thus will display the counting LEDs on the board.

4.2.2.1 In the search bar of the IP Catalog, type "**counter**", and **double click** on **LPM_COUNTER FPGA IP** as below. Even if part of the word of count is typed, Quartus will show the closest IP.

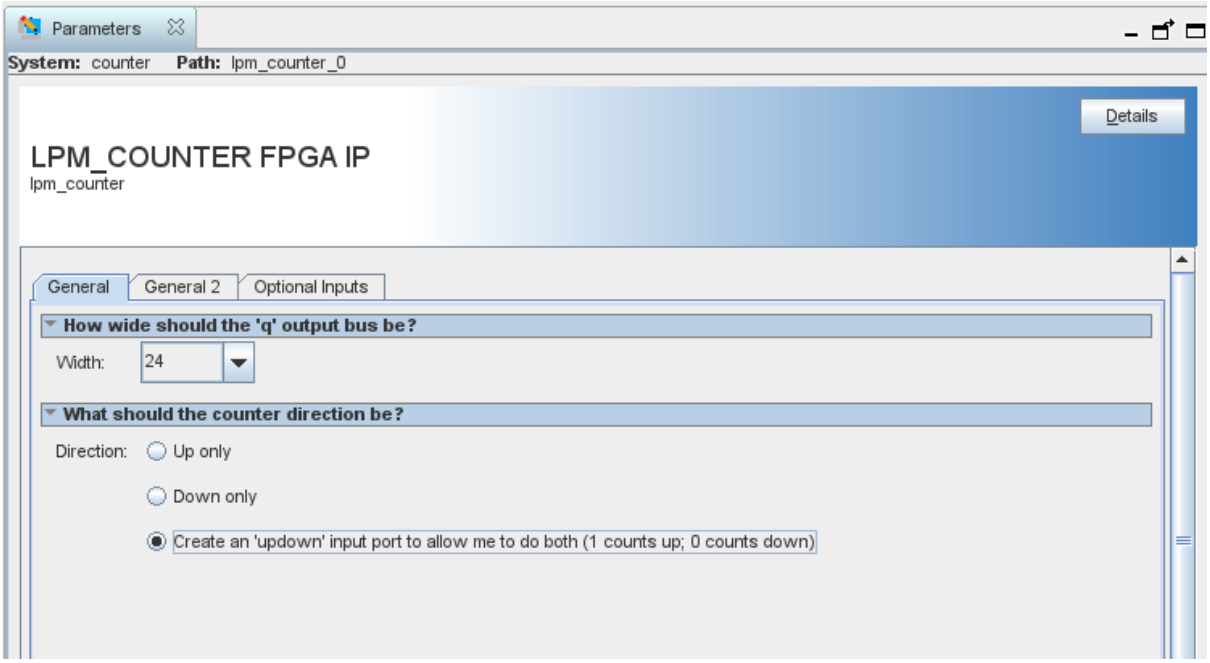


4.2.2.2 In the New IP Variant window enter **counter** for the IP variation file name as:

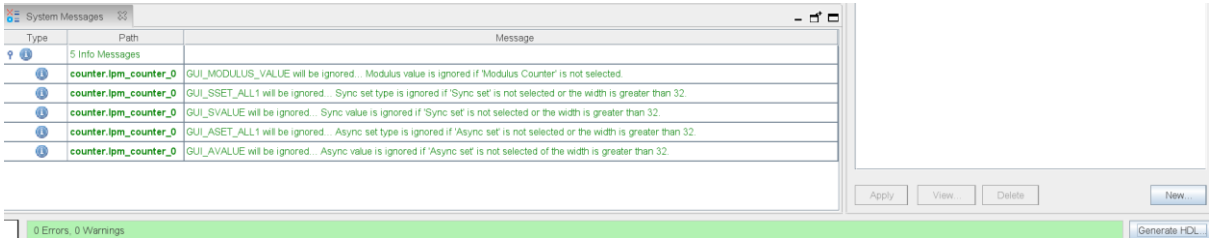


4.2.2.3 Select **Create.**

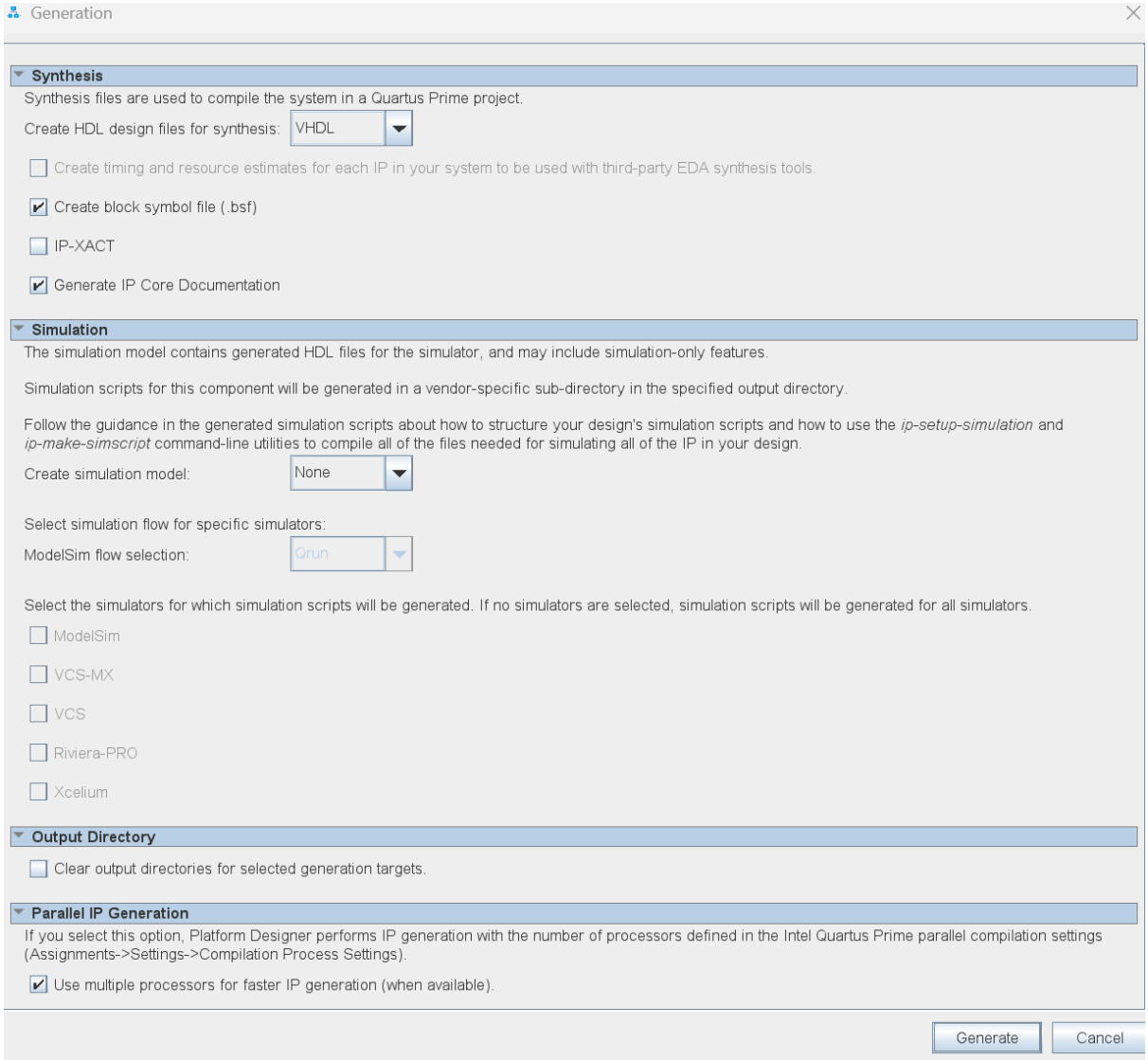
4.2.2.4 Check that the setting is set to **24 bits**. This will ensure that the most significant bit of the counter is toggling about every second. The three most significant bits of this counter will connect to the R, G, and B connections of LED_RGB0 on the board. Select the **updown** option so that the counting can go up or down.



