

CSC/EEE120
Introduction to Quartus Software & FPGA's
Reference Guide

Table of Contents

INTRODUCTION TO FPGAs:	1
INSTALLING QUARTUS (optional):	2
INTRODUCTION TO QUARTUS SOFTWARE:	6
Starting a New Project:.....	6
Setting the Default Pin Voltage:.....	11
Creating a Circuit Schematic:.....	13
Simulating with Waveforms:.....	17
Assigning Switches and LEDs:.....	24
Downloading into FPGA:.....	31
Writing Verilog Code:.....	33
Creating Device Symbols.....	38

INTRODUCTION TO FPGAs:

In our class, we will be using Intel's DE10-Lite Development and Education FPGA board. An FPGA (Field-Programmable Gate Array) is an integrated circuit that can be configured by the user after manufacturing. Unlike traditional processors, which use fixed-architecture chips designed for sequential processing, FPGAs consist of numerous small logic blocks and configurable interconnects that can be changed to form complex circuits, enabling parallel processing and high performance. When you load a new design, the FPGA's configuration memory is updated based on your code, causing changes to which logic paths are enabled, how signals are routed, and what functions are implemented. The physical layout of the chip doesn't change – the same transistors and interconnects are reused, but their logical function is rewritten. They are called “field-programmable” because you can reconfigure them in the field without replacing any of the chips.

To use your FPGA, you will use Quartus Prime Lite Software to configure (program) the FPGA on a computer and then download it into a FPGA. While the software is available on our classroom computers, you may also install it on your personal computer by downloading from: <https://www.altera.com/downloads/fpga-development-tools/quartus-prime-lite-edition-design-software-version-20-1-1-windows>, (get **Version 20.1.1 for Windows**). This is NOT the newest version of the software available, but some of the built-in features we will use have been discontinued in the newer versions. Note: the software will not work on Apple computers.

INSTALLING QUARTUS (optional):

Note: **Please make sure the hardware board DE-10 Lite USB hardware is NOT plugged in yet as it will install faulty drivers by default.** When you go to the link given above and scroll down a little, you will see:

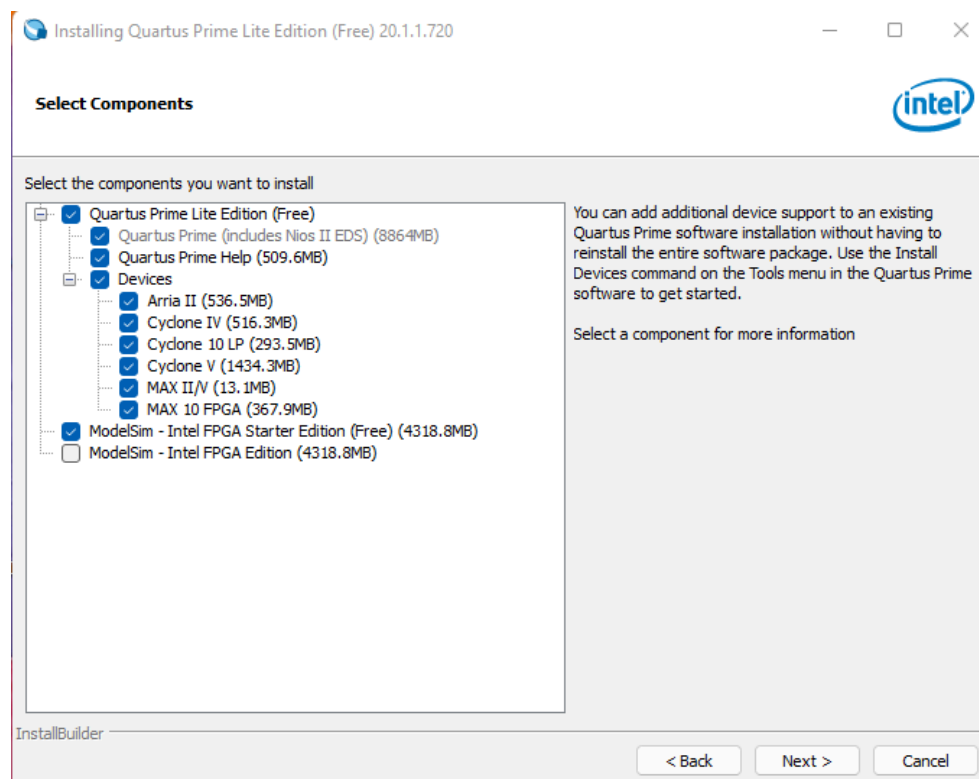
The screenshot shows the Quartus Prime Lite download interface. At the top, there are four dropdown menus: 'Software Type' (set to 'FPGA Development Tools'), 'Software Package' (set to 'Quartus Prime Lite'), 'Version' (set to '20.1.1'), and 'Operating System' (set to 'Windows'). These four dropdowns are circled in red. Below them is a red-bordered box with the text 'Verify you are choosing the correct version.' Below this is a section titled 'Multiple Download' with a 'Download and Install Instructions' link. Under 'Multiple Download', there is a section for 'Quartus Prime Lite Edition Software (Device support included) (Quartus-lite-20.1.1.720-windows.tar)' with details: 'Size: 5.9 GB', 'Version: 20.1.1', and a 'Download' button. A red arrow points to the 'Download' button, and a red-bordered box with the text 'Click to Download' is placed next to it. Below this, there is a section titled 'Download and install instructions:' with a list of three steps: 1. Download the software .tar file and the appropriate device support files. 2. Extract the files into the same temporary directory. 3. Run the setup.bat file. A red curly bracket groups these steps, and a red-bordered box with the text 'Installation Instructions' is placed next to it. At the bottom left of this section is a link 'Read FPGA Software Installation FAQ'.

The download might take a long time, and the program takes up quite a large amount of space, approximately 6 GB. (In fact, the file is too large for Canvas, which is why you must go to the source to download it.) The download is for a single file with a “.tar” extension, which means it is in a compressed format, and the needed files must first be extracted from it.

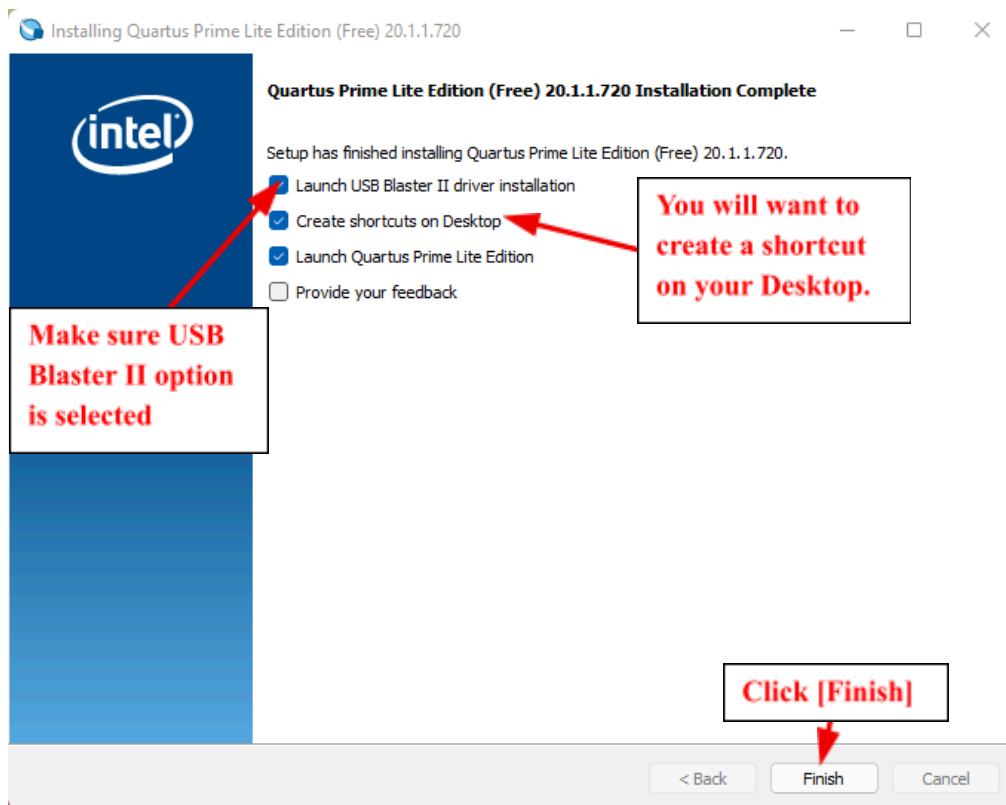
When the downloaded compressed file is then extracted, there will be a new folder called “Quartus-lite-20.1.1.720-windows”, which contain another subfolder called “components” and two files: “readme.txt” and “setup.bat”. To start the installation, double-click on the “setup.bat” file. Once it starts running, the following window appears:



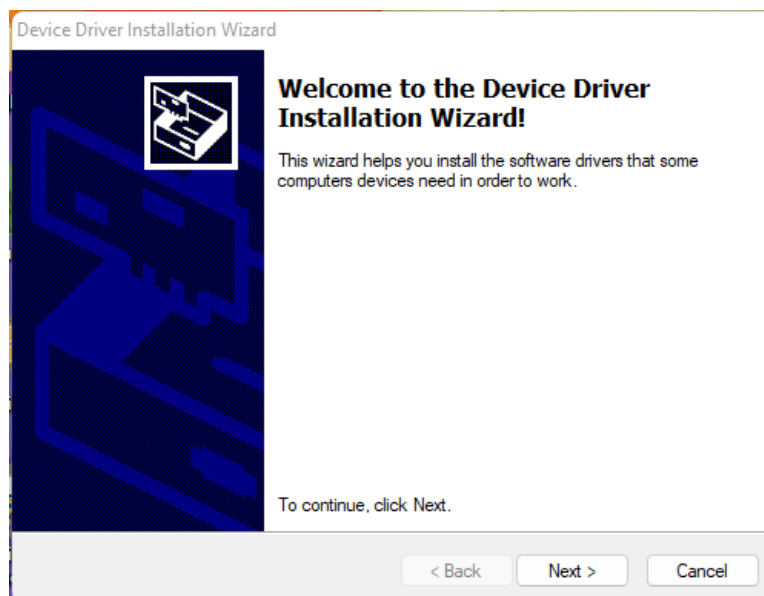
There will be several screens that you must go through, agreeing to the terms and clicking on [Next >] as the installation advances. Depending on your computer, the process may take well over an hour. At one of the screens, you will choose the components to install. Please make sure the following are selected:



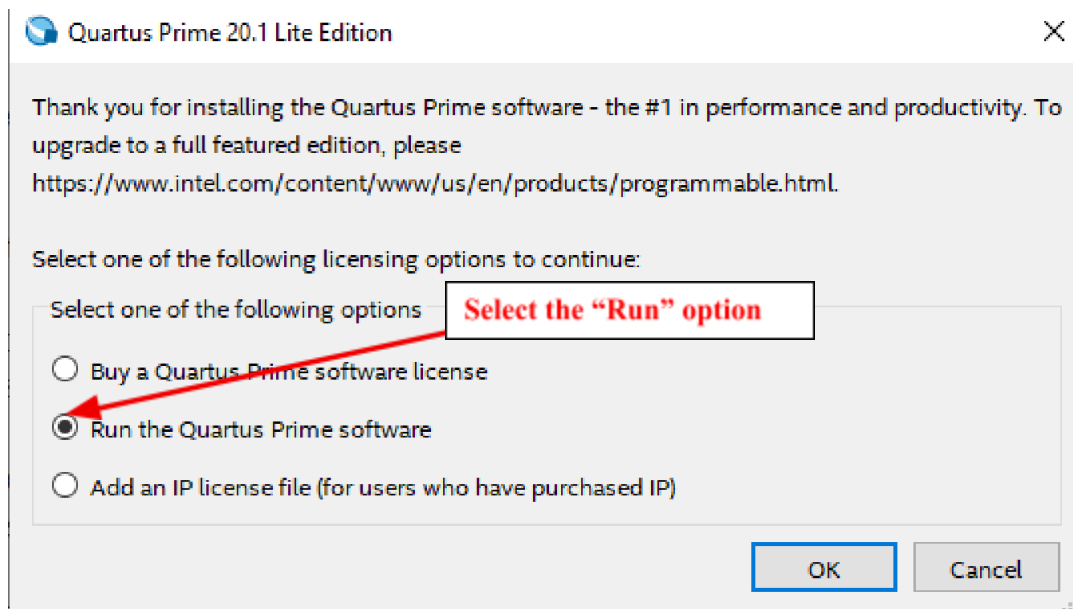
When it is almost finished installing, you will see the screen:



Make sure that the options of “Launch USB Blaster II driver installation” and “Create shortcuts on Desktop” are selected. The other choices depend on your preferences. After clicking on [Finish], it will start the installation of the USB driver, which goes fairly quickly:



Once finish the installation process, you *might* get the following screen:



Now you are done with the software installation and ready to use the Quartus Prime Lite software to program your FPGA.

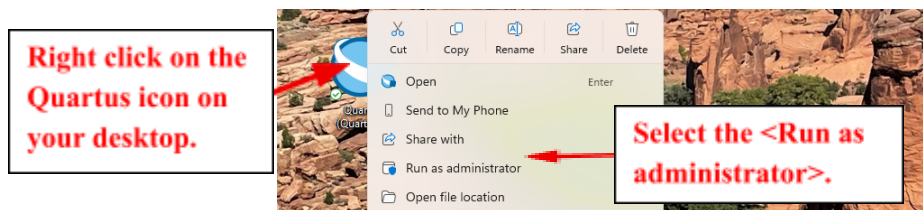
INTRODUCTION TO QUARTUS SOFTWARE:

Your FPGA is a very powerful device and is often used in industry. However, this is an introductory course in digital design, so we will only cover a few of the basic functions you can implement. The following instructions are meant to be a reference to perform most of the functions we will be using the Quartus software for in our class.

Starting a New Project:

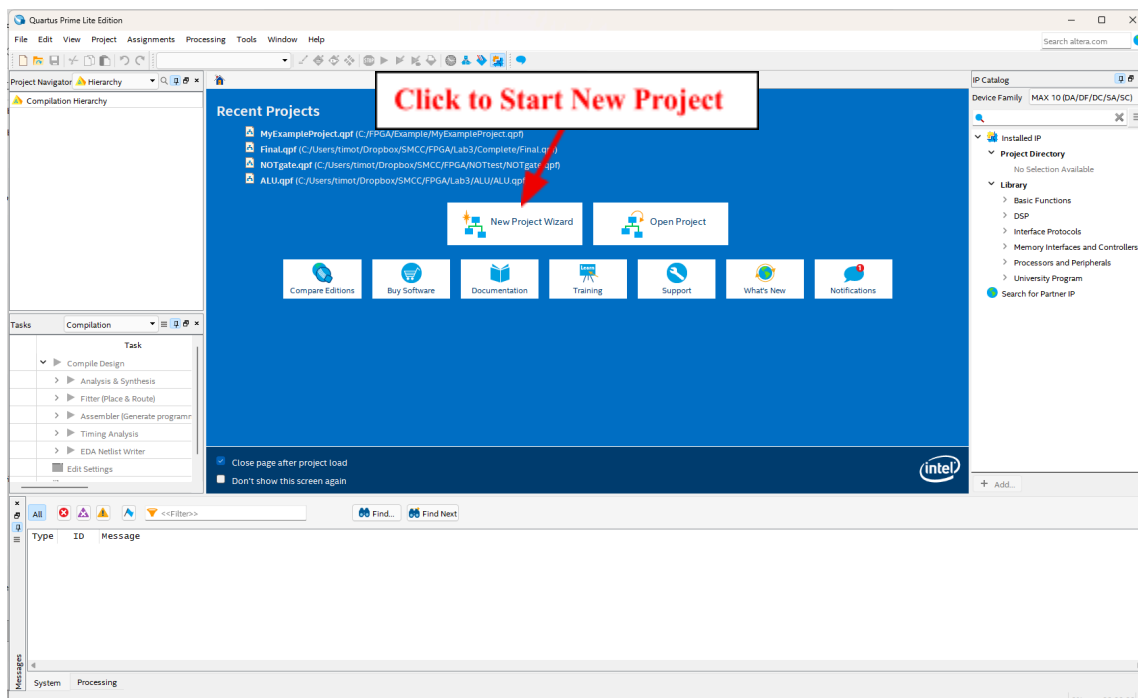
In general, each project creates numerous files with multiple subdirectories. It is highly recommended that you place each project within its own folder. As shown below, we will create a new project called “MyExampleProject” and place it in the folder C:\FPGA\Example.

When you start the Quartus software, you will want to “Run as Administrator.”

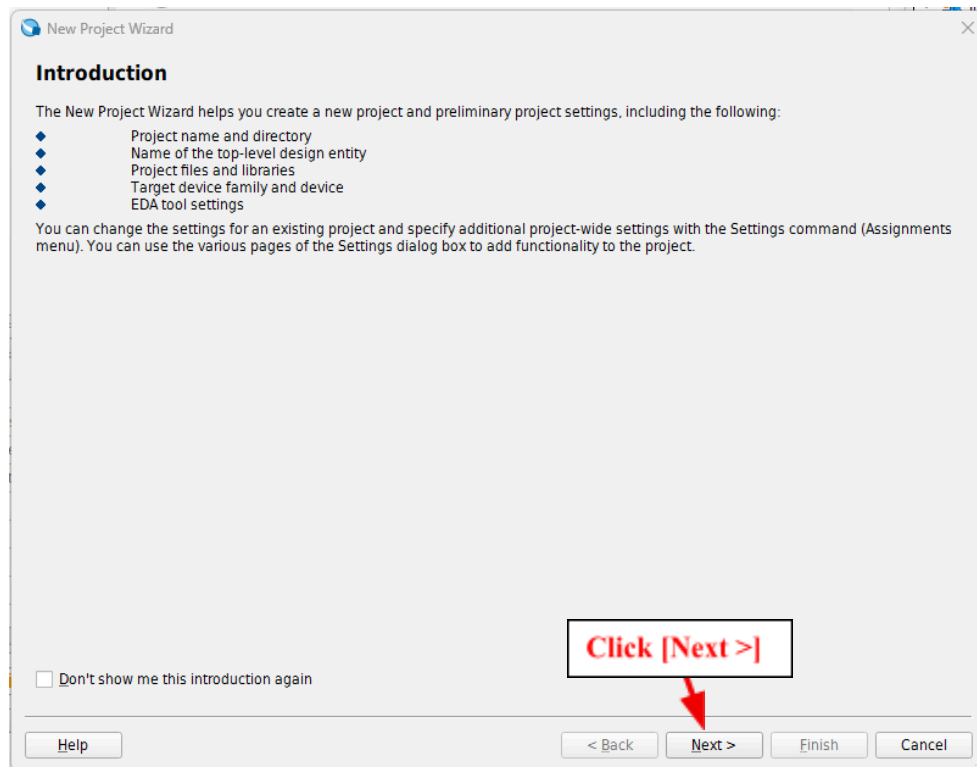


If you do not start Quartus as an administrator, when you “Finish” the New Project creation as described below, you are likely to encounter the error message of “Can’t open project – you do not have permission to write to all the files or create new files in the project’s database directory.” If this happens, you will want to close Quartus and restart it as an administrator.

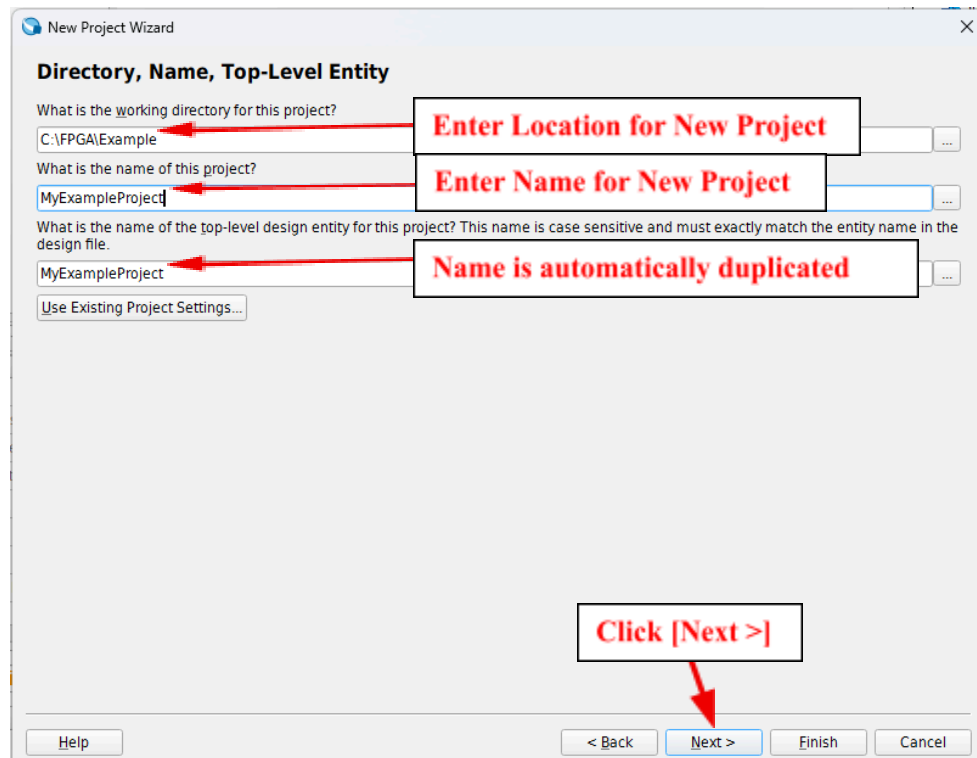
When you first start Quartus Prime 20.1.1 you will see a screen similar to:



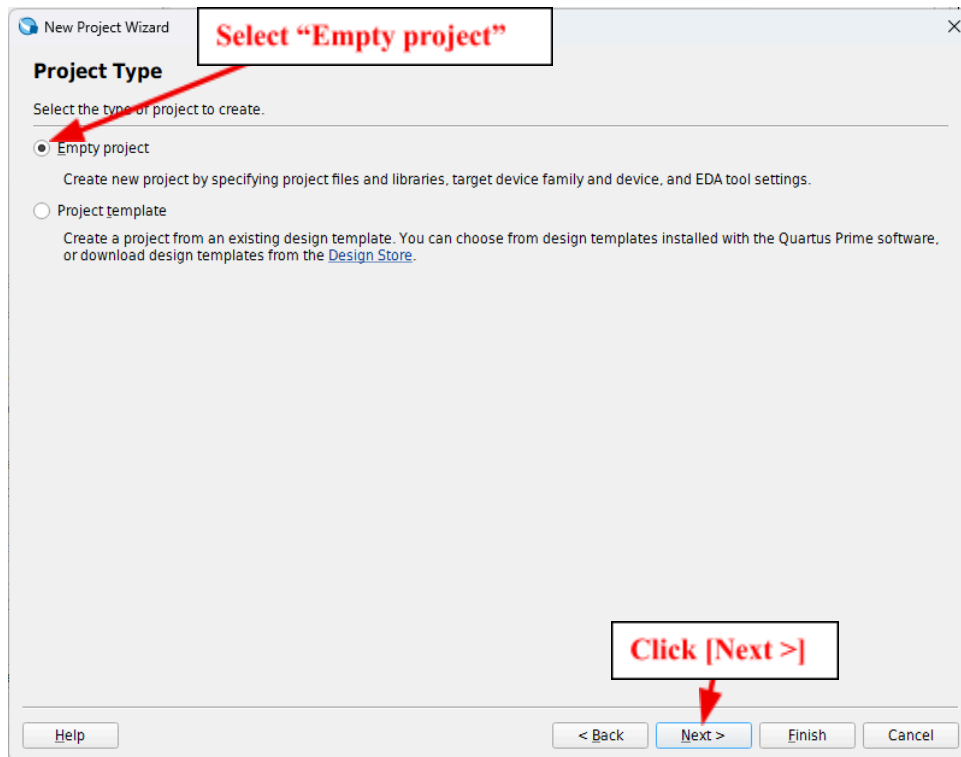
Next, you will get an “Introduction” window, for which you don’t need to do anything.



