Tutorial: Xilinx ISE 14.7 and Digilent Nexys 3

This tutorial will show you how to:

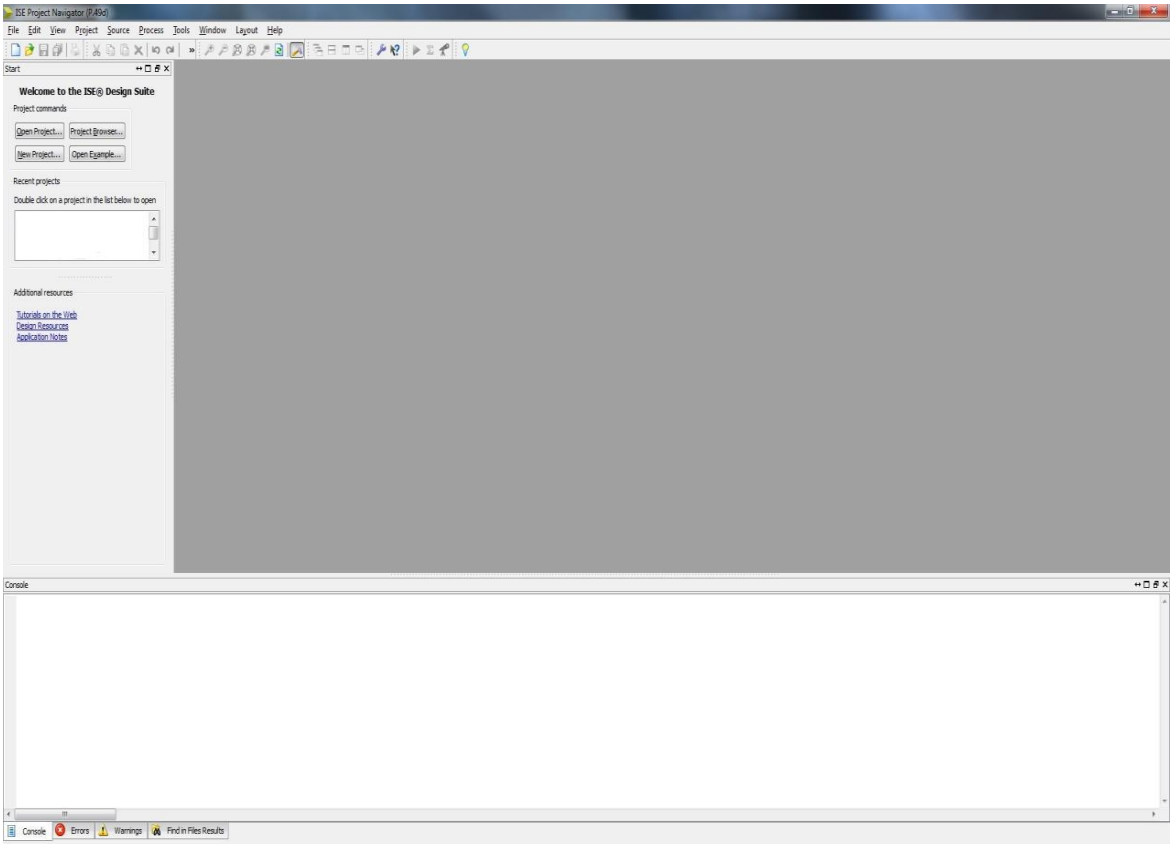
- **Part I:** Set up a new project in ISE 14.7
- **Part II:** Implement a function using Schematics
- **Part III:** Simulate the schematic circuit using ISim
- **Part IV:** Constraint, Synthesize, Implement, Generate, and Program for Nexys 3 FPGA board

Part I: Set up a new project in Xilinx ISE 14.7

Attention: Make sure to use the appropriate version of the ISE, 64 bit navigator for a 64 bit OS, and 32 bit for 32 bit. If you don't pay attention to this, there will be unexpected behavior in the ISE software and thing may not work properly!

1. Open the **Xilinx ISE Design Suit 14.7**. You can click on the ISE icon on the desktop, or search **Start → All Programs → Xilinx ISE Design Suite 14.7 → ISE Design Tools → Project Navigator**

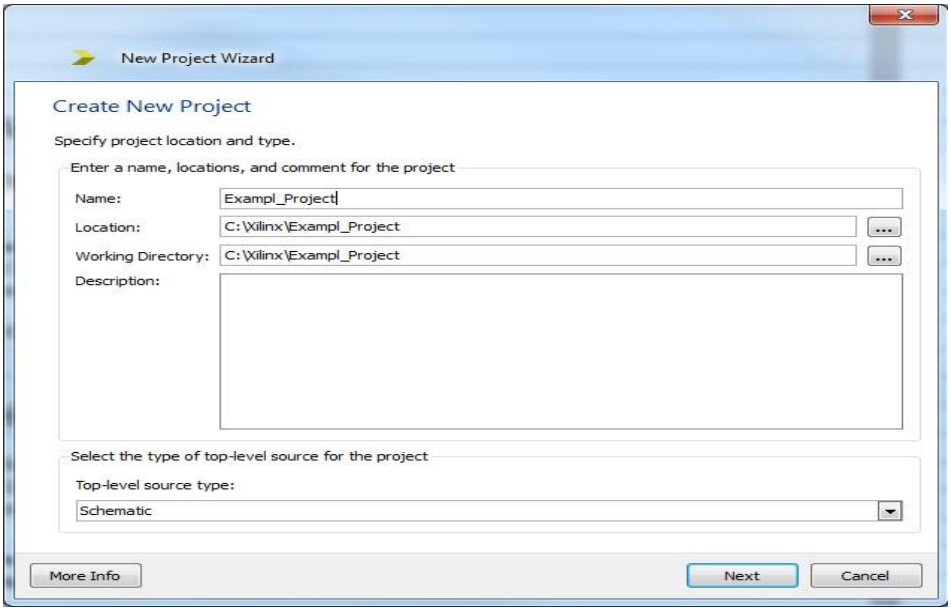
   The screen should look like the following, the ISE by default opens the last project otherwise none when open first time:



2. Now either press the **New Project** tab or select **File → New Project**… and change the Name and Location to whatever you like.

Attention: Xilinx does not allow spaces in path or file names! For example "C:\COE 203" will not work, same for the file name! Use the under_score for spaces if you need to.

The selected "Top-level source type:" is Schematic because that's what we're planning on using first. This is not critical, as you can always add a new source file of any type later. The dialog box for the project wizard looks like:

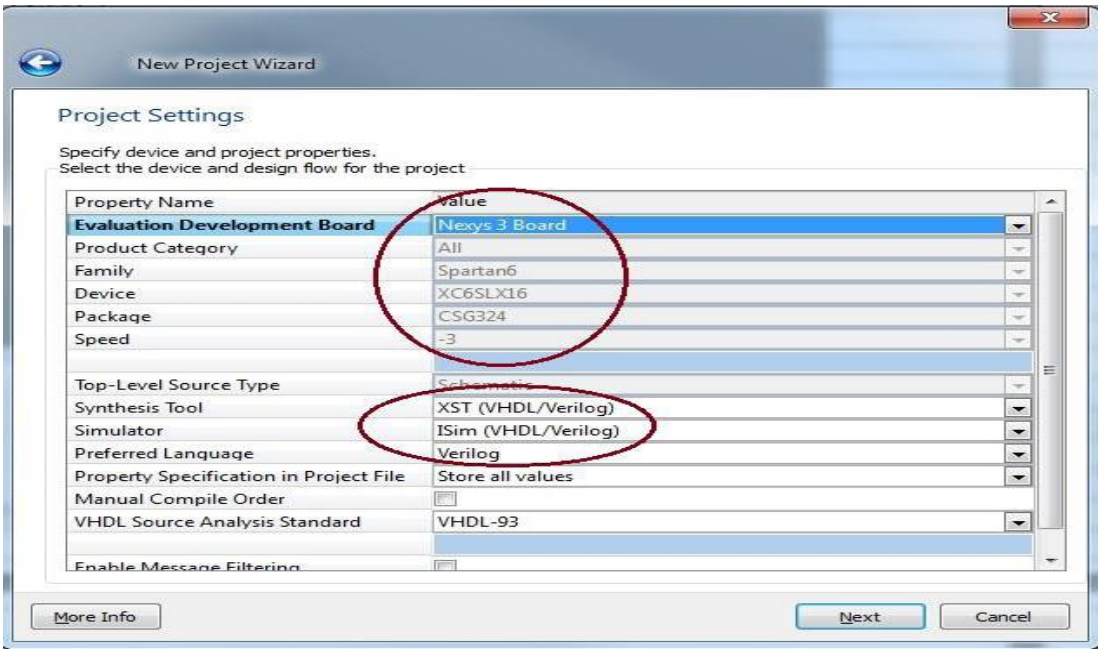


3. Click NEXT and in the next dialog box you should fill in the fields as shown here. You can do this in two ways, one is to select an "Evaluation Development Board" from the drop down list, and in our case you should select "Nexys 3". This will automatically fill out the board information in the next five sections. If the board does not exist in the list then you can set correct choices according to the following image.

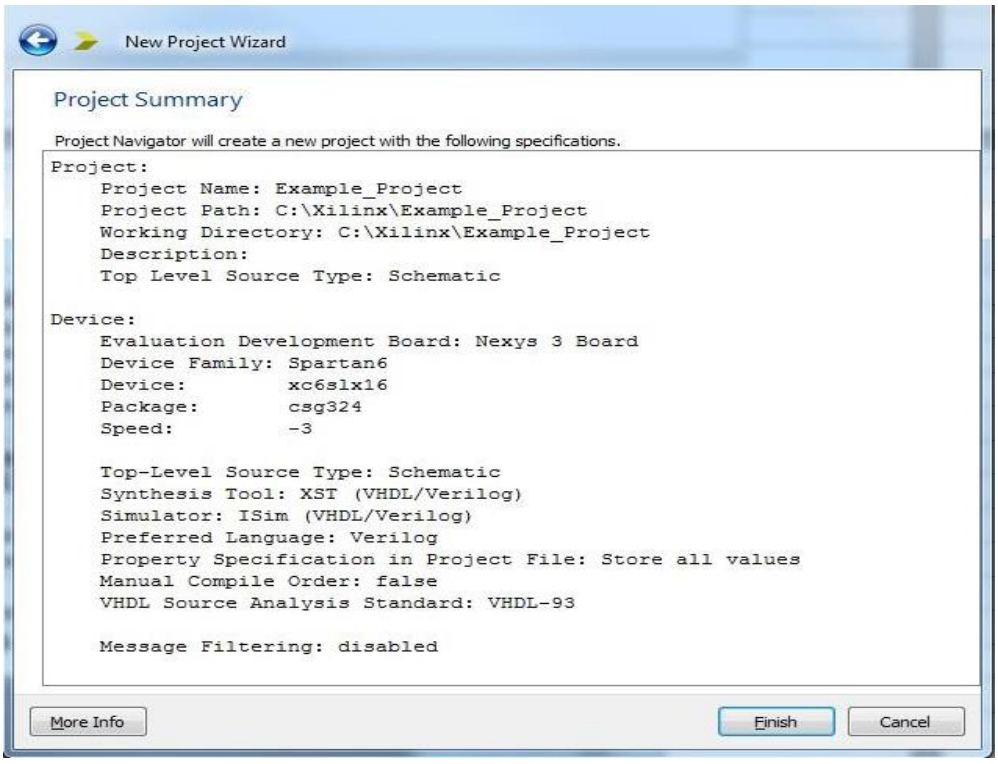
We are using a General Purpose product in the Xilinx **Spartan6** family. The specific chip on the Nexys 3 board is an **XC6SLX16** in a **CSG324** package and the **–3** speed grade.