4.2.2.5 Select **Generate HDL** (bottom right-hand corner of the screen) so that it can generate an RTL file.

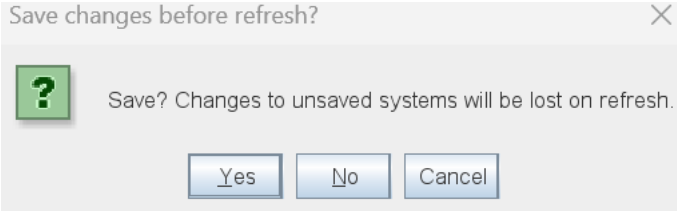


4.2.2.6 Change the **settings to be VHDL**, as VHDL was used for this workshop. Verilog could have been used as well.

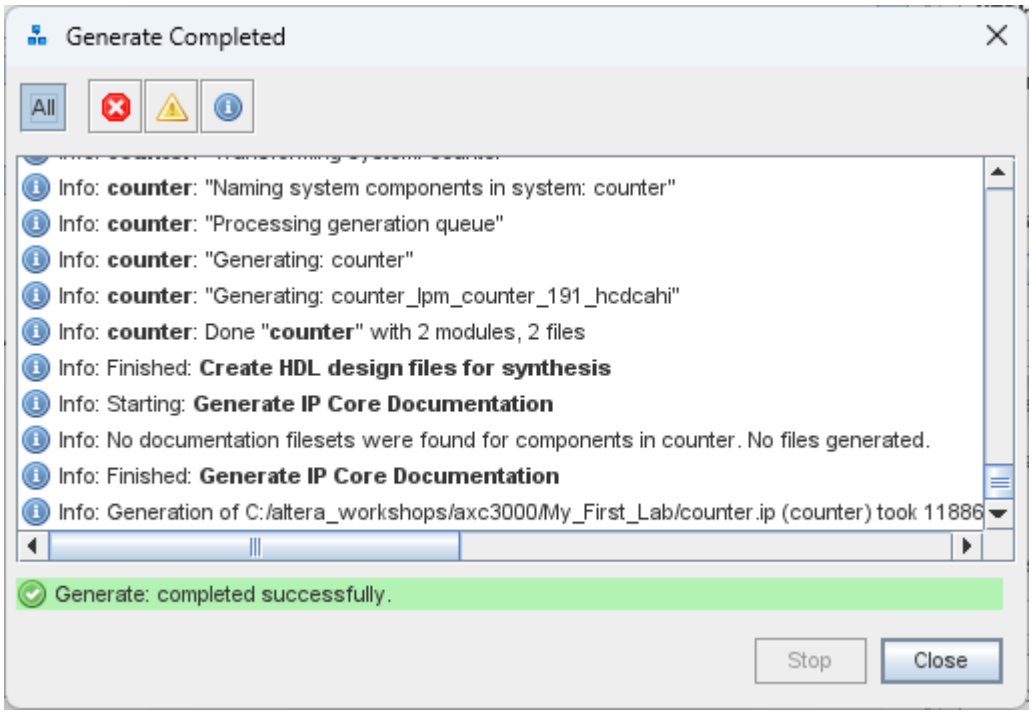


4.2.2.7 Select **Generate.**

4.2.2.8 Select **Yes** for Changes to the unsaved systems will be lost on refresh to save the changes.



4.2.2.9 Select **Close** (bottom right of the window)

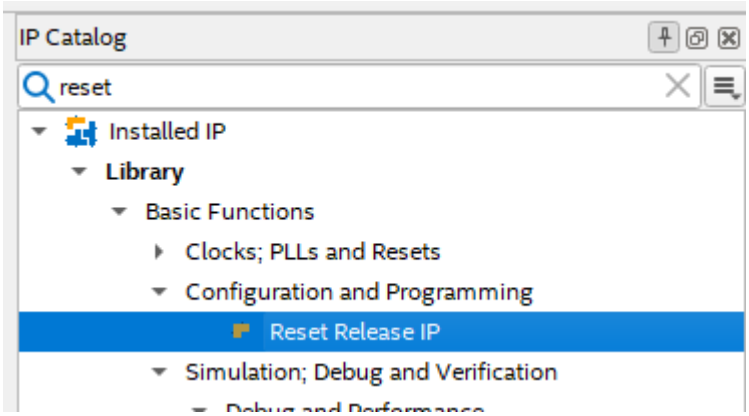


4.2.2.10 Close the IP Parameter Window for the counter by selecting **File** → **Exit**

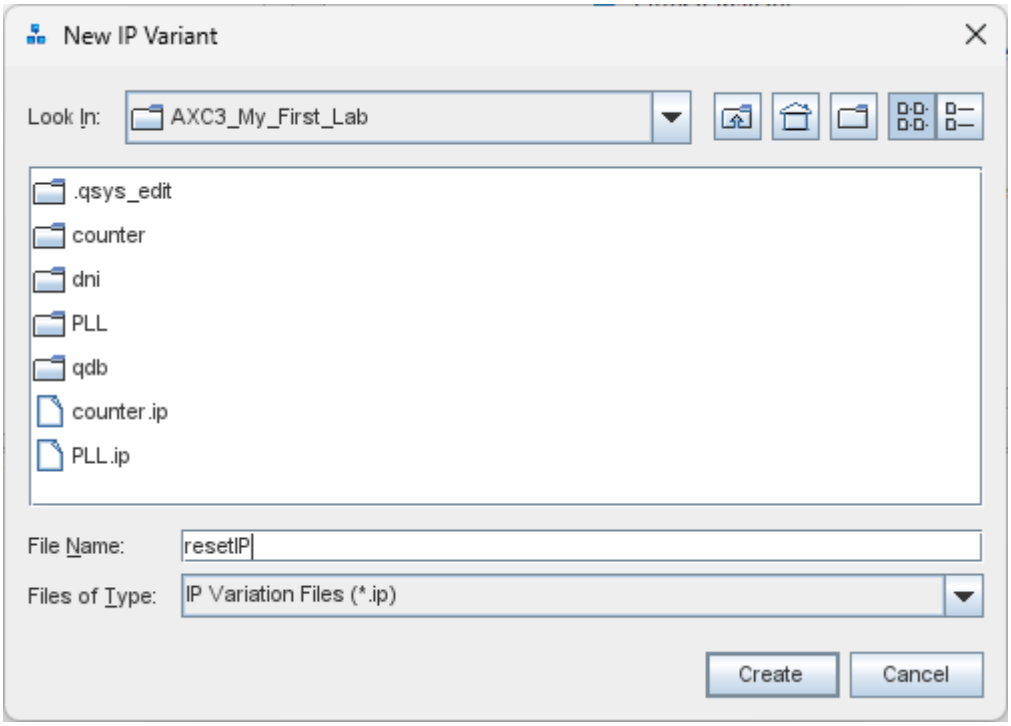
4.2.3 Reset Release IP

For Agilex 3 devices (and other families), it is recommended to use the Reset Release IP. The Reset Release is an IP that generates a signal to indicate the device configuration is complete. It holds the user logic in reset until the configuration is complete, and user mode is entered. If this IP is not implemented, Quartus will report a warning that the configuration might not be correct.

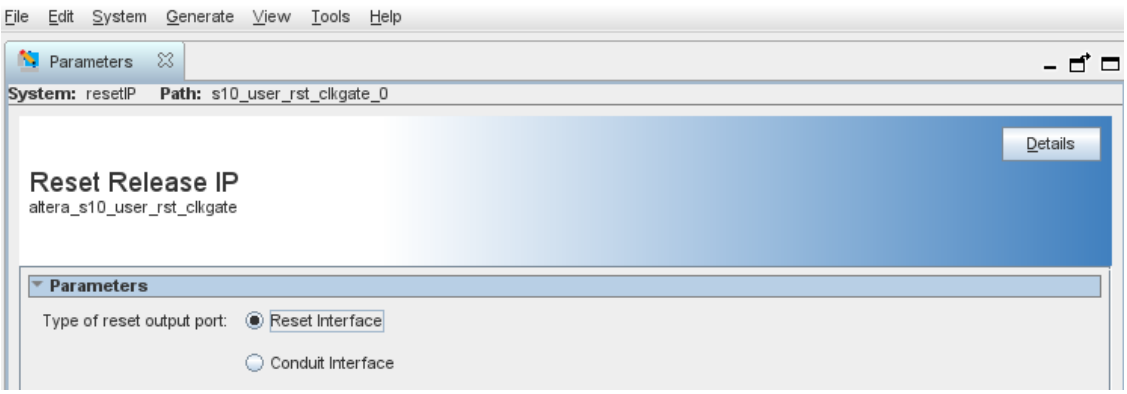
4.2.3.1 In the search bar of the IP Catalog, type "**reset**", and **double click** on **Reset Release IP** as below. Even if only part of the word of reset is typed, Quartus will show the closest IP.



4.2.3.2 Type the **name resetIP** and select **Create**

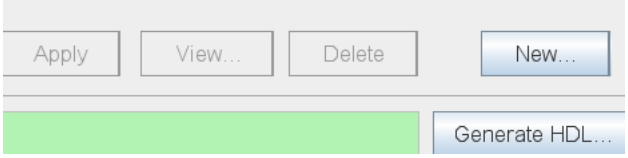


4.2.3.3 Select **Reset interface**

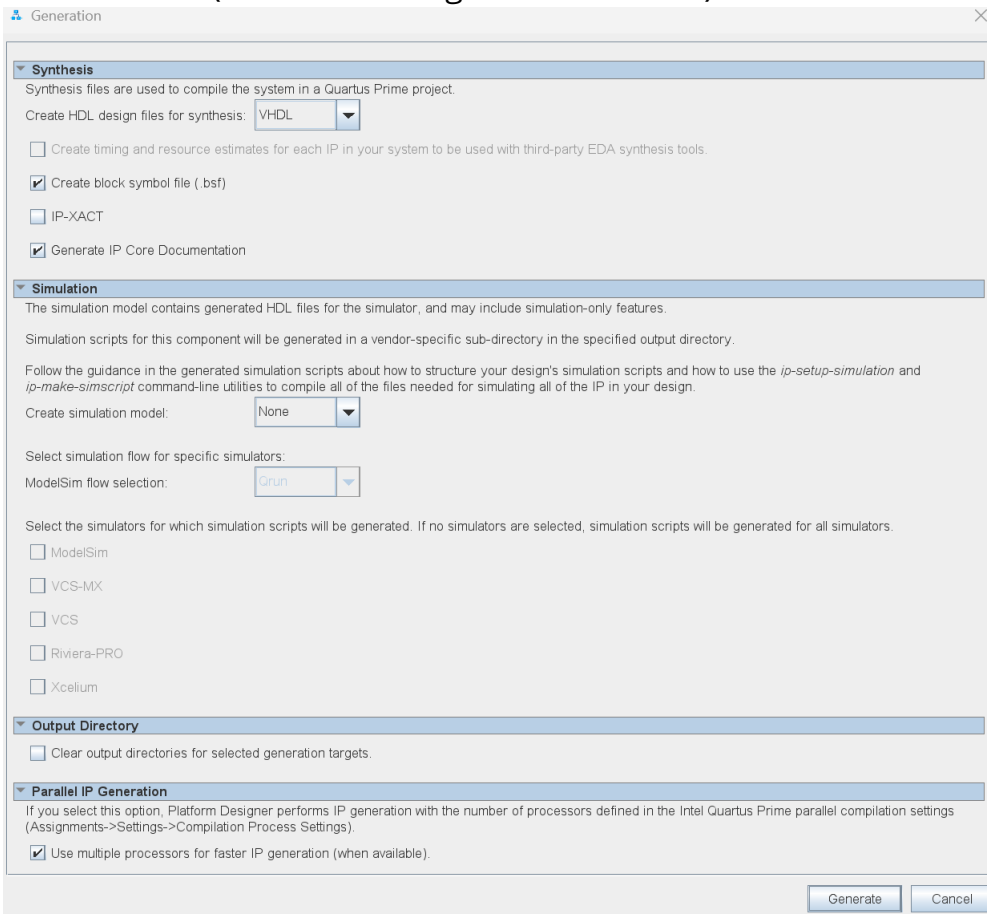


Conduit interface is a legacy reset output.

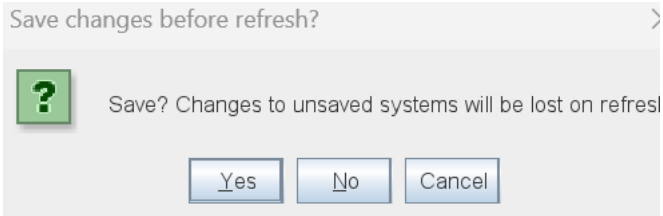
4.2.3.4 Select **Generate HDL** in bottom right of the window as



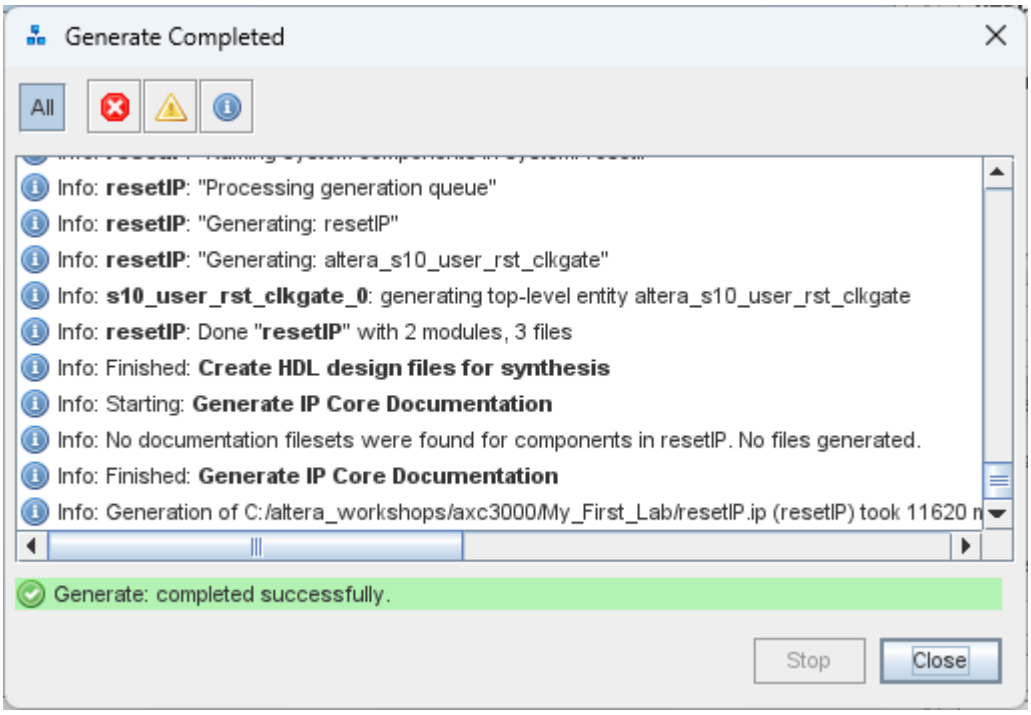
4.2.3.5 Select **Generate** (in the bottom right of the window)



4.2.3.6 Select **Yes** to the window of Save? Changes to unsaved systems will be lost on refresh.



4.2.3.7 Select **Close** after the Generation is complete



4.2.3.8 Close the IP Parameter Window by selecting **File** → **Exit**

4.2.4 Top-level file

The IP blocks (a PLL and a counter) needed to be connected to form a functional design. To make this process easier, the top level VHDL file, that was already created and added when the Quartus setup was done, will make the connections. The VHDL file looks like this:

```
10
11
12  entity AXC3_My_First_Lab is
13  port (
14   clk: in std_logic;
15   reset : in std_logic;
16   updown: in std_logic;
17   LED : out std_logic_vector(2 downto 0)
18  );
19   end entity;
20
21  architecture rtl of AXC3_My_First_Lab  is
22
23  component pll is
24        port (
25           refclk   : in  std_logic := 'X'; -- clk
26           rst      : in  std_logic := 'X'; -- reset
27           outclk_0 : out std_logic         -- clk
28        );
29   end component pll;
30
31  component counter is
32     port (
33        clock  : in  std_logic;
34        updown : in  std_logic;
35        q      : out std_logic_vector(23 downto 0)
36     );
37   end component counter;
38
39  component resetip is
40     port (
41        ninit_done : out std_logic   -- ninit_done.reset
42     );
43   end component resetip;
44
45  --***** uncomment the next five lines of code if using board farm hardware
46   --component jtag_source is
47   -- port (
48   --    source : out std_logic_vector(1 downto 0)
49   -- );
50   --end component jtag_source;
51
52   signal tempclk : std_logic;
53
54   signal count : std_logic_vector(23 downto 0);
55
56   signal ninit_done : std_logic;
57
58   signal system_reset : std_logic;
59   signal up_down : std_logic;
60
61  --***** uncomment the next line of code if using board farm hardware
62   --signal jtag_src : std_logic_vector(1 downto 0);
63
64   begin
65
66   LED(2 downto 0) <= count(23 downto 21);
67
68   u0: PLL port map
69     (
70      refclk => clk,
71      rst => system_reset,
72      outclk_0 => tempclk
73     );
74
75   u1: counter port map
76     (
77      clock => tempclk,
78       updown => up_down,
79       q => count
80     );
81
82   u3: resetip port map
83     (
84      ninit_done => ninit_done
85     );
86
87  --***** uncomment the next two lines of code if using local hardware
88   --system_reset <= ninit_done or (not reset);
89   --up_down <= updown;
90
91  --***** uncomment the next six lines of code if using board farm hardware
92   --system_reset <= ninit_done or jtag_src(0);
93   --up_down <= jtag_src(1);
94
95  --u4 : jtag_source   port map
96   -- (
97   --  source => jtag_src
98   -- );
99
100  end rtl;
```

The two sub-blocks (components) are the PLL and the counter. The top-level file connects signals i.e. connects the output clock port from the PLL ('tempclk' signal) to the input clock port of the counter. The updown signal of the counter is for the counting, while LED output port will be connected to the actual LEDs on the board.

