Be Boulder.

DE10-Lite PWM Guide

Version 1.0 **09/15/2017** **ECEN 5863**

Table of Contents

OVERVIEW

The DE10-Lite is a FPGA evaluation kit that is designed to get you started with using an FPGA. The DE10-Lite adopts Altera's non-volatile MAX® 10 FPGA built on 55-nm flash process. MAX 10 FPGAs enhance non-volatile integration by delivering advanced processing capabilities in a low-cost, instant-on, small form factor programmable logic device. The devices also include full-featured FPGA capabilities such as digital signal processing, analog functionality, Nios II embedded processor support, and memory controllers.
The DE10-Lite includes a variety of peripherals connected to the FPGA device, such as 64MB SDRAM, accelerometer, VGA interface, 7-segment LEDs, LEDs, switches, pushbuttons and several different options for expansion connectivity.

Module Summary:
The Hardware labs are based on completing the initial portion of the design of a simple voltmeter.
In **Section 1** you will prepare for the project acquiring files and other resources.
In **Section 2** you will examine the system design.
In **Section 3** you will learn how to use Create a system design using Quartus Prime.
In **Section 4** you will learn how to simulate your design using ModelSim.
In Section 5 you will place and route the design.
In Section 8 you will create a programming file that could be used to configure the FPGA fabric.

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

Caution:

Do not continue until you have read the following:

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

1. Getting Started

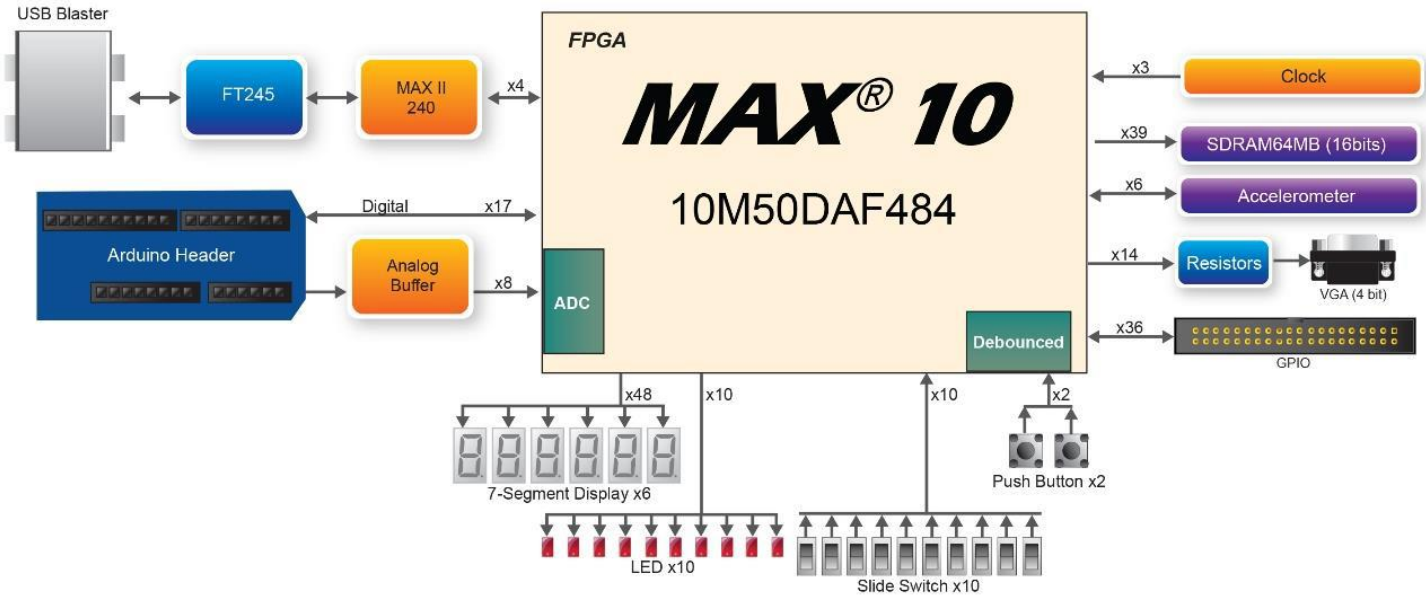
Your first objective is to ensure that you have all of the items needed and to install the tools so that you are ready to create and run your design.

List of Required Items:

- Altera Quartus Prime Version 16.1 FPGA Development Software Tool
- **Module Design Files: pwm_led_top.bdf, debouncer.v etc.**
- Terasic DE10-Lite Development Kit

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

1.1. DE10-Lite Development Kit



1.2. Extract the Lab Files.

- Create a folder **C:\AlteraPrj\DE10litePWM** on your PC.
- Copy PWM.zip to this directory
- Extract to \PWM

CONGRATULATIONS!!

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

2. System Design

Section Objective

In this Section you will review the architecture of the design that will be created in Quartus Prime with the DE10-Lite Kit as the target. Typically, a design starts with system requirements. These system requirements become inputs to the system definition. System definition is then the first step for implementation in the design flow process.

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

• Creating a Quartus Prime Project

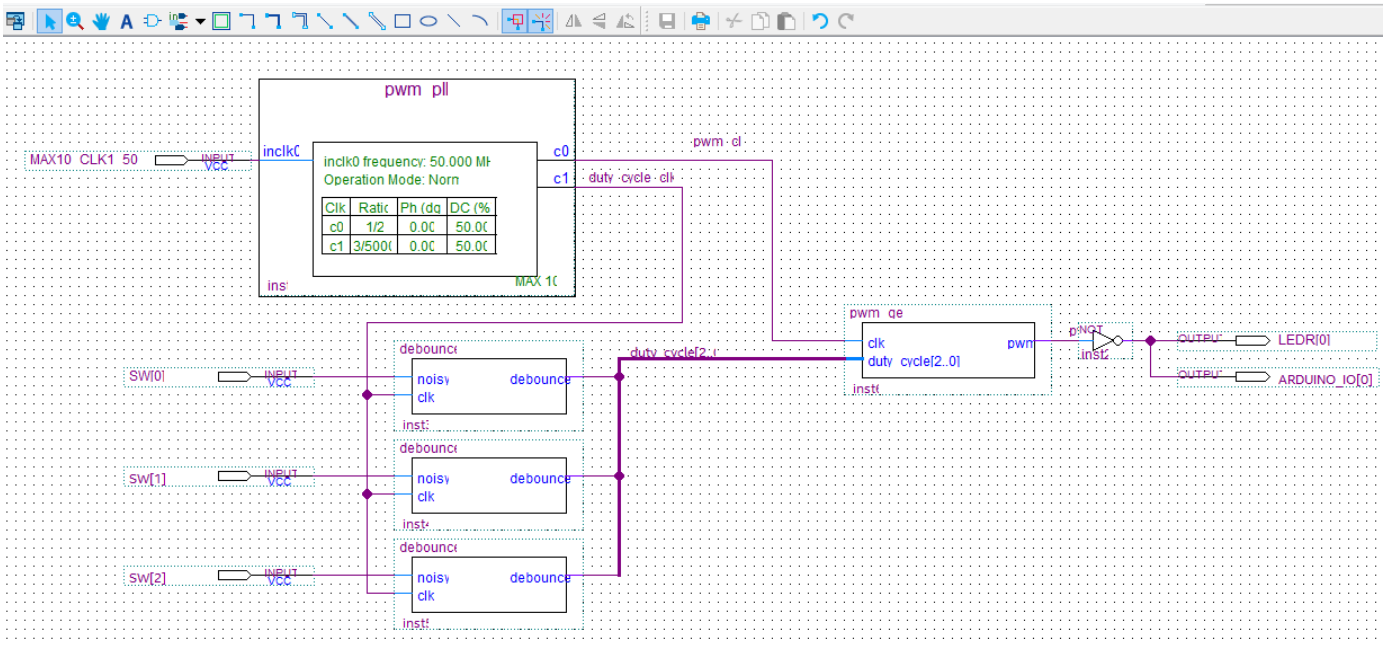
• Implementing a MAX 10 design with Schematic Entry, including inclusion of HDL-based blocks.

• Simulating the design, running place and route, creating pin assignments and a programming file for the MAX10 silicon

2.1. System Requirements

During this exercise, you will follow a step-by-step guide to create a simple design. To do this you will configure a PLL block, connect together a few simple blocks created with Verilog code, assign pins, and download to the target board. The inputs are a System Clock at 50 MHz, and 10 toggle switch inputs to encode the PWM duty cycle. The output is an LED, and a pin on the Arduino GPIO Connector.

2.1.1 System Block Diagram



The design above is a circuit that will be used to generate a PWM output which will drive one of the LEDs on the kit and vary the LED intensity. 3 toggle switches on the kit will be used to increase or decrease the PWM duty cycle value between 0 and 100% in 8 steps, 12.5% each. The **`pwm_pll`** block is a PLL within the MAX 10 FPGA device which will be used to take the incoming 50 MHz clock input and generate lower clock frequencies used to clock the PWM logic. The debouncer blocks are created from Verilog code to sample the incoming switches, debounce them and generate a single pulse output when the switch is thrown. The **`pwm_gen`** block is created from Verilog code and generates the PWM'ed output which will drive the LED and Arduino output.

Components of MAX10 Device Used

This tutorial uses the MAX10 pll block and logic fabric.

2.2. PWM Generation

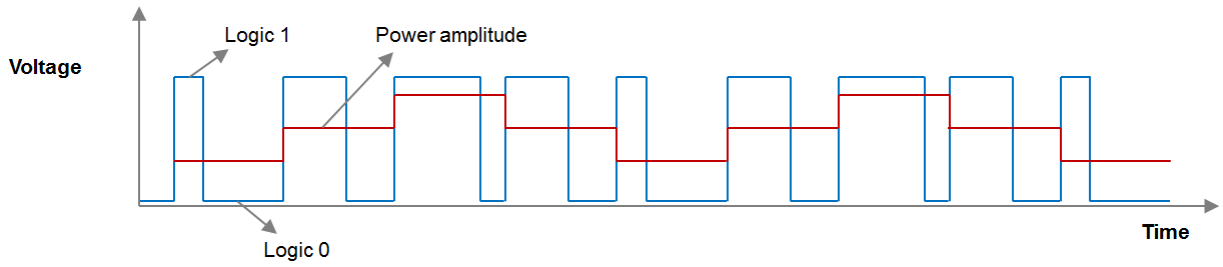
Pulse width modulation (PWM) is a simple method of using a rectangular digital waveform to create an analog output. PWM uses the width of the pulse to represent an amplitude.

The period of the wave is usually kept constant, and the pulse width, or ON time is varied.

The duty cycle is the proportion of time that the pulse is ON or HIGH, and is expressed as a percentage:

Duty cycle = 100% * (pulse ON time)/(pulse period)

Whatever duty cycle a PWM stream has, there is an average value. If the ON time is small, the average value is low; if the ON time is large, the average value is high. Therefore, by controlling the duty cycle, we control the average output value (represented as the red line below).



CONGRATULATIONS!!

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

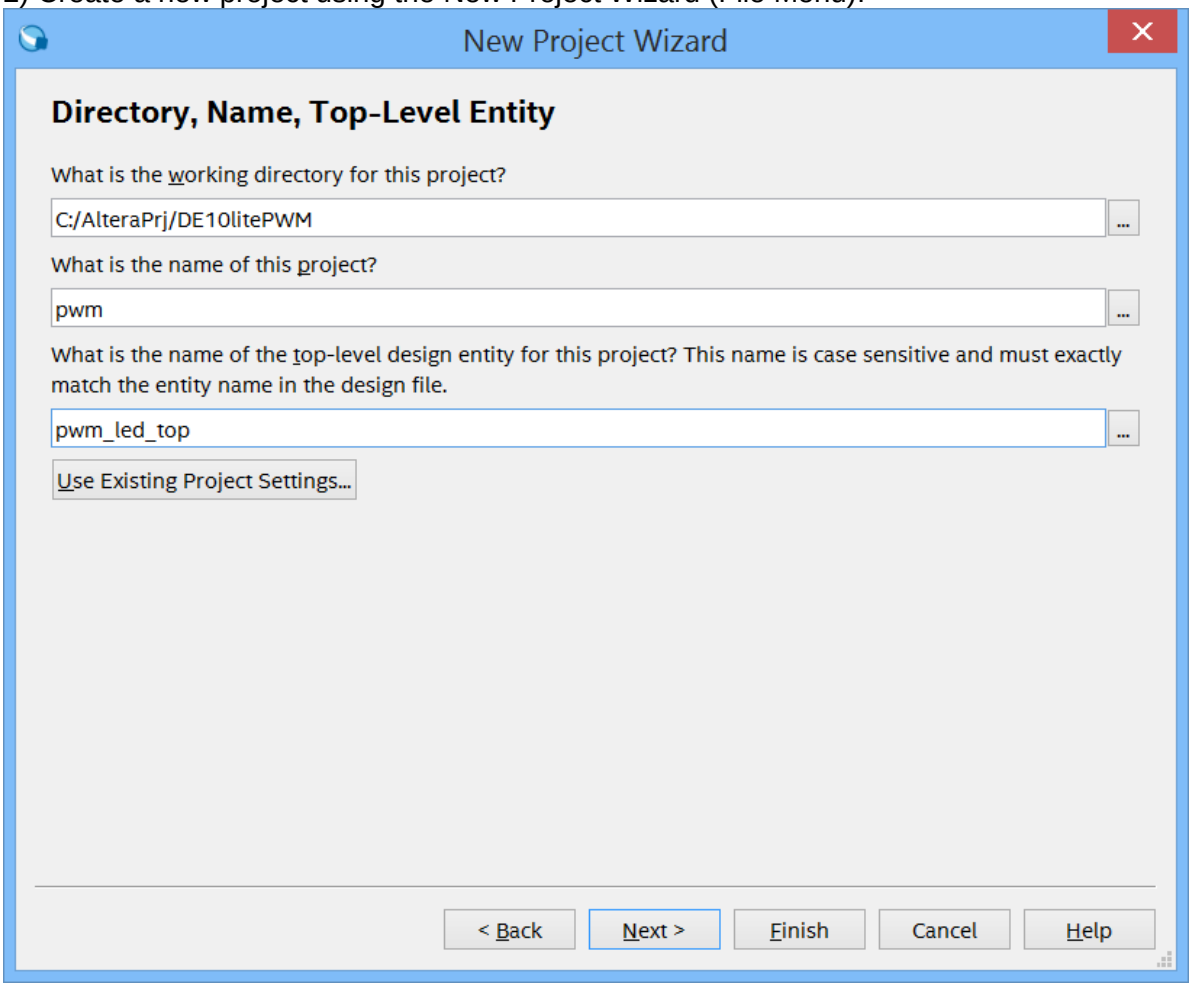
3. Creating the Design

In this section you will create the fabric design using SmartDesign. Some source files have been provided in th e lab files, as the design is combination of HDL files, IP cores, and Macro Blocks.

3.1. Create the Quartus Project

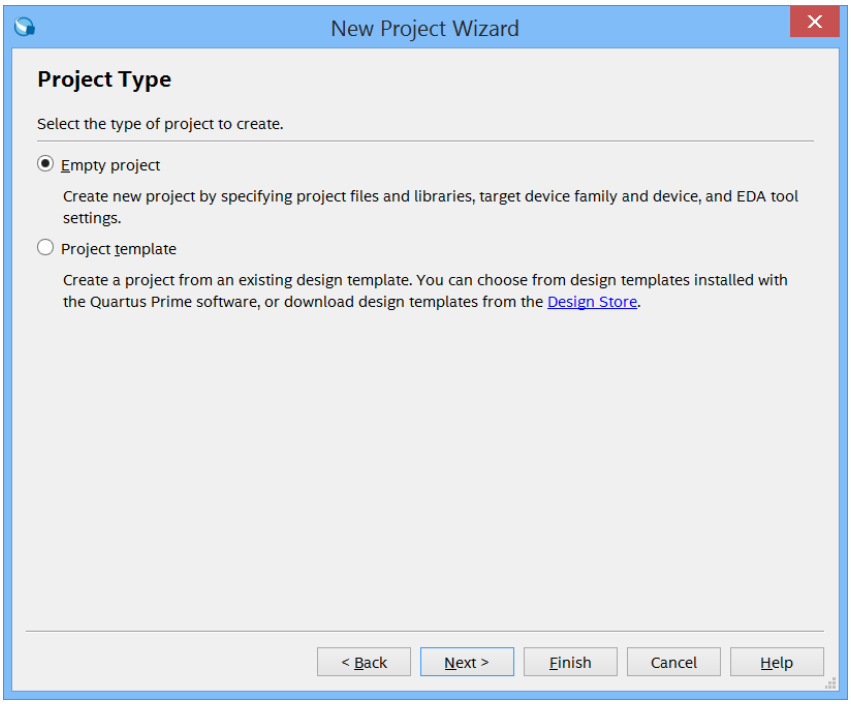
1) Launch Quartus Prime 16.1 (64-bit)

2) Create a new project using the New Project Wizard (File Menu).

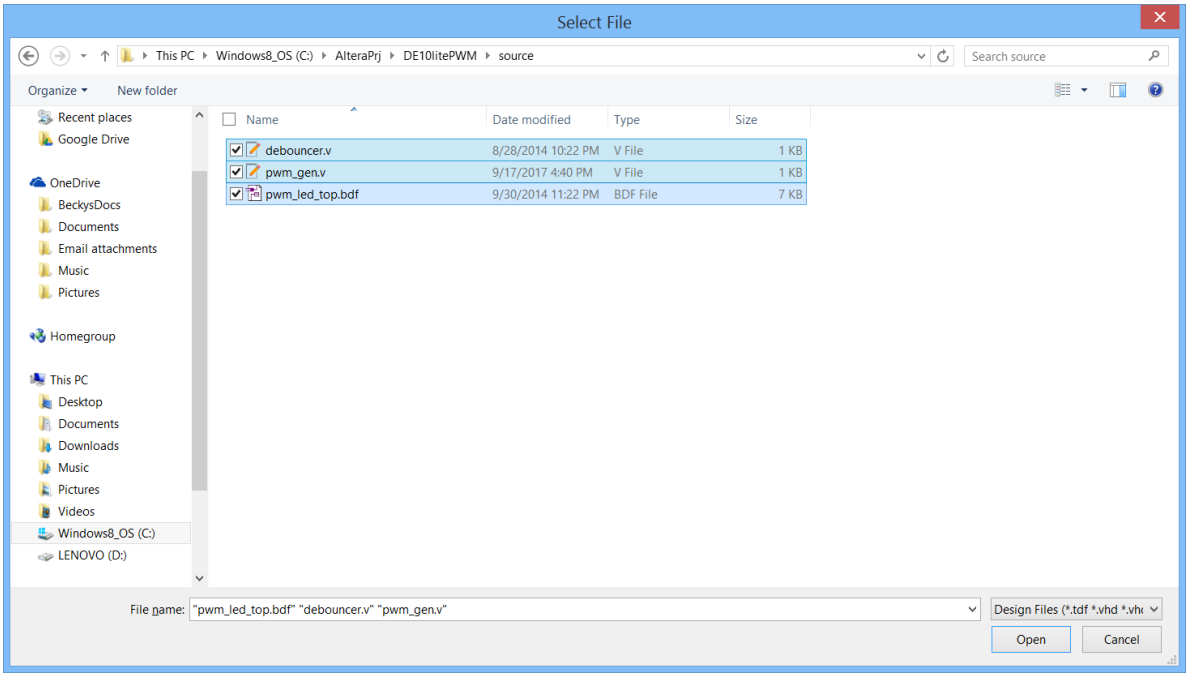


3) Browse to the path where you extracted the lab files: **c:\AlteraPrj\PWM**
Specify the name of the project: **pwm**
Specify the name of the top-level design entity: **pwm_led_top** (It is a common naming convention to include the word “top” in the top-level design entity to make it clear and obvious which entity is at the top of the hierarchy.)

