Fpga Manual

Dillon Dotson (Senior Engineering), Patrick Durofchalk (Senior Computer Engineering), Miguel Gonzalez(Sophomore Computer Engineering)

Table of Contents

Table of Contents 1

Getting Started 1

Instructions on FPGA Board and Xilinx software 1

Installing Software 2

Using Xilinx Project Navigator 3

Implementation / Programming 6

Example Circuits 8

Debugging 29

Getting Started

I. Instructions on FPGA Board and Xilinx software

Field-Programmable Gate Array (FPGA) - A logic chip which can be programmed to function as an array of computational logic blocks.

Xilinx Software - Xilinx Software allows a designer to graphically create a logic circuit which can be tested and simulated prior to implementation on the FPGA board.

FPGA vs Breadboarding[1]

- FPGA's simplify design, implementation, testing, and debugging
- Reusable - Easily reprogrammed and reconfigured
- Most are non volatile
- Useful in prototyping IC designs
- Low power
- High number of gates/parts in the same area
- HDL helps create libraries (new gates or parts) and functional blocks
- Do not need to worry about voltage, current, or power to each gate or circuit part

- ❖ Circuit can be simulated and debugged before implementation; therefore saving time and money

FPGAs can be programmed using schematics to layout a design, or by using an HDL like Verilog or VHDL. As well as being cheap to program for an application, FPGA hardware is relatively inexpensive. FPGAs can be used for complex applications, such as image processing or image compression, and are suitable for use in portable phones, digital cameras, and other complex digital applications. Xilinx, as well as other companies sell intellectual property (IP) cores, and opencores.org provides cores and source code for free under the GPL.

Evaluating the performance of an FPGA is difficult. Most manufacturers don't provide a gate count for their products, because it tends to be misleading. Mostly, FPGA manufacturers give a number of "logic units" or "logic cells," which consist of a lookup table and a flip-flop, as well as assorted connection logic.

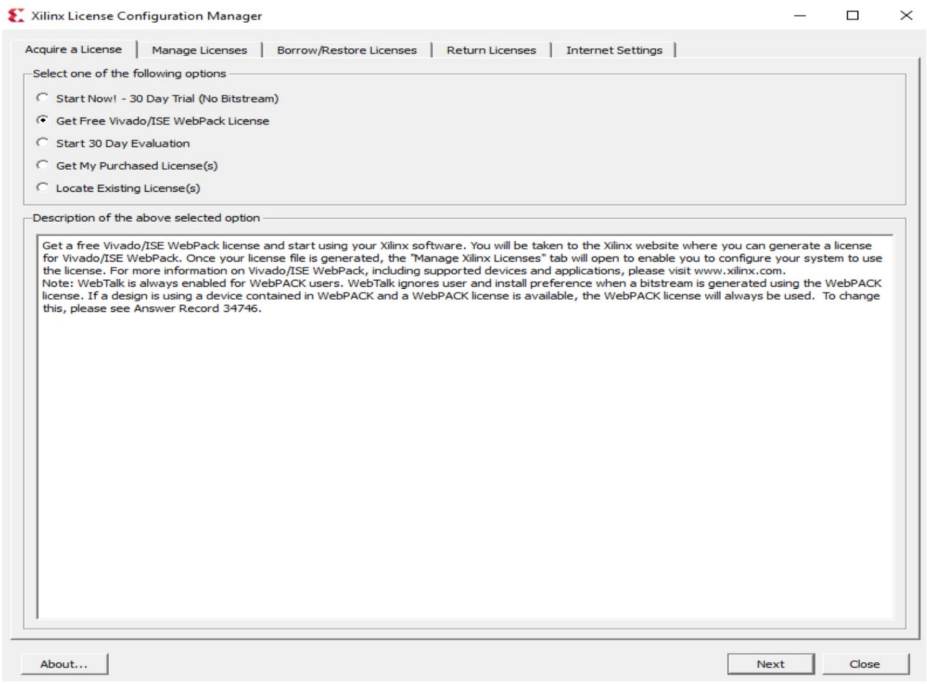
Because different manufacturers use different types of basic gates (AND vs. NAND, etc), the gate count of a logic unit isn't the same across different manufacturers, or even different FPGAs.

FPGAs are becoming more sophisticated, and their applications are expanding as they become more capable. When doing calculations that can be made in parallel, the FPGAs are programmed to deal specifically with this problem. Because of the high speed I/O capabilities of the FPGA, it is ideal for acceleration of certain problems.

II. Installing Software

Installation Notes:

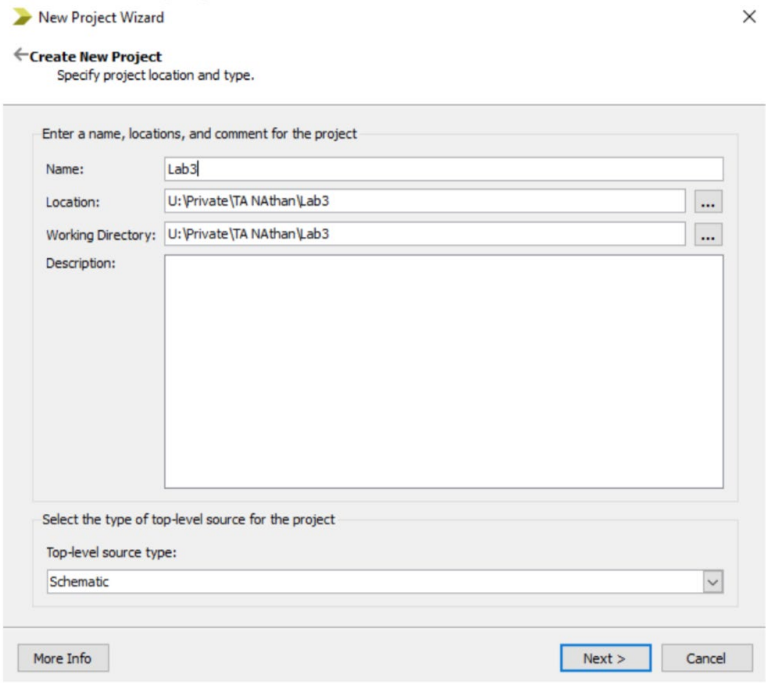
1. With Windows 10, you will need to run the 32-bit version of the ISE Webpack software:
2. You will need to create an account to download the software. Then, fill the verification form,
3. where company name is Elizabethtown College.
4. Only one license can be activated per account, so use a different account for each computer.
5. To activate the license, launch the "Manage Xilinx Licenses" app under the folder Xilinx Design Tools. This can be found by clicking "All Apps" after pressing the Windows button.
6. Select the "Get Free Vivado/ISE Webpack License" option to acquire a license. Then fill in the information to download a license folder. Then, go to the "Manage Licenses" tab to load the license file and you can now launch the fully functional free software.



III. Using Xilinx Project Navigator

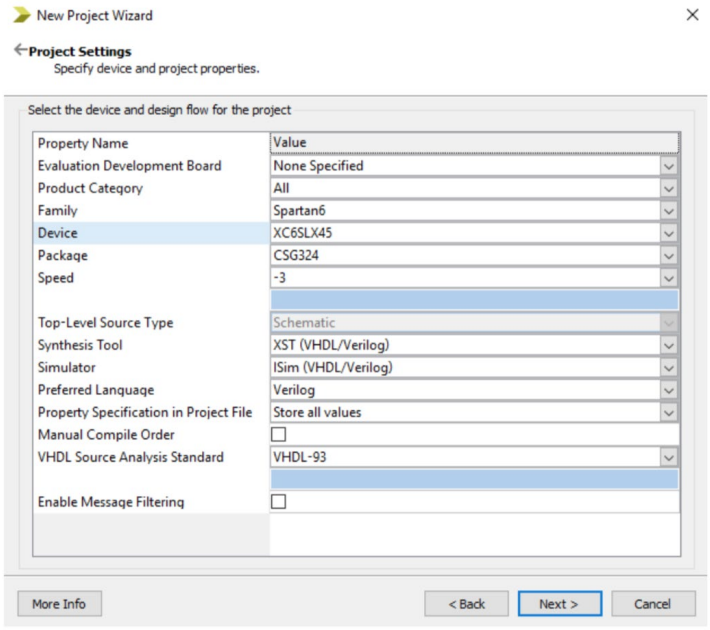
a) Start program called “32-bit Project Navigator”. It can be found by pressing the start menu, “All Apps”, “Xilinx Design Tools”

b) Create a new project



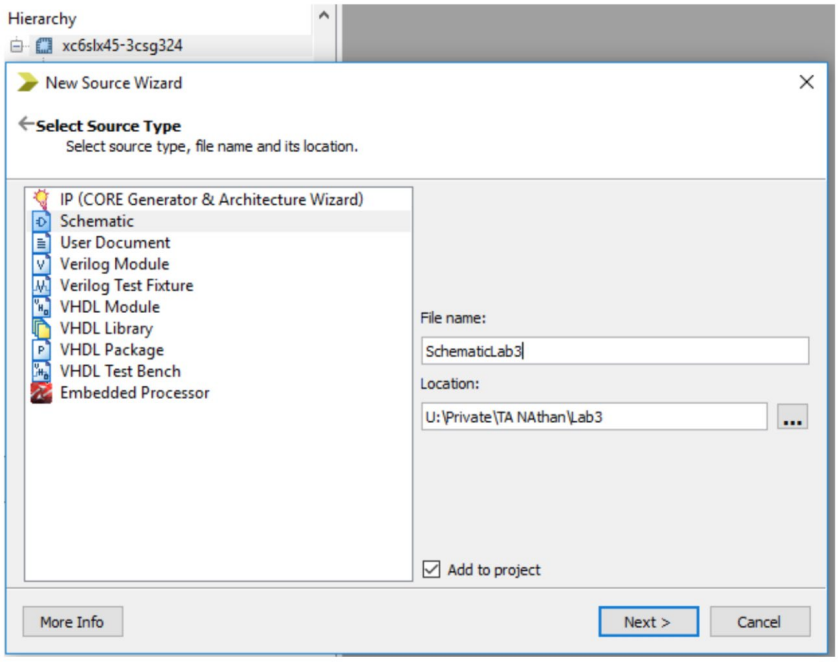
c) Choose a name, and make sure to save it in a location that is not read-only (your public or private folder will work). Change the Top-level source type to “Schematic”. Press “Next”

d) Select “Spartan” in the “Family” box, “XC6SLX45” in the device box and “CSG324” in the “Package” box

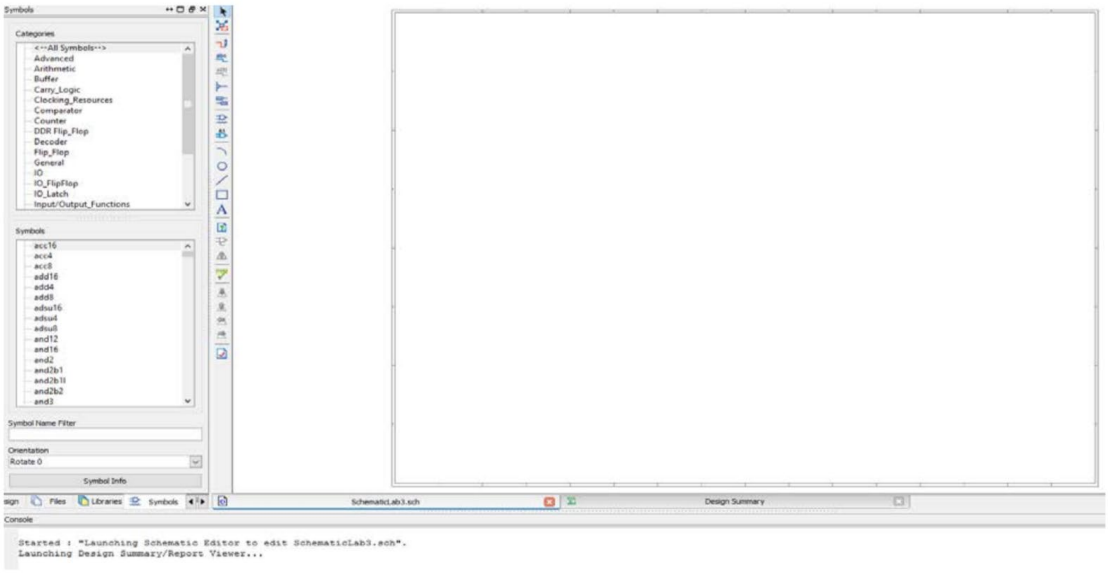


e) When the “Project Summary” box appears, select “Finish”

f) Under the Hierarchy window on the top left, right-click on “xc6slx45-3csg324”, and choose “New Source”. Press “Finish”



g) The schematic window will appear and you will be able to navigate between the hierarchy window under the “Design” Tab and the logic circuit component under the “Symbols” Tab

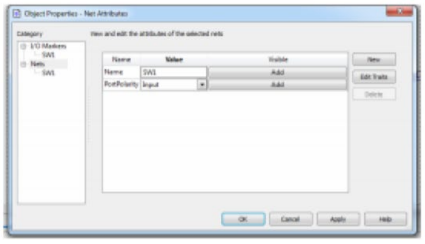


h) Like Logisim, the Symbol tool lets you add parts by connecting the respective leads together with the mouse or with the wire drawing tool. Common gates like the AND gate are located under the "Logic" category. To create an Input, you must first add an "IBUF" from the IO category. Then, click "I/O Marker" and click the first leg. For an output, add an Obuf and add the I/O marker to the second leg.

iii. ***Input(s) & Output(s):*** Must be assigned **PIN LOCALIZATION NUMBERS**. (see step l)

i. Inputs and outputs should be named in such a way to help illustrate their purpose

i. To input a name, double click on the symbol, click "Nets" in the category on the left of the window that appears and type the desired name in the value field under the "Name" row



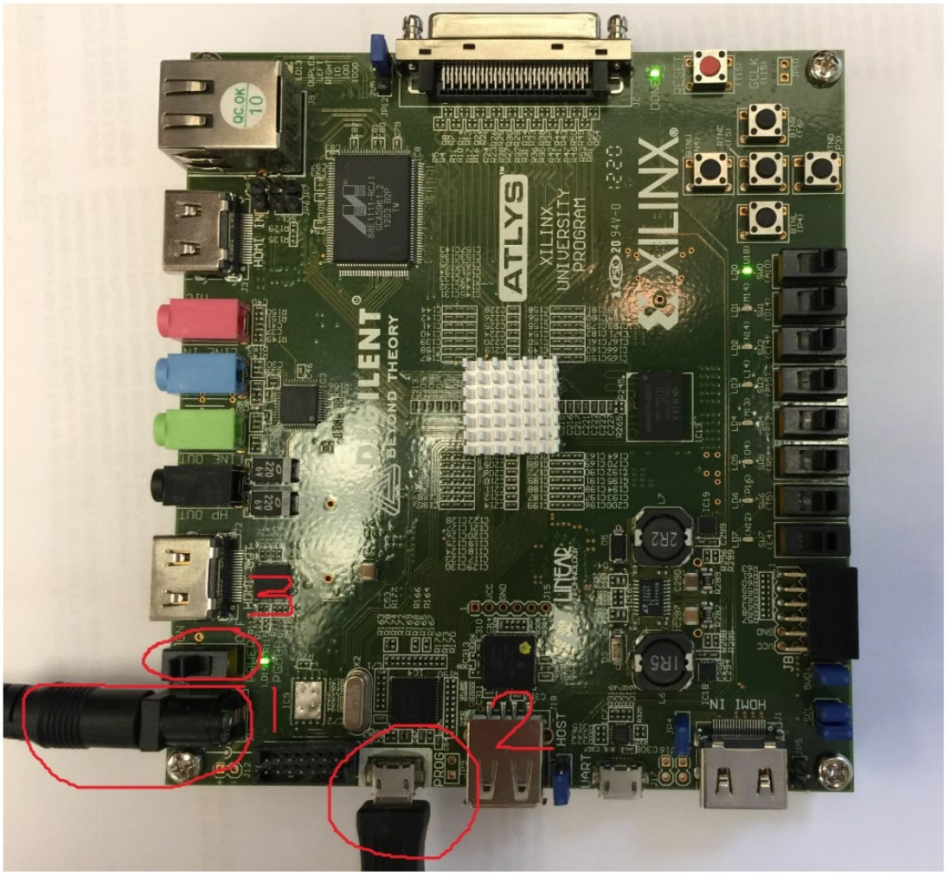
You created the constraint file

IV. Implementation / Programming

Make sure to use the power cord from the Xilinx box. It should be a 5V output. Any voltage superior to this could permanently break the FPGA board.

a) Once the circuit is properly designed, you can proceed to implement your circuit on the FPGA.