Note that the system_reset signal can be asserted by either the end of configuration (ninit_done), or by asserting the reset signal.

Optional logic has also been included to support testing in the remote board farm. Please note those lines of code available for the local or remote testing options. <u>Please uncomment those lines of code as directed.</u>

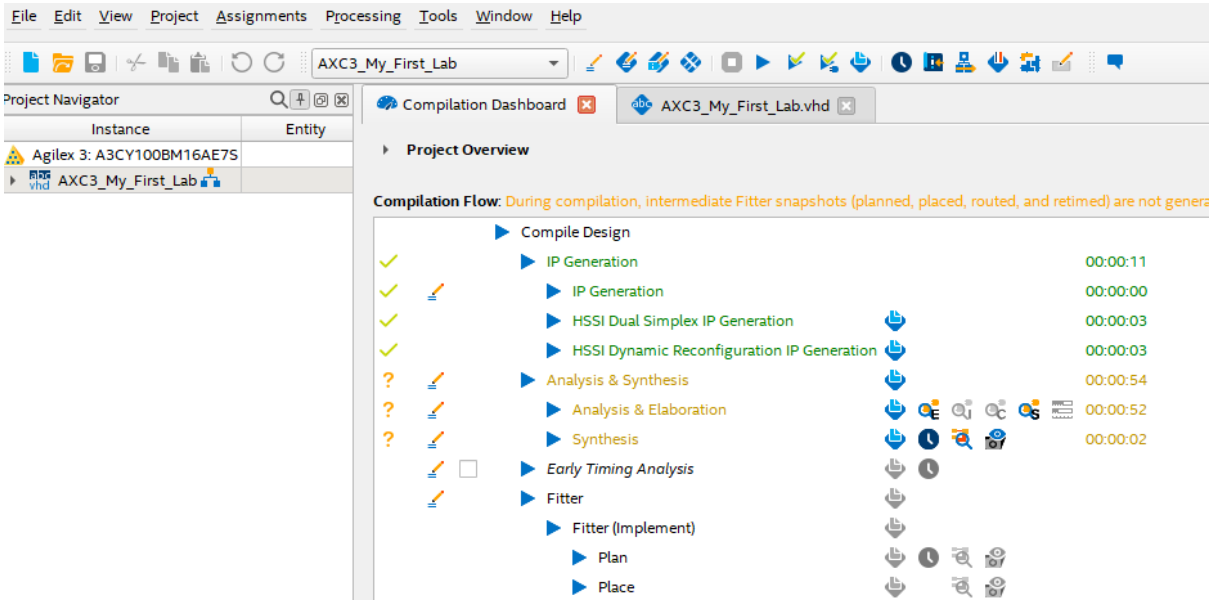
4.3 Compile Design

4.3.1 Analysis and Synthesis

Analysis and synthesis is a step where the design files are converted into a netlist that Quartus can use for later steps. Quartus checks legality of the connections. The netlist is used by Quartus so that the timing and IO and other constraints can be assigned.

4.3.1.1 Run Analysis and Synthesis by clicking on button on the toolbars, or **Processing → Start → Analysis and Synthesis**.

There should be no errors as seen below:

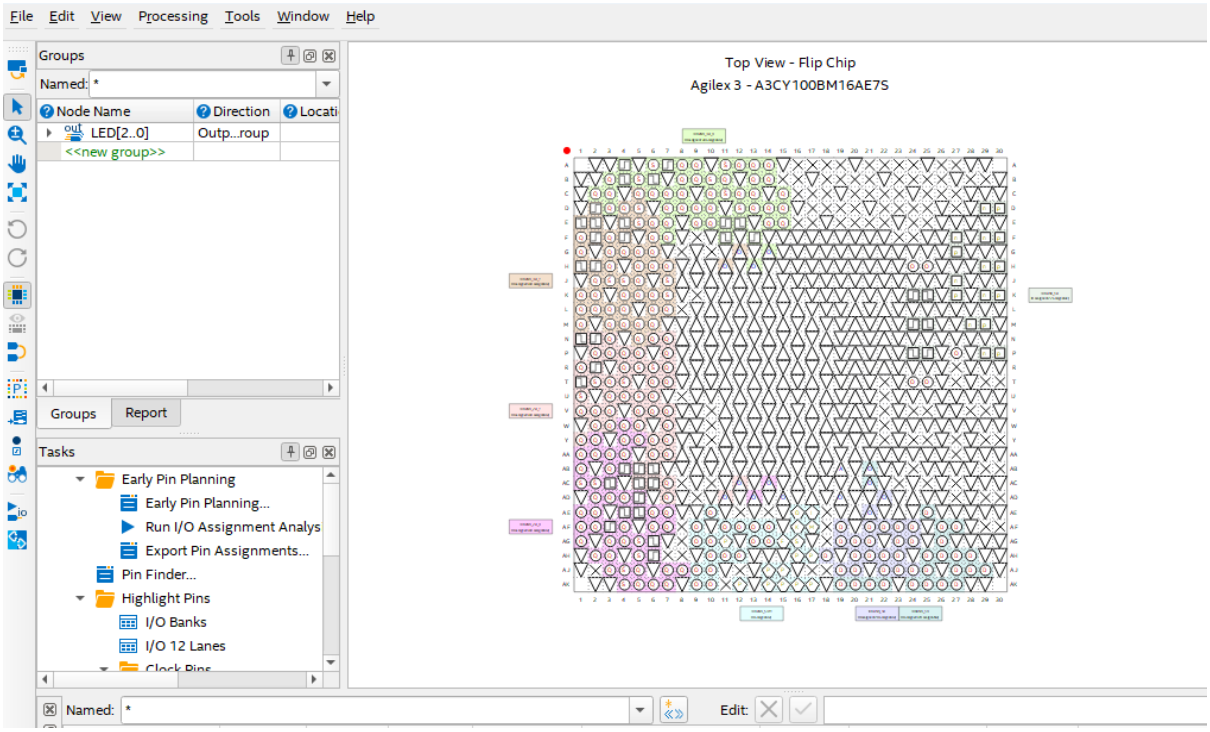


The items in green with the yellow checkmarks shows the steps that Quartus did.
If there are errors (where Quartus will report an error), the errors will need to be fixed before continuing.

4.3.2 Pin Assignments

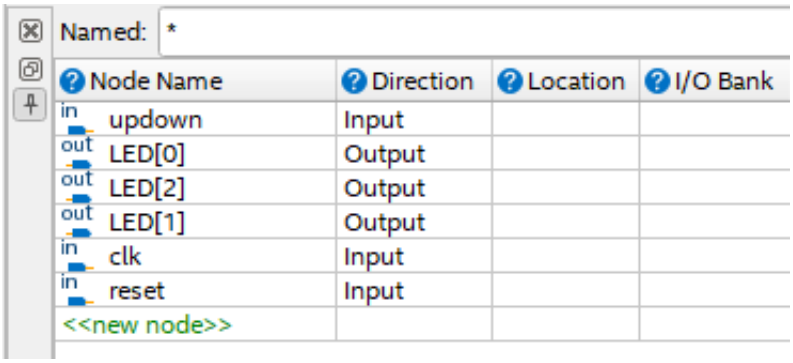
Pin Assignments are needed so the correct signals are assigned to the correct pins on the board. Without pin assignments, Quartus will use default assignments which will not correspond to the board pins.

4.3.2.1 Open **Pin Planner** by clicking on button on the toolbars, or **Assignments → Pin Planner**.



4.3.2.2 There are different ways to enter pins. A common way is to use a TCL script (which can be used for many pins). The way we do it in this workshop is to use the GUI as there are only 6 pins.

4.3.2.3 At the bottom of the window, there are various pins. The reset, various LEDs and the clk pins need to have assigned pin locations and I/O standards.



Named: *

Node Name	Direction	Location	I/O Bank
in updown	Input		
out LED[0]	Output		
out LED[2]	Output		
out LED[1]	Output		
in clk	Input		
in reset	Input		
<<new node>>			

These pins will need locations and I/O bank standards so that Quartus will place the pins in the correct location (i.e. correspond to the board schematic) and have the correct voltage standards as defined by the board (to avoid conflict). Without

assignments, Quartus will make assumptions on the locations, which will not match the board.

4.3.2.4 Select **the Location** field associated with the reset pin and **type in the word Pin_A12**.
The pull-down menu would also work. This pin is a DIP Switch on the board. The corresponding I/O bank will be entered automatically.

Named: *reset

Node Name	Direction	Location	I/O Bank
reset	Input	PIN_A12	3A_B

4.3.2.5 Use the **pull-down menu to select the I/O standard to be 1.3 V LVCMOS**. This is the I/O standard for this pin on the board.

Named: *reset

Node Name	Direction	Location	I/O Bank	I/O Standard
reset	Input	PIN_A12	3A_B	1.3-V LVCMOS

4.3.2.6 **Repeat the previous steps to assign the rest of the pins for Location and I/O assignments.** The order of the pins in the list does not matter. These pins correspond to the schematic on the board.

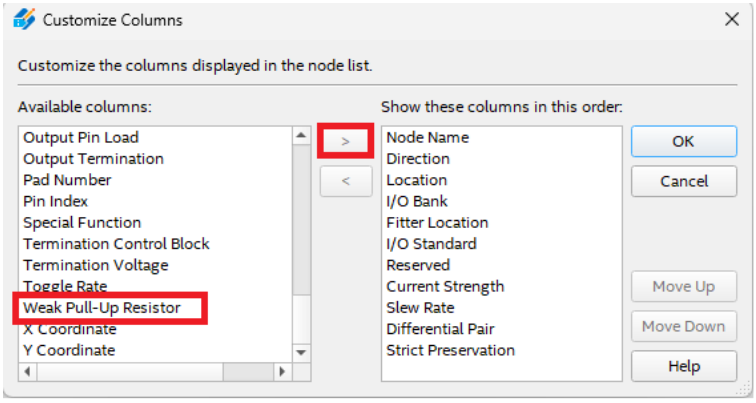
Named: *

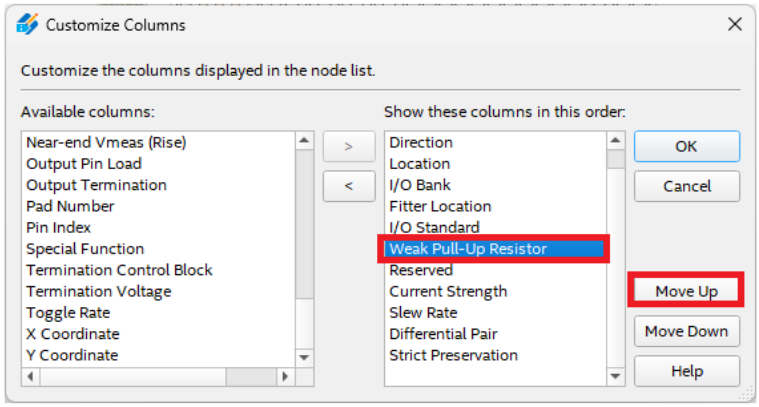
Node Name	Direction	Location	I/O Bank	I/O Standard
updown	Input	PIN_A14	3A_B	1.3-V LVCMOS
LED[0]	Output	PIN_AK20	5B	3.3-V LVCMOS
LED[2]	Output	PIN_AH22	5A	3.3-V LVCMOS
LED[1]	Output	PIN_AK21	5B	3.3-V LVCMOS
clk	Input	PIN_A7	3A_B	1.3-V LVCMOS
reset	Input	PIN_A12	3A_B	1.3-V LVCMOS

4.3.2.7 Right click in the spreadsheet part of Pin Planner. Select **Customize Columns**.

4.3.2.8 Add the **Weak Pull-Up Resistor** column. Select Weak Pull-up Resistor in Available Columns and press the > button. Select Weak Pull-up Resistor in the right hand column and then use the Move-Up button to place at under the I/O Standard column selection.