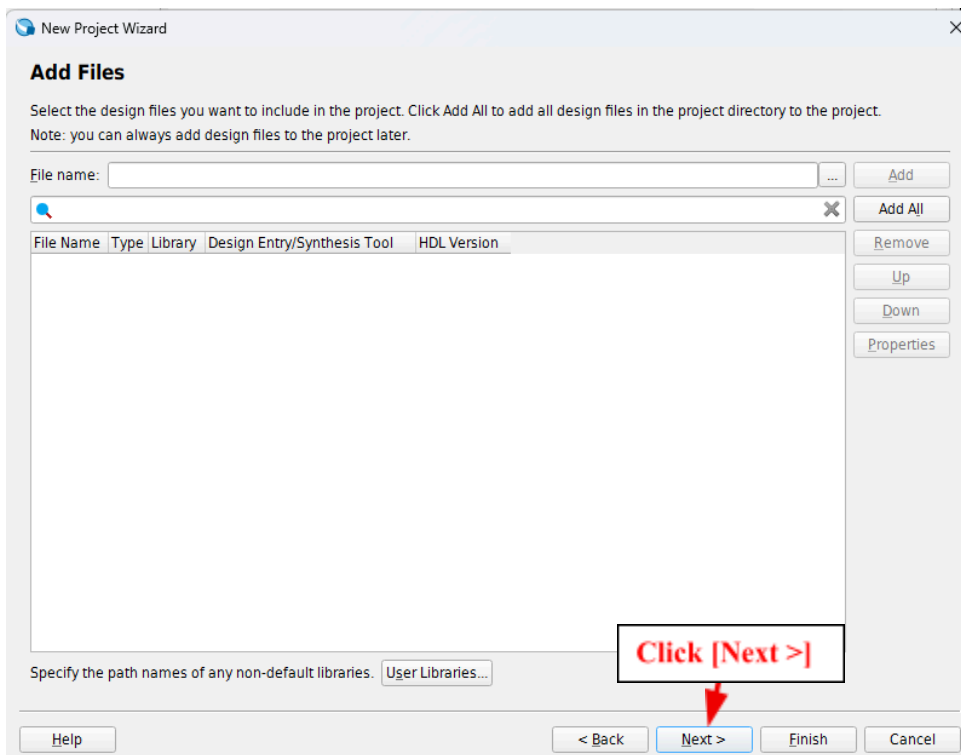
Then on the next window, you will need to enter the location where the new project is to be stored and the name of the new project.



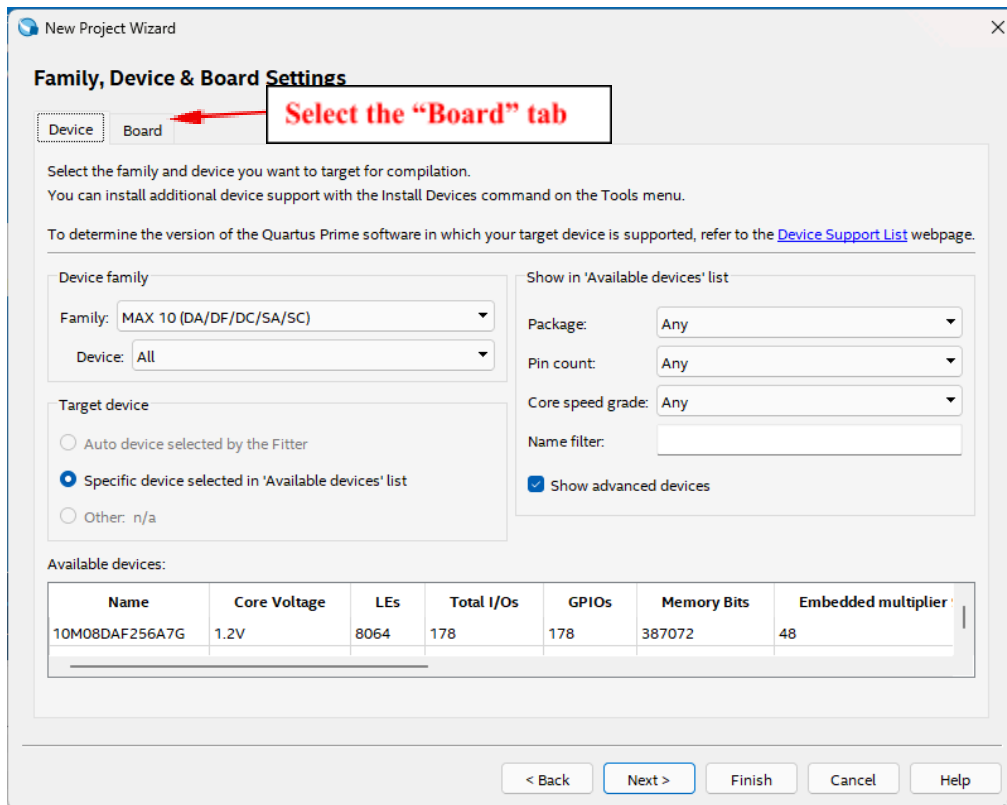
If you have not already created the folder at the specified location, you will be asked if you want to create it. If asked, answer “Yes”. Then the following screen will appear:



You will then be asked about adding files, which we will not be doing:



Next, you will see the screen of:



Family, Device & Board Settings

Device Board **Select the "Board" tab**

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

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

Device family

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

Target device

☐ Auto device selected by the Fitter
☒ Specific device selected in 'Available devices' list
☐ Other: n/a

Show in 'Available devices' list

Package: Any Pin count: Any Core speed grade: Any Name filter:

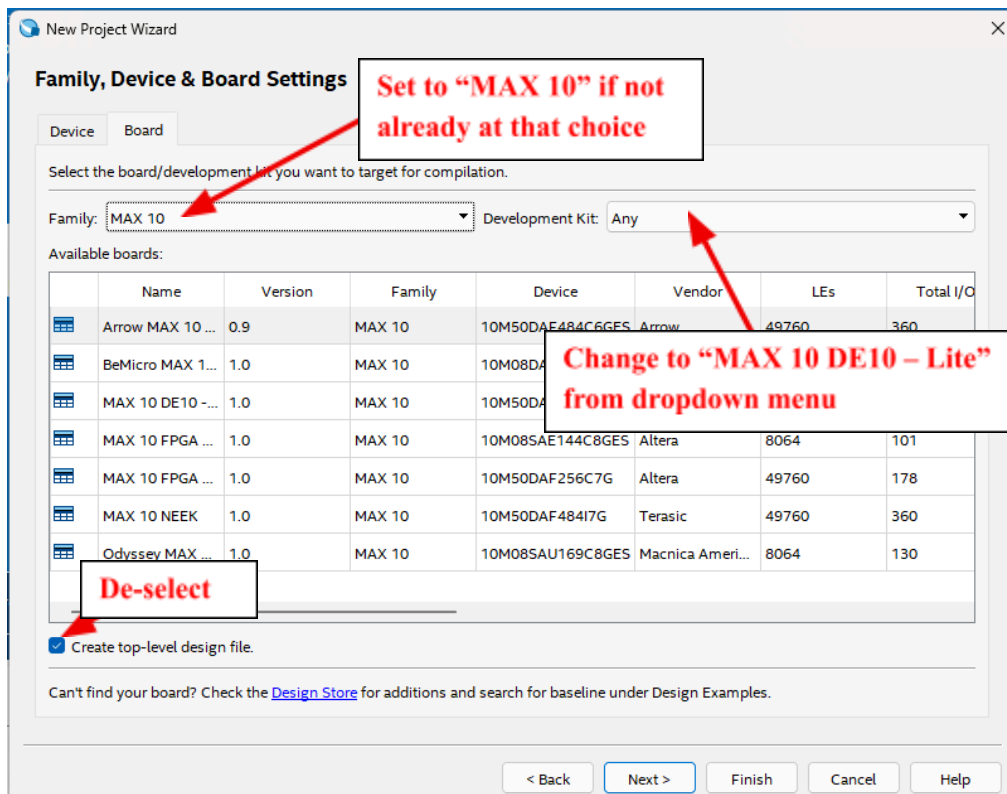
☒ Show advanced devices

Available devices:

Name	Core Voltage	LEs	Total I/Os	GPIOs	Memory Bits	Embedded multiplier
10M08DAF256A7G	1.2V	8064	178	178	387072	48

< Back Next > Finish Cancel Help

Upon selecting "Board" you should see:



Family, Device & Board Settings

Device Board **Set to "MAX 10" if not already at that choice**

Select the board/development kit you want to target for compilation.

Family: MAX 10 Development Kit: Any **Change to "MAX 10 DE10 - Lite" from dropdown menu**

Available boards:

Name	Version	Family	Device	Vendor	LEs	Total I/O
Arrow MAX 10 ...	0.9	MAX 10	10M50DAE484C6GES	Arrow	49760	360
BeMicro MAX 1...	1.0	MAX 10	10M08DA...			
MAX 10 DE10 -...	1.0	MAX 10	10M50DA...			
MAX 10 FPGA ...	1.0	MAX 10	10M08SAE144C8GES	Altera	8064	101
MAX 10 FPGA ...	1.0	MAX 10	10M50DAF256C7G	Altera	49760	178
MAX 10 NEEK	1.0	MAX 10	10M50DAF484I7G	Terasic	49760	360
Odyssey MAX ...	1.0	MAX 10	10M08SAU169C8GES	Macnica Ameri...	8064	130

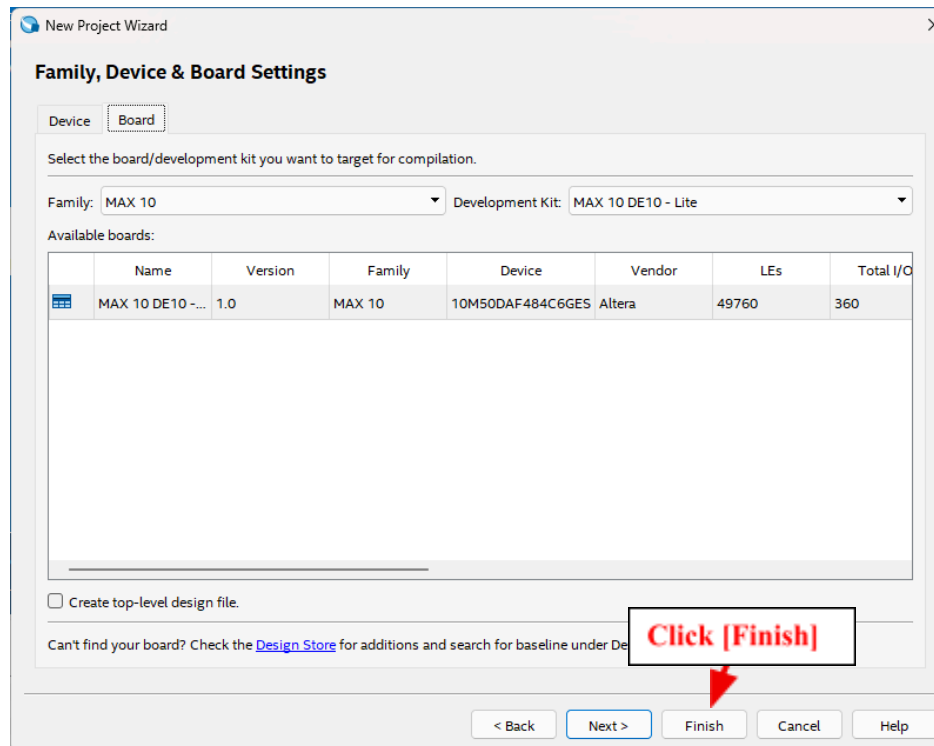
De-select

☒ Create top-level design file.

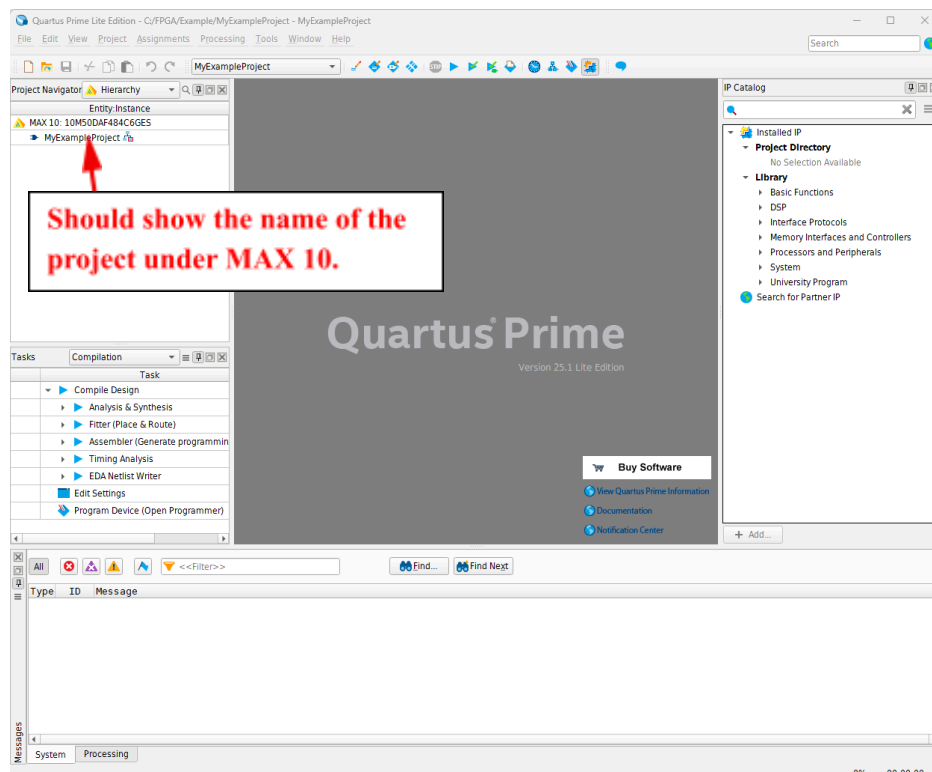
Can't find your board? Check the [Design Store](#) for additions and search for baseline under Design Examples.

< Back Next > Finish Cancel Help

Make sure you don't forget to deselect the option to "Create top-level design file." Now your window looks like this:



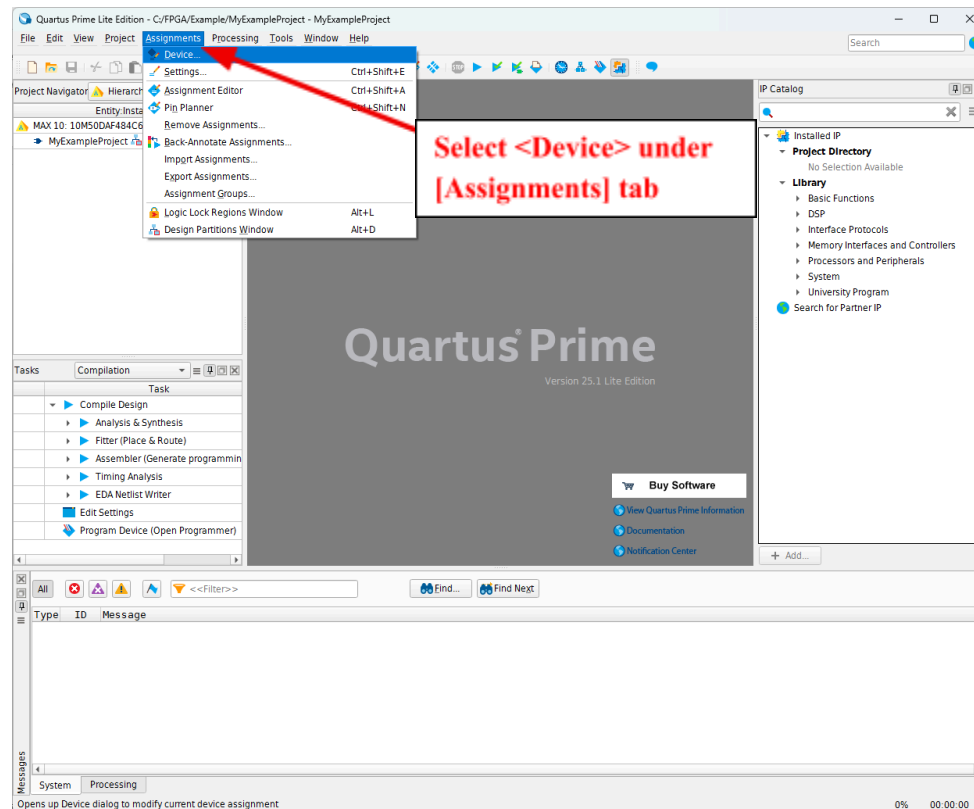
If you prefer, you can also click [Next >] for the next two windows, but there is nothing that you need to change. You will then be returned to the first window:



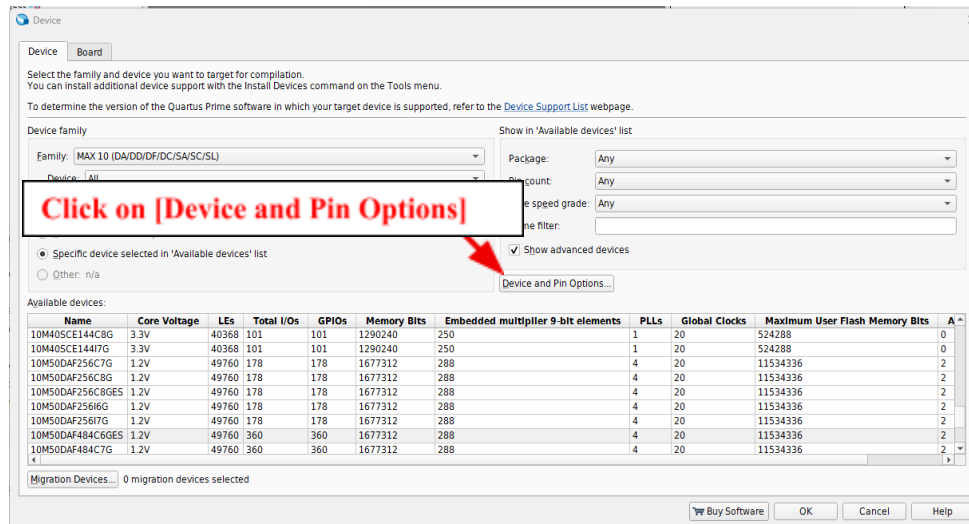
Setting the Default Pin Voltage:

This step is optional since each of the pin voltages can be set individually when creating a design, but since we will only use 3.3V on our FPGA board, doing this will save you time in the long run.