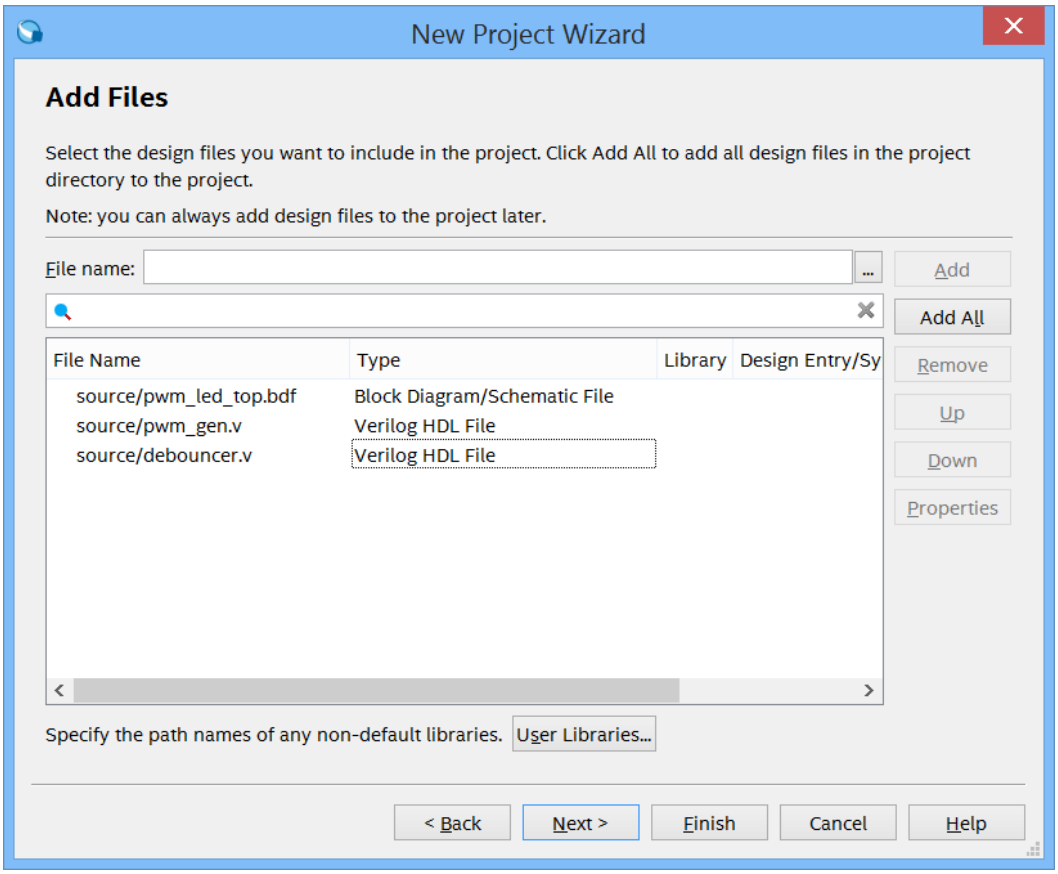
4) Click Next. Select Empty Project.



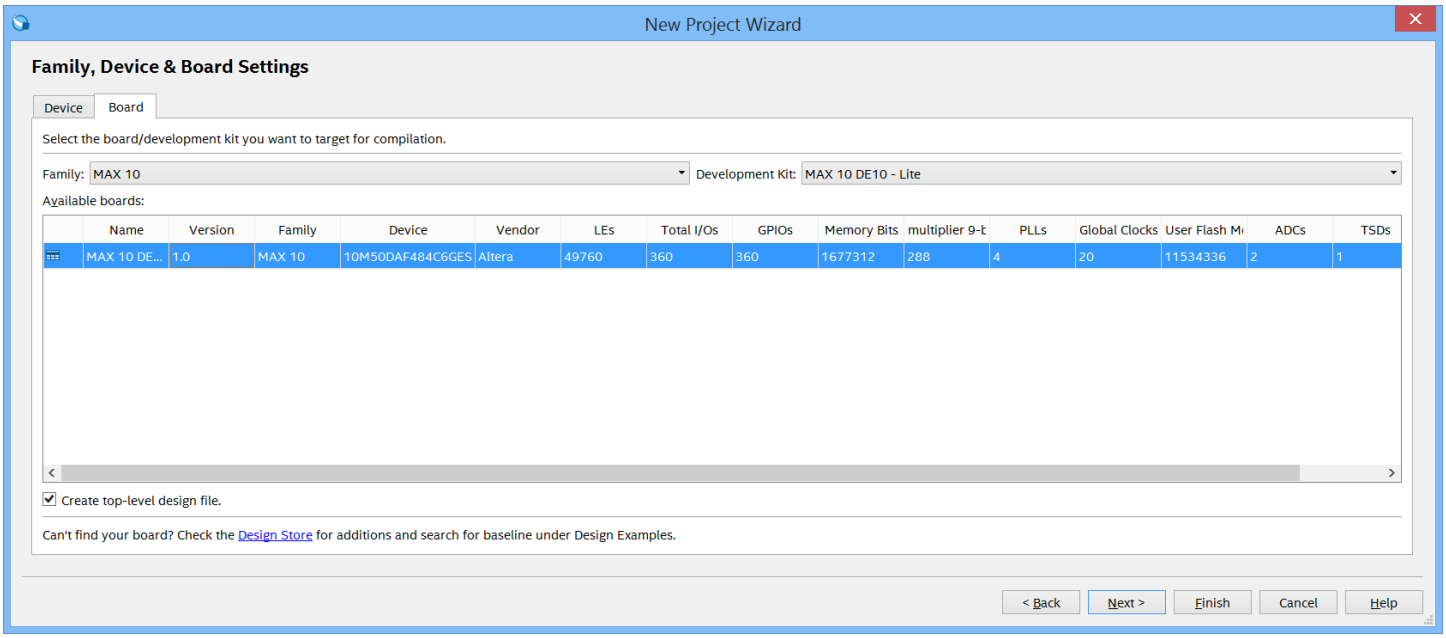
5) Add Files [page 2 of 5] and click on the button and browse into the **source** directory where you located the two provided Verilog files (**debouncer.v** and **pwm_gen.v**) and the provided schematic file (**pwm_led_top.bdf**). Select all 3 files and Add them to the files listing.



Once you have correctly added the 3 files, your Add Files dialogue box should look as follows:

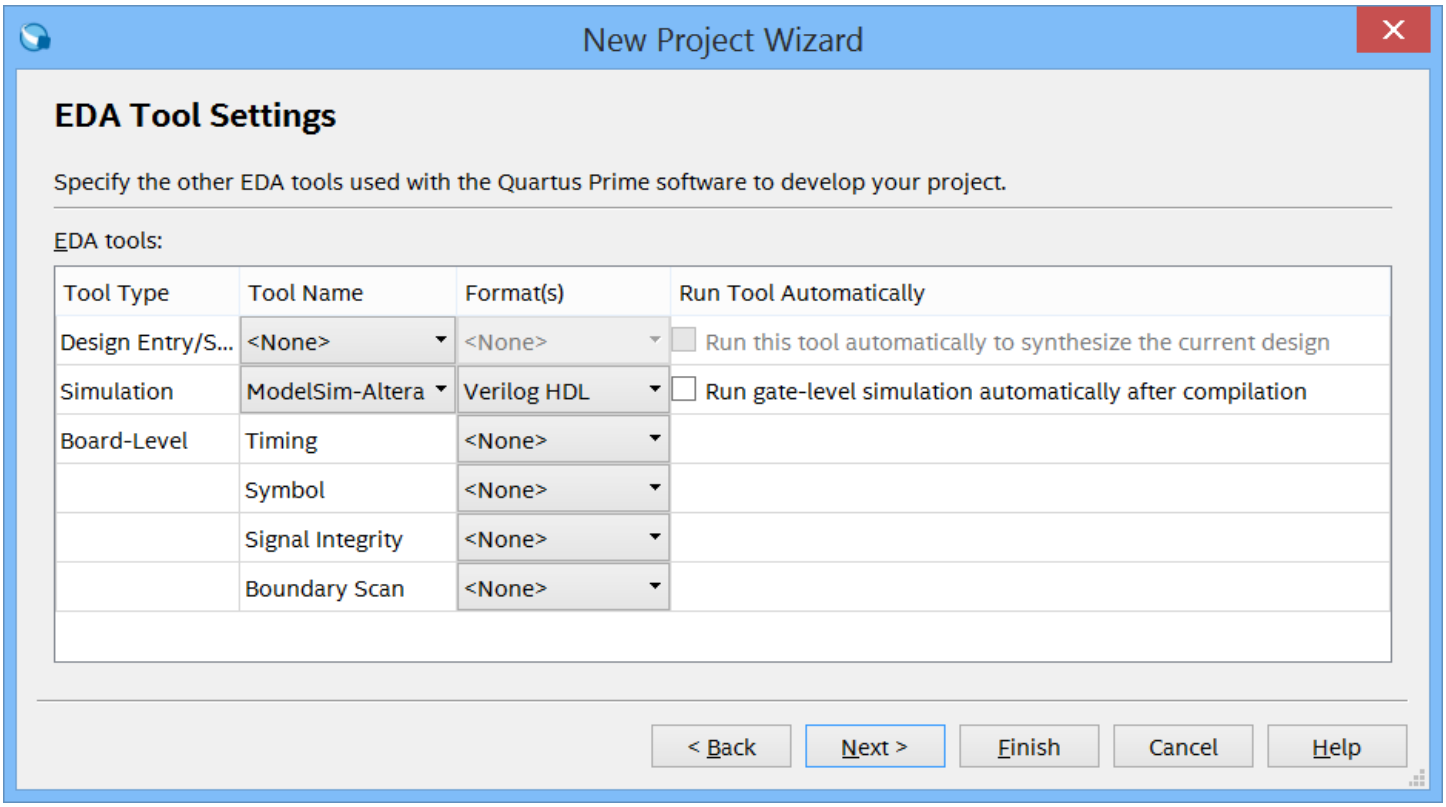


6) Click next to reach: Family, Device, and Board Setting [page 3 of 5], select the device on your kit by using the filters as shown below. Select Board, MAX10 for the family, and then MAX10 DE10-Lite for the kit. This automatically selects the 10M50DAF484C6GES part.



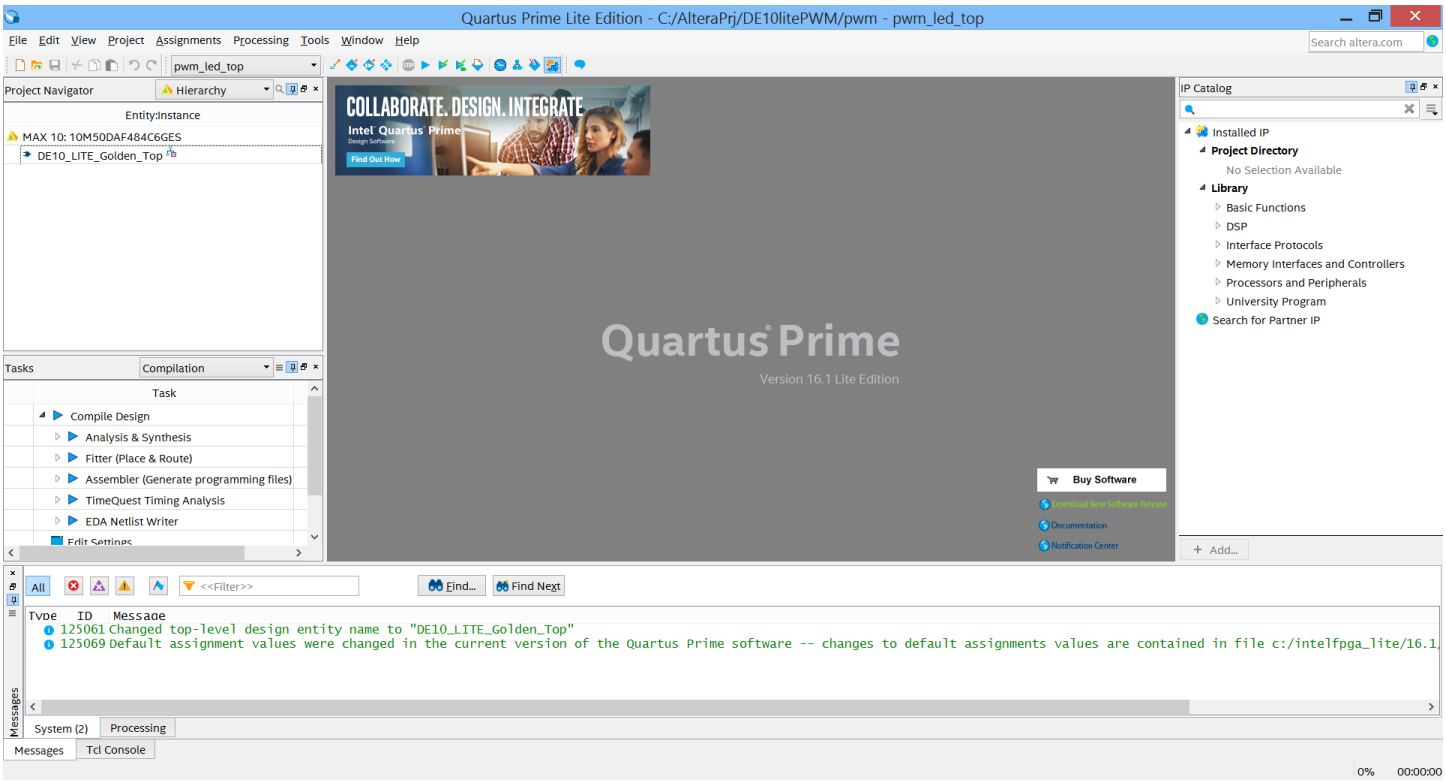
After making your selection, look at your kit and confirm that the part number marked on your device matches your selection.

7) In EDA Tool Settings, Select Modelsim-Altera for the Simulation Tool. Click Next.

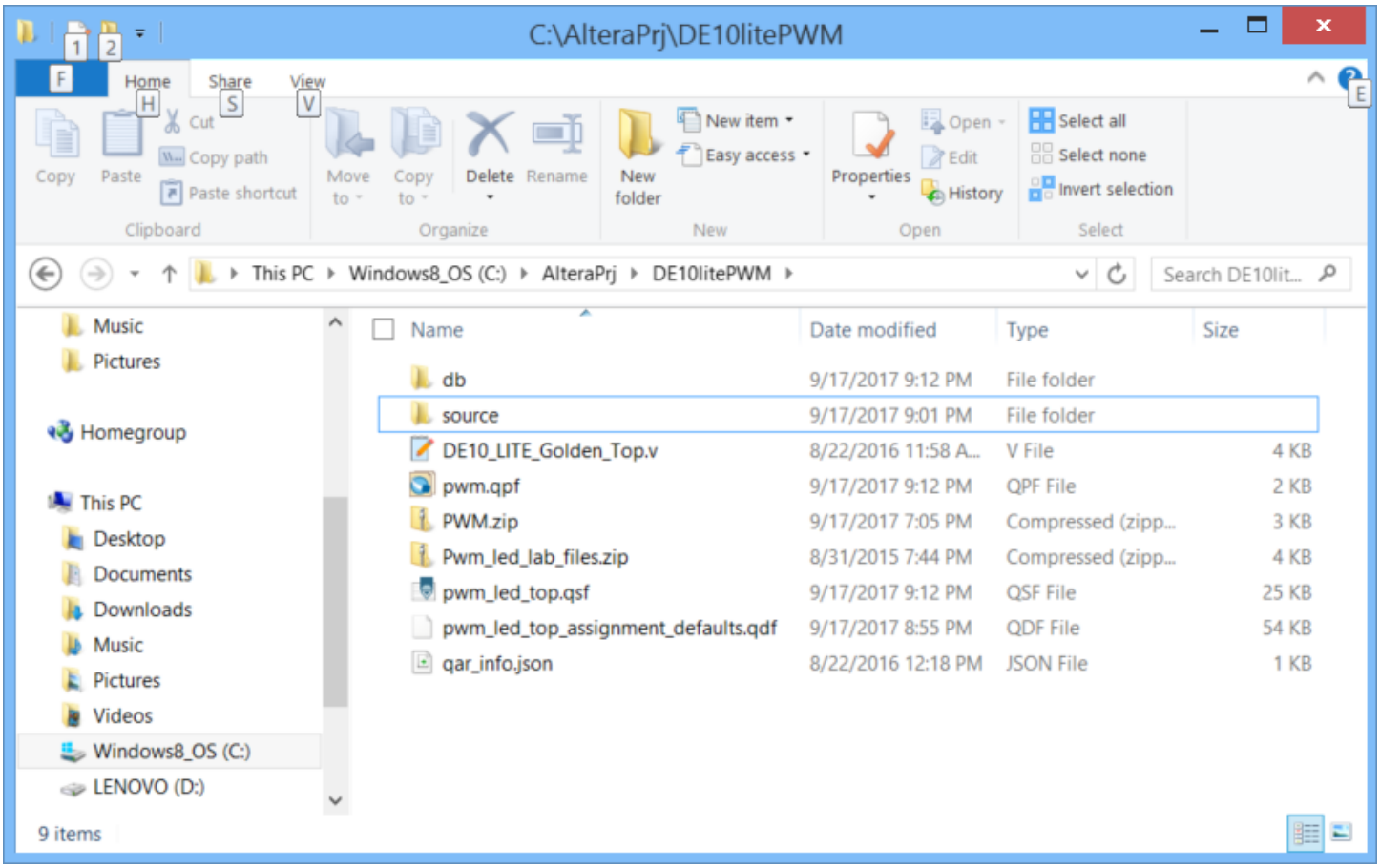


8) Click Next for the summary, and Click Finish.

