

# *ECE 3170 Cryptographic Hardware for Embedded Systems*

Fall 2025

Assoc. Prof. Vincent John Mooney III

Georgia Institute of Technology

Lab 1, 50 pts.

Due Tuesday, September **10th** prior to 11:55pm  
(please turn in homework electronically on Canvas)

This lab sets up Quartus and ModelSim on your computer, and then goes through DES in C and VHDL. While this document is long, the vast majority is simply installing Quartus and troubleshooting problems that may pop up in the installation. The bulk of your time will be spent in the final pages working on the DES C code and VHDL.

## **VHDL Help**

There is plenty of documentation available on how to write good VHDL. Some good simple examples can be found [here](#). Some good YouTube videos introducing VHDL basics can be found [here](#).

### **Installation and Use of the Quartus Prime software**

This first section prepares you to begin actual labs using the DE-10 Standard board but it still requires a little bit of time. It is a tutorial to familiarize you with the installation and basic functionality of the Quartus software. The installation will be done in the Prelab 1 steps, and this will require a computer running Windows or Linux. Then a tutorial of creating a VHDL file and implementing it on the development board will be done in the Lab 1 steps.

This tutorial has been written using Quartus Prime version 19.1, but it is largely applicable to earlier versions, as far back as version 15. This document is largely derived from ECE 2031 Lab 0, so you may be able to skip some of the install steps if your computer is already set-up for use with a version of the DE-10 board from ECE 2031.

**NOTE: some previous ECE 2031 classes from a few years ago may have used a different FPGA device – specifically, the DE-10 Lite board – so it is possible that you may need to install Cyclone V device support (i.e., your previous environment may not work).**

Documentation and resources for the DE-10 Standard board can be found at:

<https://rocketboards.org/foswiki/Documentation/DE10Standard>

# Lab 1 Prelab Steps

This course will make extensive use of a computer-aided design (CAD) application called Quartus Prime, originally developed by the Altera Corporation, which is now a division of Intel. Quartus Prime will allow you to "capture" your designs with schematics and by other means, as well as fit the designs to be implemented on the FPGA board that is used as the primary development platform. Even when we build circuits with discrete integrated circuits, Quartus is a convenient way to draw schematics.

## Requirements to begin

You must have a computer meeting one of the following general descriptions:

- A Windows 10\* PC
- Windows 10\* in a Boot Camp hard disk partition
- Windows 10\* in a Parallels VM
- Windows 10\* in some other VM (e.g. VMware)
- A PC running one of the following Linux distributions (although it is likely others will work)
  - Red Hat Enterprise Linux 6
  - Red Hat Enterprise Linux 7
  - SUSE SLE 12
  - Ubuntu LTS (at least 14.04)
- One of the Linux variants above, in some VM

\* Windows 7 and Windows 8 still seem to work fine, but are not recommended.

Although all of these configurations SHOULD be supported, the only ones being actively tested by the instructors are specifically:

- A Windows 10 PC
- A PC running Ubuntu 18.xx

You will know by the end of Lab 1 if you have any issues with your computer.

If your computer has unresolvable issues, you may also use any of the computers in Klaus 1446 and skip to the start of the actual lab.

In all configurations, the following also apply:

You need a functional USB port, with a full-size [USB Type A receptacle](#), or some adapter, such as the USB-C to USB-A adapter needed for recent MacBooks ([one example, but there are others](#))

For the limited number of users who may still be running a 32-bit version of their operating system, you may have to upgrade to the 64-bit version of the operating system.

You will need at least 15 GB of disk space for the Quartus installation.

You must have administrative rights, allowing you to install software on the system. To check in Windows, open Settings (Gear icon), select Accounts, and next to your account name, it should show *Administrator*

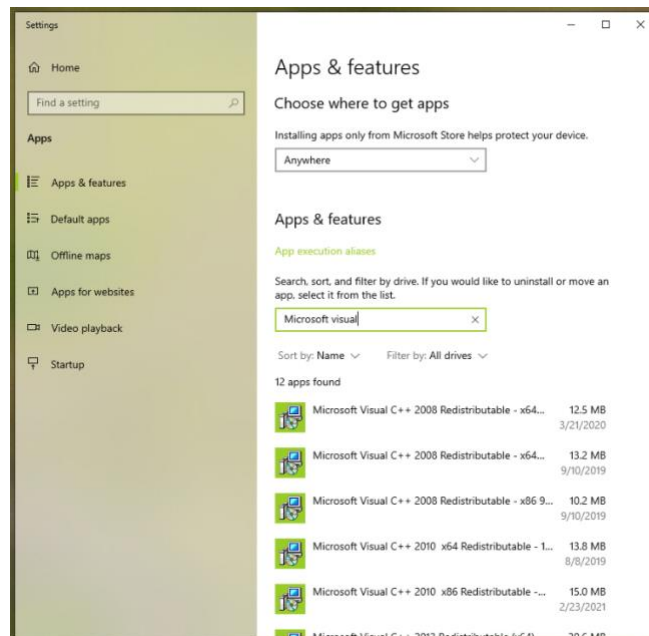
Once you have a computer at hand meeting these requirements, begin following the installation steps below. Most of them should be similar for all computer configurations, as long as you are doing them within the Windows or Linux operating system on your computer. This first step is the main exception, since it is not required for Linux users.

## Step 1.

Linux users should skip this step.

On any Windows system (including Windows running on a Mac VM or Bootcamp), the Quartus installation will require the Visual C++ Runtime system, officially the *Microsoft Visual C++ Redistributable*. Your system *probably* has it already, but it will be worthwhile to check now. There are two ways to check:

Option A) Open your Windows Settings (Gear icon), go to Apps & Features, and look for any entries beginning with *Microsoft Visual C++ Redistributable*.... If you have one or two ending with a date that is 2015 or later, you should be fine, and you can skip Step 2 and go to Step 3 below.



## Step 2.

(Optional, only if required based on the result of Step 1.) Go to the URL below, then download and run `vc_redist.x64`

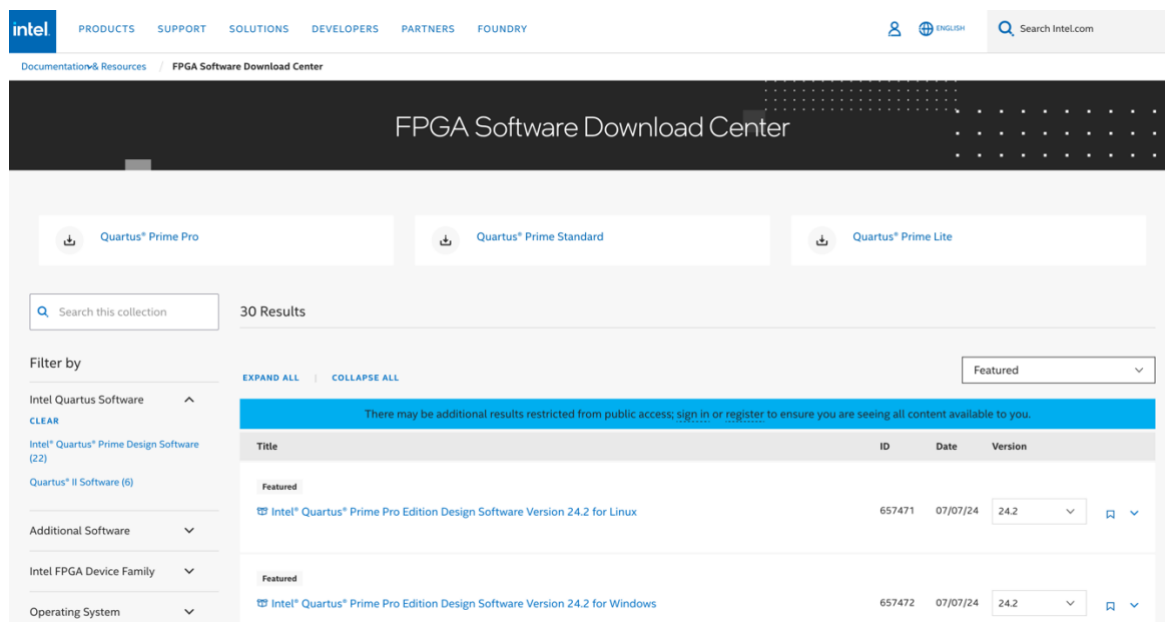
<https://www.microsoft.com/en-us/download/details.aspx?id=48145>

## Step 3.

In a browser **under Windows or Linux**, go to the download site:

<https://fpgasoftware.intel.com/19.1/?edition=lite&platform=windows>.

The top of the resulting page will resemble the image below.



Click on Quartus Prime Lite, then

select the Lite edition,  
select release 19.1, and  
select Windows (or Linux, if it applies to you).

If the Downloads list is empty, you can select and install the 18.1 version.

We have found that this website **may not work properly with Chrome or Firefox**, so if you have trouble downloading files in the steps that follow, try **Edge** or other browsers.

You can also use the following links if the previous one doesn't work:

**Windows:** <https://www.intel.com/content/www/us/en/software-kit/664527/intel-quartus-prime-lite-edition-design-software-version-19-1-for-windows.html>

Linux: <https://www.intel.com/content/www/us/en/software-kit/664524/intel-quartus-prime-lite-edition-design-software-version-19-1-for-linux.html>

It does not matter if you are on a Mac — you should not be in MacOS at this point, so select the Windows version of Quartus.

The screenshot shows the Intel FPGA Software Download Center page for Intel® Quartus® Prime Lite Edition Design Software Version 19.1 for Linux. The page has a blue header with the Intel logo and navigation links: PRODUCTS, SUPPORT, SOLUTIONS, DEVELOPERS, PARTNERS, and FOUNDRY. A search bar is on the right. Below the header, the page title is "Intel® Quartus® Prime Lite Edition Design Software Version 19.1 for Linux". A table lists software details: ID (664524), Date (3/31/2019), Software Type (FPGA Development Tools), Software Package (Quartus® Prime Lite), Version (19.1), and Operating Systems (Linux). A yellow banner below the table states: "A newer version of this software is available, which includes functional and security updates. Customers should [click here](#) to update to the latest version." Below this, there is a section with links for users to upgrade, technical recommendations, and support team contact. Further down, there are links for OS support, release notes, and critical issues/patches. At the bottom, there is a "Downloads" section with tabs for Multiple Download, Individual Files, Additional Software, and Copyleft Licensed Source.

## Step 4.

Further down the web page in your browser are the actual downloads, under the Downloads section. If the options for **Multiple Download** and **Individual Files** do not appear, switch the version to 18.1.

The screenshot shows the "Downloads" section of the Intel Quartus Prime Lite Edition Design Software Version 19.1 for Linux page. The "Downloads" tab is selected, and the "Individual Files" sub-tab is active. Below the tabs, there is a section titled "Devices" with a dropdown menu. The dropdown menu is open, showing a list of devices. The first device is "Intel® Cyclone® 10 LP Device Support". Below it, there is a blue button labeled "Download" with the filename "cyclone10lp-18.1.0.625.qdz". To the right of the button, the size is listed as "Size: 266.1 MB" and the SHA1 hash is "SHA1: ec942162e4aee221359da8fe63d61f6264b816".

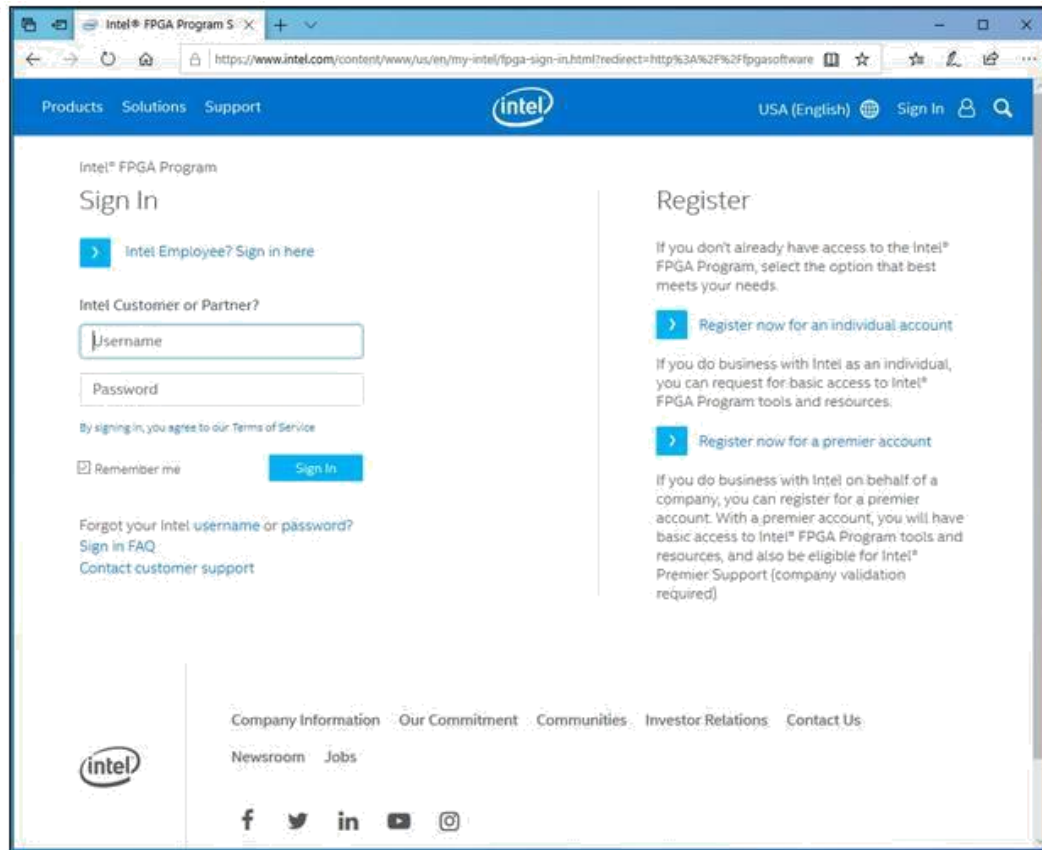
If you don't mind a larger installation, you may choose the multiple download option and install everything, but this will take more disk space. We won't include instructions for that here, but if you do so, skip to step 6.

To install only the minimum requirements, we will use the **Individual Files** option.

We will start by downloading **Quartus Prime (includes Nios II EDS)**, which can be found under the **Intel Quartus Software** section (below **Devices**). In the past, clicking this redirected to a

sign-in page. If this happens, follow the rest of this step to create an individual account. If the file downloads, skip to step 5.

The sign in screen appears below. Take the option to "Register now for an individual account," go through the necessary steps, and navigate back to the screen above, but now with signed-in status.



After signing in, when you get redirected back to the download, you will probably be in the wrong version of Quartus Prime. It may be easiest to follow the link again, now that you are signed in: <https://fpgasoftware.intel.com/19.1/?edition=lite&platform=windows>

In any event, **make sure you still have the correct operating system selected, along with the Lite version 19.1.**

## Step 5.

There are three tabs in the window — choose the **Individual Files** tab. There may be a "Download Selected Files" button. That seems to be non-functional, with no way to select files in browsers I have tested. Instead, simply follow the instructions in the drop-down "Download and Install Directions, clicking on "More"." Specifically, download these four files to a single temporary folder:

"Quartus Prime (includes Nios II EDS)"

"ModelSim-Intel FPGA Edition (includes Starter Edition)"


"Cyclone V device support. (1434.3MB)" — (under "Devices")

"Quartus Prime Help" (in the "Additional Software" tab) — you can omit this, if you prefer to save a little space.

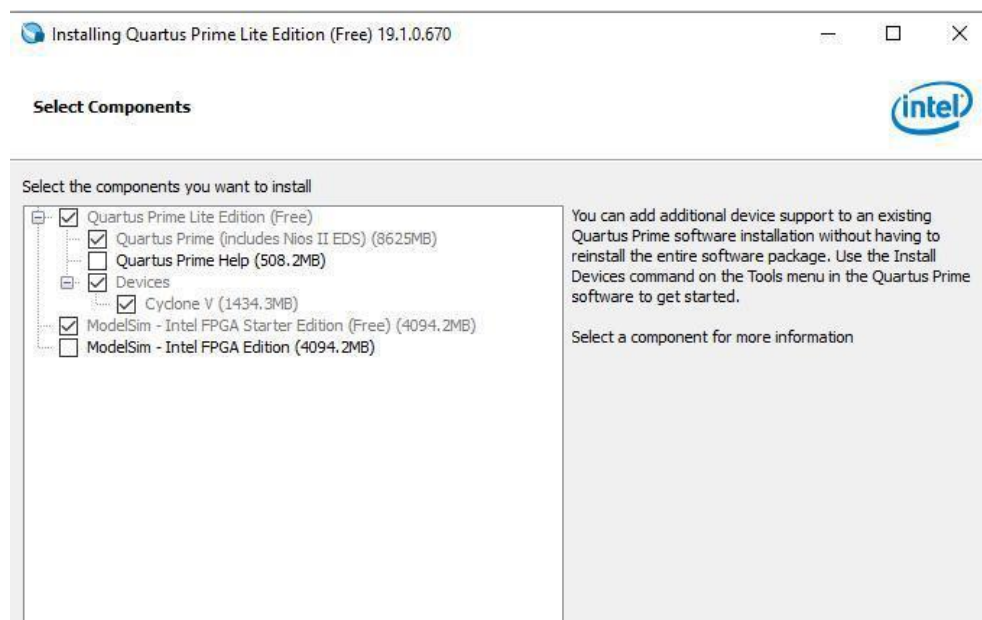
## Step 6.

Double-check after your files have completely downloaded. Go to the folder where you downloaded, and you should see four files (three if you omitted the Quartus Prime Help). If your installation file does not start with "QuartusLiteSetup-19.1...", then you have downloaded the wrong version.

The most common installation error is that one or more files is missing or wrong, and it can result in having to start over. Verify that you have the files shown here:

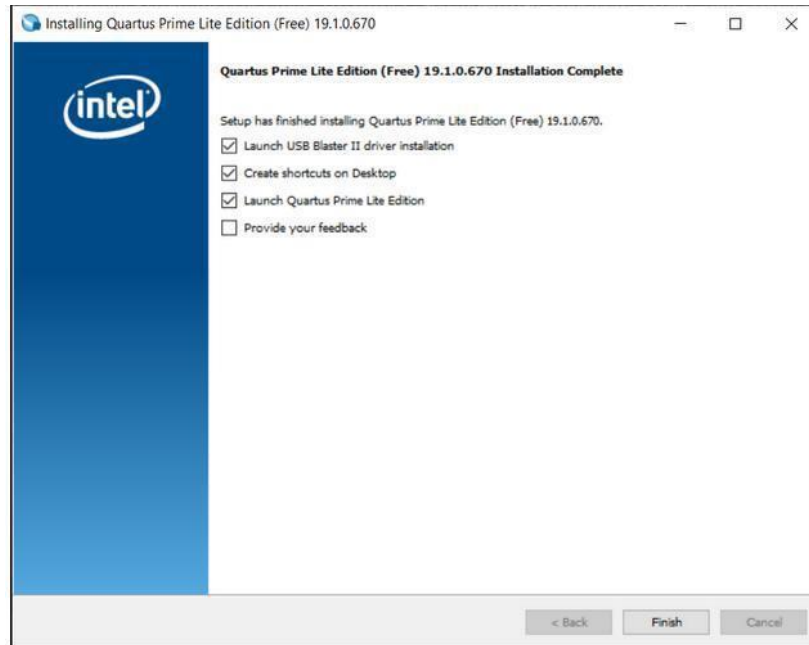
 QuartusLiteSetup-19.1.0.670-windows.exe	8/10/2021 6:44 PM	Application	1,599,126 KB
 ModelSimSetup-19.1.0.670-windows.exe	8/10/2021 6:42 PM	Application	991,415 KB
 cyclonev-19.1.0.670.qdz	8/10/2021 6:44 PM	QDZ File	1,412,204 KB
 QuartusHelpSetup-19.1.0.670-windows.exe	8/18/2021 6:39 PM	Application	282,067 KB

Still in that folder, run the *QuartusLiteSetup...* executable, **not** the *QuartusHelpSetup* or the *ModelSimSetup*. Follow all the instructions to complete the installation. It is recommended that you accept the default installation folder, usually *C:\intelFPGA\_lite\19.1*. One step will be to confirm that you want to install the other items that you downloaded into the folder. The example below shows the items that you must install. It also shows that you should NOT install any edition of ModelSim other than the "Starter Edition," so if another one is there, leave it unchecked.



