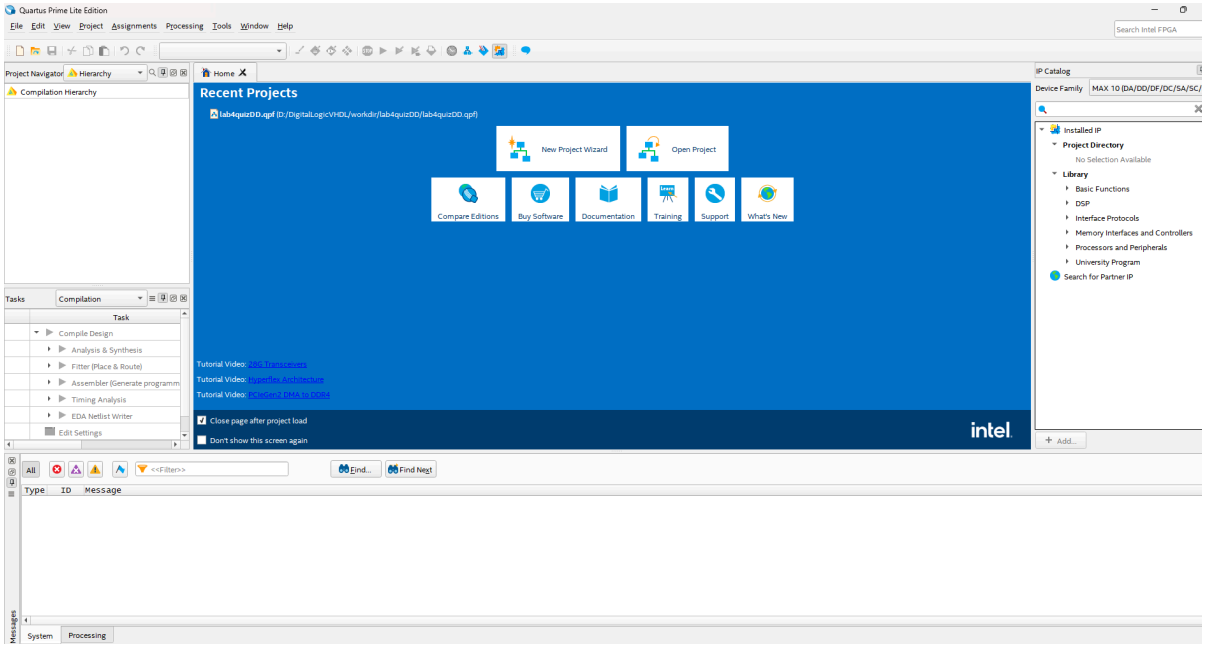
Herbert Wertheim College of Engineering
Department of Electrical & Computer Engineering
UNIVERSITY of FLORIDA