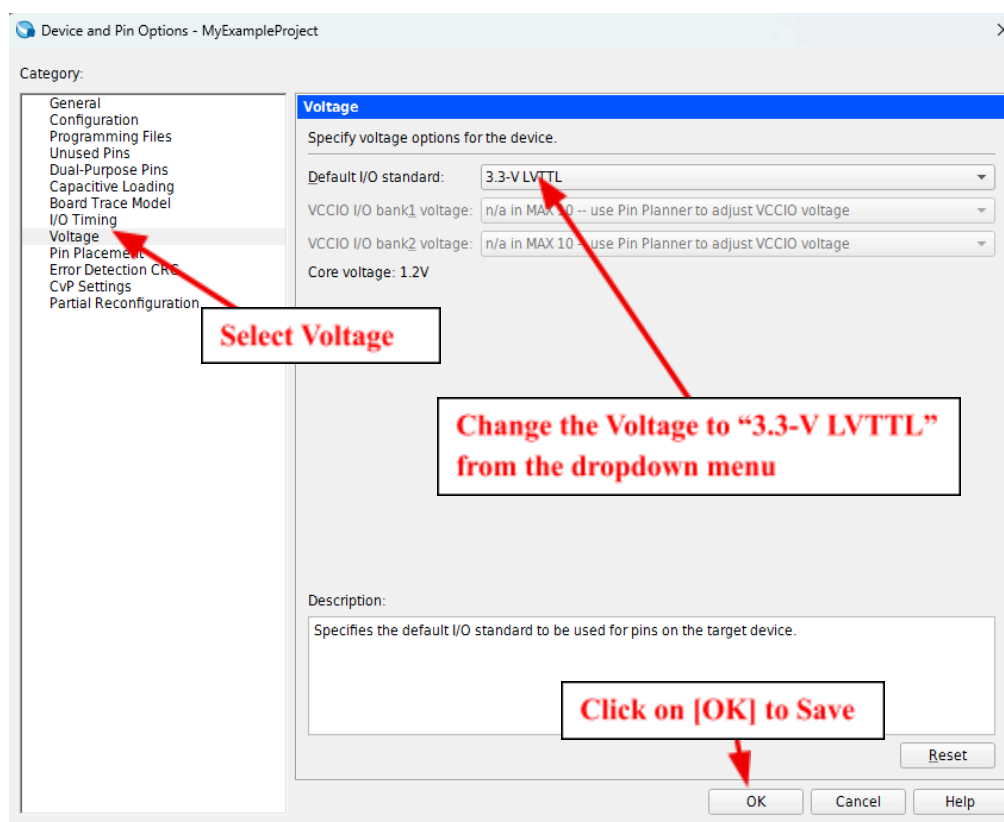
Once you have created your new project, go to the [Assignments] tab and select <Device> from the dropdown menu.



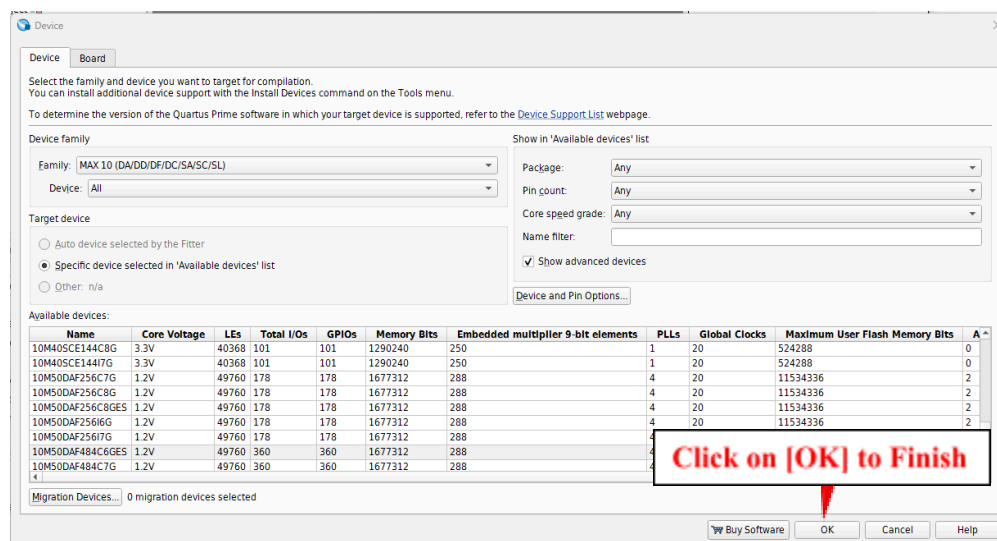
Now, you should see the window of:



Another window will appear:



This will return you to the previous window:

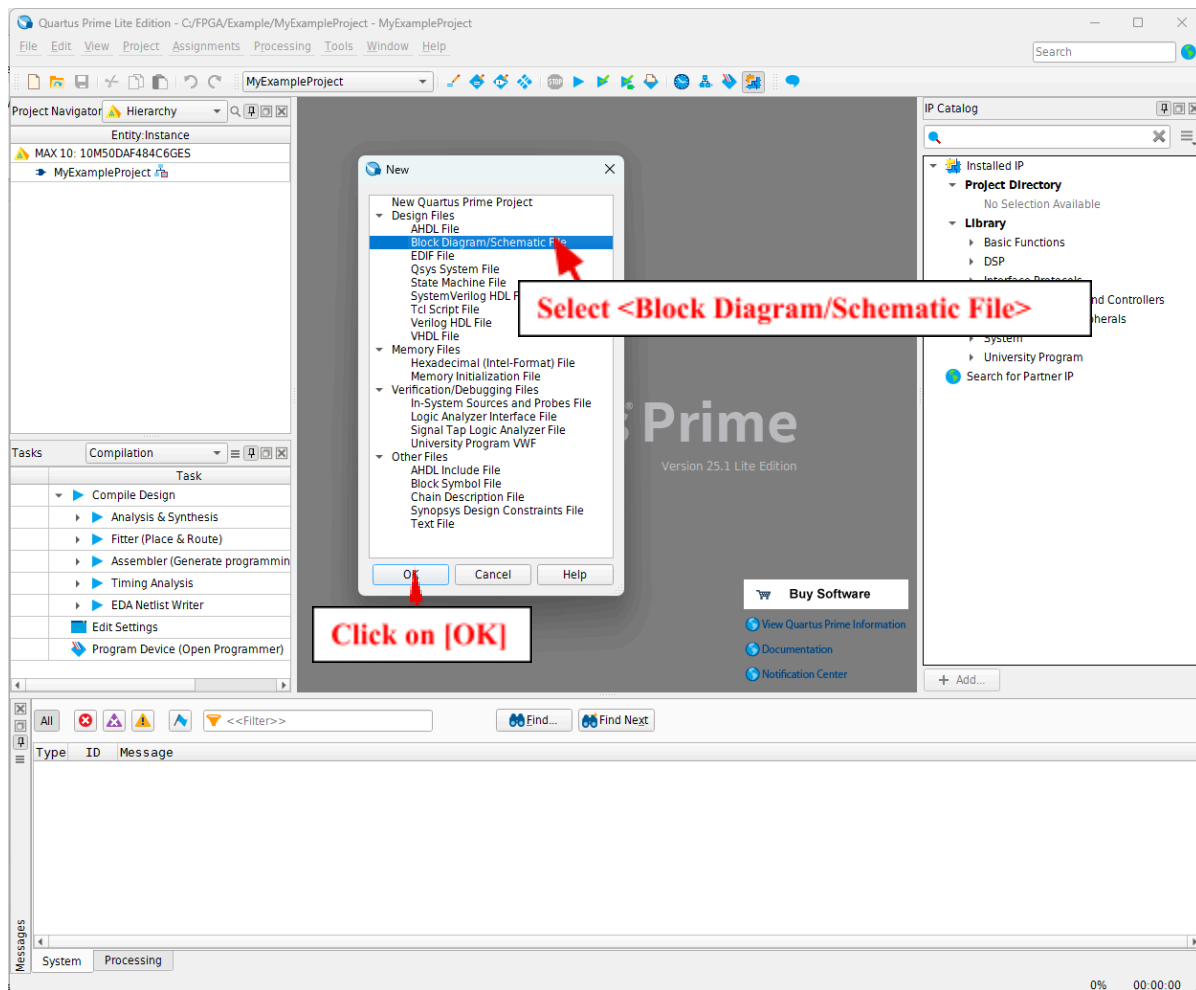


You have now set the default pin voltage for the pins (built-in switches and LED's) on your FPGA. You can also set them when you assign the pins before downloading your program into the FPGA; however, having the default voltages set correctly makes the process go faster.

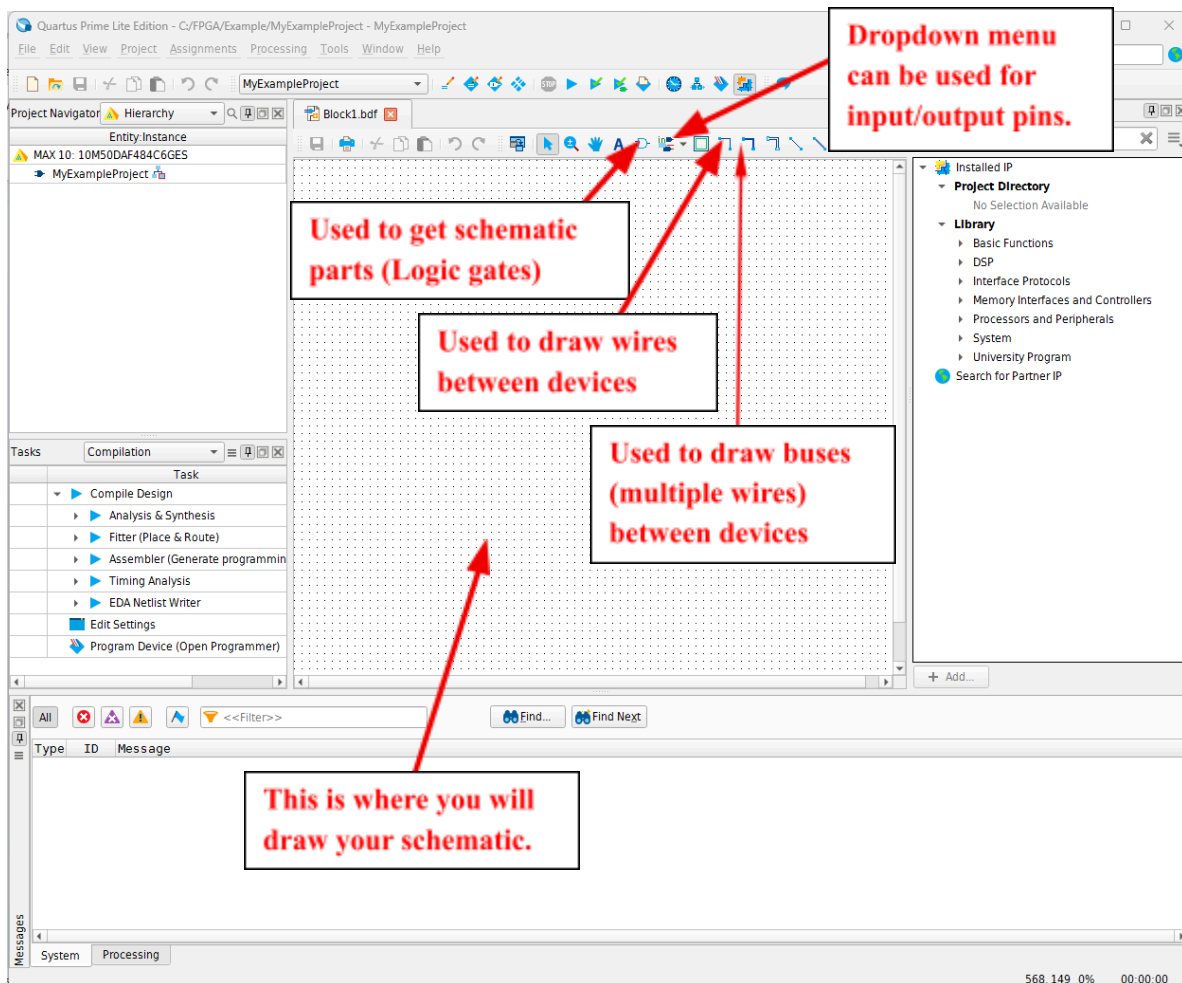
Creating a Circuit Schematic:

In our FPGA labs we will explore two different approaches to programming your FPGA. The first method is to create a circuit schematic, which is a lot like creating a circuit in LogicWorks. The second method is to write a program in Verilog, which is similar to writing a program in JAVA or C++. This section shows how to set up the Quartus software to create a circuit schematic.

After creating a new project (and possibly setting the default pin voltages), go to the [File] tab and select <New...> from the dropdown menu. The screen shown below will appear:

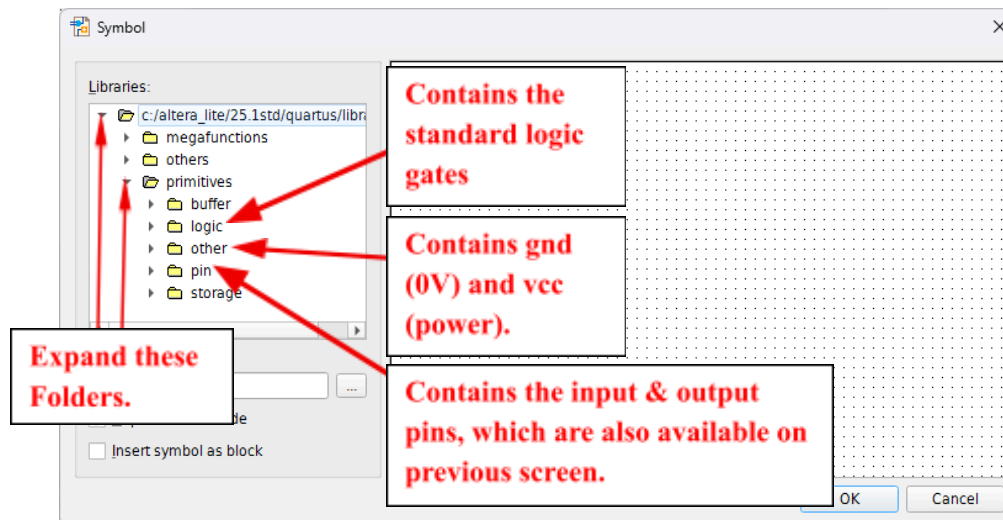


After selecting <Block Diagram/Schematic File> and clicking [OK], the following screen appears:



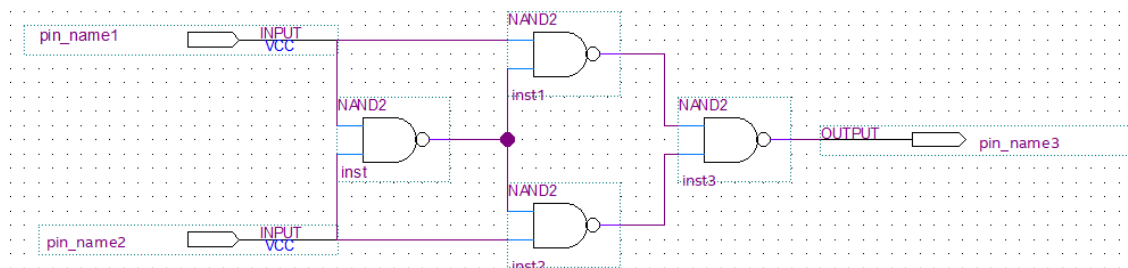
Notice the screen contains a grid of dots. If you prefer not to see them, you may turn them off by going to the [View] tab and unselecting the <Show Gridlines> option.

If you click on the logic gate icon [], the following window appears:



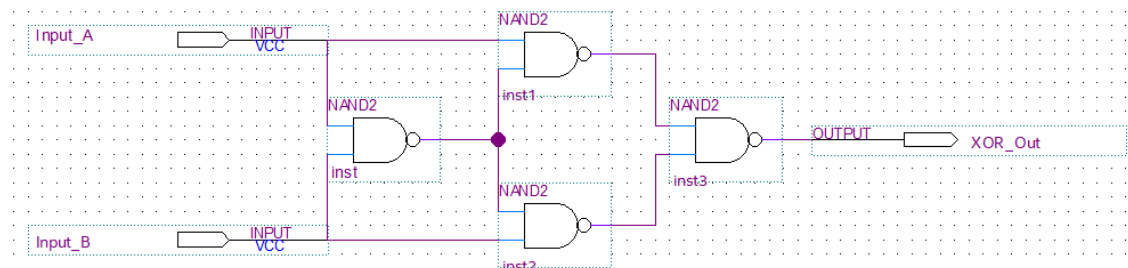
While the different folders contain a multitude of various functions, in general you will only be using the gates within the subfolders of [logic], [other], and [pin] under the [primitives] folder. Once you select your part, a preview of it appears on the right grid. When you click on [OK], you return to the previous window, with the part “attached” to your cursor. You may place multiple copies of the part in your diagram by simply clicking on different locations within the window. To get out of the part placement mode for that particular part, press the [Esc] button on your keyboard. Note: you may create more complex parts from logic gates or Verilog code, much like you can create more complex parts in a hierarchical fashion in LogicWorks. (This process is describe in the “Creating Device Symbols” section of this document.) If these parts are stored in your project’s root directory, that will also appear in the list of folders on the previous screen and will also need to be expanded to access the parts you created.

In your first FPGA lab, you will be creating an XOR gate using NAND gates, so we will use this as our example circuit. Your schematic will appear as:



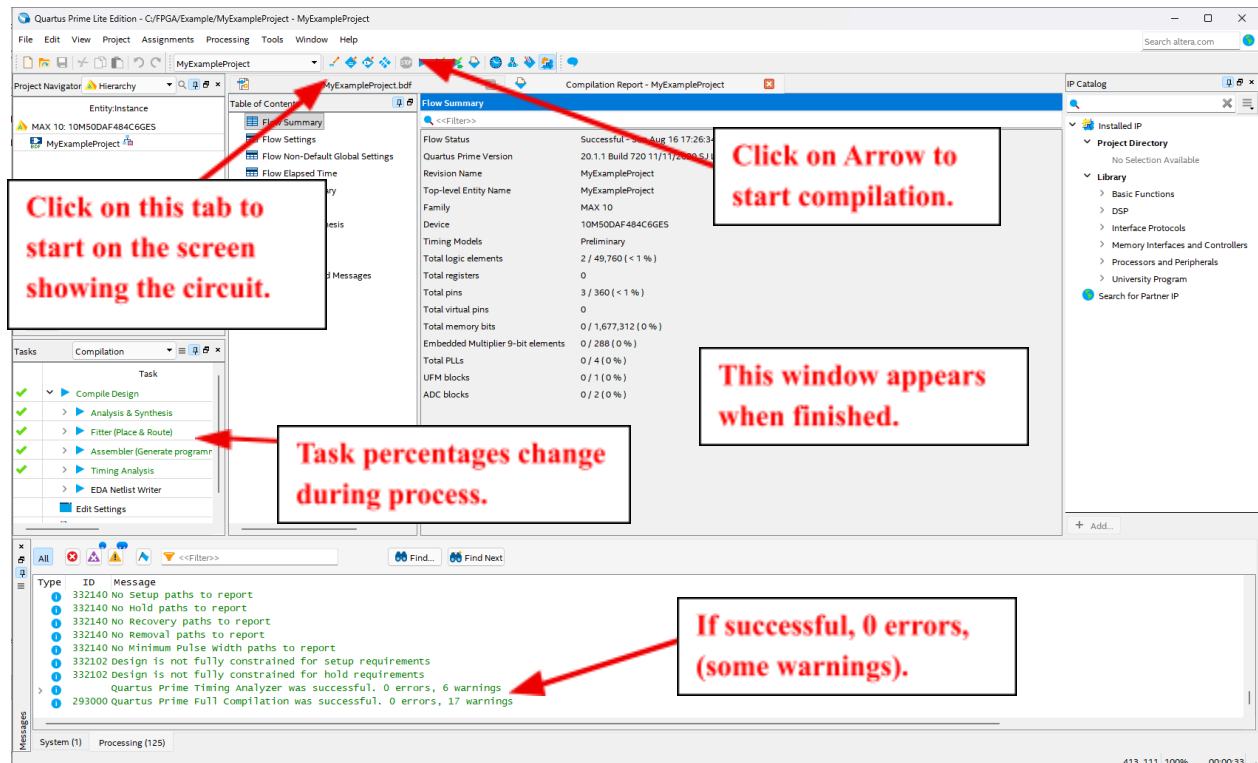
This schematic uses 4 copies of the 2-input NAND gate, 2 copies of the INPUT Pin and 1 copy of the OUTPUT Pin. The inputs and outputs were connected using the “Orthogonal Node Tool”.

You will also want to name the input and output pins on your schematic. You can do this simply by clicking on the “pin_name#” text. As shown below, the inputs were named “Input_A” and “Input_B”, while the output was named “XOR_Out”. Note: **DO NOT USE SPACES IN THE PIN NAMES**.



Once you have created a correct schematic of your circuit (be certain that there is no “X” at any of the device pins or large round nodes except where multiple wires meet at a connection), save your file by clicking on the floppy disk icon along the tool bar above the schematic. Use the same name for the file as the overall project, but the file extension will be “.bdf”. Also, when saving the file, in the window that appears, there should be a check box labeled “Add file to current project” near the [Save] button. Make sure that this box is checked before saving.

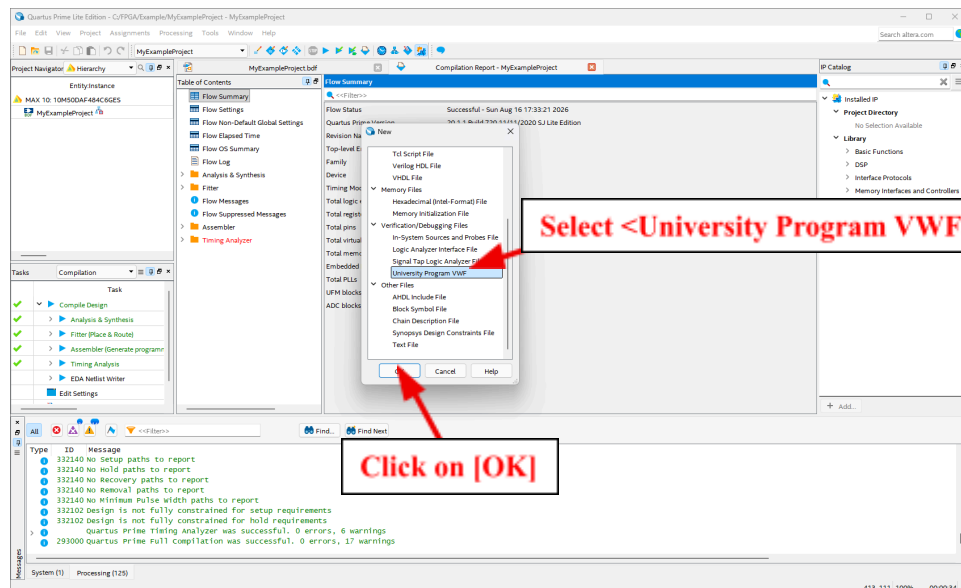
You will now need to compile your circuit. To do this, click on the blue arrow along the top toolbar. As it performs this task, which takes a couple of minutes, you will see the percentages change in the Tasks window, and on the bottom screen you will see various messages as it progresses. You may get some warnings, which you do not need to worry about, but if you get errors, you will need to fix your circuit. Below is the window you should see when your compilation is finished.