## Step 7

The installation can take a while. When it nears the end, the following window appears. Leave these three items checked, or uncheck the option for creating shortcuts, if desired. But make sure that the option to Launch USB Blaster II driver is checked.



You will be presented with the Device Driver Installation Wizard. Make sure you click Next to continue. It is very possible that this installation will fail, giving the response below.

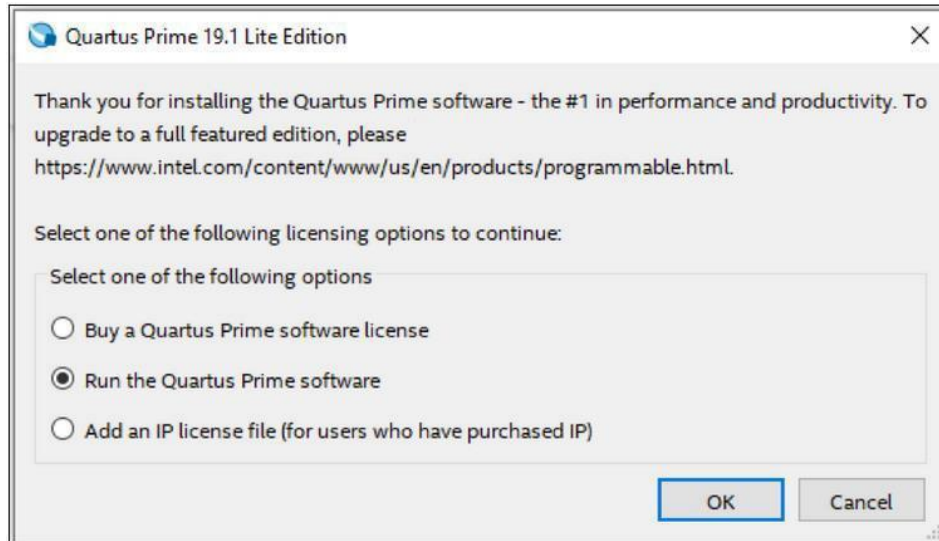
**If you get this error, it will affect what you do in a later step. Consider yourself in the group of users who need to follow the steps to manually install the USB-Blaster, in a step coming up very soon.**





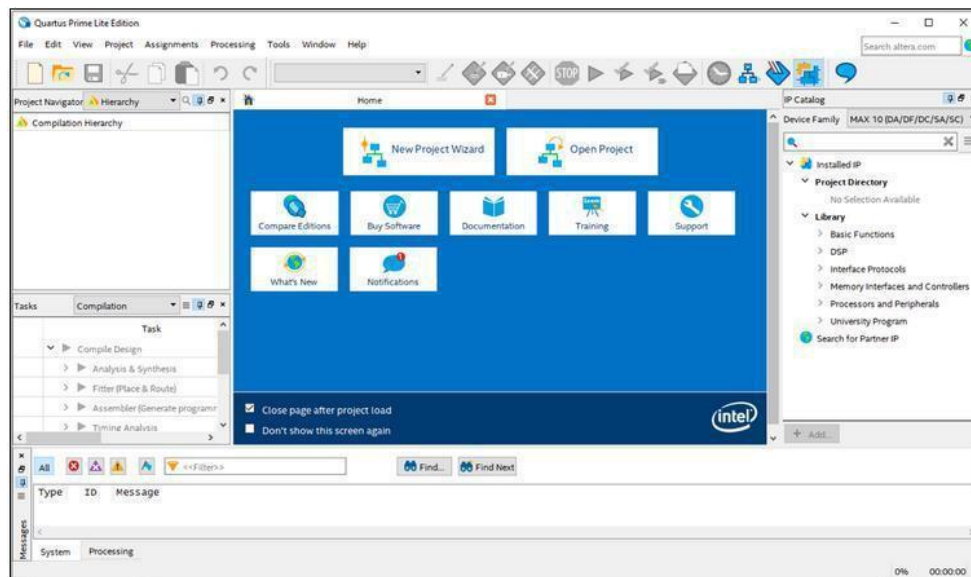
## Step 8.

As it finishes the installation, you will be presented with the option below. At this time, and at any future time where it may ask, select "Run the Quartus Prime software." No purchase is necessary for this class.



## Step 9.

When Quartus opens, it will appear similar to the image below. If it takes a long time, you may need to relaunch it, the previous popup should not appear again.



You can delete the installation files from the temporary folder where you launched them.  
<https://www.microsoft.com/en-us/download/details.aspx?id=48145>

## Step 10.

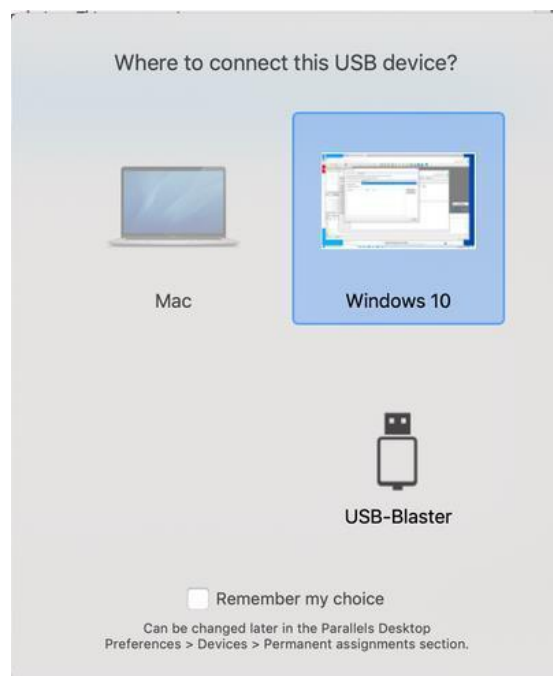
If, only a couple steps back, you received an error when installing the USB-Blaster driver, then do not proceed with any further steps until you first complete the steps in a separate section below, titled *Manual USB-Blaster Installation*.

Return to the next step, when finished.

## Step 11.

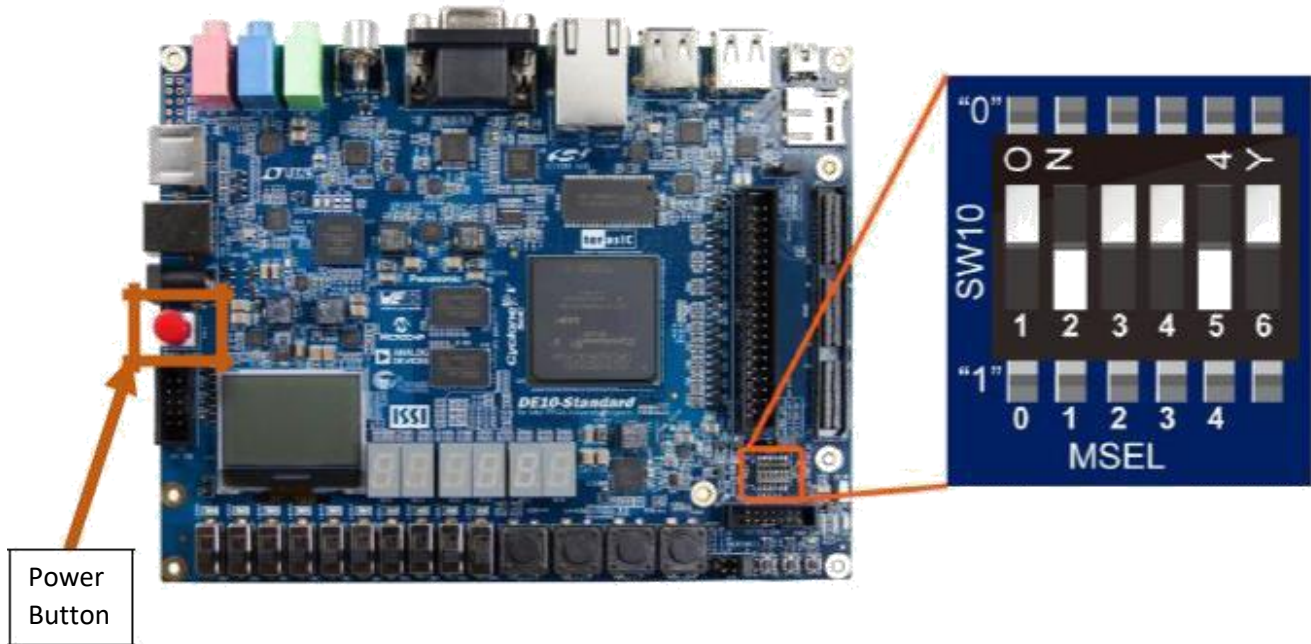
There is one additional thing to check at this time if you have already received your DE10-Standard FPGA development board. If not, this will happen near the end of the Lab 1 steps.

Connect your DE10-Standard board to your computer with the supplied white/gray USB cable. If you have Windows running in a VM on a Mac, you may get a prompt asking you if the newly-detected device should be connected to the Mac or to Windows 10. Choose Windows, if you have this prompt.



The DE10-Standard should power up and immediately begin running the default programming file that is stored in non-volatile memory. We are going to verify that you can reprogram it through your cable.

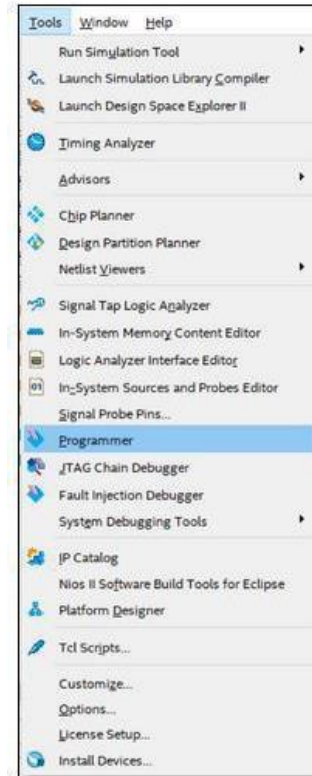
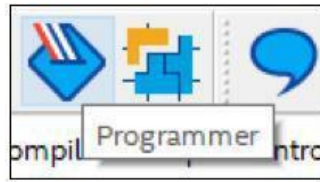
If it does not turn on immediately, check the following:



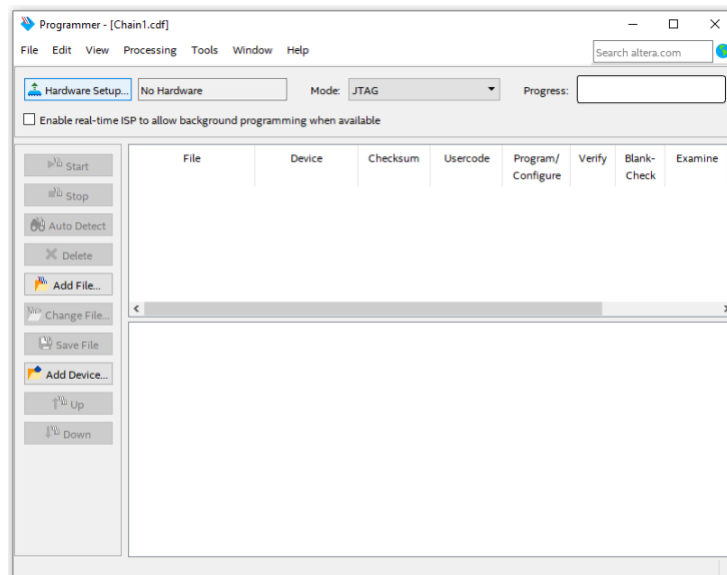
Ensure the MSEL switches are in the positions shown above and the red power button is pushed to the “down” position.

## Step 12.

In Quartus, find the Programmer by either method below. Either look for the icon at the top (the one on the left of the three shown below) or the menu item under Tools, also shown below. Click on it to open

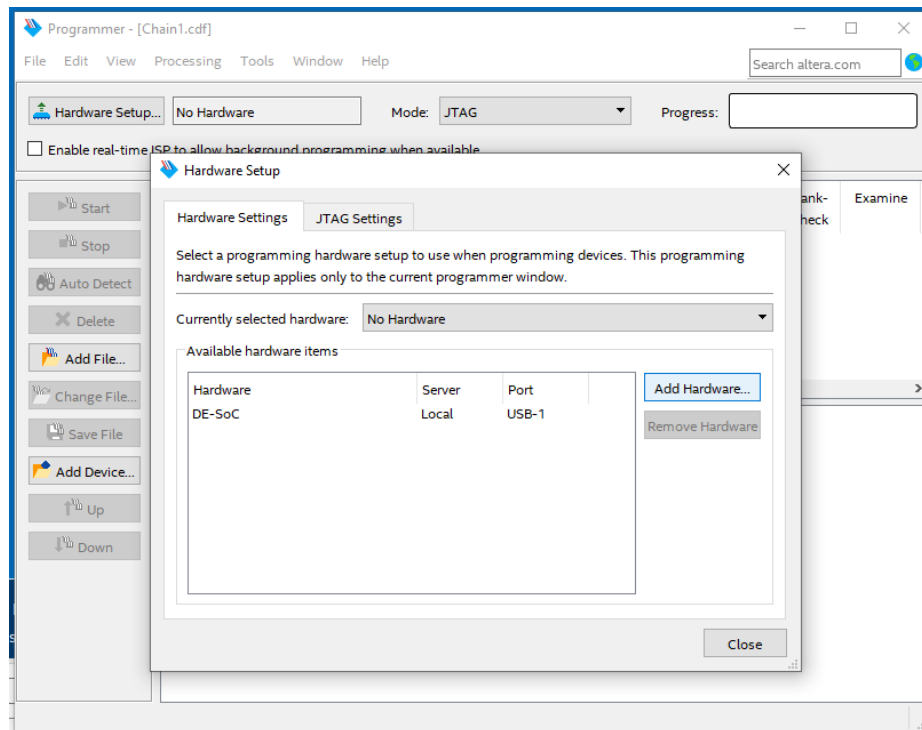


The Programmer will come up and appear similar to the image below. In particular, it will say "No Hardware" near the top, just to the right of "Hardware Setup," because it has never been configured to use the USB connection, called the *USB-Blaster*. Go ahead and click on "Hardware Setup."



### Step 13.

The “Hardware Setup” window is shown below. In the in “Available hardware items,” if “USB-Blaster” or “DE-SoC” appears, then you are done. But usually, you need to click on "Add Hardware," and in the window that comes up, select USB-Blaster (or EthernetBlaster if only that shows up).



If you can do that successfully, (unless you are on linux) then you will also be done. If not:

A) First make sure that the USB cable is plugged in **completely** to your computer and to the board. It is possible for it to supply power, yet not be connected properly. Then try again.

B) Should that fail, continue with the next step.

**Linux users will need to take some additional steps, even if USB-Blaster is present, because of USB access permissions.**

The best description I've found is here: <https://blog.atomminer.com/fighting-altera-usb-blaster-on-ubuntu/>

Some students in summer reported that the script provided there didn't work, and someone updated it to work for their setup, so you might have better luck with this script: [usbblaster.sh](https://github.com/atomminer/usbblaster.sh).

**Step 14.**

If the previous step was unsuccessful, and you still are unable to make the USB-Blaster appear, contact the instructors for assistance. There may be a discussion thread in Canvas on this topic, if needed.

**Step 15.**

Continue with the Lab Steps for Lab 1.

## Manual USB-Blaster Installation

If you were able to use your USB-Blaster the first time you needed it to program your FPGA board, you should not need to follow any of the steps below. But if you saw errors near the end of the installation of Quartus Prime, indicating that NO device drivers were installed successfully, continue here. The problem was probably that Windows would not install the unsigned device driver provided by Intel.

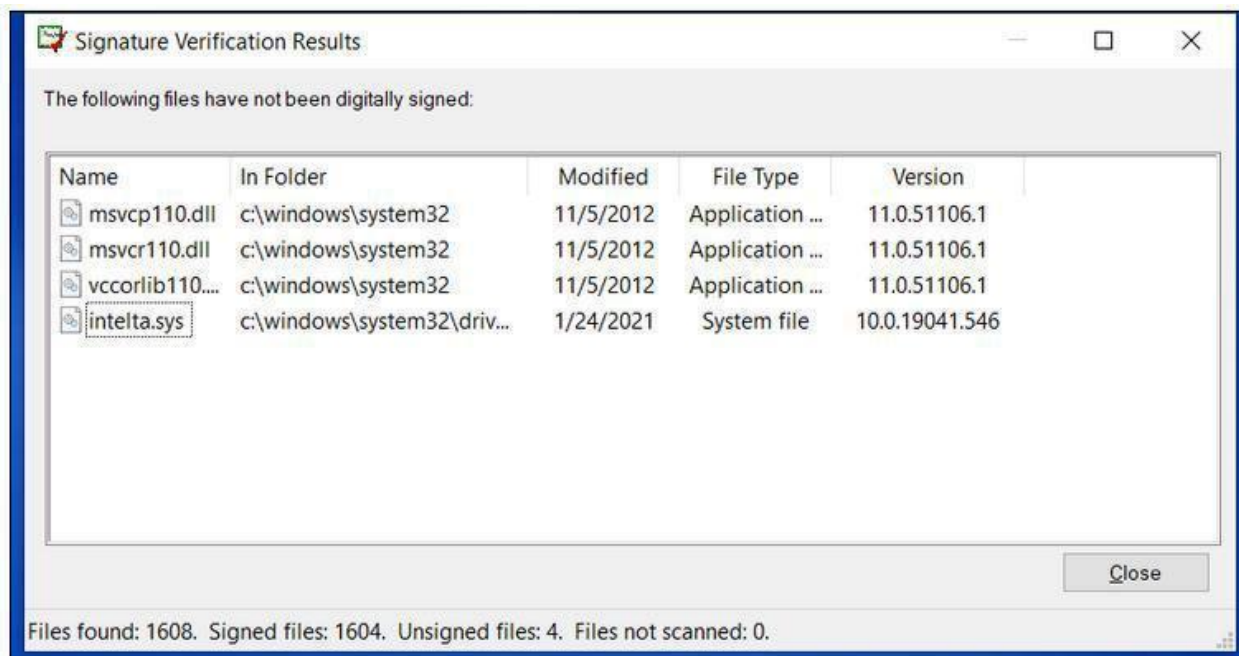
### Requirements to begin

You must have successfully installed Quartus (except possibly the USB-Blaster device driver) on a Windows machine, either a PC, or Windows running on a Mac.