Dr. Christophe Bobda
Lucas Aguirre & Lucas Perez
EEL3701C - Fall 2025

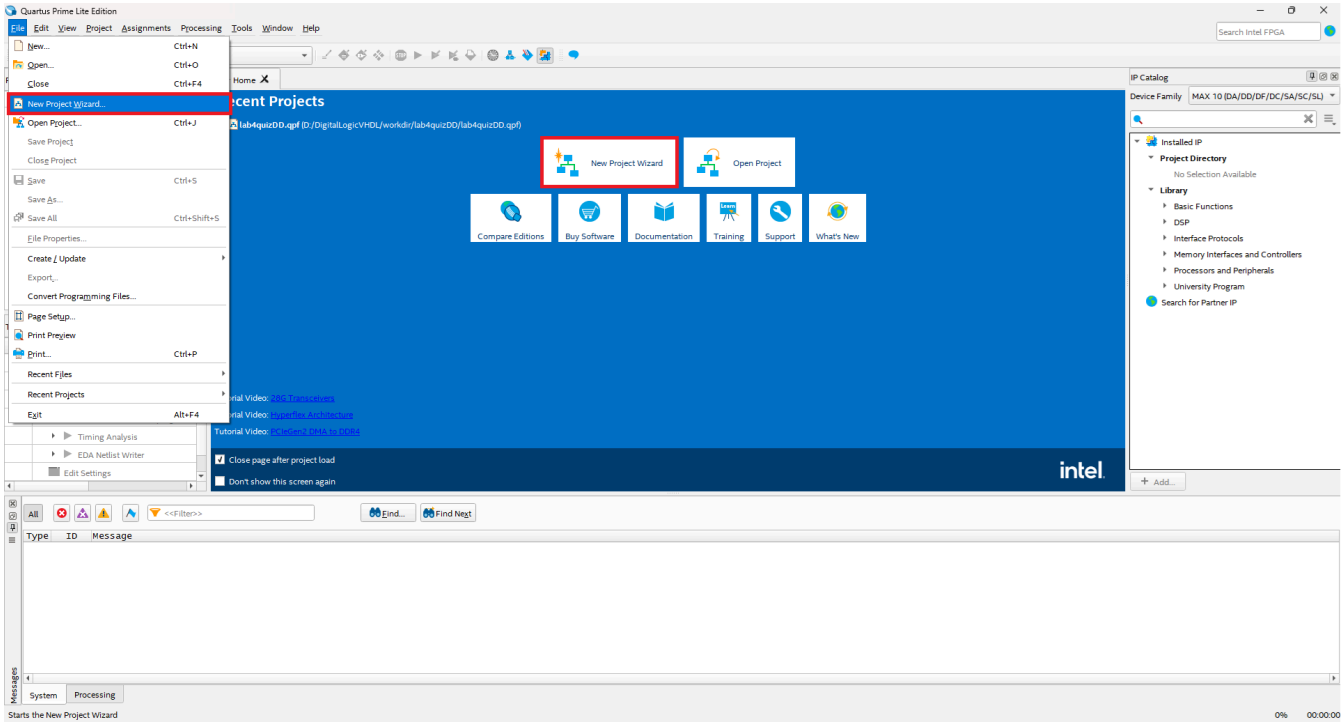
Quartus Tutorial

Creating a Project - Walk Through

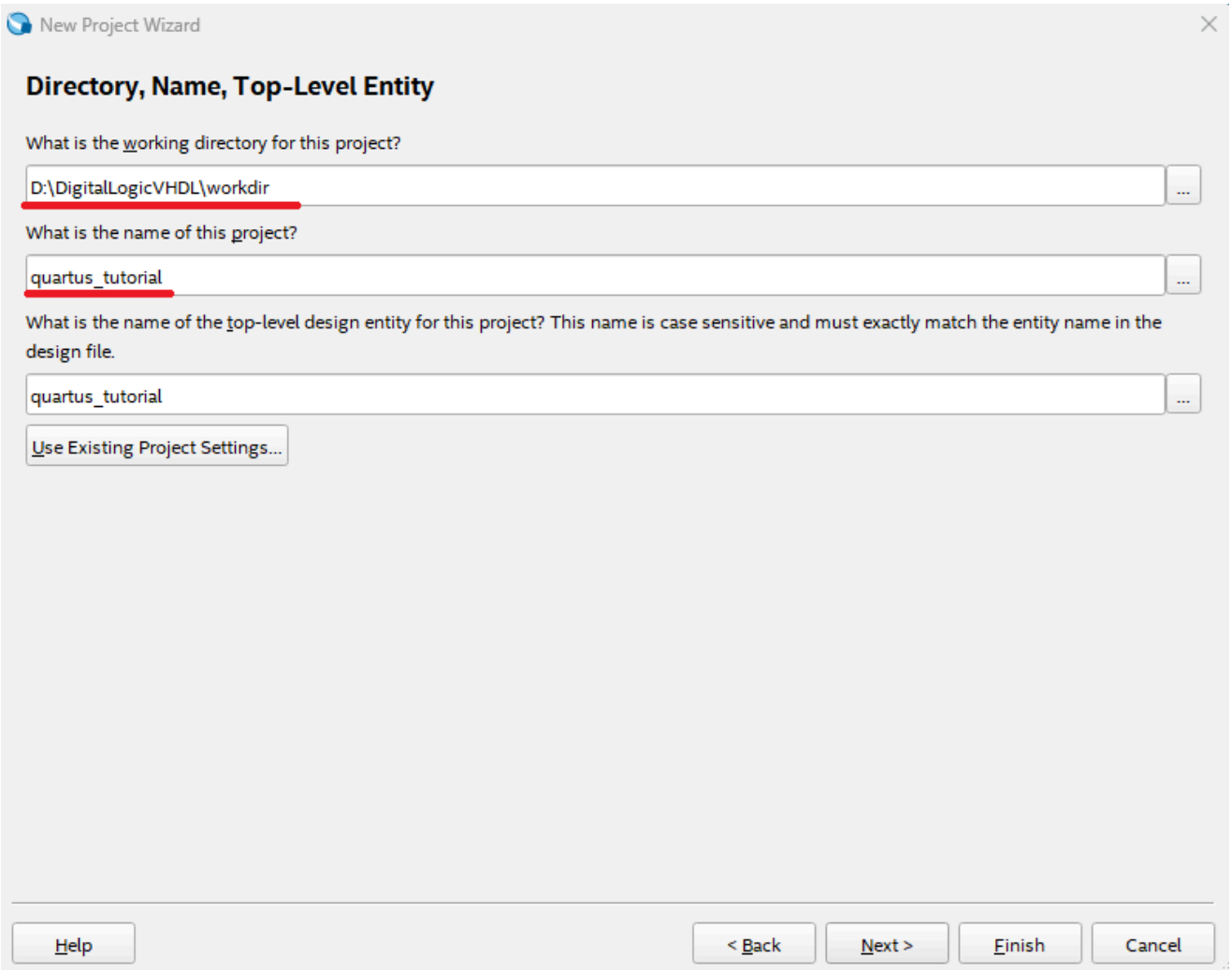
1. Open the Quartus prime software previously installed. You will be met with a screen similar to the screenshot below.