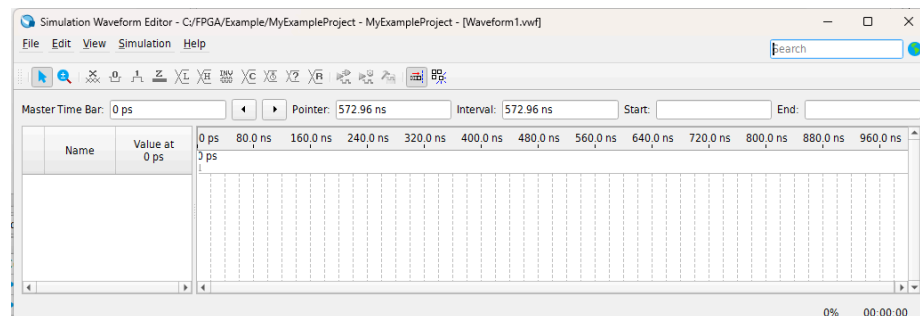
If your schematic is compiled with no errors, congratulations! You have designed a working circuit. Of course, we still need to simulate it and/or download it into the FPGA board.

Simulating with Waveforms:

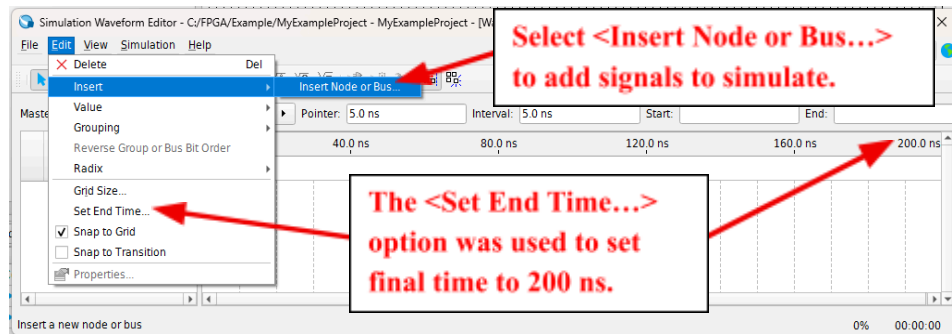
After compiling either a circuit schematic or Verilog code, it is possible to simulate its operation within the Quartus software before actually downloading it into your FPGA board. To do this, go to the [File] tab and select <New...> from the dropdown menu. The following screen will appear:



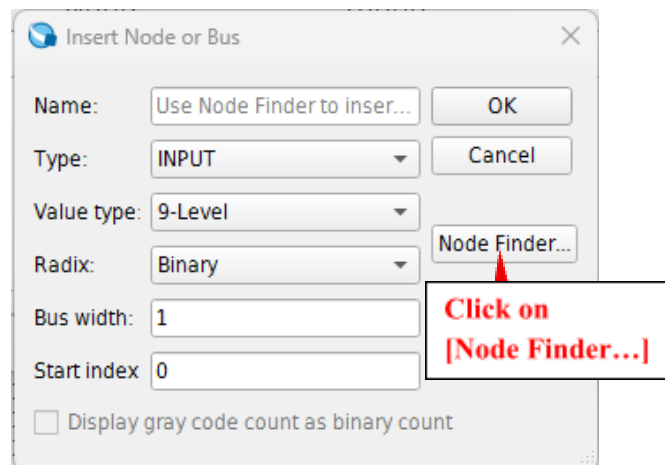
After selecting “University Program VWF”, the following screen then appears:



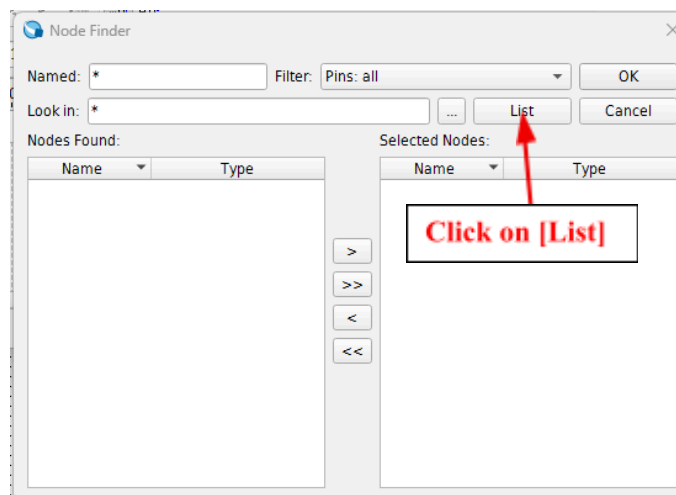
The first thing you might want to do at this window is to set the “End Time”, which is an option under the [Edit] tab. For the examples shown below, the end time was set to 200 ns. Next, you will want to add the signals that you want to simulate. To do this, again go to the [Edit] tab and select <Insert> ⇒ <Insert Node or Bus...>, as shown on the screen below:



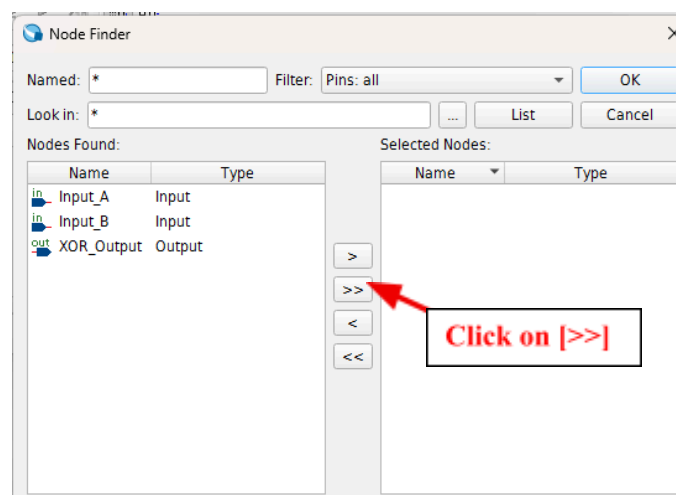
After selecting <Insert Node or Bus...>, a new window appears:



By clicking on [Node Finder...], yet another window appears:

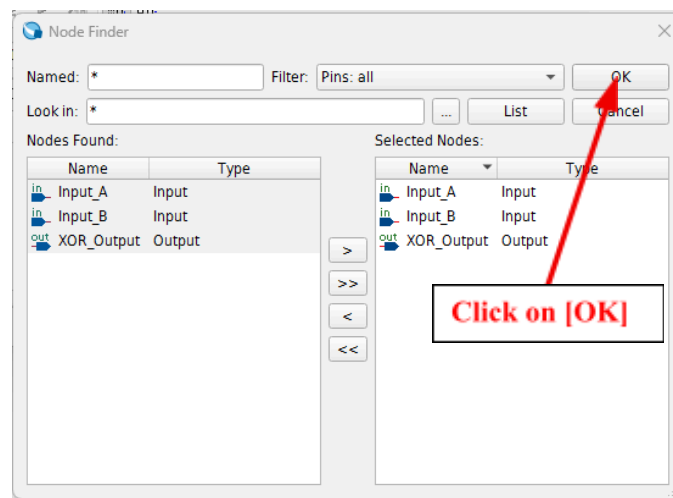


After clicking on [List], you should see all of the circuit's inputs and outputs listed on the left side of the screen:

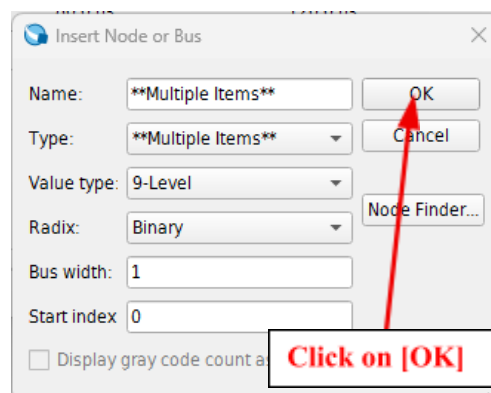


If you want to see the timing diagram for all of these signals, click on the [>>] button.

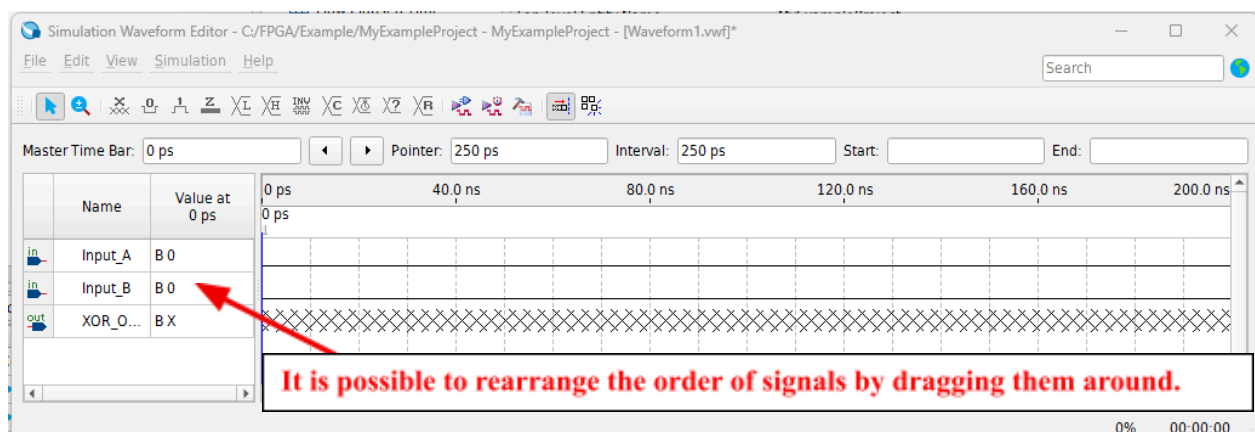
The screen now appears as:



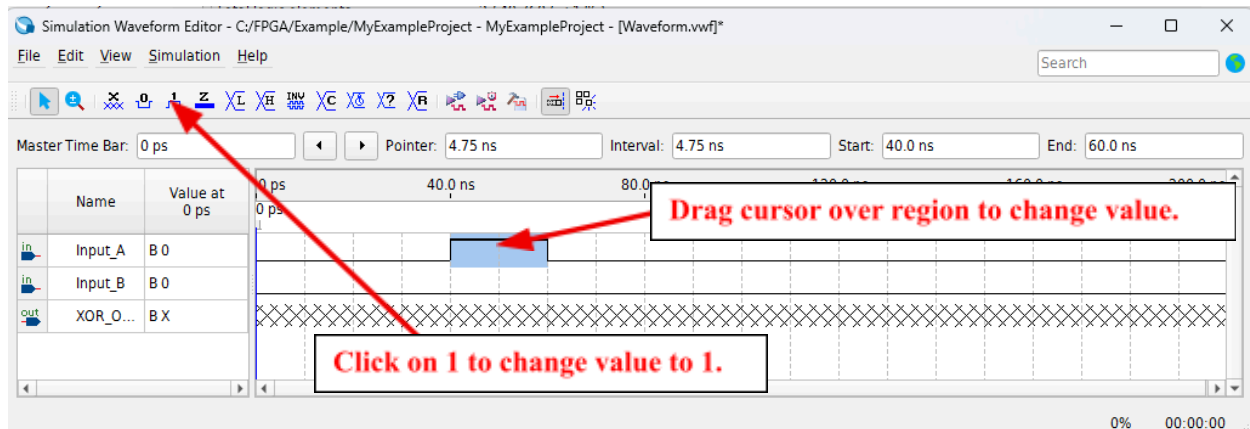
After clicking on the [OK] button, the previous “Insert Node or Bus” screen appears with “**Multiple Items**” showing for the Name and Type. Click on [OK] again.



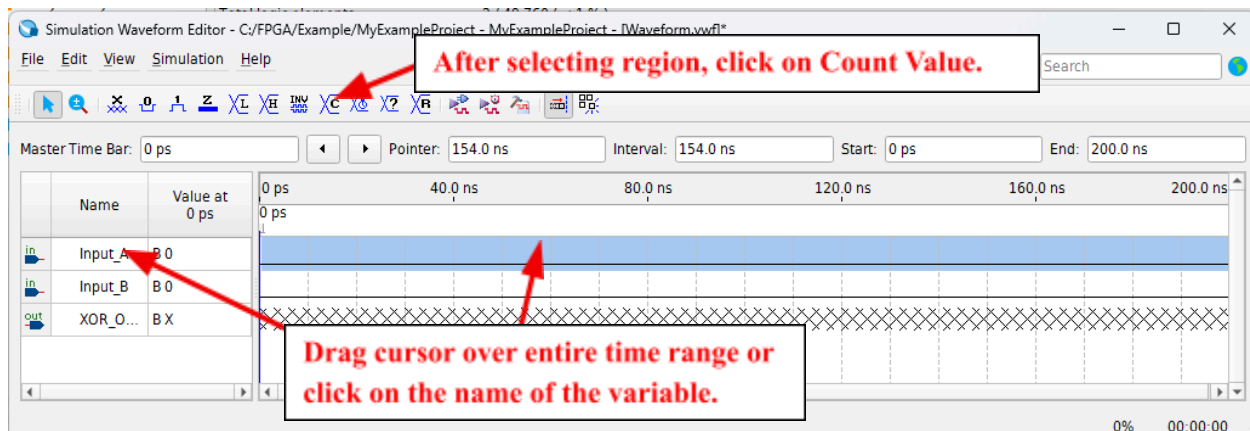
Now you will be back at the Waveform window, but with the signal names listed on the left side.



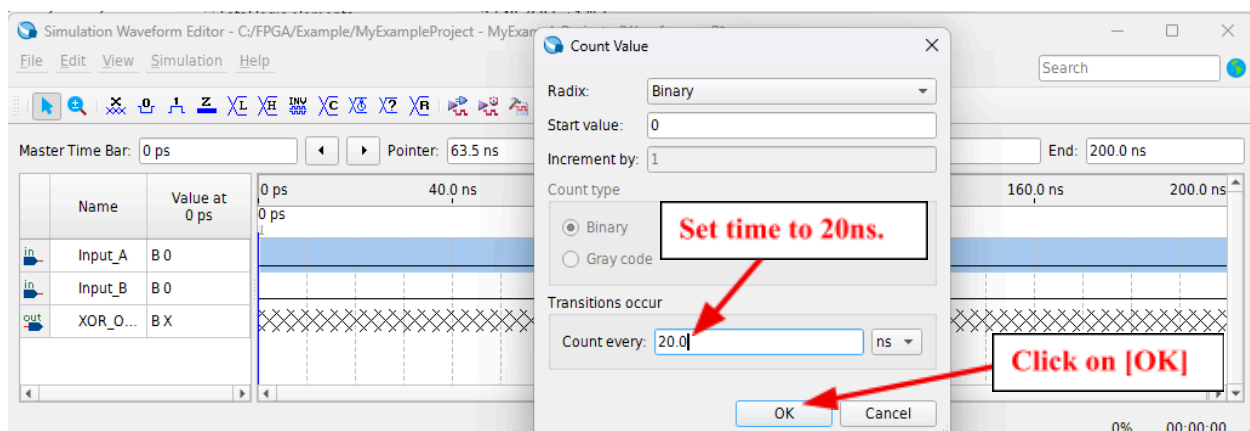
The two input signals have a constant value of 0, and the output signal is undetermined (simulation hasn't been run yet). To set the value of the input signal during a specific time region, it is possible to simply drag the cursor over a region with the mouse's button held down. If you want to include the entire simulation time, you may either drag the mouse over the entire region or just click on the variable name in the left column.



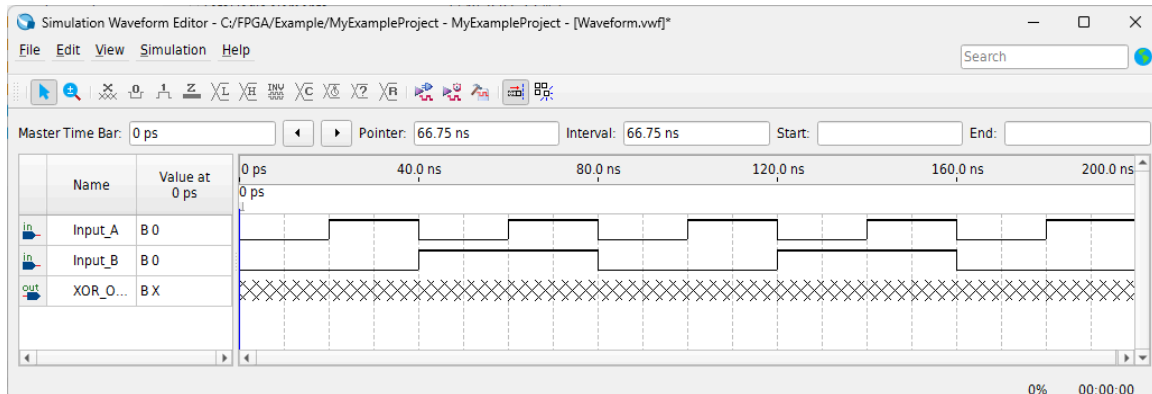
Another way to set the values is to use the count function. (This way is particularly useful when the input is a bus containing multiple wires.) Select the entire time range as shown below:



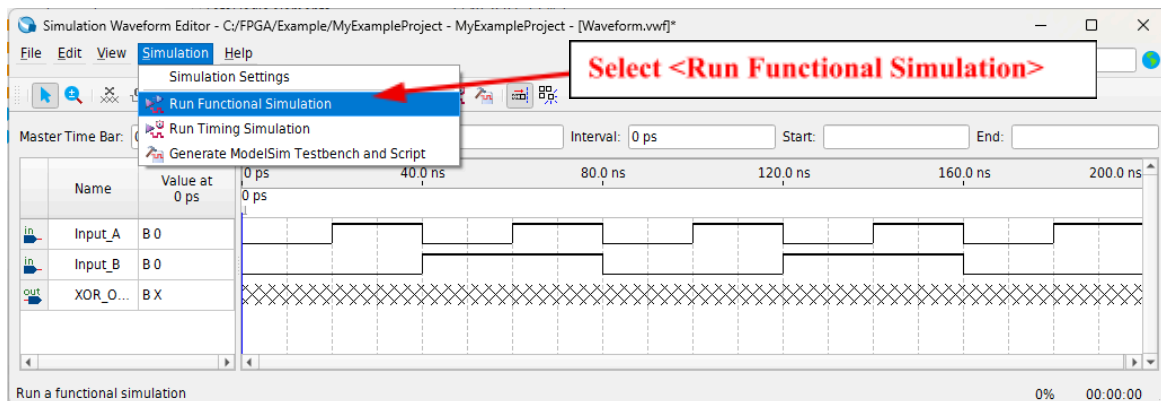
After clicking on the "Count Value" icon, another window will appear:



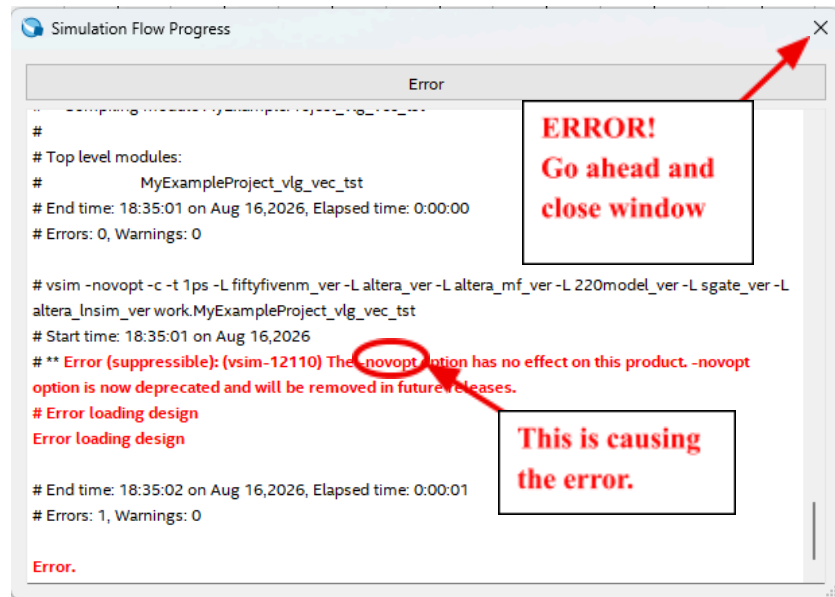
Then do the same for Input_B but set its count time to 40 ns. When finished, the input signals should appear as:



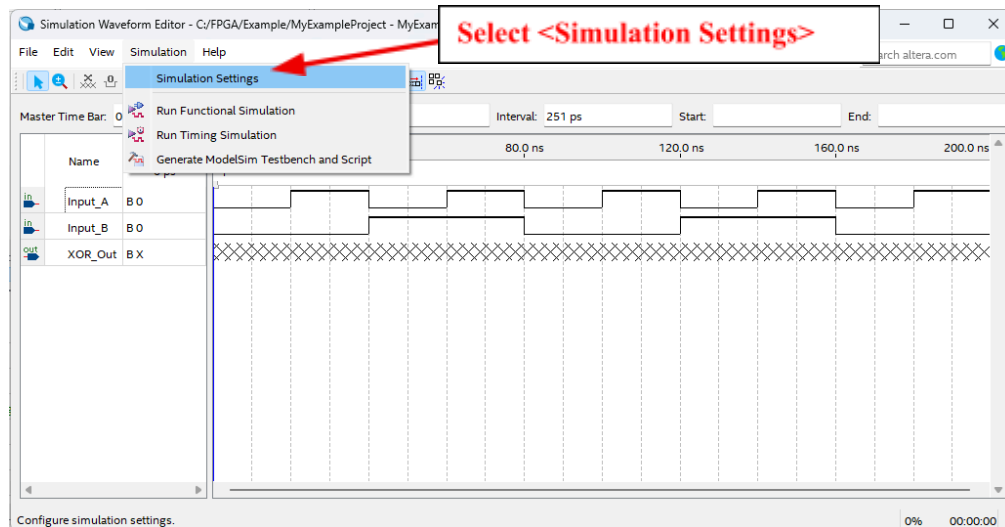
Notice that because there is a period doubling from A to B, all combinations of their values appear within the time range. To simulate the output for the input signals shown, go to the [Simulation] tab and choose the option <Run Functional Simulation>.