### Step 1.

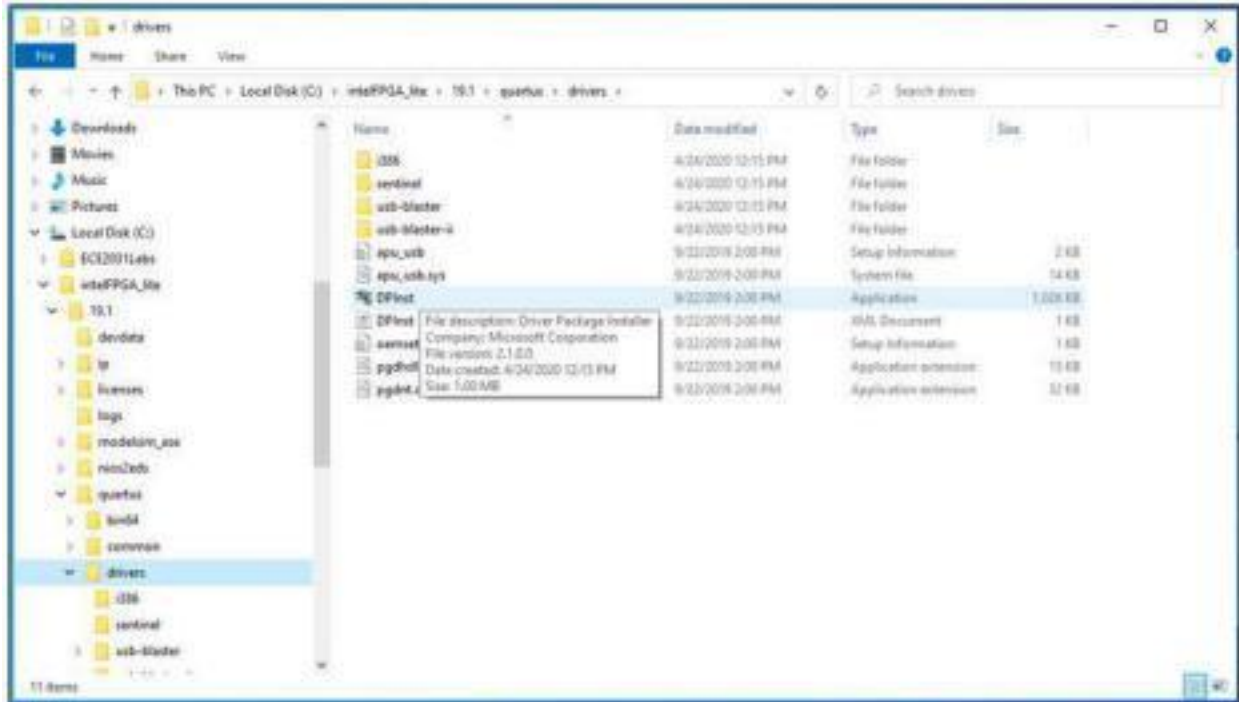
None of this is relevant to Linux users. If you have trouble programming the board on Linux, we will try to help, but we have very limited experience using Quartus on Linux.

First, check to see if the driver possibly DID install correctly. In the text entry area of the Windows Start Menu, type `sigverif` and press return. It will take a while to find all device drivers that are unsigned, but when it is done, see if your computer is missing the file name `intelta.sys`. The image below shows the result in a system that DOES have the driver installed correctly, along with three other unrelated drivers.



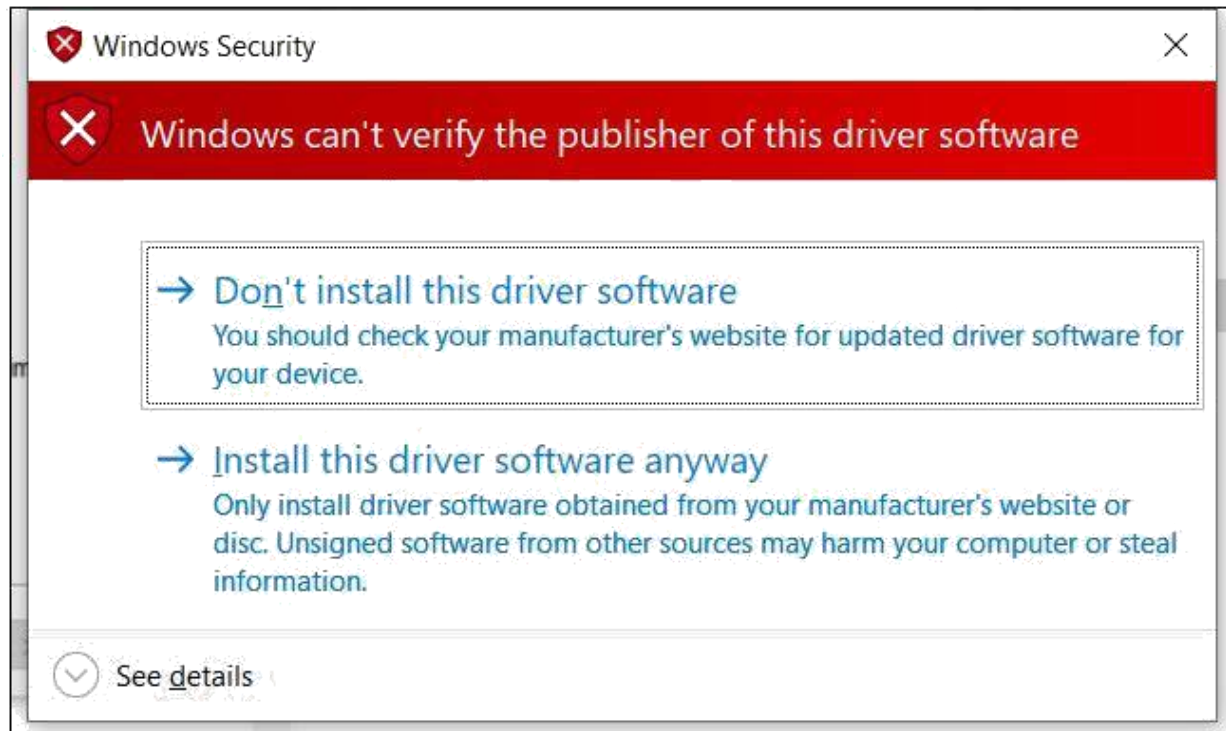
## Step 2.

Assuming your system does **not** have the driver, open Windows File Explorer and navigate to the folder *19.1/quartus/drivers*, within the folder that you used to install Quartus (which defaults to */intelFPGA\_lite* on the system drive). Below, the file *DPinst* (the application, not the XML document) is highlighted.



Double-click on that application and continue with the installation. Ignore warnings like the one below, and select "Install this driver software anyway".

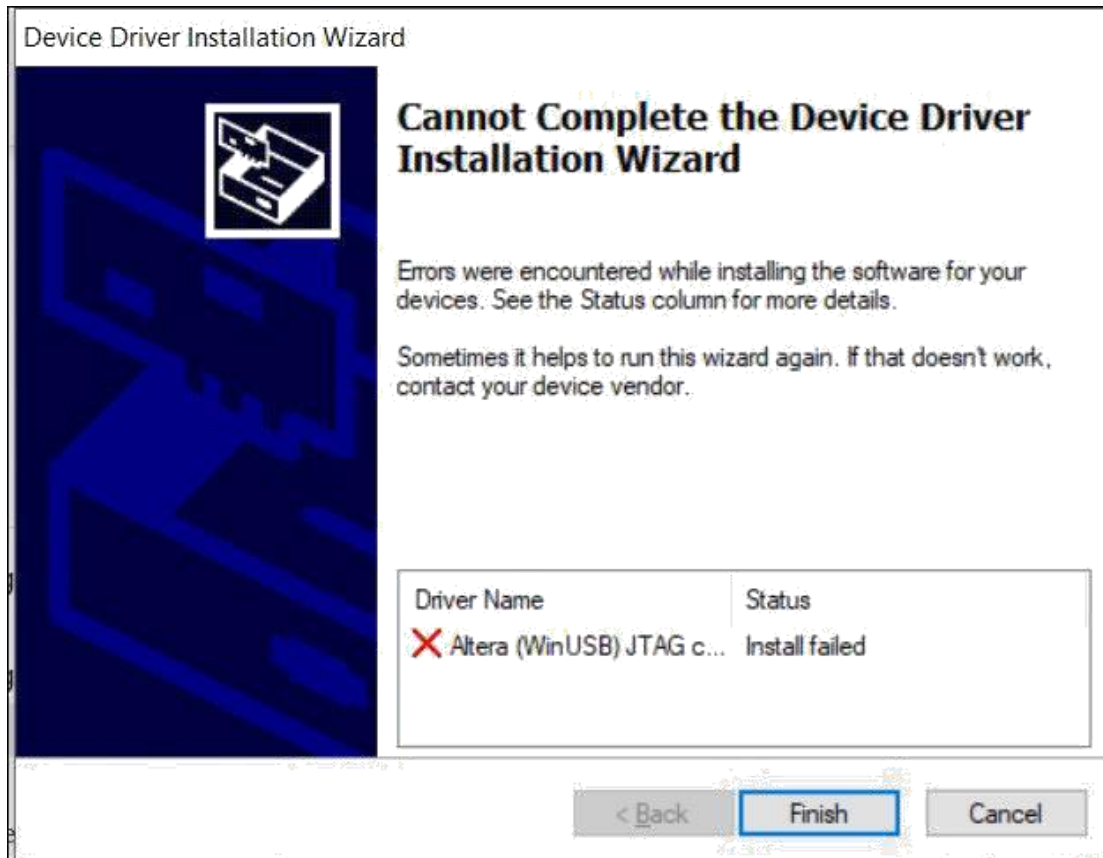




When it is finished, it may show that some items were not installed. But if the Altera USB-Blaster Device Driver was installed, as shown below, then this part was successful.

### Step 3.

If the previous step was unsuccessful, you probably saw a message similar to the one below, or one that showed multiple devices, with none installed successfully.



The only known cause for this is that your Windows system is set to not allow unsigned device drivers (those not tested by Microsoft) to be installed. It can be complicated to override this setting, but first we will try the easy way.

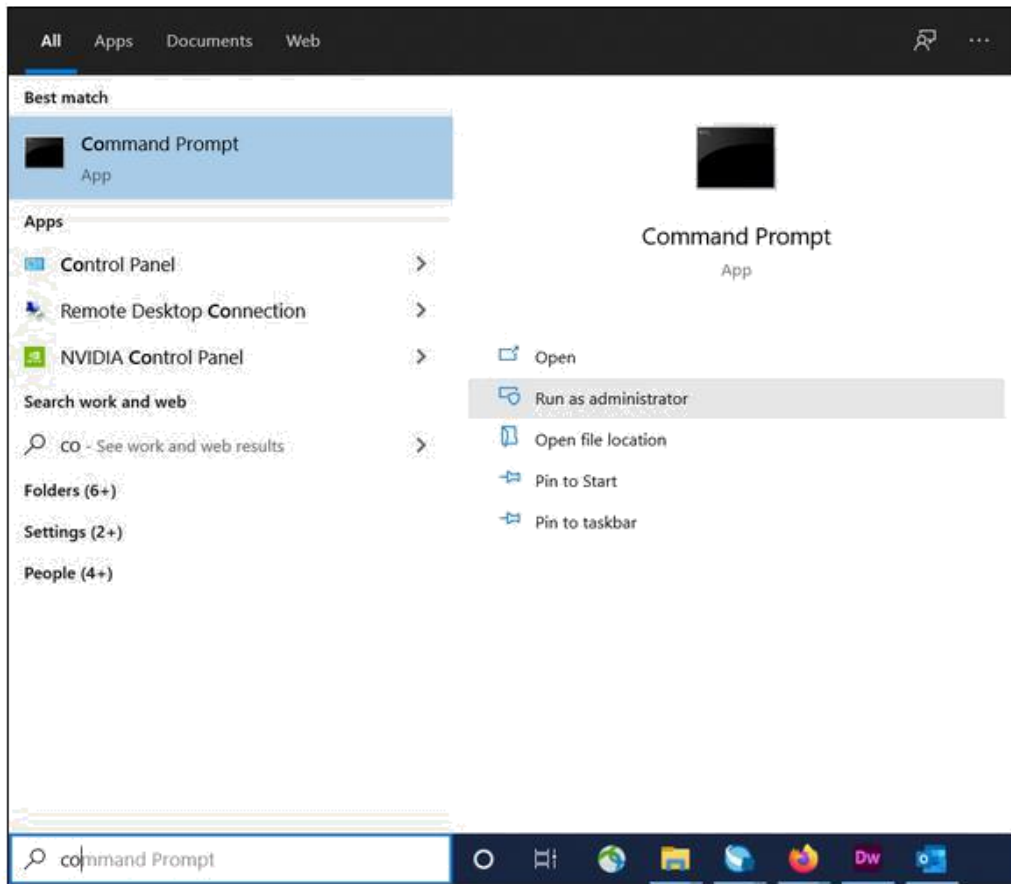
#### Step 4.

First, launch an Administrator Command Prompt by

clicking the Start button, then

typing at least the first few letters of "Command Prompt," and

clicking on "Run as administrator" in the Command Prompt window that comes up, as shown below.

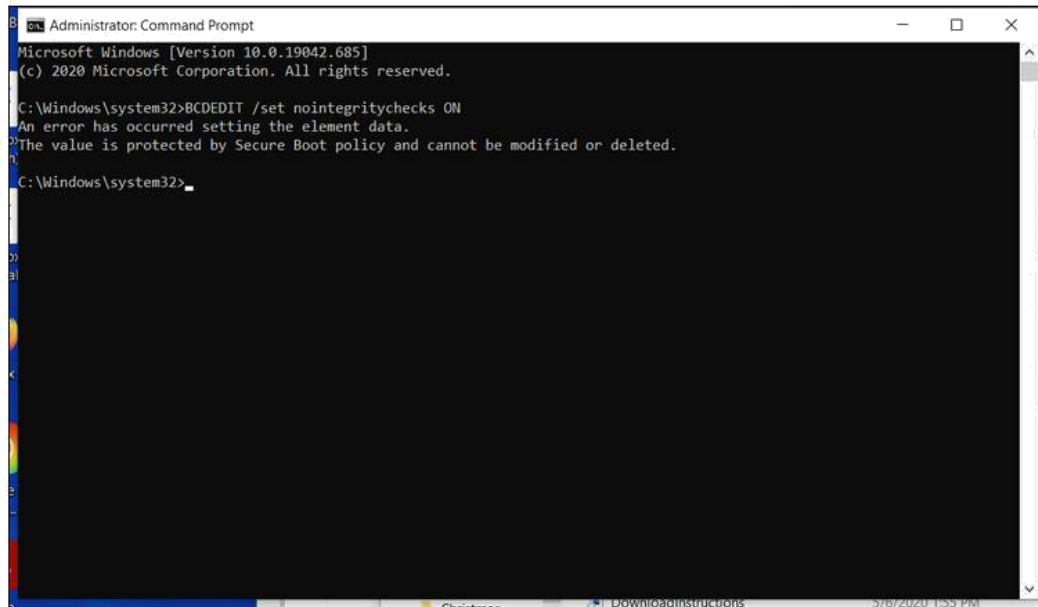


## Step 5.

In the Command Prompt, type the following command:

```
BCDEDIT /set nointegritychecks ON
```

If you get an error related to the Secure Boot Policy, as shown below, continue with the next step. If you do **not** get this error, then you can go back to Step 2 above and see if it succeeds. But if it does not, continue to the next step.



```
Administrator: Command Prompt
Microsoft Windows [Version 10.0.19042.685]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Windows\system32>BCDEDIT /set nointegritychecks ON
An error has occurred setting the element data.
The value is protected by Secure Boot policy and cannot be modified or deleted.

C:\Windows\system32>
```

## Step 6.

If you have not succeeded yet, then we have the more complicated situation where we have to reboot the system in a mode that will allow installation of unsigned drivers. Begin by printing the instructions that follow, or making some notes, or opening them on a different device, because you will need to reboot your computer.

If your computer has Bitlocker enabled, **you will need to know your Bitlocker recovery key**. If you do not have it saved somewhere, do a search on "recover bitlocker" for tips (and consider saving it in whatever you use for password management in the future).

You need to get the advanced boot options menu, and the first step is to hold down the Shift key while you click the "Restart" option in Windows. Your computer will restart into the menu below. Choose "Troubleshoot."

# Choose an option



**Continue**

Exit and continue to Windows 10



**Turn off your PC**



**Use a device**

Use a USB drive, network connection, or Windows recovery DVD



**Troubleshoot**

Reset your PC or see advanced options

In the Troubleshoot menu, as shown below, select "Advanced Options."

# ← Troubleshoot



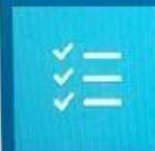
## Reset this PC

Lets you choose to keep or remove your personal files, and then reinstalls Windows.



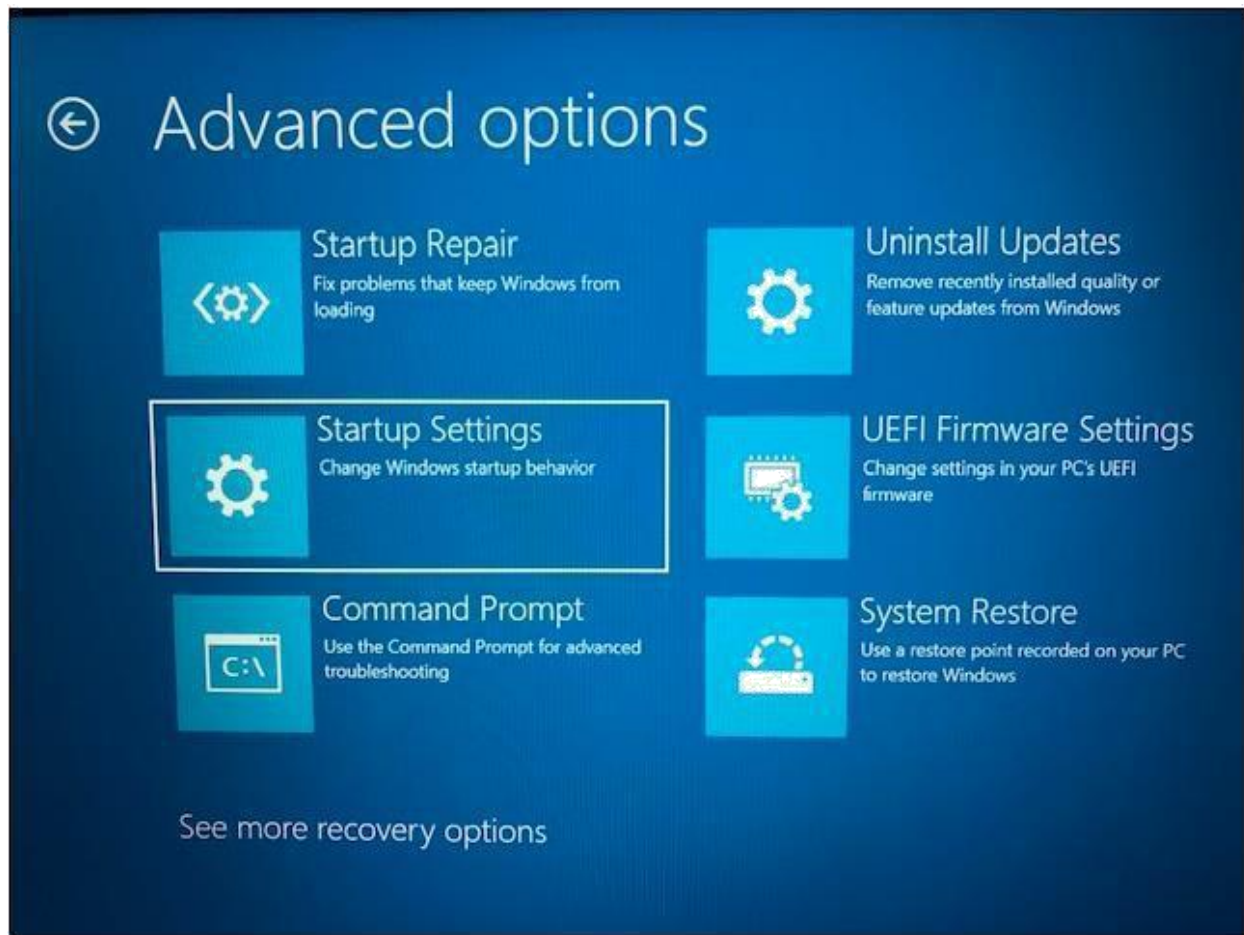
## Factory Image Restore

Restore your system software to a saved system image.