4.3.2.9 Close the Customize Columns window by selecting **OK**.





4.3.2.10 Select **on** in **the Weak Pull-up Resistor** column for the **reset** and **updown** Node Names

Named: * Edit:

Node Name	Direction	Location	I/O Bank	Fitter Location	I/O Standard	Weak Pull-Up Resistor
LED[0]	Output	PIN_AK20	5B	PIN_AK20	3.3-V LVCMOS	
LED[2]	Output	PIN_AH22	5A	PIN_AH22	3.3-V LVCMOS	
LED[1]	Output	PIN_AK21	5B	PIN_AK21	3.3-V LVCMOS	
clk	Input	PIN_A7	3A_B	PIN_A7	1.3-V LVCMOS	
reset	Input	PIN_A12	3A_B	PIN_A12	1.3-V LVCMOS	on
updown	Input	PIN_A14	3A_B	PIN_A14	1.3-V LVCMOS	on

4.3.2.11 Close the Pin Planner by selecting **File** → **Close**. The settings are automatically saved.

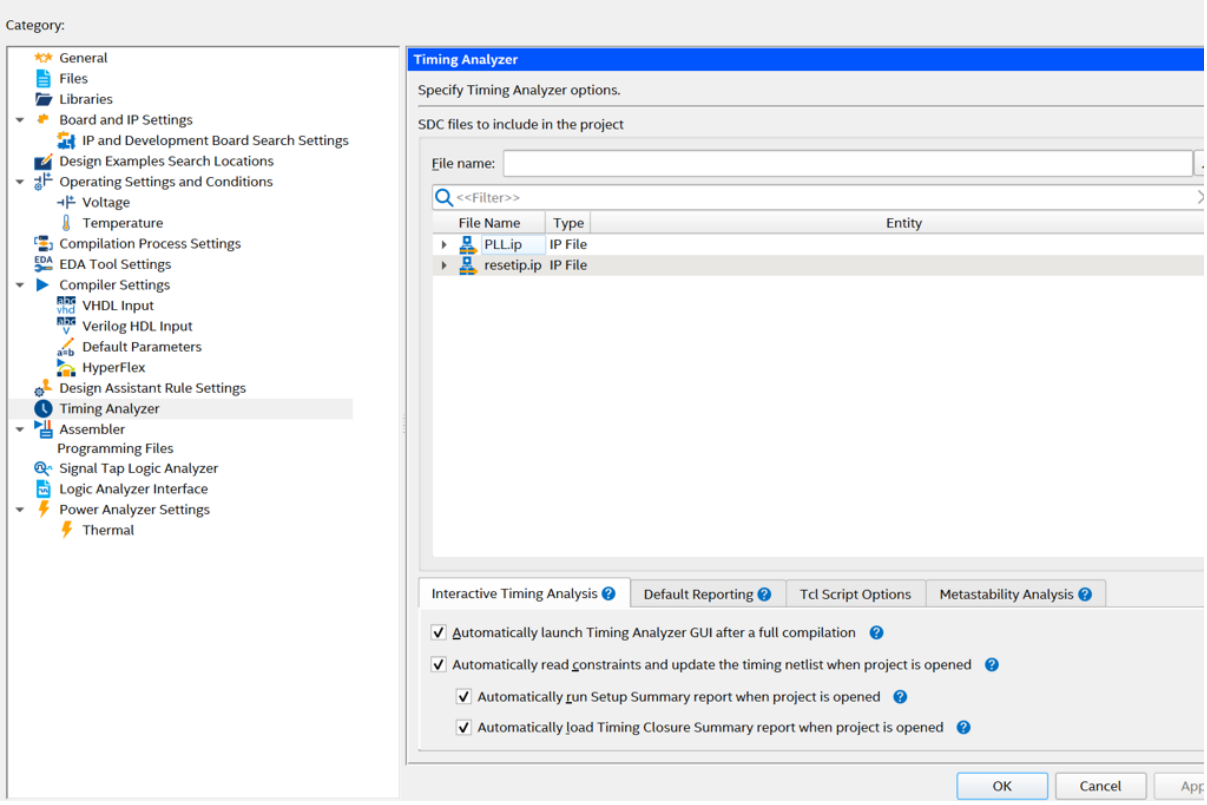
4.3.3 Timing Constraints

Timing Constraints are very important in a FPGA because they instruct Quartus on how fast/slow different parts of the design are to run. Quartus uses these constraints to meet/exceed the assignments. Providing accurate timing will enable the software to easily create a FPGA design. If the timing is over constrained the software will spend longer trying to meet the timing.

Timing constraints are stored in .sdc files, where sdc stands for Synopsys Design Constraint. SDC files are used for FPGA timing.

Quartus automatically generates and uses a .sdc file for the Altera PLL, thus the .sdc file does not need to be generated. It generates the .sdc file when the settings in the PLL IP are set. The .sdc file for the Reset Release IP are also automatically set and added.

4.3.3.1 Select **Assignments** → **Settings -> Timing Analyzer** to see the timing constraint files**.**



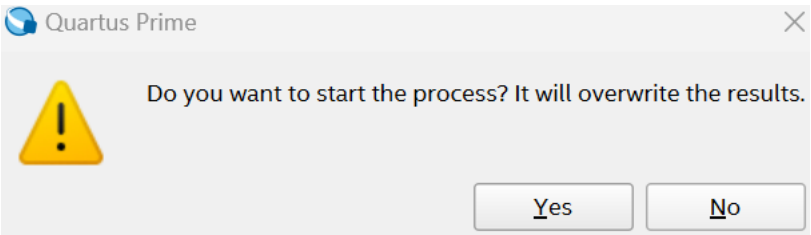
4.3.3.2. Select **Cancel** (bottom right of the window) as no changes were done.

4.3.4 Compiling the Design

Compilation is where the software takes the database that it created during the earlier step (Analysis and Synthesis) along with the pinouts and the timing constraints and then the software places and routes (connect) the design into the device. During the Assembler stage of the compilation, a programming file is generated, which can be programmed.

4.3.4.1 Start Compilation by clicking on ▶ button on the toolbars, or **Processing** → **Start Compilation**.

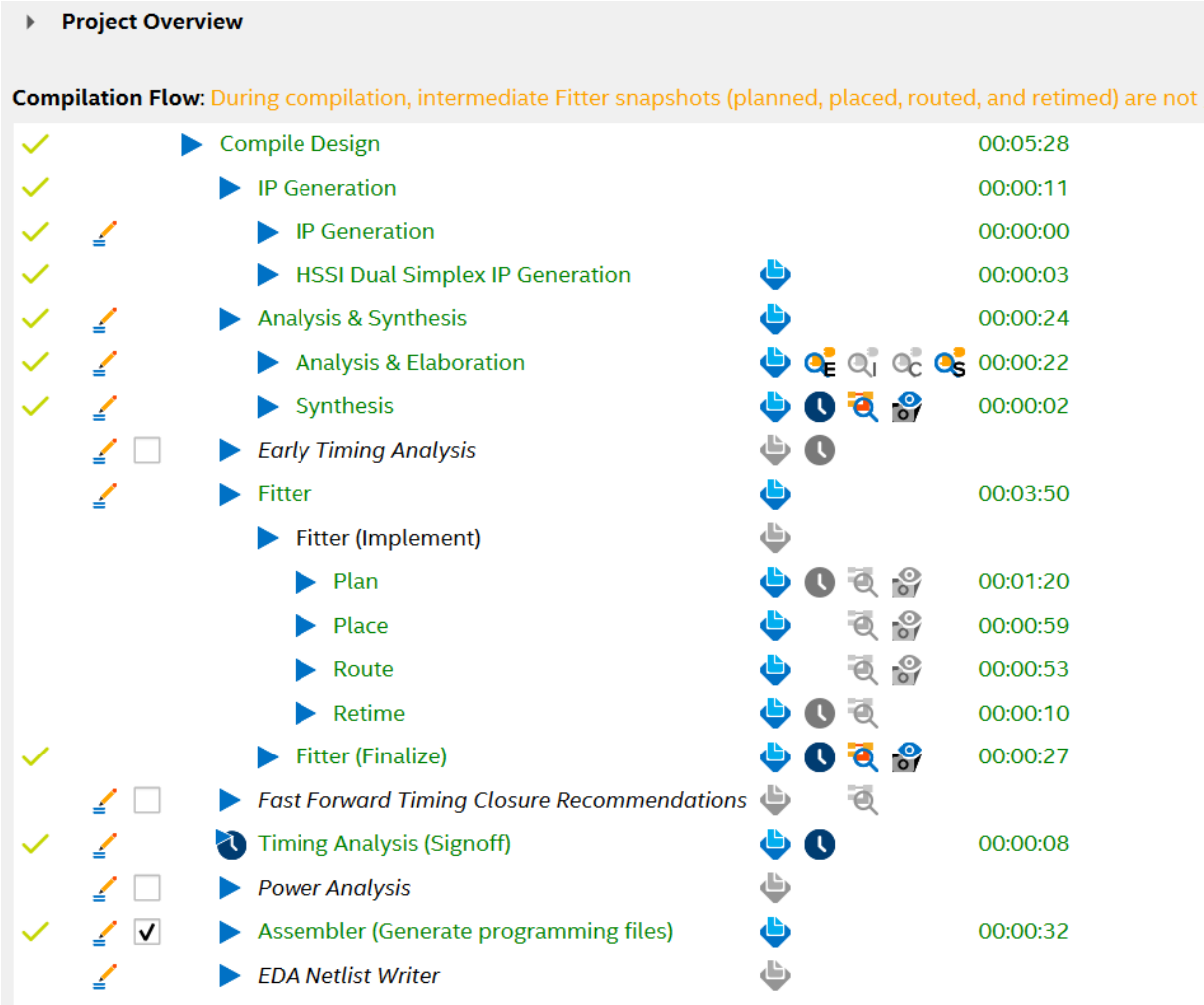
4.3.4.2 A window will be seen that states



4.3.4.3 Select **Yes**

4.3.4.4 There should be no errors during compilation. If there are errors, they should be fixed before re-compiling. The 100% in the lower right corner or a green checkmark next to the Compile Design in the Compilation task window indicates that the compilation was successful.

As Quartus compiles the design, there will be a progress bar that shows the steps that it does and the checkmarks as the steps are complete. Notice the checkmarks as steps are completed. The time used for each step depends on your computer.
There will be a Timing Analyzer Window that appears. **You can close the Timing Analyzer Window** with **File -> Close.** Our focus for this workshop is the compilation rather than timing.
Once the compilation is complete, the view should look like:



The time for the results varies depending on the computer used.

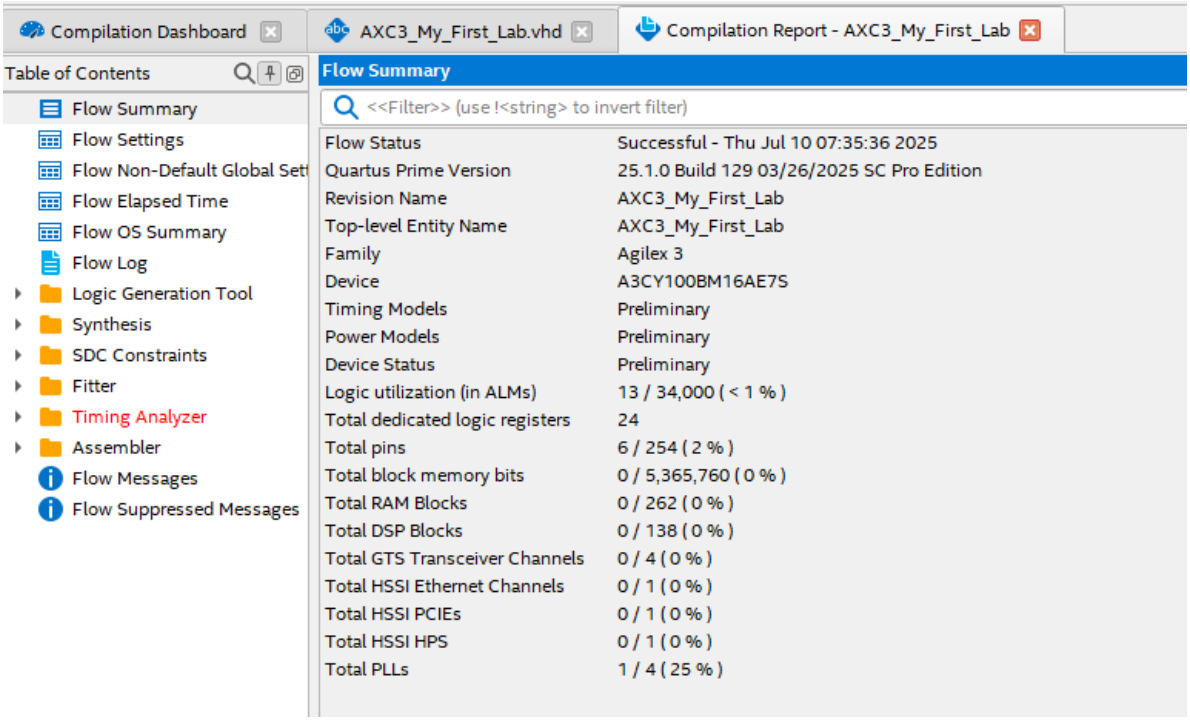
4.3.5 Compilation Report and Timing Results

Compilation report can be reviewed to see the number of resources used and the fmax of the Agilex 3.