2. Click File on the top left and click “New Project wizard” or if you have the Home screen on startup, click New Project Wizard under Projects.

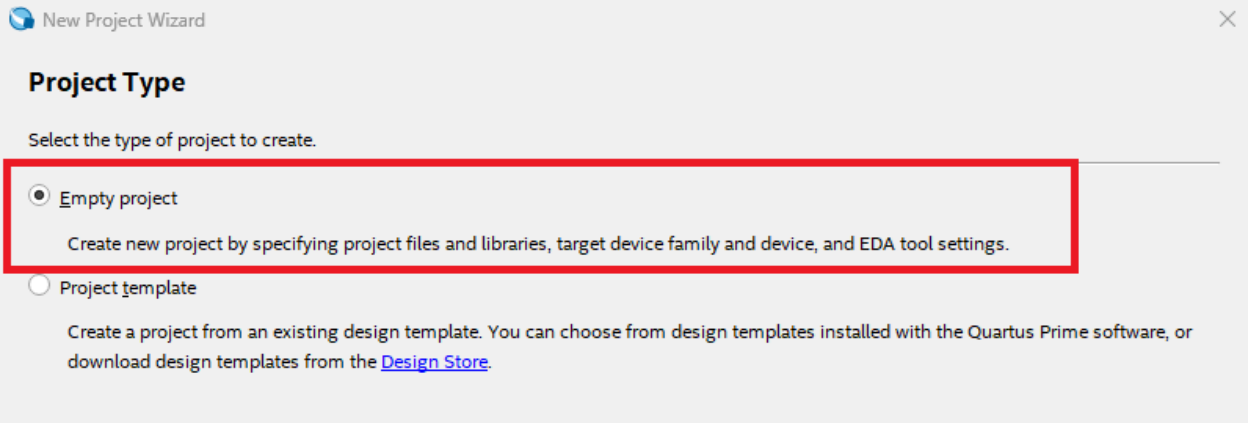


3. Click next and then choose a working directory for your projects. If this is your first project, you must *create* a working directory. It’s recommended to have your working directory somewhere that you can easily access it. Then, define a name for your project **(MAKE SURE THERE ARE NO SPACES IN THE FILE PATH) ->** Click next.