## Advanced options

In the Advanced Options menu, as shown below, select "Startup Settings."



Clicking "Restart" on the following screen (shown below) will result in a reboot. But before clicking it, note two things:

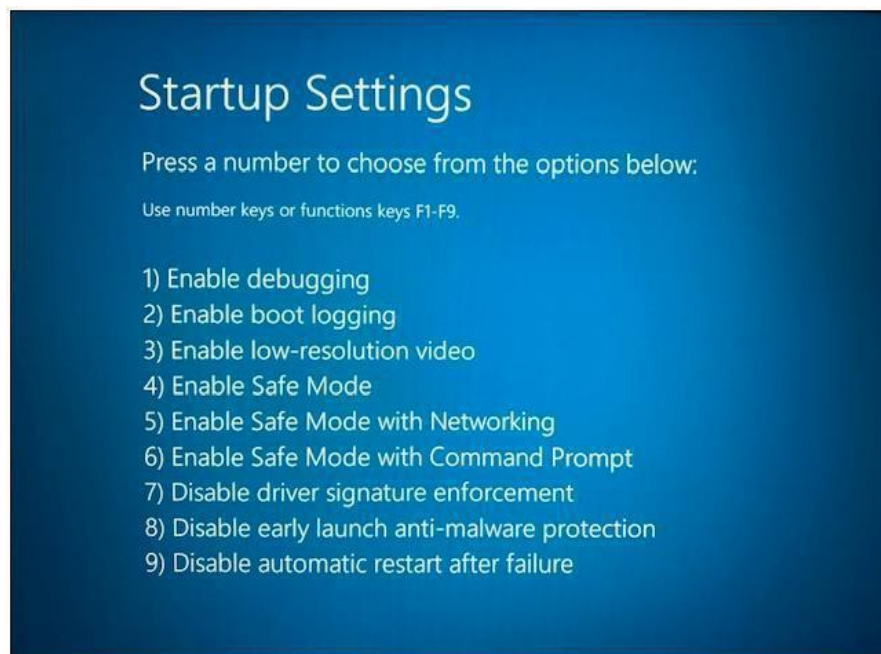
If your computer has Bitlocker enabled, this is where you will need to know your Bitlocker recovery key.

You will need to do the driver installation on this next reboot. If you boot the system again, you will lose the ability to install unsigned drivers, and you will have to go back to restarting while holding the shift key, then going through the remaining menus.





Once the computer restarts, you will be presented with the following menu. Select option 7, "Disable driver signature enforcement".



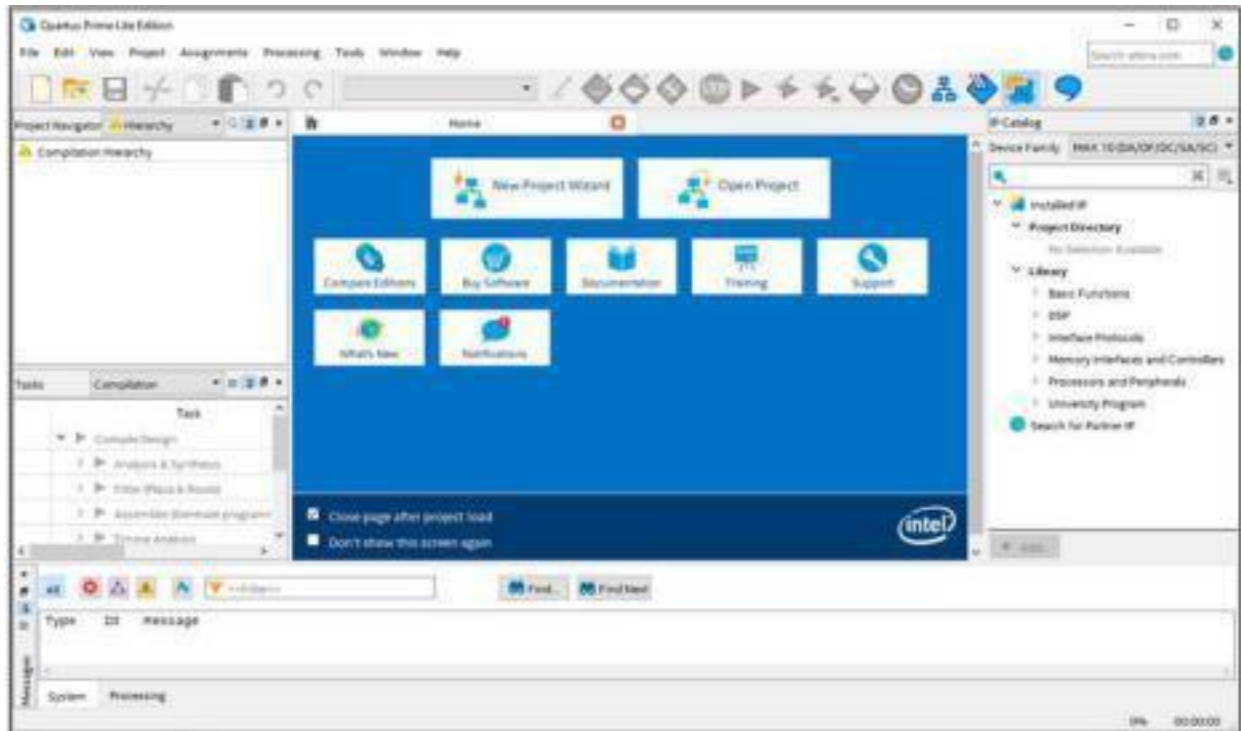
Once back in Windows, return to step 2 above and run DPInst.exe, which should now succeed. If not, contact the instructors, because this is the last process we have found necessary for anyone so far.



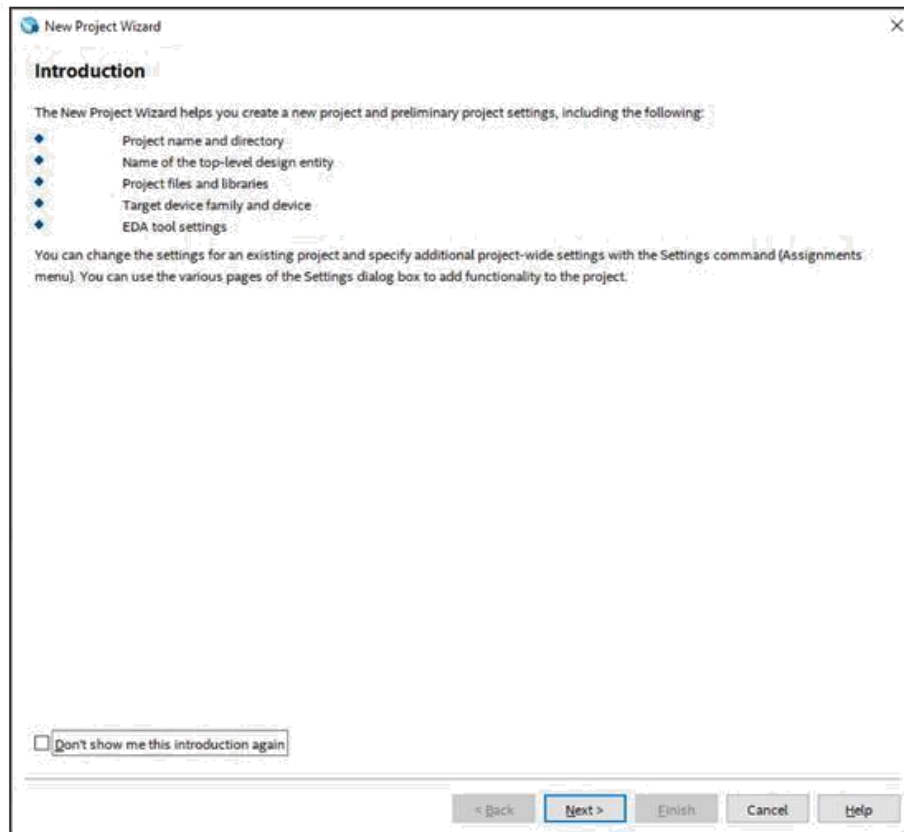
# Lab 1 Main Lab Steps (After Prelab)

The concepts and procedures outlined here should enable you to begin working on laboratory projects. Additional features will also be explained throughout the laboratory exercises. However, you should refer back to this tutorial if at any time you forget any of these basic functions.

## Step 1.

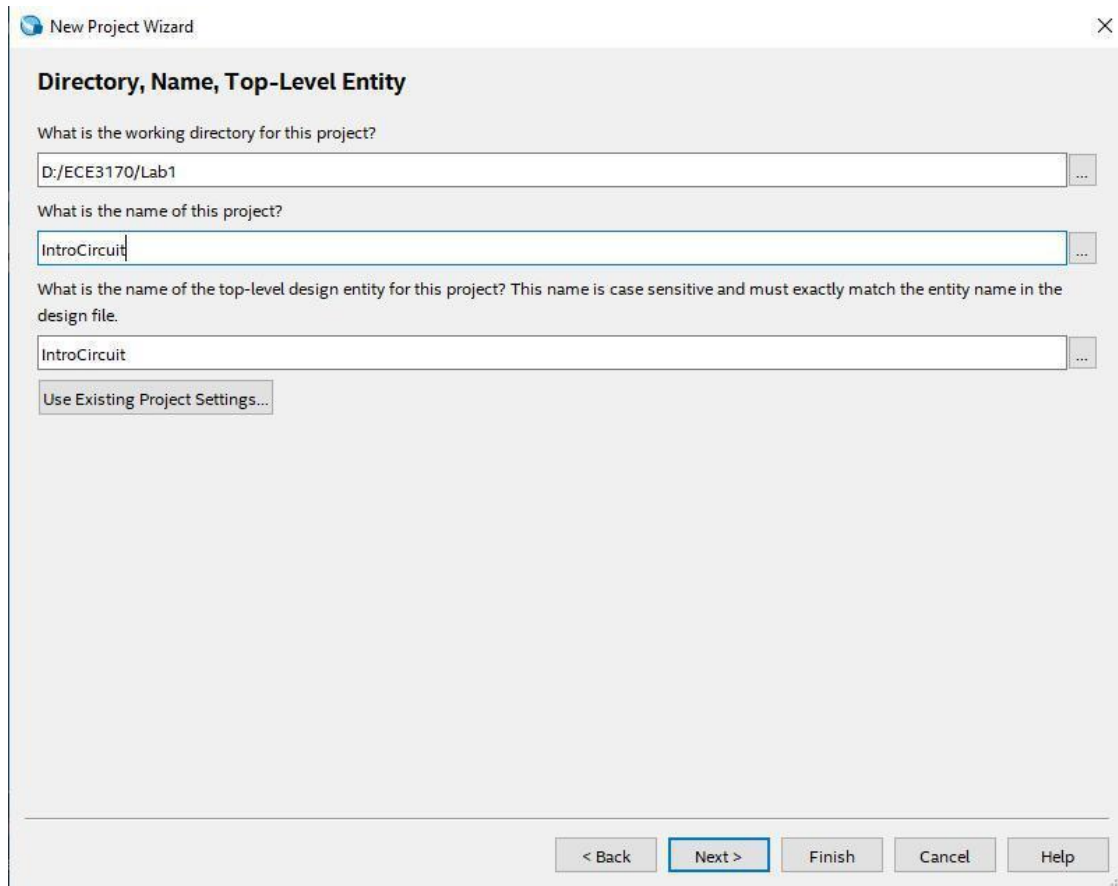


Upon starting Quartus Prime (and perhaps dismissing a window that appears with the Lite edition), the main interface window will be presented as seen above. From the menu at the top, select **File => New Project Wizard...** to create a new project. (**NOT File ==> New**, because there is a fundamental difference between creating a project and creating a single file.) At this point, Quartus will display an introduction screen for the project wizard.



Get in the habit of reading information windows before moving on. On this screen, select **Next** to advance to the next window. You may also want to check the box in the lower left corner to avoid displaying this particular introduction screen again.

## Step 2.



New Project Wizard

**Directory, Name, Top-Level Entity**

What is the working directory for this project?

D:/ECE3170/Lab1

What is the name of this project?

IntroCircuit

What is the name of the top-level design entity for this project? This name is case sensitive and must exactly match the entity name in the design file.

IntroCircuit

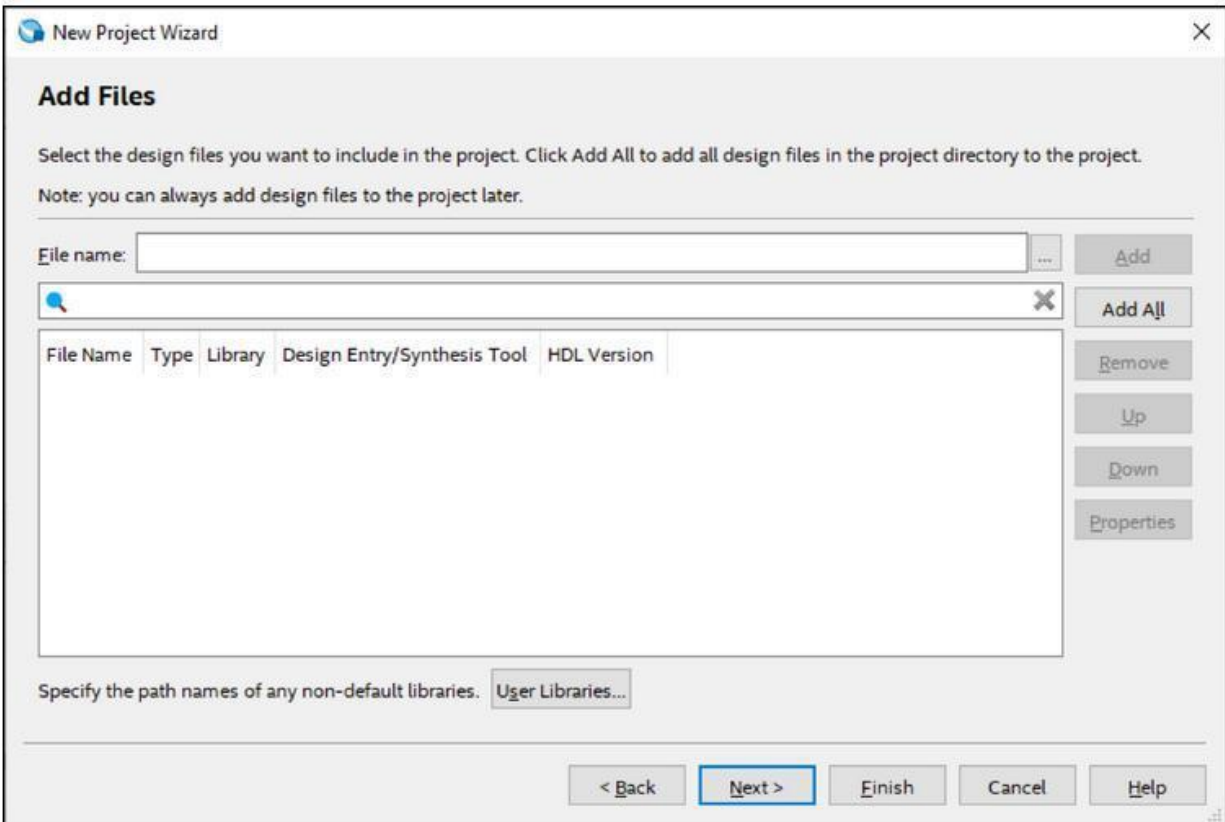
Use Existing Project Settings...

< Back Next > Finish Cancel Help

Enter a location to store your project files, as in the example above. **NOTE: You cannot store the project files in the default location as that folder is read-only. Choose a location outside of the C:/IntelFPGA\_lite folder.** Include a dedicated directory (i.e., folder) for the particular project, which is D:/ECE3170/Lab1 in the example here. Next, give the project a name. The example uses *IntroCircuit* for the name of the project. The last entry is the top-level design entity, which is an important concept. A project can consist of as little as a single design file, such as a single schematic. Or, it can contain one file which describes how other design files are interrelated. We will create this project with only one file, which by definition must be the top-level design entity. Quartus will default to using the project name for the top-level design file, which works well here.

Then, select **Next** to advance to a window which asks you to choose between an "Empty project" and "Project template". Select **"Empty Project"**

### Step 3.



Press **Next** to advance to the window above. Allow Quartus to create the project directory, if it does not yet exist. The user has the opportunity to add any files to the project that define logic, such as schematics created beforehand. Since there are no design files to add to this project, click **Next** to advance to the device selection window of the figure below.

## Step 4.

New Project Wizard

### Family, Device & Board Settings

Device Board

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

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

Device family

Family: Cyclone V (E/GX/GT/SX/SE/ST)

Device: All

Target device

☐ Auto device selected by the Fitter

☒ Specific device selected in 'Available devices' list

☐ Other: n/a

Show in 'Available devices' list

Package: Any

Pin count: Any

Core speed grade: Any

Name filter:

☒ Show advanced devices

Available devices:

Name	Core Voltage	ALMs	Total I/Os	GPIOs	GXB Channel PMA	GXB Channel P
5CSXFC6D6F31A7	1.1V	41910	499	457	9	9
5CSXFC6D6F31C6	1.1V	41910	499	457	9	9
5CSXFC6D6F31C7	1.1V	41910	499	457	9	9
5CSXFC6D6F31C8	1.1V	41910	499	457	9	9
5CSXFC6D6F31C8ES	1.1V	41910	499	457	9	9
5CSXFC6D6F31I7	1.1V	41910	499	457	9	9
5CSXFC6D6F31I7ES	1.1V	41910	499	457	9	9

< >

< Back Next > Finish Cancel Help

You will need to select the correct device for the board you are using, first by selecting a device family, and then by selecting a specific target device. If you followed the recommendation to only install device support for the Cyclone 5 family, then your "Family" choice will default to Cyclone V (E/GX/GT/SX/ST), but if you chose to install support for other devices if already had support for other families installed, you may have to select this family yourself from the dropdown.

Now, you need to find the exact device by reading the corresponding code near the middle of the FPGA IC (*Integrated Circuit*), on your DE10-Standard board. Look closely at the board and

make a note of the complete device name for the Cyclone V chip. It is the long string directly below "Cyclone V SoC."

Continuing with the *Family, Device & Board Settings* dialog, we now need to specify the exact device, using the device name you just found. To select the specific device, you could

- scroll through the list at the bottom of the window to find it
- or start typing the name in the *Name filter* box, or
- if you knew the Package, Pin Count, or Core speed grade, use those dropdowns.

Select the correct device in the bottom list. Press **Next** to continue to the window below.

New Project Wizard

### EDA Tool Settings

Specify the other EDA tools used with the Quartus Prime software to develop your project.