You should see a Project in Quartus like this:

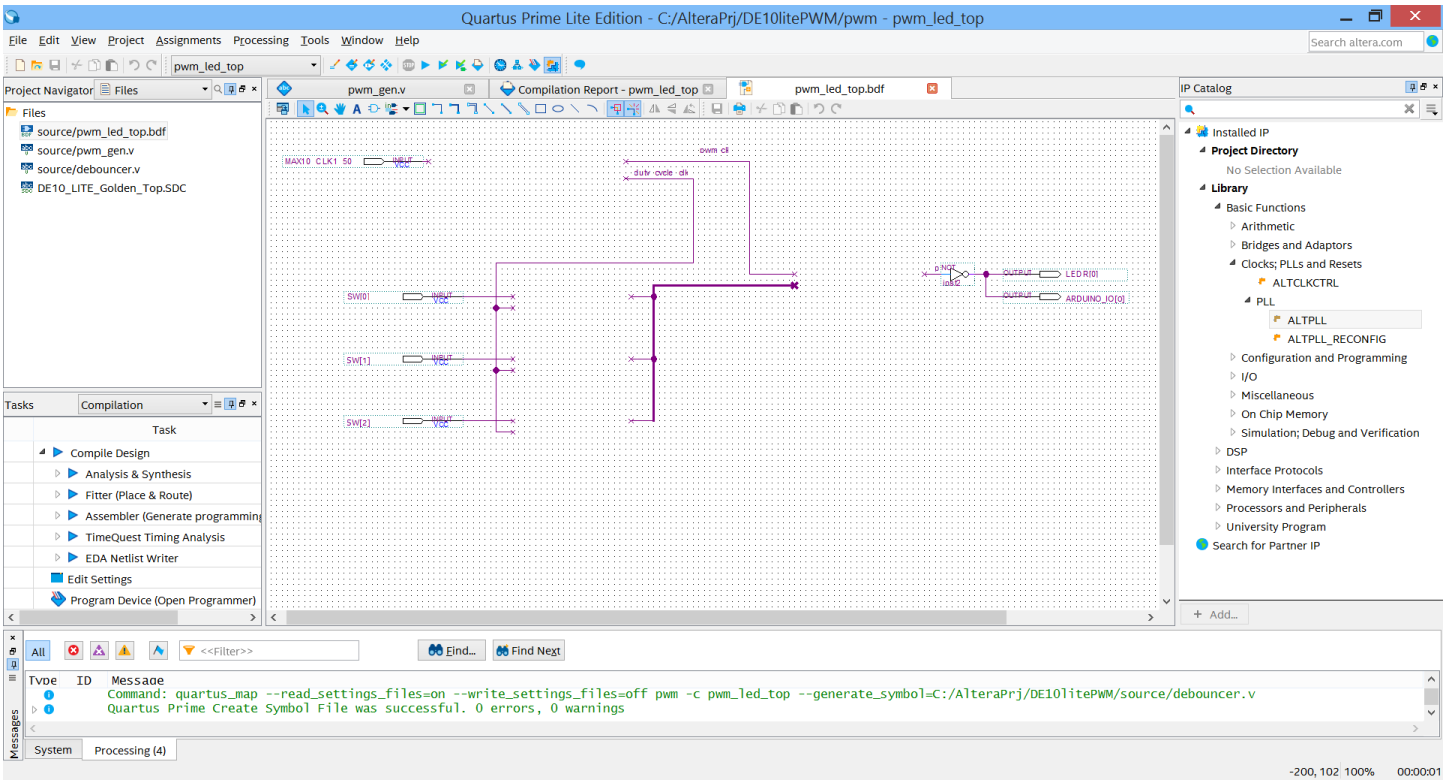


Note that the message indicates the top-level design entity was changed to DE10_LITE_Golden_Top. This isn't what we want. If we look at our project directory, we see several new files have been created by the tool:



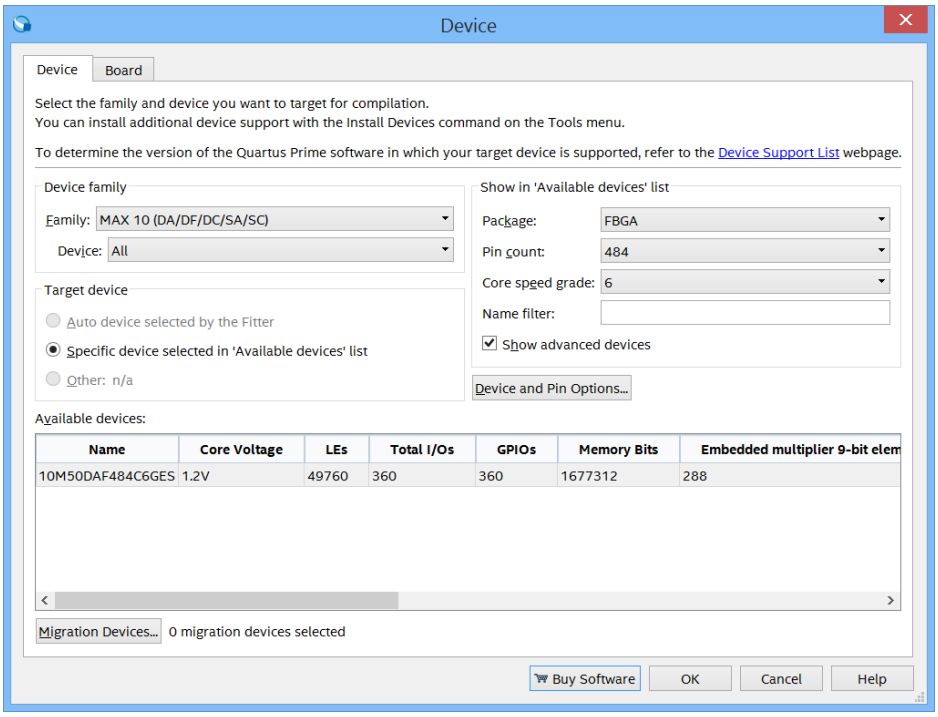
The pwm_led_top.qsf is a Quartus settings file, which includes pin assignments which we need. The .qdf file is a design file that we also want. The pwm.qpf is the project file, it is essential. The DE10_LITE_Golden_Top.v is a Verilog file that defines all the input and output ports. If we were doing the design in Verilog, we could start with this file and it would be very helpful. However, our top-level is going to be a schematic, one that is given.

9) In the Project Navigator, select Files and then right click on the .bdf file set it as the top-level entity. Back under Hierarchy, the file name should now be pwm_led_top. Double click on this to see the beginning of the schematic that you will complete:

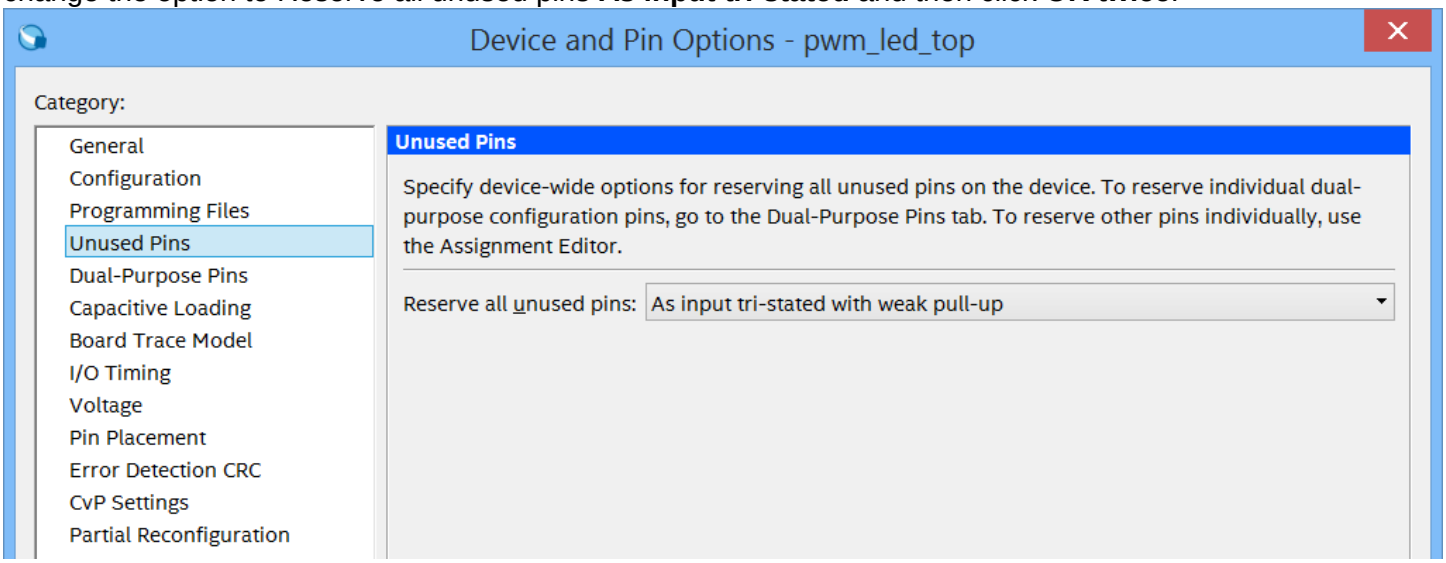


What about the now unused pins? An option exists in the Quartus project to deal with any user I/O pins that are unused by our design.

10) Go to the **Assignments** Menu and choose **Device**, then in the Device dialog box, select **Device and Pin Options…**



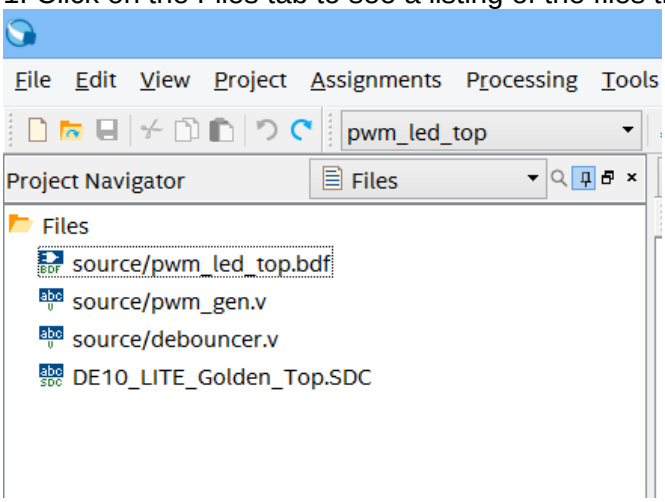
This brings up the Device and Pin Options dialog box. Select **Unused Pins** and use the pulldown menu to change the option to Reserve all unused pins **As input tri-stated** and then click **OK twice**:



Congratulations, you've just completed setup of the Quartus Project.

3.2. Examine the Design Files

In the top left corner of Quartus, there is a **Project Navigator** pane, which has several tabs, including Hierarchy, Files, Design Units, IP Components and Revisions.
1. Click on the Files tab to see a listing of the files that we have added to our project.



Double-click on each of these files to view the contents.
 source/pwm_gen.v – This Verilog file takes a 10-bit duty cycle input and generates a PWM output with 1024 intensity levels (0.1% each).

 source/debouncer.v – This Verilog file debounces button or switch inputs to product a single output pulse when a button or switch on the kit is pressed.

 source/pwm_led_top.bdf – This is a partially completed schematic will be used to connect the various blocks of the design.

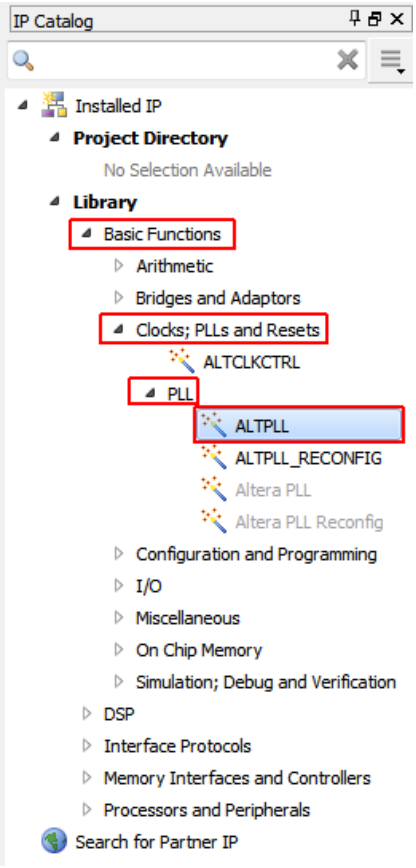
3.3. Create a PLL

In addition to the above files, one additional item we will use for our design is a PLL. The DE10-Lite kit has an onboard 50MHz crystal oscillator which comes into the FPGA on one of the MAX10 FPGA's clock input pins. MAX10 FPGAs include PLLs within the device that can be used to generate other clock frequencies. For our design, we will use a PLL to take the MAX10_CLK1_50 input and generate 10MHz and 30kHz clock outputs.

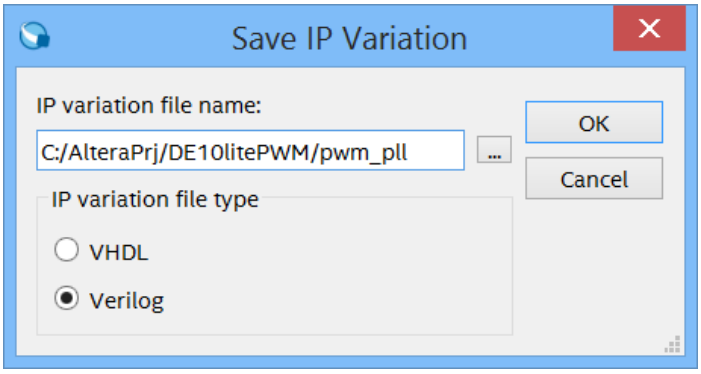
1. From the **Tools** menu, select **IP Catalog.**

2. An IP Catalog pane should now be present on the right side of your Quartus window if it wasn't already.

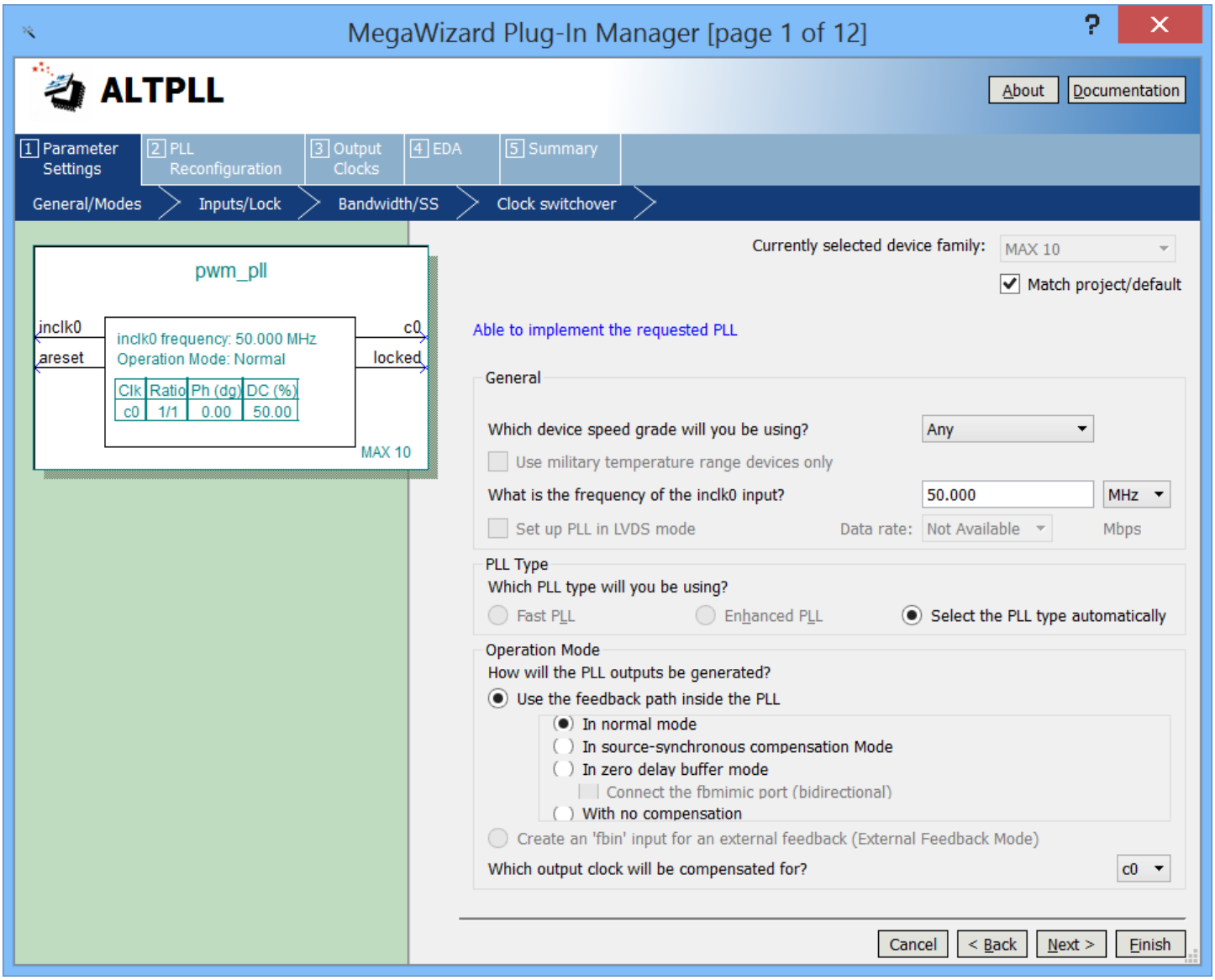
3. Locate the ALTPLL IP Component from the IP Catalog. It is found under **Basic Functions** -> **Clocks: PLLs and Resets** -> **PLL** -> **ALTPLL** and double click to launch the IP Component's wizard.



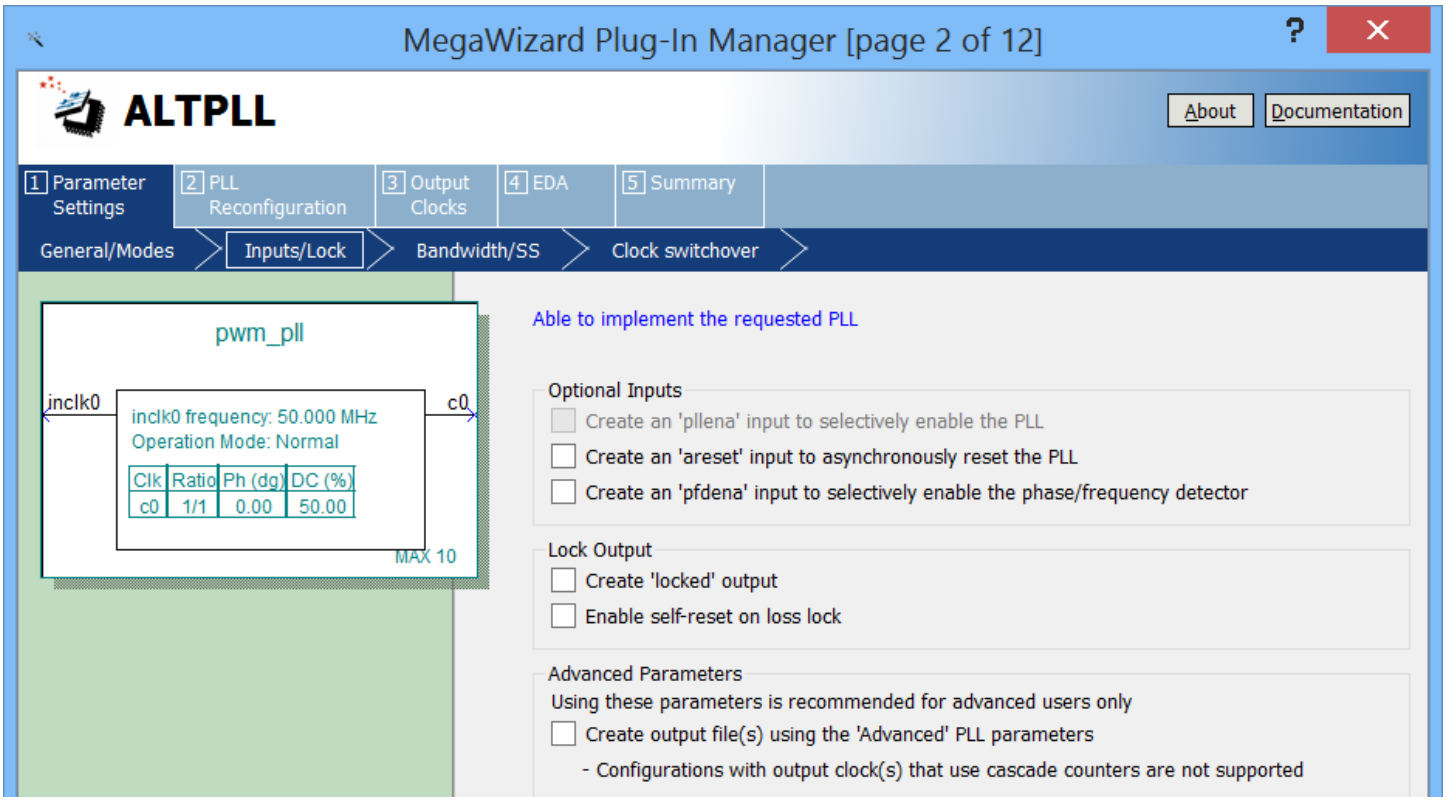
4. A dialog box will appear asking where you wish to save the IP Variation. Save it within the source sub-directory as pwm_pll:



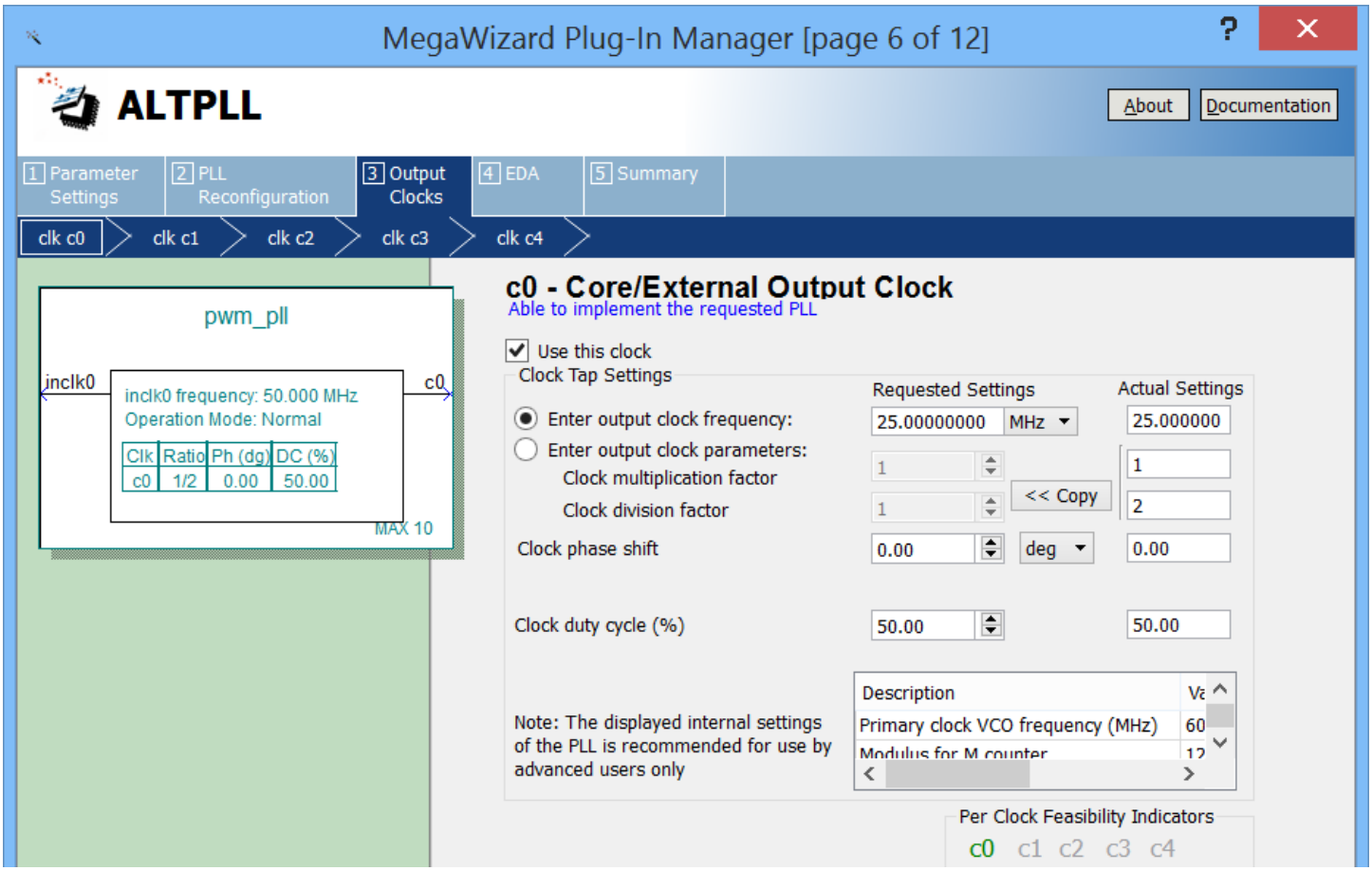
5. The ALPLL MegaWizard will launch. On the first screen of the wizard, change the frequency of the inclk0 input to be 50.000 MHz to match the 50MHz clock input on our kit. Leave the other options on this screen at the default setting.



6. Click **Next** to move to the next page of the MegaWizard "**Inputs/Lock**." Uncheck the options for creating the 'areset' input and the 'locked' output as our design will not need those options.

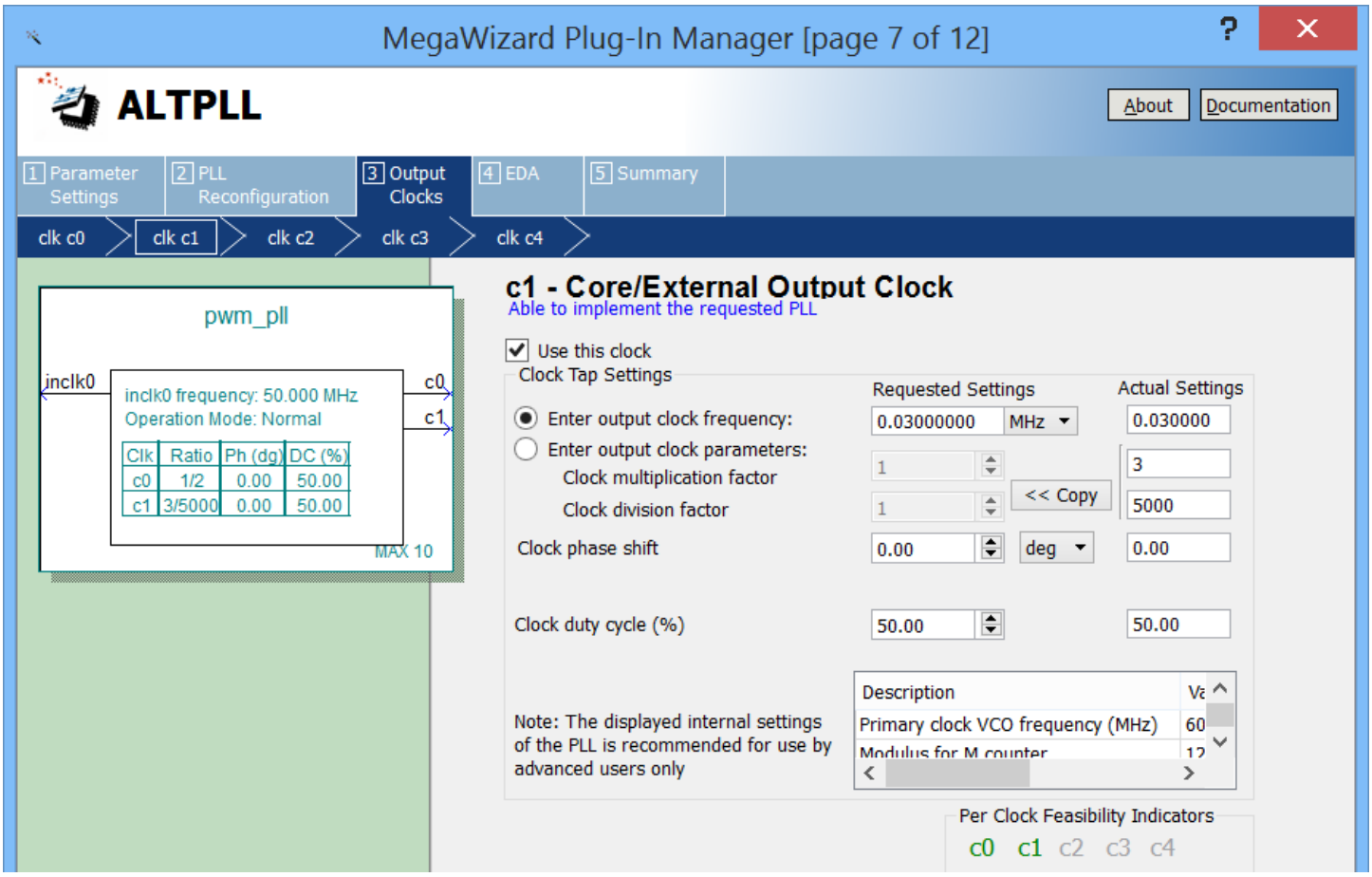


7. Click on the "**[3] Output Clocks**" tab to jump ahead to the MegaWizard page [6 of 12] where we will set our output clock. Change it to 25 MHz.



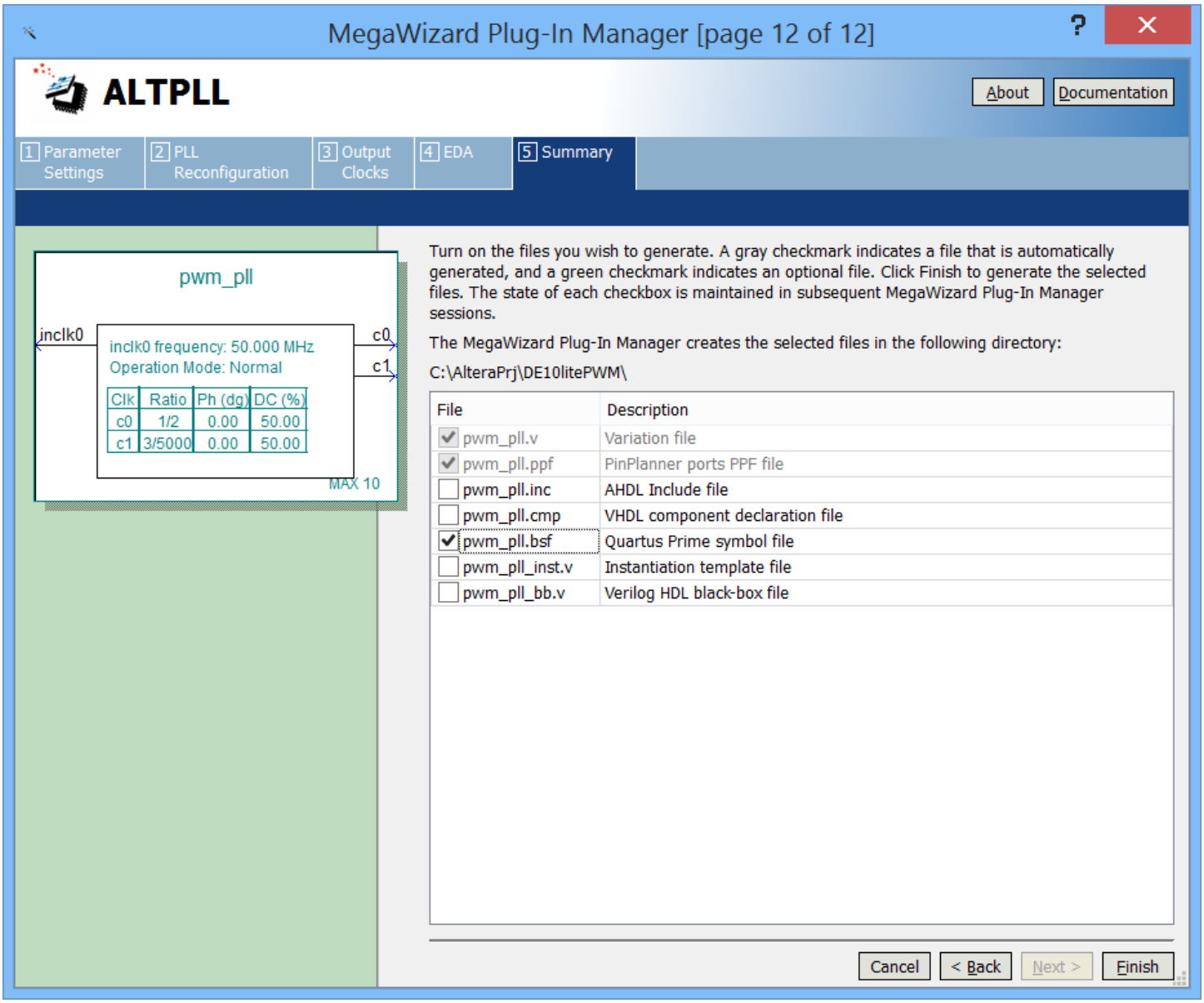
8. Click on the "**clk c1**" tab so that we can enable a 2nd output of the PLL to generate a slower 30kHz clock output.

9. Check the "**use this clock**" box and input an output clock frequency of **0.030 MHz**.

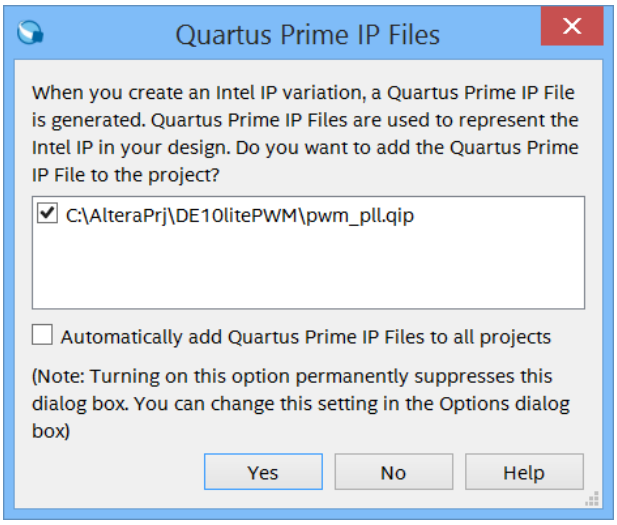


10. Click **Finish** which will take you to the **Summary** tab.

11. On the **Summary** tab, select the **pwm_pll.bsf** to have the Wizard generate a **Quartus symbol file** so that we can place the PLL within a schematic sheet and click **Finish** to complete the PLL



12. Click Yes to add the qip file.

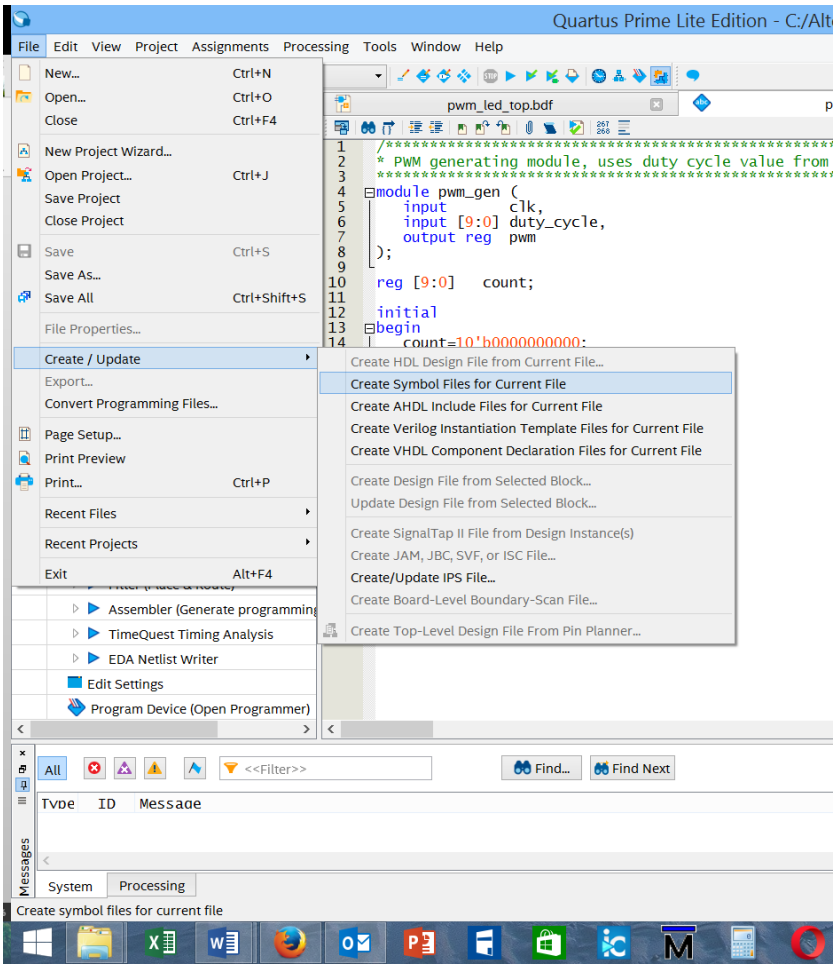


13. The PLL MegaWizard generated an IP variation file which is now located in your source subdirectory. Also, a Quartus IP file with file extension of *.qip has also been automatically added to your Quartus project. You should now see the **source/pwm_pll.qip** file in your Files listing.
The same process of using the IP Catalog could be used in your own FPGA design to create IP blocks such as counters, numerically controlled oscillators, FIR filters, FFTs, and DDR3 controllers, to name a few examples. Explore the IP Catalog to see other items that can be created.

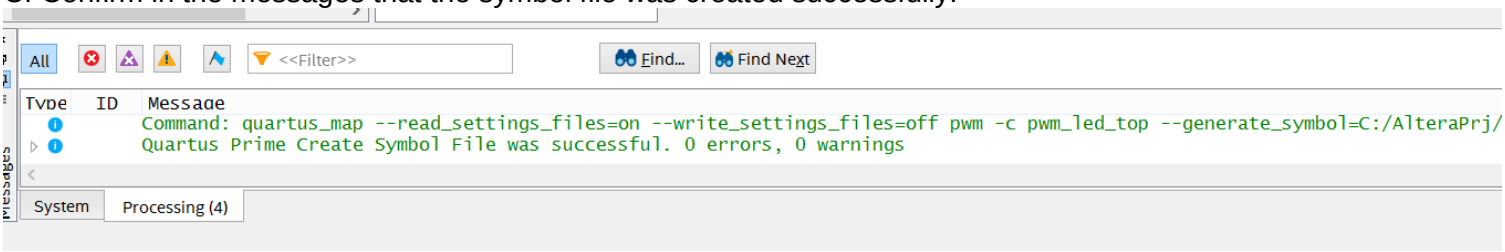
3.4. Create Symbols for the Verilog source files.

The design uses several Verilog files which each define a design entity. The different design entities will need to be connected together to create our FPGA design. While the Verilog design entities can be connected together with Verilog code, another option exists in the Quartus software. It is possible to create symbols for Verilog files and then the design entities can be placed onto a Quartus block diagram file and can be connected together using the Quartus schematic editor.
1. For each of the 3 Verilog files in the project, perform the following:
a. Open the Verilog source file.

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



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

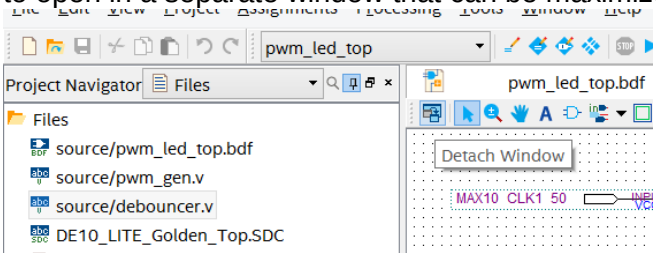


Then repeat these steps to create the symbol for the debouncer.v Verilog file.