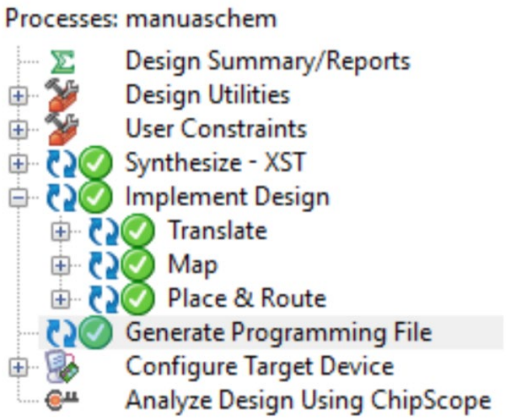
b) Setup the FPGA board in three steps as illustrated in the picture below:



Plug in the power cord, then the usb cable from the board to the computer, and finally turn on the power switch.

c) Click on the “Implement” button (green arrow on the middle left of the design window)

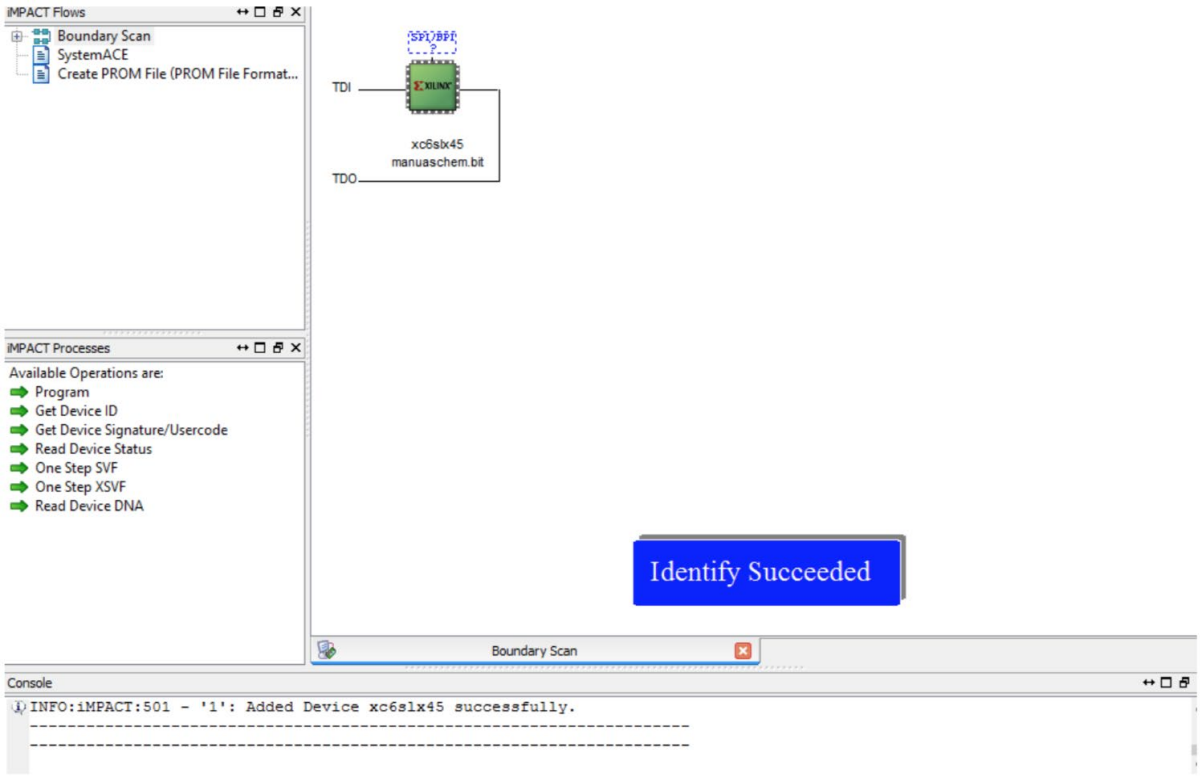
d) In the hierarchy, click on the schematic file. On the bottom left, you should see the list of processes. Click on “Synthesize –XST” and “Generate Programming File”. After that, you should see only green status like in the picture below:



e) The next step is to download the circuit into the FPGA. To accomplish this, press the Tools tab then click on “Impact”, and press “OK” to the warning message.

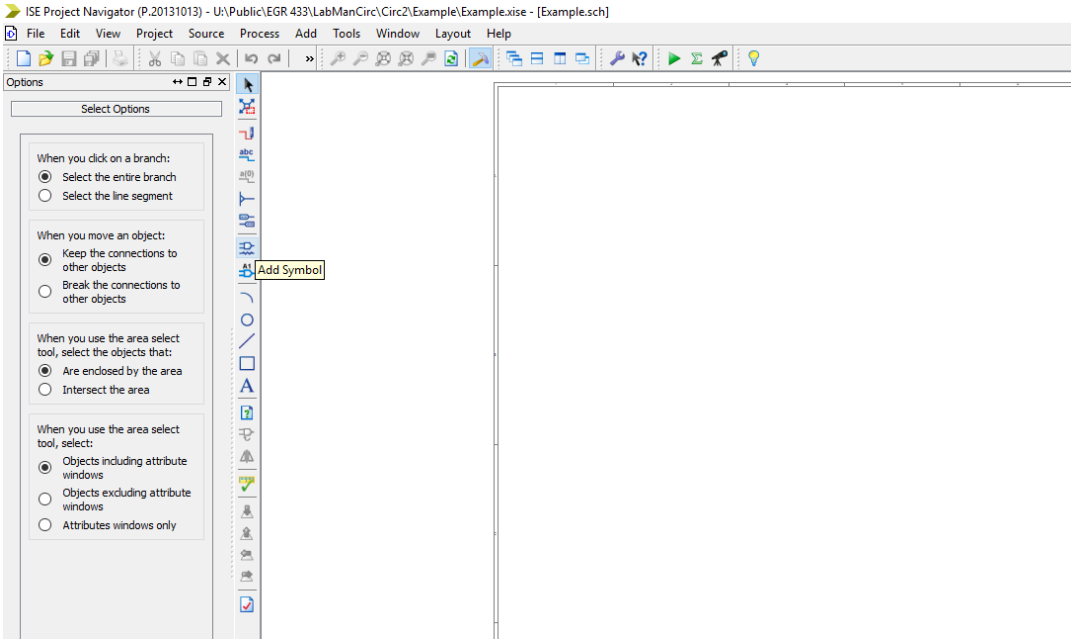
f) Double-click “Boundary Scan”, then right-click and select “Initialize Chain”. A message should pop-up to assign configuration files, press “Yes”. Then open the saved schematic. Another message should pop-up about Flash PROMs, you should press “No”. Then the Device Programming Properties window will come up and click on Device 1 under Boundary-Scan and press “OK”.

g) Finally, if everything went well, you should see the equivalent of the picture below. To launch the program, press “Program” under Available Operations. It should display “Program Succeeded”

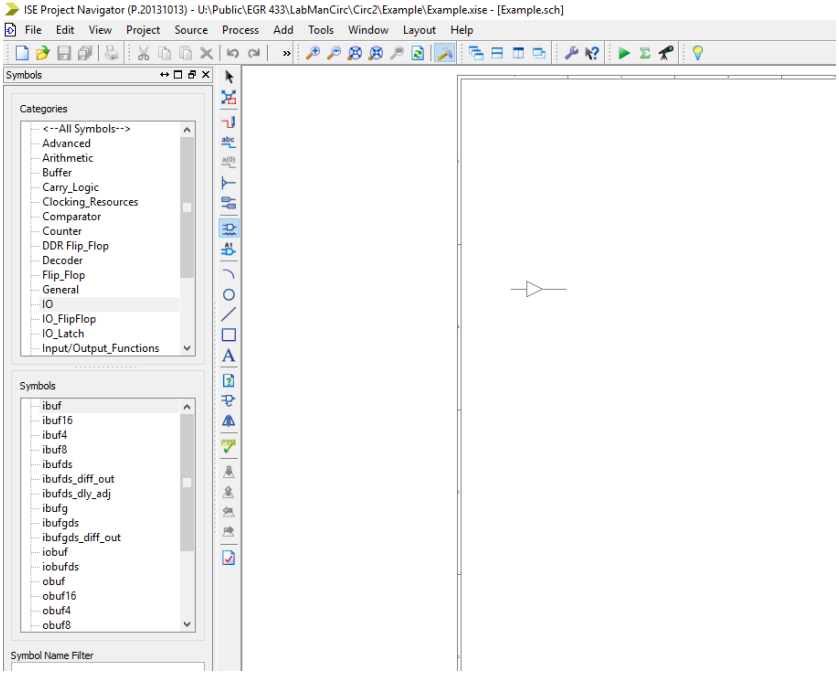


Example Circuits

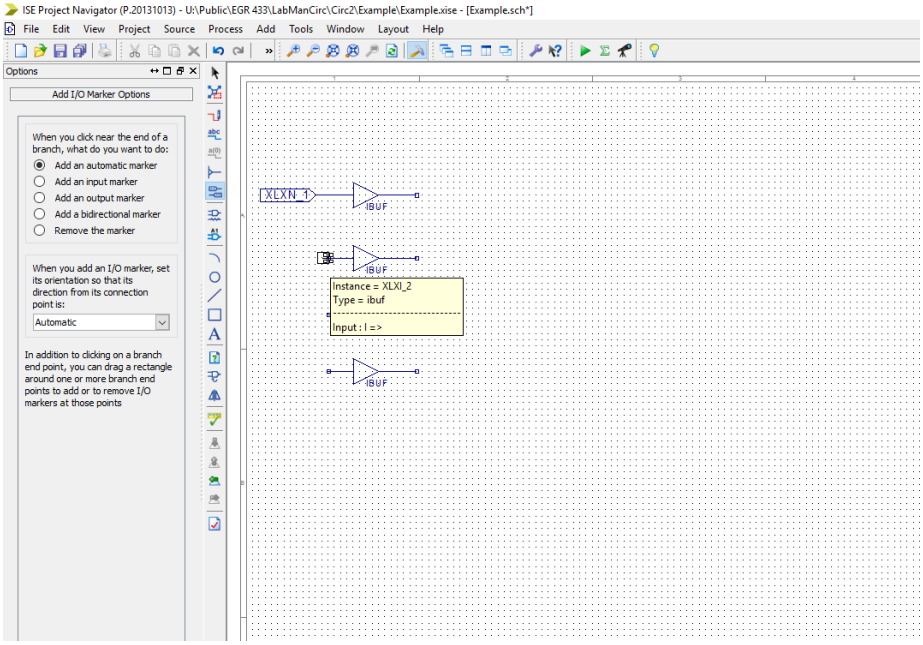
Let’s start by creating a simple four person voting machine that states if option A won, option B won, or a tie occurred. Start by creating a new project and schematic as previously described.



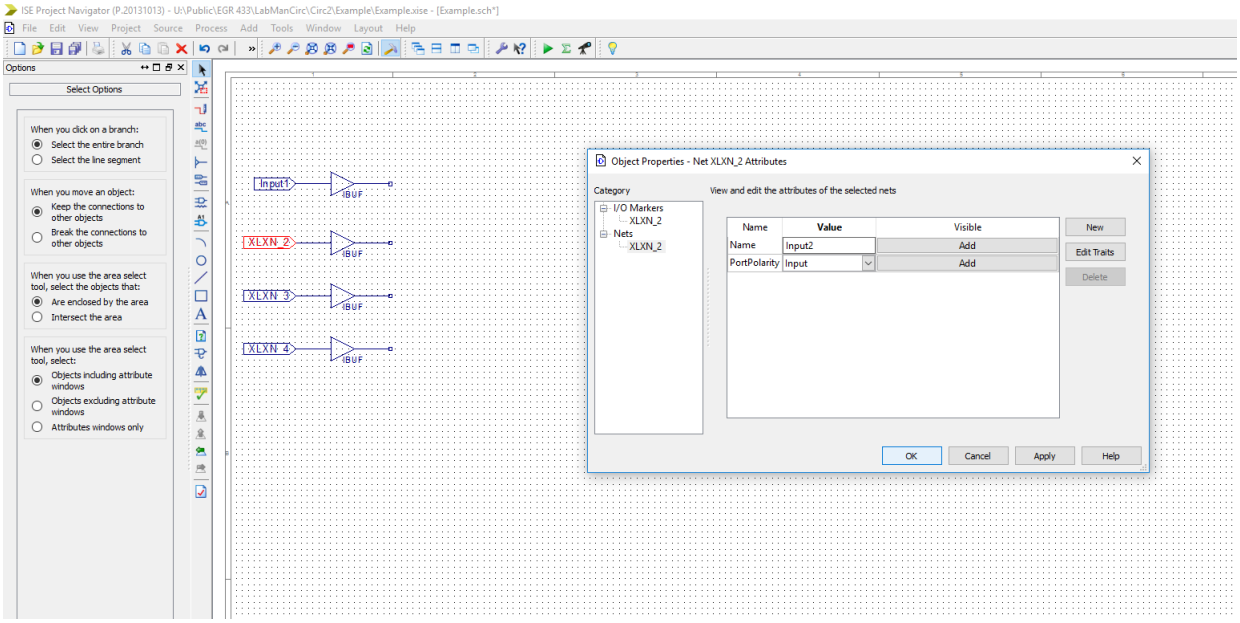
Click on the above icon to access the option to add symbols.



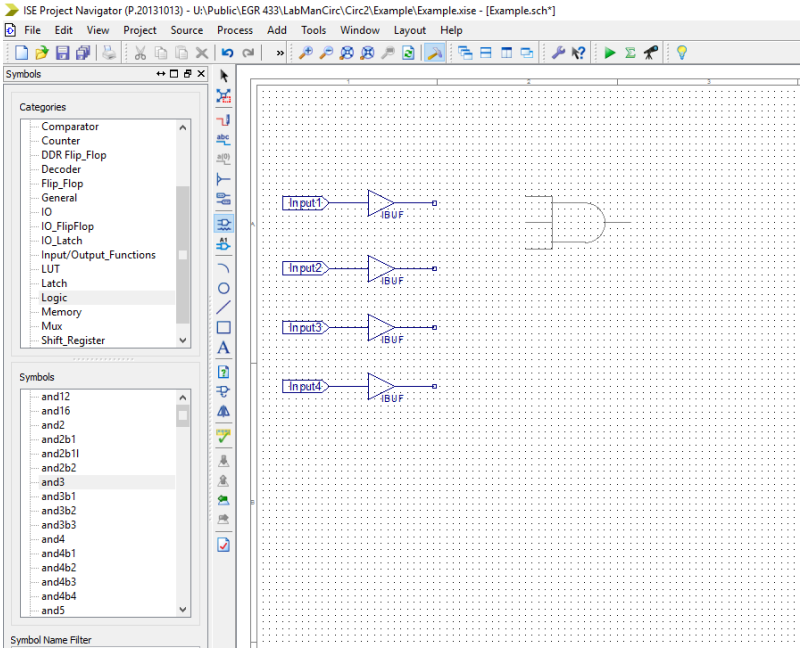
Under the IO Components, select the input buffer (ibuf). Place four of them on the schematic.



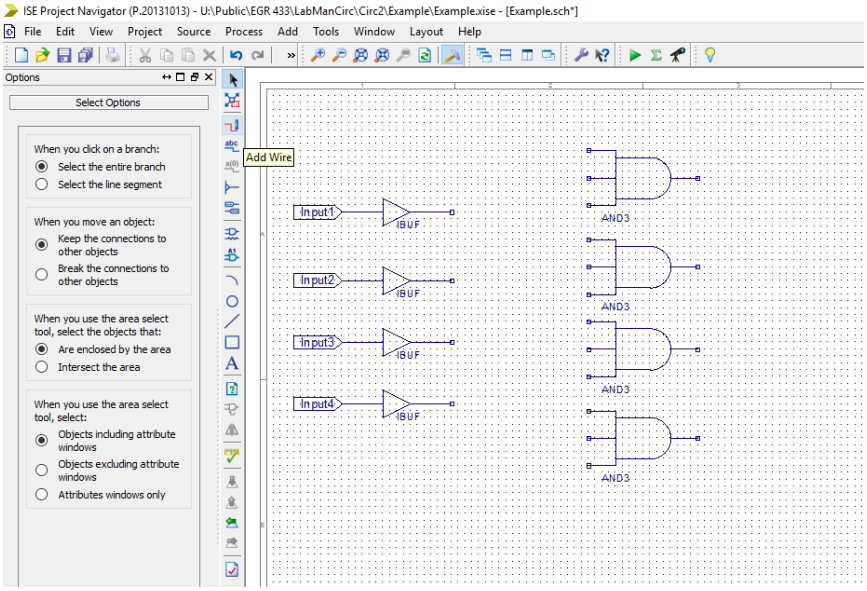
Click on the above icon to access the I/O markers. Put an input marker (or an automatic) on the left side of each of the input buffers.



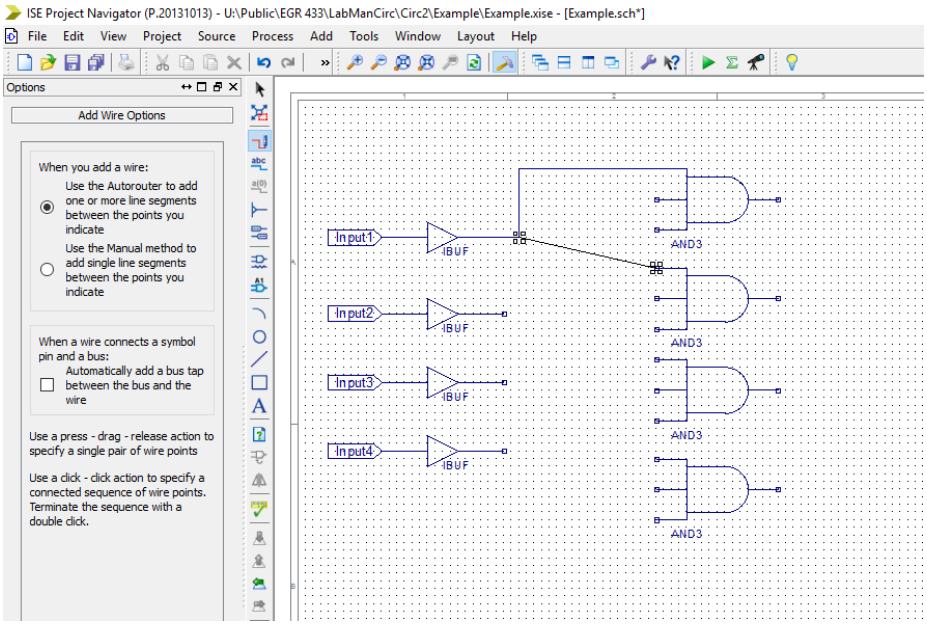
You can double click on the input markers to change their name. To do so, go to the XLIN_2 under Nets and type the desired name in the name field.