Attention: If you fail to set the correct options in this part, you will not be able to implement your design and program it on the Nexys 3 board!

Please make sure that the Synthesis Tool is **XST**, the Simulator is the **ISim**, and the Preferred Language is **Verilog**. This is very important for proper operation.

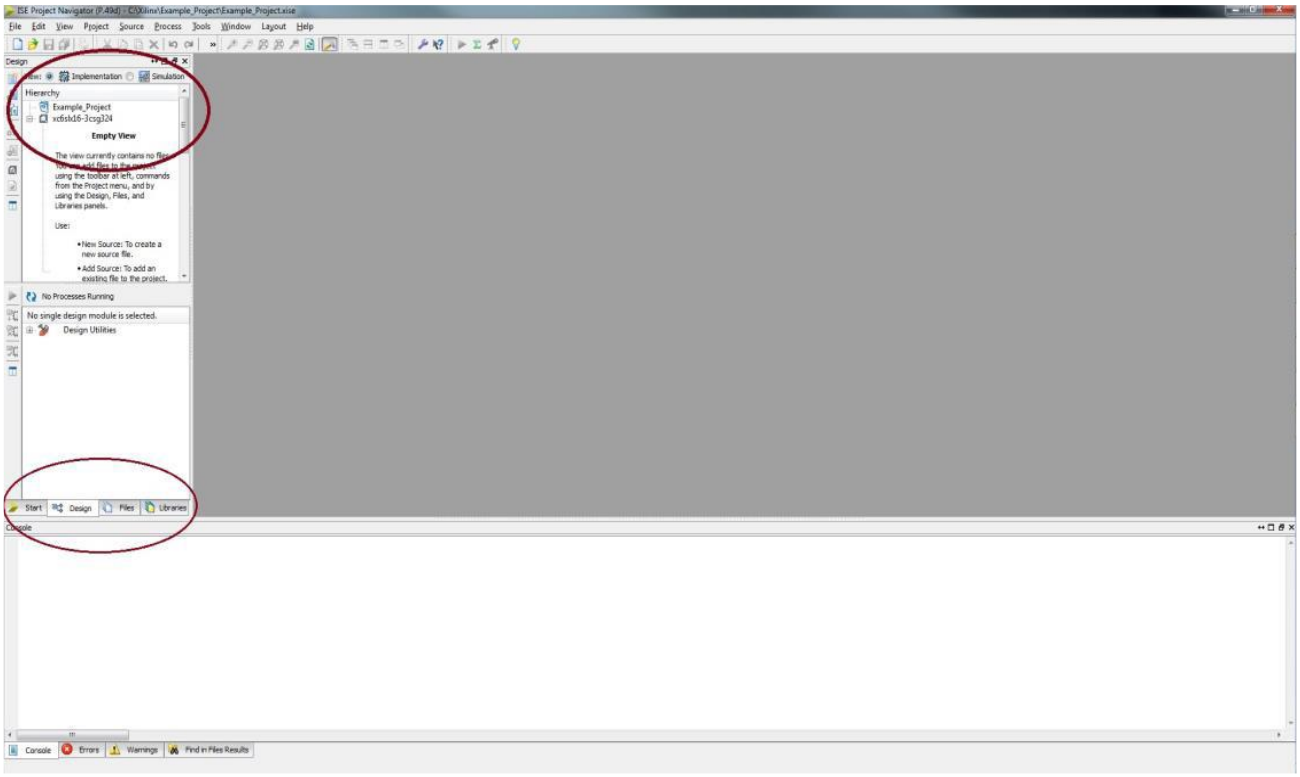


4. Click **NEXT** and review the project summary page and then click **FINISH**, it is always good to double-check the summary to prevent headaches due to the problems you can face while implementing your design if the information is incorrect.

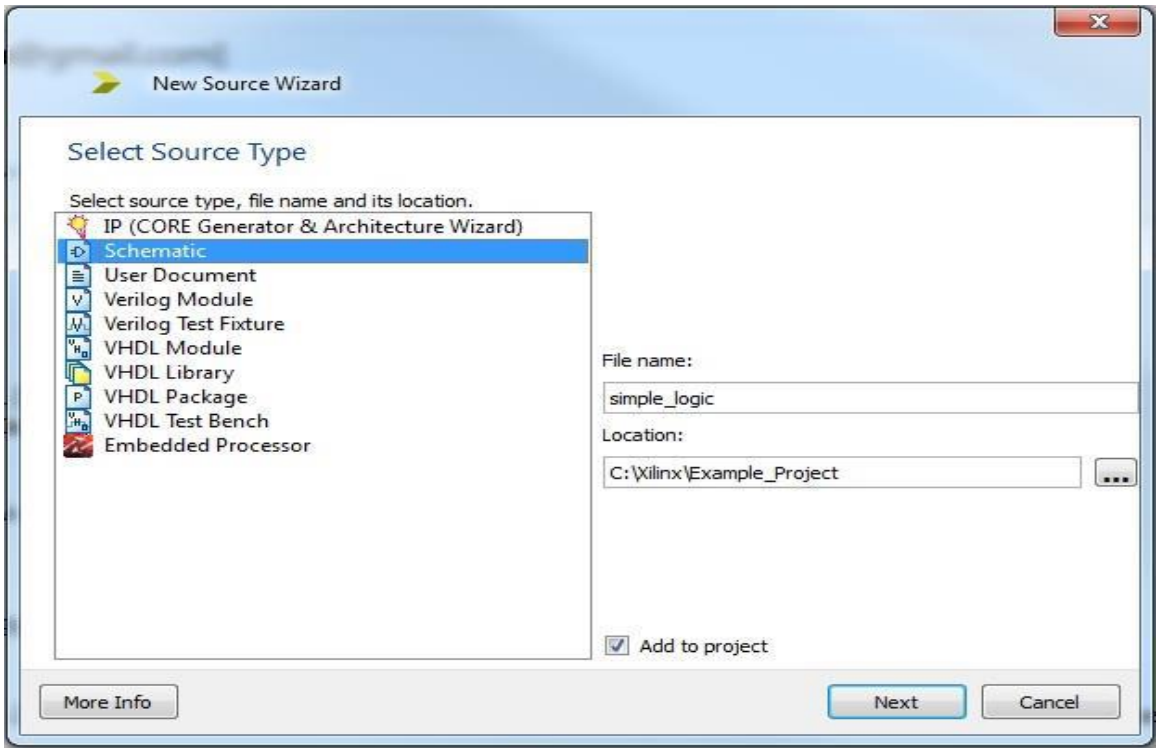


Part II: Implement a function using Schematics

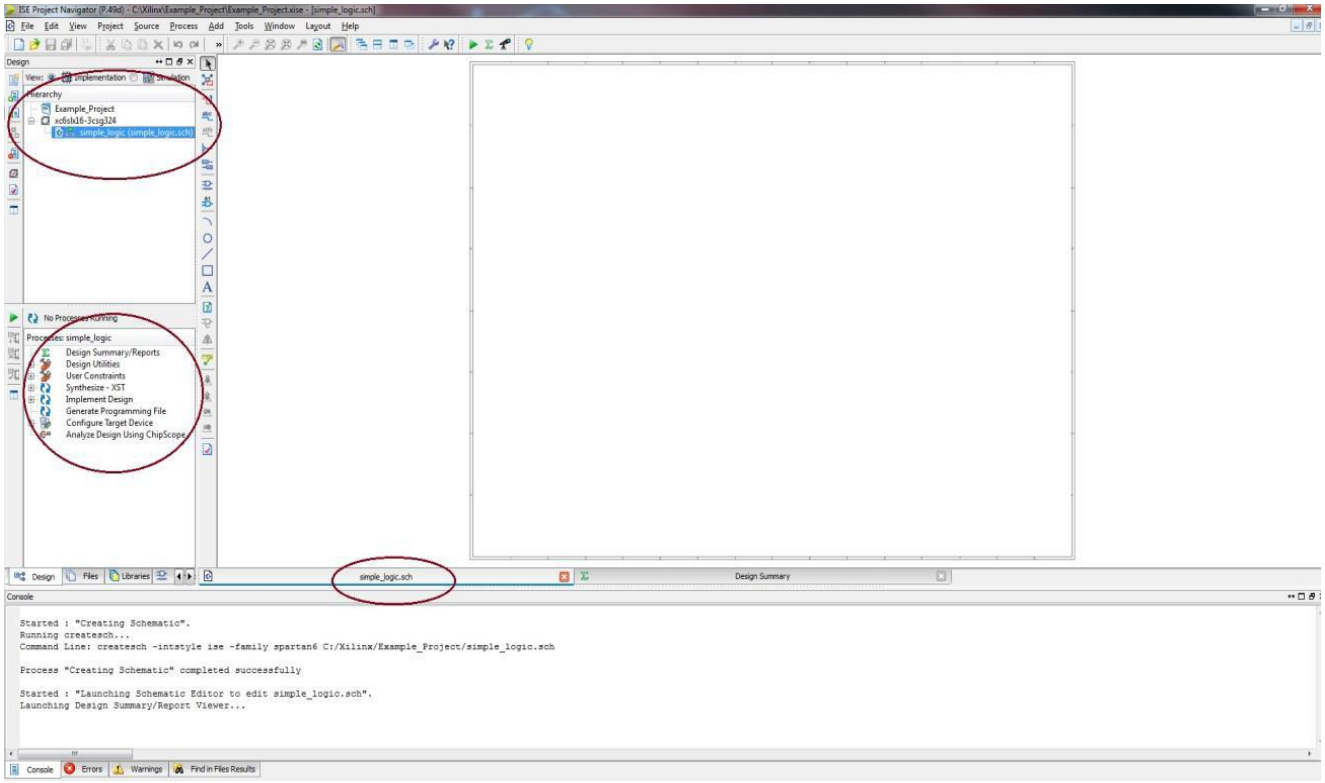
1. Now you should have a new project that targets the correct Xilinx part and other features of the ISE system. Notice the window to the left, and also notice the four tabs Start, Design, Files, …, and in the case of an open schematic a Symbols tab will appear. Ensure that the "Implementation" choice is selected on the design pane.