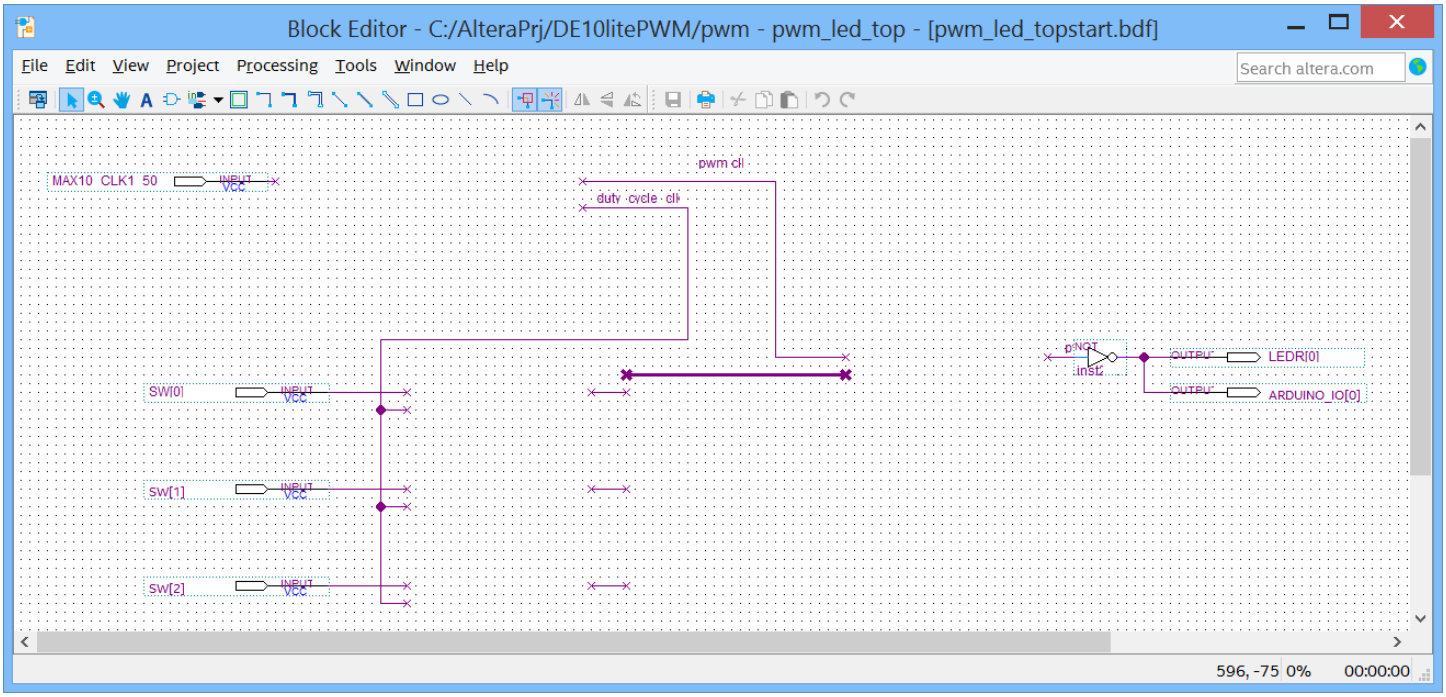
3.5. Complete the Schematic

1. Open the **source/pwm_led_top.bdf** file and the partially completed schematic should appear in the Quartus window.

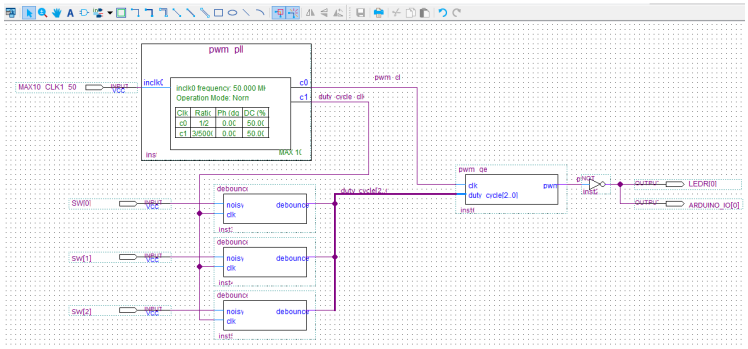
2. You may find it useful to right click on the schematic tab and select "**Detach Window**" to allow the schematic to open in a separate window that can be maximized to see the entire schematic.



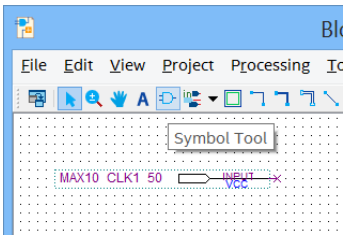
3. The partially-completed schematic should appear as follows:



4. Our goal is to insert the symbols for the various blocks in the design to create a completed schematic which will look as follows:

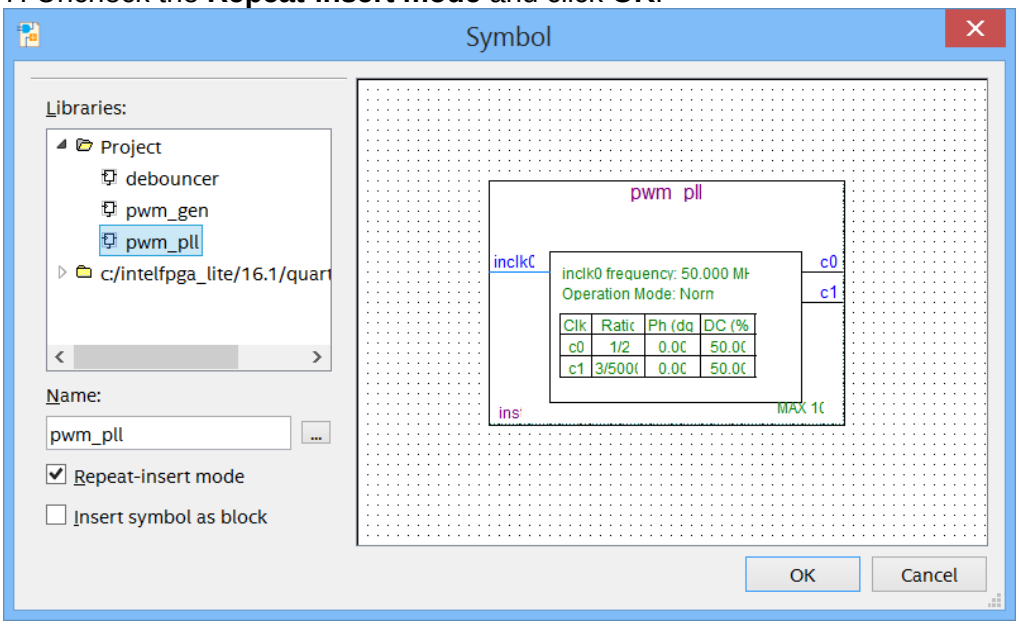


5. Click on the **Symbol Tool**

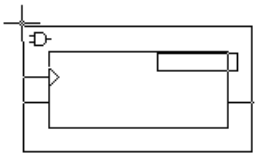


6. Under the **Project** library, you will find the symbols that we have generated previously in this project. Select the **pwm_pll** symbol.

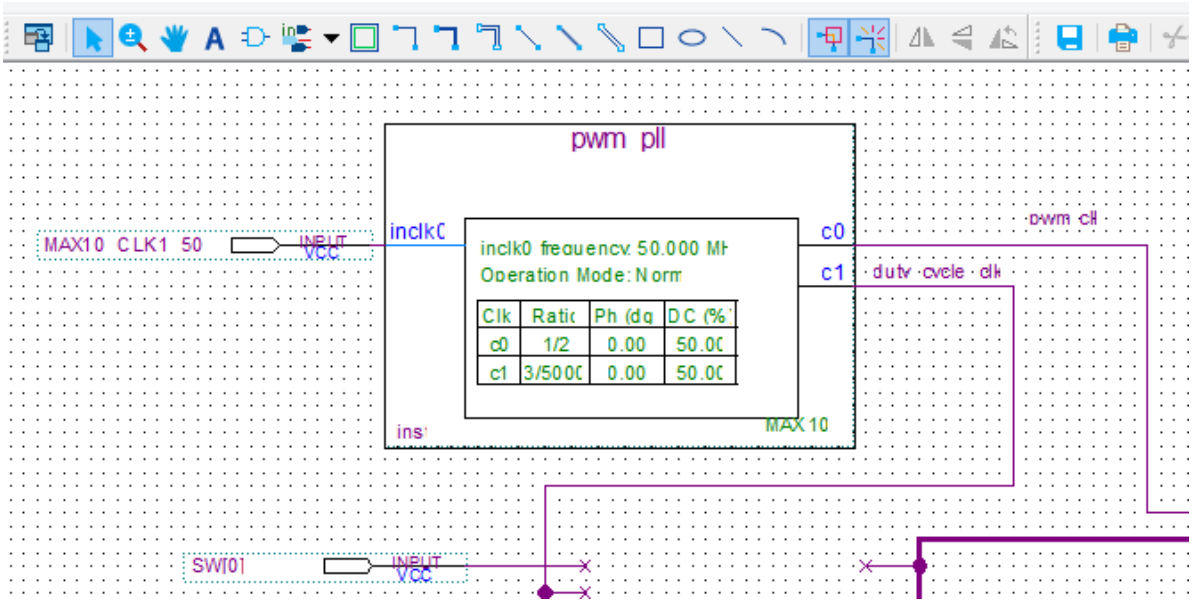
7. Uncheck the **Repeat-insert mode** and click **OK**.



8. You will now have a floating symbol that can be placed onto the schematic sheet.

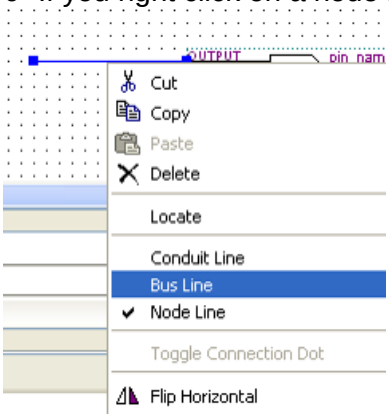


9. Place it in the correct location at the top left such that the **SYS_CLK** input pin lines up with the **inclk0** port and the **c0** and **c1** ports line up with the **pwm_clk** and **duty_cycle_clk** wires as follows. (You may need to scroll up a little to create room for the symbol.)



Below are a few tips to related to schematic entry:

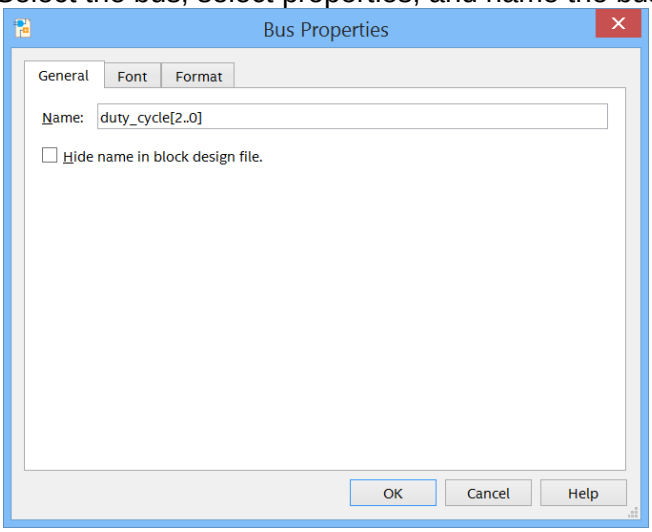
- Note that Quartus schematic sheets use square brackets `[]` and two periods `..` to denote bus notation. E.g. `duty_cycle[3..0]`
- In the symbol selection dialog box, if you know the name of the symbol that you need (e.g. “input”, “not”, “and2”, “and3”, etc.) you can type it into the “Name” field, rather than to navigate the library tree to find it.
- You can find the orthogonal node line and orthogonal bus line icons on the toolbar:
- Alternatively, when you hover over a node on a symbol, the cursor will change to the node line drawing icon and you can directly begin drawing even if you do not have the node line tool selected.
- If you right click on a node line, you can change it to a bus line from the context menu:



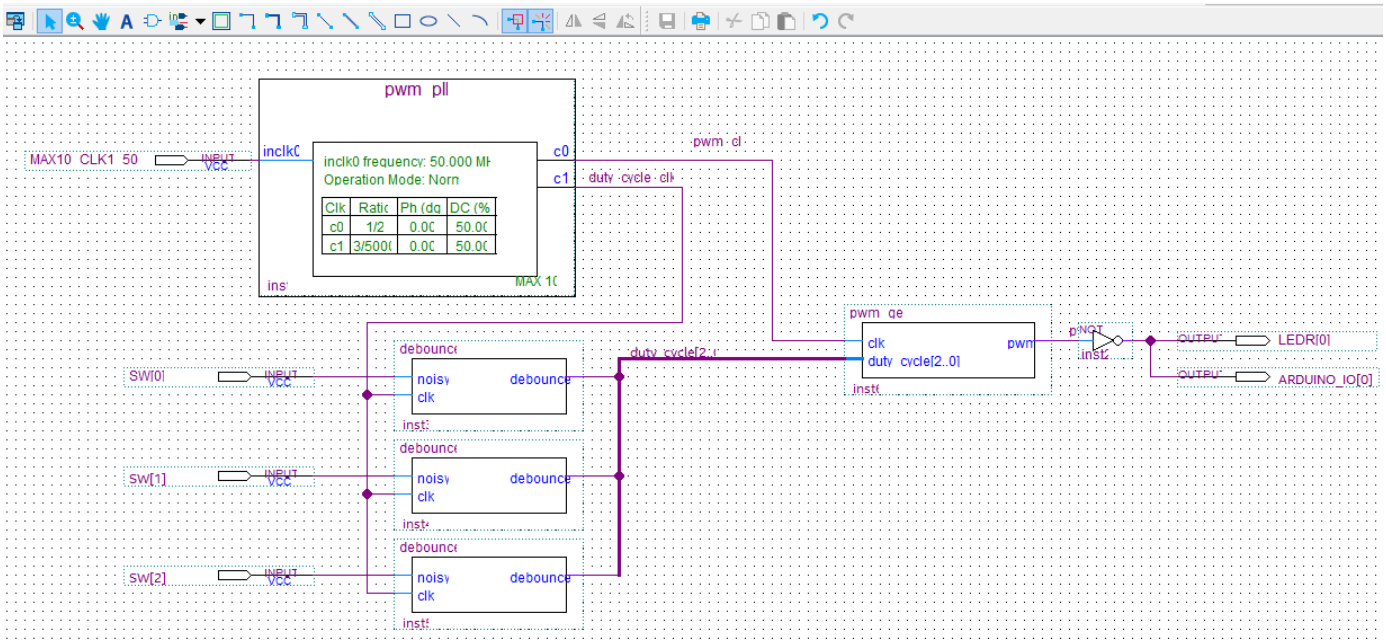
10. Repeat in the same manner to add 3 debouncer symbols and the pwm_gen symbol to the schematic sheet. It is helpful to check the repeat block mode when adding the debouncers. It is easier to place the debouncers and then line the switch inputs up to them. When tying the debouncer outputs to the duty cycle input, start from each debouncer output so that you can label it with the duty_cycle bus name. For the top line out of the debouncer, right click on the line and select properties. Enter duty_cycle[0] for the name. Do the

same for the next 2 lines, duty_cycle[1] and duty_cycle[2]. Extend the bus form the pwm_gen block to the right of the 3 debouncer blocks, and tie each debouncer output to the bus.

Select the bus, select properties, and name the bus duty_cycle[2..0].



The schematic should now appear as follows:



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

Congratulations, you've completed the design entry of the PWM circuit.

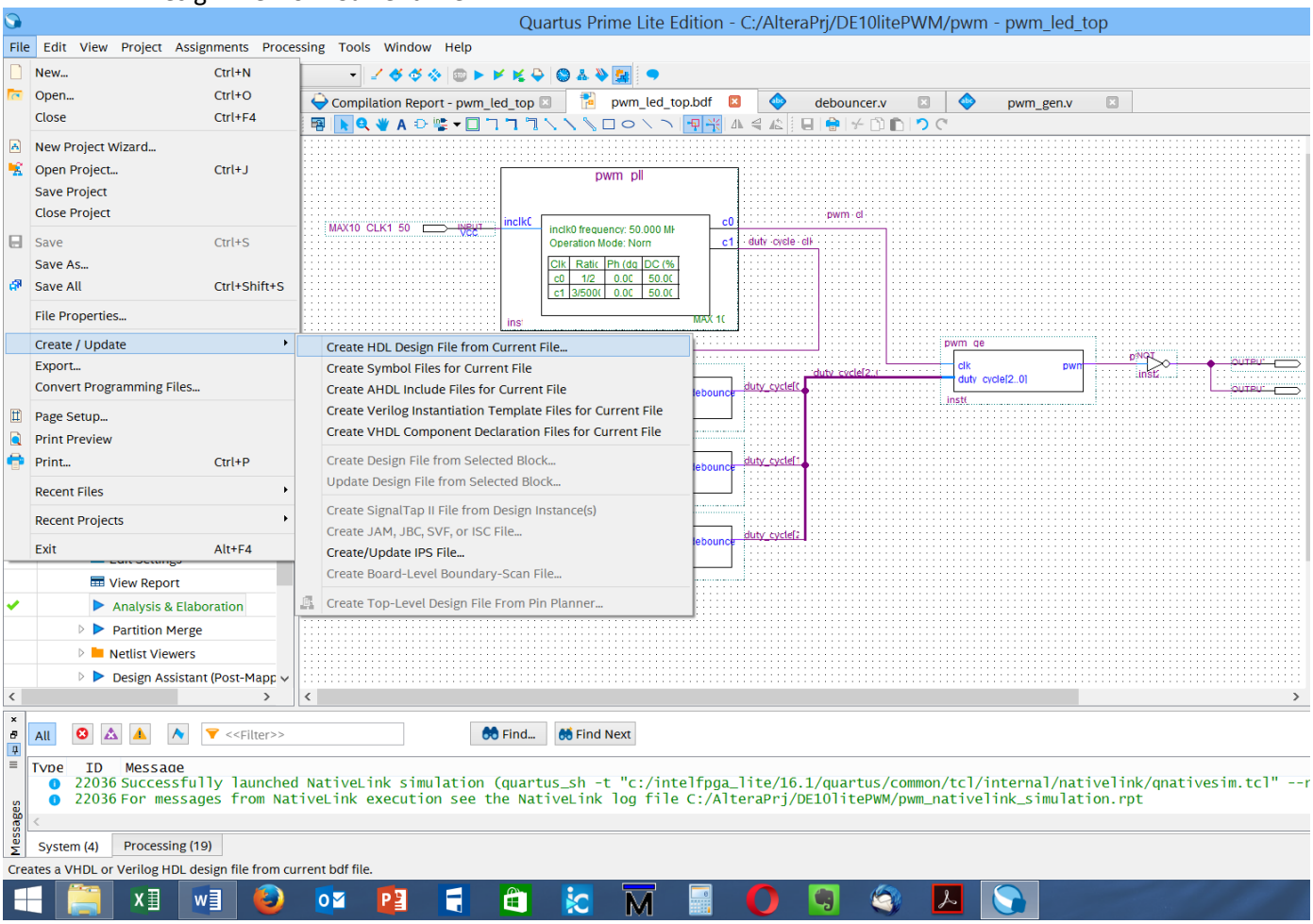
4. Simulating the Design

Section Objective

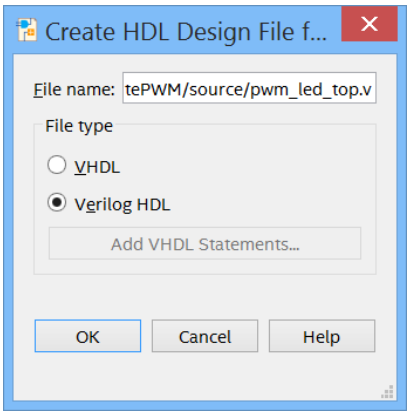
In this section simulate the design using ModelSim, perhaps the most common simulation interface used in FPGA development. The PWM has a clock, which can be used to drive the simulation so no testbench is required, although we will force the inputs.

4.1. Prepare Simulation Files and Settings

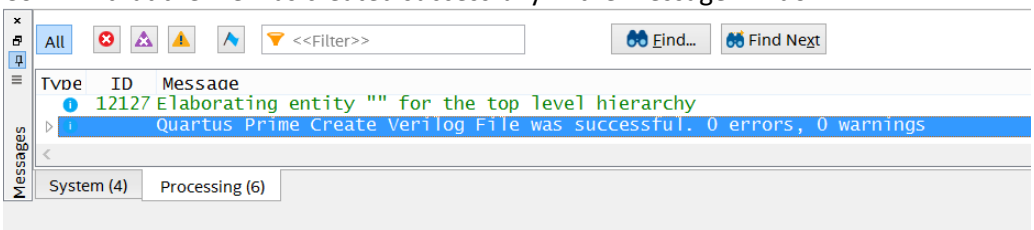
1. The simulator cannot simulate a .bdf file. To provide a top-level file that can be simulated, we need to convert the .bdf to a .v file. With the pwm_led_top.bdf tab highlighted, select the File menu, Create/Update, and Create HDL Design File from Current File.