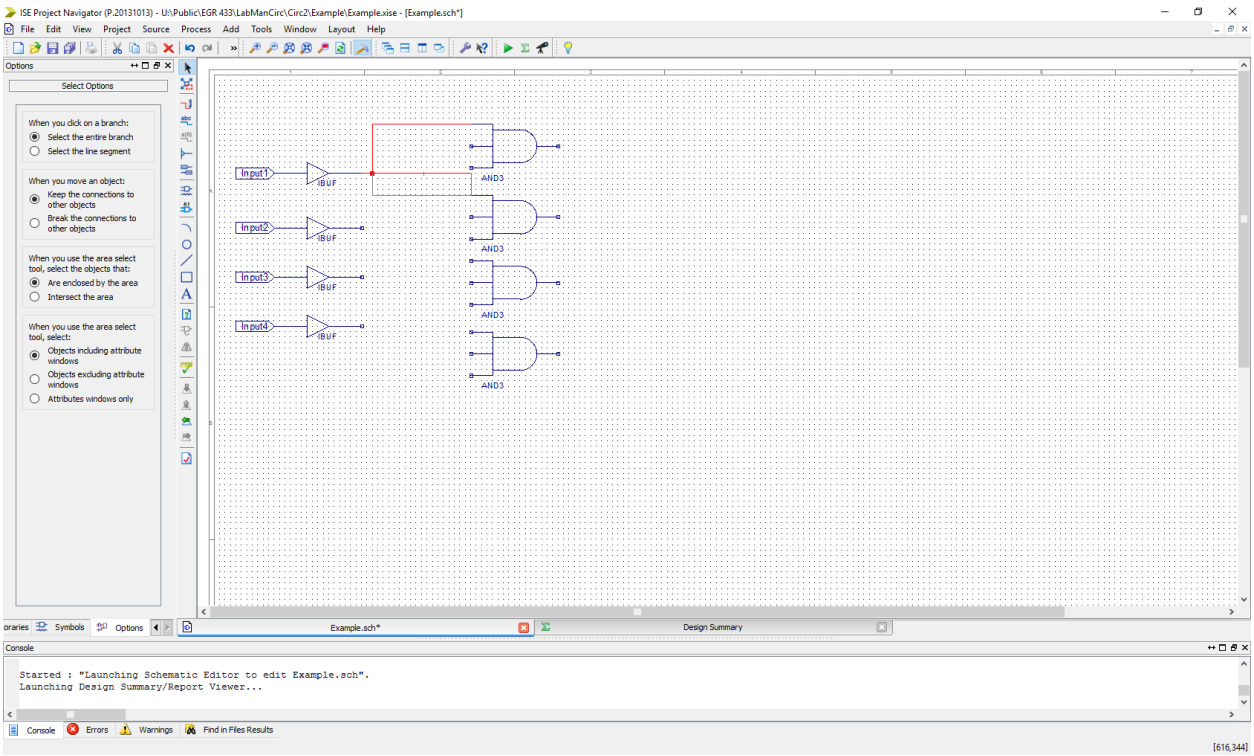
Next, go back to the icon to add symbols, but this time go to logic. Click on the three-input AND gate (and3) put four of them on the schematic.



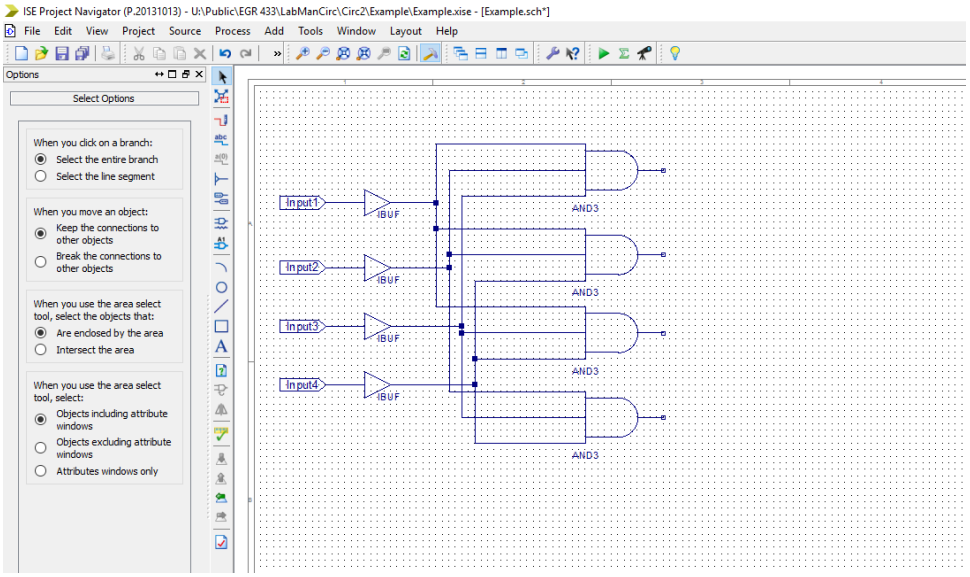
Click on this icon to start adding wires to the circuit.



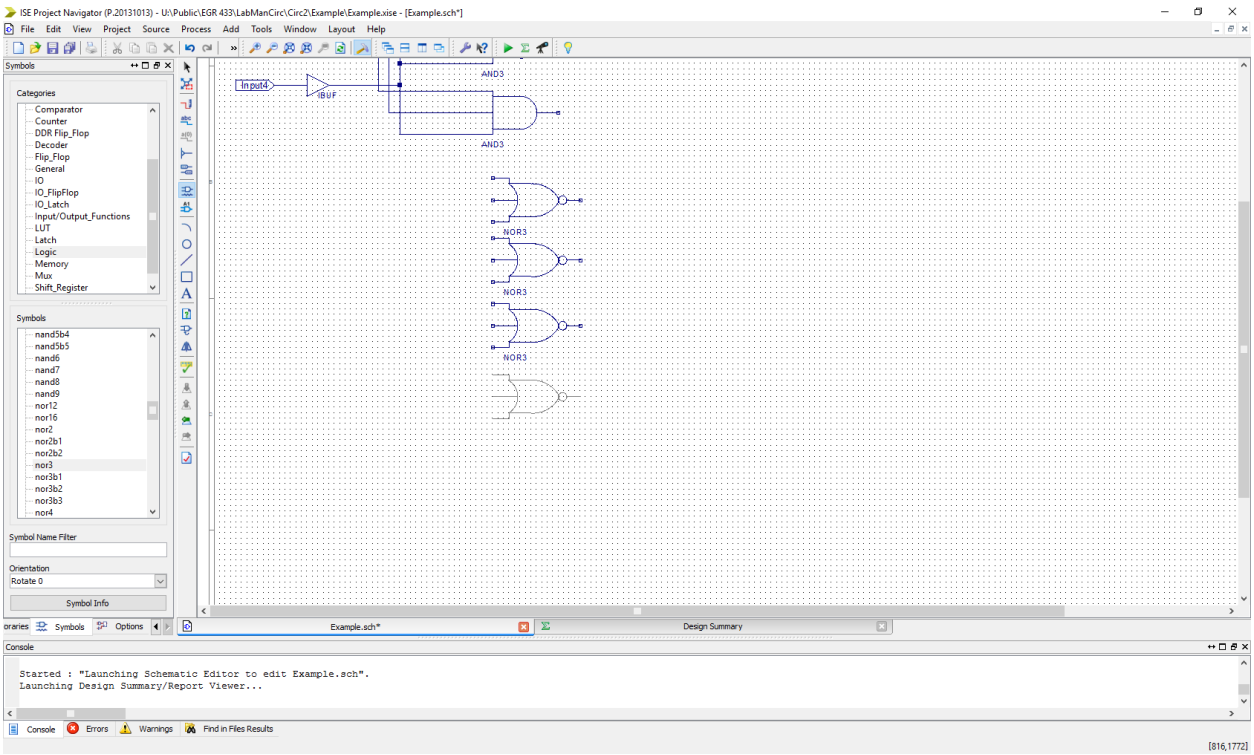
To add a wire to the schematic, either click on two different points or drag the wire from one point to another. These points can either be a blank space, one of the tiny blocks connected to each symbol, or another wire.



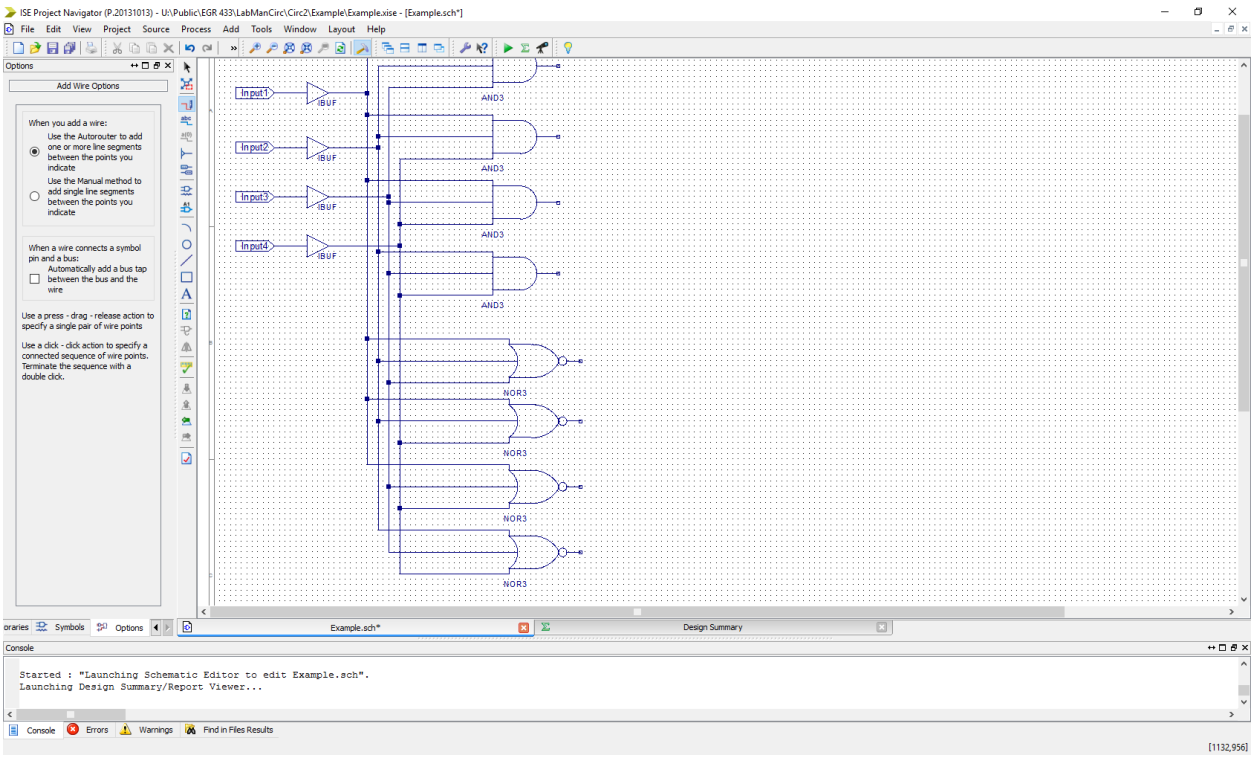
If you do not like how Xilinx automatically placed the wire, you can change its position by clicking on the wire and dragging it.



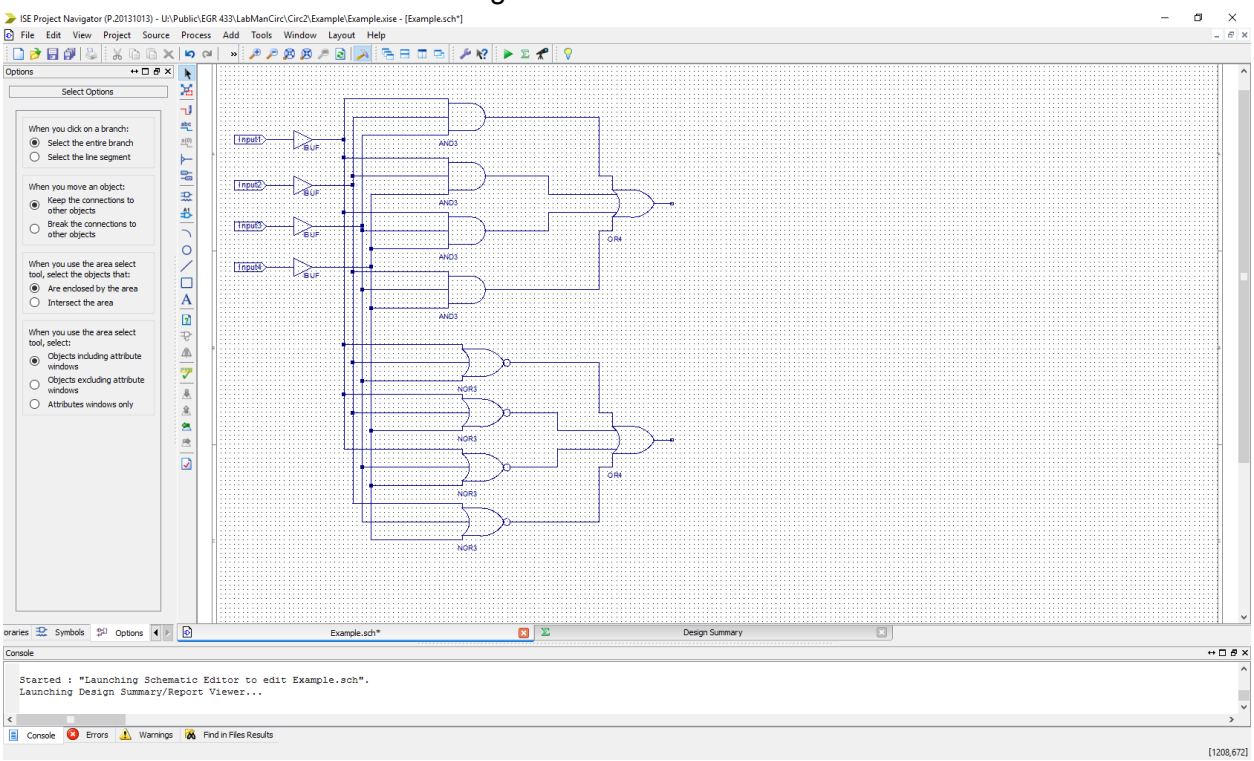
Wire the four AND gates as shown above.



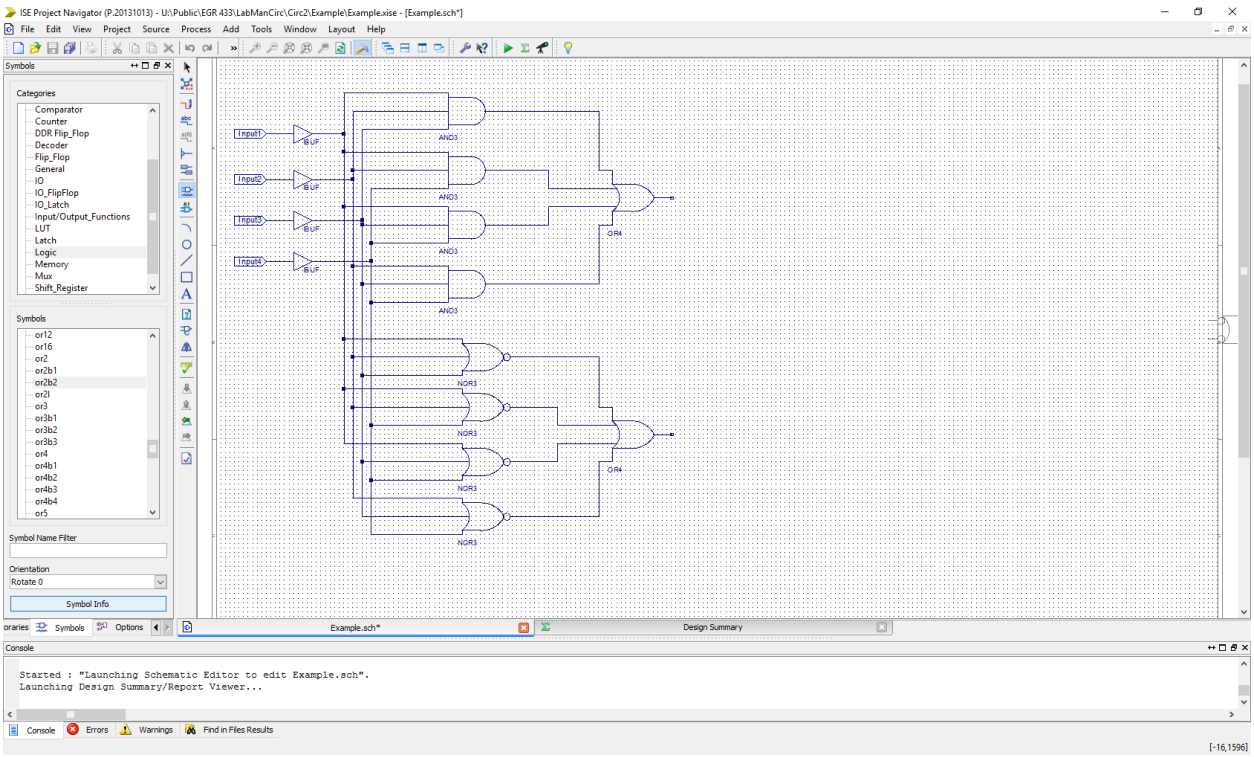
Next, place four three-input NOR gates (nor3).



