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.

🗶 Xilinx License Configuration Manager	_			\times
Acquire a License Manage Licenses Borrow/Restore Licenses Return Licenses Internet Settings				
Select one of the following options				_ [
C Start Now! - 30 Day Trial (No Bitstream)				
Get Free Vivado/ISE WebPack License				
C Start 30 Day Evaluation				
C Get My Purchased License(s)				
C Locate Existing License(s)				
Description of the above selected option				
Get a free Vivado/ISE WebPack license and start using your Xliinx software. You will be taken to the Xliinx website where for Vivado/ISE WebPack. Once your license file is generated, the Yhanapa Xliinx Licenses' Tab will open to enable you to the license. For more information on Vivado/ISE WebPack, including supported devices and applications, please will twwn. Note: WebPalk is always enabled for WebPACK users. WebPalk jonces user and install preference when a bitstream is ge license. If a design is using a device contained in WebPACK and a WebPACK license is available, the WebPACK license will this, please see Answer Record 34746.	configure your .xilinx.com. enerated using t	system the We	bPACK	
About	Next		Close	

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

Name:	Lab3	
ocation:	U:\Private\TA NAthan\Lab3	
Working Directory:	U:\Private\TA NAthan\Lab3	
Description:		
elect the type of t	a loud as was for the project	
elect the type of to	p-level source for the project	

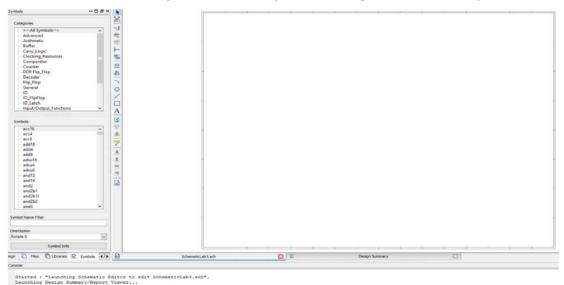
- c) Choose a name, and make sure to save it in a location that is not read-only (your public or private folder will work). Change the Top-level source type to "Schematic". Press "Next"
- d) Select "Spartan" in the "Family" box, "XC6SLX45" in the device box and "CSG324" in the "Package" box

elect the device and design flow for the p	roject	
Property Name	Value	
Evaluation Development Board	None Specified	`
Product Category	All	•
Family	Spartan6	•
Device	XC6SLX45	•
Package	CSG324	`
Speed	-3	•
Top-Level Source Type	Schematic	
Synthesis Tool	XST (VHDL/Verilog)	•
Simulator	ISim (VHDL/Verilog)	•
Preferred Language	Verilog	•
Property Specification in Project File	Store all values	•
Manual Compile Order		
VHDL Source Analysis Standard	VHDL-93	•
Enable Message Filtering		
Enable Message Filtering		

- 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"

Hierarchy i / / / / / xc6slx45-3csg324	^
New Source Wizard Select Source Type Select source type, file name and its lo	×
 IP (CORE Generator & Architecture) Schematic User Document Verilog Module Verilog Test Fixture VHDL Module VHDL Library VHDL Package VHDL Test Bench Embedded Processor 	Wizard) File name: SchematicLab3 Location: U:\Private\TA NAthan\Lab3
More Info	Next > Cancel

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



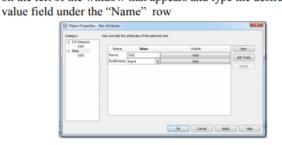
h) Like Logisim, the Symbol tool lets you add parts by connecting the respective leads together with the mouse or with the wire drawing tool. Common gates like the AND gate are located under the "Logic" category. To create an Input, you must first add an "IBUF" from the IO category. Then, click "I/O 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 1)

Rin Kalt Herr I	A D D X
iyebda	
Categories	
 All Symbolic 1 	
- Advanced	
- Anthermetry	
Buffer	
- Carry Legis	
Clecking, Resource	
Campandor	
Countar	
DDR Flip, Flep	
Decoder	
- Rip, Rep General	
Ceneral	
L N	
Symbols	
acc36	
and	
acch	
-105	
adauth	
alud	
ated	
and	
and the	
and2	
andbi	
Symbol Name Piller	
Orientation	
Retain 0	
Synbol	ids.

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

 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

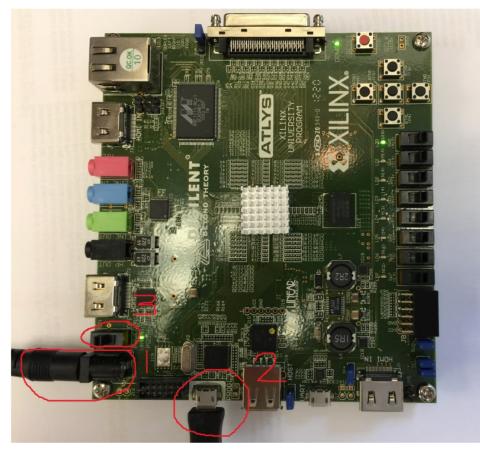


You created the constraint file

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:



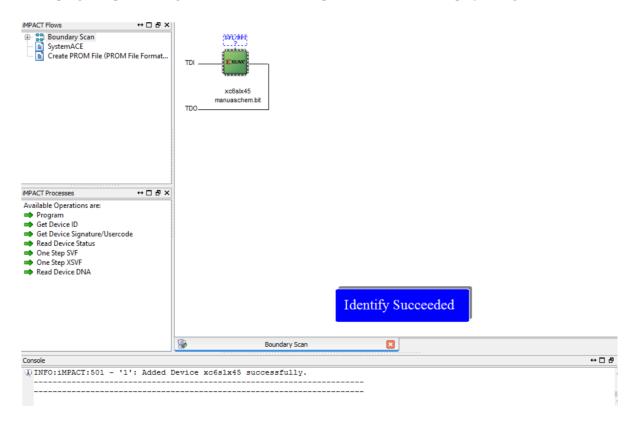
Plug in the power cord, then the usb cable from the board to the computer, and finally turn on the power switch.