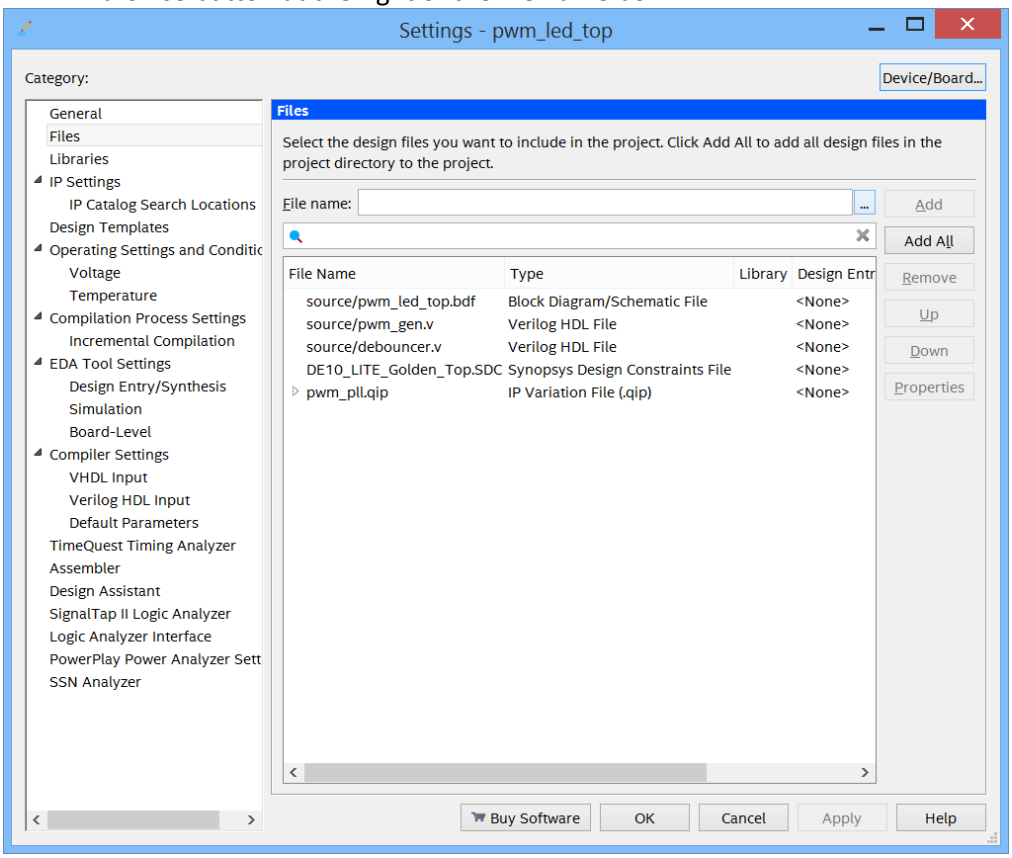
2. Choose Verilog HDL and select OK:



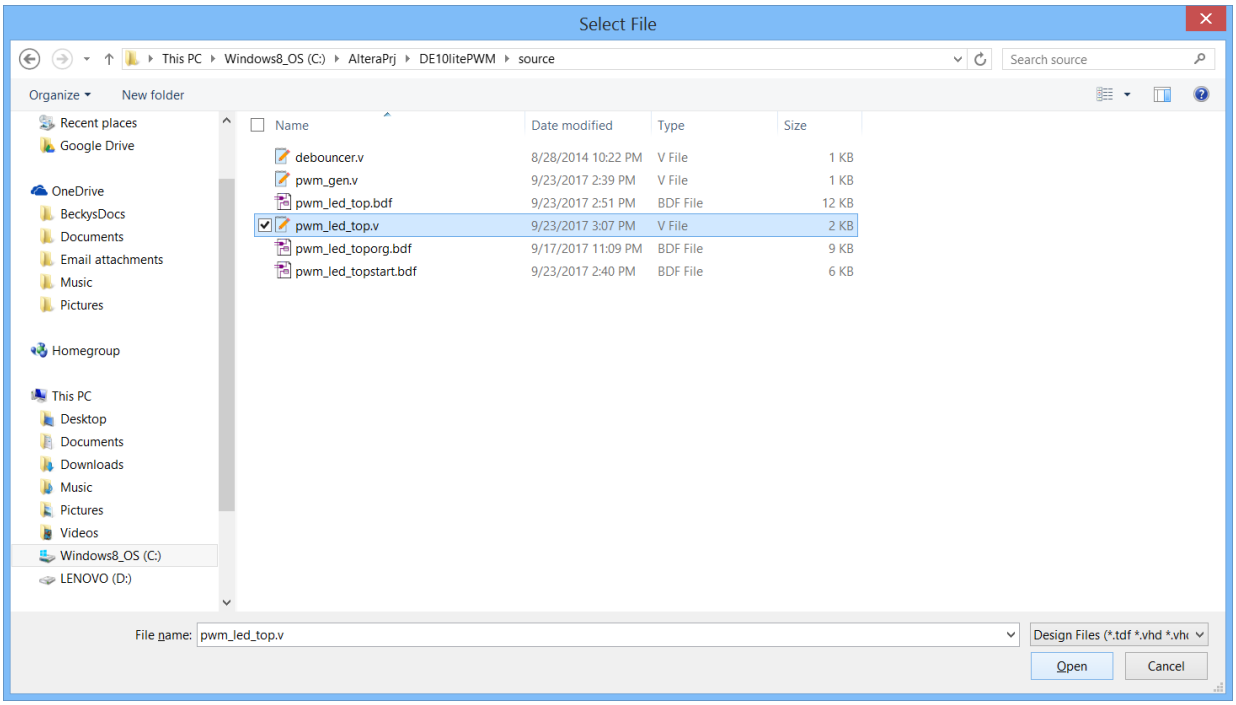
Confirm that the file was created successfully in the message window:



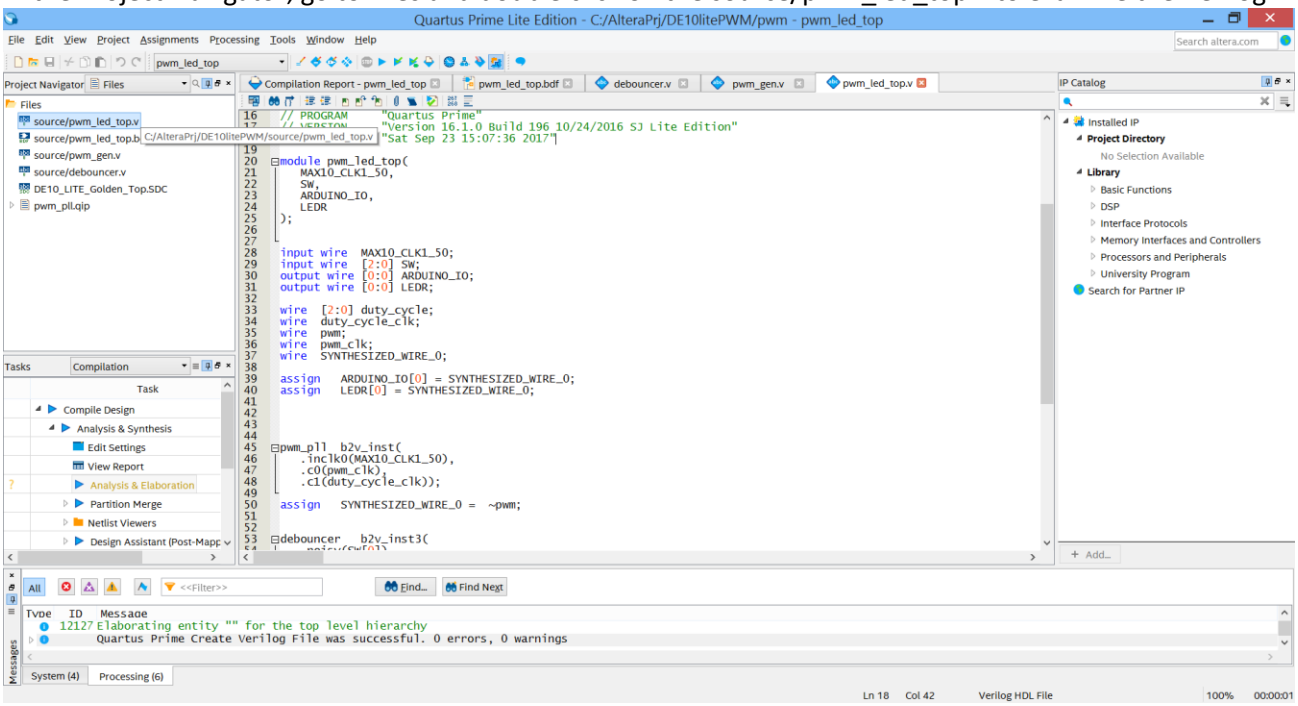
3. From the Project menu, select Add/Remove files from project. The Settings dialog should appear. Select the browse button at the right of the file name box.



4. In the source directory, select pwm_led_top.v and click Open.



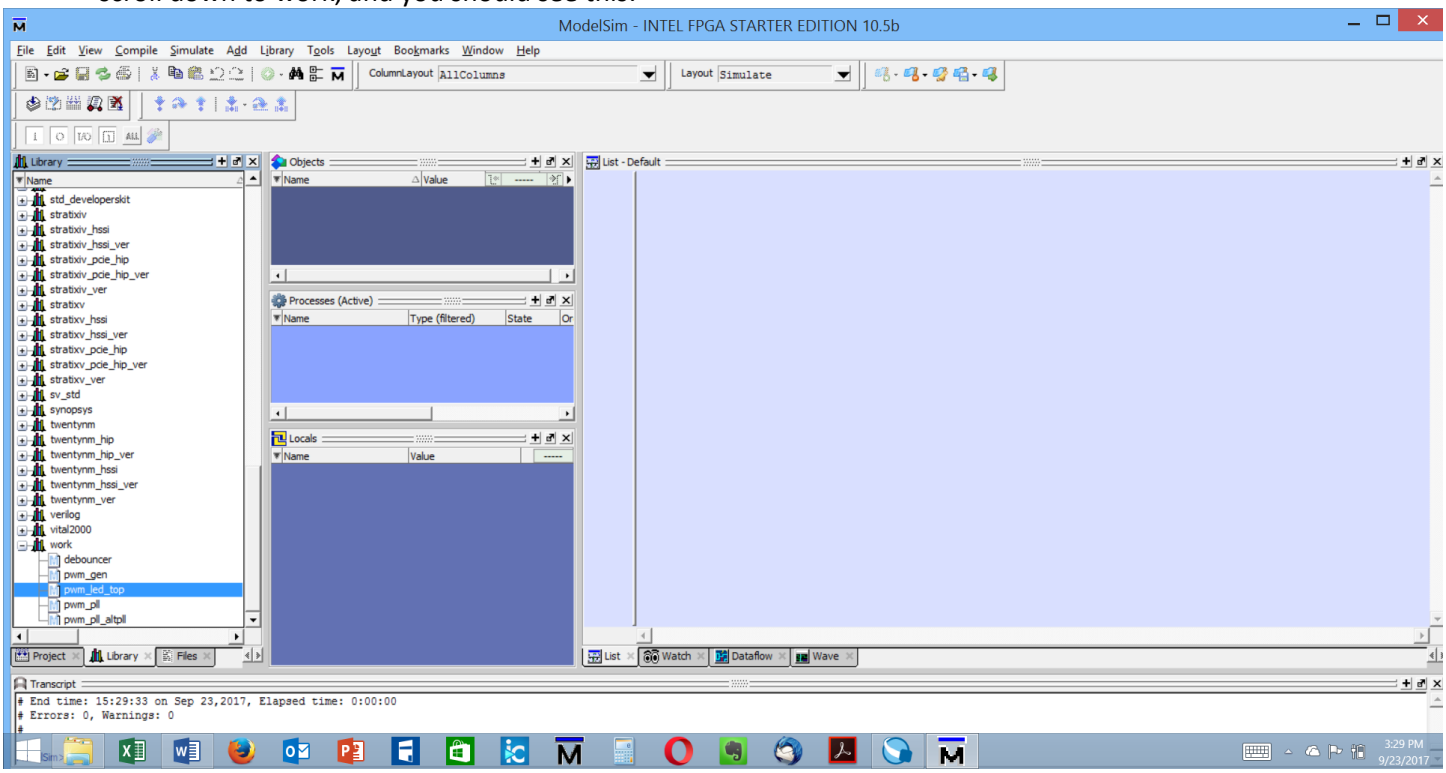
5. In the Project Navigator, go to Files and double click on the source/pwm_led_top.v to examine the Verilog file.



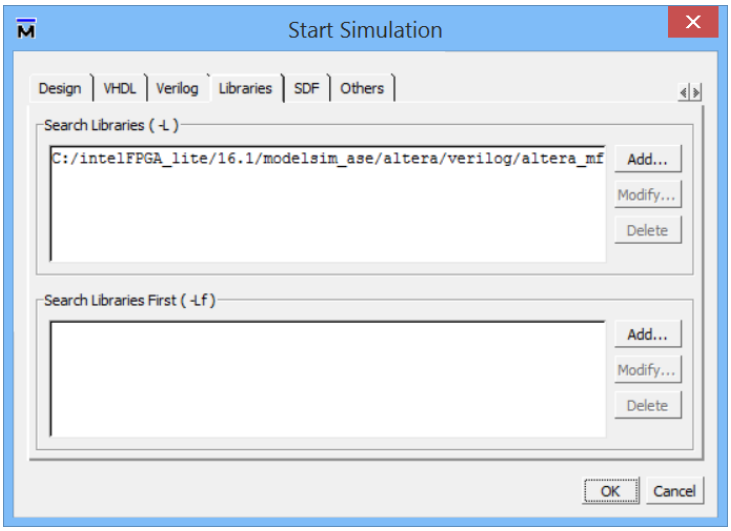
6. Right click on source/pwm_led_top.v and select Set as Top-Level Entity.
7. Right click on source/pwm_led_top.bdf and select Remove file from project. Run Analysis & Elaboration.

4.2. Simulate the Design

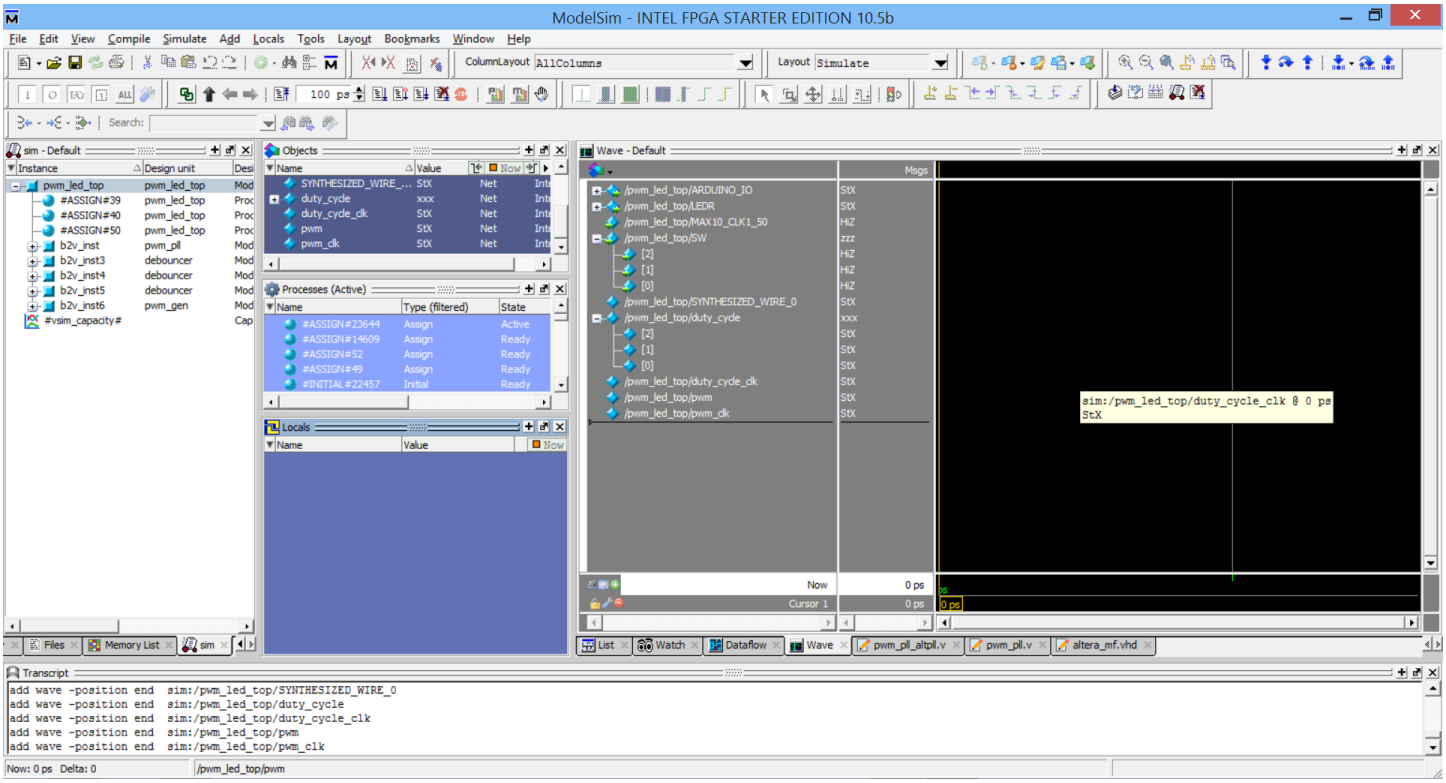
8. From Quartus, select Tools and then Run Simulation Tool, and then RTL Simulation. If installed correctly and chosen as the EDA tool in settings, then ModelSim-Altera Edition should start. Click on the Library Tab, and scroll down to work, and you should see this:



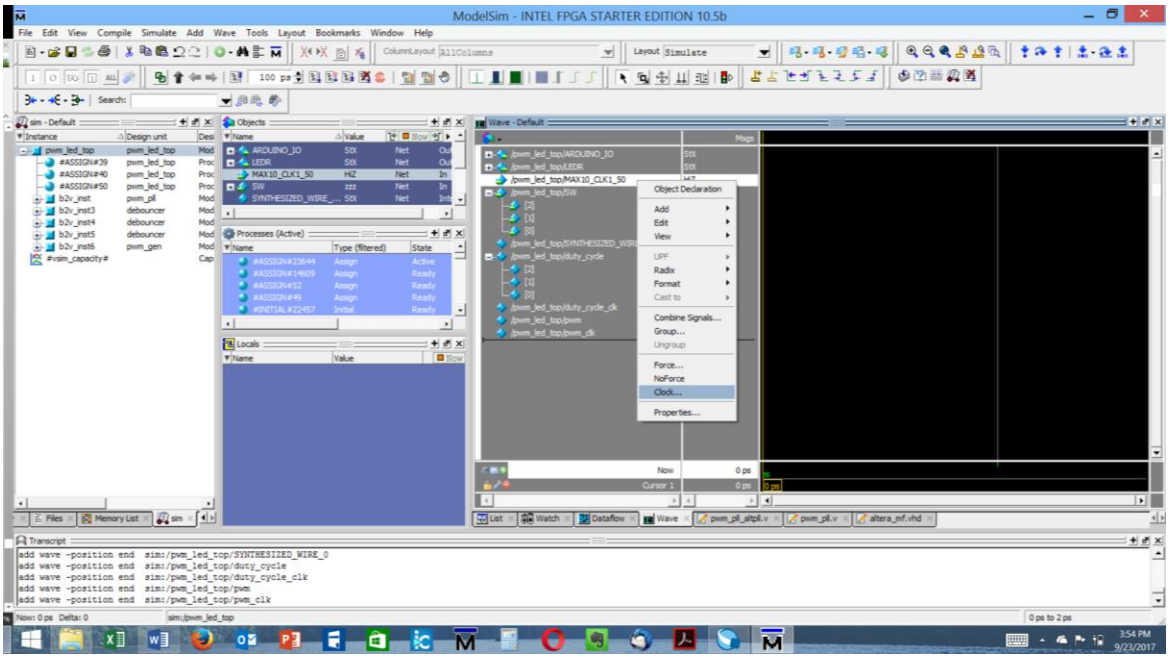
9. From the top menus, select Simulate. In the Start Simulation Dialog box, scroll down to work and select pwm_led_top.v as the design unit. Be sure to change the resolution from default to ns. On the Libraries tab, select Add, and then navigate to C:/intelFPGA_lite/16.1/modelsim_ase/altera/Verilog/latera_mf to add the library that contains the alt_pll.
   Click OK and simulation should begin.