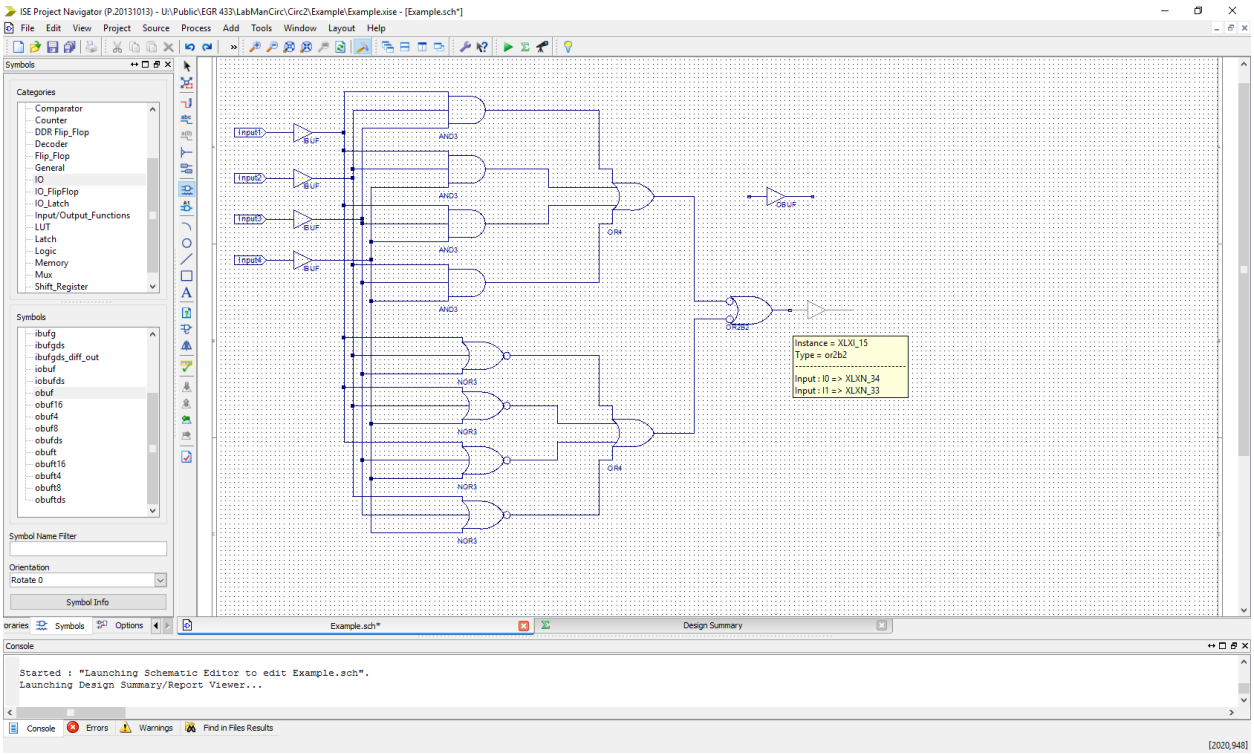
Wire them the same as the four AND gates.



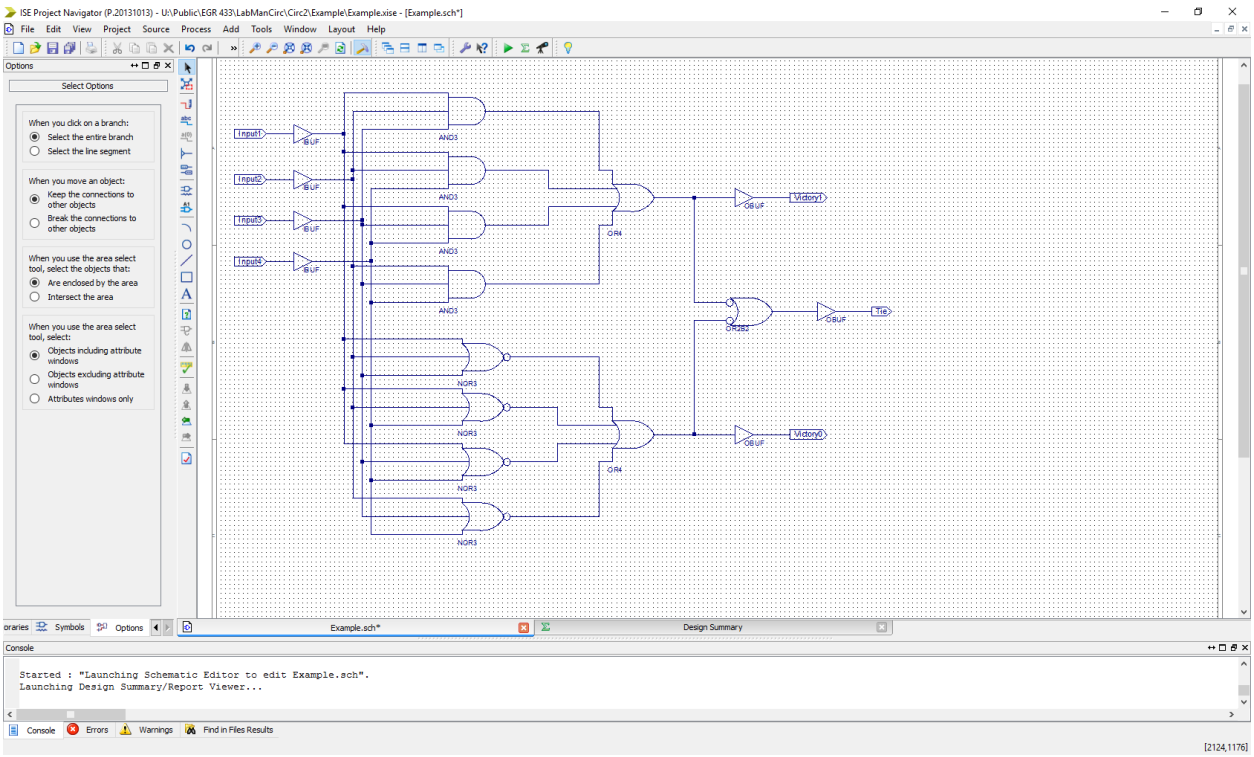
Add two four-input OR gates (or4) and wire each of the AND gates to one and each of the NOR gates to the other.



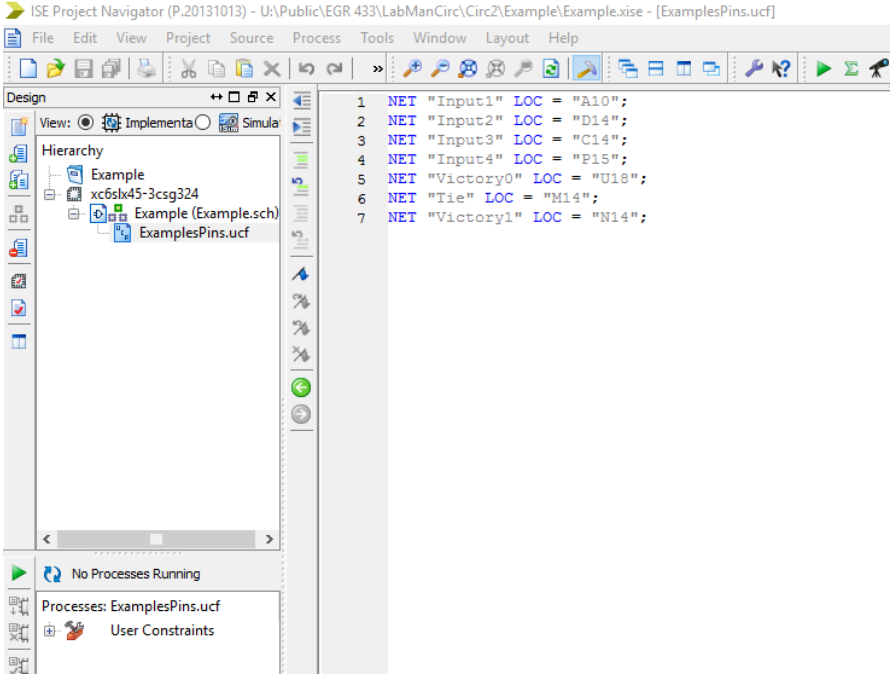
Next, we are going to add a symbol that tells us when it is a tie (neither of the four-input OR gates have a value of 1). Instead of using a two-input NOR gate, let's use the and2b2 gate. To figure out what it is, click on the name of the symbol and then click Symbol Info. Place it on the schematic and wire both OR gates to it.



Go back to IO and add three output buffers (obuf).

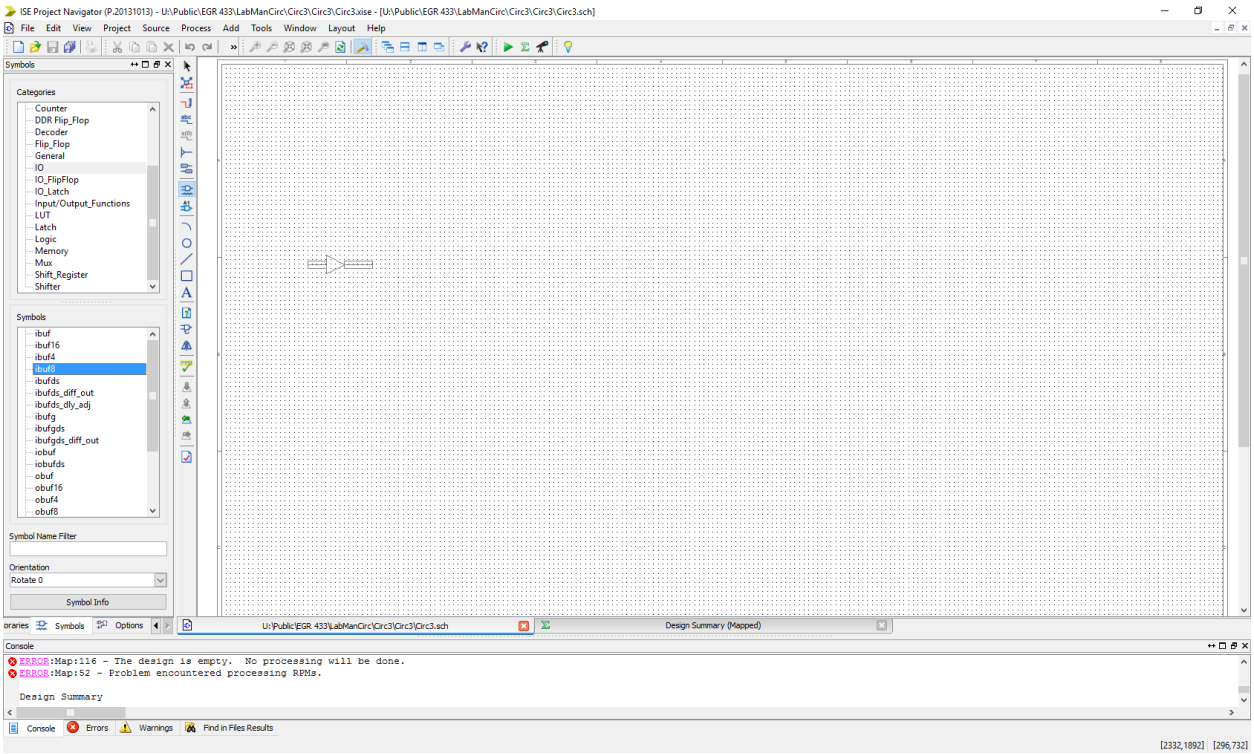


Connect three output markers received from the same location as the input markers on the right side of each output buffer. This is what the completed circuit should look like.

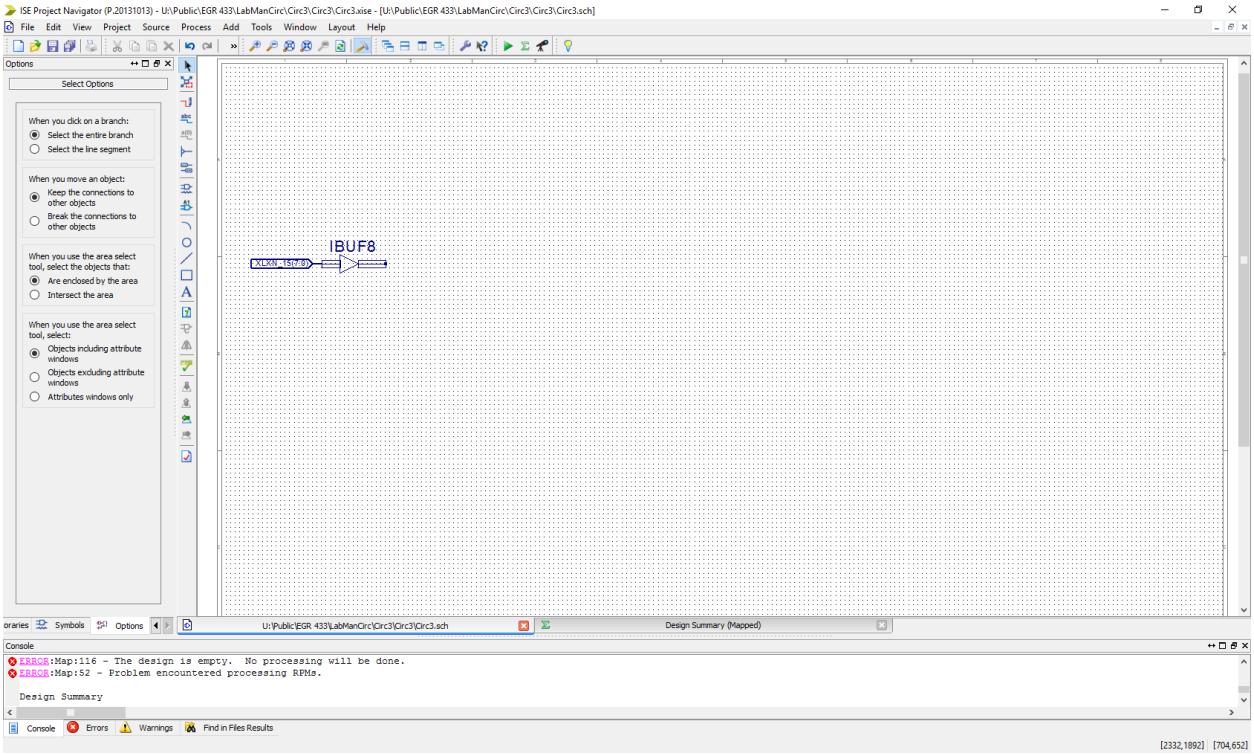


The final step is to create the implementation constraints file and state where the inputs and outputs are to be located on the FGPA. You are now done with the first example circuits.

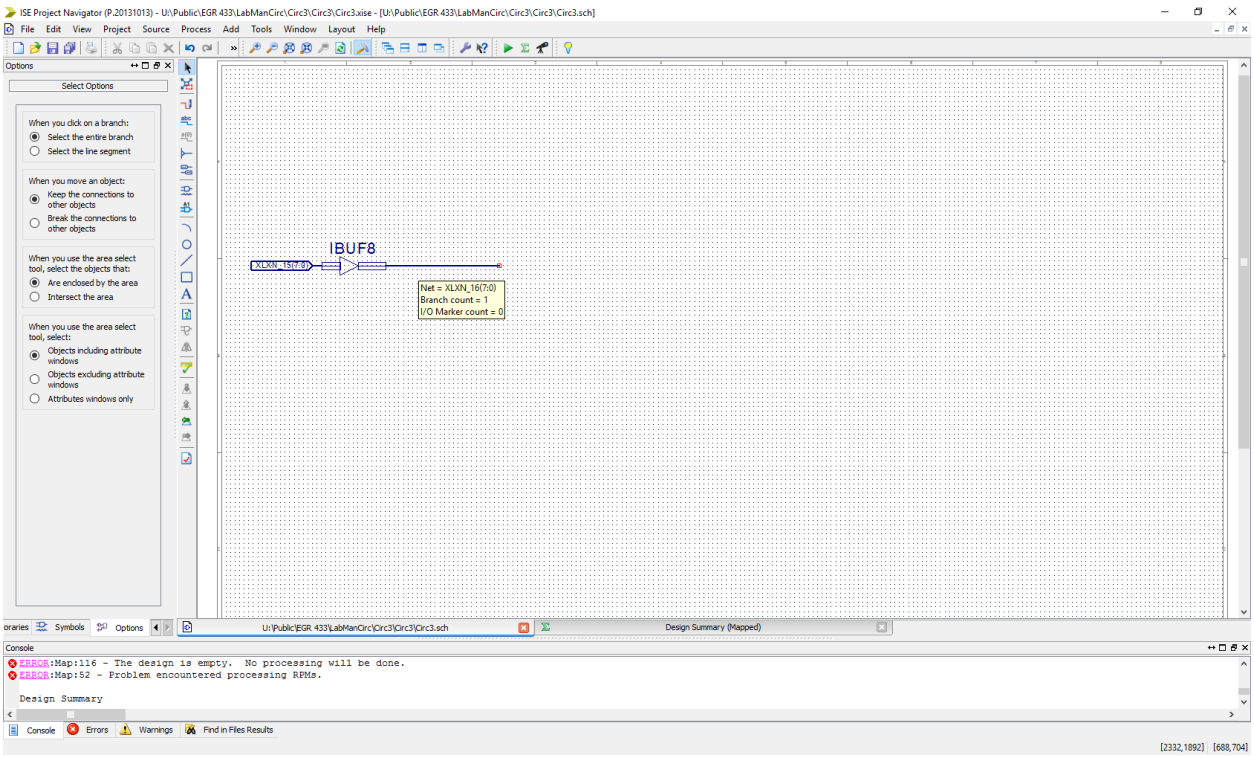
Next, let's create an example schematic that makes use of buses in Xilinx. We will compare an eight bit input from the switches on the FGPA to a customizable constant.