2. Now you can create a new schematic in your project. Choose **Project → New Source** or right click on the Hierarchy section of the design windows to get the dialog box that adds a new source file to your project (or use the New Source widget on the left vertical tool bar). Source files can be of many types. Select **Schematic** and name the File name: according to your experiment (here it is simple_logic). Now click **NEXT** and after observing the summary page click **FINISH**.



3. Now you have a blank schematic view, and also a Design Summary view in the main pane of the ISE window. You can switch between windows in the main pane with the tabs along the bottom. The (empty) schematic looks like below. Also note the Hierarchy created and the number of processes such as "Synthesis", "Implement", "Generate", etc. You can run on it.

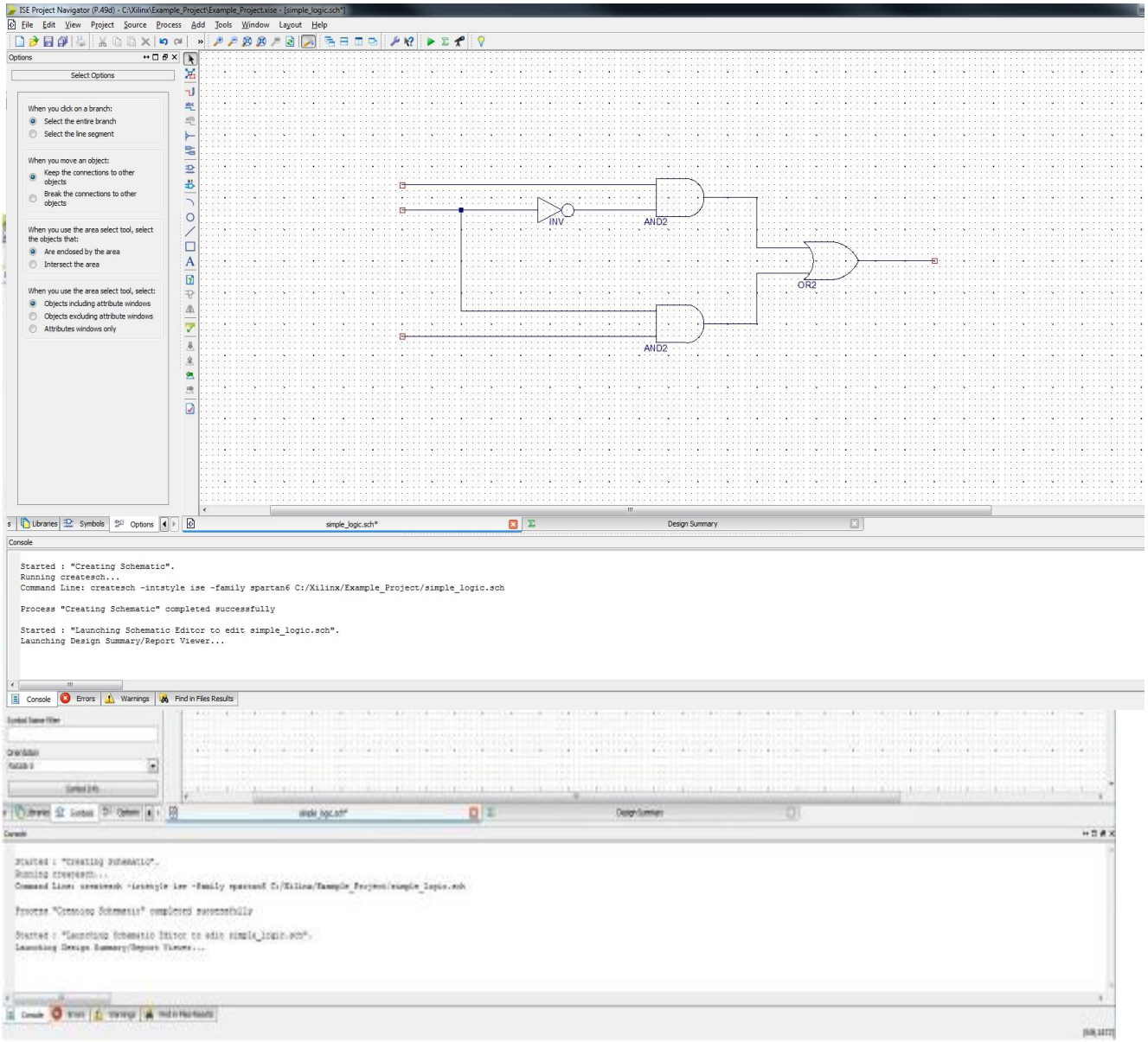


4. If you don't see this exact view, you may be looking at a different tab in a window. Each pane has tabs at the bottom that let you switch to look at different things. For example, for the left pane in the image above there are multiple tabs to allow different things, "Files", "Snapshot", "Libraries", and "Symbols" tabs that you can click on and get different information about the project. The lower left pane is the Processes tab and Options tab that show different tools and steps. The main window in the figure above has a "simple_logic.sch" tab for the new schematic, and a "Design Summary" tab. If you open more schematics, or other types of files (like Verilog files) the main pane will have additional tabs.

   Now you can select components from the Symbols tab of the upper left pane and drag them to your schematic. You can narrow down your choices using the Categories, or by typing the first few characters of the symbol you're looking for in the Symbol Name Filter, or just scroll through the lists and see what's there. The important category for now is Logic: **General logic gates**.

Grab some components from those categories to make a simple schematic to implement a simple function and drop them into the schematic page. Now we need to add wires and I/O markers by using the tools on the vertical bar between the schematic page and the side pane. Also pay attention to the "Options" tab at the bottom of the left pane. This section gives you some options on the schematic such as "selecting the entire wire branch" or "selecting line segments". This is good to know if you wanted to only remove a piece of wire and not everything that it is attached to.

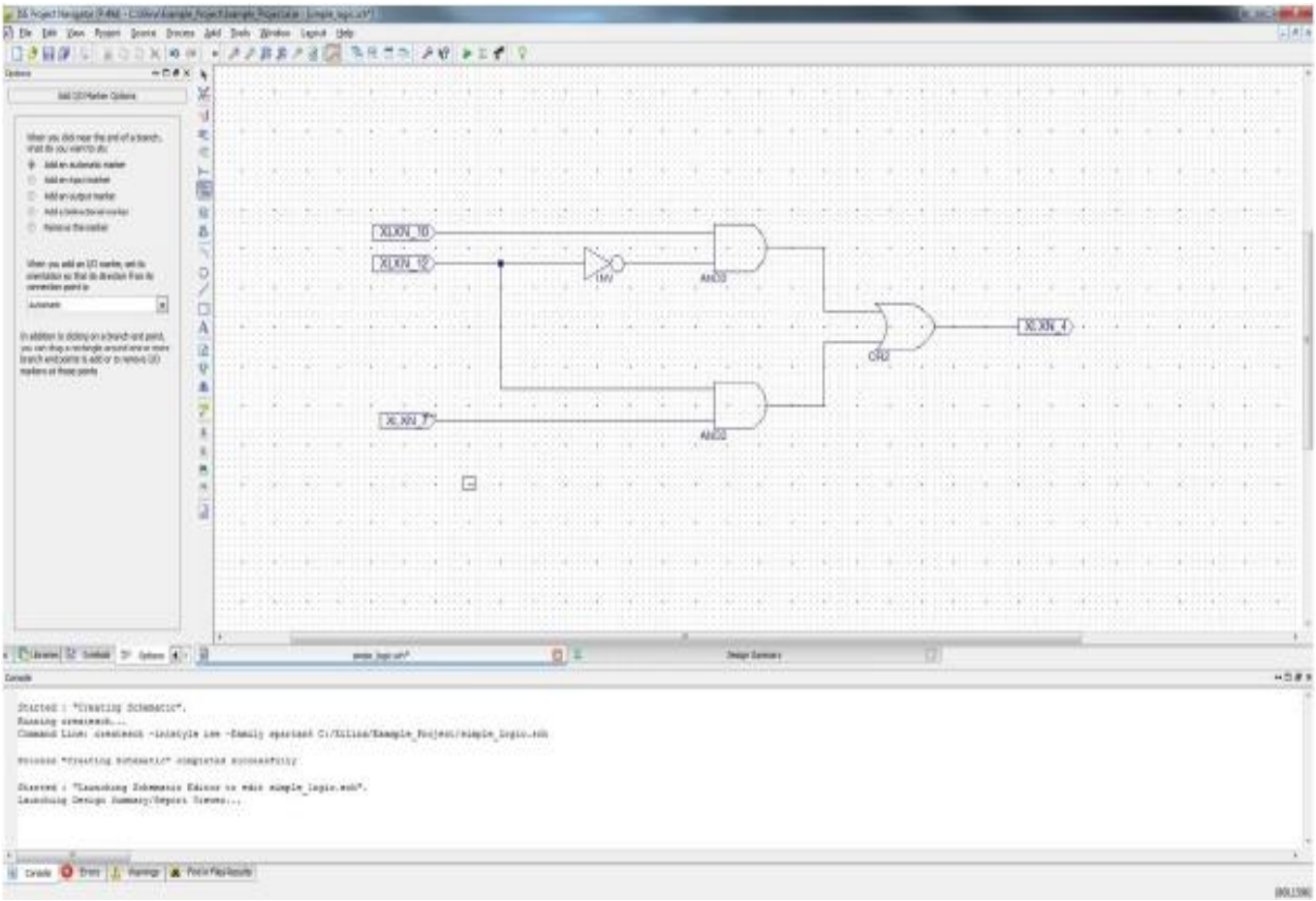
5. Use the wiring tool to wire up the components. It is in the tool bar and looks like a red line and a pencil. You could also use **Add →Wire** from the menu. Use the following components (from the **Logic** category) in this example:
    a. **and2:** a two-input AND gate
    b. **or2:** a two-input OR gate