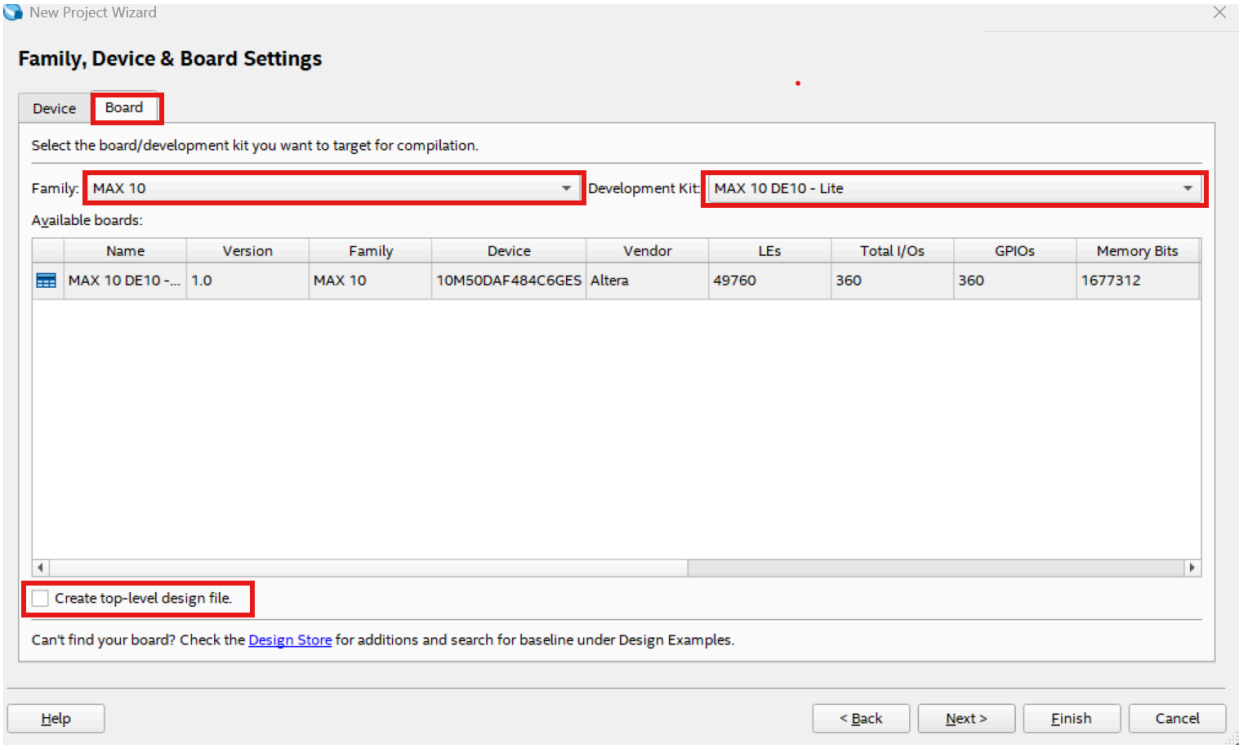


The image above shows a user defined working directory and project name.

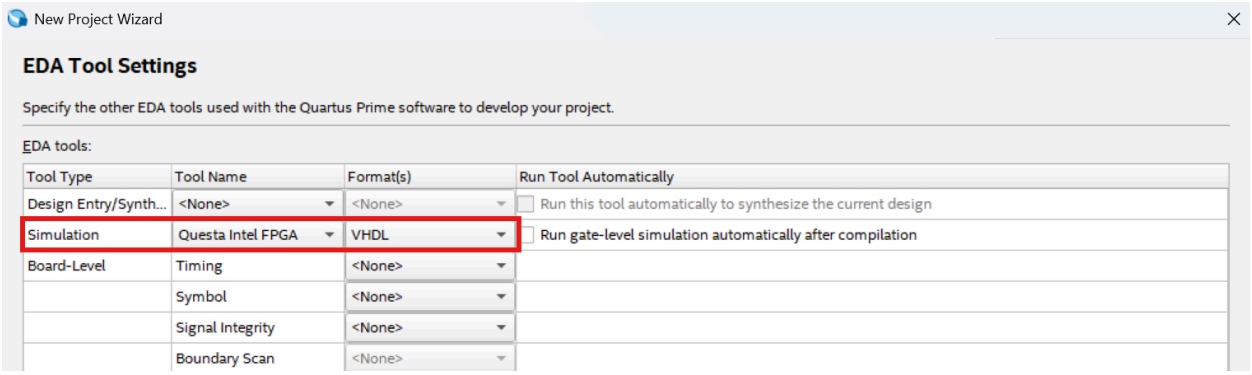
4. Select Empty Project and click next **twice** (skipping add files page).



5. On the Family, Device, and Board settings page select **Board -> Development Kit -> Select Max 10 DE10 - Lite**. Then unselect “Create Top-Level Design file.” Double check that Family is MAX 10.



6. In EDA Tools, Select Simulation tool type “Questa Intel FPGA” and change format to VHDL.



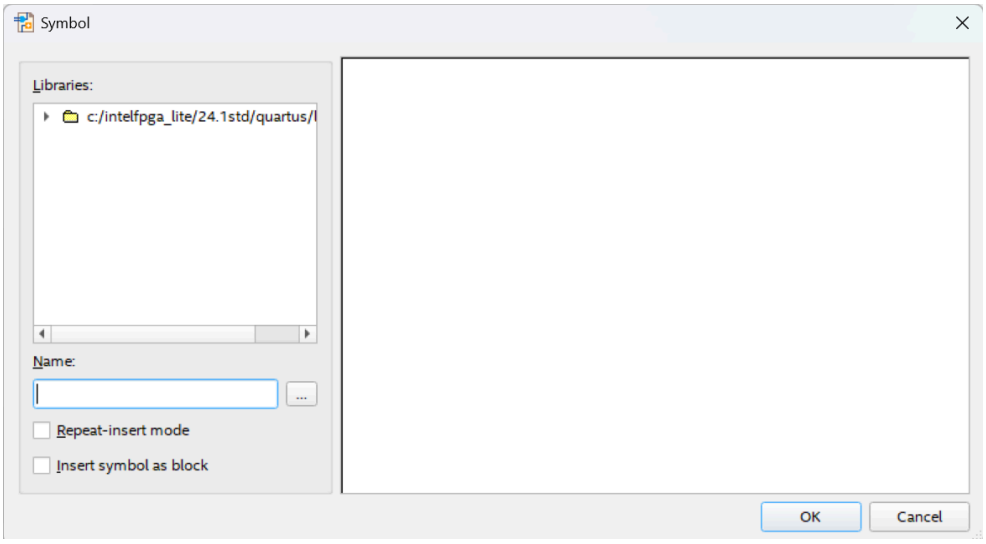
7. Click next and then Finish.