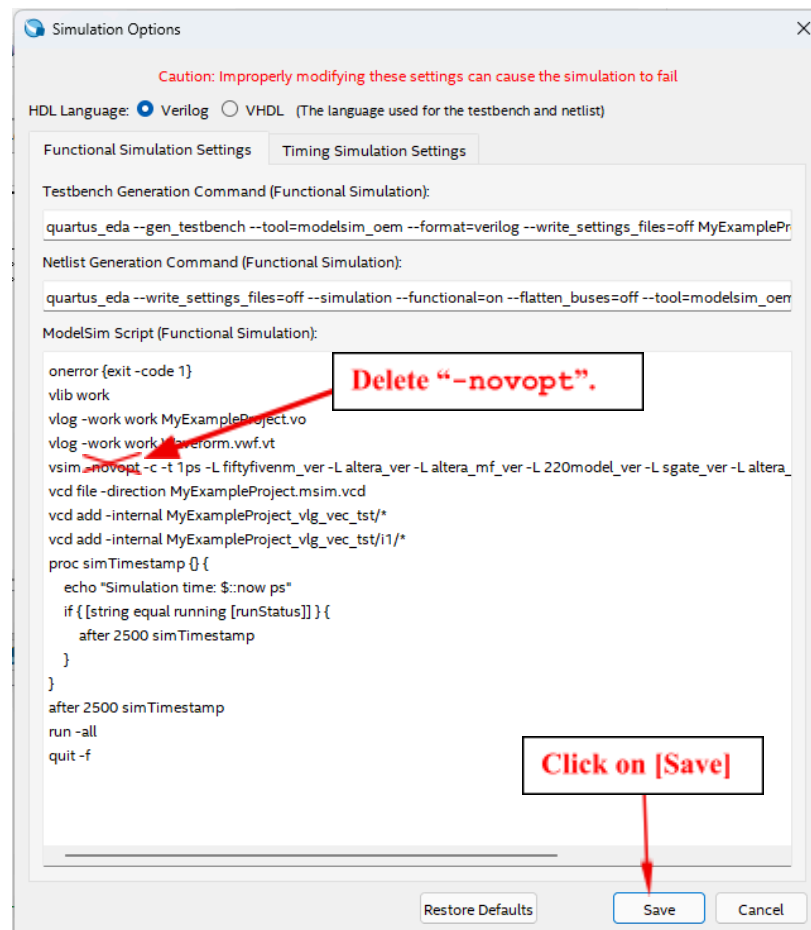
Most likely, you will see an error message:



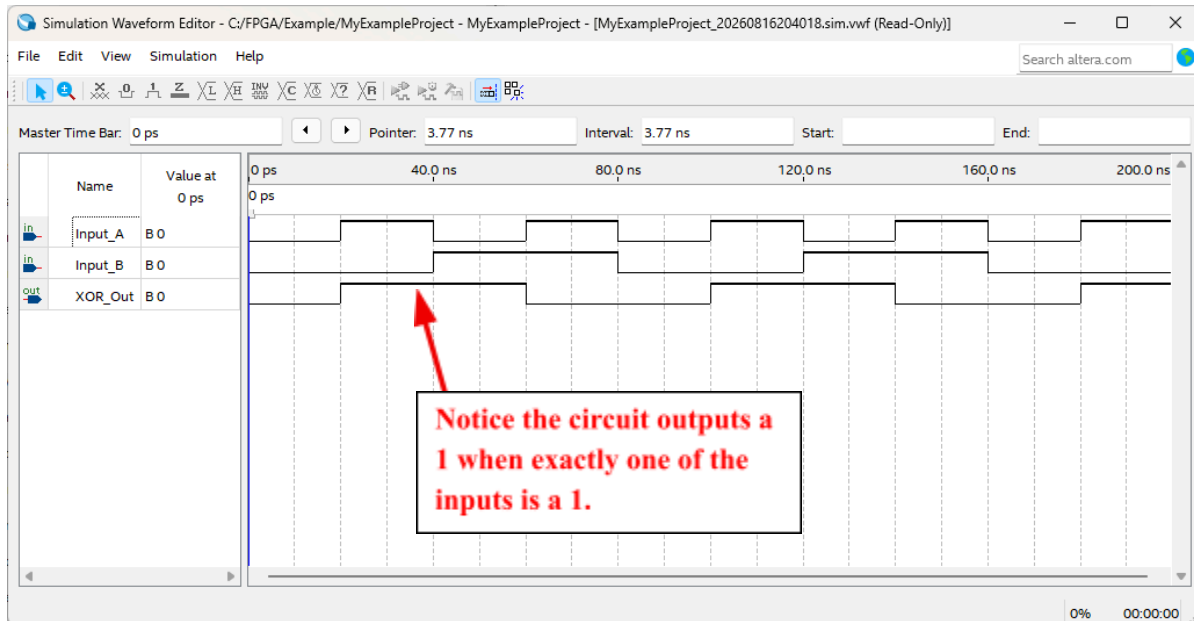
If that is the case, (**you have an error**), you will need to change the simulation settings. First, close the error message window. Then go back to the [Simulation] tab and select <Simulation Settings>.



The Simulation Options window opens. You will need to find the “-novopt” command in the options and delete it, as shown below:



Now, retry to simulate the output by going to the [Simulation] tab and choose the option <Run Functional Simulation>. Another “Simulation Flow Progress” window will open, showing various messages as the circuit is simulated. If everything is correct, this window closes and a new window opens with the new XOR_Out waveform shown:

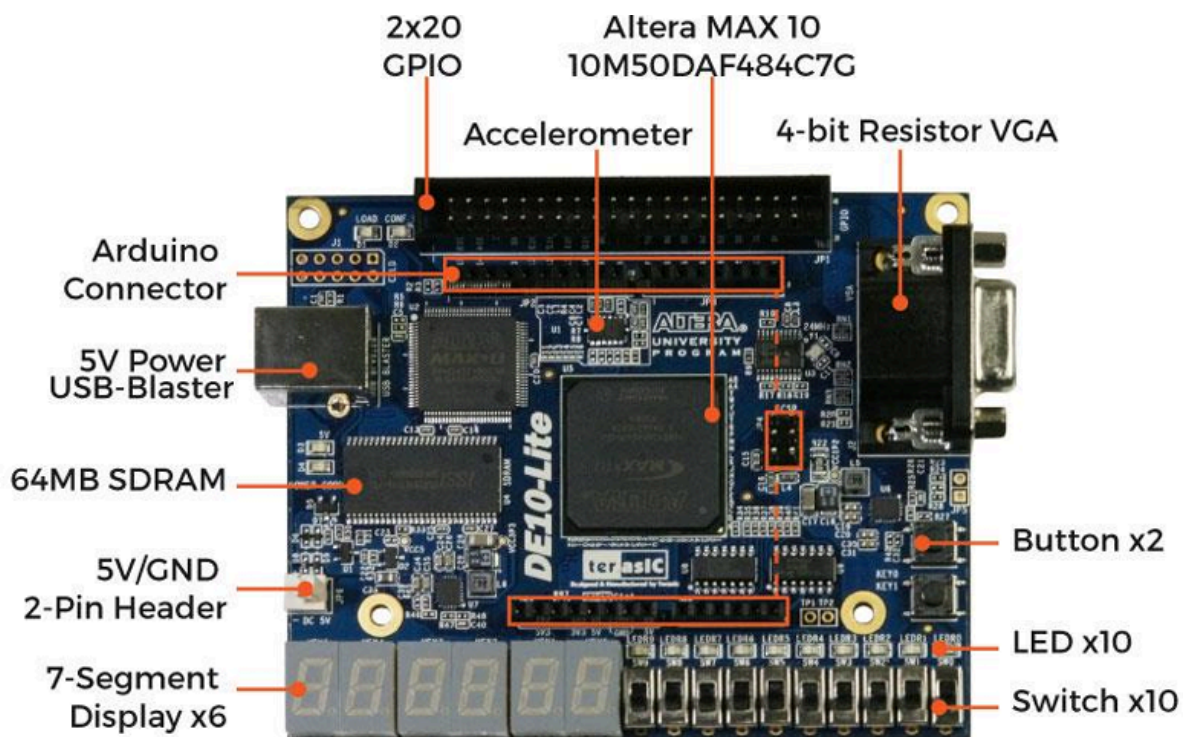


Depending on the instructions for your particular lab, you may need to print out a copy of the waveform simulation.

Assigning Switches and LEDs:

Once you have created a schematic or written Verilog code, you will need to assign Switches to the inputs and LEDs to the outputs. (Note: Assigning switches and LEDs can be done without simulating the waveform. Basically, the simulator uses your computer to determine what the output should be, but it is not necessary to simulate the circuit to download it into your FPGA board and test it out.)

Below is a picture of our DE10-Lite board:



In general, for our class we will only be using the “7-Segment Display x6” and “LED x10” for outputs, “Switch x10” for normal inputs, and the “Button x2” for clock inputs. If you look closely at your board, you will see that the LEDs and switches are labeled, starting with the number 0.

Unfortunately, to use the switches and pins, you need to know the pins they connect to on the actual FPGA chip, and they are not organized into a nice pattern. The diagram below shows the 10 slider switches and the pins they are connected to:



The table below shows the pin assignment of the **Slider Switches** along with the voltages to use:

Signal Name	FPGA Pin No.	Description	I/O Standard
SW0	PIN_C10	Slide Switch[0]	3.3-V LVTTTL
SW1	PIN_C11	Slide Switch[1]	3.3-V LVTTTL
SW2	PIN_D12	Slide Switch[2]	3.3-V LVTTTL
SW3	PIN_C12	Slide Switch[3]	3.3-V LVTTTL
SW4	PIN_A12	Slide Switch[4]	3.3-V LVTTTL
SW5	PIN_B12	Slide Switch[5]	3.3-V LVTTTL
SW6	PIN_A13	Slide Switch[6]	3.3-V LVTTTL
SW7	PIN_A14	Slide Switch[7]	3.3-V LVTTTL
SW8	PIN_B14	Slide Switch[8]	3.3-V LVTTTL
SW9	PIN_F15	Slide Switch[9]	3.3-V LVTTTL

The table below shows the pin assignment of the **Push Buttons** (used as manual clocks) along with the voltages to use. Note: The Schmitt Trigger is used on the output give a “clean” transition between high and low voltages. The output is normally high until the button is pushed.

Signal Name	FPGA Pin No.	Description	I/O Standard
KEY0	PIN_B8	Push-button[0]	3.3 V SCHMITT TRIGGER"
KEY1	PIN_A7	Push-button[1]	3.3 V SCHMITT TRIGGER"

Similarly, the pin assignments for the **LEDs** directly above the slider switches are:

Signal Name	FPGA Pin No.	Description	I/O Standard
LEDR0	PIN_A8	LED [0]	3.3-V LVTTTL
LEDR1	PIN_A9	LED [1]	3.3-V LVTTTL
LEDR2	PIN_A10	LED [2]	3.3-V LVTTTL
LEDR3	PIN_B10	LED [3]	3.3-V LVTTTL
LEDR4	PIN_D13	LED [4]	3.3-V LVTTTL
LEDR5	PIN_C13	LED [5]	3.3-V LVTTTL
LEDR6	PIN_E14	LED [6]	3.3-V LVTTTL
LEDR7	PIN_D14	LED [7]	3.3-V LVTTTL
LEDR8	PIN_A11	LED [8]	3.3-V LVTTTL
LEDR9	PIN_B11	LED [9]	3.3-V LVTTTL

For each of the 6x 7-segment displays, there are 8 pins used with one of them controlling the decimal point, (“DP” in the table below). Thus, there are a total of 48 pins controlling them.

The diagram below shows the order of the seven LED segments (common anode) and decimal point for one of the displays. A segment is turned on or off by applying a low logic level (to turn on) or high logic level (to turn off) from the FPGA, respectively. (Unconnected or unspecified segments are “floating” and will remain off.)

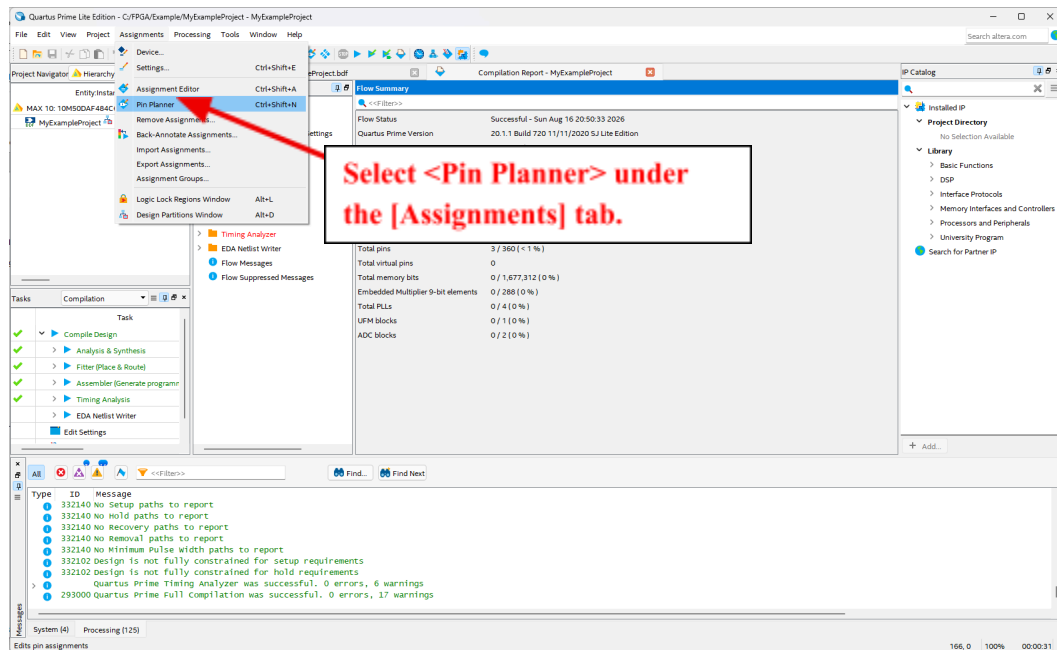


The pin assignments for the **Seven Segment Digits** are:

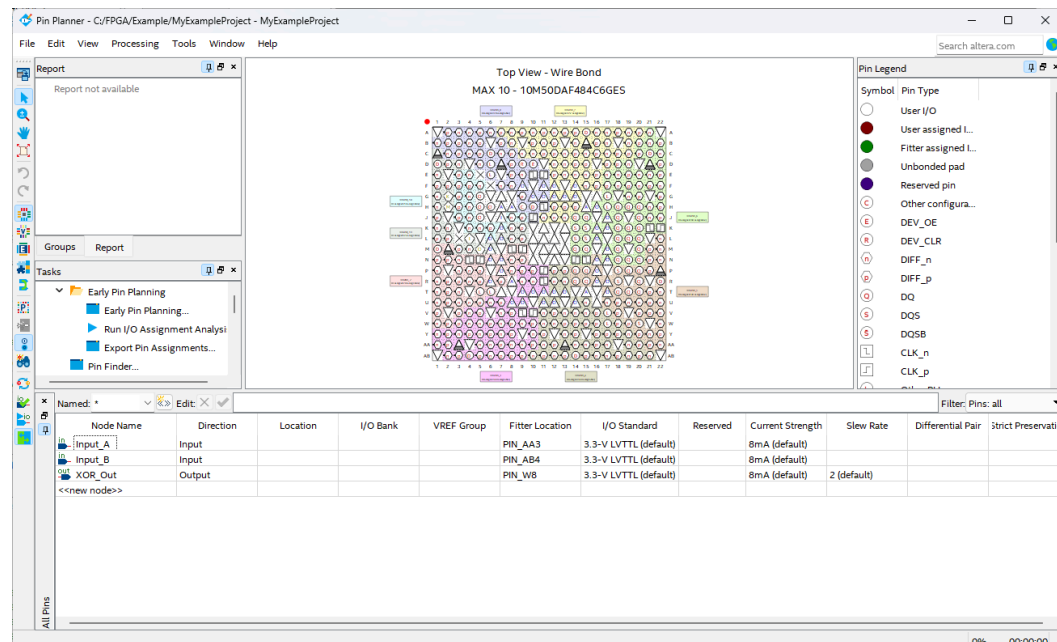
Signal Name	FPGA Pin No.	Description	I/O Standard
HEX00	PIN_C14	Seven Segment Digit 0[0]	3.3-V LVTTTL
HEX01	PIN_E15	Seven Segment Digit 0[1]	3.3-V LVTTTL
HEX02	PIN_C15	Seven Segment Digit 0[2]	3.3-V LVTTTL
HEX03	PIN_C16	Seven Segment Digit 0[3]	3.3-V LVTTTL
HEX04	PIN_E16	Seven Segment Digit 0[4]	3.3-V LVTTTL
HEX05	PIN_D17	Seven Segment Digit 0[5]	3.3-V LVTTTL
HEX06	PIN_C17	Seven Segment Digit 0[6]	3.3-V LVTTTL
HEX07	PIN_D15	Seven Segment Digit 0[7], DP	3.3-V LVTTTL
HEX10	PIN_C18	Seven Segment Digit 1[0]	3.3-V LVTTTL
HEX11	PIN_D18	Seven Segment Digit 1[1]	3.3-V LVTTTL
HEX12	PIN_E18	Seven Segment Digit 1[2]	3.3-V LVTTTL
HEX13	PIN_B16	Seven Segment Digit 1[3]	3.3-V LVTTTL
HEX14	PIN_A17	Seven Segment Digit 1[4]	3.3-V LVTTTL
HEX15	PIN_A18	Seven Segment Digit 1[5]	3.3-V LVTTTL
HEX16	PIN_B17	Seven Segment Digit 1[6]	3.3-V LVTTTL
HEX17	PIN_A16	Seven Segment Digit 1[7], DP	3.3-V LVTTTL

HEX20	PIN_B20	Seven Segment Digit 2[0]	3.3-V LVTTTL
HEX21	PIN_A20	Seven Segment Digit 2[1]	3.3-V LVTTTL
HEX22	PIN_B19	Seven Segment Digit 2[2]	3.3-V LVTTTL
HEX23	PIN_A21	Seven Segment Digit 2[3]	3.3-V LVTTTL
HEX24	PIN_B21	Seven Segment Digit 2[4]	3.3-V LVTTTL
HEX25	PIN_C22	Seven Segment Digit 2[5]	3.3-V LVTTTL
HEX26	PIN_B22	Seven Segment Digit 2[6]	3.3-V LVTTTL
HEX27	PIN_A19	Seven Segment Digit 2[7] , DP	3.3-V LVTTTL
HEX30	PIN_F21	Seven Segment Digit 3[0]	3.3-V LVTTTL
HEX31	PIN_E22	Seven Segment Digit 3[1]	3.3-V LVTTTL
HEX32	PIN_E21	Seven Segment Digit 3[2]	3.3-V LVTTTL
HEX33	PIN_C19	Seven Segment Digit 3[3]	3.3-V LVTTTL
HEX34	PIN_C20	Seven Segment Digit 3[4]	3.3-V LVTTTL
HEX35	PIN_D19	Seven Segment Digit 3[5]	3.3-V LVTTTL
HEX36	PIN_E17	Seven Segment Digit 3[6]	3.3-V LVTTTL
HEX37	PIN_D22	Seven Segment Digit 3[7] , DP	3.3-V LVTTTL
HEX40	PIN_F18	Seven Segment Digit 4[0]	3.3-V LVTTTL
HEX41	PIN_E20	Seven Segment Digit 4[1]	3.3-V LVTTTL
HEX42	PIN_E19	Seven Segment Digit 4[2]	3.3-V LVTTTL
HEX43	PIN_J18	Seven Segment Digit 4[3]	3.3-V LVTTTL
HEX44	PIN_H19	Seven Segment Digit 4[4]	3.3-V LVTTTL
HEX45	PIN_F19	Seven Segment Digit 4[5]	3.3-V LVTTTL
HEX46	PIN_F20	Seven Segment Digit 4[6]	3.3-V LVTTTL
HEX47	PIN_F17	Seven Segment Digit 4[7] , DP	3.3-V LVTTTL
HEX50	PIN_J20	Seven Segment Digit 5[0]	3.3-V LVTTTL
HEX51	PIN_K20	Seven Segment Digit 5[1]	3.3-V LVTTTL
HEX52	PIN_L18	Seven Segment Digit 5[2]	3.3-V LVTTTL
HEX53	PIN_N18	Seven Segment Digit 5[3]	3.3-V LVTTTL
HEX54	PIN_M20	Seven Segment Digit 5[4]	3.3-V LVTTTL
HEX55	PIN_N19	Seven Segment Digit 5[5]	3.3-V LVTTTL
HEX56	PIN_N20	Seven Segment Digit 5[6]	3.3-V LVTTTL
HEX57	PIN_L19	Seven Segment Digit 5[7] , DP	3.3-V LVTTTL

After creating either a circuit schematic or writing Verilog code, and then compiling it, you need to assign switches and LEDs to the inputs and outputs before downloading it into your FPGA board. To do this, go to the <Pin Planner> option under the [Assignments] tab, as shown below:



The following window will then appear:



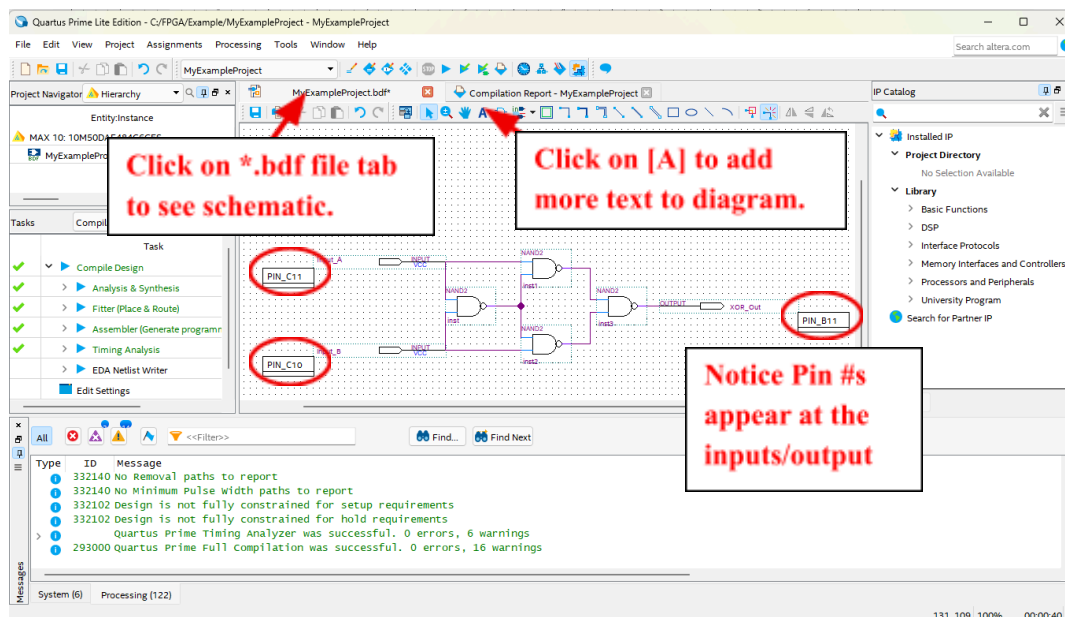
In our example circuit, we had only 2 inputs and 1 output, which is why there are only 3 listed on the bottom screen, but we would have more listed if our circuit had more inputs and outputs. Also notice that for our "I/O Standard" column, we have "3.3-V LVTTTL" listed because we previously set that as our default voltage. If the default voltages were not previously set, the "I/O Standard" column can have its values changed individually for each input and output.