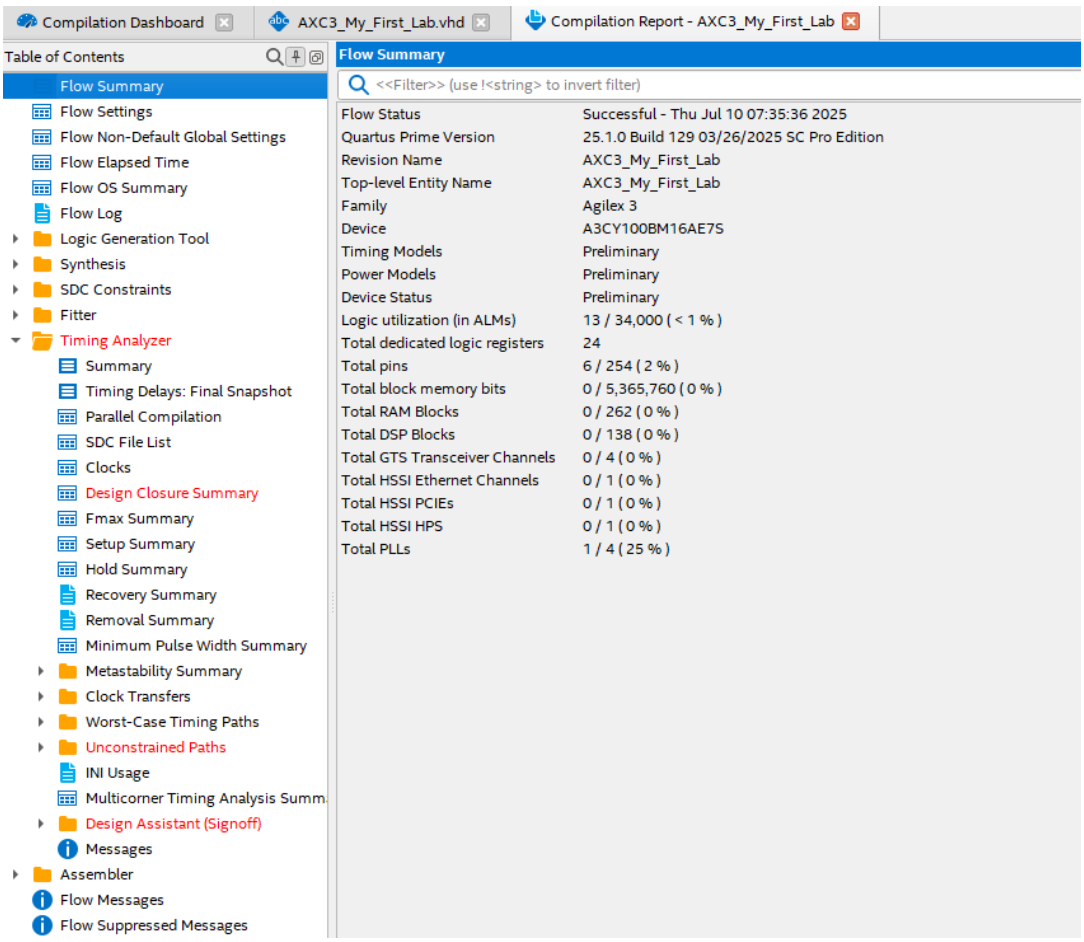
4.3.5.1 **Select Processing -> Compilation Report**.

The Compilation Report provides detailed reports of the compilation results. These results include items such as the number of resources being used, fmax of the design and more.

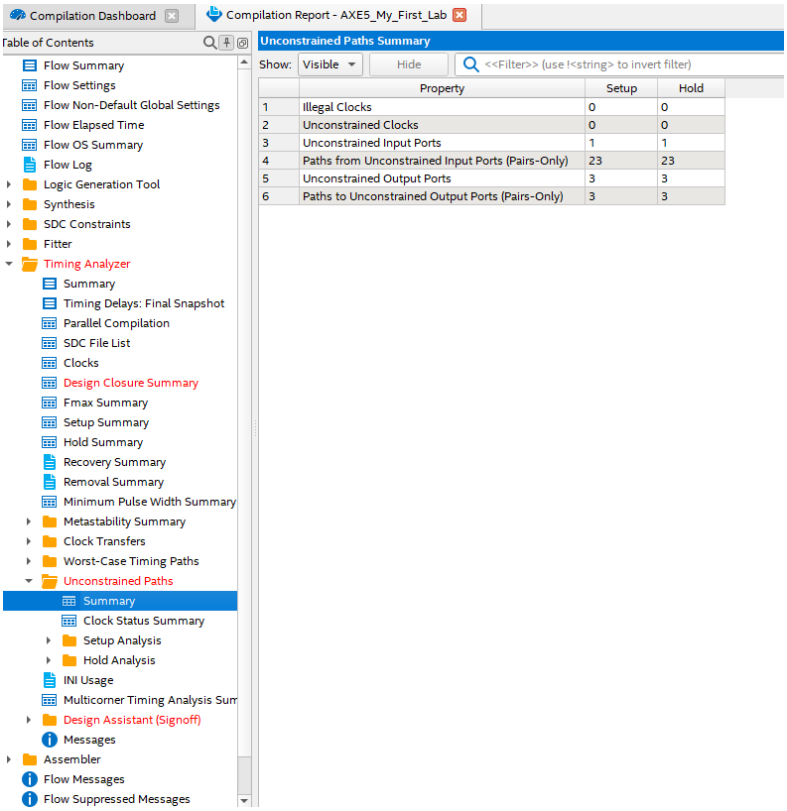
In this case, it shows that it uses less than the 1% of the logic being used, (ie 13 ALMs), 6 pins, 1 PLL but zero of the other resources. An ALM is an adaptive logic module, which is a building block of Agilex 3.
The text in red shows that items should be reviewed.



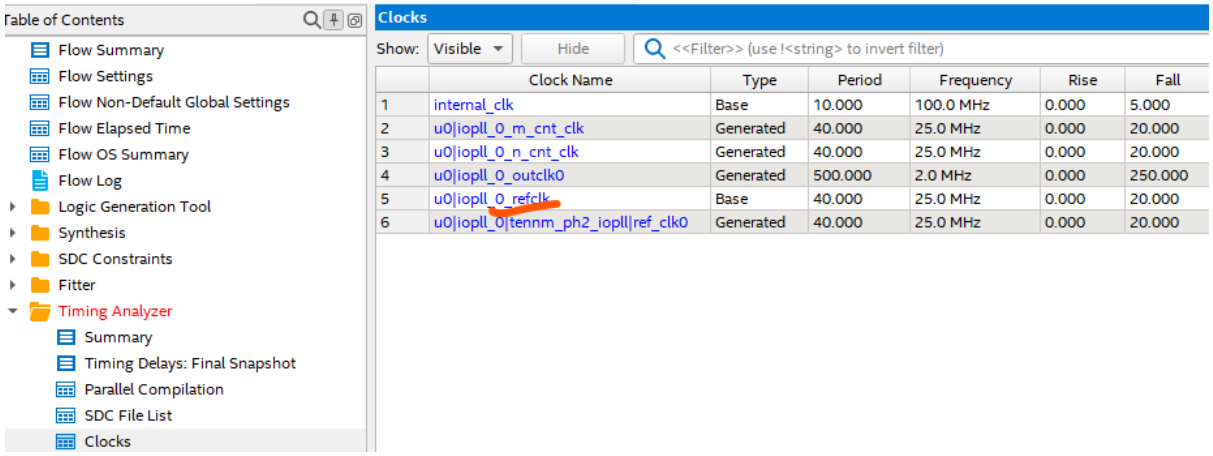
4.3.5.2 **Expand the Timing Analyzer** section to see the sections that have issues.



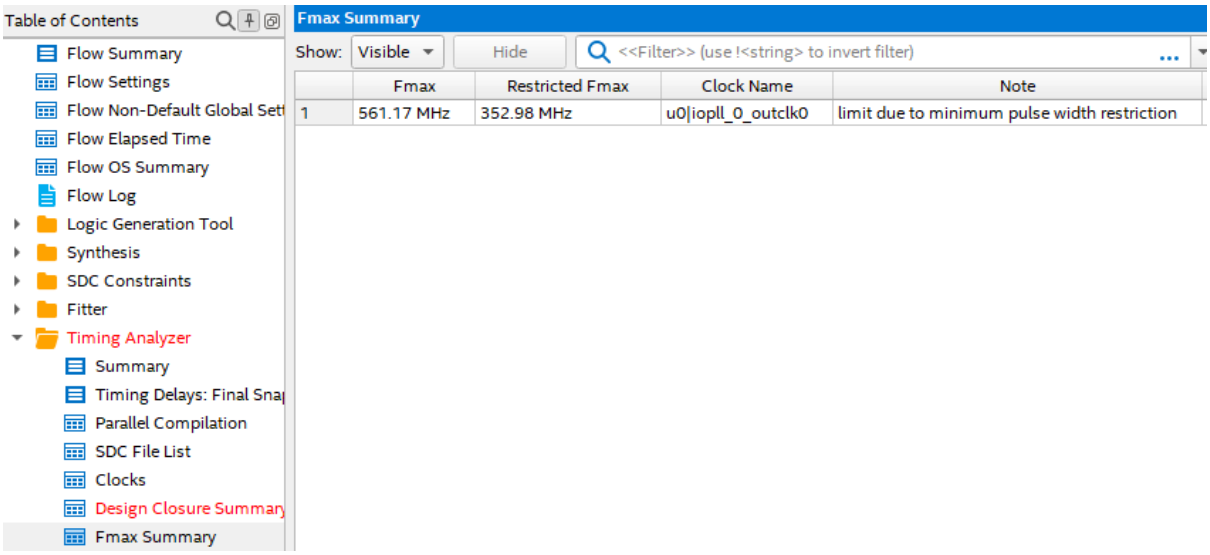
4.3.5.3 **Select the unconstrained paths and select Summary**. The input and output ports are not constrained with setup and hold constraints. Accurate timing analysis would need to have these timing constraints added for these pins. The timing constraints could be assigned with the .sdc file. In this case, this setup and hold for these input and output pins will not be assigned.



4.3.5.4 Select the **Clocks**. The clock of 25 MHz is the base fmax, which is the frequency of what the PLL was configured. Frequency is 1/period, thus the frequency of 25 MHz equals 1/40 (period). The generated clocks are the clocks that Quartus generates automatically.



4.3.5.5 **Select the Fmax summary** under the Timing Analyzer section. Note the fmax of 561.17 MHz. This is determined by the delay from the output of a register in the 24-bit counter, through a cloud of combinational logic to the input of the next register.



The clocks in the Agilex run very fast. As per the datasheet, the Agilex 3 in the 7S (which is the speed grade for the device on the board) the maximum fmax is 353 MHz.

Symbol	Parameter	Condition	Min	Typ	Max	Unit
f_{IN}	Input clock frequency source from reference clock input	-6S	10	—	900(57)	MHz
		-7S	10	—	625(57)	MHz
	Input clock frequency source from core clock input	-6S	10	—	415(57)	MHz
		-7S	10	—	353(57)	MHz

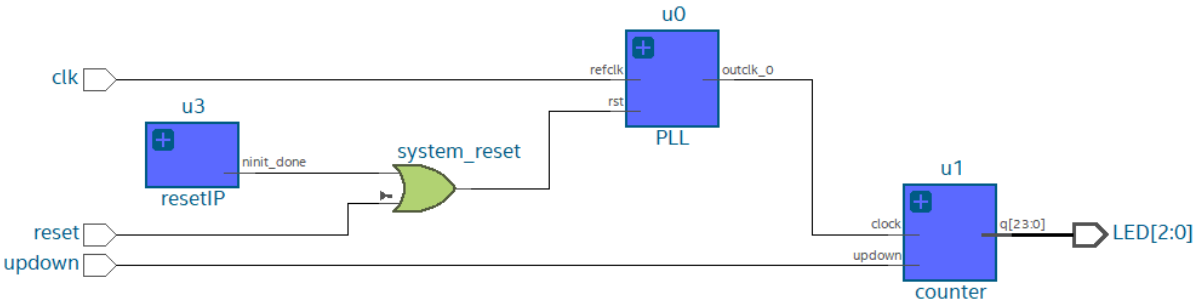
4.3.6 RTL Viewer

The RTL Viewer is a way to visually see the results of the RTL design.