BDF Example - Compilation and Synthesis:

In this example, we will be using a Block Diagram File to create and simulate a XOR gate.

1. After creating a new project, we can create a new BDF by clicking **File→New→Block Diagram/Schematic File**. This will create a blank canvas for us to design our circuit on.

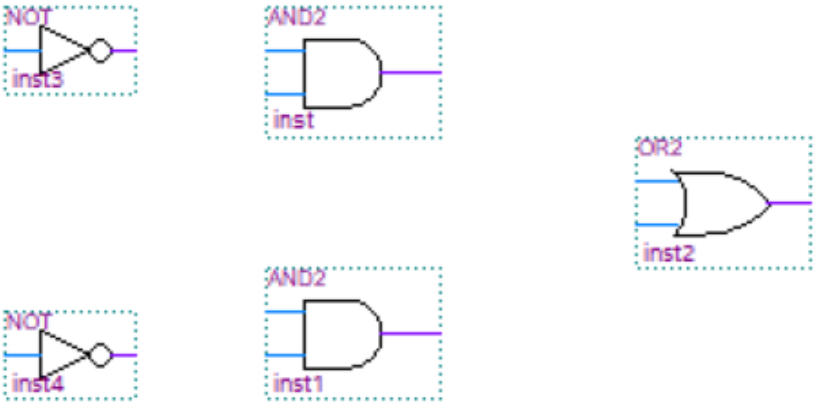
2. You can add components by **double clicking** on the page or **right clicking** and hovering over **insert** and clicking **symbol**. You'll be met with the window below



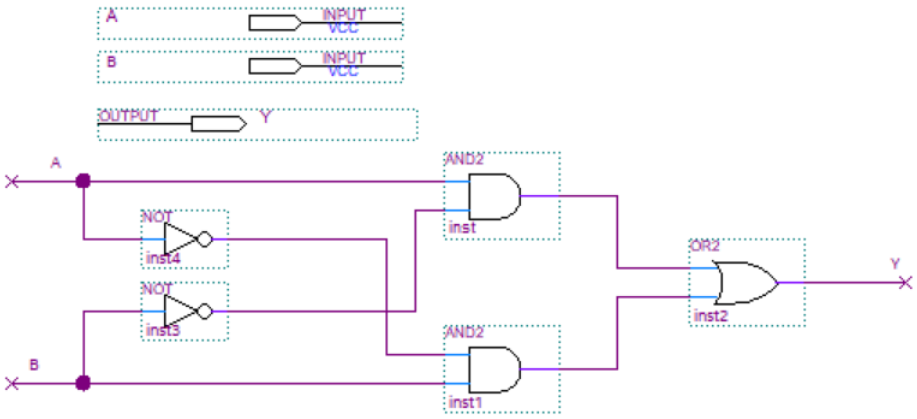
3. For this example we are doing a XOR (Exclusive OR) gate. We will need the following gates. Under **Name:** search for the following gates (see side note)
    - 2 → 2 input AND gates
    - 1 → 2 input OR gates
    - 2 → NOT gates

Side note: When searching for gates, they have the naming convention of and2, and4, or2 and so on. Basically the name of the gate followed by the number of inputs it takes.

4. Place these gates on the block diagram canvas similar to the image below (more complex circuits could get messy so try and line gates up to keep it clean)