EDA tools:

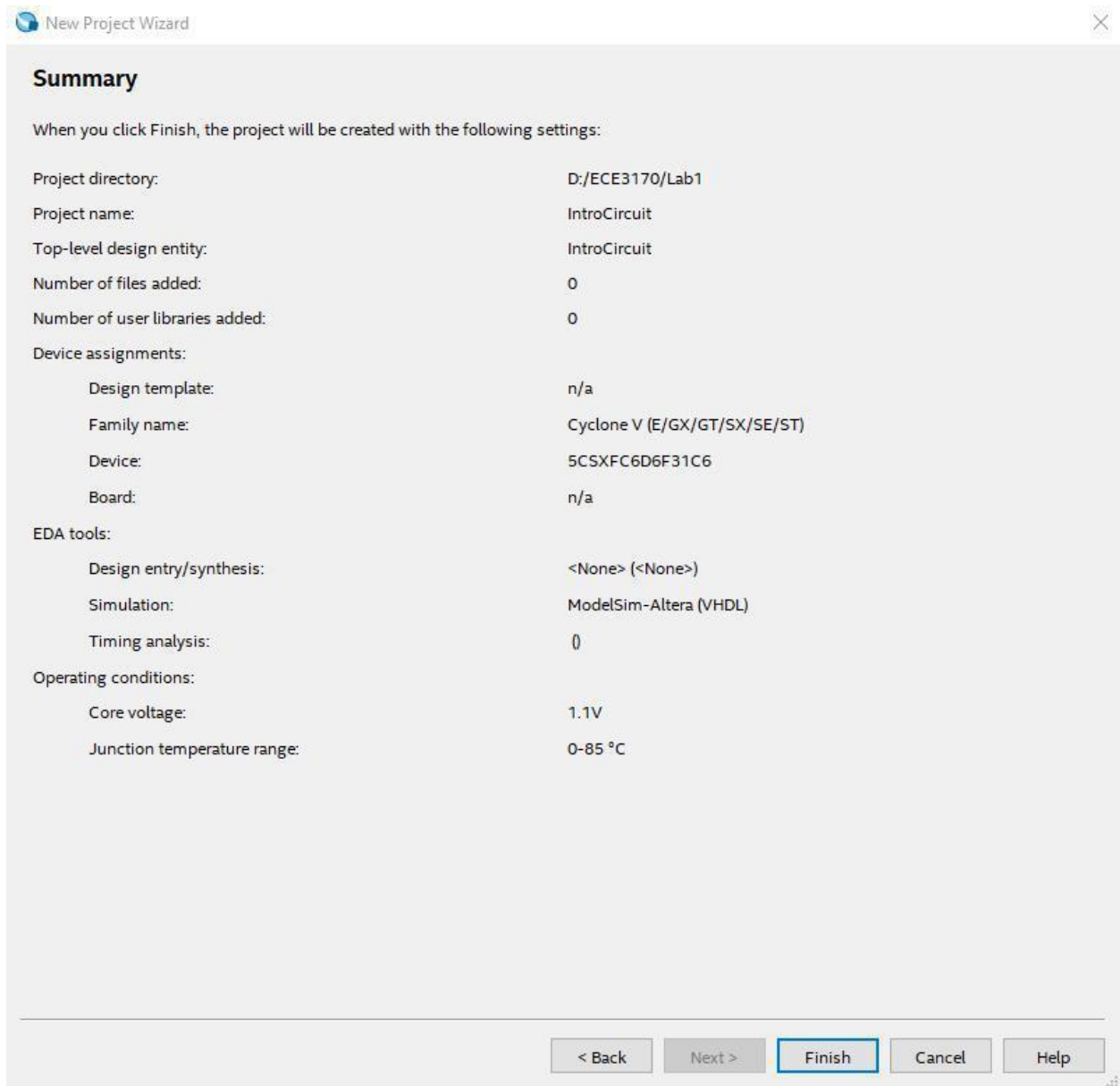
Tool Type	Tool Name	Format(s)	Run Tool Automatically
Design Entry/Synth...	<None>	<None>	<input type="checkbox"/> Run this tool automatically to synthesize the current design
Simulation	ModelSim-Altera	VHDL	<input type="checkbox"/> Run gate-level simulation automatically after compilation
Board-Level	Timing	<None>	
	Symbol	<None>	
	Signal Integrity	<None>	
	Boundary Scan	<None>	

< Back   **Next >**   Finish   Cancel   Help

It is common for commercial entities to use third-party tools to develop hardware and software configurations for FPGAs. This window allows Quartus to communicate with and use such tools within the development environment. We will mostly be using the default tools included within Quartus, but we will use a version of a widely-supported simulator called ModelSim. To the right of Simulation,

select *ModelSim-Altera* (NOT generic *ModelSim*), and  
select *VHDL* in the second dropdown (under *Format(s)* )

Then click **Next** to proceed to the summary window below.



The screenshot shows the 'New Project Wizard' window, specifically the 'Summary' tab. The window title is 'New Project Wizard' with a close button. The 'Summary' section states: 'When you click Finish, the project will be created with the following settings:'. Below this, the settings are listed in a two-column format:

Project directory:	D:/ECE3170/Lab1
Project name:	IntroCircuit
Top-level design entity:	IntroCircuit
Number of files added:	0
Number of user libraries added:	0
Device assignments:	
Design template:	n/a
Family name:	Cyclone V (E/GX/GT/SX/SE/ST)
Device:	5CSXFC6D6F31C6
Board:	n/a
EDA tools:	
Design entry/synthesis:	<None> (<None>)
Simulation:	ModelSim-Altera (VHDL)
Timing analysis:	()
Operating conditions:	
Core voltage:	1.1V
Junction temperature range:	0-85 °C

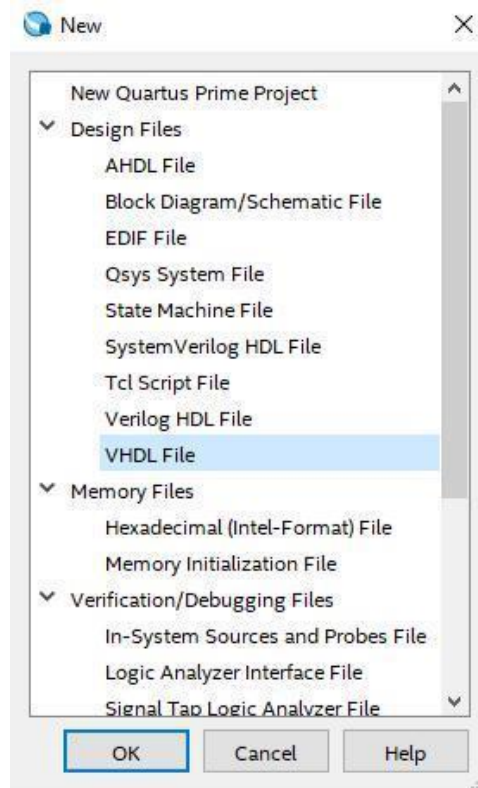
At the bottom of the window, there are five buttons: '< Back', 'Next >', 'Finish' (which is highlighted with a blue border), 'Cancel', and 'Help'.

With the summary window in view, press **Finish** to close the wizard and create the new project files. You will be returned to the main window of Quartus.

Optionally, this is a good time to take a break. Note that you can choose the menu option **File => Save Project** and then **File => Close Project** to return to a place where you can close Quartus entirely. You would, of course, have to get back to the same point with **File => Open Project**, selecting the *IntroCircuit* project just saved. Note that this is different from **File => Open File**, which would open an isolated file, without context to the project settings, including the device assignments.

## Step 5.

Design projects are composed of one or more design files, and this project will be defined by a single VHDL file. Earlier, the concept of a *top-level design entity* was mentioned, to define the one file that contains the overall design definition in some entity (device). To reiterate, we're only going to have one file, containing the one top-level design entity. From the menu, select **File => New...** to bring up the dialog below, and select **VHDL File** as was done here.



Quartus will only let you save it after you type something in it, so go ahead and add the library and use statements that will be used for this file:



```
library ieee;
use ieee.std_logic_1164.all;
```

This will be your top-level design file for this project, so save it using the same name as your project so that Quartus automatically uses it as the top-level design.

## Step 6. Device ports

Add this entity and port statement to your VHDL. The entity name should match your file name, because Quartus will be looking for a device with that name.

```
entity IntroCircuit is
port(
    B0    : in  std_logic;
    B1    : in  std_logic;
    B2    : in  std_logic;
    S0    : in  std_logic;
    S1    : in  std_logic;
    S2    : in  std_logic;
    S3    : in  std_logic;
    clk   : in  std_logic;
    LED0  : out std_logic;
    LED1  : out std_logic;
    LED2  : out std_logic;
    LED3  : out std_logic;
);
end IntroCircuit;
```

-- Describes the device from the outside  
-- Defines the signals coming into and out of the device

## Step 7. Device architecture

We will implement a few simple logic expressions to ensure proper operation of the DE-10 Standard board. Because this program is quite simple, you probably do not need to declare any internal signals. You can assign each output with a single line of VHDL. The architecture name is up to you (here it's been named "behavior"), but it does need to be declared as an architecture **of your device**, so use your device name from the entity statement for that part of the architecture declaration.

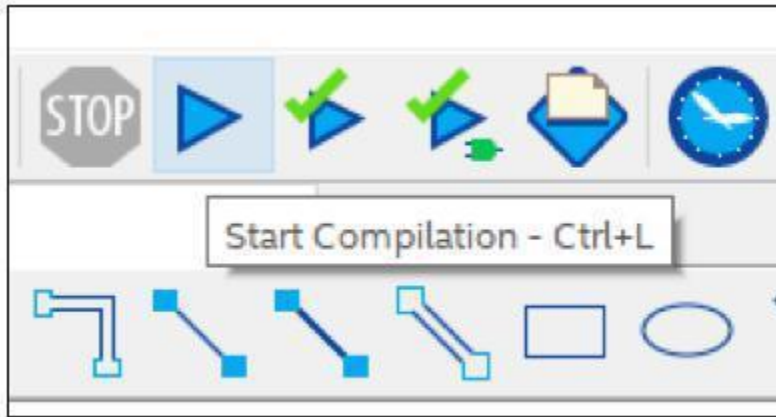
```
architecture behavior of IntroCircuit is
begin
    LED0 <= NOT B0;
    LED1 <= B1 AND S0;
    LED2 <= S1 XOR S2;

    process(clk, B2) is
    begin
        if rising_edge(clk) and B2 = '0' then
            LED3 <= S3;
        end if;
    end process;

end behavior;
```

## Step 8.

Save the file and compile the project. With the file saved, compile the design by selecting **Processing => Start Compilation**, or by clicking on the corresponding icon (the triangle to the right of STOP) in the menu bar shown below.



Fix any compilation errors before moving on.

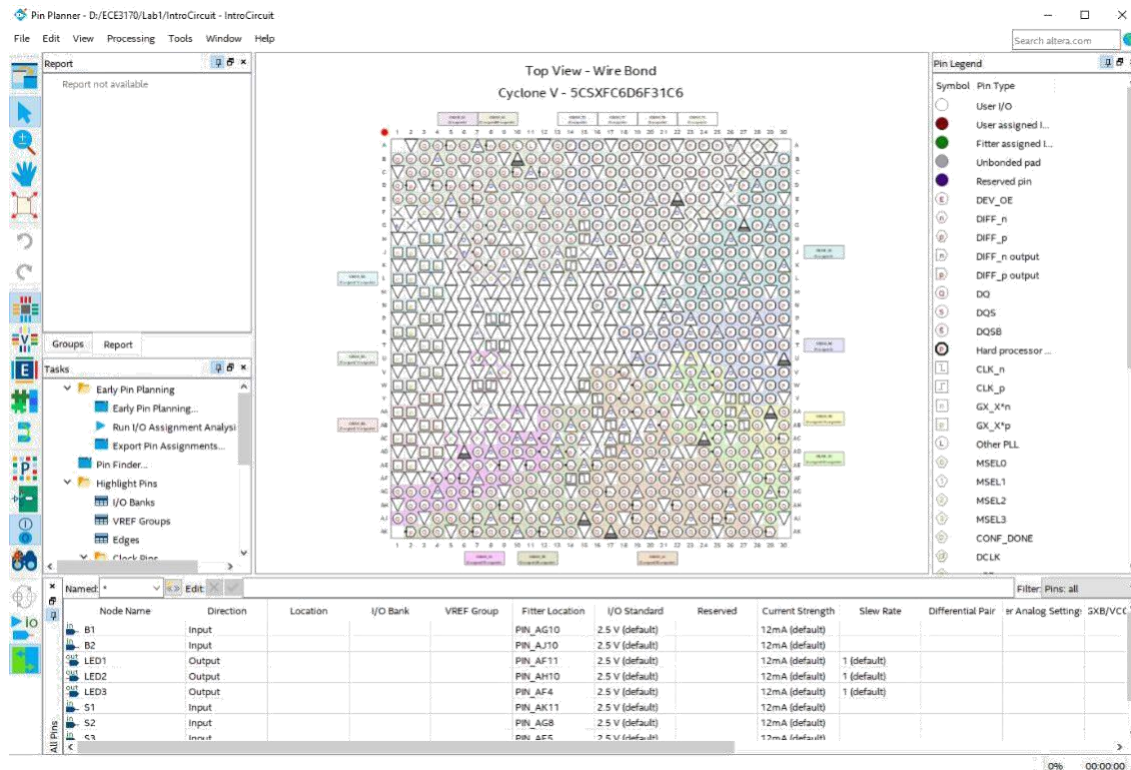
**When Quartus reports a compilation error in a VHDL file, double-clicking the error should take you to where it found the error. Note though that it takes you to where Quartus *noticed* the error, not necessarily where the error occurred.**

## Step 9.

The final step in the design process is to add pin assignments to the design file. We chose input names that correspond to elements of the development board -- the buttons, switches, and LEDs. But Quartus isn't designed to work with just our board, and it does not know where those devices connect to pins on the FPGA. That information is in the manual for the board, and the relevant information will be provided below.

Open the pin planner by selecting **Assignments => Pin Planner** (or find the icon in the toolbar), which brings up the window below. Notice that the I/O pins are listed in the table at the bottom of the Pin Planner.

Since the pins in your design must be assigned to pins on the FPGA, the Fitter stage of compilation made arbitrary choices, displayed in the **Fitter Location** column of the Pin Planner (not shown below). These are not correct, since Quartus has no knowledge of the development board and the usage of FPGA pins relative to switches, LEDs, and pushbuttons that we wish to use.



For each I/O pin in the list, double-click on the **Location** cell (not the **Fitter Location** cell) and enter the pin numbers shown below. Note that you can omit "PIN\_" as you are typing, if you like, since Quartus will fill in the leading characters. Or you can select from a drop-down list, instead of typing. Again, you would have to search the documentation for the board to know that, for example, the LED called LED0 is actually connected to pin AA24 on the FPGA chip.

Node Name	Direction	Location	I/O Bank	VREF Group
B0	Input	PIN_AJ4	3B	B3B_NO
B1	Input	PIN_AK4	3B	B3B_NO
B2	Input	PIN_AA14	3B	B3B_NO
clk	Input	PIN_AC18	4A	B4A_NO
LED0	Output	PIN_AA24	5A	B5A_NO
LED1	Output	PIN_AB23	5A	B5A_NO
LED2	Output	PIN_AC23	4A	B4A_NO
LED3	Output	PIN_AD24	4A	B4A_NO
S0	Input	PIN_AB30	5B	B5B_NO
S1	Input	PIN_Y27	5B	B5B_NO
S2	Input	PIN_AB28	5B	B5B_NO
S3	Input	PIN_AC30	5B	B5B_NO

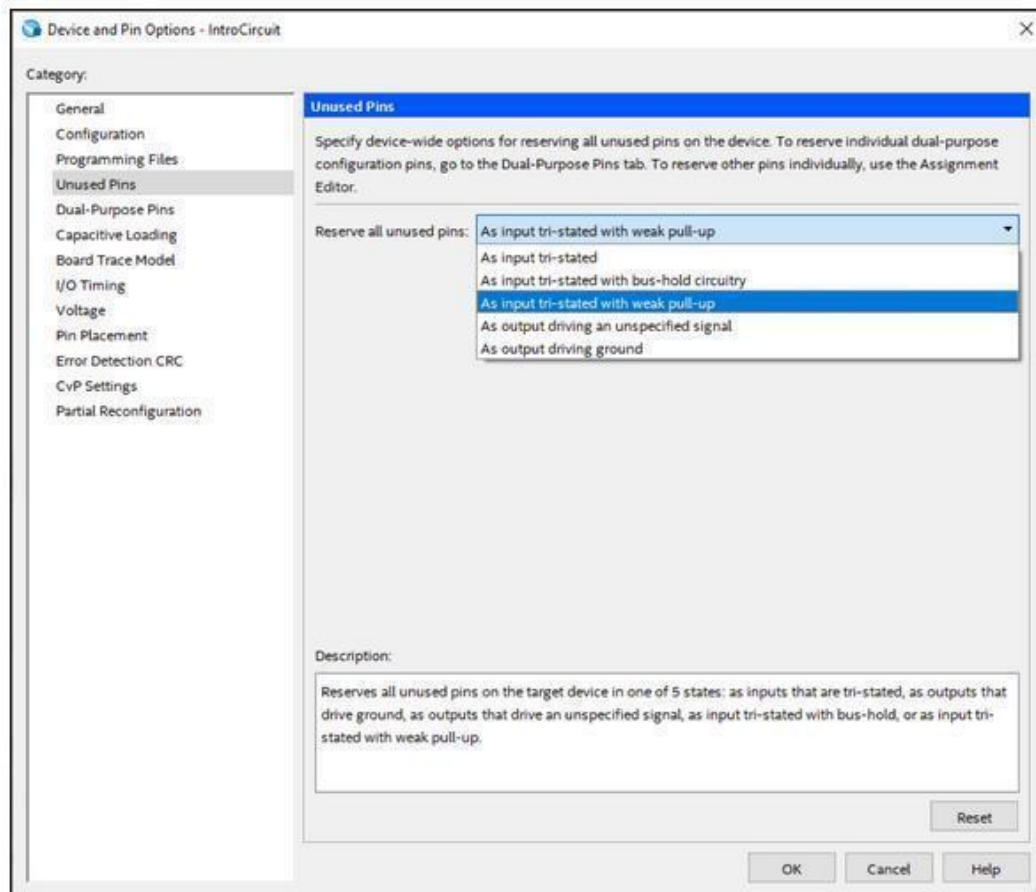
With all of the pin assignments made, close the Pin Planner window. It is very important to note that even though the correct pin numbers have been assigned, the project must be compiled again before these pins will actually be used in any files that can be programmed onto the DE10-Standard board. In this case, we will compile at least one more time after making a change in the next step.

## Step 10.

The following changes to default project settings are critical to the correct and safe operation of designs programmed to the DE10-Standard. We will remind you to change them in the first couple labs, but you need to remember to change them whenever you create a new Quartus project.

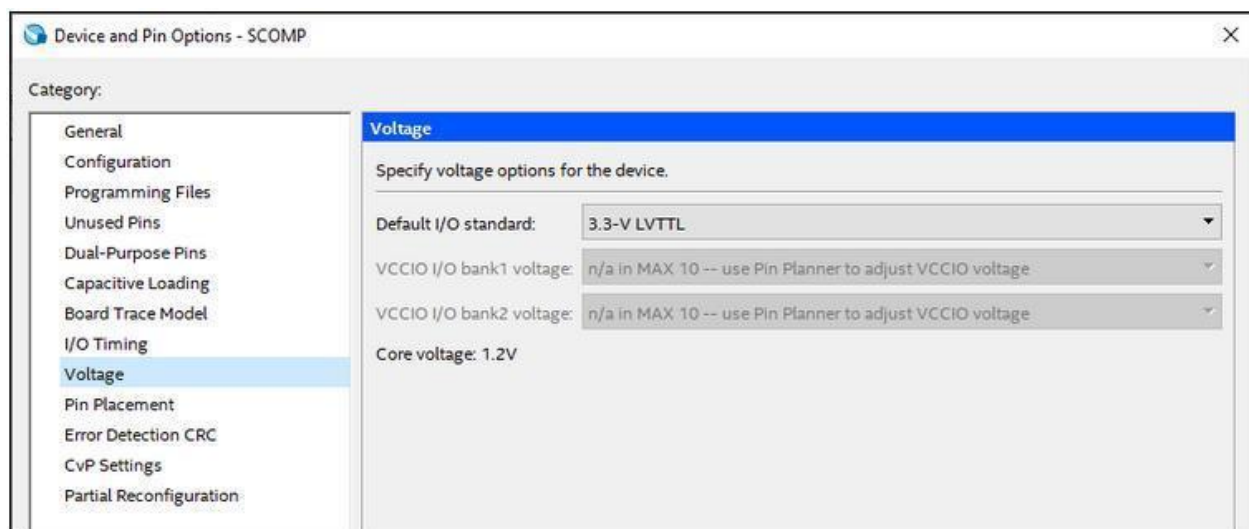
We have just told Quartus what to do with eight of the many pins that are connected to the Cyclone V FPGA chip. But there are many other pins, and if we are not specific, Quartus may use them for intermediate signals, or drive them deliberately high or deliberately low. This could result in strange patterns on the LEDs or other outputs. It could even damage something on the board that isn't meant to be driven.