Reference GuideAssigning Switches and LEDs40

In this example, we will use slide switches SW0 and SW1 for the inputs and LEDR9 for the output. You can do this by setting the locations to the pins listed in the previous tables, i.e.:

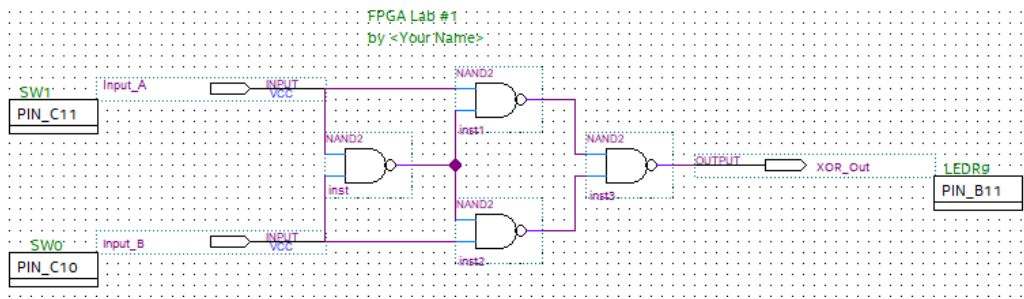
Node Name	Direction	Location	I/O Bank	VREF Group	Fitter Location	I/O Standard	Reserved	Current Strength	Slew Rate
Input_A	Input	PIN_C11	7	B7_NO	PIN_AA3	3.3-V LVTTTL (default)		8mA (default)	
Input_B	Input	PIN_C10	7	B7_NO	PIN_AB4	3.3-V LVTTTL (default)		8mA (default)	
XOR_Out	Output	PIN_B11	7	B7_NO	PIN_W8	3.3-V LVTTTL (default)		8mA (default)	2 (default)
<<new node>>									

Notice that only the “Location” blanks were completed with the dropdown menu at each, while the other columns were automatically filled in based on the selected locations. (Rather than using the dropdown menus for the locations, you may also type in the pins, such as “C11” or “C10”, and pressing [Enter].) Once completed, you can just close the window (nothing to save). Now the locations should appear next to the pins on the circuit schematic, i.e.:



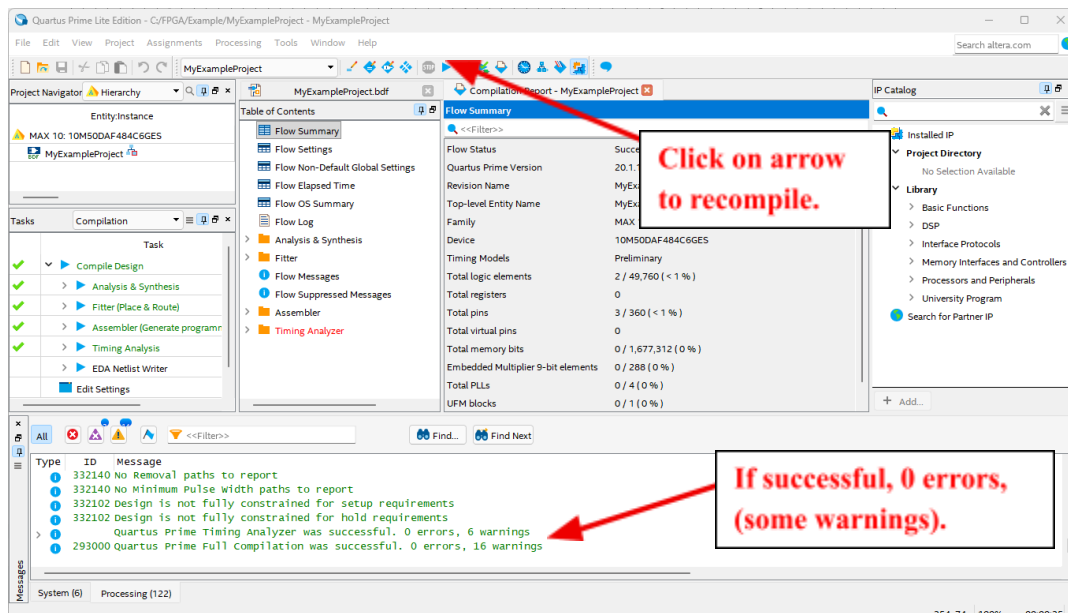
Note: if you created your circuit using Verilog and not by drawing a circuit schematic, you will still need to assign pins to the inputs and outputs, but there will be no changes to observe.

While the pin numbers will appear on the schematic next to the inputs and outputs, their labels are not particularly useful for determining which switches and LEDs are being used. Consequently, it is highly recommended that you add text to your schematic to make it clear which ones you are using. As shown in the above figure, this can be done by clicking on the [A] button above the schematic. For this example, the following text was added:



In the above diagram, PIN_C11 was labeled as “SW1,” PIN_C10 was labeled as “SW0”, and PIN_B11 was labeled “LED9,” so it is now clear which switches and LED will be used on the FPGA board. Also notice a Title and name were added to the circuit. Note: the text option does not let you do any formatting on the text you add.

After the pins have been assigned, the circuit needs to be recompiled with the new pin assignments. If you don’t do this next, when you download it into your FPGA, you won’t get any error messages, but the FPGA just won’t do anything, because it doesn’t know what switches and LEDs to use for input and output. After being recompiled, the following screen should appear:

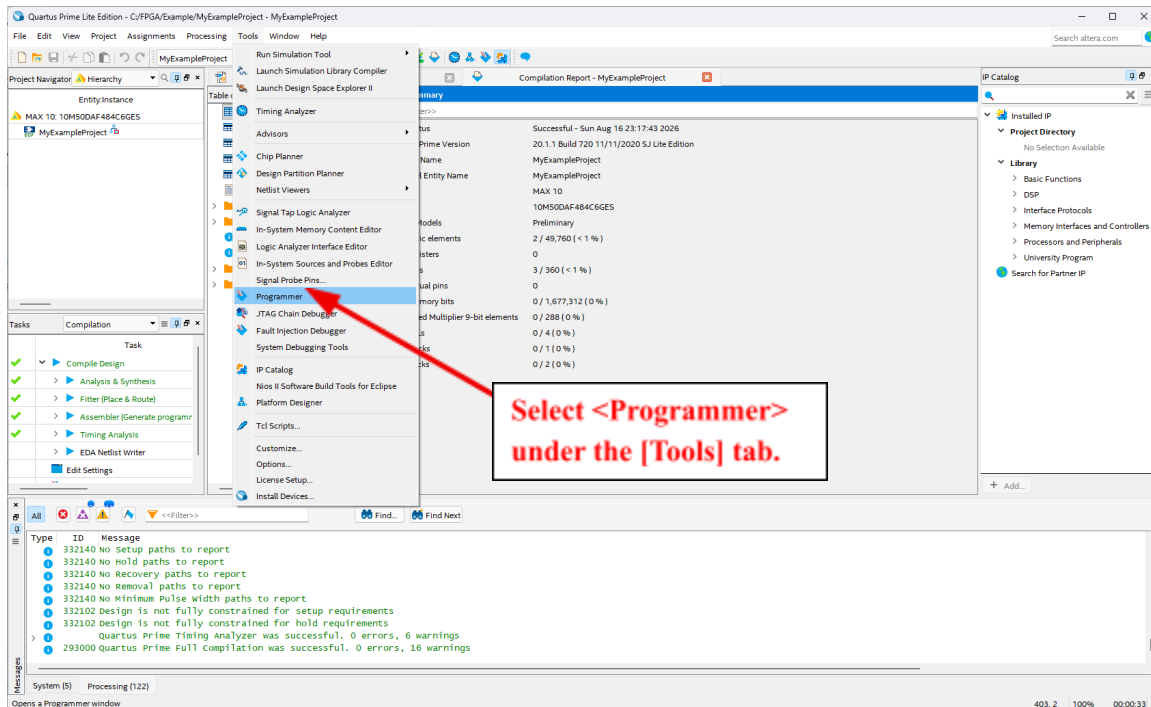


Note: if you instead created your circuit using Verilog and not by drawing a circuit schematic, you will still need to recompile your design with the pin assignments before downloading it into your FPGA board.

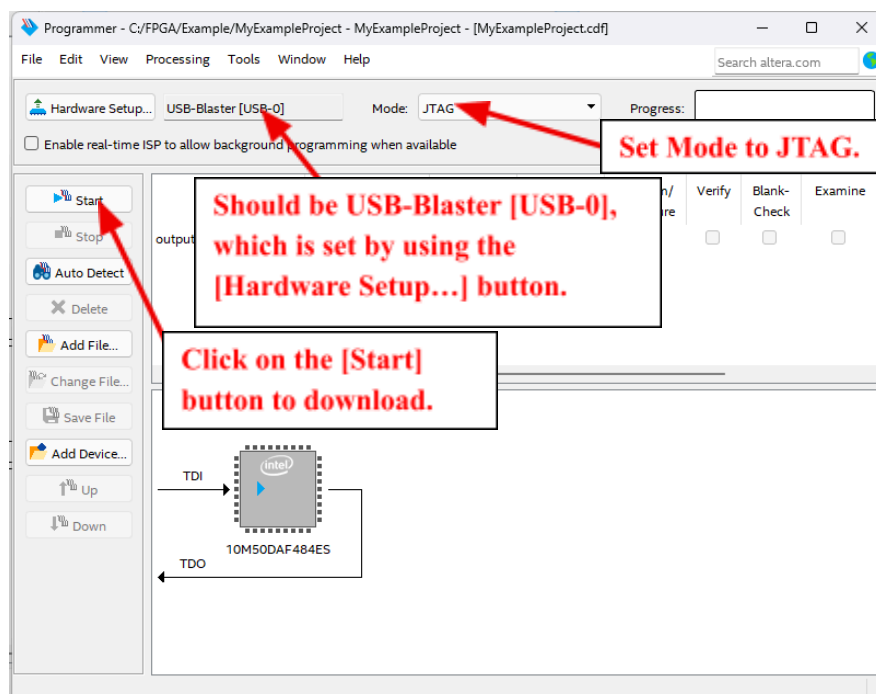
Now, the circuit is now ready to be downloaded into your FPGA board.

Downloading into FPGA:

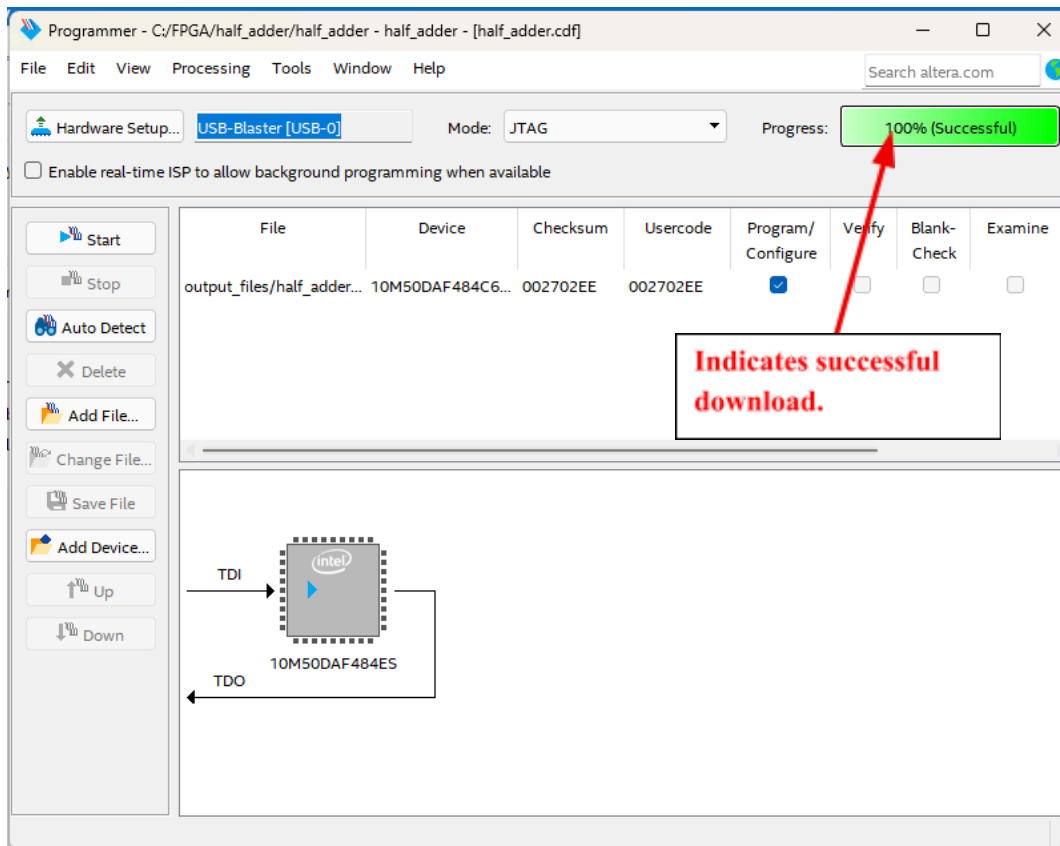
To start the process of downloading your compiled code into your FPGA board, the first thing you need to do is make sure that your FPGA board is connected to your computer via the USB A-to-USB B cable that came with your board. Then, you need to go to the [Tools] tab along the top toolbar and go to the <Programmer> option.



When you do, a new window opens, as shown below:



Once the hardware and mode are set correctly, you may press the [ Start] button to download your code into the FPGA. If everything is set and connected correctly, on your FPGA board the top left LED (green) under “LOAD” will briefly light up and then turn off while the “CONF” LED next to it lights up. Also, on your computer window the “Progress:” bar should turn green and show “100% (Successful)”.



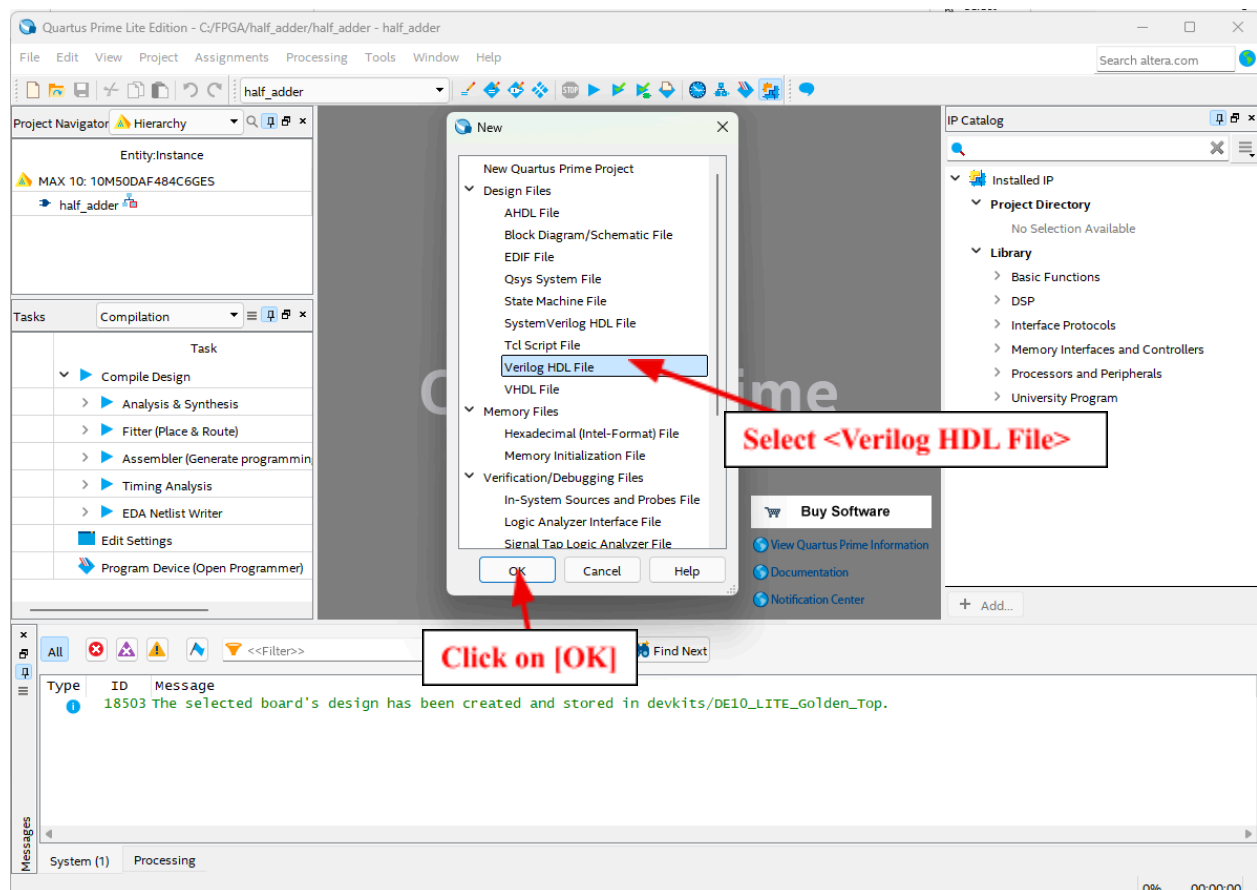
At this point, you should be able to use the input switches that you set and observe changes in the output LEDs based on the function you programmed your FPGA to perform. Note: if you did not do the pin assignments and then recompile before downloading, the download can still be “successful”, but the switches won’t do anything and the LED’s will not light up.

Depending on the instructions for your particular lab, you may need to demonstrate the operation of your FPGA to your instructor.