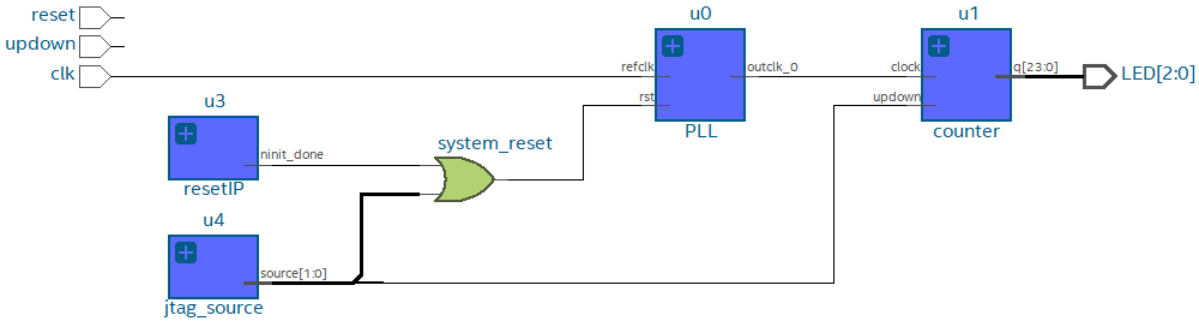
4.3.6.1. Select **Tools** → **Netlist Viewers** → **RTL Analyzer** (**Elaborated**). This view shows the RTL as seen by Quartus. This view can be useful to see how Quartus "interpreted" your RTL code and thus it can be seen if the code shown is what is expected. In this case, the RTL Viewer shows the expected view. (There are other options in the Netlist Viewer that can be explored). Elaborated means the view of the design after the step of Analysis and Elaboration (after netlist extraction) but before the mapping and synthesis or fitter optimizations. Elaborated closely corresponds to the original source code.

The RTL Analyzer view will vary depending on whether the code was uncommented for local hardware verification or board farm verification.

The image below shows the RTL Analyzer view for local board verification.



The image below shows the RTL Analyzer view for board farm verification. Note that the reset and updown signals are no longer driven by input pins, but rather by the jtag_source ip block. This allows for remote control of these signals via JTAG.



4.3.6.2. In the **RTL Analyzer** window, select **File** → **Close**.

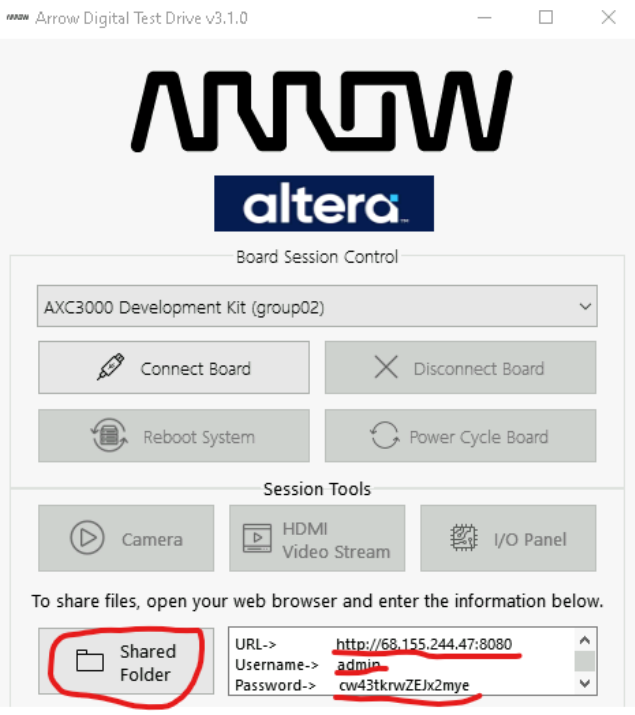
4.4 Verify the Design Locally

The next step is to see if the design works. If you have built the lab using CloudLabs then complete section 4.4.1. If you have built the lab on a local PC then skip to section 4.4.2.

4.4.1 File Download

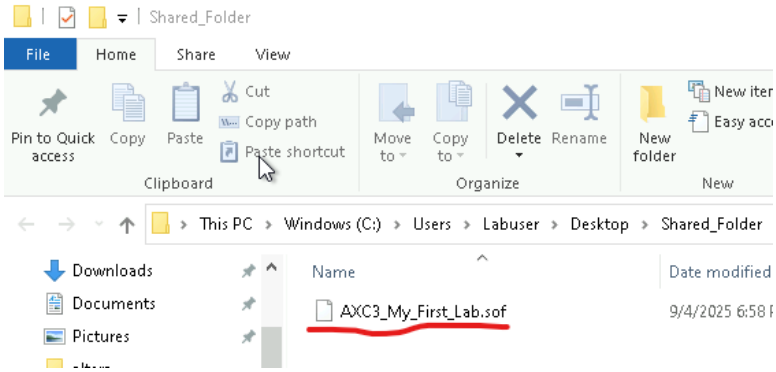
Click on the ArrowDTD icon in CloudLabs to launch the tool.

4.4.1.1 Open the **Shared Folder** utility in the CloudLab VM. Open a browser in your local PC. Navigate to the **URL** indicated. Use the provided username and password.

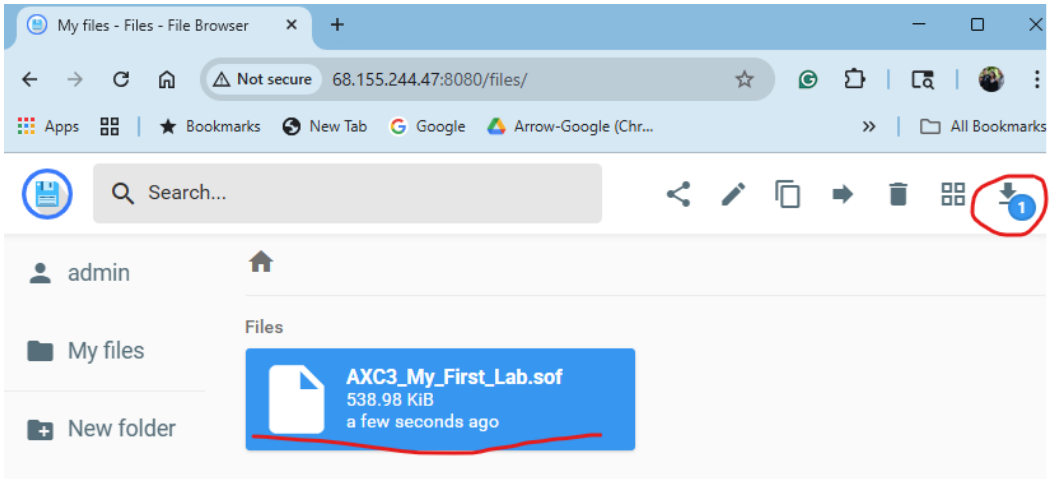


4.4.1.2 **Drag** and drop the following file into the **Shared** folder.

c:\altera_workshops\axc3000\My_First_Lab\output_files\AXC3_My_First_Lab.sof



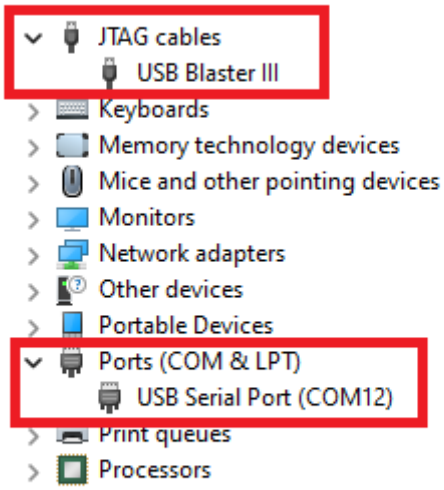
4.4.1.3 This will be reflected in the **browser** on the local PC. **Download** the SOF file to a convenient location on the local PC.



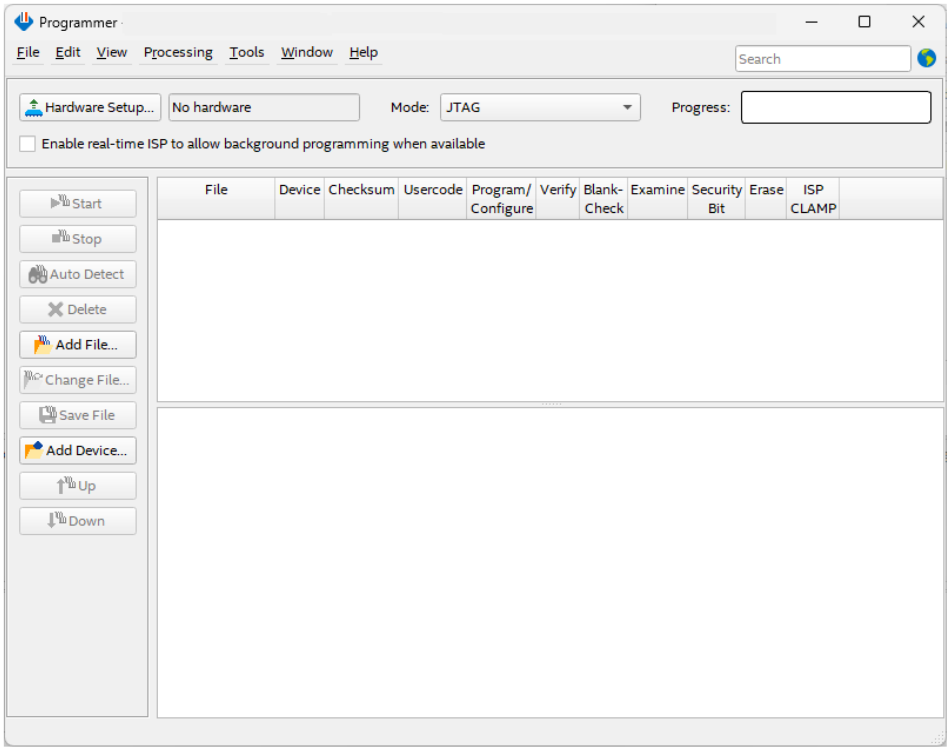
4.4.2 Program the FPGA

The next step is to **program** the design into the FPGA on the board. The **rest of the instructions** will be done on the **local computer**.

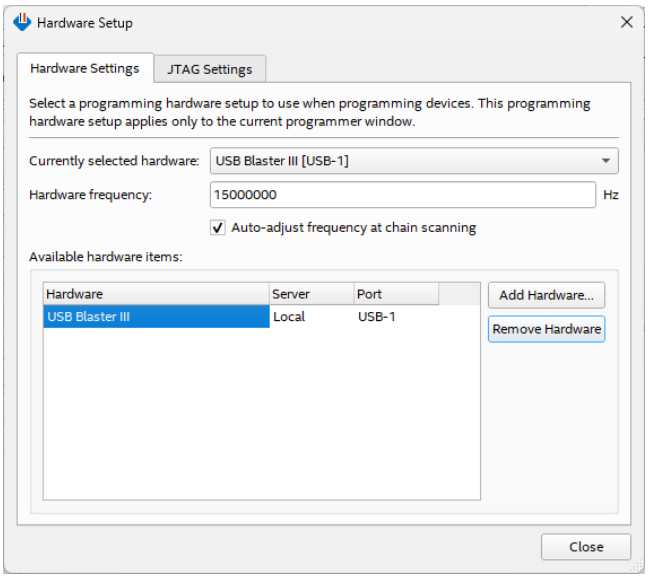
4.4.2.1 **Connect your AXC3000 to your PC using an USB C cable**. Since the Altera USB Blaster III should be already installed, the Window's Device Manager should display the following entries, highlighted in red (COM port number may differ depending on your PC):