10. Select the Wave Tab. Click into the Objects window, hit control-A and then drag all the signals to the wave window:

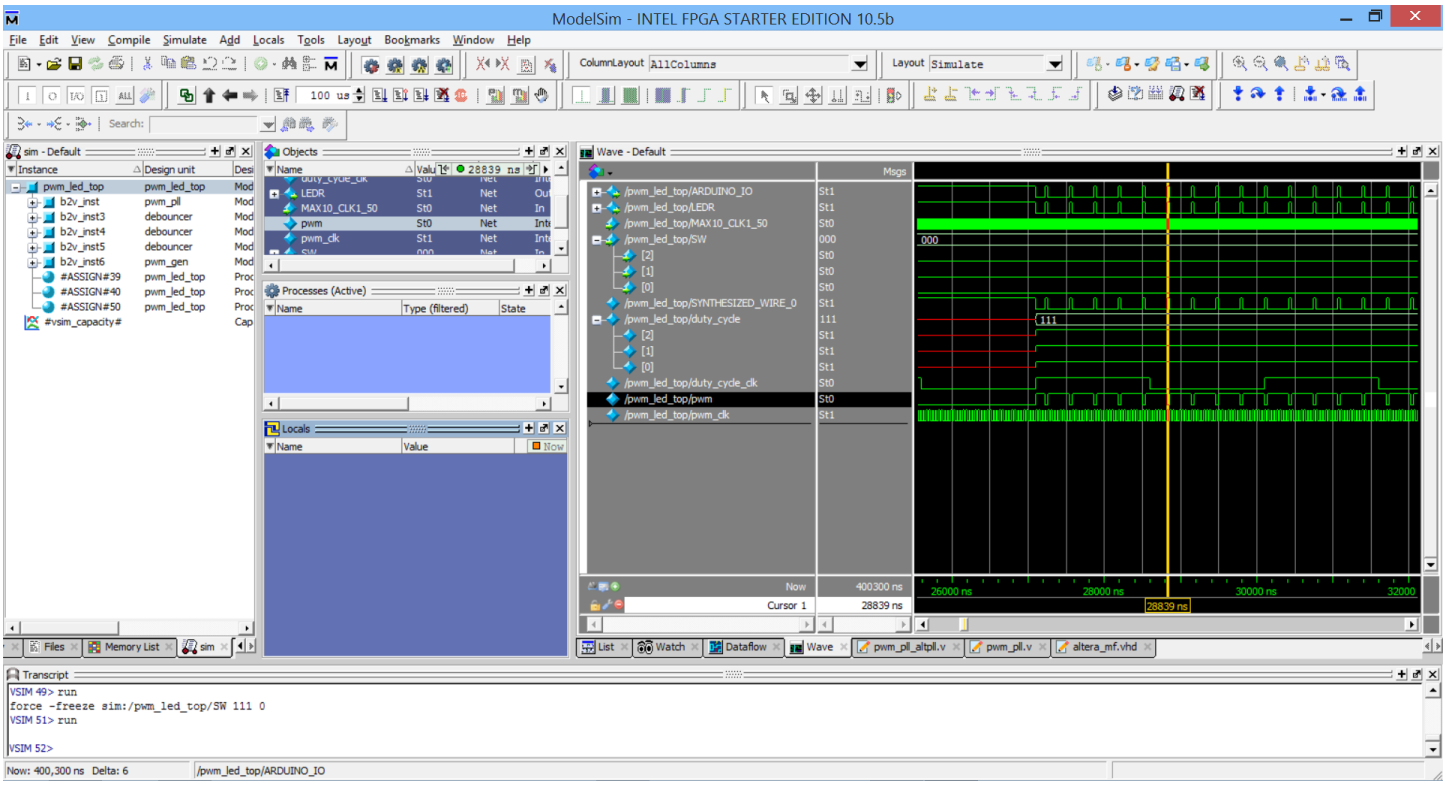


11. Right click on the MAX10_CLK1_50 and select clock.



12. Set the clock period to 20 and select OK. Repeat with the duty cycle clock, but set the clock period to 3000.
13. Change the run length time to 100 ns.

14. Click on the down arrow doc to run, or select from the top menu Simulate – Run – Run 100. You should see a clock waveform on the MAX10 clock.
15. Right click on the pwm_led_top/SW signal, select Force, and set the signals to 000.
16. Run 100 again.
17. It takes some time for the debouncer clock to allow the switch inputs through. Change the simulation run length to 100 us, and Run 100.
18. You should see the duty cycle go to 111, and the pwm signal will begin. You can force the SW[] inputs to other values, and verify that the pwm output is as expected.



19. Close the ModelSim simulator (**File > Quit**). Click **Yes** when asked if you want to quit.

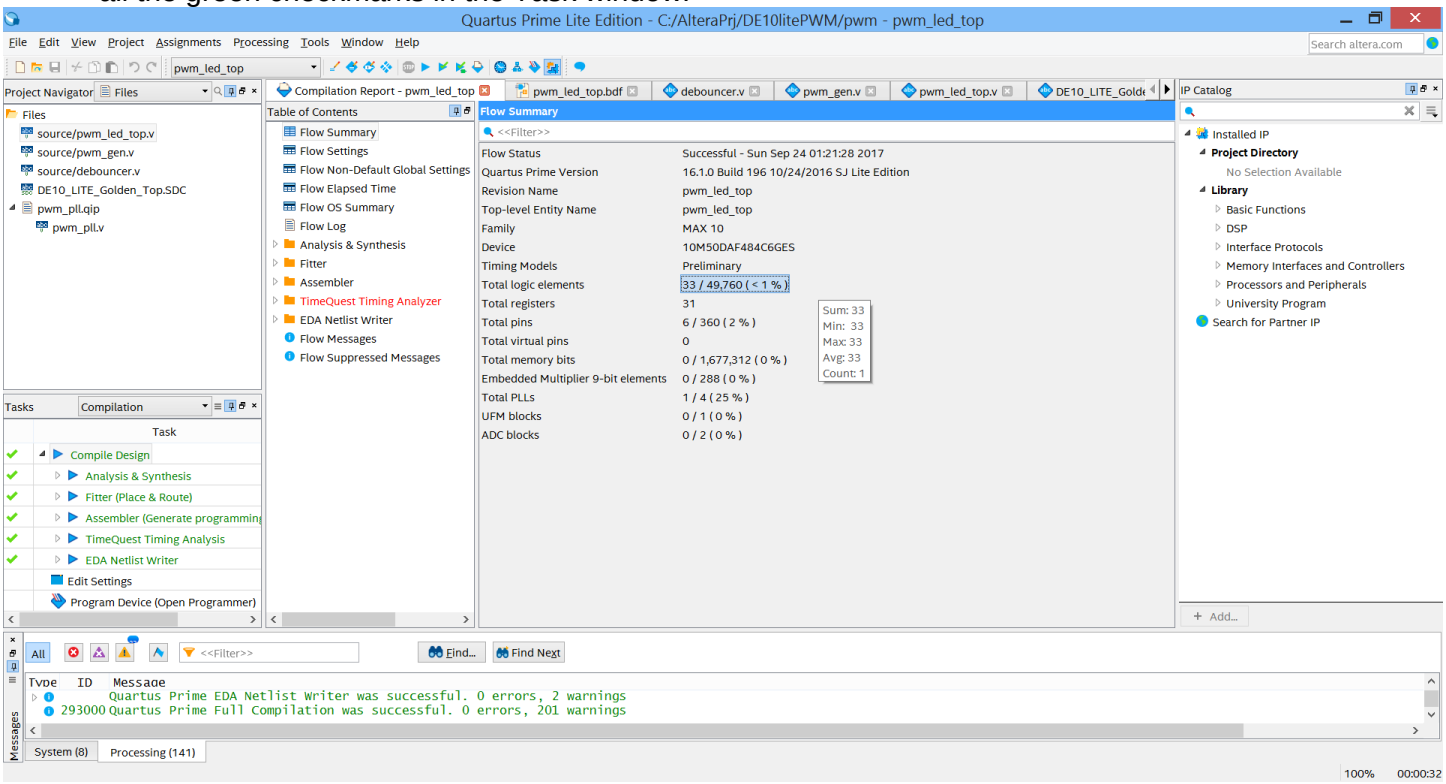
CONGRATULATIONS!!

You have just simulated your FPGA system!

5. Placing and Routing the Design

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

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



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

CONGRATULATIONS!!

You have just placed and routed your FPGA design.

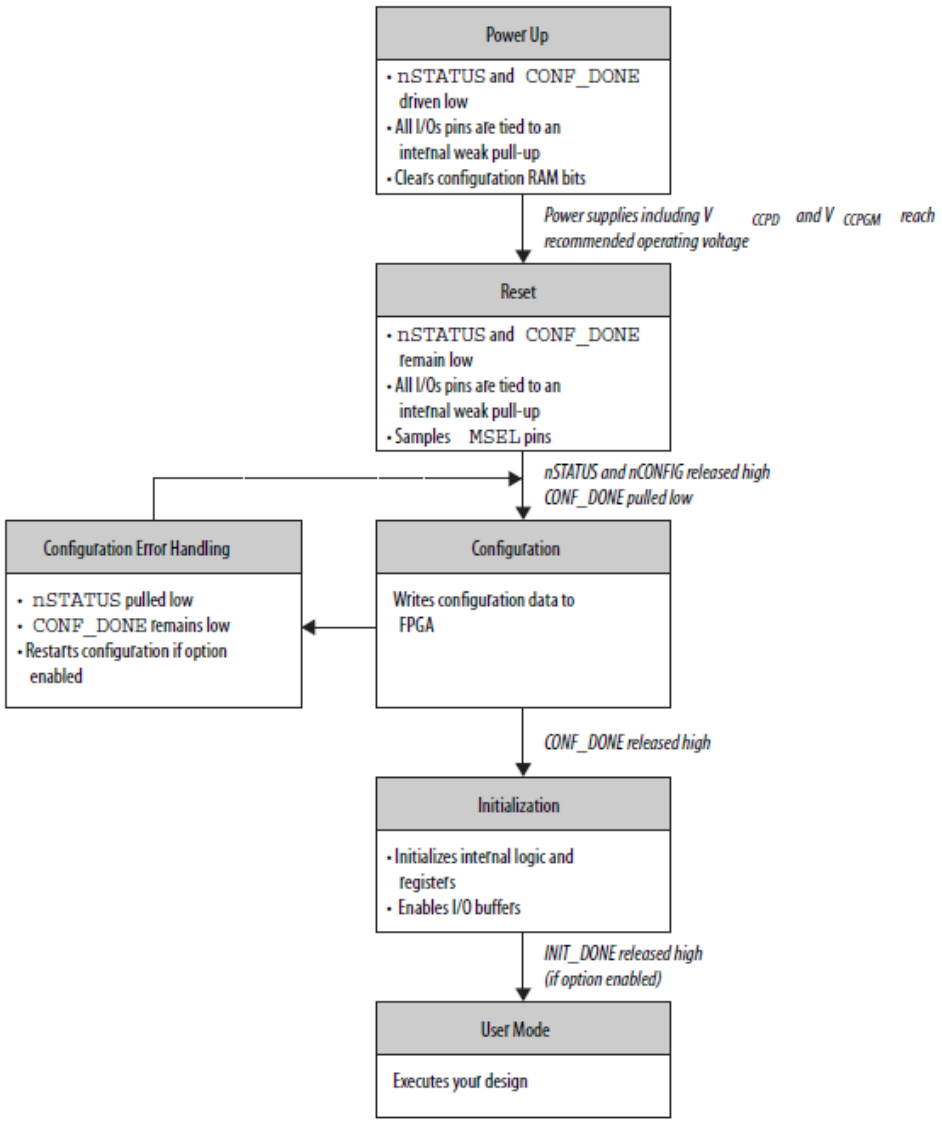
6. Creating a Programming File

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

6.1. Introduction to Altera Programming

Now that you've done all this wonderful logic design work in the FPGA tool, you would naturally like to see it in action in hardware. To do this, you need to program, or configure, the FPGA. Programming usually describes the action of placing into memory contents of a file that will be used to define the behavior of the device, whether it be a microcontroller or a programmable logic device. Configuration is the process of loading that memory into the device to establish the characteristic behavior of the device. Once configuration of an FPGA is complete, it is ready to perform the hardware tasks it was designed for. The configuration Sequence for Altera Cyclone V devices is shown in more detail in the sequence chart here:

Figure 7-1: Configuration Sequence for Cyclone V Devices

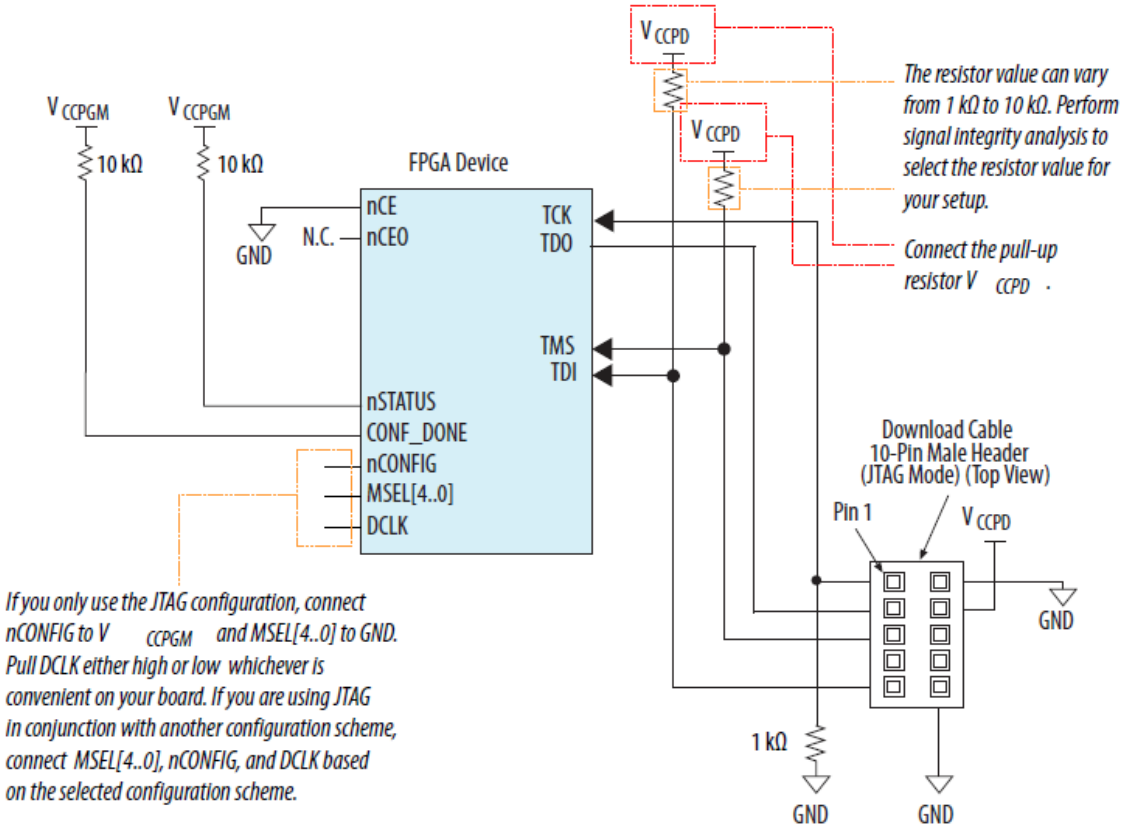


For Altera FPGAs, the configuration memory is typically loaded either directly through the JTAG programming cable or via a memory on the board connected to the FPGA. In fact there are several configuration modes for Cyclone V devices, as seen in the following table:

Configuration Mode	Data width	Max Clock Rate (MHz)	Decompression	Design Security	Partial Reconfiguration	Remote System Update
JTAG	1	33	-	-	-	-
Active Serial (AS) through EPCS or EPCQ serial FLASH memory configuration device	1 or 4	100	Yes	Yes	-	Yes
Passive Serial (PS) through CPLD or External Microcontroller	1	125	Yes	Yes	-	-
Fast Passive Parallel	8	125	Yes	Yes	-	-
Fast Passive Parallel	16	125	Yes	Yes	Yes	Parallel Loader
Configuration Via Protocol (CVP) via PCIe	x1, x2, and x4 lanes	-	Yes	Yes	Yes	

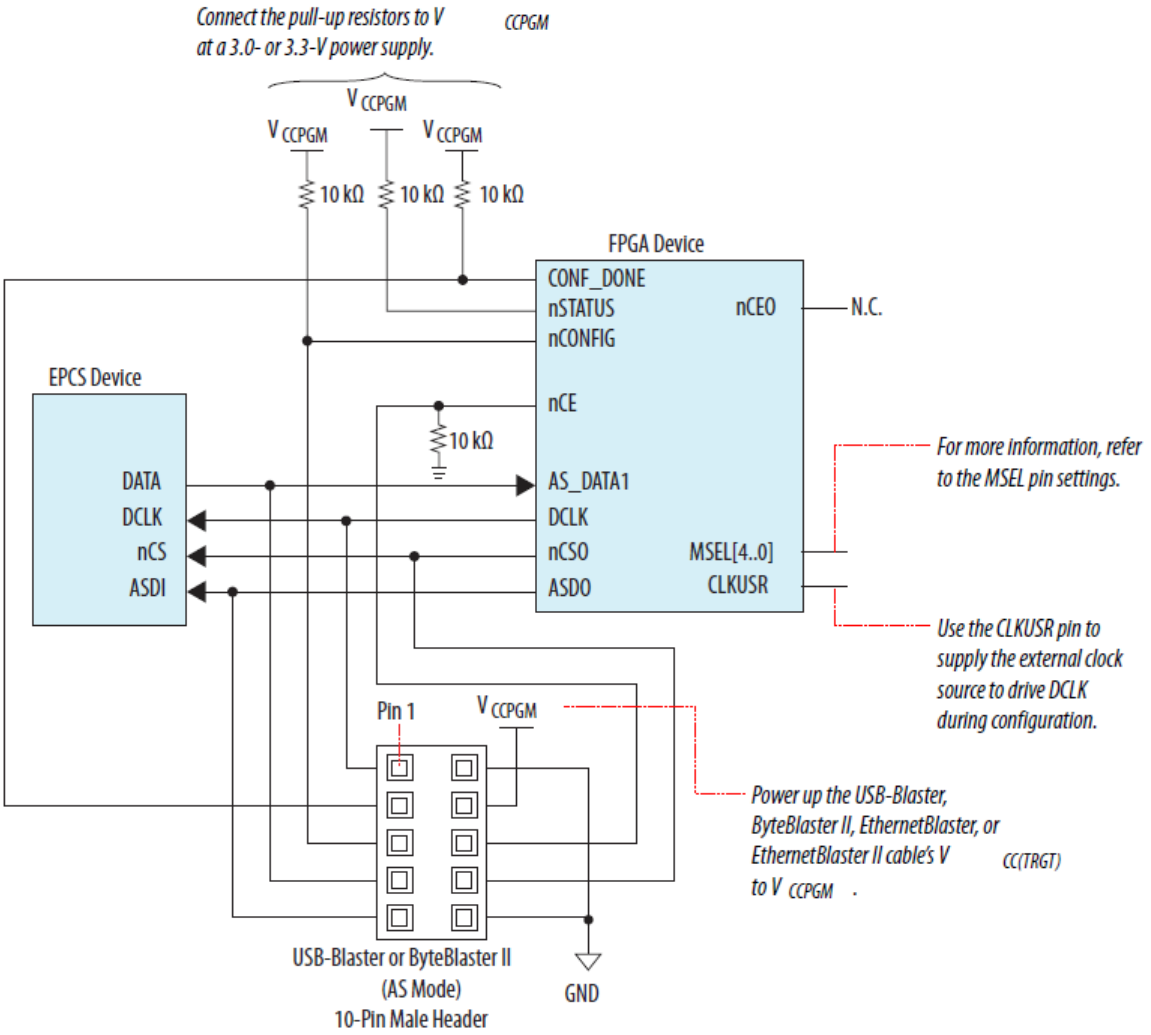
The hardware design of each of these configuration modes is different. The partial schematic shown here depicts the hardware design for a JTAG configuration interface, which is commonly used on evaluation boards. JTAG configuration can be used in conjunction with other configuration modes, but it always has precedence. In JTAG configuration, JTAG commands and data are sent to the FPGA directly to program the internal configuration memory, and once the transfer is complete the FPGA is configured and ready to run.