Fortunately, there is an option to define what to do with all of the pins that are not needed in our design. From the main Quartus menu, select **>Assignments => Device...** to bring up the Device settings window. It is the same dialog that we used earlier to define the target FPGA device. Look for the **Device and Pin Options...** button and click it to bring up the window shown below. Select the **Unused Pins** category on the left, and then **As input tri-stated** on the right. (Not **As input tri-stated with weak pull-ups**, as shown below.)



Without getting too technical, this option forces all of the unused pins on the FPGA to be passive inputs. In normal new design situations, this would not matter, since the board design would typically be done after the FPGA is designed. However, the projects in these laboratory exercises will target your DE10-Standard FPGA board, which has several other ICs and connectors on it in addition to the FPGA. Many of these ICs connect directly to the FPGA I/O pins. To avoid causing erroneous board operation or potential damage to the FPGA or other ICs, it is important to change this setting every time a new project is created. Once complete, press the **OK** button to close this window. Press the **OK** button on the settings window to complete the options change.

In the same Device and Pin Options dialog, go to the Voltage tab, and change the Default I/O Standard to "3.3-V LVTTTL", as shown here:



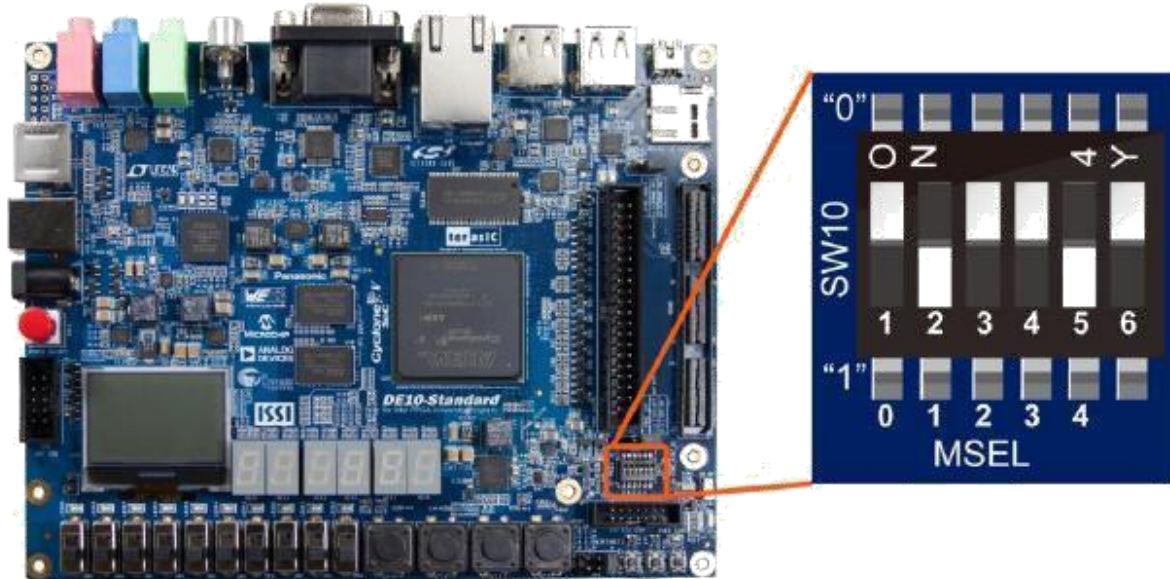
The FPGA on the DE10-Standard can configure its pins to use different voltages, and the majority of devices on the board require 3.3 V. Changing this default settings will save you from needing to individually configure each pin (and from the design not working if you forget to change one).

**Compile your project one more time** (because you just changed some settings that need to be incorporated into the programming file), either from the toolbar button (the triangle) or the **Processing** menu. If you have any errors that you cannot figure out, use online resources to ask instructors or TAs.

## Step 11.

Normally in this course, we will simulate our design and possibly fix some errors. But for this simple project, we will go straight to a hardware test.





For programming of the FPGA using JTAG, the above shown switch positions are required (MSEL[4:0] set to "10010"). **NOTE: These positions will change in later labs when we program the HPS (Arm Processor). They will need to be set appropriately depending on if we are programming the FPGA or HPS.**

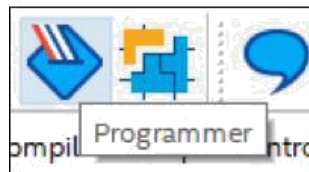
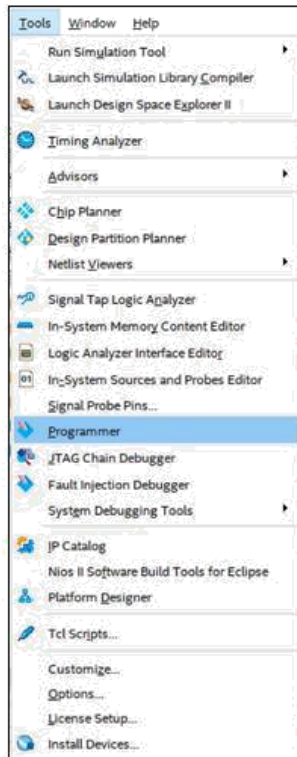
Connect your computer to your target development board. This should require only a USB cable between the computer and the DE10-Standard board, using one of the cables supplied with the board as shown below.



The term "Programming" is used by Quartus, and we will use it as well to minimize confusion. But you should always think of this as "configuring" the FPGA to connect its logic, flip-flops, and other hardware internally. There is no program "running" on the board -- it will be implementing hardware.

Before configuring/programming the board with your design file, open the device setting window by selecting **Assignments => Device...** from the menu. Verify that the correct FPGA device is still selected. If the wrong device is shown, select the correct FPGA and press the **OK** button. This could require you to go back and reassign pin numbers, but should not be necessary if you completed earlier steps with the correct device assignment.

Open the programmer tool, by selecting **Tools => Programmer** from the menu or selecting the icon from the toolbar.



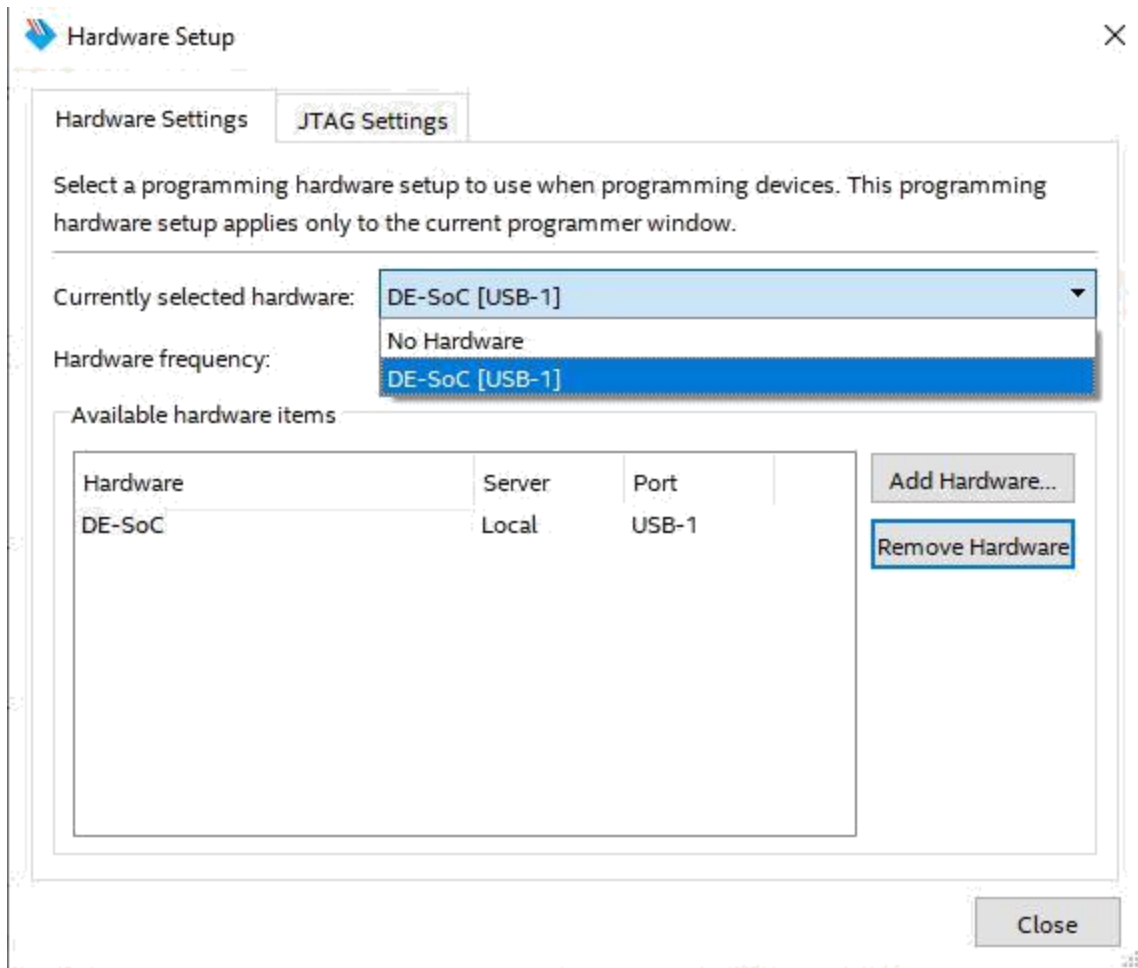
**NOTE: This next section is NOT the same as in ECE 2031.**

You should see a window similar to the one below. When you open the Programmer while working within a project that has been compiled, it should correctly select the file holding the logic (top window pane, left column) and the correct device (second column). It also shows a diagram of the expected connection to the device, which for our case is not the correct connection.

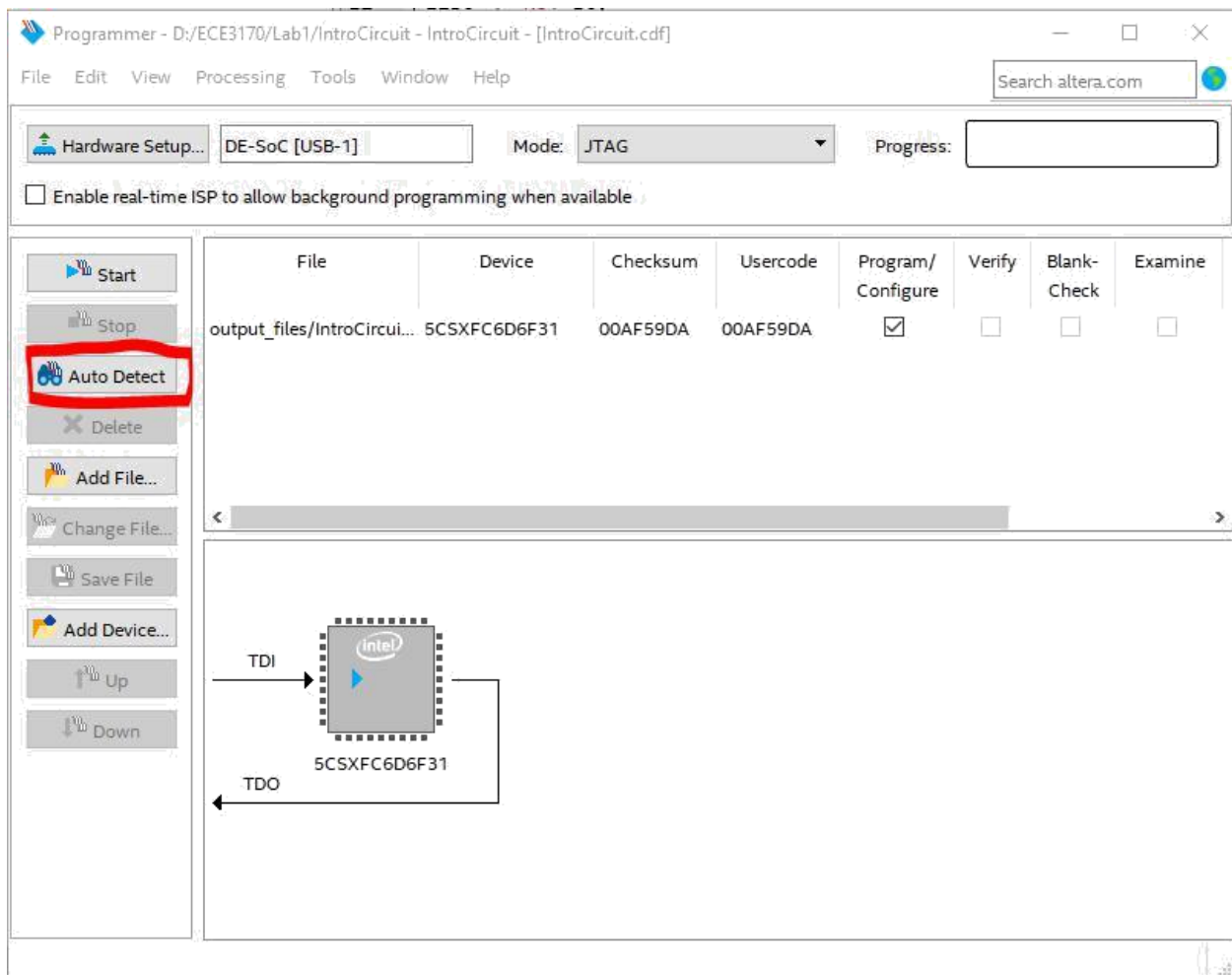
We will program the board using the JTAG chain. This is in volatile memory and will not be stored after a power off of the device.

The **Hardware Setup** refers to the programming hardware between your computer and the chip, including the USB port, the cable, and some interface hardware on the board. If the **Hardware Setup** is listed as **No Hardware**, which is normally the case for first-time use, click on the **Hardware Setup...** button to display the window below.

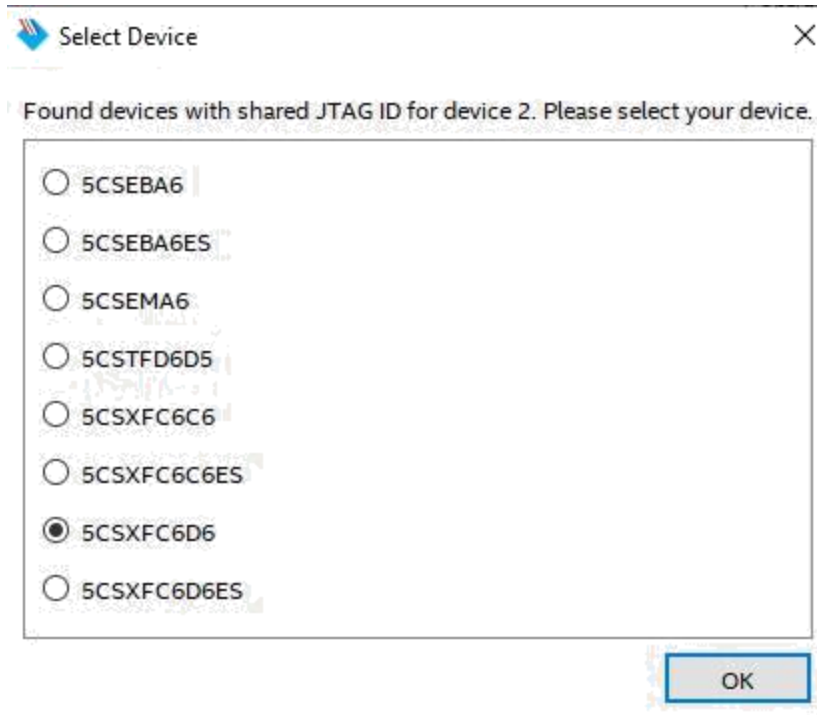




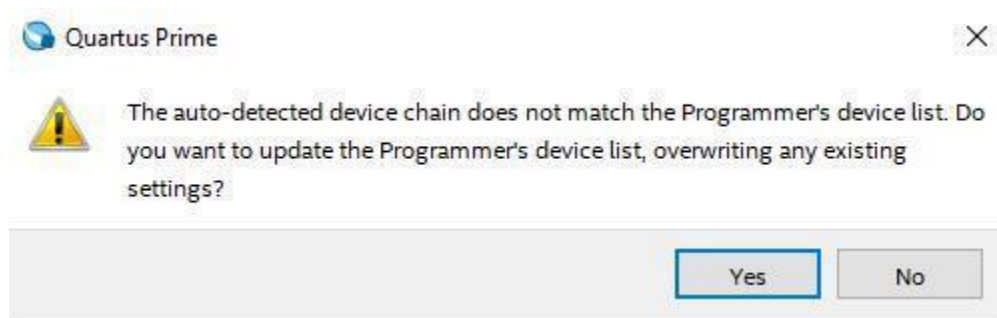
Under the **Currently selected hardware** option, choose **DE-SoC [USB-X]** and close the window. If no programming hardware is listed, make sure that the USB cable is in place, and possibly try the **Add Hardware** button. If it still does not work, go back to the steps taken when Quartus was installed, because those steps included an initial test of the USB Blaster setup. Close the Hardware Setup window after it correctly shows the USB-Blaster as the currently selected hardware.



Select "Auto Detect" shown in the image above.



Select the detected device associated with the board as shown above.