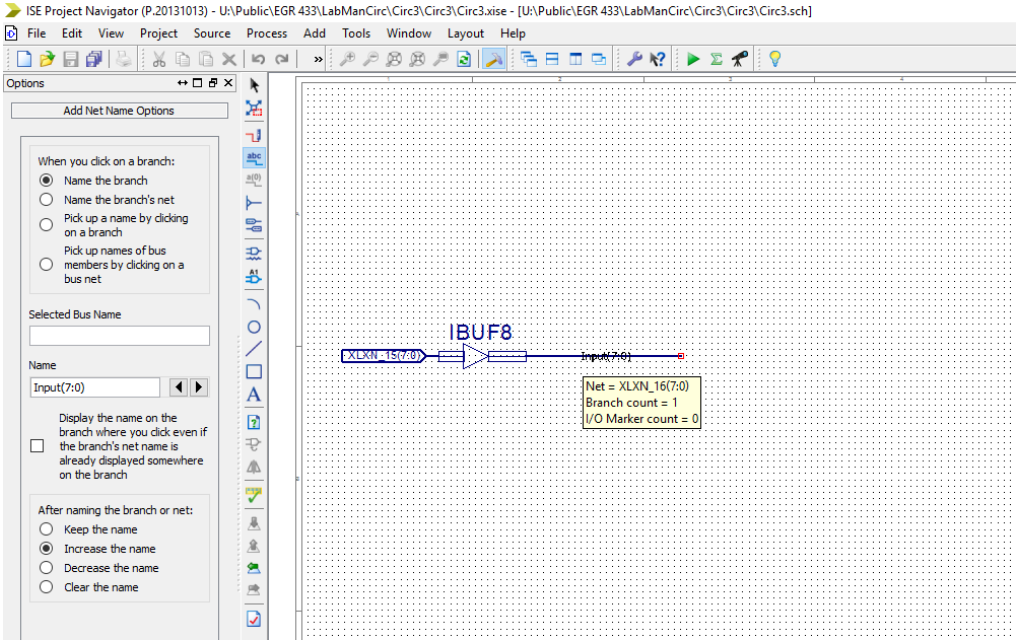
In the IO symbols, select the eight-input buffer (ibuf8).



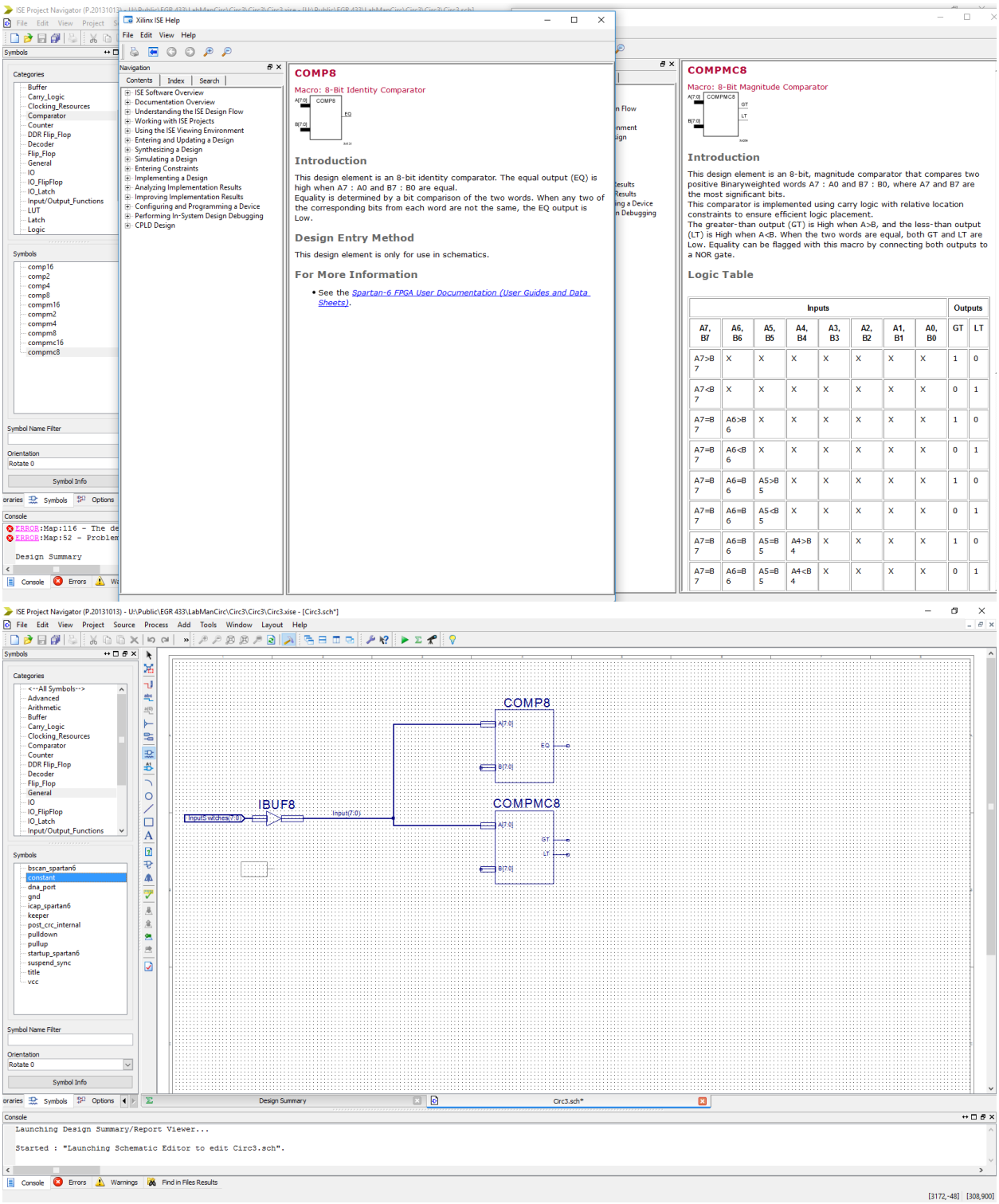
Connect an input marker to it.



Create a wire going from the input buffer.

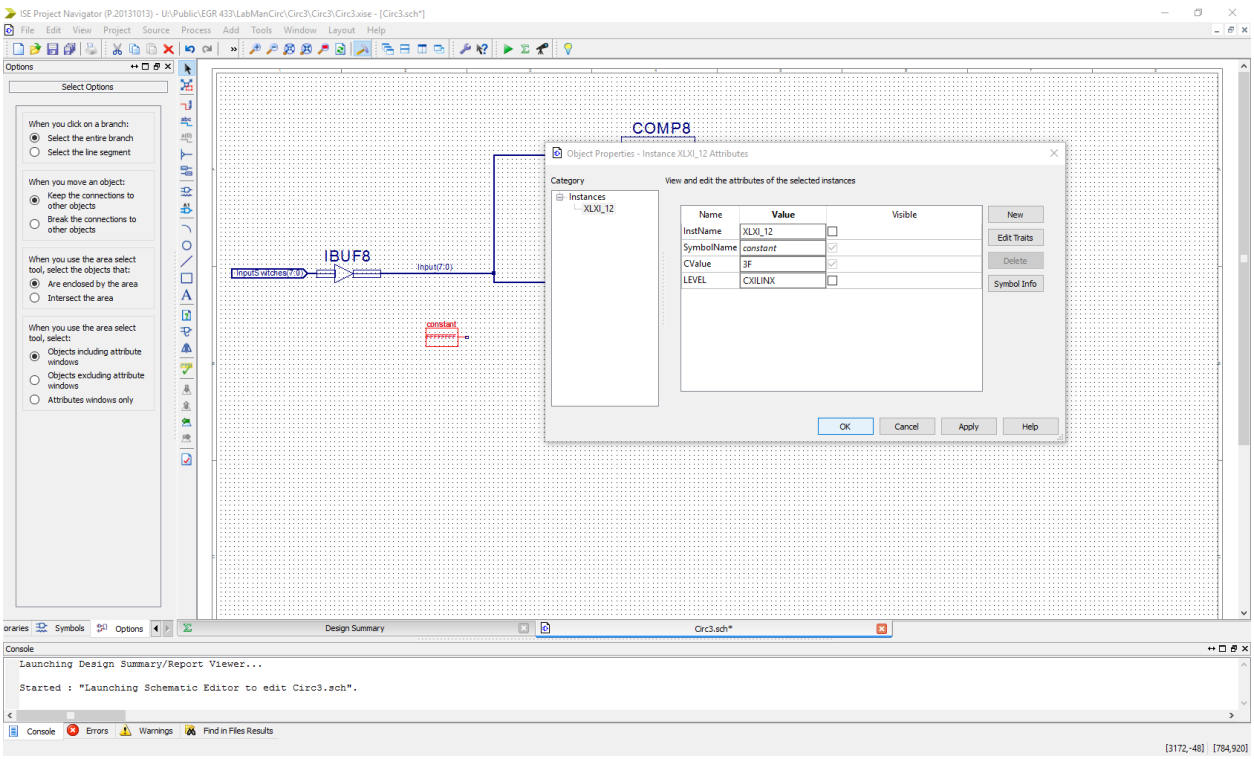


Click on the Net Namer icon, input a name for the net. Pay careful attention to the syntax. For an eight bit but, you need to include (7:0) at the end of the name. Click on the wire to add this net name to it.

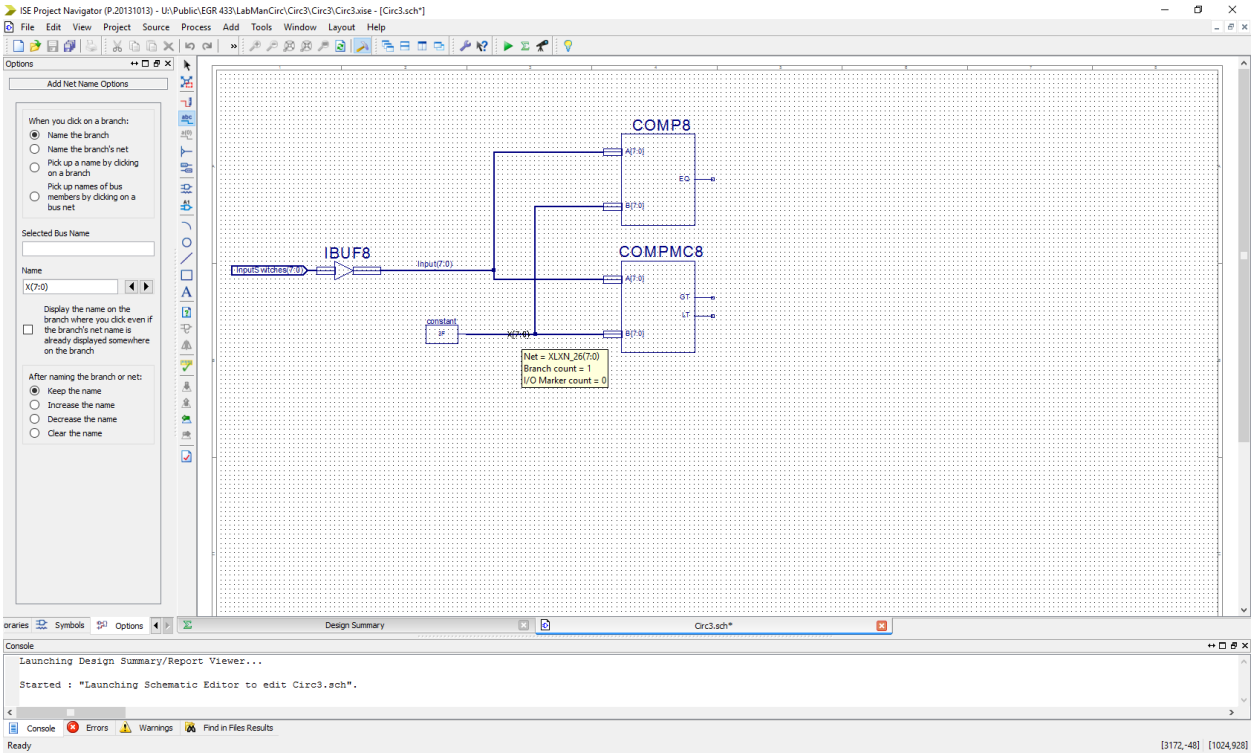


There are two different comparators in Xilinx, one that says whether or not they are equal and one that says whether the first input is greater than or less than the second. We will be using

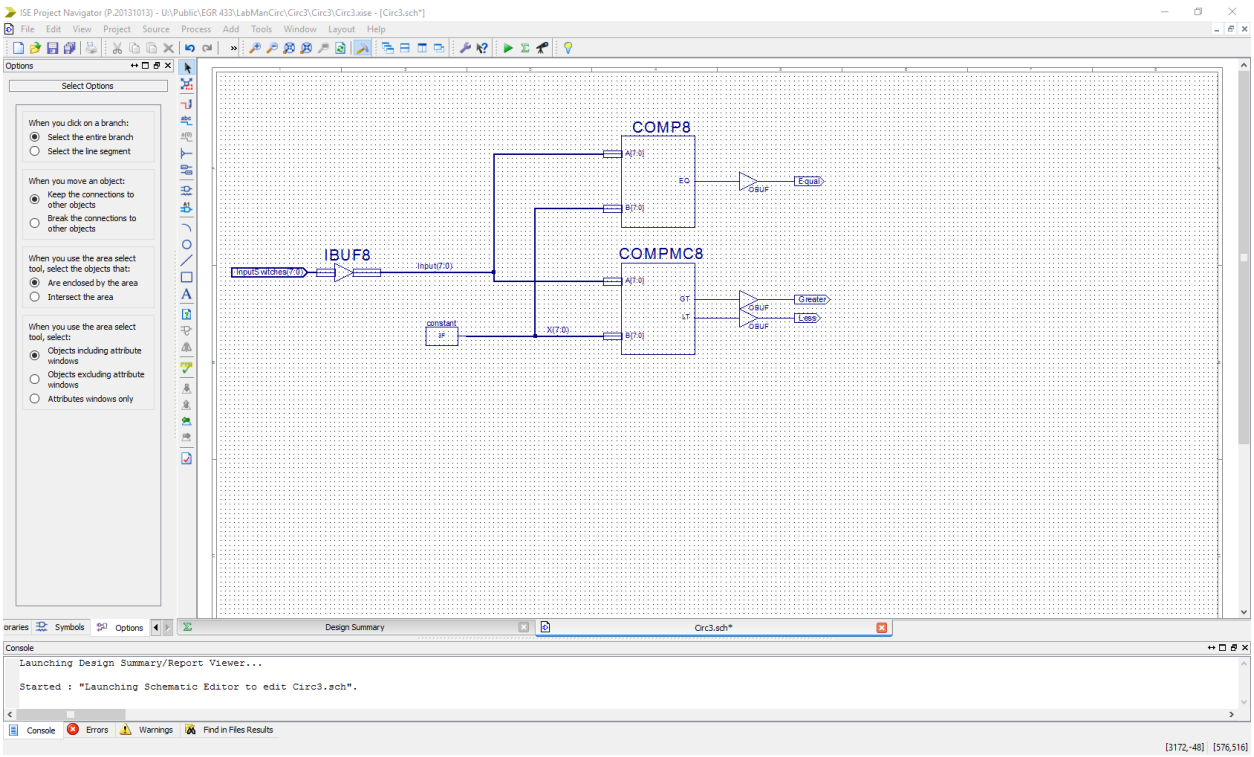
both.



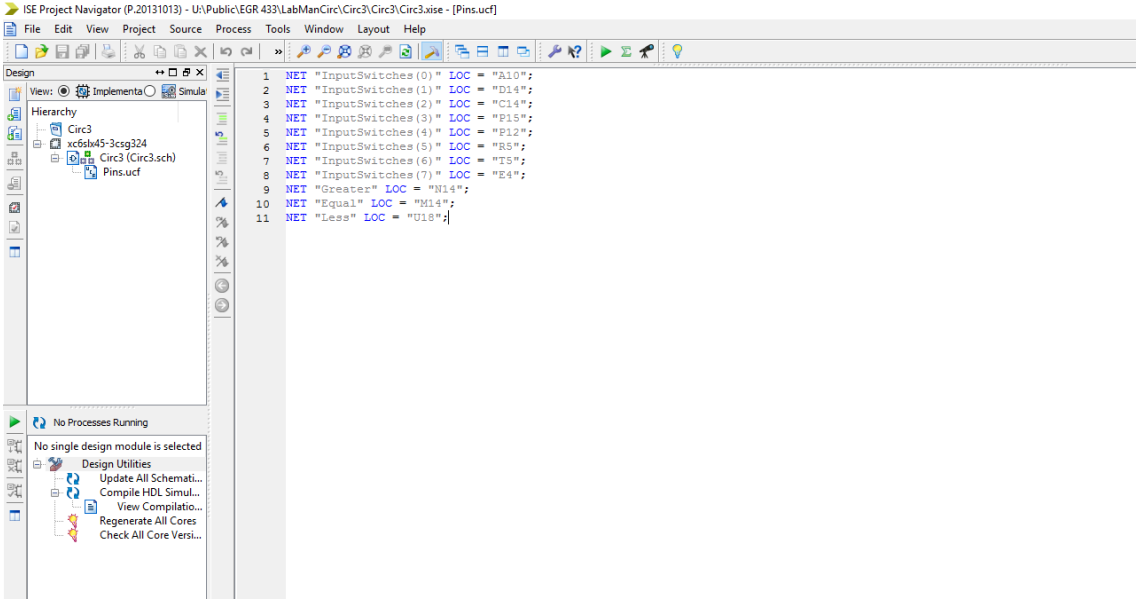
Next, create a constant. It can be found under the general category in Symbols. Efit the constant to be an eight bit number (two hex digits).