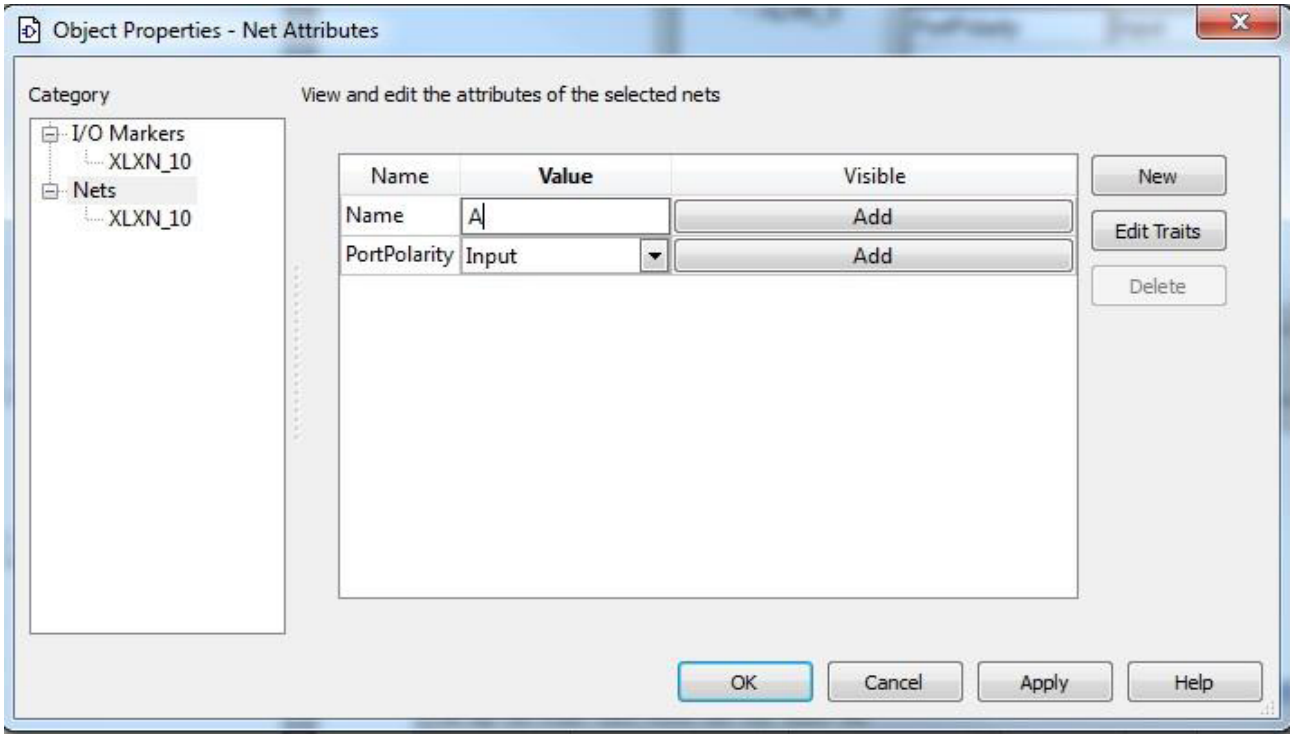


    c. **inv:** an inverter

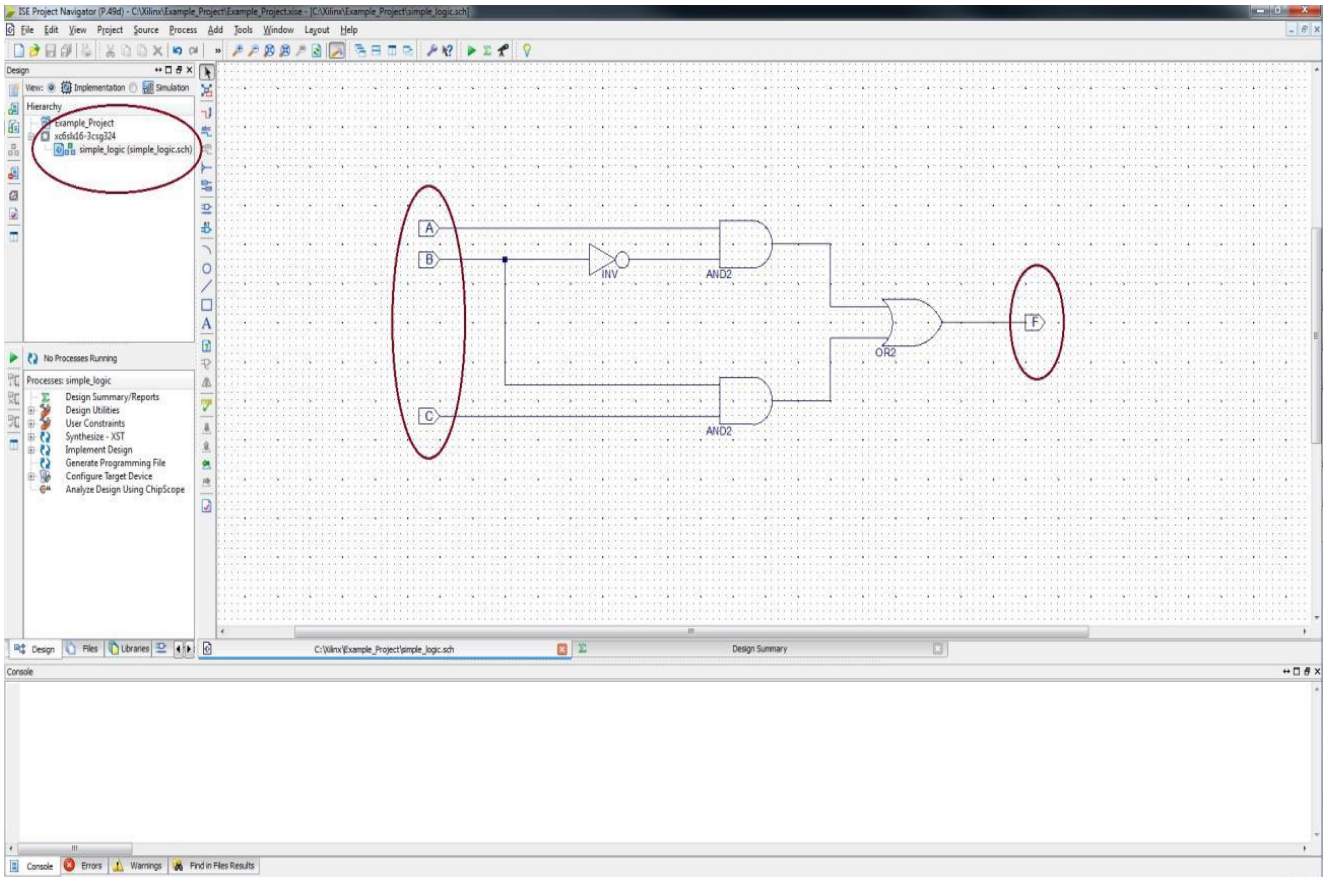
6. Now place **I/O Markers** to the inputs and outputs. Use the **I/O Marker** widget that looks like two little labels. You can also use the **Add → I/O Marker** command from the menu. Click on the end points of the wire to add the marker.

7. Always change the name of the marker to a good identifier. Double click the marker, or select the marker and right click to get a menu and choose Edit → Properties. Then click on "Nets" and then edit the "Name", also observe the Port Polarity, then click OK. Rename the inputs A, B and C and the output F. An I/O Marker dialog box looks like the following:





8. Now the resulting schematic looks like the following and it is ready for simulation or synthesis. Once the schematic is saved it should show up in the **Sources** pane in the upper left of the ISE screen. When you complete your schematic diagram, save it. If there is any error, fix it.

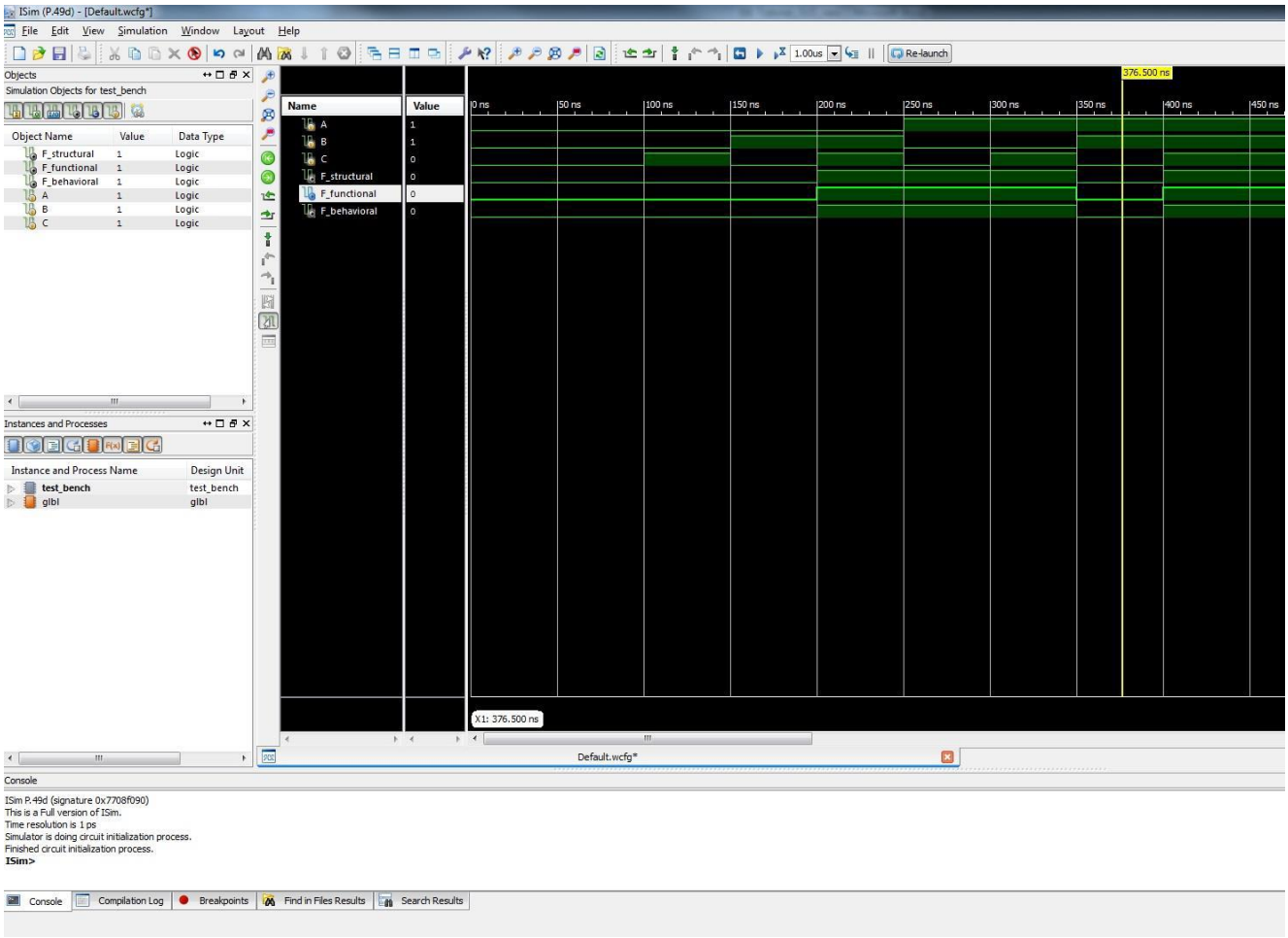


Part III: Simulate the schematic circuit using the ISim simulator

The objective here is to simulate the design to verify its functionality. ISE provides an integrated simulation flow with the ISim simulator that allows simulations to be run from the Xilinx Project Navigator GUI. We introduce the concept of simulation and how to verify the function of a circuit through behavioral simulation.