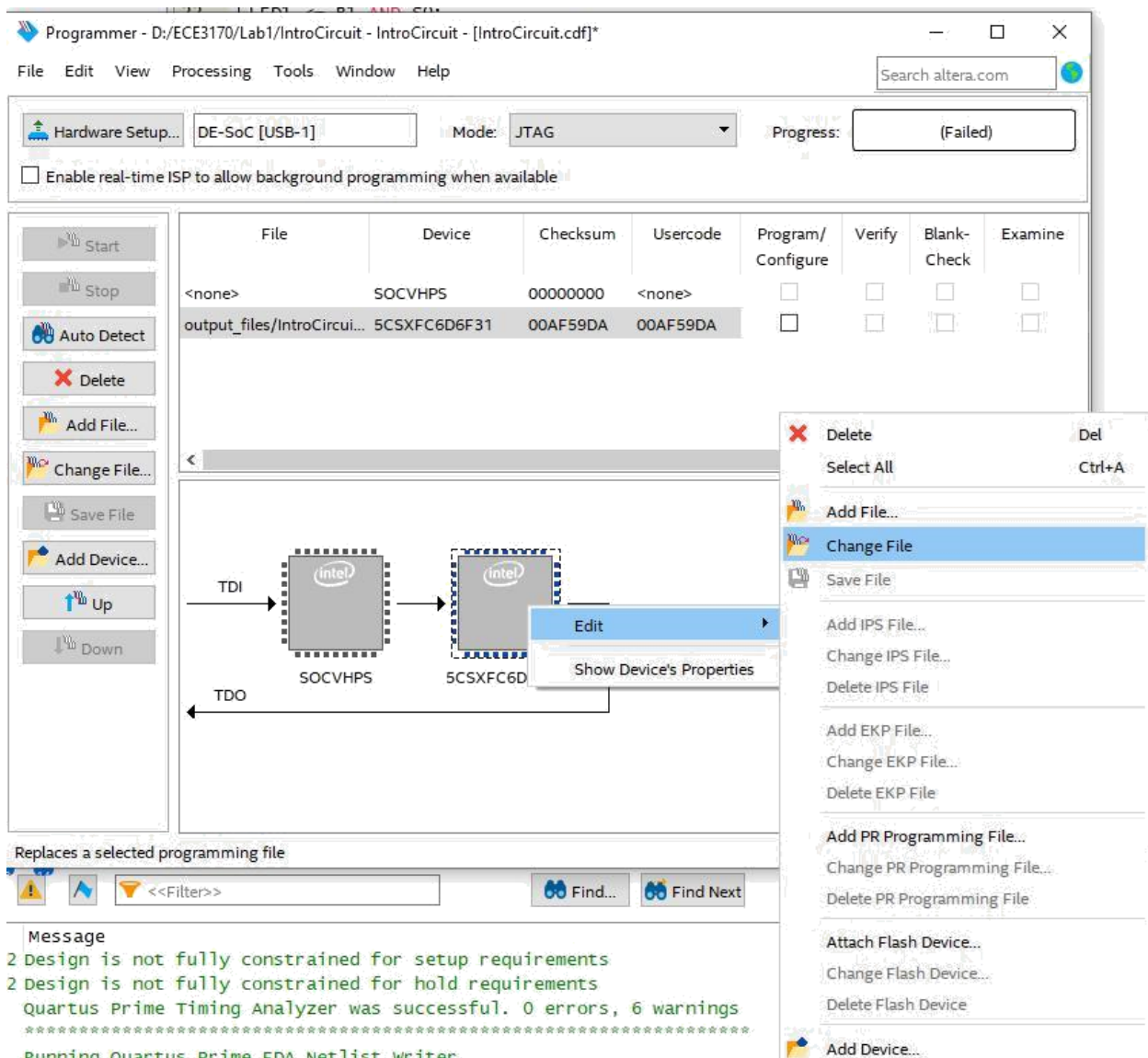


Press "Yes" when you receive the above pop-up.

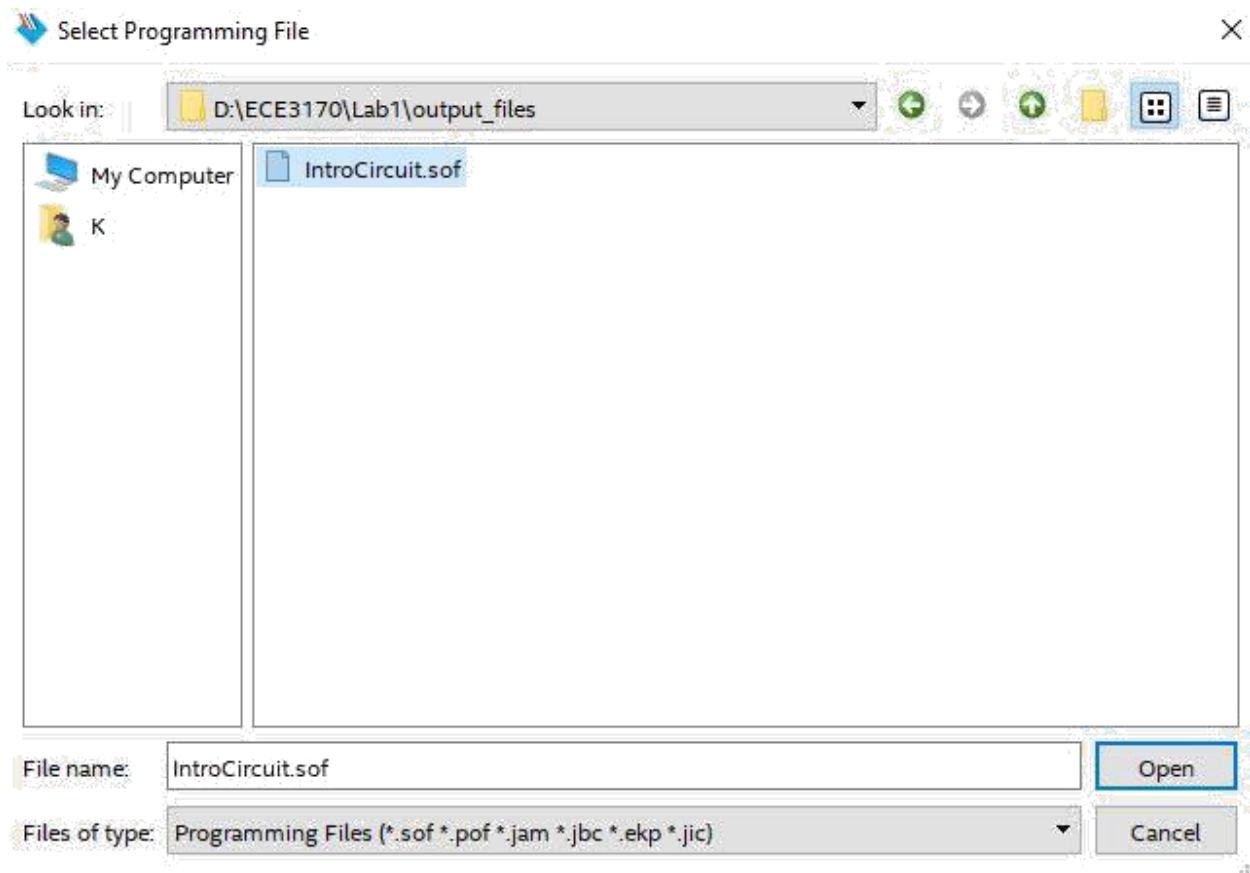
You should now see what is shown below:



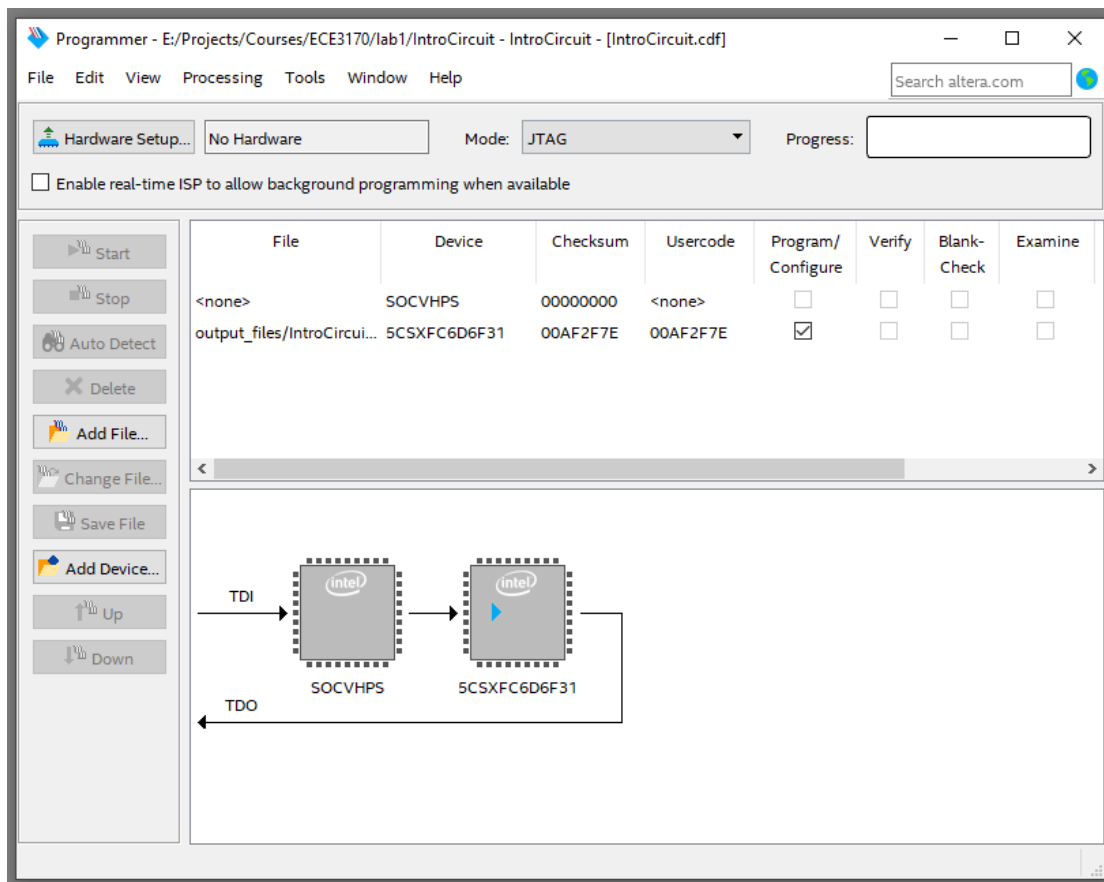
Right click on the FPGA (the chip on the right) device and open the .sof file to be programmed by navigating to **Edit...Change File**



You should see a file IntroCircuit.sof located in the “output\_files” folder. Select this file.



Select the checkbox for Program/Configure on the FPGA as shown below, and then press **Start** to program the FPGA. Take a screenshot for your lab report showing the programmer has successfully programmed the FPGA. (Essentially the below screenshot but with a success instead of "Failed" which is shown now)

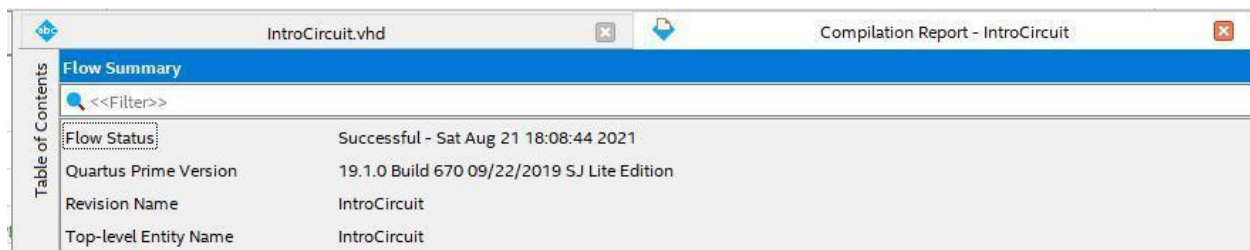


Verify that the LEDs operate as expected with usage of the push buttons and switches. Note that the push-buttons generate a low logic level when pressed (active low).

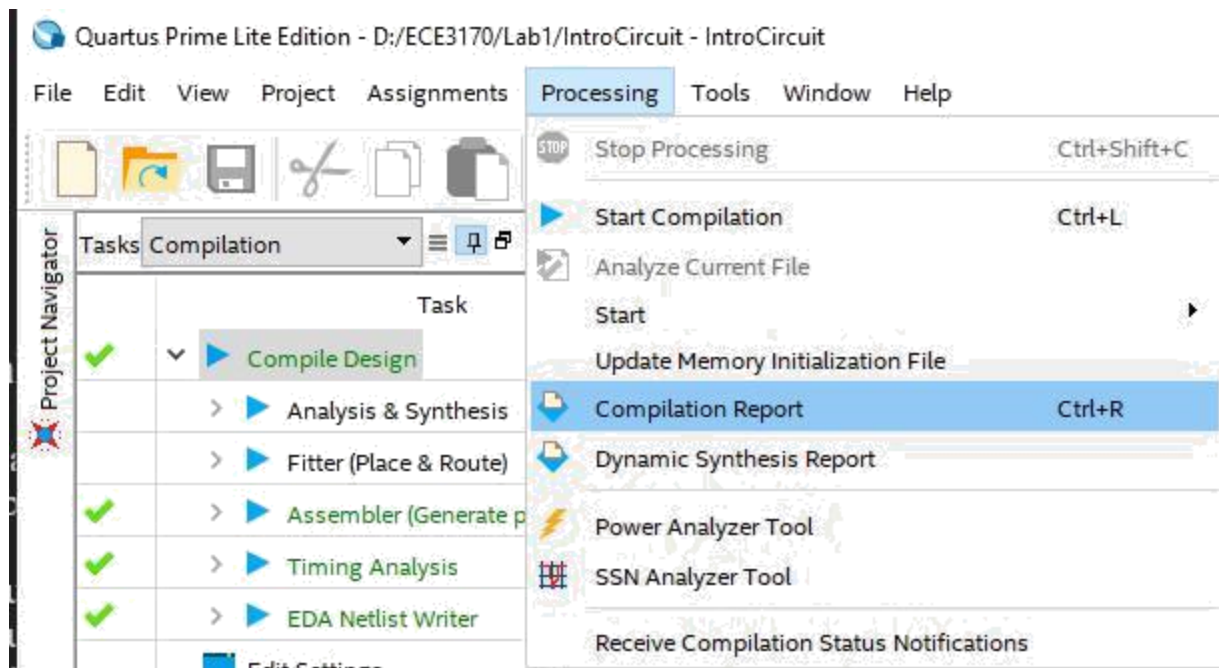
## Examining the compilation report

### Step 1.

After compiling, the compilation report should automatically open in Quartus



If it does not, you can open it by **Processing -> Compilation Report**



## Step 2.

With the Compilation Report open, navigate to the table of contents on the left, and open **Analysis & Synthesis -> Resource Usage Summary**. Take a screenshot of the statistics that appear. This will be required for the lab report.



# Verifying DES in C/C++ and VHDL

In this portion of the lab, you will simulate and verify a VHDL implementation of the Data Encryption Standard (DES). You will be given VHDL code which implements both encryption and decryption, which you will then simulate. A sample testbench is also provided to help you set up the simulation. After simulating the sample testbench, you will be asked to modify it to check all your test cases. To be able to do an accurate comparison, you are given a correctly functioning DES C code. Please **type** your answers to the questions in this lab into a lab report.

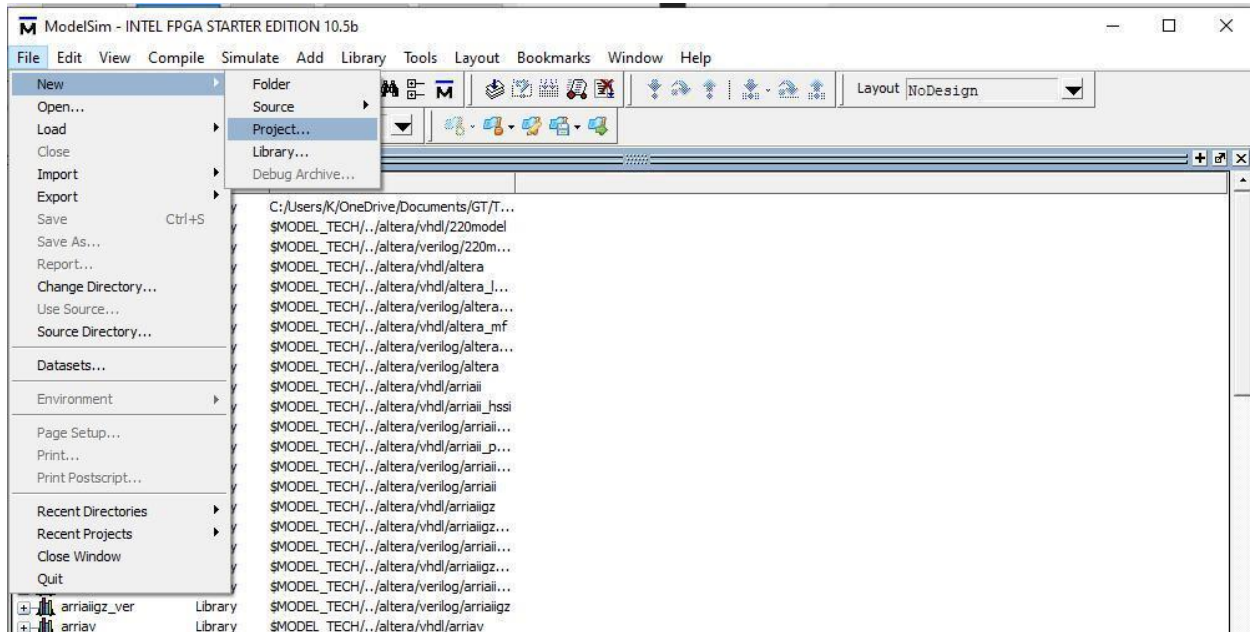
When programming in any language, it is useful to debug, test, or simulate your code to verify its functionality. When programming in VHDL, the convention is to have functional VHDL code and a testbench which tests the code. In this section, you will simulate the provided testbench (des\_cipher\_top\_tb.vhd) with the given VHDL code in ModelSim.

To better understand the structure of the VHDL code, you are given the hierarchy of the provided VHDL files below:

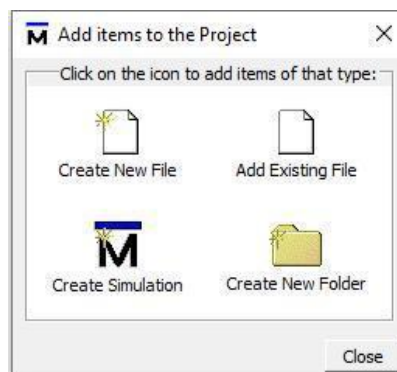
```
des_cipher_top_tb.vhd    -- testbench
  o des_cipher_top.vhd -- top level
    ■ des_top.vhd
      block_top.vhd
        .
        .   add_key.vhd
        .
        .   add_left.vhd
        .
        .   e_expansion_function.vhd
        .
        .   p_box.vhd
        .
        .   s_box.vhd
            .
            .   s1_box.vhd
            .
            .   s2_box.vhd
            .
            .   s3_box.vhd
            .
            .   s4_box.vhd
            .
            .   s5_box.vhd
            .
            .   s6_box.vhd
            .
            .   s7_box.vhd
            .
            .   s8_box.vhd
        .
        .
        .   key_schedule.vhd
```

# Simulating DES in ModelSim

1. Open Model-Sim
2. Create a new project by selecting **File -> New -> Project...**

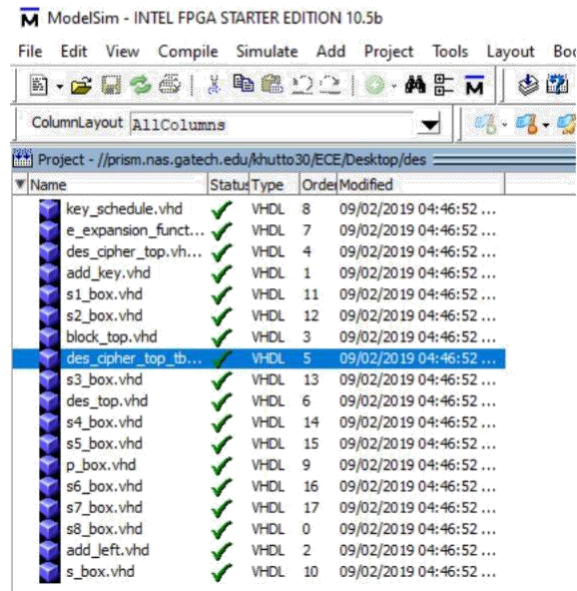
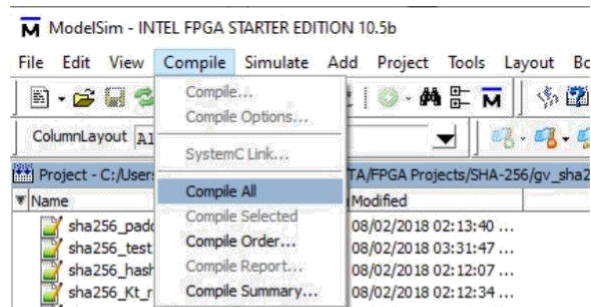


3. Name the new project “DES” and create the project.
4. Select “Add Existing File”



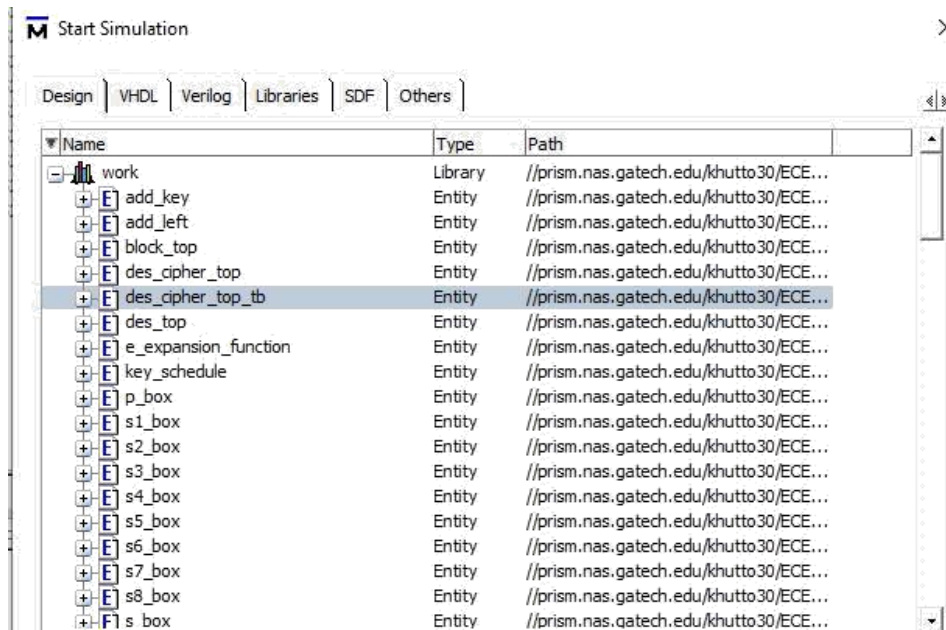
5. Browse to the location you saved the provided VHDL files and add all the VHDL files.
6. Compile the project by selecting **Compile -> Compile All**. Compile All works in a predefined order on the files. Because some of the VHDL files have dependencies on other VHDL files, depending on the default compilation some of the files may fail to compile the first time. Simply **Compile All** again until all files successfully compile.