4.4.2.2 From the Start menu open **Altera 25.1.1.125 Pro Edition --> Programmer**.

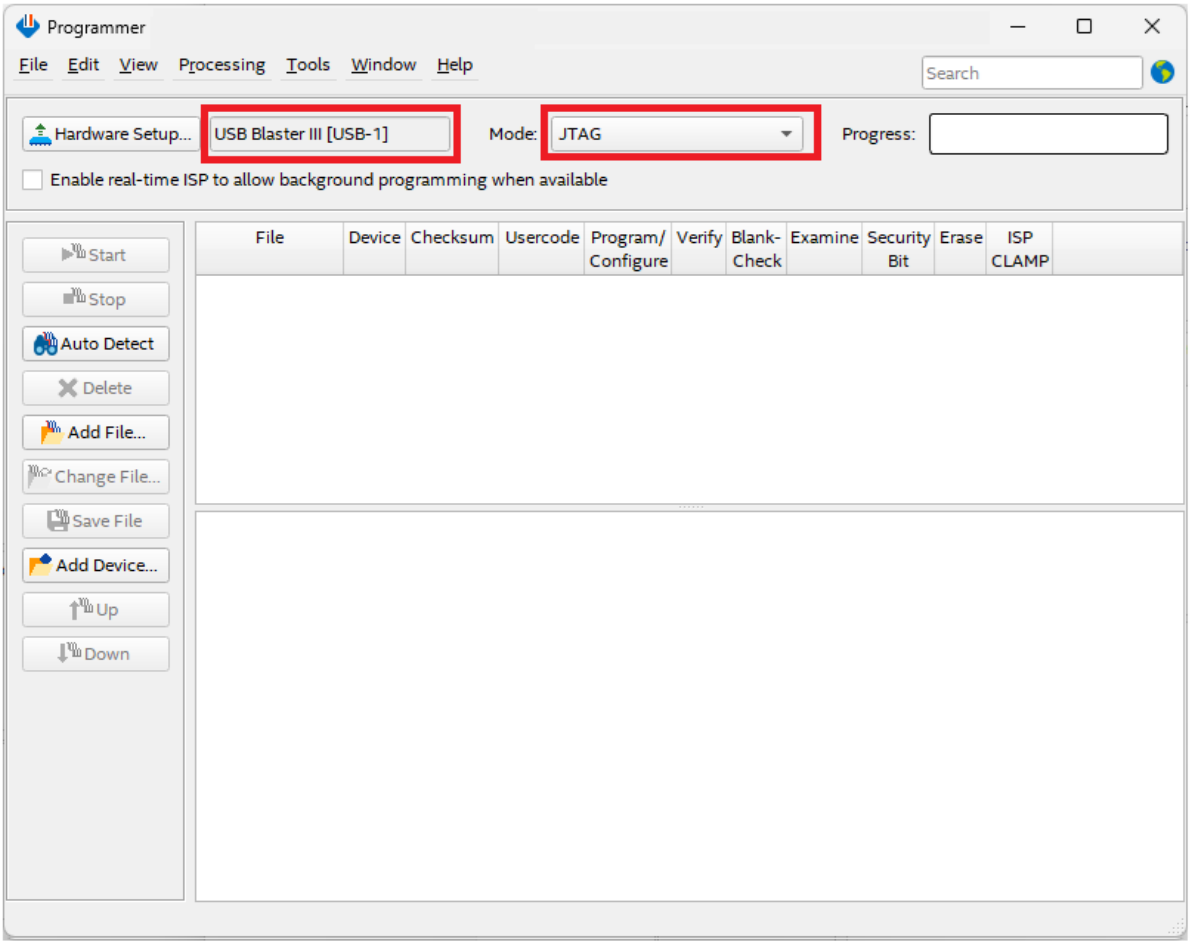


4.4.2.3 Click **Hardware Setup...** and double click **USB Blaster III** entry in the Hardware Setup window. **Currently selected hardware** should now show USB Blaster III [USB0] (the USB port number may be different depending on your PC).



4.4.2.4 Click **Close**.

4.4.2.5 Make sure the hardware setup is Altera-USB-Blaster-III [USB0] and the mode is JTAG. Click **Auto Detect**.



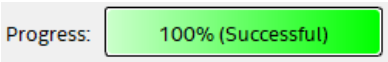
4.4.2.6 **Double click** the first <**none**> line item to choose the programming file.

4.4.2.7 If the file was downloaded from Cloudlabs then navigate to the **folder where the sof file was downloaded** and select the **.sof** file. If the design was built **locally**, then select it in the **output_files** sub directory.

4.4.2.8 Click **Open**.

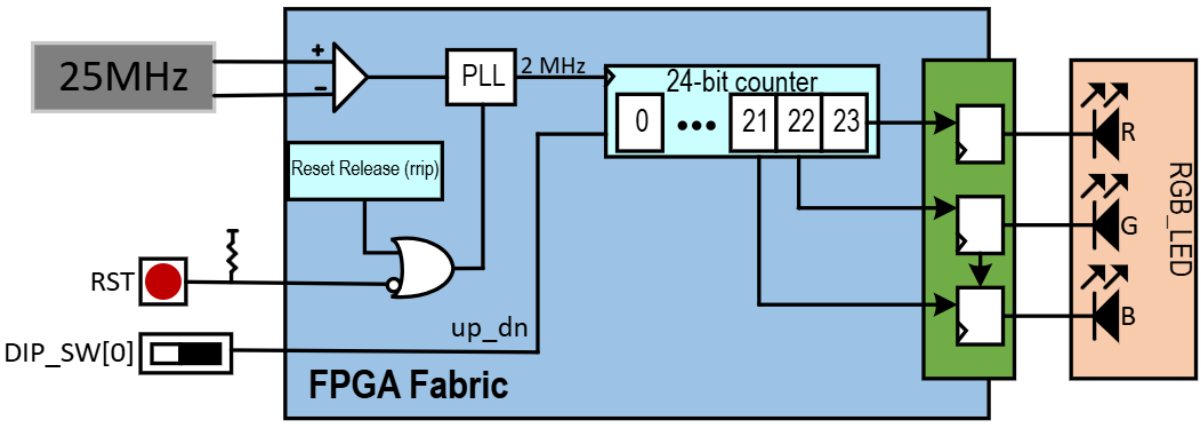
4.4.2.9 Make sure the Programmer shows the correct file and the correct part in the JTAG chain and check the **Program/Configure** checkbox.

4.4.2.10 Click **Start** to program the board. When the configuration is complete, the Progress bar should show 100% (Successful).



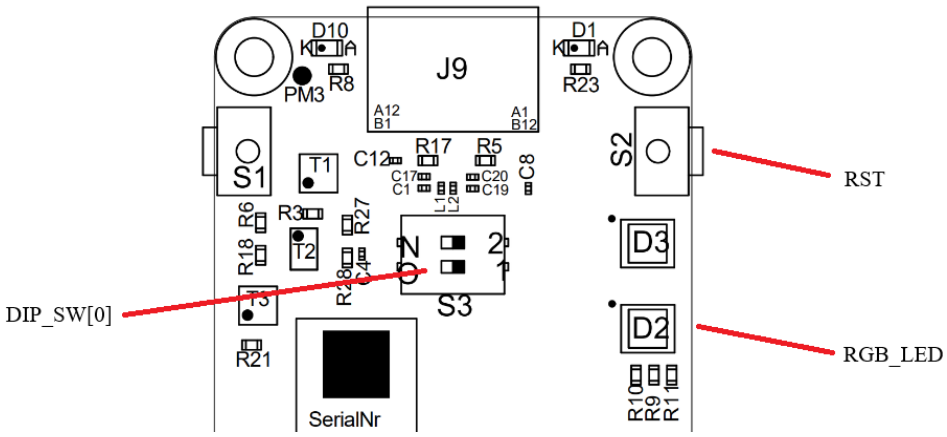
A block diagram of the design is shown below. In summary, the PLL generates a 2 MHz output clock from the 25 MHz input clock. This is used to clock a 24-bit binary counter. The three most significant bits of the counter are connected to the red, green, and blue LEDs of RGB0. The LED will cycle through eight colors, with RED being on the longest since it is connected to the most significant bit of the counter.

If you press the RST push button, the design will be held in reset and the LED colors stop changing. Changing the position of DIP_SW[0] will reverse the direction of the colors, since the count is down instead of up.



4.4.2.11 On the AXC3000 Agilex 3 board, the LED should be lit. Try using the pushbutton to see what changes that you see.

The picture below shows the location of the DIP switch, push button and LED on the AXC3000 board.



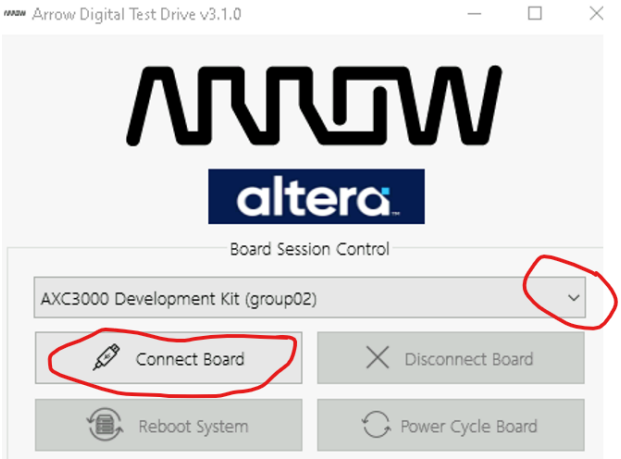
4.5 Verify the Design Remotely

The following instructions require the user to be utilizing a CloudLabs Windows Virtual Machine.

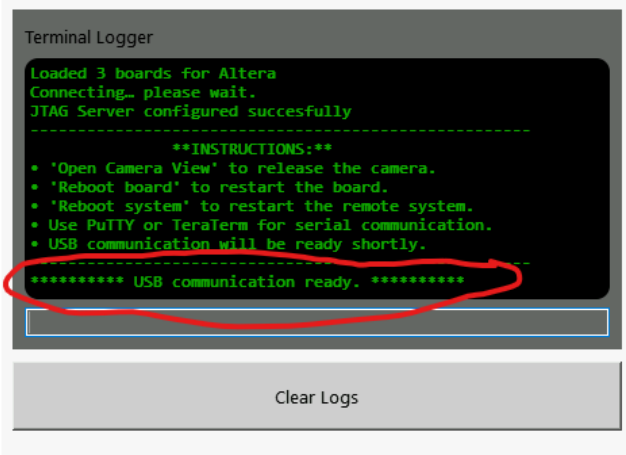
4.5.1 Connect to the Board Farm

4.5.1.1 Connect to the remote AXC3000 Board using the ArrowDTD tool. Click on the ArrowDTD icon to launch the tool.

4.5.1.2 Select an available board from the dropdown menu and click connect board.



Wait until the board is ready.

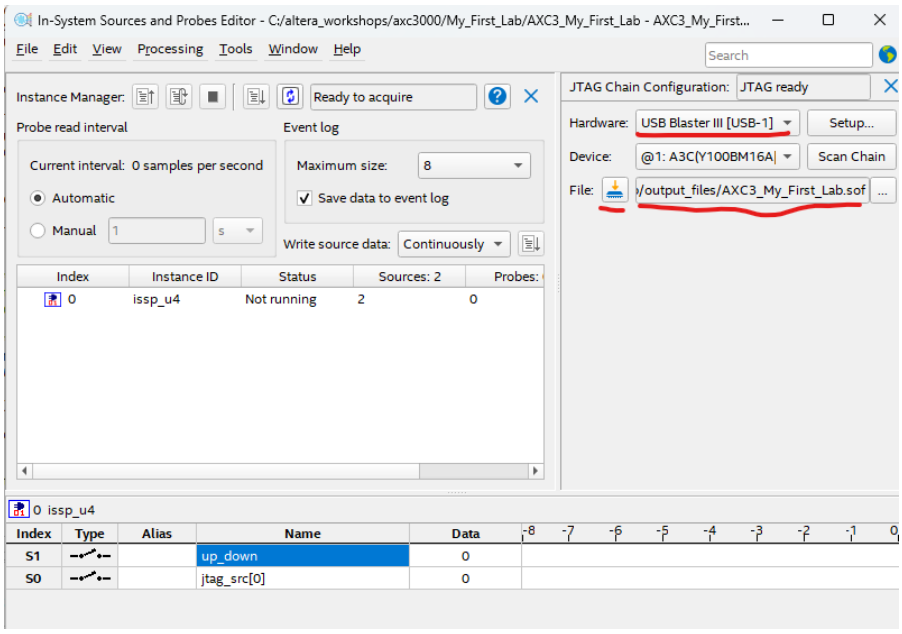


4.5.1.3 Open the Camera View of the board.

4.5.2 Programming – Configuration

4.5.2.1 Open the Quartus Sources Editor. Tools --> In-System Sources and Probes Editor.

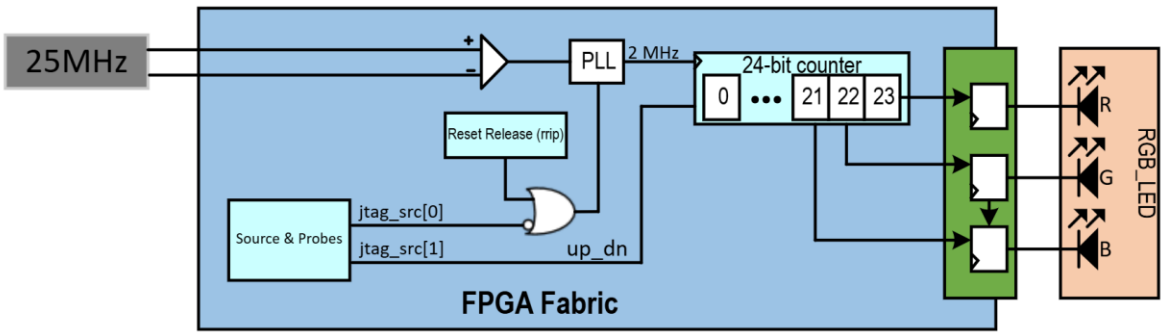
4.5.2.2 Program the device. Select the USB Blaster III in the Hardware dropdown field. Wait until the JTAG chain has been scanned. This can take up to 20 seconds. Select the output_files/AC3_My_First_Lab.sof file to download and then press the File download button. Observe the color sequencing of AXC3000 board LEDs in the camera view.