1. In the project navigator to the left, click on the **Design** tab, then click on your schematic file. Change to the simulation mode by selecting the **Simulation** radio button.
2. Press on the **+** mark in front of ISim Simulator to expand the list. Right click on the **Simulate Behavioral Model** and choose **Process Properties** to change simulation attributes. Uncheck the mark in front of **Run for Specified Time.** This will not limit the simulation for a specific run time. Press **OK**.
3. To run the simulation, double click on **Simulate Behavioral Model,** or right click and press **Run.**
4. ISE will launch ISim in a separate window. Note that the simulation will fail to run if a current process of ISim is working, close any instance of ISim before running any simulation.
5. ISim will launch automatically. The wave window displays the signals, buses and their waveforms. Note that there are four signals shown; A, B, C, and F.
6. Right click on input A in the objects window, and choose Force clock. Add the following values:
   Leading Edge Value: 0
   Trailing Edge Value: 1
   Period: 1 us
7. Alternatively, you can write the following Tcl command in the console window:
   isim force add A 0 -value 1 -time 500 ns -repeat 1 us
8. Similarly, right click on input B in the objects window and choose Force clock. Add the following values:
   Leading Edge Value: 0
   Trailing Edge Value: 1
   Period: 0.5 us
9. Alternatively, you can write the Tcl command in the console window:
   isim force add B 0 -value 1 -radix bin -time 250 ns -value 0 -radix bin -time 500 ns -value 1 -time 750 ns -repeat 1us
10. Similarly, right click on input C in the objects window and choose Force clock. Add the following values:
    Leading Edge Value: 0
    Trailing Edge Value: 1
    Period: 0.25 us
11. Enter 1 us inside the simulation time toolbox in the toolbar, and then press Run for the Time Specified in the Toolbar icon. Or type the following Tcl command in the console window.
    Run 1 us
12. The simulator will show the behavior of the gates according to the specified input signals, press in Zoom to Full View in the toolbar to show the entire simulation period.
13. The simulator will show the behavior of the gates according to the specified input signals, press in Zoom to Full View in the toolbar to show the entire simulation period. You can Zoom in and Zoom out using the icons in the toolbar.

14. To restart the simulation, press on Restart icon on the toolbar. Note that restarting the simulation will also remove the force clock values. You have to apply force clock to the input signals before running the simulation again.
15. Verify that the circuit is working correctly by checking the time diagram (waveform).



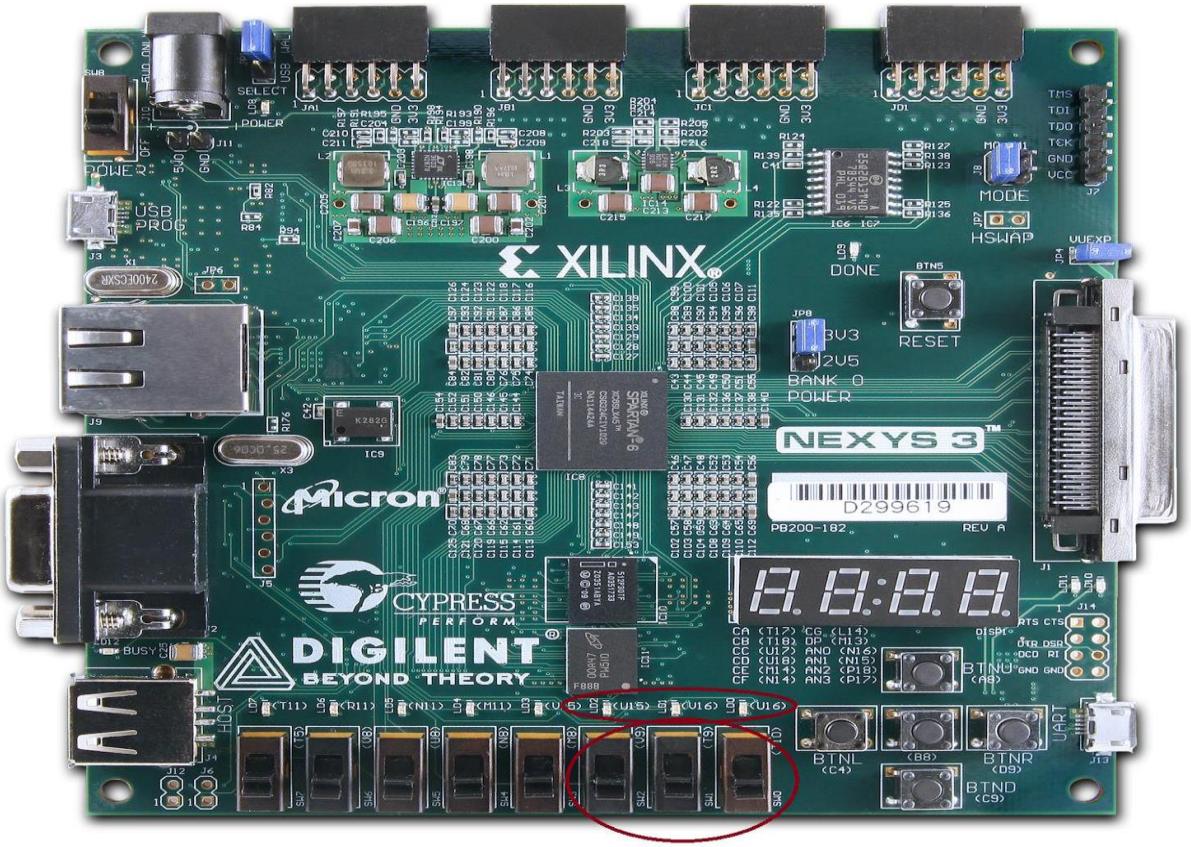
16. Close ISim and return to ISE.

Part IV: Constraint, Synthesize, Implement, Generate Bitstream, and Program the Nexys 3 FPGA board

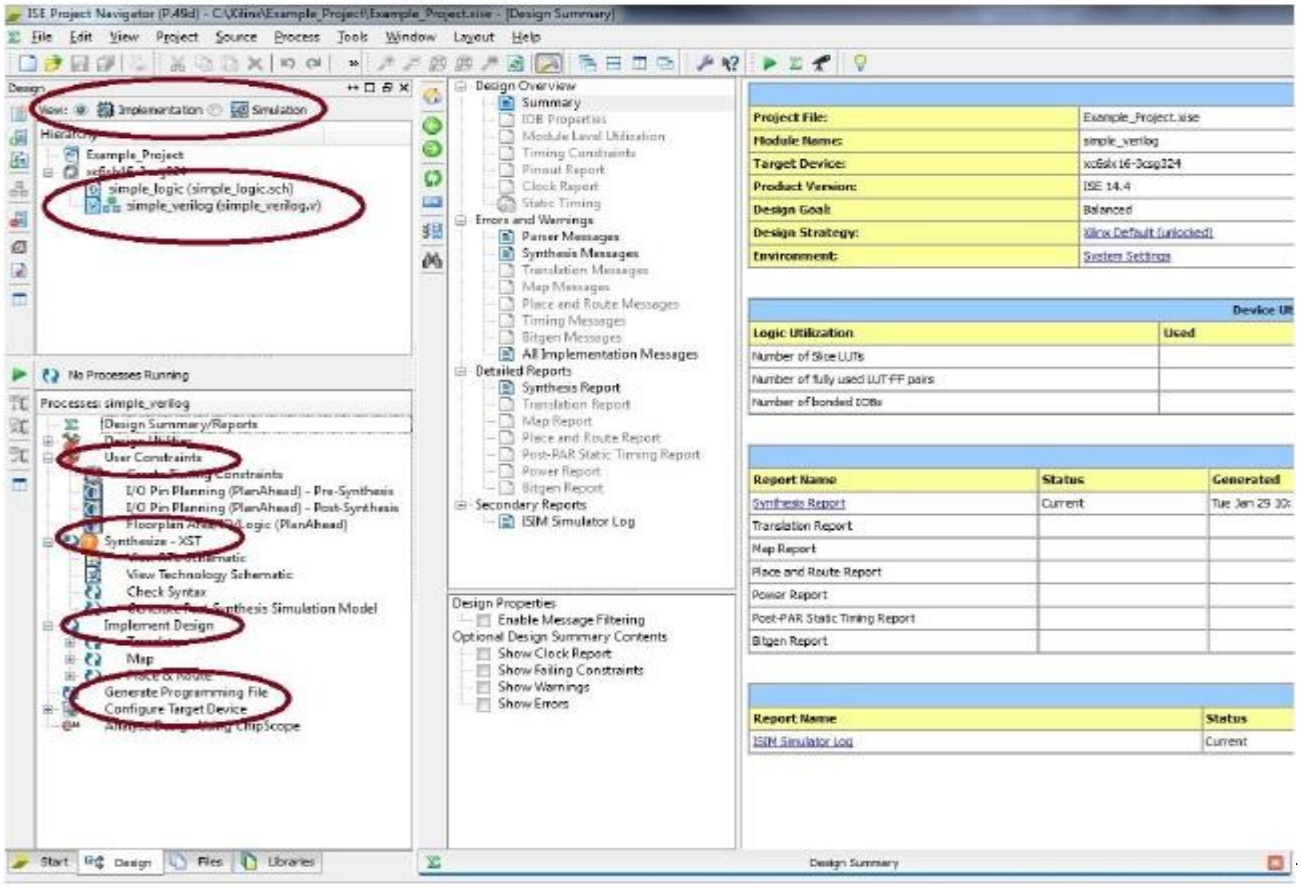
Overview: ISE will convert the schematic diagram into a set of configuration bits that are used to program the Xilinx FPGA board. Those configuration bits are in a **.bit** file and are downloaded to the Xilinx part in this next section of the tutorial.

The first three toggle switches on the board for A, B, and C, and the first LED for F is used.