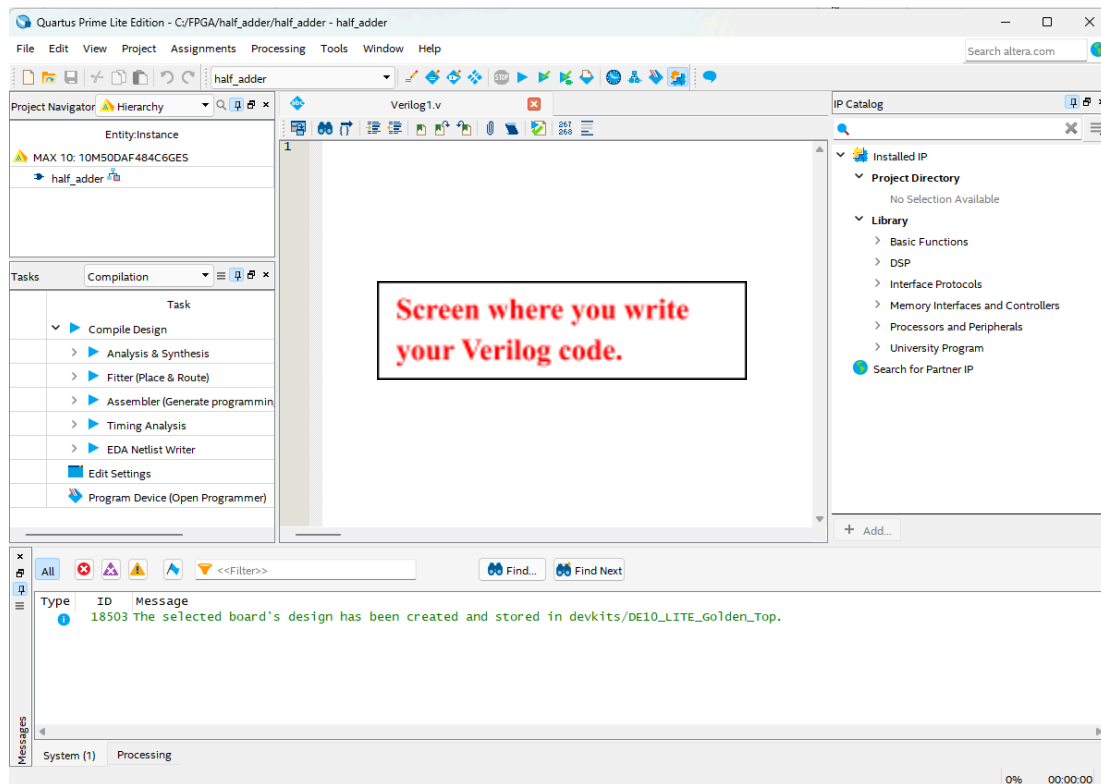
Writing Verilog Code:

Previously, you saw how to create a circuit schematic, simulate it, and then download it into your FPGA board. This section shows how to instead write a program in Verilog, which is similar to writing a program in Java, or C++, or Python. Like the circuit schematic method, it can be simulated and/or downloaded into your FPGA.

After creating a new project called “half_adder” (and possibly setting the default pin voltages), go to the [File] tab and select <New...> from the dropdown menu. The screen shown below will appear:



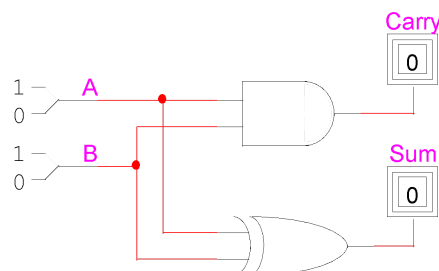
After clicking on [OK] the following screen appears:



Before we begin writing Verilog code, it is important to review the logic operators available:

Logic Function	Operator
NOT	~
AND	&
NAND	~&
OR	
NOR	~
XOR	^
XNOR	~^ or ^~

In your second FPGA lab, you will be creating a half-adder using Verilog, so we will use this as our example. While we will not be creating the circuit schematic in Quartus for a half-adder, it is useful to refer to the circuit schematic to implement a half adder:



Notice that the half-adder takes 2 input bits, (named A and B) and outputs their sum as a 2-bit answer, with Carry being the MSB (most significant bit) and Sum being the LSB (least significant bit). It uses an AND gate to determine the value of the Carry bit, and an XOR gate to determine the value of the Sum bit.

The Verilog code to implement this half-adder is:

```

module half_adder (A, B, Sum, Carry);

    input A, B;
    output Sum, Carry;

    wire Sum, Carry;

    assign Sum = A^B;    // bitwise XOR
    assign Carry = A&B; // bitwise AND

endmodule    // end of the half_adder module

```

Module header format:
“name (variables used);”
List inputs first.

Define variables as inputs or outputs

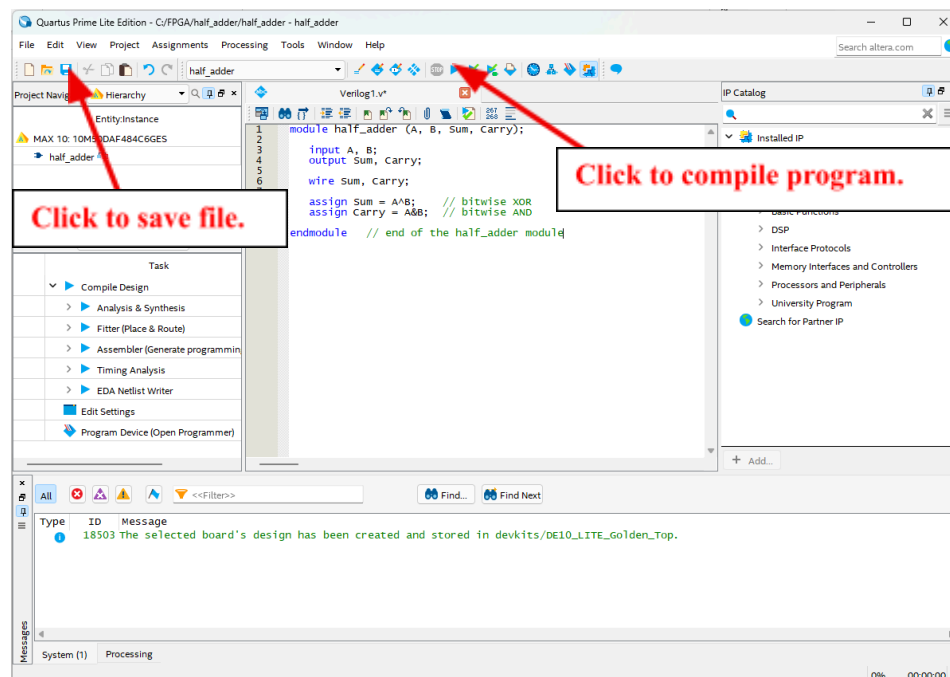
“wire” is an Output Data Type

Outputs assigned values based on inputs

When writing Verilog code, there are some rules/guidelines to consider:

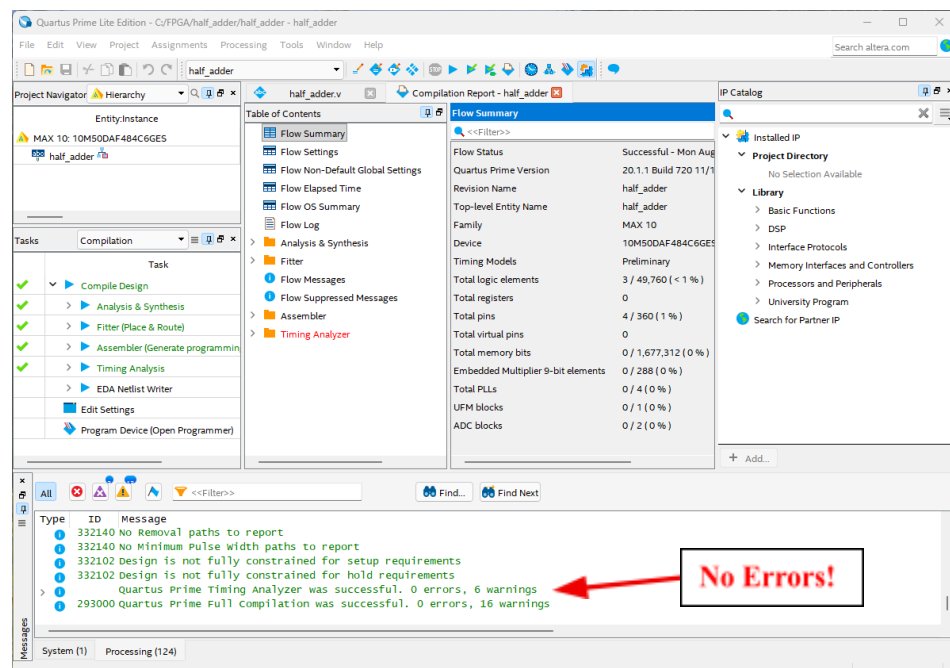
- 1) The code should be written as “modules” which act as functions in other languages. The first line should be the module header, and the last line should be “endmodule”.
- 2) The names of the module and variables should have no spaces.
- 3) The name of the module should be the same as the project name.
- 4) Verilog is case sensitive, and while variables may have capital letters, in general all commands are lower case.
- 5) Each command should end in a semicolon, (;). (Optional for “end” or “endmodule”.)
- 6) In the module header, the input variables should be listed before the output variables.
- 7) All of the variables listed in the module header should be declared as inputs or outputs.
- 8) Output variables need a data type declaration, which can be either “wire” or “reg”. The “wire” type depends continuously on the input values and is used for combinational logic. The “reg” (Register) type depends on previous values or state and is used for sequential logic. (Recall that unlike combinational logic, sequential logic refers to a digital circuit where the output depends not only on the current inputs but also on the history of inputs, stored as the circuit’s state; e.g. flip-flops.)
- 9) The “assign” statement associates an output variable to input variables via a logical function. Unlike other programming languages, these logical statements are not processed in sequential order but are implemented in parallel on the FPGA board.
- 10) Text after “//” on a line are comments and are not part of the compiled program.
- 11) Indentation is optional but encouraged for readability.
- 12) If a command uses more than one subcommand, such as a list of commands to be performed under an “if conditional”, then the subcommands should be grouped with a “begin” and “end”. (This was not used above but will be demonstrated in future FPGA lab exercises.)

Now type or copy the above Verilog code into the empty Verilog window, i.e.:

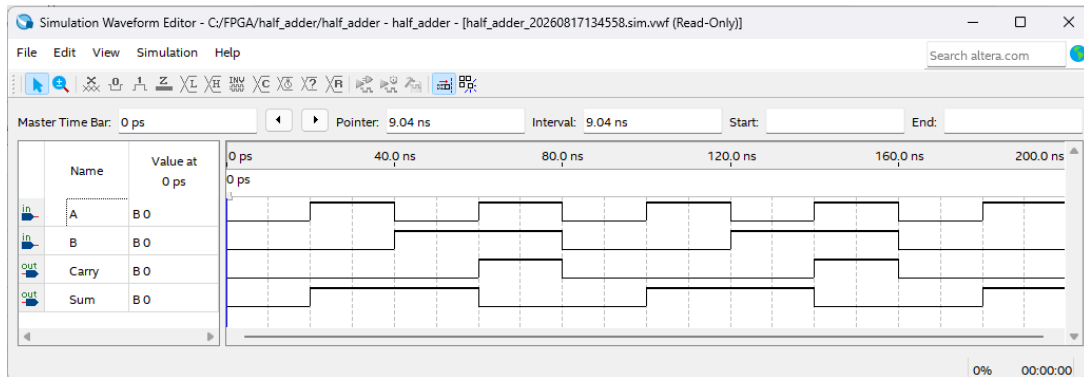


Notice in Quartus that the commands become blue text and comments become green text.

Once the code appears correct, save the file as “half_adder.v”, which should be the same name as the project. (The file should have a “.v” file extension. Notice that in the screen above, the file has a “*” after its name, which means that the latest version isn’t saved. Once saved, the “*” disappears.) Then to test the code, compile it by pressing the [▶] button. If there are no errors, you should get a screen similar to:



Now that you have a compiled Verilog program with no errors, you can do the same operations you did with your circuit schematic. First, you can simulate the waveforms (don't forget to remove the "novopt" from the simulation options). When done correctly, you should see the following waveforms:



Before downloading your code into your FPGA board, you will need to do a pin assignment. One possible layout is:

- Input A using switch SW1, (PIN_C11).
- Input B using switch SW0, (PIN_C10).
- Output Carry using LEDR1, (PIN_A9).
- Output Sum using LEDR0, (PIN_A8).

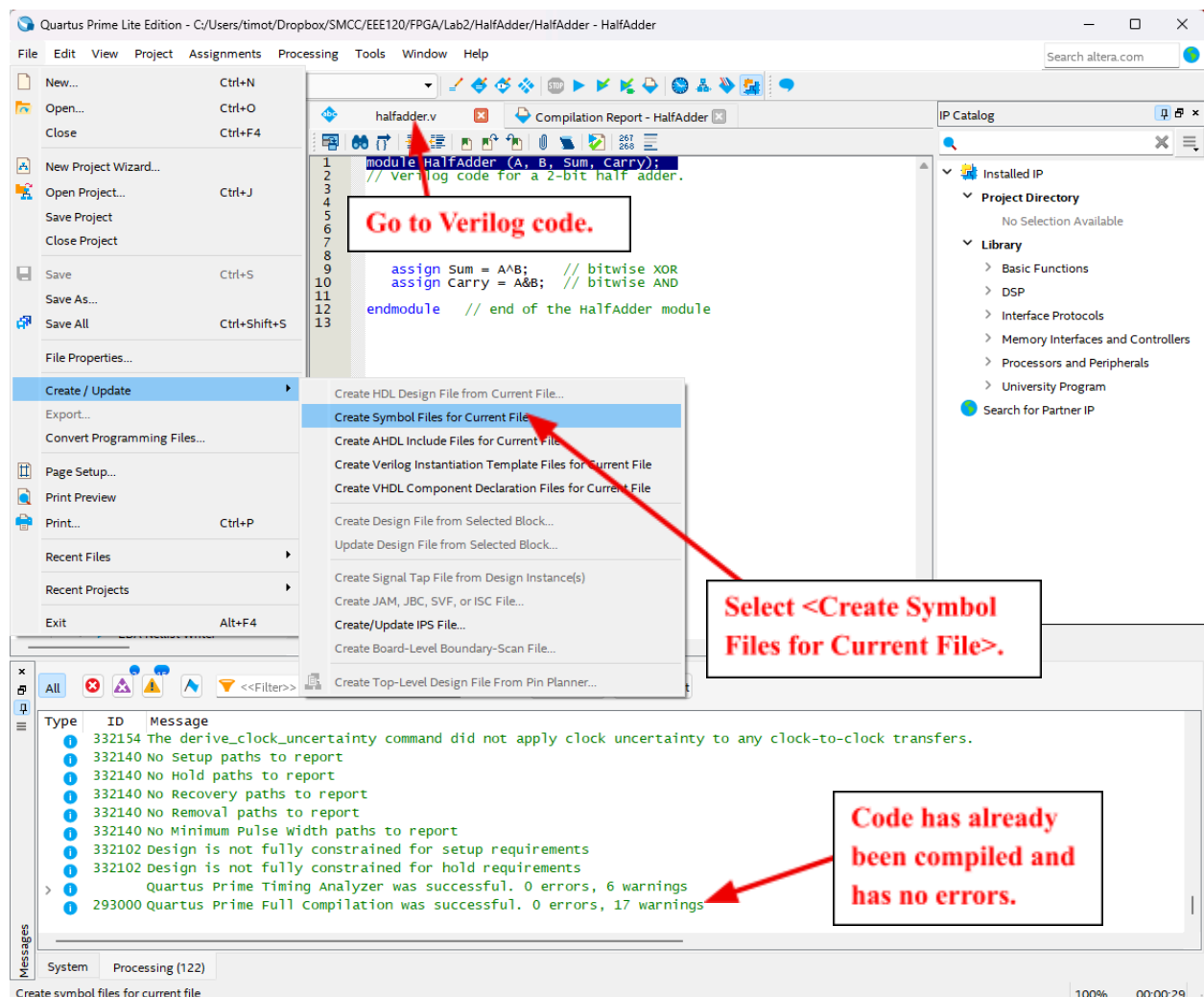
Node Name	Direction	Location	I/O Bank	VREF Group	Filter Location	I/O Standard	Reserved	Current Strength	Slew Rate
A	Input	PIN_C11	7	B7_NO	PIN_Y1	3.3-V LVTTL (default)		8mA (default)	
B	Input	PIN_C10	7	B7_NO	PIN_U6	3.3-V LVTTL (default)		8mA (default)	
Carry	Output	PIN_A9	7	B7_NO	PIN_W6	3.3-V LVTTL (default)		8mA (default)	2 (default)
Sum	Output	PIN_A8	7	B7_NO	PIN_V4	3.3-V LVTTL (default)		8mA (default)	2 (default)
<<new node>>									

Before downloading the program into your FPGA board, don't forget to recompile it after making the pin assignments. Otherwise, you won't get an error message, but your FPGA just won't do anything. After being compiled and downloaded into your FPGA, you should be able to move the switches and verify the operation of your half-adder.

Creating Device Symbols

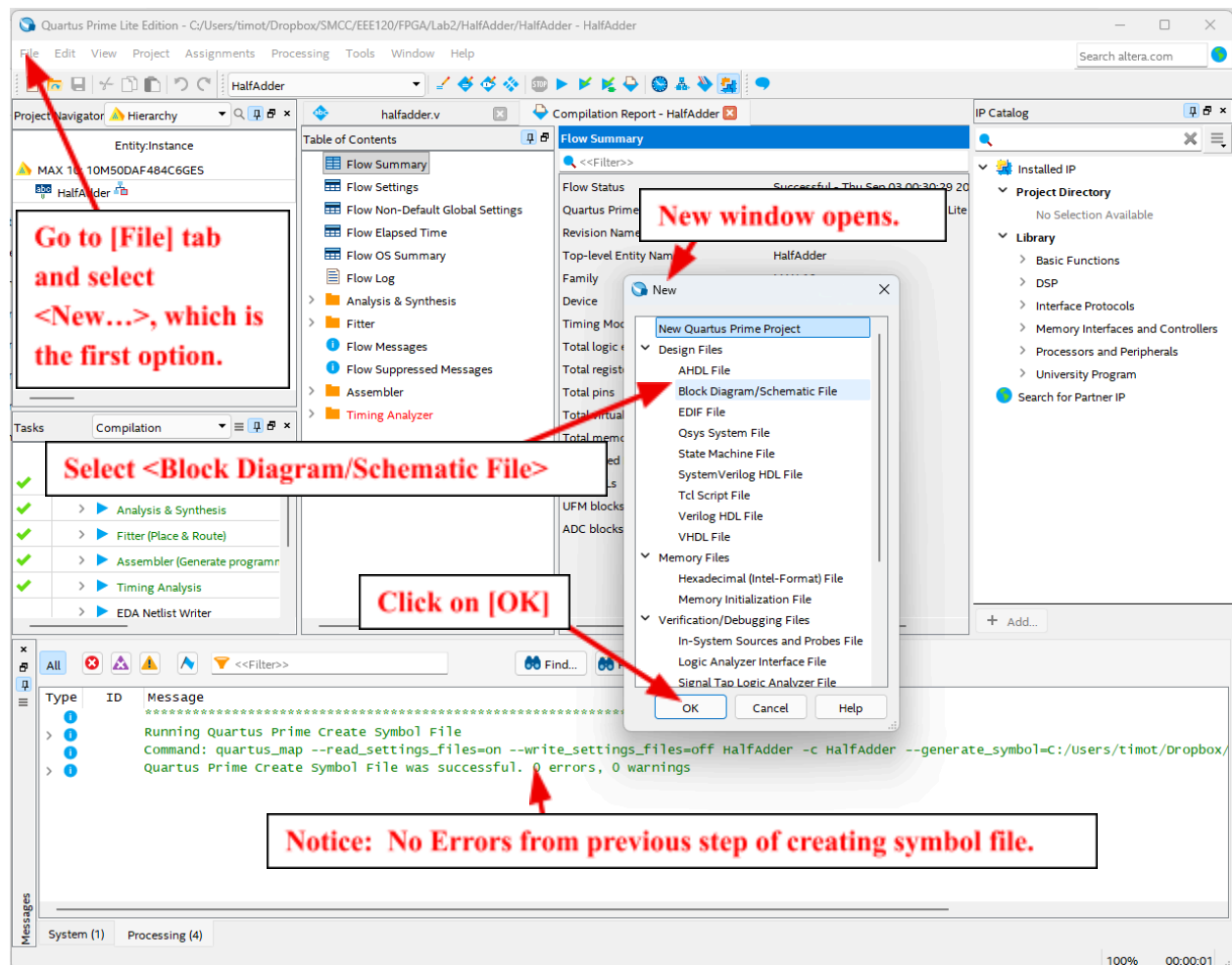
In LogicWorks, you were able to create new device symbols from circuits you had created. You can basically do the same thing in Quartus, but the device does not need to be just created from basic logic gates. It can also be created from Verilog code, which will be the focus of this section. These devices can then be connected with other devices to create larger, more complex circuits.

After successfully writing Verilog code and verifying that there were no errors, go to the tab of *.v code, so that you see your code. (You will NOT want to assign pins before generating the device symbol. If you have already assigned them, you may go back to delete the pin assignments individually and then recompile your code. If the pins are assigned first, the generated symbol will include the pin assignment, so you are not able to assign the input/output pins when your new device symbol is connected within a circuit. In general, you only want to assign pins to the top-level design that you will be downloading into the FPGA board.) Then go to the [File] tab and select <Create/Update> ⇒ <Create Symbol Files for Current File> from the dropdown menus. The screen shown below will appear:



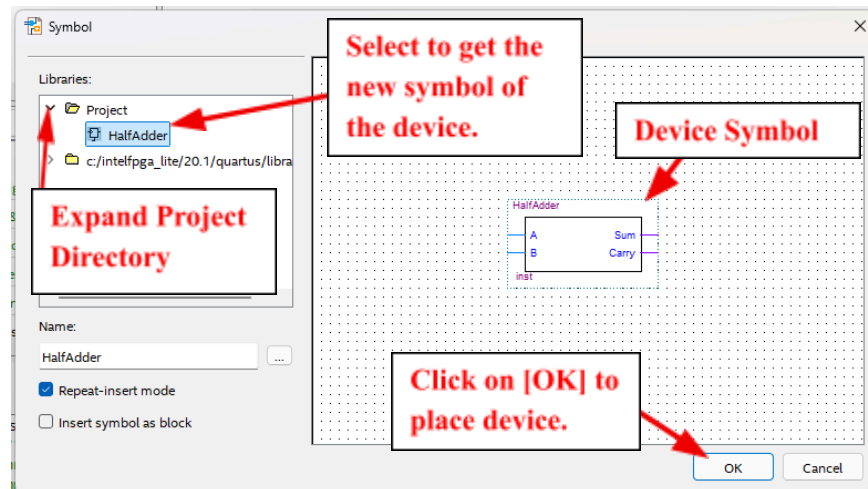
After selecting the option of <Create Symbol Files for Current File>, you should get a message in the bottom window that says “Quartus Prime Create Symbol File was Successful. 0 errors, 0 warning”, (different message than the “0 errors” from compiling the code). If you go to the folder containing the project files, you should see a new file with the *.bsf extension. This is the file which contains the new circuit device symbol. (If you want to use this symbol along with others, you will need both the *.bsf and *.v files, which can be copied to a folder containing these same file types for the other parts you want to also use in your new circuit schematic.)