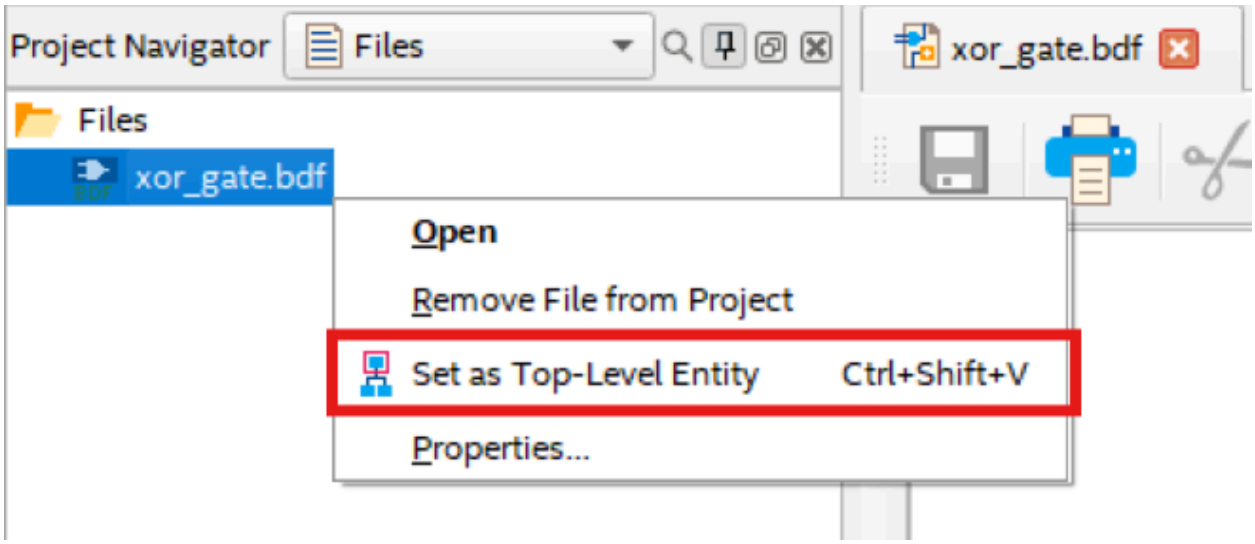


5. We can then connect these gates together as well as give them input and output pins for simulation purposes (Look up “Input” and “Output” the same way we did for the gates). See below

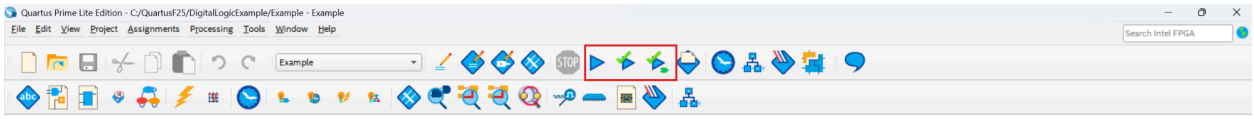


a. As you can see above, we added 2 input pins, and an output pin. These can be renamed to anything you want but for this example we will use A, B, and Y
b. Each wire is named by **left clicking** then typing the name that corresponds with the pin. To rename pins, **double click** then type the name you want

6. Once we are done wiring everything up like the above picture, we can save this BDF by clicking **File→Save**. Once you are prompted to name your BDF, use the name xor_gate.bdf
7. To compile and synthesize the circuit that we created, we need to set our BDF as a *"Top Level Entity"*. We can do this by clicking **Project Navigator →Files** then right clicking "**xor_gate.bdf"** and selecting **"Set as Top-Level Entity"**



8. Now that the xor_gate.bdf is the Top-Level Entity, we have a couple options for compilation. These are **Full Compilation, Analysis & Elaboration,** and **Analysis & Synthesis.** The three buttons below execute the three compilations in order from left to right.



This part of the process is identical for both VHDL and BDF files. To not be redundant, reference the VHDL documentation on compilation and programming on *page 9* and continue reading

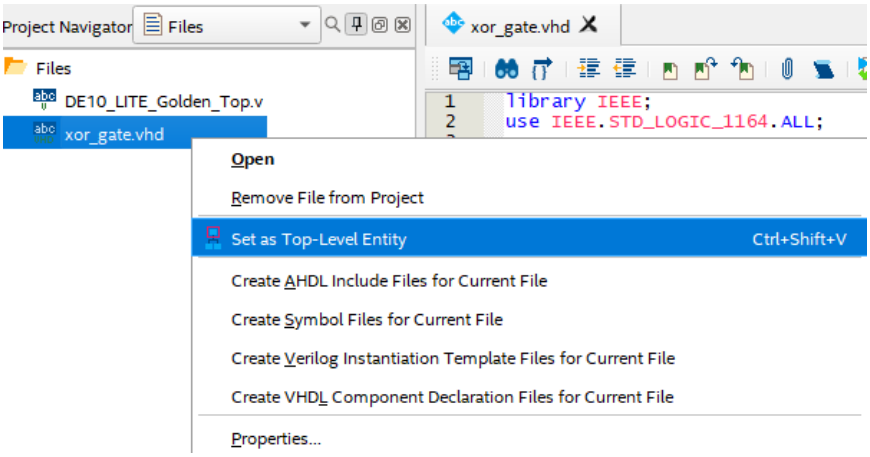
VHDL Example - Compilation and Synthesis:

In this example, we will follow the BDF and create a XOR gate in VHDL.