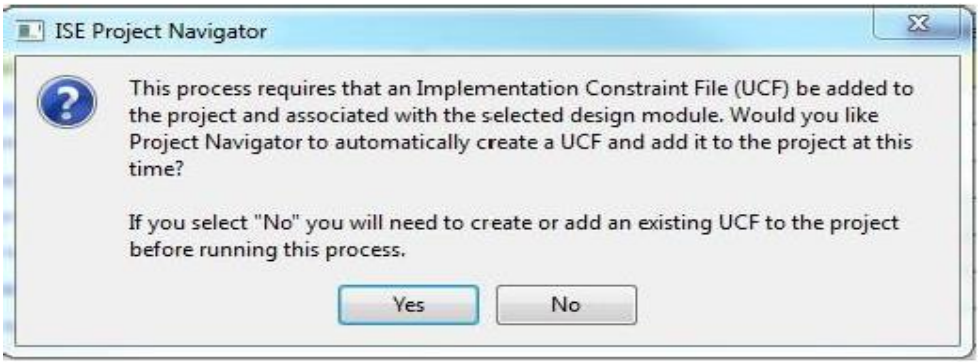
UCF (User Constraints File): Because we're headed towards putting this on the Xilinx FPGA on the Nexys 3 board, we need to set some constraints. In particular, we need to tell ISE which pins on the Xilinx chip we want **A**, **B**, **C**, and **F** assigned to so that we can access those from switches and LEDs on the Nexys 3 board. For that we need a "User Constraints File".



1. First, you need to ensure that you're in the **Implementation** view by selecting the Implementation radio button on top and that the module you're trying to implement on the board is set as the **top module** in the top left pane in order to get the options to synthesize, implement, and generate the design. Now if you look at the bottom left pane you can see a number of processes you can run on this schematic top module.



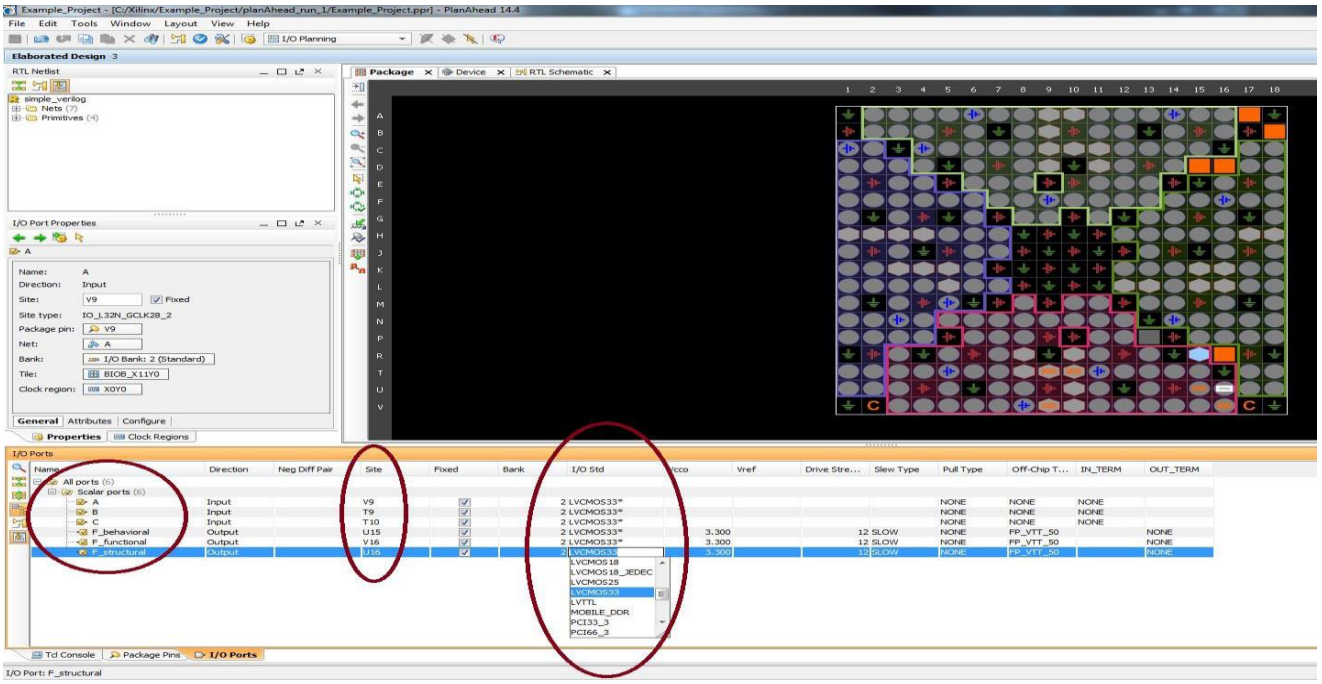
2. **Constraint:** Now it starts with creating a floor plan by setting the UCF file. To do this take a look at the User Constraints drop down option in the bottom left pane. We can set the pins in two different ways. Double click on the **I/O Pin Planning (Planahead) – Post-Synthesis**. This should bring up a message box for adding a new UCF file to your design, so click yes and this will kick start another Xilinx tool called Plan Ahead. This program allows you to set all constraints on all I/O pins in the design. Please follow these steps carefully.



3. **Pin Assignment:** We need to edit the initial UCF for the details of connections between the ports and pins in the design, so as Plan Ahead opens, take a look at the horizontal bottom pane. After expanding the Scalar Ports drop-down you should see all of the I/O pins. Click on the name of the pin (A, B, C, …) and then look at the properties table line listed in front of it. The only property that we want to change is the Site. The Site is the Pin# (FPGA board Pin Identifier). You can find all sorts of the information including the module schematics and the Pin #s for all of the Nexys 3 Peripherals by studying the Nexys 3 manual found at

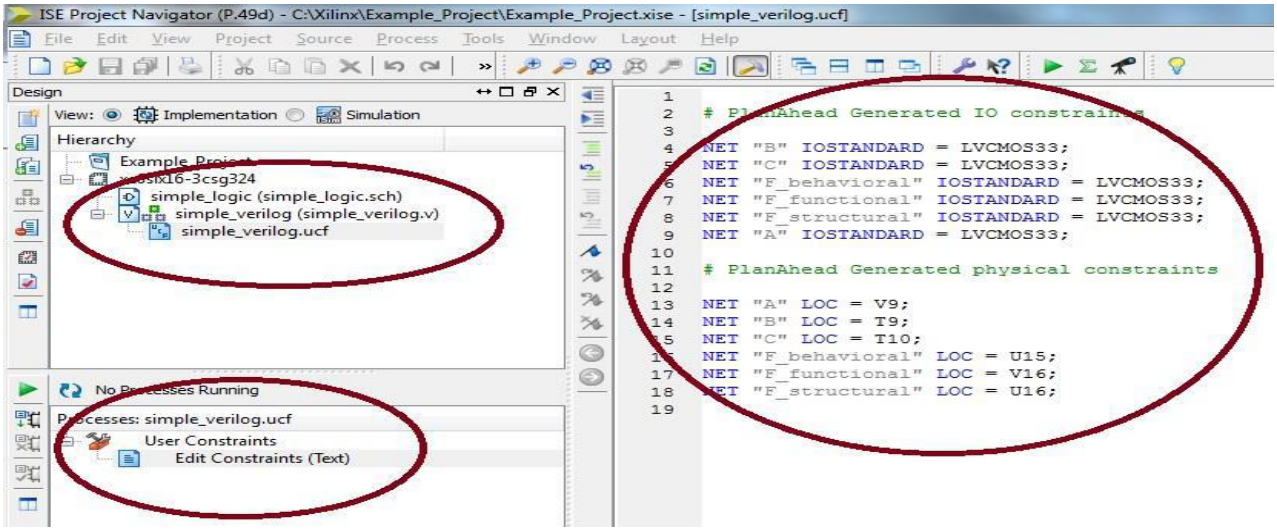
   http://www.digilentinc.com/Data/Products/NEXYS3/Nexys3_rm.pdf

   In order to change a property for any port, click on the rectangular space in the correct column and the correct row for the respective port. This click causes a drop-down menu to appear and then you can select the right choice. You can either type or select the correct choice. If there is a choice already selected then clicking on the text will allow you to change it, then hit enter. For the above circuit look for the three switches and an LED, you can also look closely at the actual switch or LED on the board and you will see an identifier in parentheses (i.e. T10 for SW0 referring to Pin #T10 connecting to Switch 0, the first toggle switch from the right). Set the correct identifier to the correct port in your design in Plan Ahead as follows:



After you're done putting the right Pin # for all the ports in the design, click the SAVE button from the top menu in the page and close down Plan Ahead.

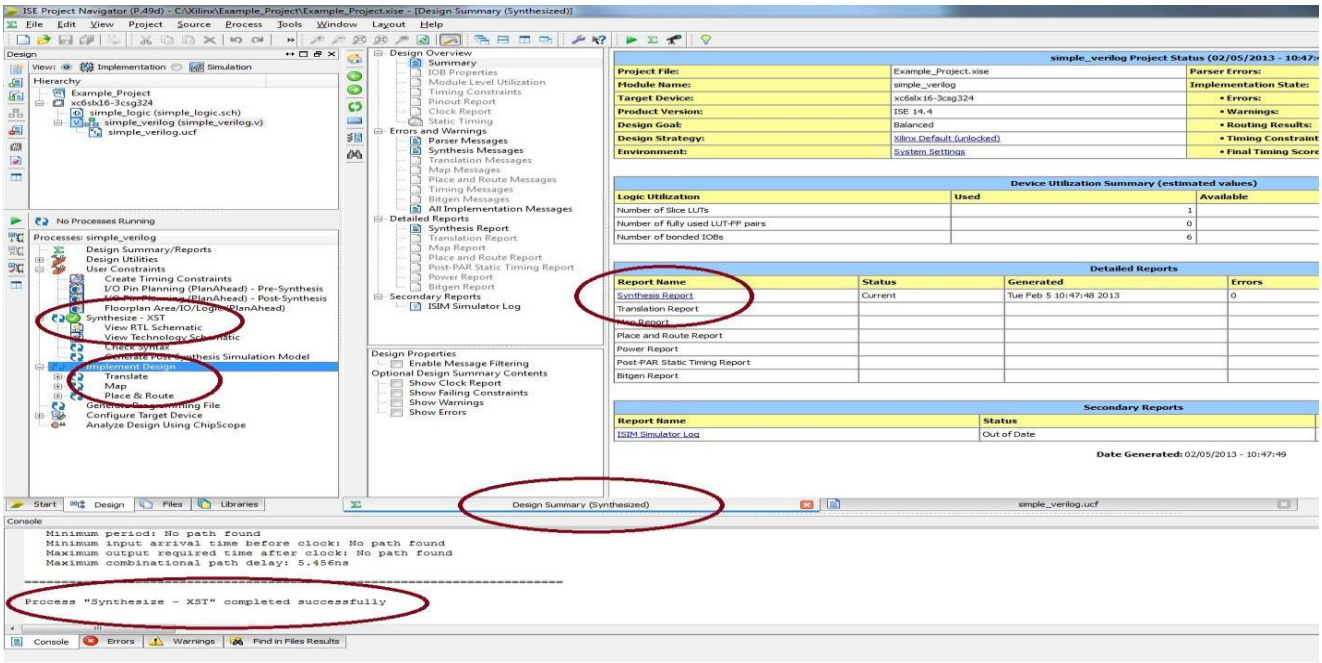
And your resulting **UCF file** should look like the following:



4. **Synthesize**: Now the design is ready for more processing and the next in line is to synthesize. This process will create a structural representation of the design (similar to compiling C code into assembly code). Do this by first ensuring that the top module is selected and highlighted in the top left pane of ISE and then simply double clicking Synthesize – XST in the lower left pane. After the process is done, you will either get a **green check mark** (✓ everything is smooth), a yellow attention mark (! there are warnings!), or a red X mark (**X** there are errors!), and orange question mark (? out of date). In the case of errors you need to investigate them by looking at the **Errors** tab on the bottom pane and fix, and then re-run synthesis. In the case of warnings make sure to review them and validate they are safe, or in the case of green you're ready for the next step.