Attach the constant to the second input in each of the comparators.

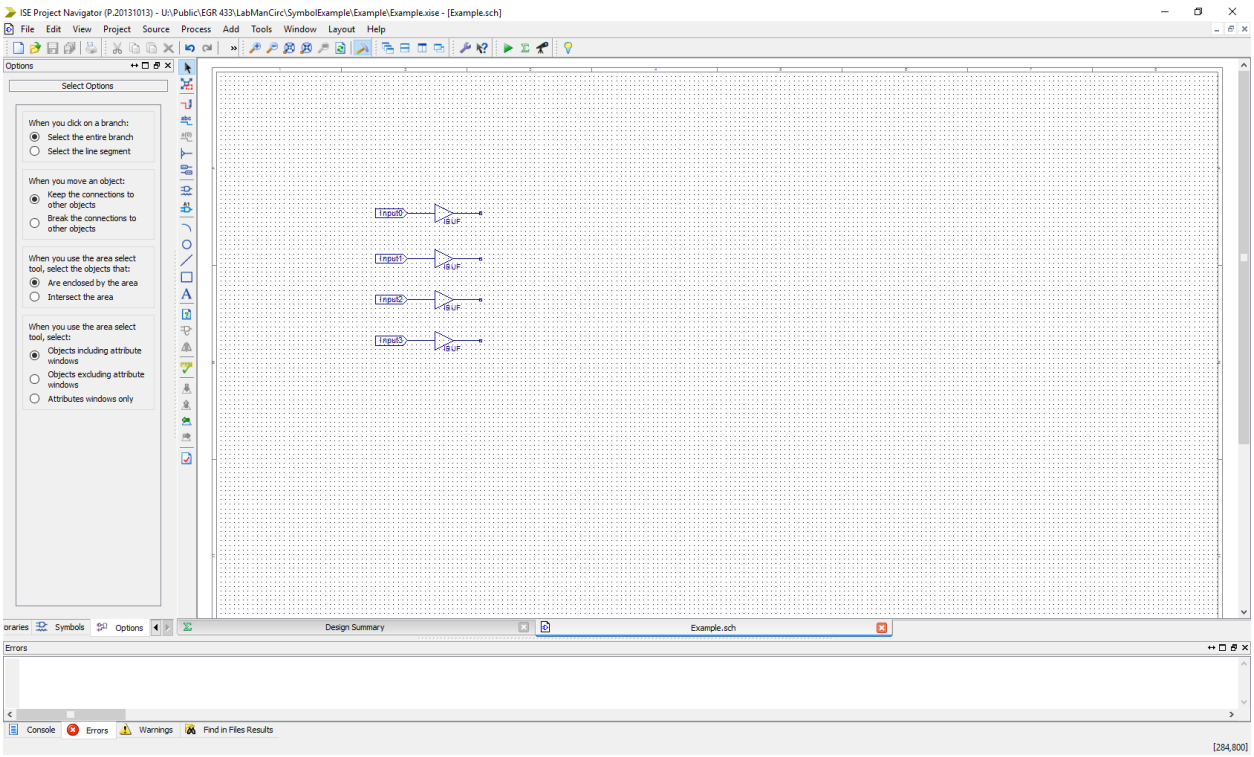


Add output buffers and output markers to each of the outputs from the comparators.

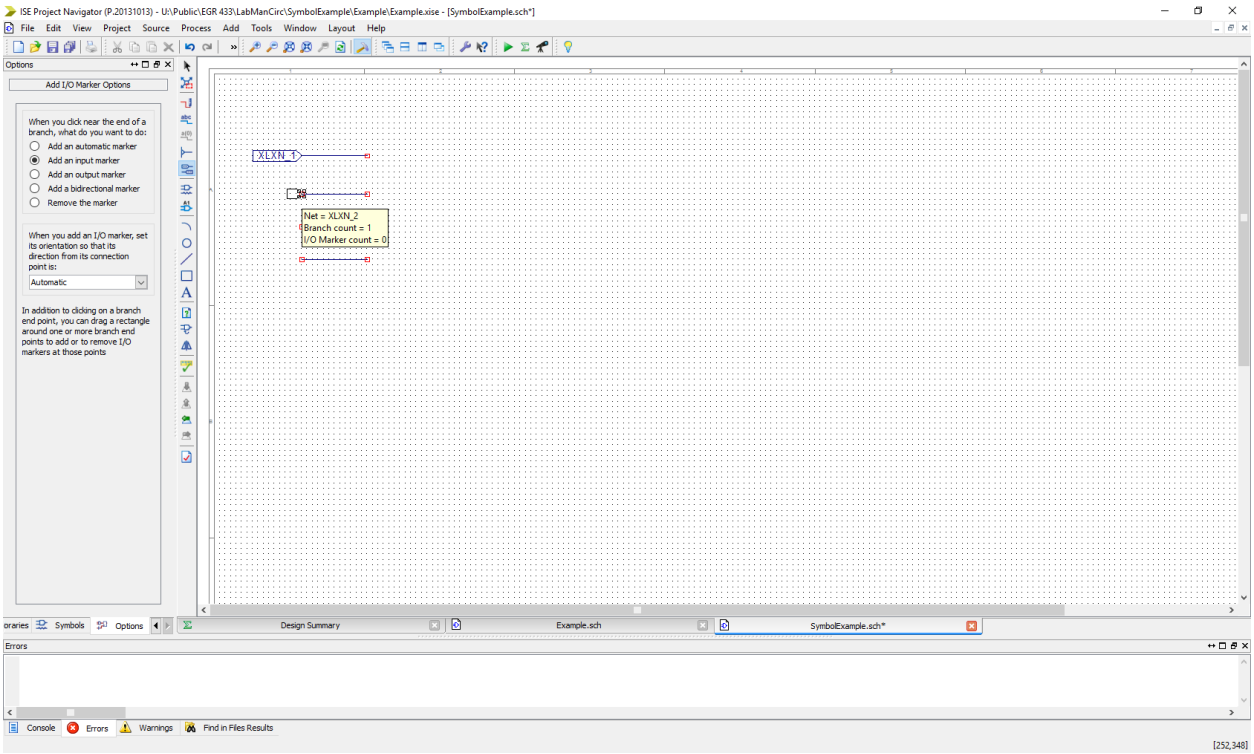


Finally, create the implementation constraints file add state where each of the pins are located.

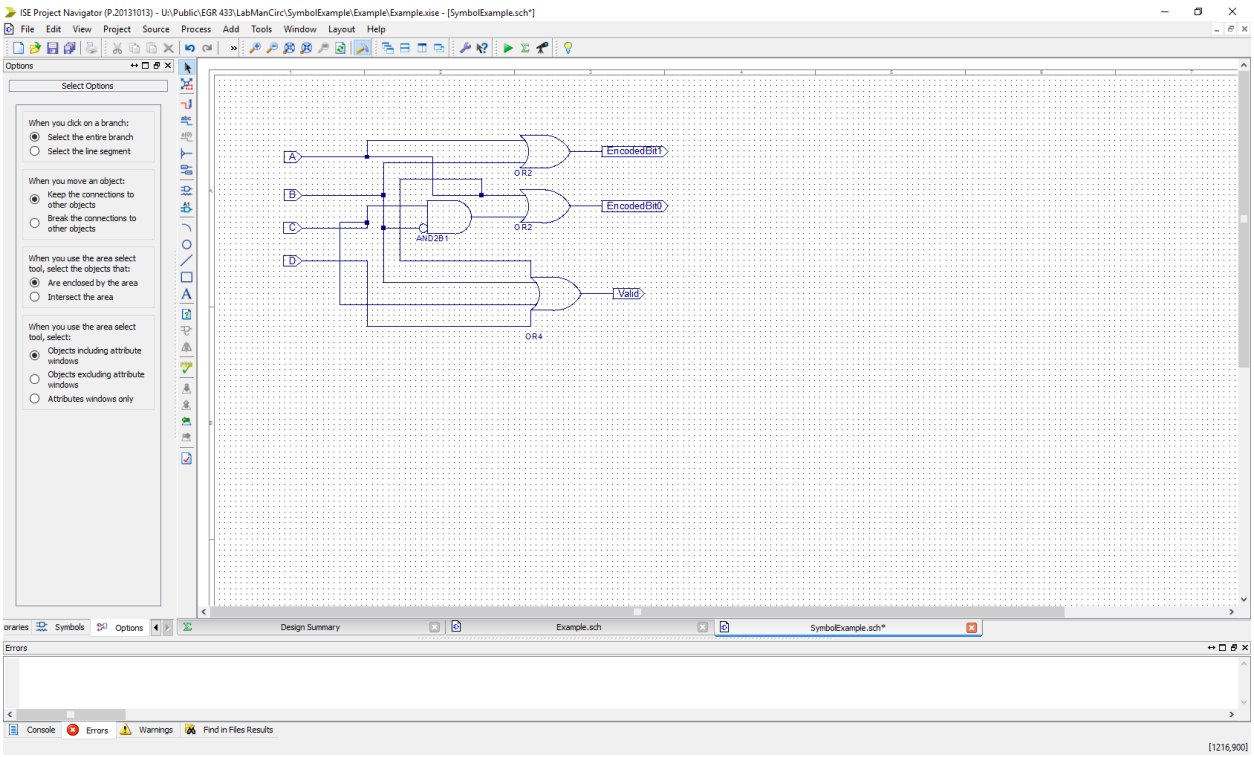
The last example circuit will go over how to create custom symbols. Start by creating two schematics, one will be the main schematic and the other the schematic to create the symbol.



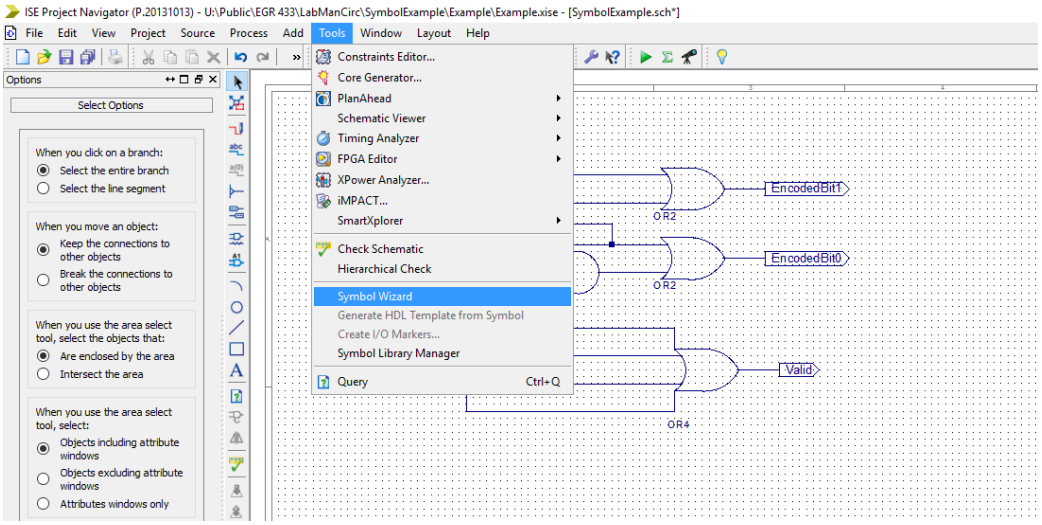
Let's start by creating the inputs for the main schematic.



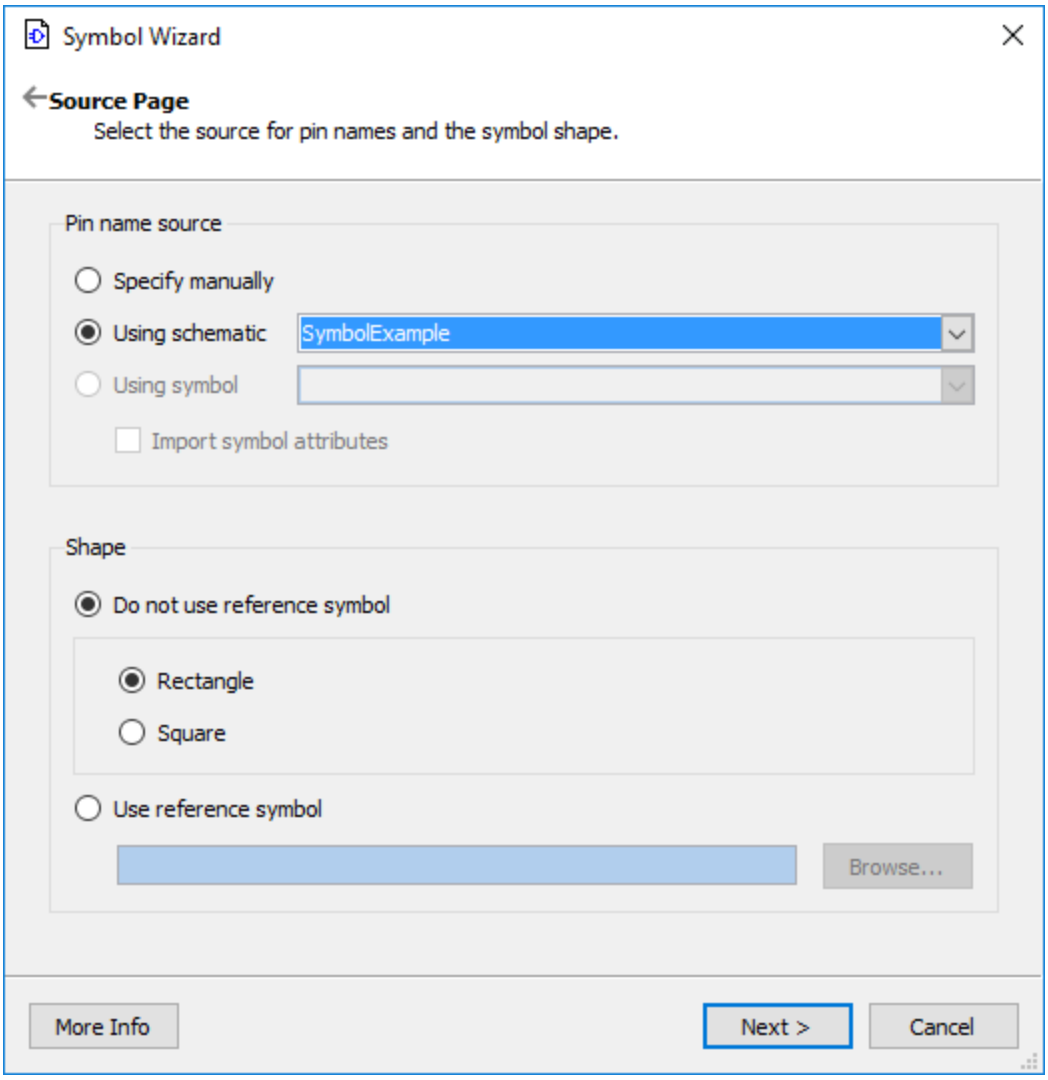
Next, let's move on to the inputs for the symbol. It is important to note that you can not include input buffers in the symbol schematic.