Figure 7-21: JTAG Configuration of a Single Device Using a Download Cable



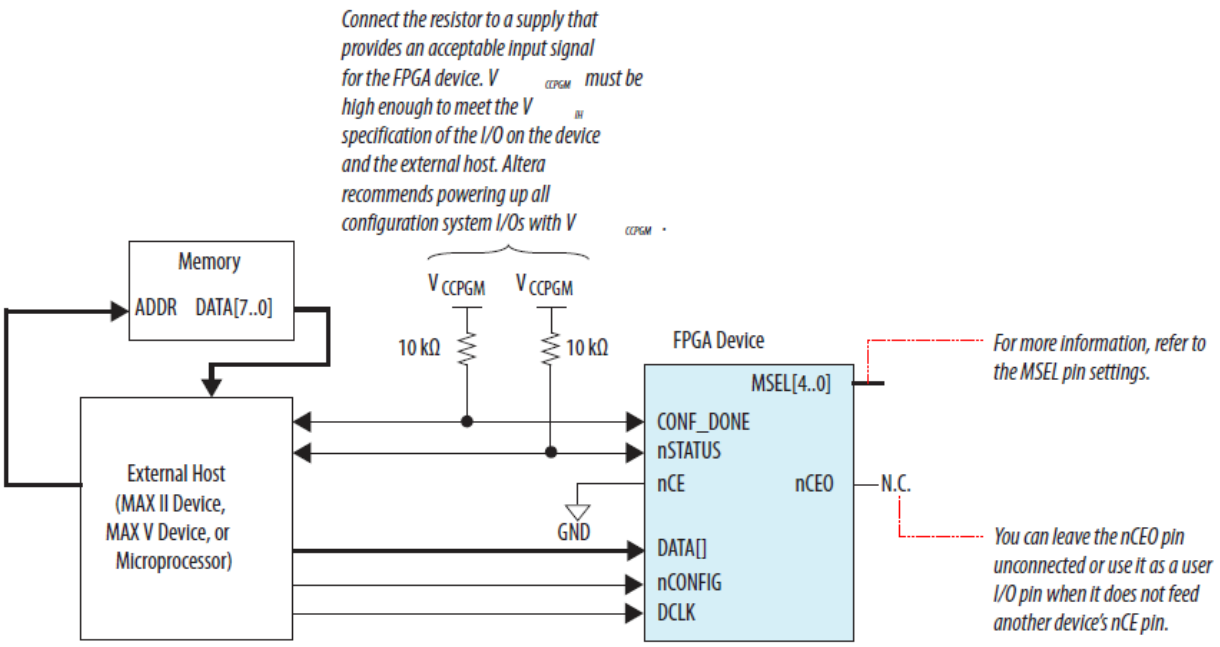
The next most common configuration mode is Active Serial, in which a serial FLASH memory device stores the configuration image, and the FPGA loads it into configuration memory on power up. A schematic for this mode is shown, along with the JTAG cable connector which in this case is used to program the memory device. Typically this connector may not be populated in production, but is used in development. Production FLASH memories are usually pre-programmed and don't need to be programmed from a cable. Some board designs will have multiple FPGAs in a chain supported by a single Large FLASH memory with multiple images.

Figure 7-14: Connection Setup for Programming the EPCS Using the AS Interface



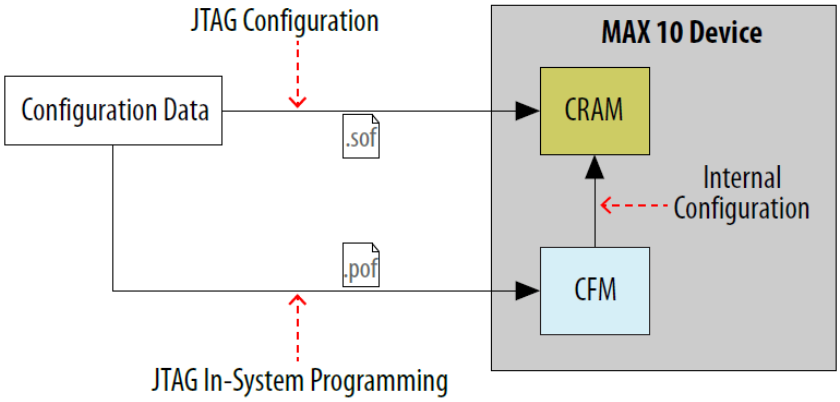
Another common configuration scheme is Fast Passive Parallel as shown here. A CPLD or Microprocessor tied to a memory device actively transfers the data to the FPGA on power up. This allows more control of the configuration process and is faster.

Figure 7-6: Single Device FPP Configuration Using an External Host



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

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



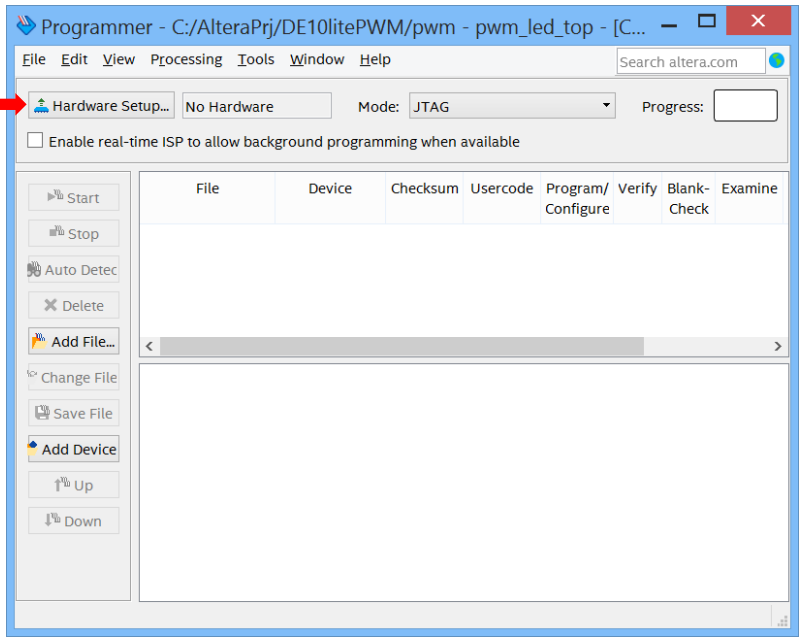
In addition to correctly designed hardware for a particular configuration scheme, you must also have a programming file to program the part. There are several types:

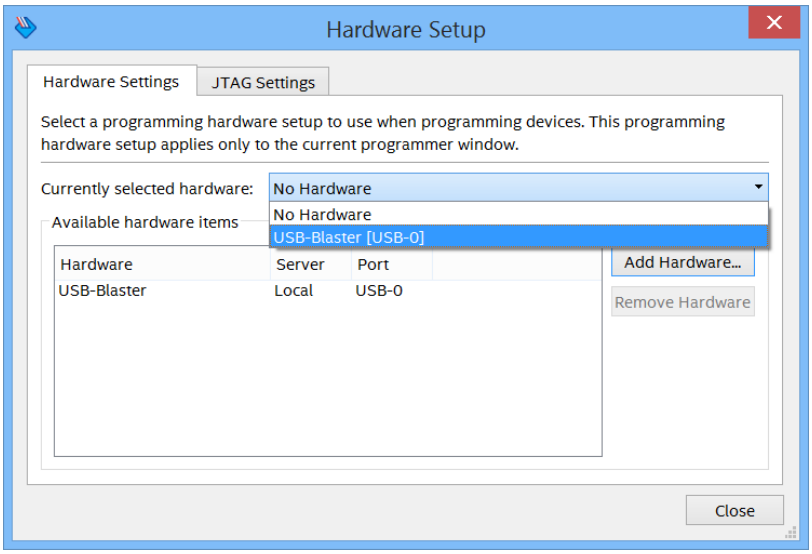
1. .sof (SRAM Object File) - which is used to configure FPGAs directly from Quartus Prime software through a download cable. This is the most common type of programming file as it is used for development when first testing the design by downloading it directly through the JTAG cable.
2. .pof (Programming Object File) – Used to Program CPLDs, FLASH FPGAs, and configuration FLASH memories.
3. .jam/.jbc – ASCII file used by processors and test equipment to program devices via JTAG
4. .jic (JTAG Indirect Configuration File) – is used to program EPCS (Altera serial configuration) devices through their dedicated configuration connection to an FPGA, as configuration devices do not have JTAG interfaces.

The Quartus Prime Programmer is a software tool that controls the programming of the FPGA. When you open the programmer from the tools menu or toolbar, it creates a chain description file (.cdf) that stores device programming chain information, as there can be more than one FPGA in the programming chain. A picture of this chain is shown in the programming window.

6.2. Programming the DE10-lite

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

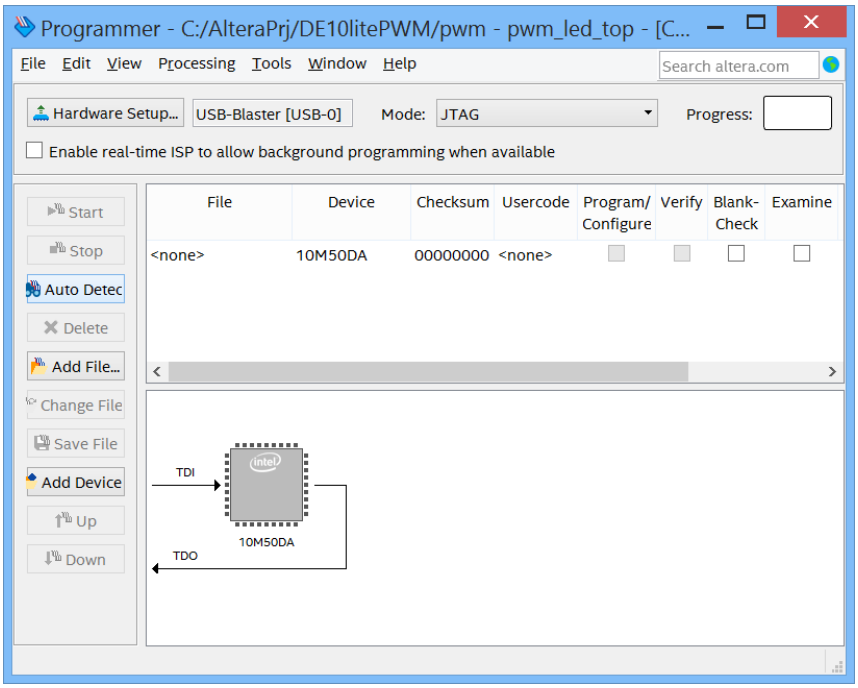




Close the Dialog Box and your setup should appear as this:

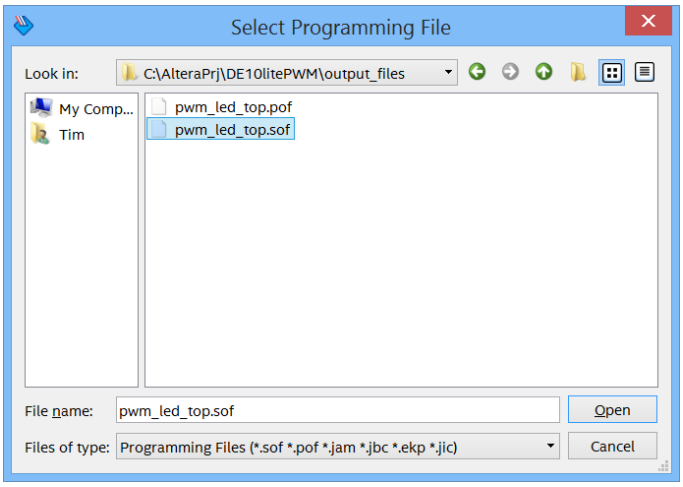
A programming device must have drivers installed and be detected correctly to be listed. Next select the programming mode – the most common is JTAG. The JTAG chain can consist of both non-Altera and Altera devices.

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

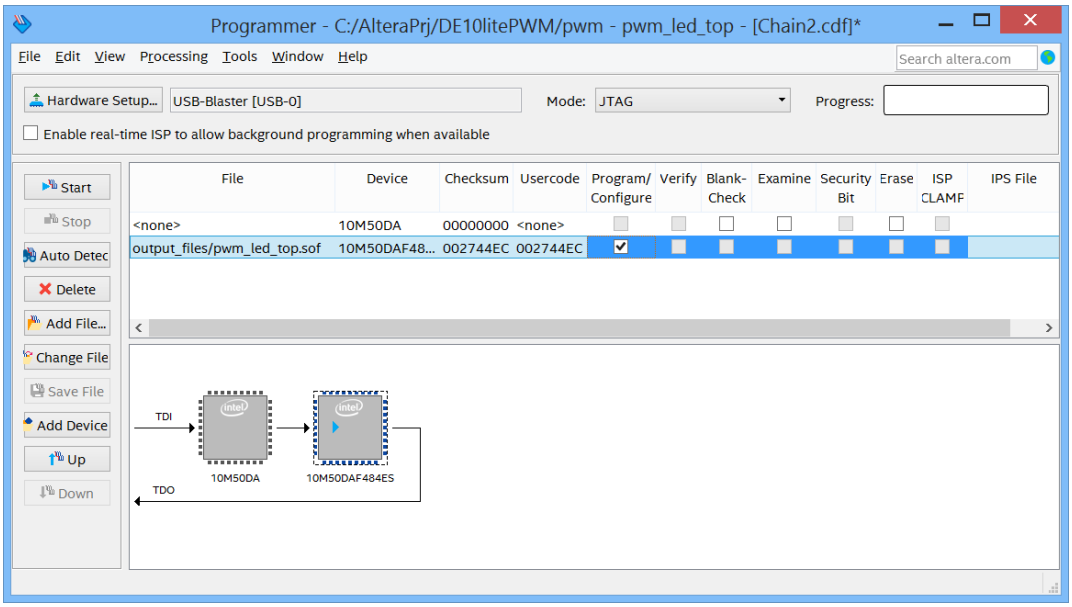


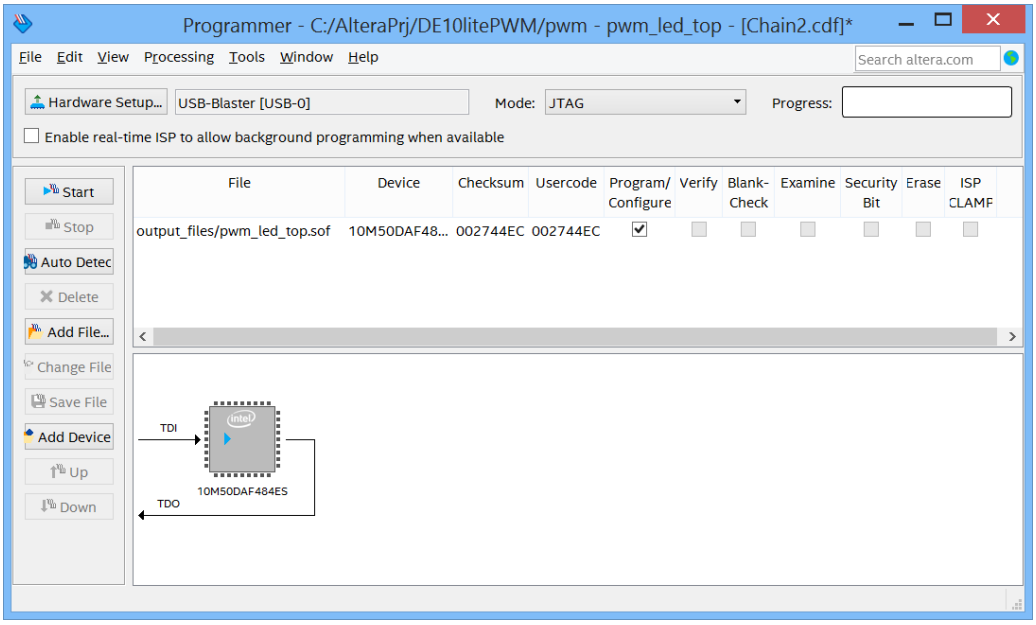
You can also verify a device after it has been programmed, or blank check a device, erase a device, or set a security bit if available.

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

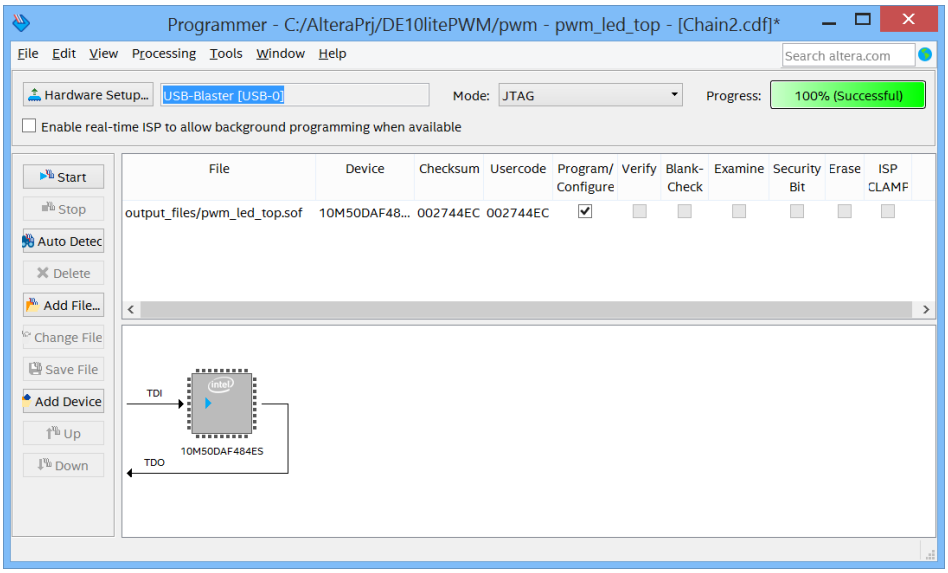


Click Open. Back in the programming window, you may have to delete any other entries that are listed.





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



7) Play with the board to see if the design behaves as expected.

CONGRATULATIONS!! You are done. You've completed your first FPGA design