   The synthesis process also creates a couple of more useful things that you should explore and study. One is the synthesis report full of information about timing, resource usage details, and etc. The other is generated RTL schematic; sometimes it is very useful to see what the XST made out of your of schematic. You can view these (highly recommended) by expanding Synthesize – XST for the schematic

and the Design Summary page in the main pane for the report.



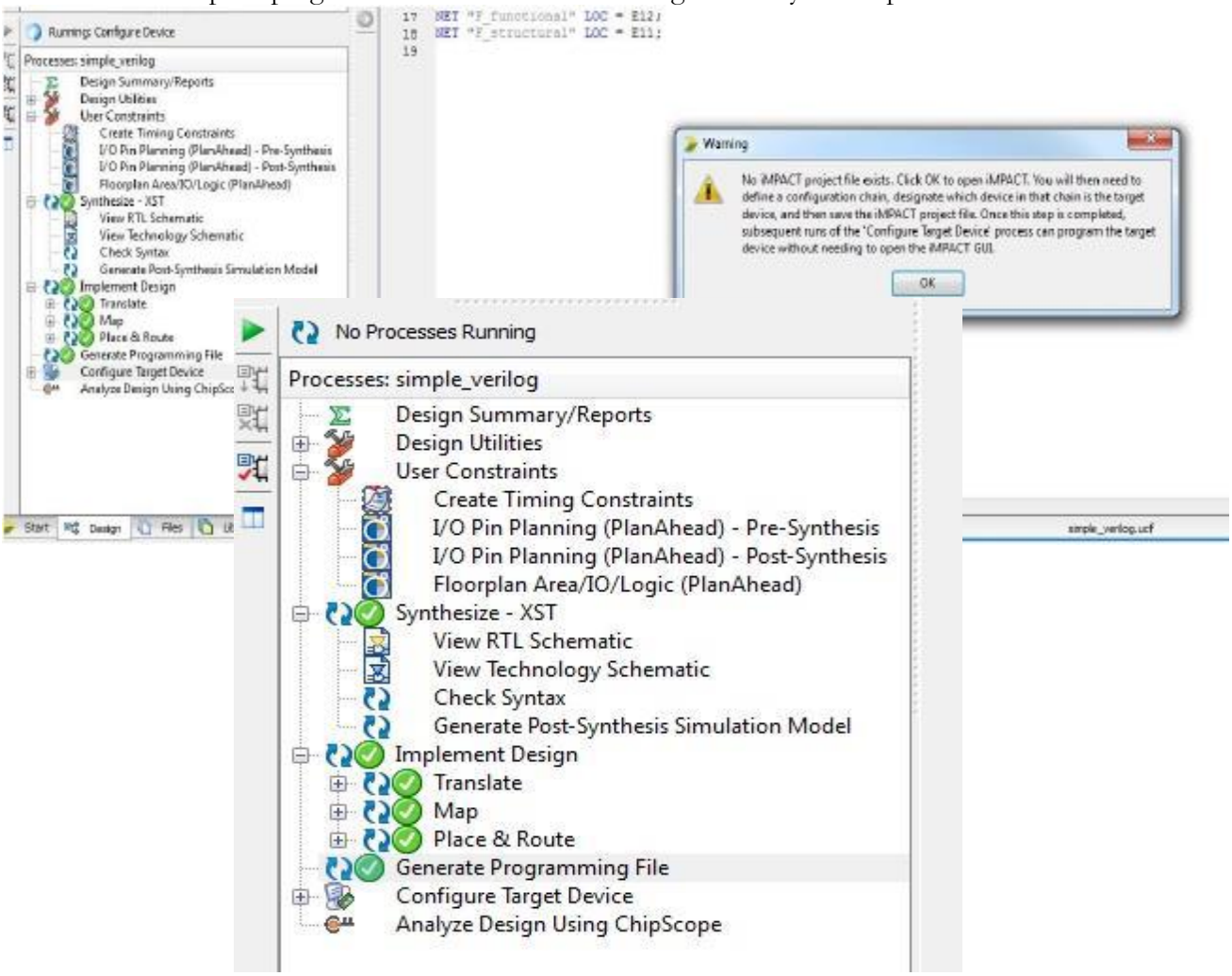
5. **Implement**: Next step is to define the hardware configuration. With your top module source file selected (simple_verilog.sch in this case), double click the **Implement Design** process in the **Processes** tab. This will translate the design to something that can physically be mapped to the particular FPGA that's on our board (the xc6slx16-3csg324). You should see a green check mark if this step finishes without issues. If there are issues, you need to read them for clues about what went wrong and what you should look at to fix things. If you expand this **Implement Design** tab (**which is not necessary**) you will see that the **Implement Design** process actually consists of three parts:

    a) **Translate**: Translate is the first step in the implementation process. The Translate process merges all of the input netlists and design constraint information and outputs a Xilinx NGD (Native Generic Database) file. The output NGD file can then be mapped to the targeted FPGA device.

    b) **Map**: Mapping is the process of assigning a design's logic elements to the specific physical elements that actually implement logic functions in a device. The Map process creates an NCD (Native Circuit Description) file. The NCD file will be used by the PAR process.

    c) **Place and Route** (PAR): PAR uses the NCD file created by the Map process to place and route your design. PAR outputs an NCD file that is used by the bitstream generator (BitGen) to create a (.bit) file. The Bit file (see the next step) is what's used to actually program the FPGA. In this part the actual transistor configuration and wire routing is decided.

6. **Generate:** In this step all information resulting from the previous steps are gathered and put into a Bit format that the USB programmer on the FPGA board (in our case Nexys3) understands. Simply double-click the **Generate Programming File** to generate a .bit file which will be used in the final step to program the board. Now the design is ready to be put on the board so we can

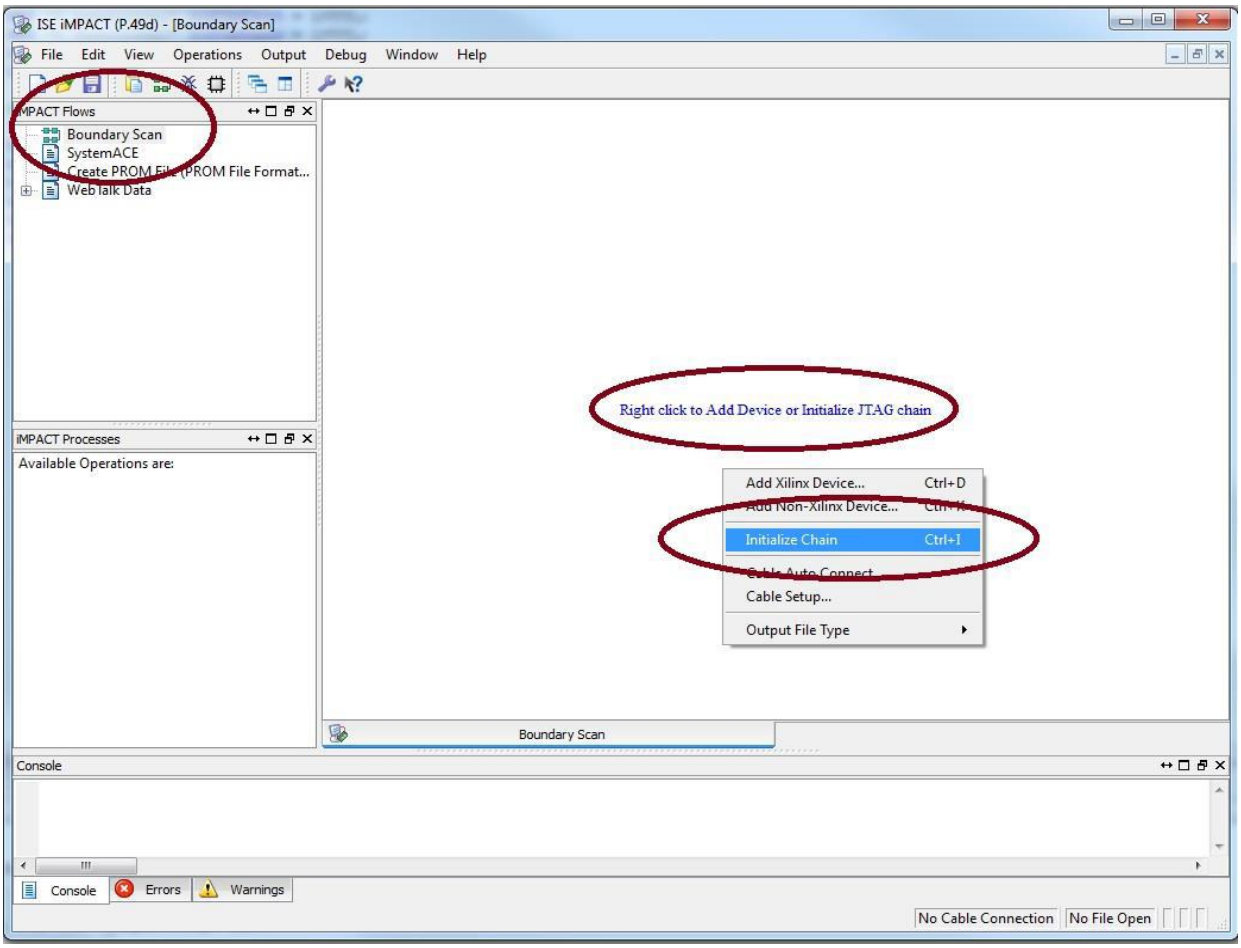


physically see its functionality.