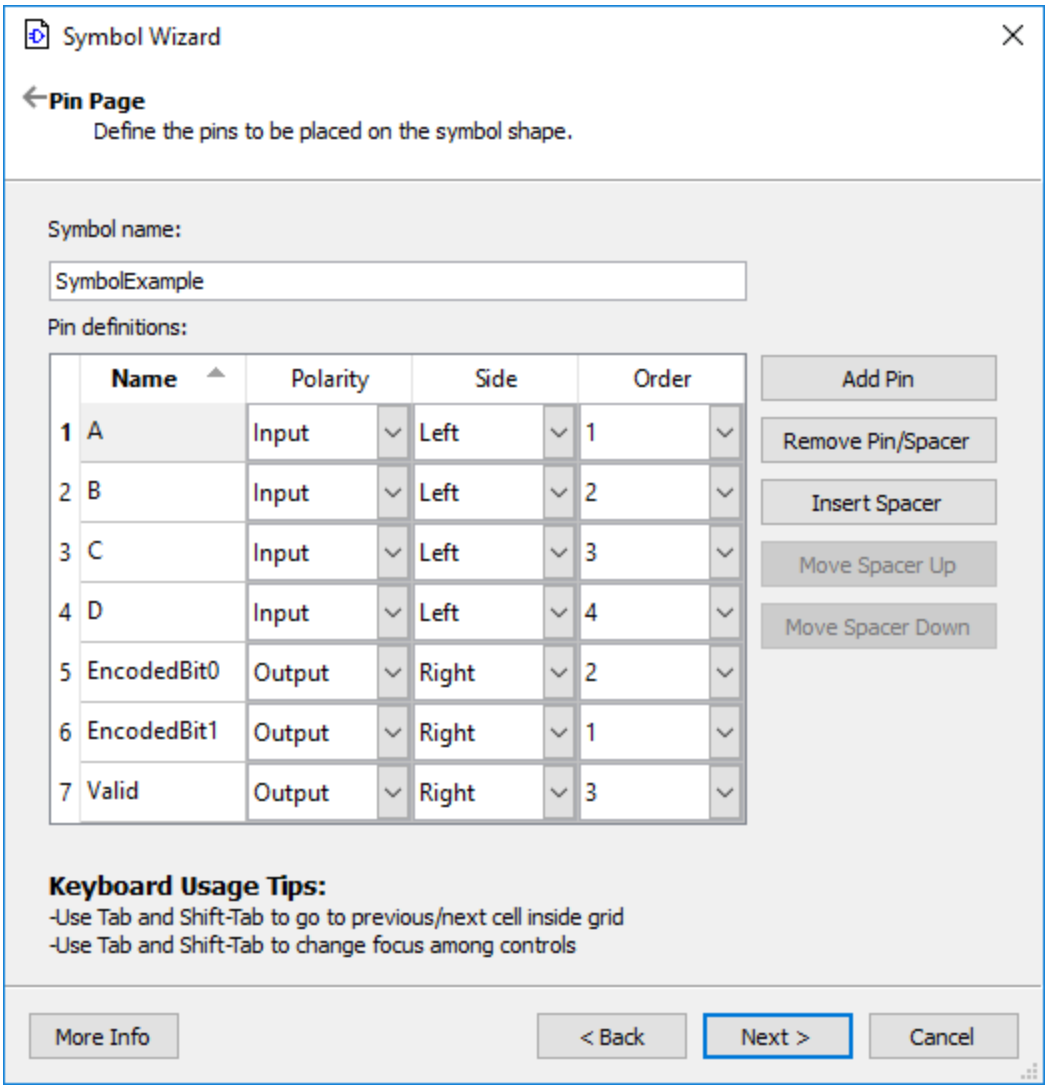
Create the circuit above (an encoder) and add output markers to each of the outputs. Once again, it is important to note that you should not use output buffers in the symbol schematic.

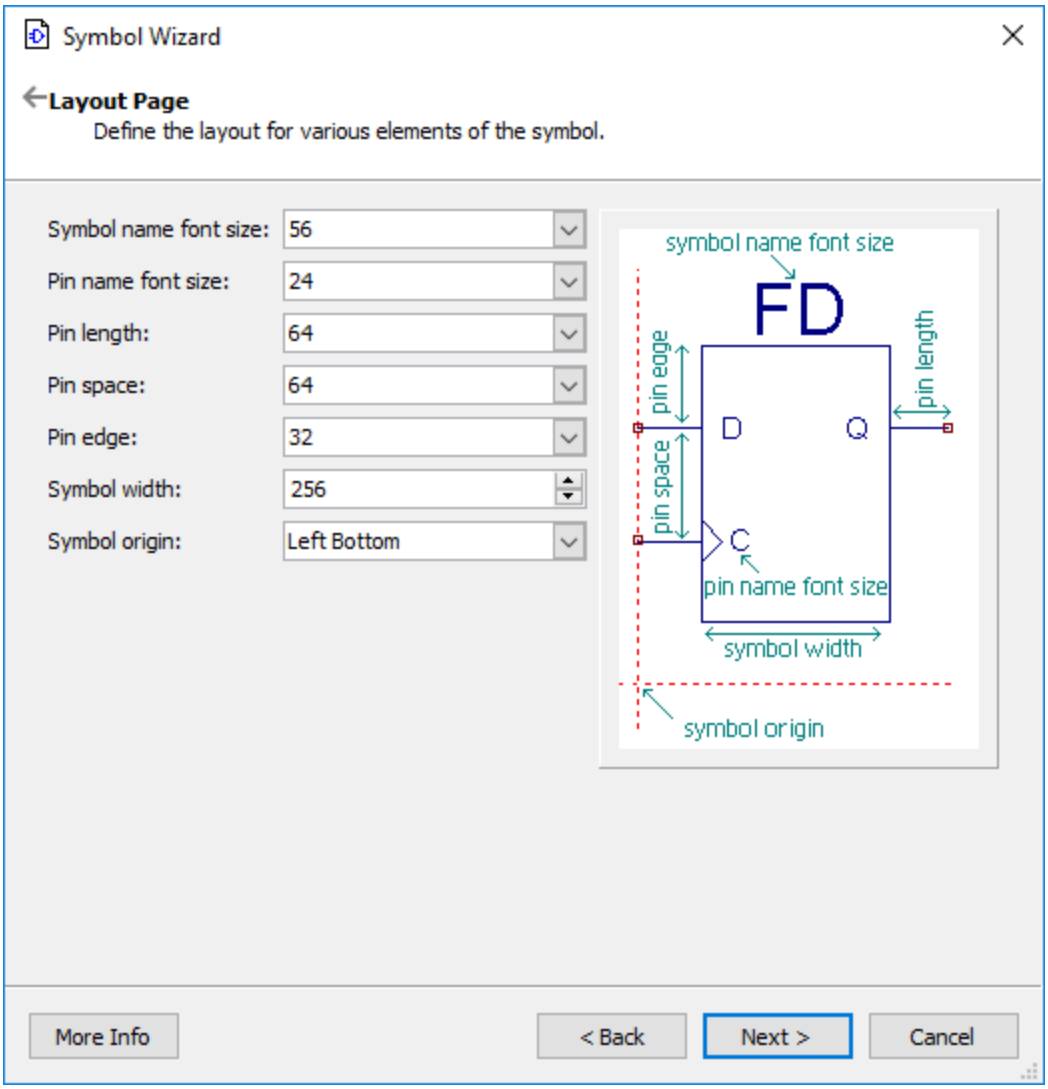


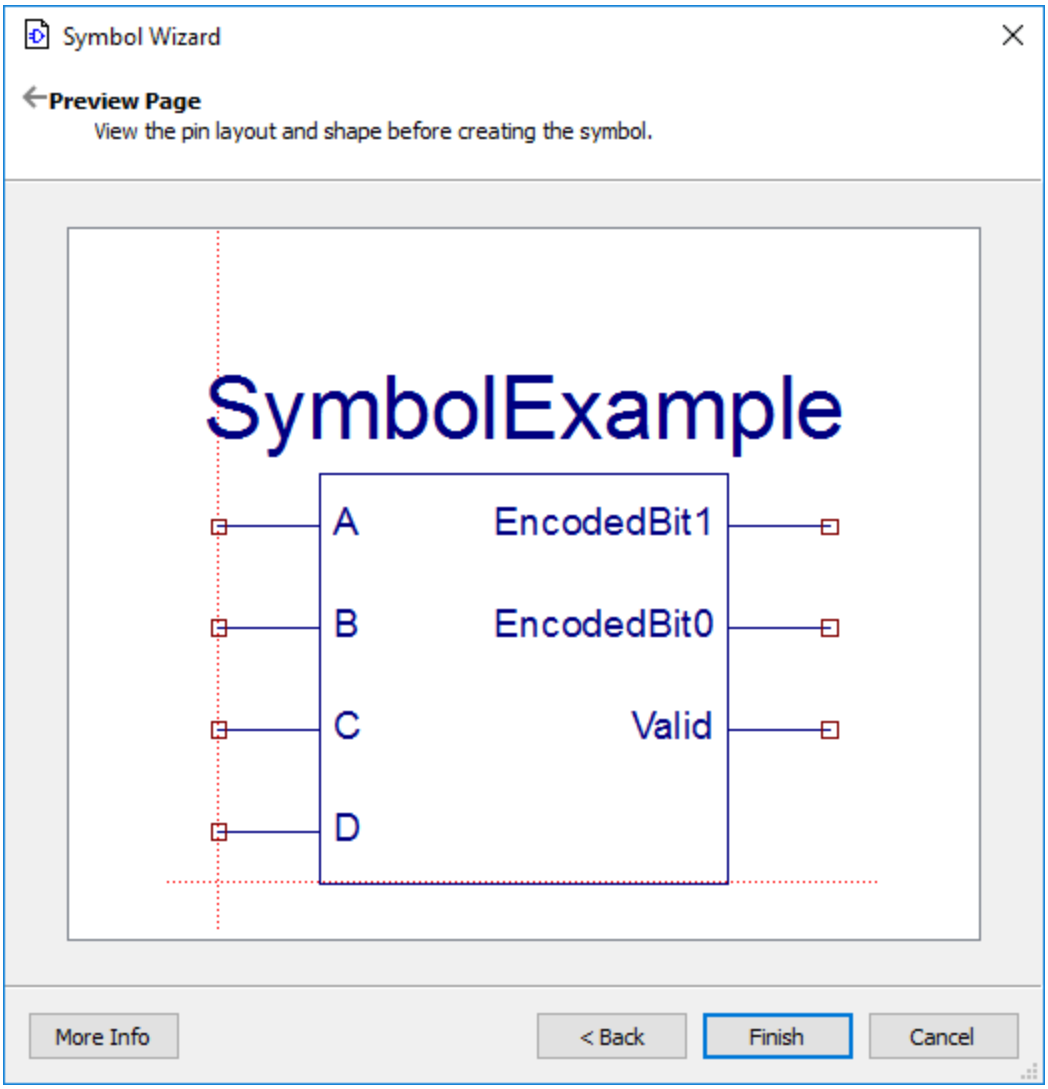
Now, time to create the symbol. Click on Tools on the top of the screen and go down to Symbol Wizard.



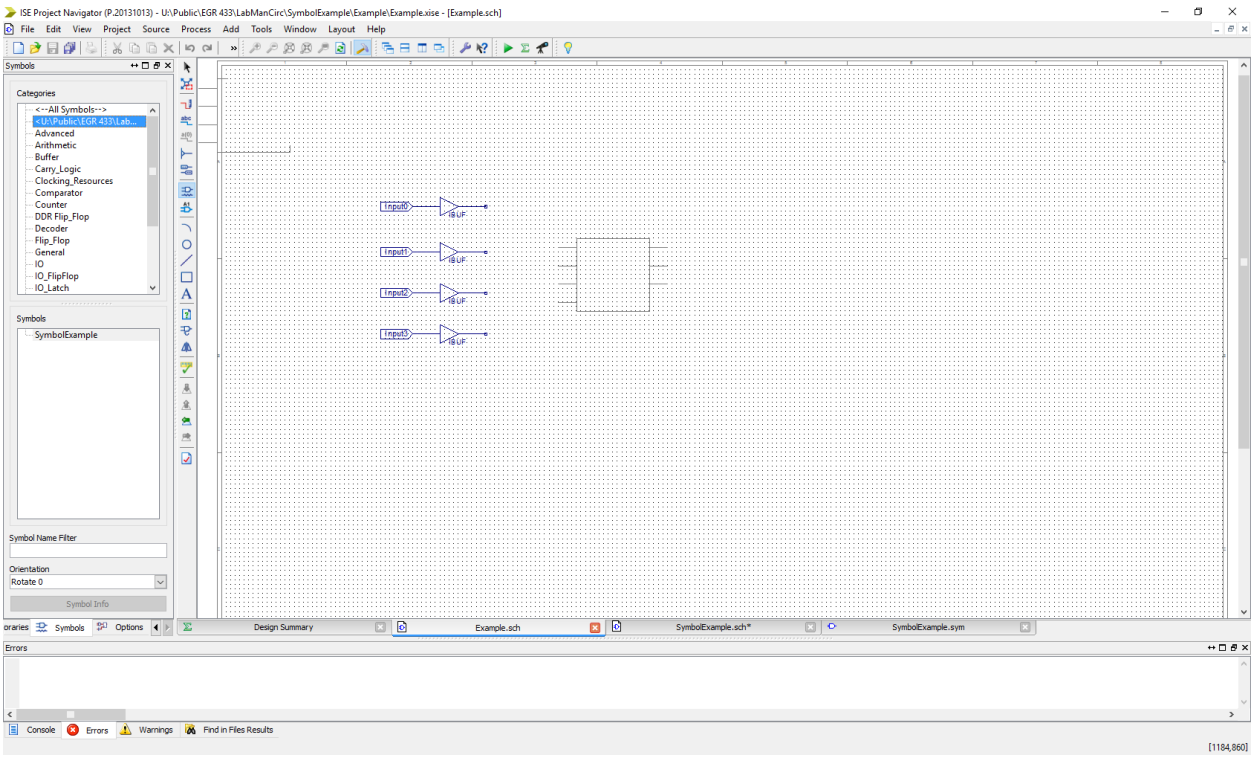
Select the schematic that is being used to create the symbol.


Symbol Wizard
Pin Page
Define the pins to be placed on the symbol shape.
Symbol name:
SymbolExample
Pin definitions:
Name
Polarity
Side
Order
1 A Input Left 1
2 B Input Left 2
3 C Input Left 3
4 D Input Left 4
5 EncodedBit0 Output Right 2
6 EncodedBit1 Output Right 1
7 Valid Output Right 3
Add Pin
Remove Pin/Spacer
Insert Spacer
Move Spacer Up
Move Spacer Down
Keyboard Usage Tips:
-Use Tab and Shift-Tab to go to previous/next cell inside grid
-Use Tab and Shift-Tab to change focus among controls
More Info
< Back
Next >
Cancel

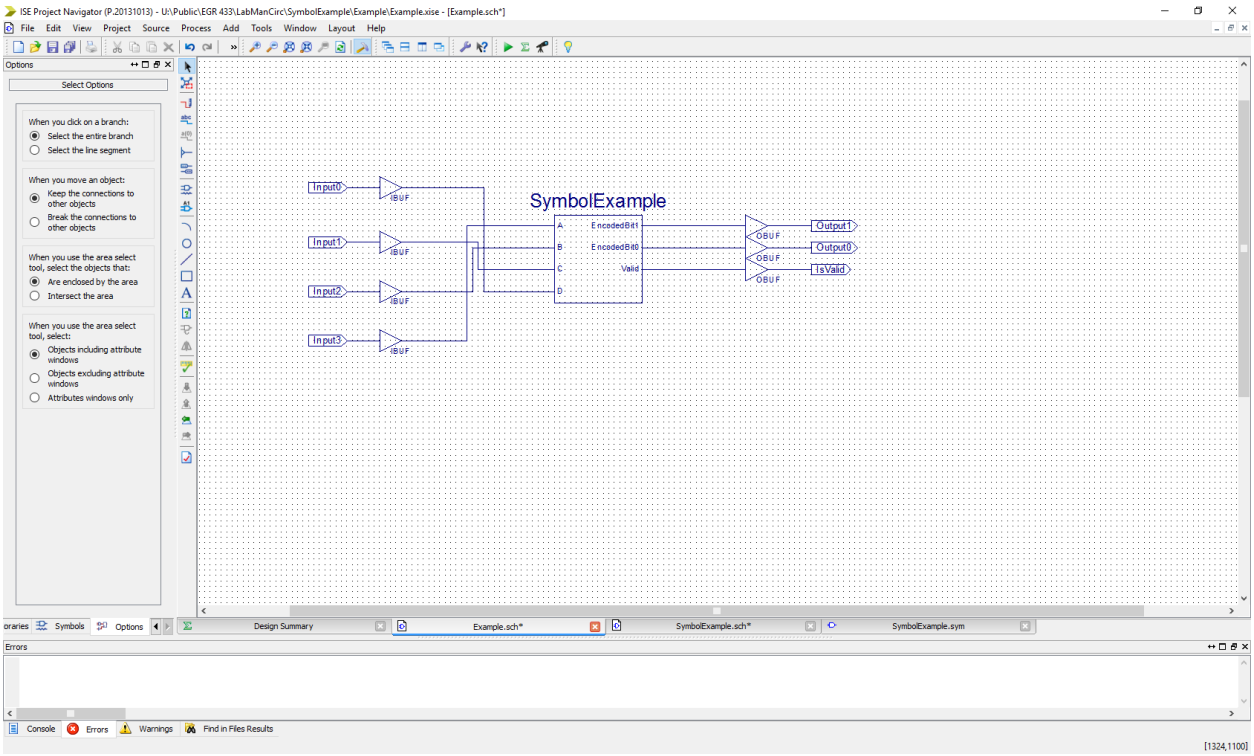

Symbol Wizard
Layout Page
Define the layout for various elements of the symbol.
Symbol name font size:
56
Pin name font size:
24
Pin length:
64
Pin space:
64
Pin edge:
32
Symbol width:
256
Symbol origin:
Left Bottom
symbol name font size
FD
pin edge
pin length
D
Q
pin space
C
pin name font size
symbol width
symbol origin
More Info
< Back
Next >
Cancel



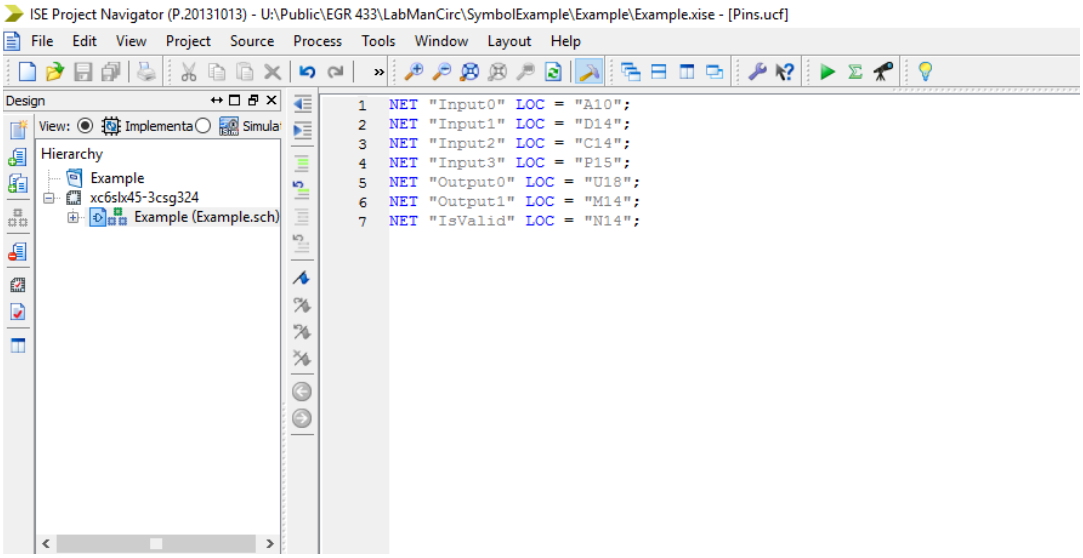
You will see these three screens. Make sure that the final symbol as shown in the third screen looks how you want it to look. If it doesn't go back and edit the previous two screens to achieve the desired appearance.