To use the new device symbol, first open a new schematic file (*.bdf file):

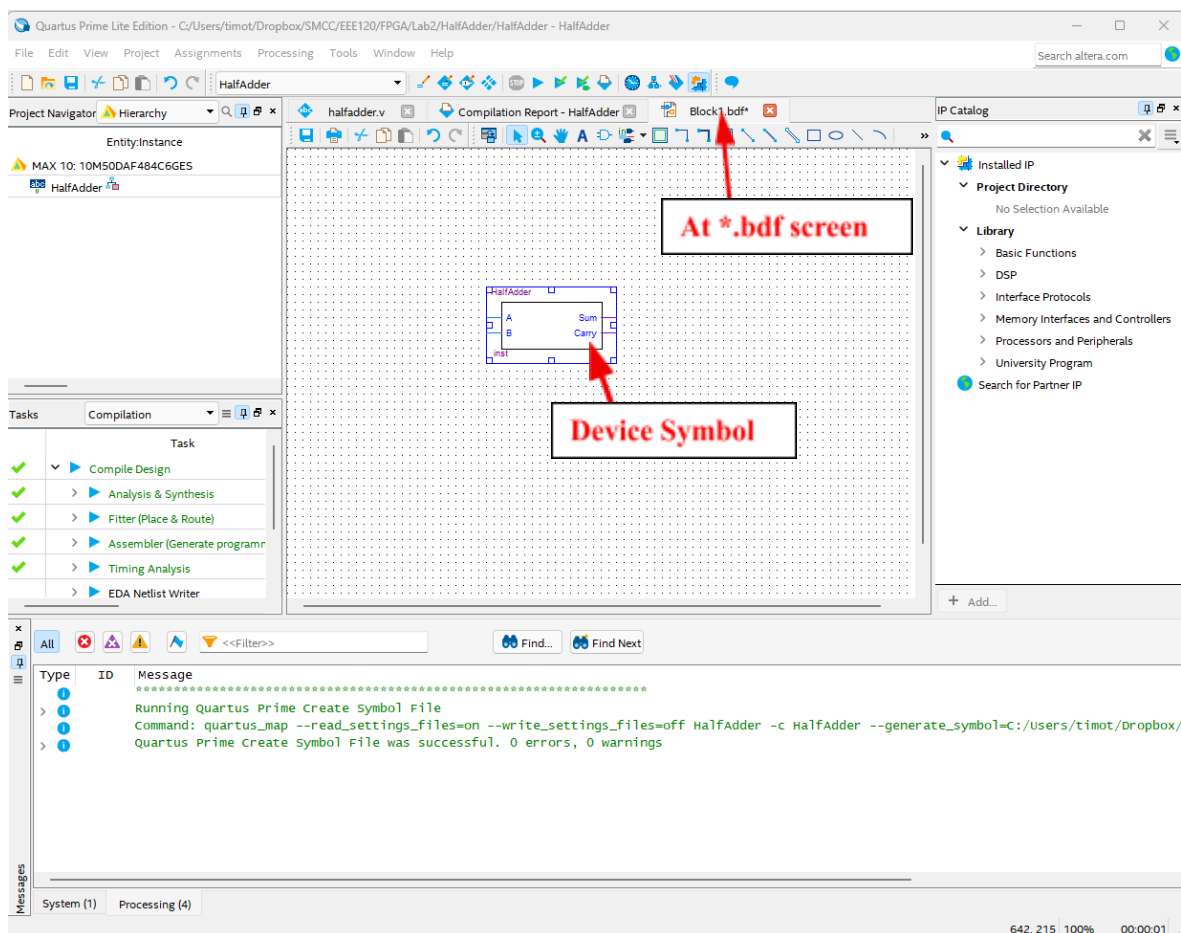


Note: the “.bdf” file extension refers to a “block diagram file,” while the “.bsf” file extension refers to a “block symbol file.”

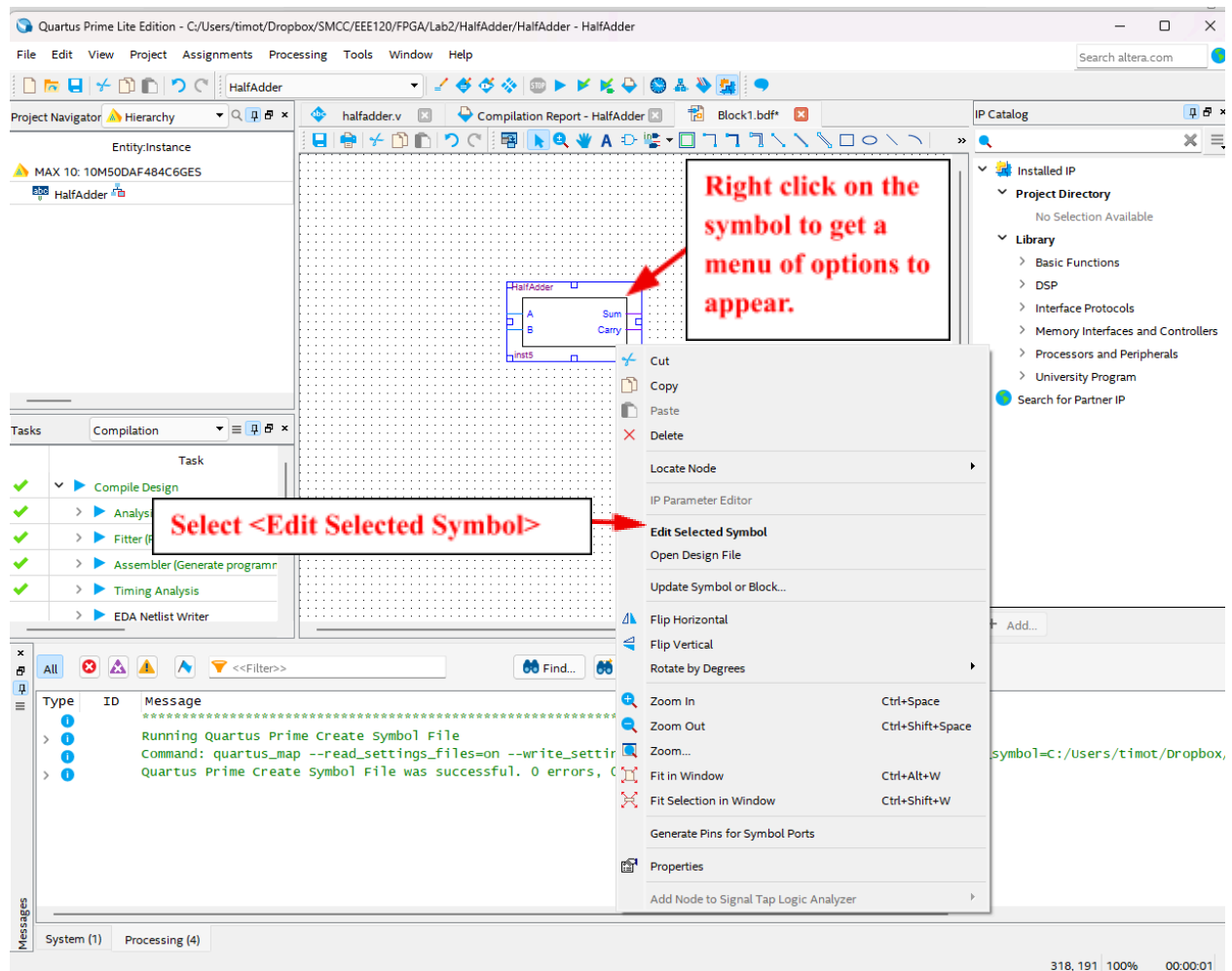
A new screen will then appear within Quartus, showing grid dots and labeled “Block1.bdf,” (see previous section of “Creating a Circuit Schematic” within this document for more details). As before, click on the symbol tool icon () and the Symbol window appears:



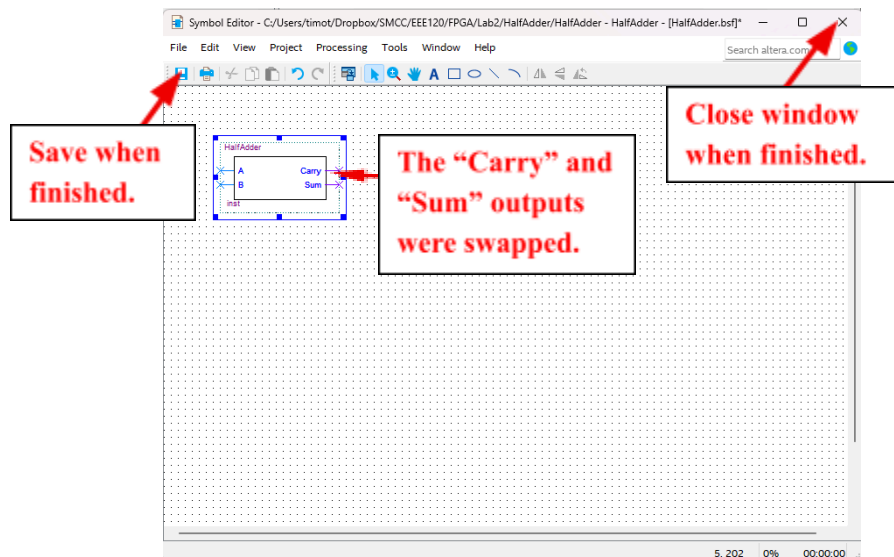
After clicking on [OK], the Symbol window will close and you will be able to place the new part in “Block1.bdf” screen, much like how you can create a circuit schematic from basic logic gates. Below shows the device symbol placed within the “*.bdf” screen:



While the screen above shows the default device symbol created by the software, it is also possible to edit its appearance. By right clicking on the device symbol, a new dropdown list of options appears, with one of the options being <Edit Selected Symbol>.

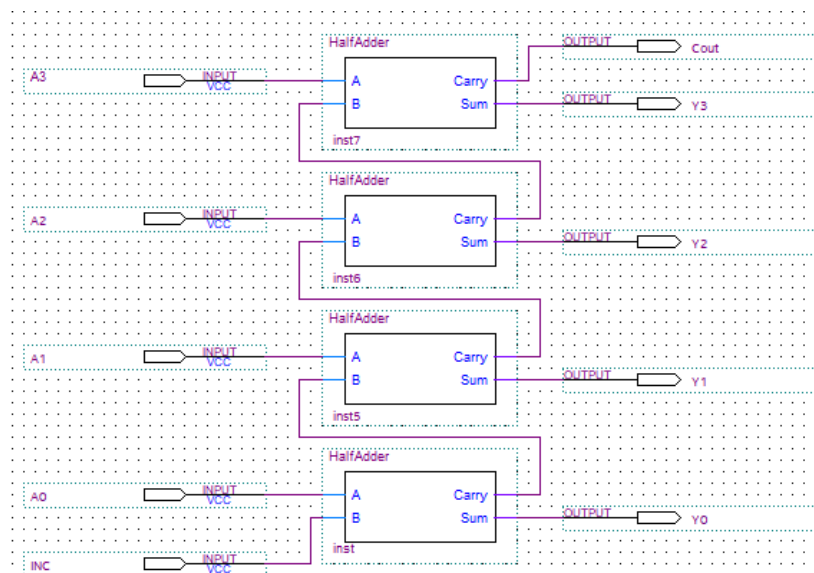


When that option is selected, a new “Symbol Editor” window appears with the device symbol shown. At this window, it is possible to change the size of the device or move the pins and labels to new locations. In this example, the order of the “Sum” and “Carry” outputs were swapped so that “Carry,” which is the MSB appears on the top.

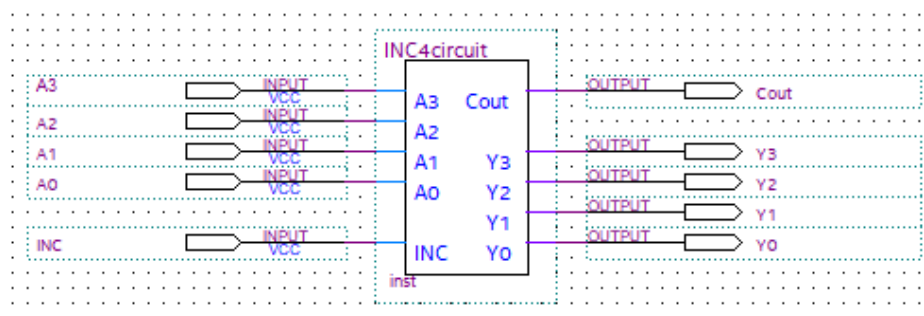


Note: while it is possible to change the name of an output on the device symbol, since the output is connected to a specific port through the Verilog code, you will want to physically swap the location of the output pins and not just rename them. When finished, simply save the new device symbol and close the window. The original symbol will still appear in the *.bdf screen, but you may delete it and get a new symbol as before via the symbol tool icon () above the screen. Now when you select the device, the new symbol will appear.

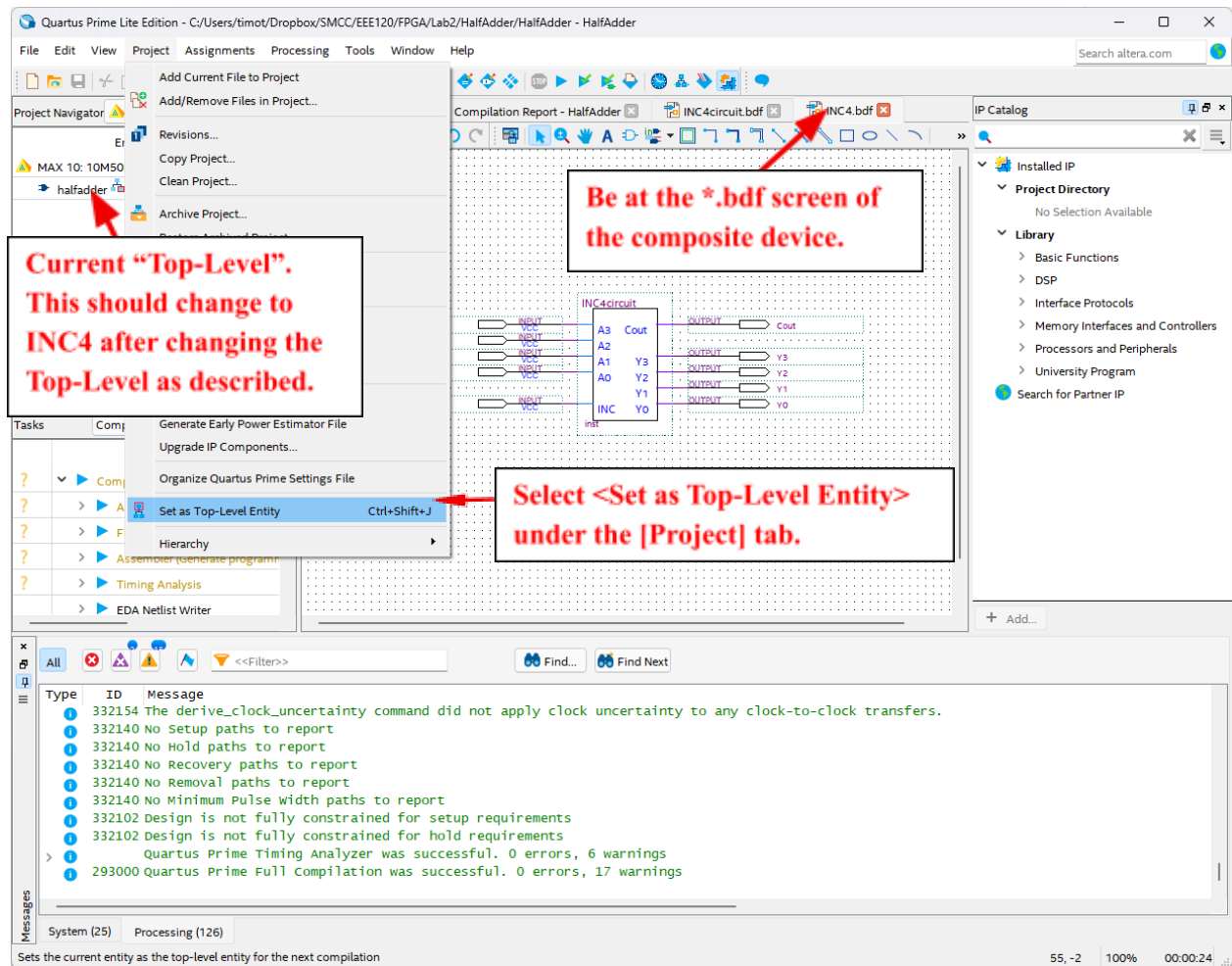
Once you have the device symbol, it is now possible to connect it to other devices or add inputs and outputs. For example, in this section we have been working with a half-adder device. As in Software Lab A, where 4x half-adders were connected together to create an INC-4 circuit in LogicWorks, we can do the same thing in Quartus. I.e., we can create the circuit shown below:



It is then possible to compile this circuit and create another device symbol from it, much like the INC-4 part was created in LogicWorks. The above circuit was saved as “INC4circuit.bdf,” compiled, and then a new device symbol file was created, called “INC4circuit.bsf.” The symbol generated was then edited to increase the pin spacing between the INC input pin and the A0 pin and also the Y3 output pin and the Cout pin. Finally, inputs and outputs were added to the schematic and labeled, producing the circuit of:



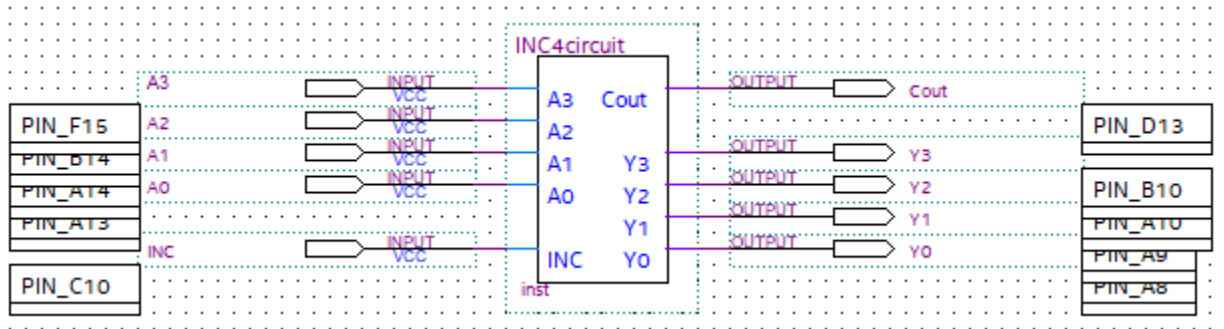
To test this circuit out by downloading into the FPGA board, you will need to assign pins to the inputs and outputs. However, if you just go to the Pin Planner option, most likely you will only see the inputs/outputs on the original half-adder circuit, (A, B, Sum, and Carry). To be able to assign pins to your new INC4 circuit, you will first need to save the circuit schematic with the new symbol and the inputs and output attached, (“INC4.bdf”), and then while at the *.bdf screen, go to the [Project] tab and select <Set as Top-Level Entity> as shown below:



Then recompile the program. Now, when you go to the Pin Planner, all of the pins in the INC4 circuit should be listed. You can test out the INC4 circuit on your FPGA board by assigning the pins as shown below:

Node Name	Direction	Location	I/O Bank	VREF Group	Fitter Location	I/O Standard	Reserved	Current Strength	Slew Rate	Di
A0	Input	PIN_A13	7	B7_NO	PIN_A13	3.3-V LVTTTL		8mA (default)		
A1	Input	PIN_A14	7	B7_NO	PIN_A14	3.3-V LVTTTL		8mA (default)		
A2	Input	PIN_B14	7	B7_NO	PIN_B14	3.3-V LVTTTL		8mA (default)		
A3	Input	PIN_F15	7	B7_NO	PIN_F15	3.3-V LVTTTL		8mA (default)		
Cout	Output	PIN_D13	7	B7_NO	PIN_D13	3.3-V LVTTTL		8mA (default)	2 (default)	
INC	Input	PIN_C10	7	B7_NO	PIN_C10	3.3-V LVTTTL		8mA (default)		
Y0	Output	PIN_A8	7	B7_NO	PIN_A8	3.3-V LVTTTL		8mA (default)	2 (default)	
Y1	Output	PIN_A9	7	B7_NO	PIN_A9	3.3-V LVTTTL		8mA (default)	2 (default)	
Y2	Output	PIN_A10	7	B7_NO	PIN_A10	3.3-V LVTTTL		8mA (default)	2 (default)	
Y3	Output	PIN_B10	7	B7_NO	PIN_B10	3.3-V LVTTTL		8mA (default)	2 (default)	

This will assign the 4x A-input bits to the leftmost switches on your FPGA board; the INC input to the rightmost switch; and the 5x output bits to the leftmost LEDs above the switches. As discussed in the “Assigning Switches and LEDs” section of this document, close the Pin Planner window and recompile the circuit schematic. After doing this, the schematic should now show the assigned pins by the input/output names:



Then use the <Programmer> option under the [Tools] tab to download the circuit into the FPGA board and test it out. With the rightmost switch down, the LEDs should light up based on the position of the 4 leftmost switches. Then when the rightmost switch is pushed up, the number represented by the LED should increase by one.