4.5.3 Testing the Design

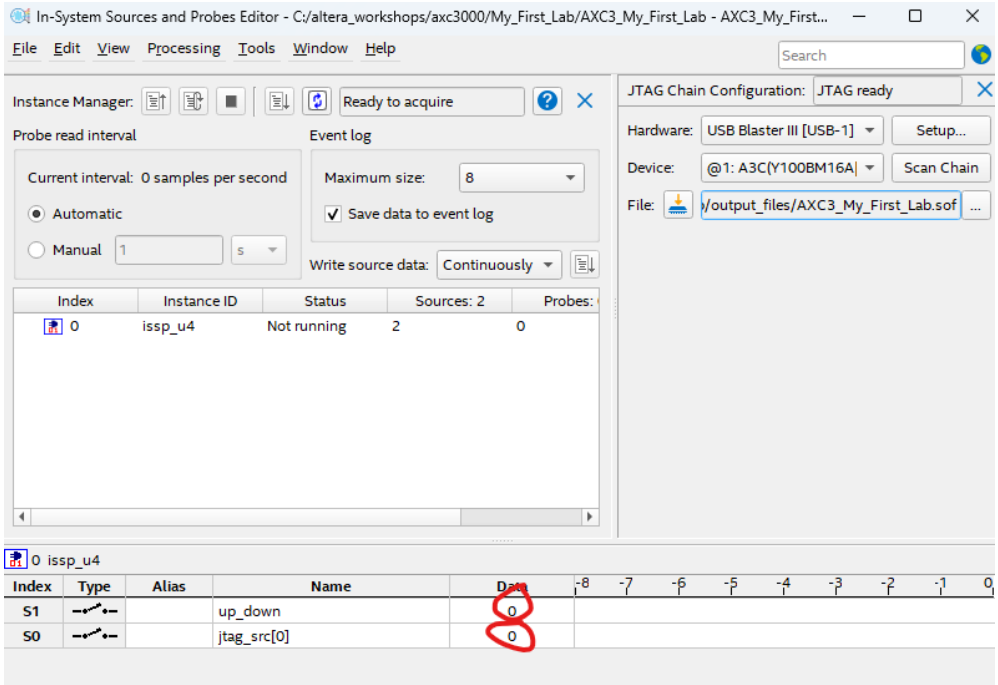
A block diagram of the design is shown below. In summary, the PLL generates a 2 MHz output clock from the 25 MHz input clock. This is used to clock a 24-bit binary counter. The three most significant bits of the counter are connected to the red, green, and blue LEDs of RGB0. The LED will cycle through eight colors, with RED being on the longest since it is connected to the most significant bit of the counter.

If you assert the jtag_src[0] signal, the design will be held in reset and the LED colors stop changing. Changing the value of the updown signal will reverse the direction of the colors, since the count is down instead of up. The updown and reset signals are controlled remotely via JTAG using the Source & Probes instantiation in the design.

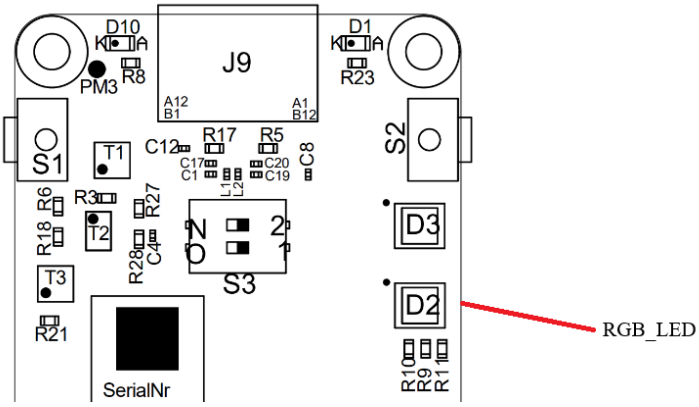


4.5.3.1 The RGB LED should be lit. Try toggling the values of the two signals (reset, updown) to see what changes occur. The signal values are toggled using the Quartus Sources and Probes Editor via JTAG (using the USB Blaster III) since

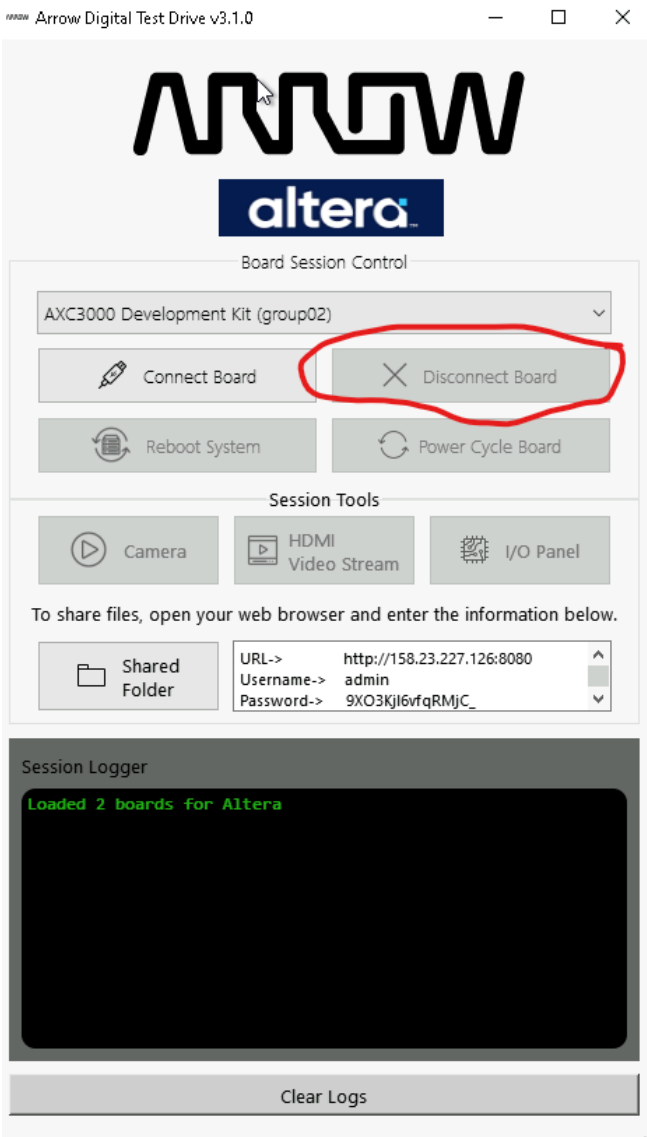
the board is possibly thousands of miles away in a remote lab. Click on the fields shown below to change the signal values. Please note that there can be a delay of 5 - 10 seconds for the signals to change due to the remote access.



The picture below shows the location of the LED on the AXC3000 board.



4.5.3.2 Please disconnect the board when debugging is complete. This will free up the board resource for others to use.



4.6 Congratulations

Summary: This workshop provided comprehensive information to help you

-understand the FPGA flow for design entry to programming

-create and parametrize IP (intellectual property)

-add assignments (such as pin assignments)

-run a complete compile

-view your design in the RTL Viewer

-review the resource and clock resources

and run the design onto the development board.

CONGRATULATIONS! YOU HAVE SUCCESSFULLY COMPLETED THE FIRST DESIGN LAB!

5 Legal Disclaimer

ARROW ELECTRONICS

EVALUATION BOARD LICENSE AGREEMENT
By using this evaluation board or kit (together with all related software, firmware, components, and documentation provided by Arrow, "Evaluation Board"), You ("You") are agreeing to be bound by the terms and conditions of this Evaluation Board License Agreement ("Agreement"). Do not use the Evaluation Board until You have read and agreed to this Agreement. Your use of the Evaluation Board constitutes Your acceptance of this Agreement.

PURPOSE
The purpose of this evaluation board is solely intended for evaluation purposes. Any use of the Board beyond these purposes is on your own risk. Furthermore, according the applicable law, the offering Arrow entity explicitly does not warrant, guarantee or provide any remedies to you with regard to the board.

LICENSE
Arrow grants You a non-exclusive, limited right to use the enclosed Evaluation Board offering limited features only for Your evaluation and testing purposes in a research and development setting. Usage in a live environment is prohibited. The Evaluation Board shall not be, in any case, directly or indirectly assembled as a part in any production of Yours as it is solely developed to serve evaluation purposes and has no direct function and is not a finished product.

EVALUATION BOARD STATUS
The Evaluation Board offers limited features allowing You only to evaluate and test purposes. The Evaluation Board is not intended for consumer or household use. You are not authorized to use the Evaluation Board in any production system, and it may not be offered for sale or lease, or sold, leased or otherwise distributed for commercial purposes.

OWNERSHIP AND COPYRIGHT
Title to the Evaluation Board remains with Arrow and/or its licensors. This Agreement does not involve any transfer of intellectual property rights ("IPR) for evaluation board. You may not remove any copyright or other proprietary rights notices without prior written authorization from Arrow or it licensors.

RESTRICTIONS AND WARNINGS
Before You handle or use the Evaluation Board, You shall comply with all such warnings and other instructions and employ reasonable safety precautions in using the Evaluation Board. Failure to do so may result in death, personal injury, or property damage.

You shall not use the Evaluation Board in any safety critical or functional safety testing, including but not limited to testing of life supporting, military or nuclear applications. Arrow expressly disclaims any responsibility for such usage which shall be made at Your sole risk.

WARRANTY
Arrow warrants that it has the right to provide the evaluation board to you. This warranty is provided by Arrow in lieu of all other warranties, written or oral, statutory, express or implied, including any warranty as to merchantability, non-infringement, fitness for any particular purpose, or uninterrupted or error-free operation, all of which are expressly disclaimed. The evaluation board is provided "as is" without any other rights or warranties, directly or indirectly.

You warrant to Arrow that the evaluation board is used only by electronics experts who understand the dangers of handling and using such items, you assume all responsibility and liability for any improper or unsafe handling or use of the evaluation board by you, your employees, affiliates, contractors, and designees.

LIMITATION OF LIABILITIES

In no event shall Arrow be liable to you, whether in contract, tort (including negligence), strict liability, or any other legal theory, for any direct, indirect, special, consequential, incidental, punitive, or exemplary damages with respect to any matters relating to this agreement. In no event shall arrow's liability arising out of this agreement in the aggregate exceed the amount paid by you under this agreement for the purchase of the evaluation board.

IDENTIFICATION

You shall, at Your expense, defend Arrow and its Affiliates and Licensors against a claim or action brought by a third party for infringement or misappropriation of any patent, copyright, trade secret or other intellectual property right of a third party to the extent resulting from (1) Your combination of the Evaluation Board with any other component, system, software, or firmware, (2) Your modification of the Evaluation Board, or (3) Your use of the Evaluation Board in a manner not permitted under this Agreement. You shall indemnify Arrow and its Affiliates and Licensors against and pay any resulting costs and damages finally awarded against Arrow and its Affiliates and Licensors or agreed to in any settlement, provided that You have sole control of the defense and settlement of the claim or action, and Arrow cooperates in the defense and furnishes all related evidence under its control at Your expense. Arrow will be entitled to participate in the defense of such claim or action and to employ counsel at its own expense.

RECYCLING

The Evaluation Board is not to be disposed as an urban waste. At the end of its life cycle, differentiated waste collection must be followed, as stated in the directive 2002/96/EC. In all the countries belonging to the European Union (EU Dir. 2002/96/EC) and those following differentiated recycling, the Evaluation Board is subject to differentiated recycling at the end of its life cycle, therefore: It is forbidden to dispose the Evaluation Board as an undifferentiated waste or with other domestic wastes. Consult the local authorities for more information on the proper disposal channels. An incorrect Evaluation Board disposal may cause damage to the environment and is punishable by the law.