You can also change the compile order with **Compile Order...** if you want, but it is probably quicker to just compile twice.

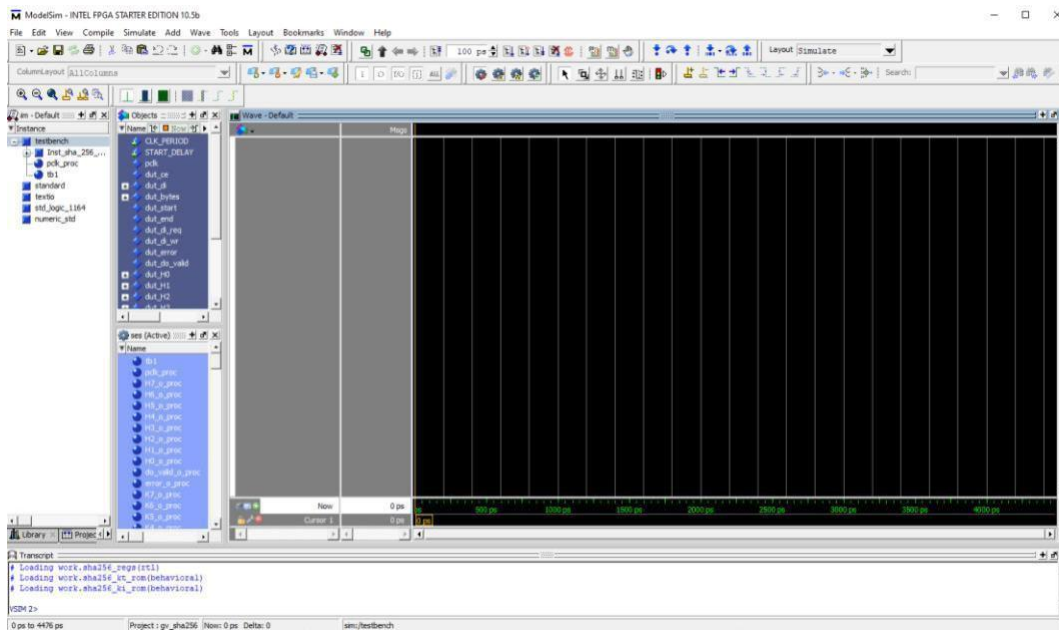


7. Start a simulation by selecting **Simulate -> Start Simulation...**

8. In the Start Simulation window, make sure you are on the design tab, expand the work library, choose the testbench for this code named: "des\_cipher\_top\_tb", and click OK

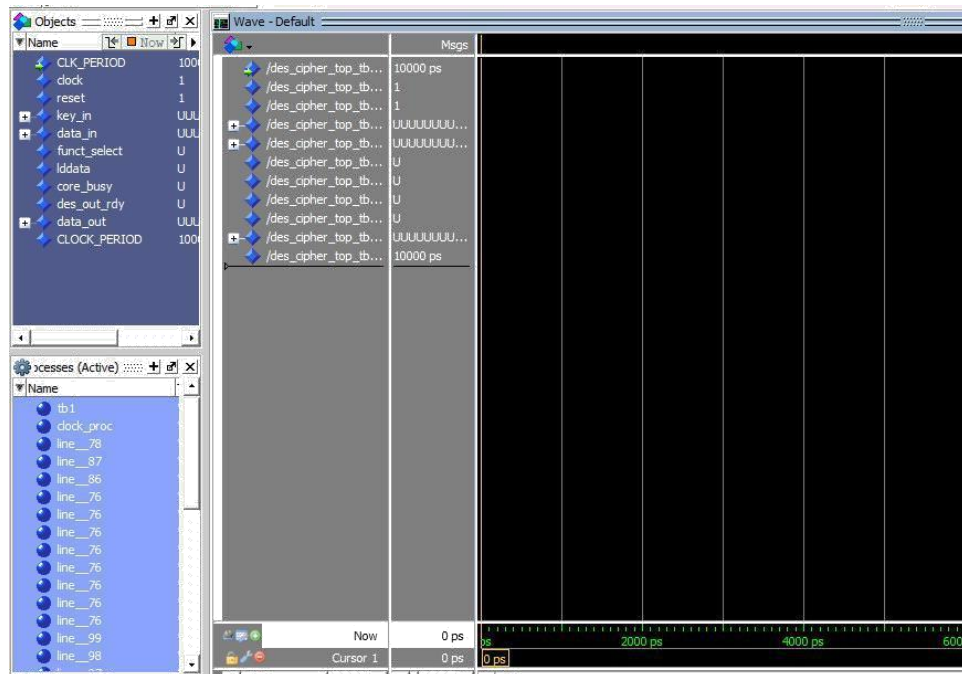


9. ModelSim will change view into simulation mode and a couple of other windows show up. You should see the below screen. If you do not see the black box to the right, select **View -> Wave**.



10. Our next step is to add some signals of interest to a wave window to monitor their changes as simulation proceeds

11. Now let us add our signals of interest to the wave window to monitor their changes as simulation proceeds. Select all the signals in the dark blue "Objects" box. Drag these into the gray area of the "Wave" window.



12. Our final step is to run the simulation for a specific time. For this testbench, running the simulation for 550 ns should be enough. To do so, type **"run 550 ns"** in the command line of the "Transcript" window.

13. Navigating back to the "Wave" window will now show you the result of the simulation for all the signals that we added. To better read the values, select all the signals and right-click, then change the Radix to Hexadecimal. Also, towards the bottom left of the signals pane (to the left of where it says "Now"), there is a blue button that has a description of "Toggle leaf names <-> full names" if you hover the mouse over it. Click on that button to show the signal names only without the hierarchy.

Look through the wave window and try to understand how the signals values are changing with respect to the simulation time. Specifically, look for the output signal that show the encrypted/decrypted value.

14. Now that we have run the simulation, make sure that you set the zoom of the wave window to a full view. To do so, right-click anywhere in the wave window and click on Zoom Full. Export an image of your simulated waveform. To do so, click on File --> Export --> Image... and save it as an image. Include this image in your submission.

15. Before ending the simulation, open the transcript window and verify that no reports are generated by the testbench indicating a failure of any of the test cases.

# Testing more plaintext and ciphertext cases

Now we will use the provided des.c code to test various plaintexts, ciphertexts, and keys, and then modify our test\_bench to confirm identical results. Inside the "lab1" directory, you can find a folder named "des\_c". This folder contains a functioning version of DES in the C programming language. We will modify and compile the provided code to generate DES output results.

1. Choose 5 independent keys you will use for testing. The keys may not be any of the known weak cases for DES (Please see [this](#) for more information). Write these in a file **Key.txt**
2. Choose 5 independent plaintext values you will use for testing. Write these in a file **Plaintextin.txt**
3. Modify the "main" function in the provided des C code to test your chosen keys and plaintexts.

This can be either hardcoded or use file I/O. For each key, test all five of your plaintext values. Store these in five separate text files. You can either manually write the files from the C codes terminal output or have your modified C code automatically create them.

**Ciphertextout1.txt** (5 ciphertext from using key1 and the plaintextin.txt)  
**Ciphertextout2.txt** (5 ciphertext from using key2 and the plaintextin.txt)  
**Ciphertextout3.txt** (5 ciphertext from using key3 and the plaintextin.txt)  
**Ciphertextout4.txt** (5 ciphertext from using key4 and the plaintextin.txt)  
**Ciphertextout5.txt** (5 ciphertext from using key5 and the plaintextin.txt)

4. Choose one Ciphertext value from each of your ciphertextoutX.txt files. Place these into a new file **Ciphertextin.txt**. Again, this can be manual or automatic by your code.
5. Now continue to modify the C code to decrypt the values you placed in **Ciphertextin.txt** with each of your chosen keys. Store these in five separate text files. You can either manually write the files from the C codes terminal output or have your modified C code automatically create them.

**Plaintextout1.txt** (5 plaintext from using key1 and the ciphertextin.txt)  
**Plaintextout2.txt** (5 plaintext from using key2 and the ciphertextin.txt)  
**Plaintextout3.txt** (5 plaintext from using key3 and the ciphertextin.txt)  
**Plaintextout4.txt** (5 plaintext from using key4 and the ciphertextin.txt)  
**Plaintextout5.txt** (5 plaintext from using key5 and the ciphertextin.txt)

Additional info on the required C Code:

Your C code is not required to write ciphertextout.txt and plaintextout.txt. If you choose not to write the results to a file, your program must print the results in the terminal. Please make sure your print statements to the terminal can be understood by the TA.

You must provide a rationale for your chosen five plaintext cases. The five cases must be somewhat unrelated to each other; e.g., you may not choose numeric sequences (e.g., adding one to each prior case).

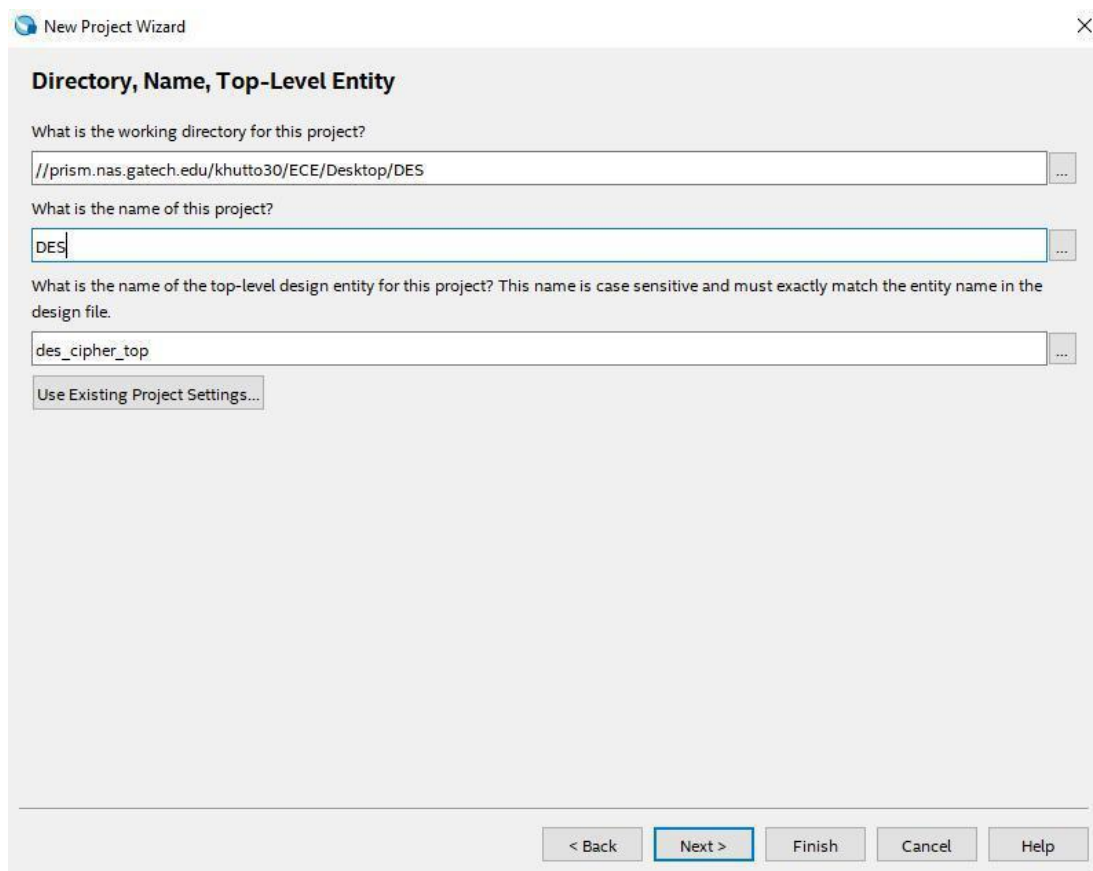
# VHDL Modification

Modify the provided DES VHDL test\_bench to test your 5 keys for the 5 plaintexts. Also test the 5 ciphertexts you placed in **Ciphertextin.txt** with your 5 keys. This is a total of 25 encryptions and 25 decryptions. You may hardcode this into the test\_bench or use file I/O. Once you are done modifying the testbench, save it, recompile it and resimulate the design.

The following link may be helpful if you want to utilize file I/O. [VHDL Example Code of File IO \(nandland.com\)](http://nandland.com/VHDL-Example-Code-of-File-I/O)

## DES VHDL Synthesis

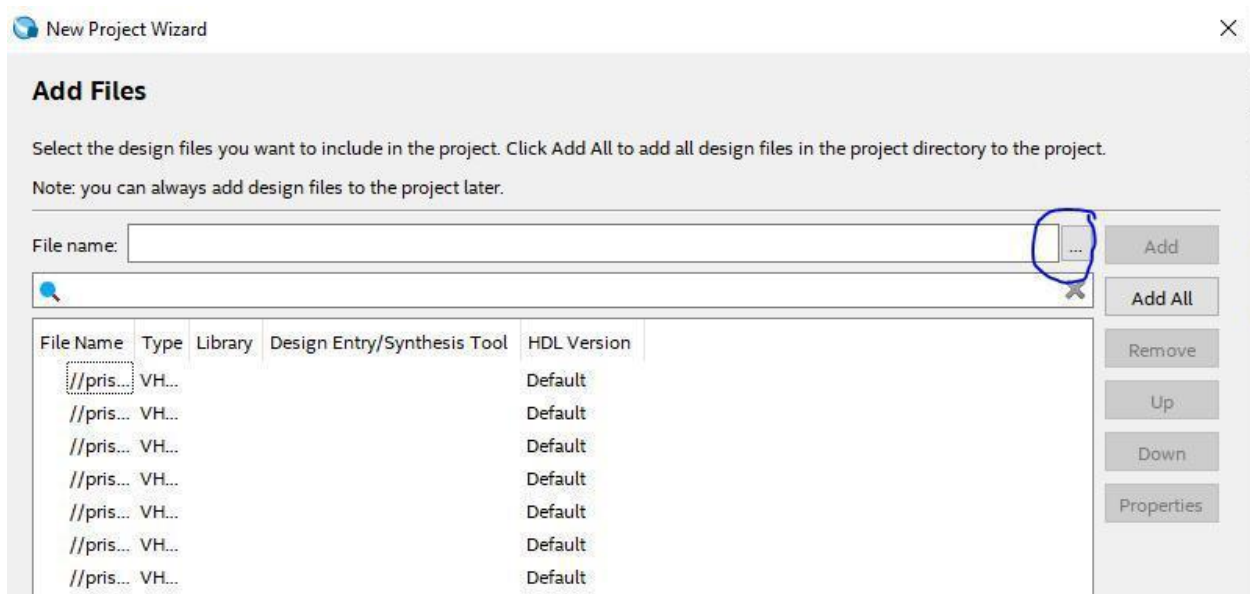
1. In Quartus, start a new project. Name the project "DES", with the top-level entity as "des\_cipher\_top" as shown below.



3. Then, select **Next** to advance to a window which asks you to choose between an "Empty project" or a "Project template." Choose the "Empty project" and select **Next** to advance.



4. On the below screen, press the button circled in blue, and the provided des VHDL files, **except don't add the test bench des\_cipher\_top\_tb**, to the project and select **Next** to advance.



5. As before, select the appropriate FPGA device and select **Next**.

6. As before, select **ModelSim-Altera** and **VHDL** and then **Finish** after verifying the project options are correct.

7. Compile the project. Again take a screenshot of the **Resource Utilization Summary**.

In this lab we will not be loading the DES VHDL onto the Cyclone V SoC. In a future lab we will explore implementing the VHDL on the FPGA with an interface through the SoC's processor. Take the time to consider how you might need to alter the provided DES VHDL to enable this. Specifically, how can we lower the number of I/O pins required to implement this solution.



# Lab Report

Include in the report your name and GTID. Write a brief explanation of the modifications that you did to the main function and to the VHDL testbench. Additionally include your reasoning for choosing the DES keys and plaintexts you tested, and a brief description of how to run your C code.

The report should include the following:

1. Screenshot showing the successful programming of the “IntroCircuit” onto the FPGA.
2. Screenshot showing the **Resource Usage Summary** for the “IntroCircuit”.
3. ModelSim screenshot showing the unmodified DES test bench waveform.
4. Screenshot showing the **Resource Usage Summary** for the DES circuit.
5. Modified C code with annotations of what you changed.
6. Modified DES test bench.
7. ModelSim screenshot showing one successful encryption and decryption from the values you chose for the test bench modification.
8. Plaintext and ciphertext files you created:

Key.txt (5 keys)

Plaintextin.txt (5 plaintext test cases)

Ciphertextout1.txt (5 ciphertext from using key1 and the plaintextin.txt)

Ciphertextout2.txt (5 ciphertext from using key2 and the plaintextin.txt)

Ciphertextout3.txt (5 ciphertext from using key3 and the plaintextin.txt)

Ciphertextout4.txt (5 ciphertext from using key4 and the plaintextin.txt)

Ciphertextout5.txt (5 ciphertext from using key5 and the plaintextin.txt)

Ciphertextin.txt (5 ciphertext test cases, please pick one ciphertext output from each of the ciphertextout.txt (ciphertextout1.txt, ciphertextout2.txt...ciphertextout5.txt)

Plaintextout1.txt (5 plaintext from using key1 and the ciphertextin.txt)

Plaintextout2.txt (5 plaintext from using key2 and the ciphertextin.txt)

Plaintextout3.txt (5 plaintext from using key3 and the ciphertextin.txt)

Plaintextout4.txt (5 plaintext from using key4 and the ciphertextin.txt)

Plaintextout5.txt (5 plaintext from using key5 and the ciphertextin.txt)

**Please compress all files into a single folder and submit on Canvas.**