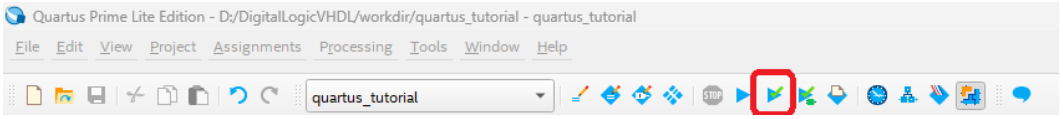
1. Writing the VHDL Code
    a. First, we will create a new **VHDL** file under **File→New→Design Files→VHDL File**
    b. The following code block below will be used to generate the XOR gate design.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity xor_gate is
    Port (
        A : in STD_LOGIC;
        B : in STD_LOGIC;
        Y : out STD_LOGIC
    );
end xor_gate;

architecture Behavioral of xor_gate is
begin
    Y <= (NOT A AND B) OR (A AND NOT B);
end Behavioral;
```

    c. Once this code block is inserted into the new VHDL file, we will save the file under **File→Save**. When prompted to name the file, use the name **xor_gate.vhd.**
2. To compile this code, we must set **xor_gate.vhd** as the *Top-Level Entity* by right-clicking the file and selecting **"Set as Top-Level Entity"** as shown below. If you do not see any files, click the *dropdown box* next to the **"Project Navigator"** text and choose the option **"Files"**. If you find any extra files like shown in the screenshot, you're free to remove them by right-clicking the file and selecting *"***Remove File from Project"**.

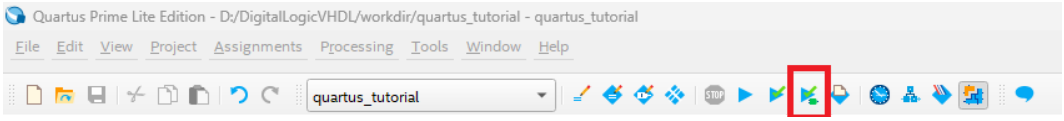


a. There are three levels of compilation within Quartus used to design RTL, **Analysis & Synthesis, Analysis & Elaboration,** and **Full Compilation**. We will go through all three versions of compilation when building this project, the first one we will use is Analysis & Elaboration. In order to perform an Analysis & Elaboration compilation we will click the "start" arrow with a green checkmark highlighted in the image below.



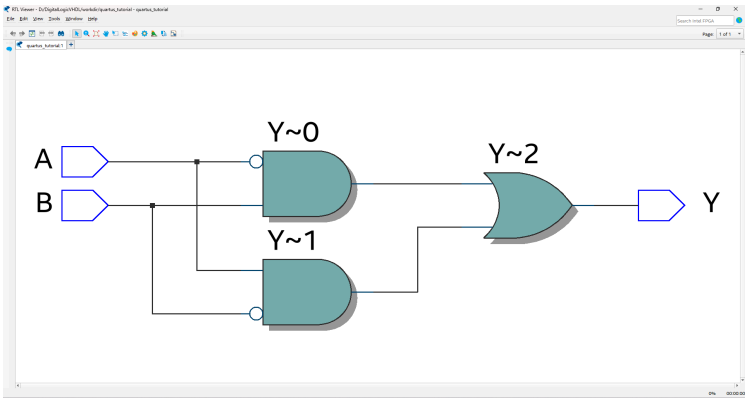
When running Analysis & Elaboration, the compiler will check our HDL source files for both syntax and semantic errors, build the design hierarchy of our project and generate a full structure of our design to ensure all modules/components can be properly instantiated. It's best to use Analysis & Elaboration early for design validation *before* committing to a complete hardware synthesis run.

b. After a successful **Analysis & Synthesis** of our project, we can run an Analysis & Synthesis compilation, denoted by the "start" arrow with a green checkmark and green component under it, highlighted in the image below.



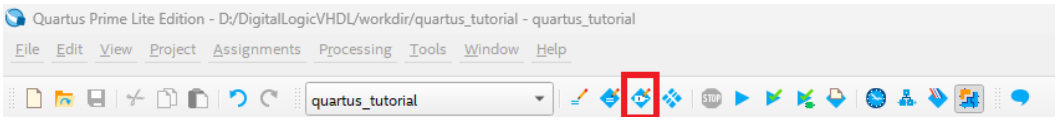
When running Analysis & Synthesis, Quartus will perform a deeper analysis that includes both parsing our HDL code and fully synthesizing our design to be converted into a technology-mapped netlist that reflects the resources available on our FPGA device (DE-10 Lite). This will prepare our project design for implementation steps such as placement and routing. You should run Analysis & Synthesis when you are ready to translate your code into real hardware logic, or want to verify the synthesized logic with tools like the **RTL Viewer.** You can

access the RTL Viewer by going **Tools→Netlist Viewers→RTL Viewer.** This will generate our design, and will look like the image below.

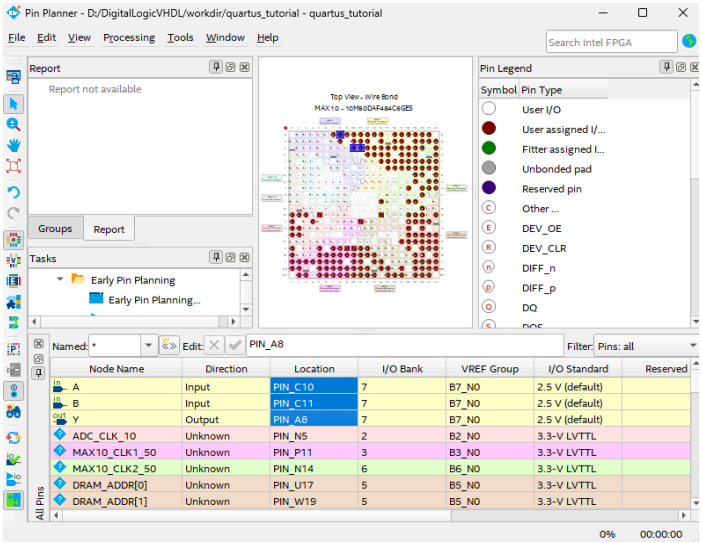


c. Now, we must assign pins to the values A, B, and Y on our DE-10 Lite. Refer to the necessary documentation for available locations to assign our values. In this example, we will be using the following:
   - A: SW0 (PIN_C10)
   - B: SW1 (PIN_C11)
   - Y: LED0 (PIN_A8)

   In order to define these pin-outs, we will open the **Pin Planner** page, by navigating to **Assignments→Pin Planner,** or the icon below

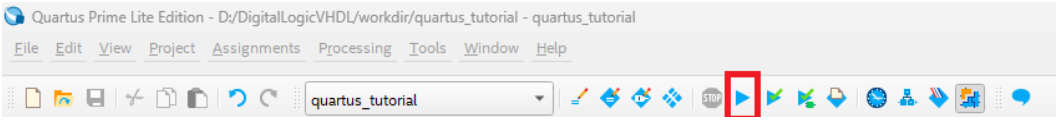


After performing the Analysis & Elaboration, three empty cells labelled A, B, and Y should appear. Under **"Location"** write the pin values defined above for A, B, and Y. The result should look like the image below.



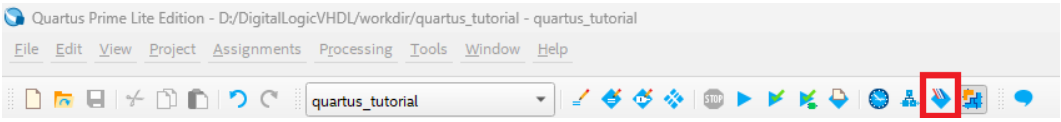
If you run into an error where it states **"Editing location assignment is not successful. The pin is already assigned."** it means that pin location has already been assigned a default value. In order to remove this error, find where the pin you wish to assign has been assigned, and remove it by either clearing the text box, or deleting the entire cell. Once done, you can assign your defined value to the pin you want. After defining all the pin assignments, we can move to the final stage of compilation.

d. Lastly, we will run **Compilation,** which is denoted by the "start" arrow highlighted below.



When running a Compilation, Quartus will perform a complete process which includes Analysis & Synthesis, Fitting (mapping logic to physical device resources and routing connections), Timing Analysis, and Assembling programming files (.sof and .pof). The outcome of the Compilation will produce all the necessary files to program (flash) an FPGA device, in our case the DE-10 Lite. Use this form of compilation right before programming your device and verifying it with previous forms of compilation.

3. To program the FPGA, we will open the **Programmer** tool under **Tools→Programmer**, or at the button highlighted below.



Next we'll open the **Add File** menu, and it will take you to the file tree of your project. From here, navigate to "**output_files"** and there you will find at least two files. Depending on what you named the project, you'll find two files names **"(PROJECT_NAME).pof"** and **"(PROJECT_NAME).sof."**

   a. When working with FPGA development in Quartus, it's important to understand the difference between .sof and .pof files.

.sof	SRAM Object File (sof), used for temporary configuration via JTAG. This file when loaded will be placed within the FPGA's RAM. This means the program is *volatile*, and will be erased when powered off.
.pof	Programming Object File (pof), used for non-volatile storage, and allows for the FPGA to retain the configuration after power cycles.

   You will mostly be working with **.sof** files in this course, but it will be defined on the lab documents otherwise if **.pof** is necessary.

   b. For the purposes of this example, double click on the **"(PROJECT_NAME).sof"** file to be flashed on to the board. Once this is done, press **Start** and success! You've now successfully completed the Quartus tutorial project, and you can view the XOR gate functionality on your board now!

   c. If you run into any problems related to Quartus not recognizing your FPGA device, refer to the *Lab Software Installation* guides of your respective device. .