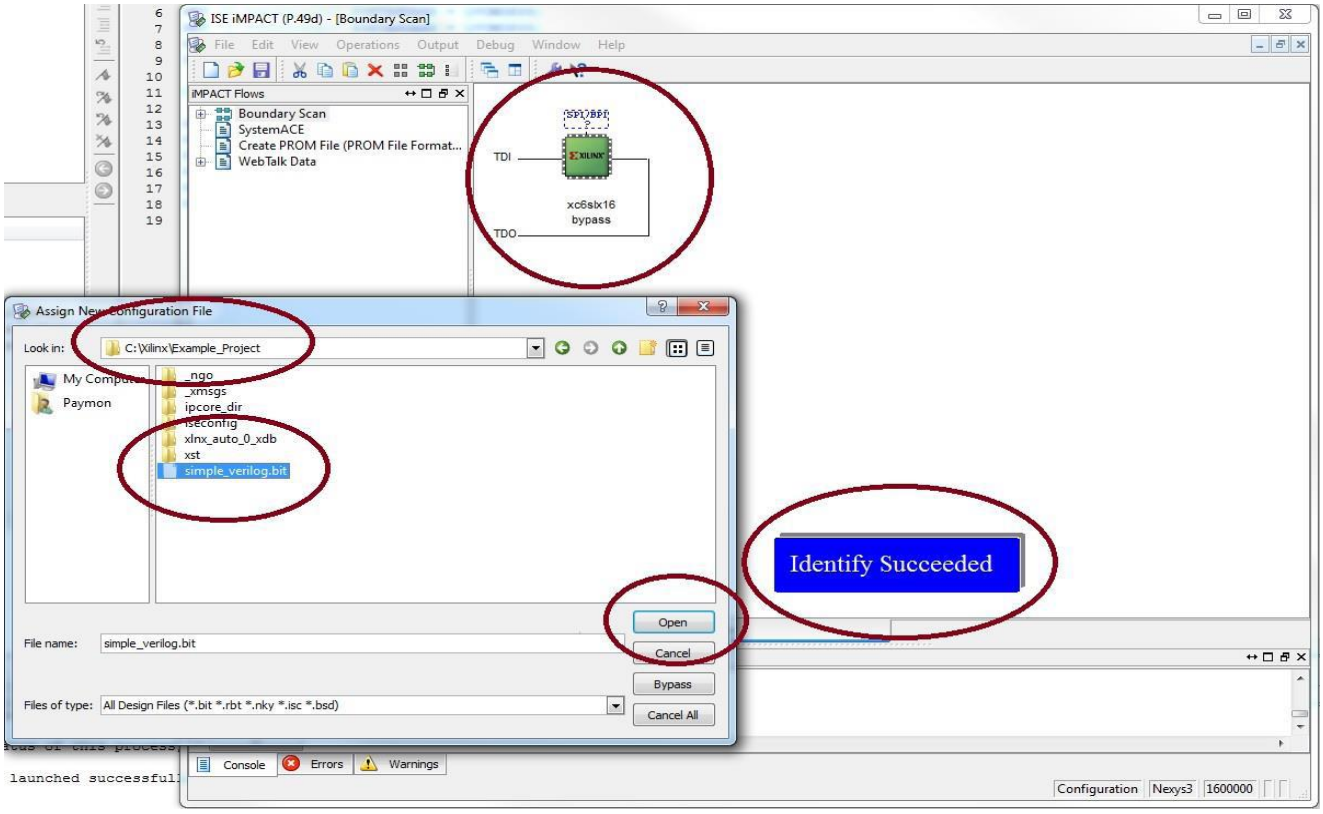
Attention: Before proceeding to the next step connect the board via the USB cable to the PC you're using, and turn on the power, the next step ensure proper connection to the board. Also note that you do not have to do steps 4, 5, and 6 individually. These processes are dependent on each other, if one needs a preceding process to be updated then it will automatically run that process before it runs itself. So you can just proceed to step 7 and watch ISE do everything.

7. **Program:** You can start the programming process by double clicking **Configure Target Device** and ISE will launch yet another Xilinx tool called **iMpact.** A warning box appears complaining about "No iMpact project file exists…", so just click OK to launch iMpact as it will automatically read your existing project.

8. In the ISE iMpact window which again looks a lot like ISE, double click **Boundary Scan** in the top left pane. In the boundary scan windows in the main pane, where it says "Right click to Add Device or Initialize chain" right-click in the middle of the page and select **Initialize Chain** or just press Ctrl + I. This will ensure there is a good connection to your board and it can communicate with it.

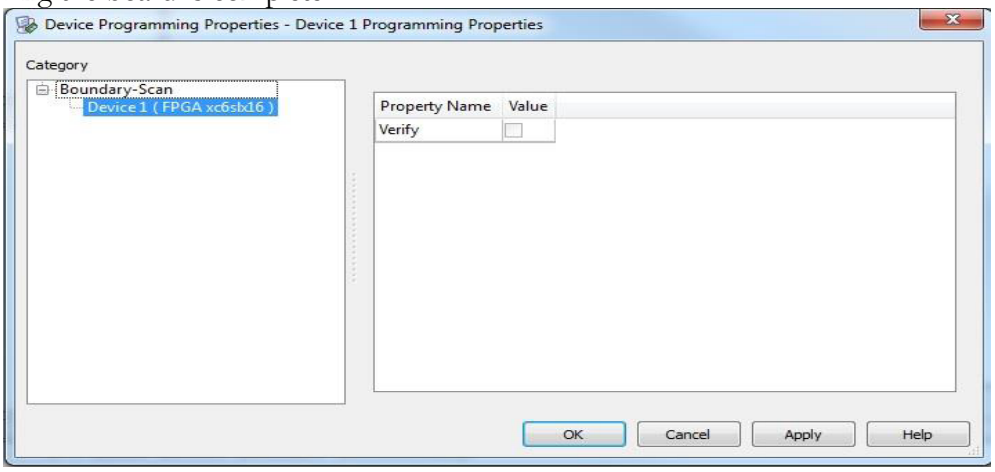


9. After iMpact verifies that the cable is connected it will prompt you to load your **.bit** file that you generated in step 6. Note that this file selection window doesn't always default to your existing project so you may need to navigate to your project folder and locate the .bit file. This file is always named to your top module so in our case it is **simple_verilog.bit.** Double-click or select the bit file and click open, again make sure it is the right file.

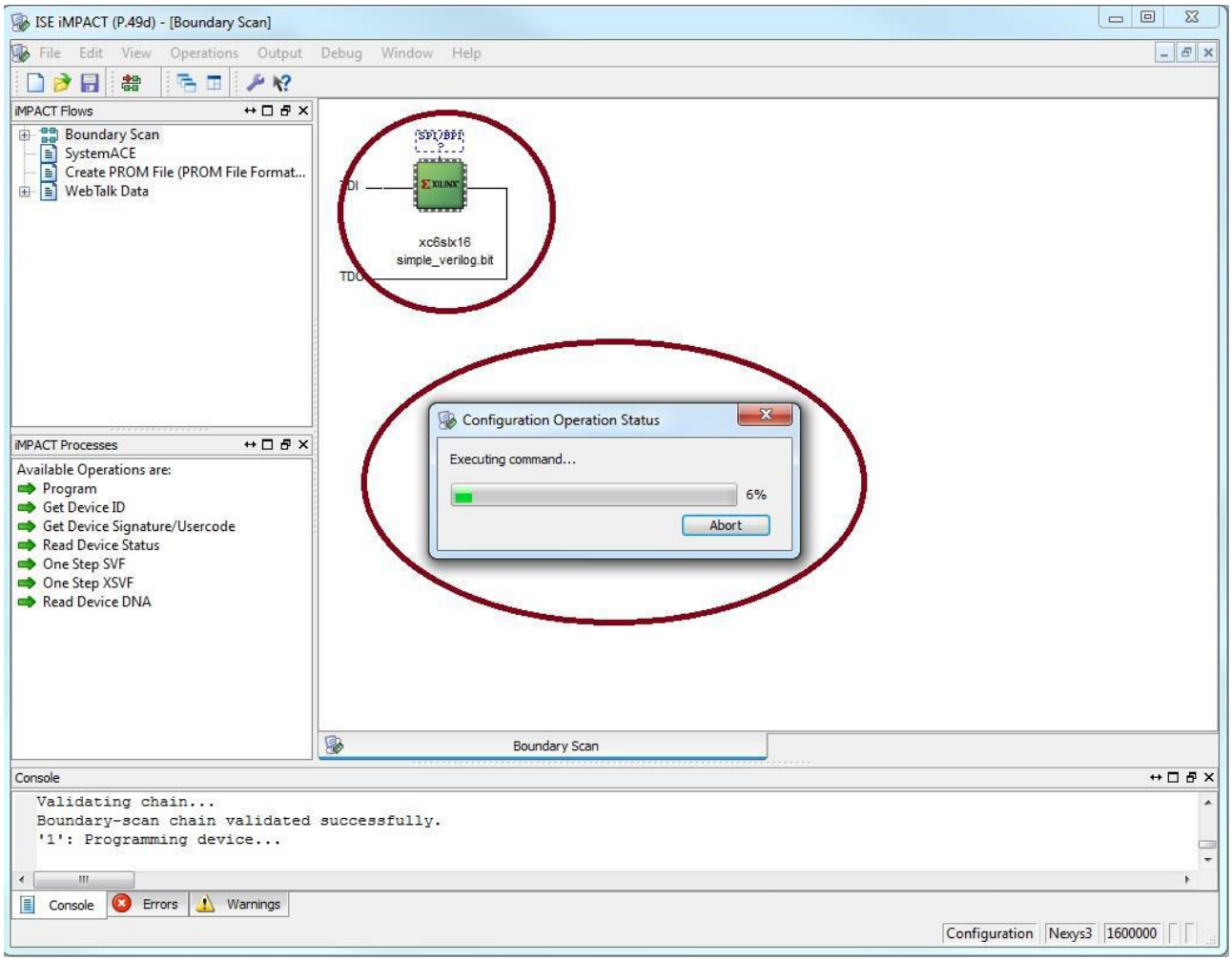


10. After the bit file is read in, iMpact prompts you to attach a PROM controller, just click NO to skip this step since we're not putting anything in the Flash memory.

11. In the next dialog box you would be verifying which device on the board you're targeting but in our case we only have the FPGA chip to program, so click Ok and the preparation for programming the board is complete.



12. All that is left is to right click on the green chip icon with the Xilinx logo in the main pane and click **Program**. After the communication bar finishes, your design is programmed to the Nexys 3 FPGA board. When the program operation completes, a blue message with "**Program Succeeded**" appears.


ISE iMPACT (P.49d) - [Boundary Scan]
File Edit View Operations Output Debug Window Help
iMPACT Flows
Boundary Scan
SystemACE
Create PROM File (PROM File Format...
WebTalk Data
SPI/BPI
TDI
TDO
xc6slx16
simple_verilog.bit
Configuration Operation Status
Executing command...
6%
Abort
iMPACT Processes
Available Operations are:
Program
Get Device ID
Get Device Signature/Usercode
Read Device Status
One Step SVF
One Step XSVF
Read Device DNA
Boundary Scan
Console
Validating chain...
Boundary-scan chain validated successfully.
'1': Programming device...
Console
Errors
Warnings
Configuration
Nexys3
1600000