You can locate the created symbols under the file location that the schematic was in in the add symbols section of Xilinx. Add the symbol to the main schematic.

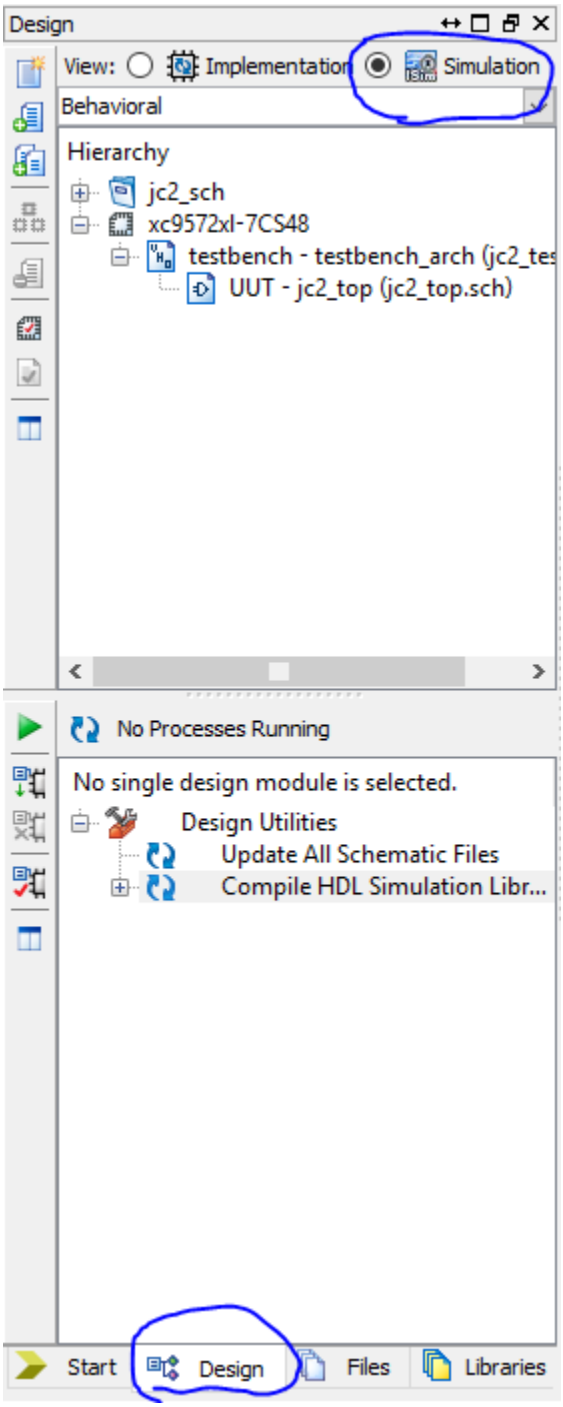


Wire the schematic.

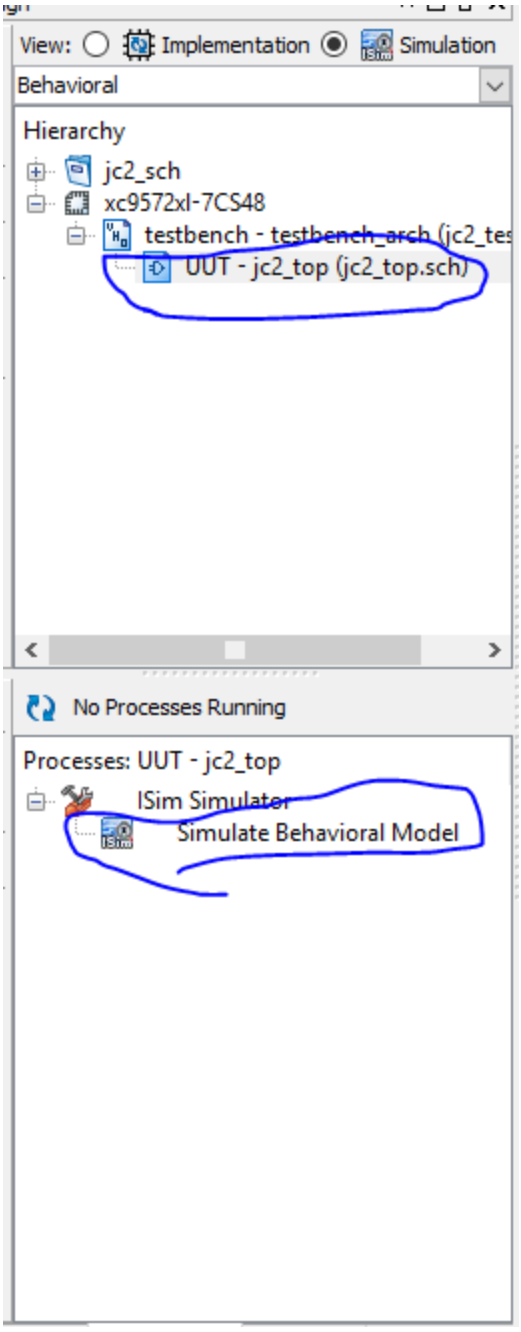


Add the implementation constraints file.

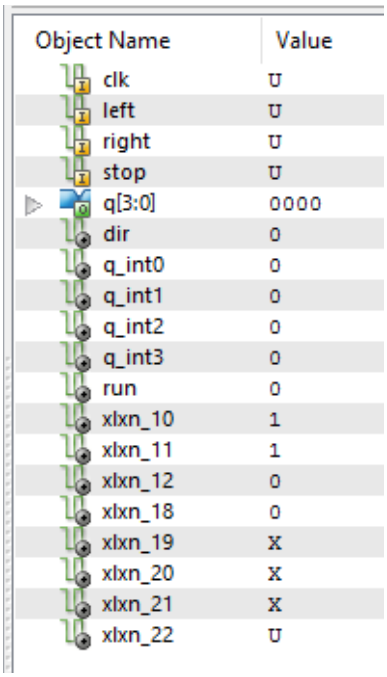
Debugging



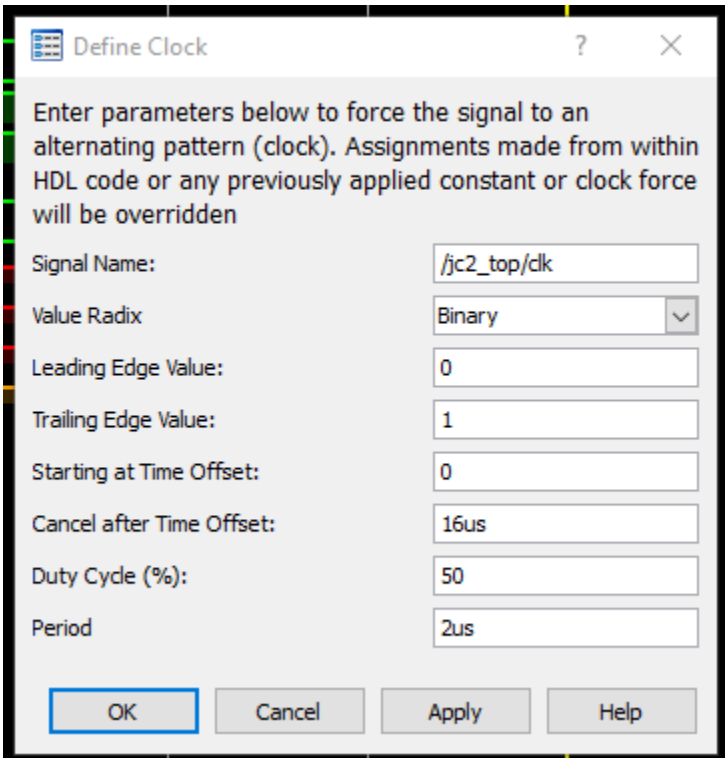
1. Click project you want to simulate, go to design switch to simulation.



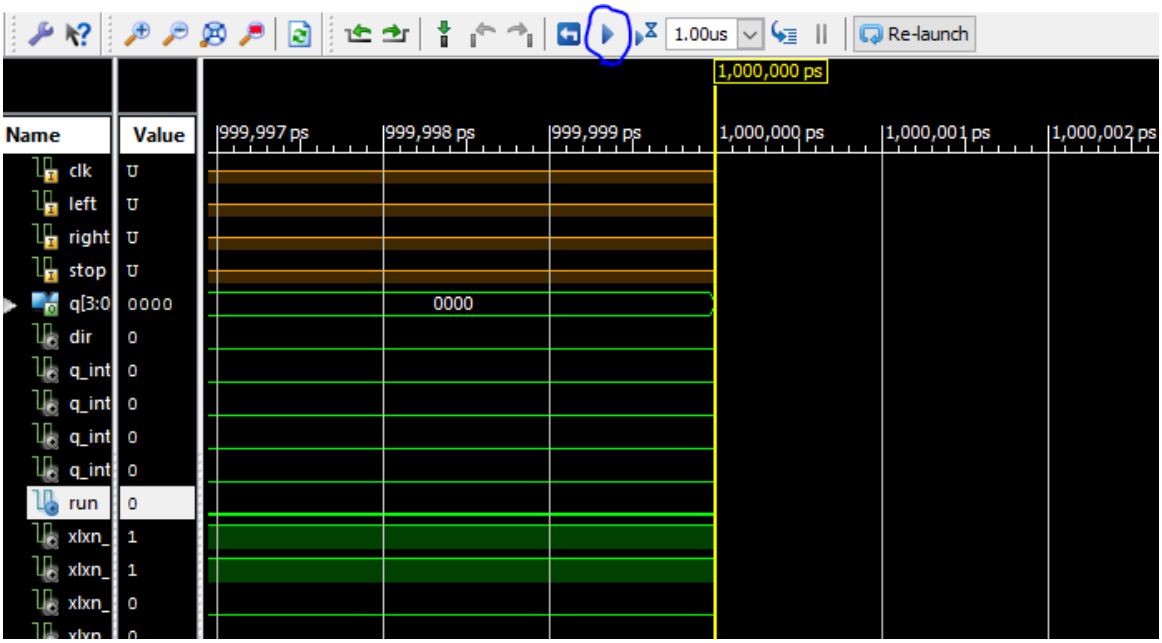
2. Click on schematic file, double click simulate behavioral model.



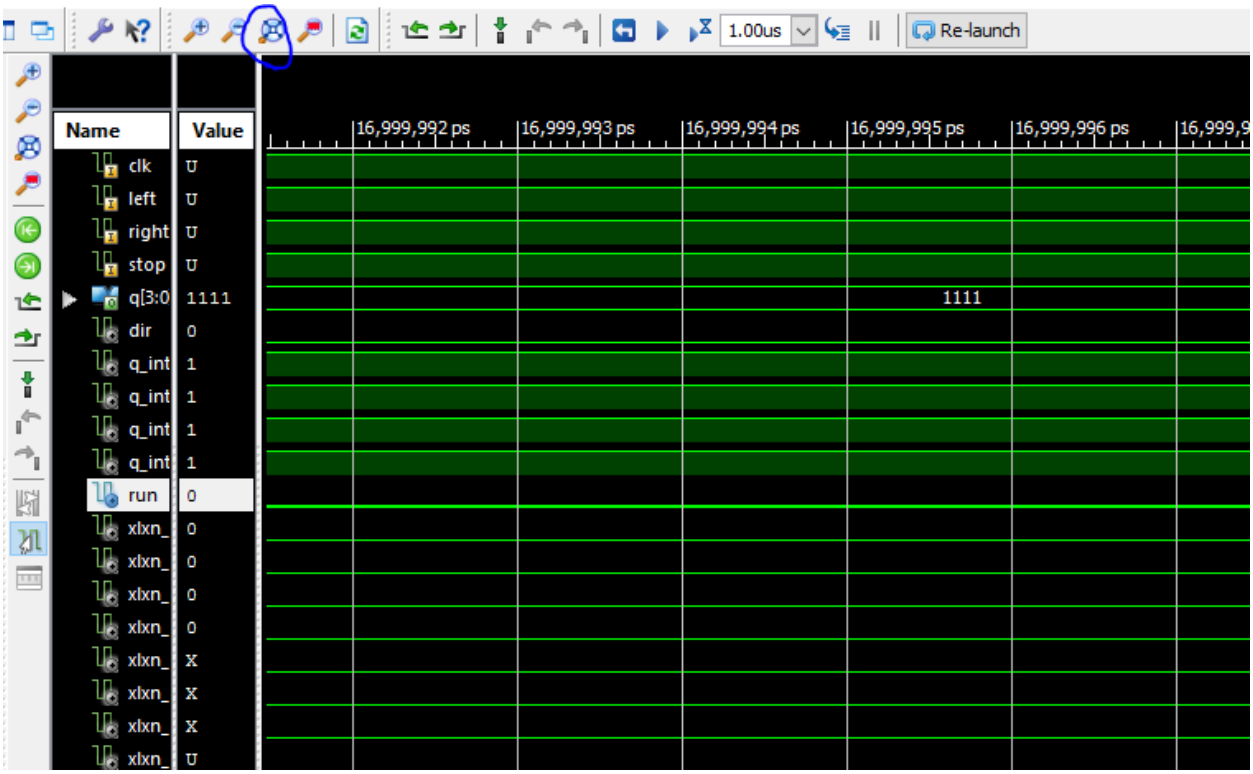
3. Right click on an input(the ones with a yellow square with an I) and choose force clock



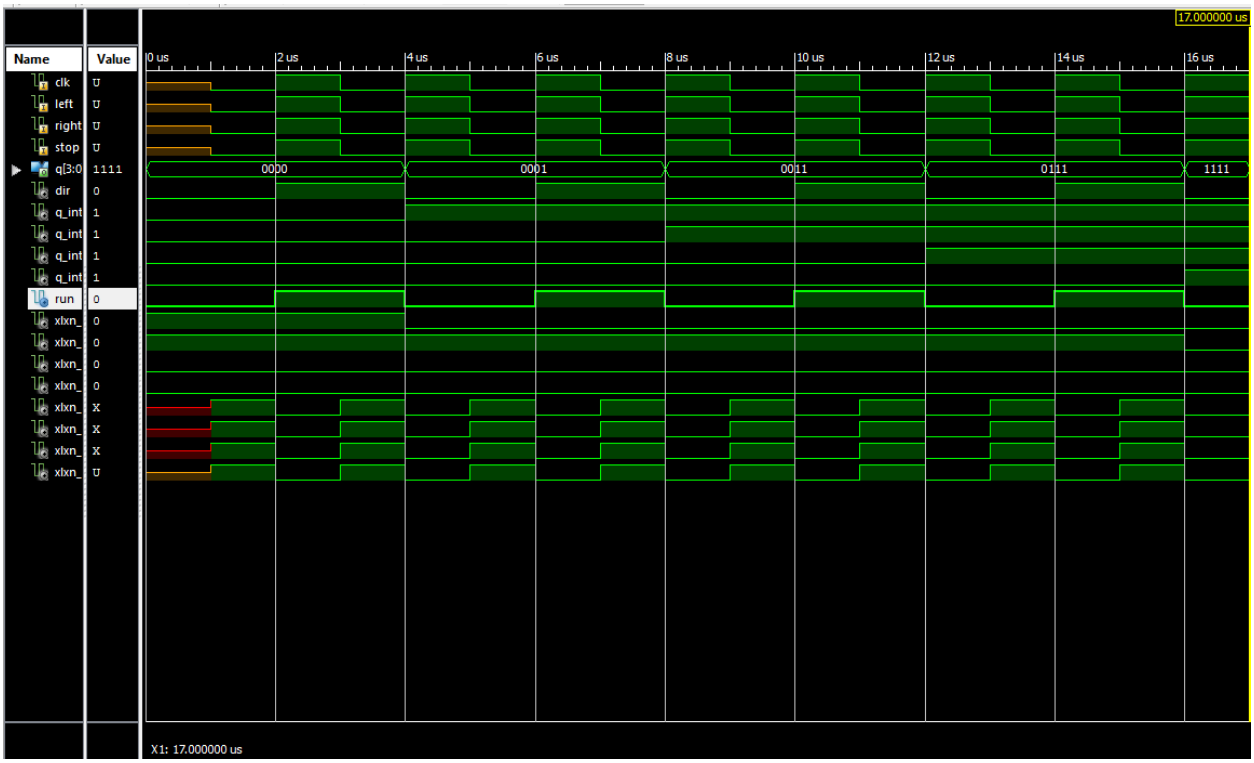
4. Force clock with chosen inputs and click apply
5. Do this for all inputs



6. When you are finished hit the run arrow



7. To see the waveforms select zoom to full view



8. After this you will be able to see the full waveforms and the characteristics of the circuit