- 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:

Processes: manuaschem

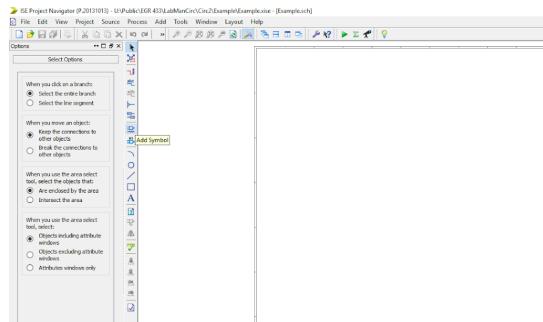


- e) The next step is to download the circuit into the FPGA. To accomplish this, press the Tools tab then click on "Impact", and press "OK" to the warning message.
- f) Double-click "Boundary Scan", then right-click and select "Initialize Chain". A message should pop-up to assign configuration files, 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"

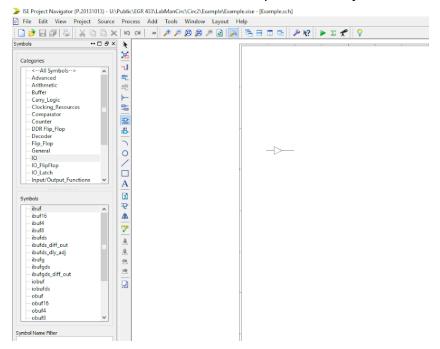


Example Circuits

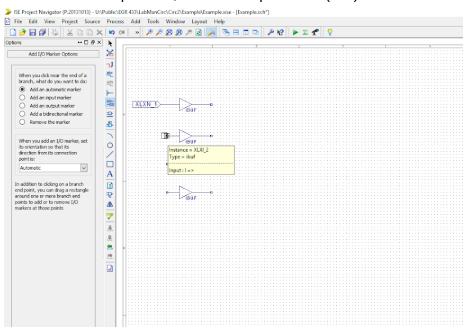
Let's start by creating a simple four person voting machine that states if option A won, option B won, or a tie occurred. Start by creating a new project and schematic as previously described.



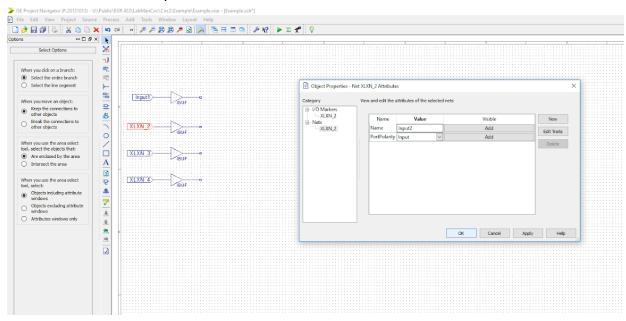
Click on the above icon to access the option to add symbols.



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



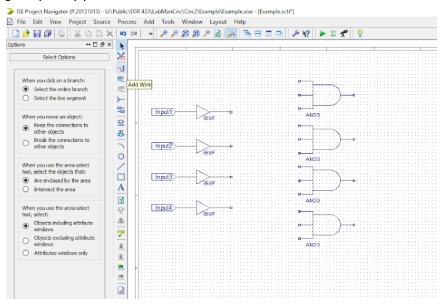
Click on the above icon to access the I/O markers. Put an input marker (or an automatic) on the left side of each of the input buffers.



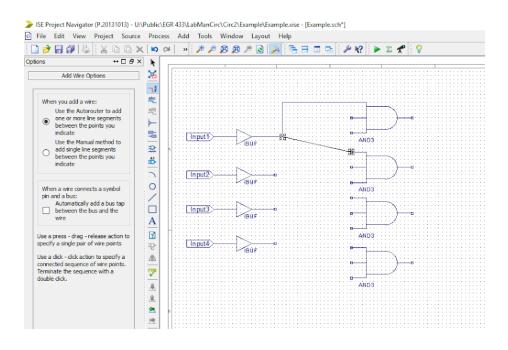
You can double click on the input markers to change their name. To do so, go to the XLIN_2 under Nets and type the desired name in the name field.

File Edit View Project Sour	ce Proces	is Add Tools	Window Layo	ut Help			
🗋 🤌 🖶 🥵 🐇 🖏 🖻	× 5 0	1 » 🏓 🏓	88/8		<i>₽</i> k?	Þ 🗵 📌 🗌	?
mbols ↔ □ &					13 13		
Categories	×						
Comparator							
Counter	abo						
- DDR Flip_Flop							
- Decoder	<u>a(0)</u>						
- Flip Flop	►						
- General			N		 		
- IO		Input1	·····		::		
IO_FlipFlop	2		IBUF		 		
		[]:::::::::::::::::::::::::::::::::::::			:::::::::::::::::::::::::::::::::::::::		
IO_Latch	₽						
Input/Output_Functions	-			11111111			
LUT	· · ·	In put2	IBME				
Latch	0				 		
Logic	1						
Memory	1 T		N				
Mux		input3>					
Shift_Register	A		IBUF				
Symbols	2	[]:::::::::::::::	N				
	÷	Input4>	\rightarrow				
and12 ^			IBUF				
anuro	Δ.						
- and2	-				 		
and2b1	7						
and2b1I	A						
and2b2							
and3	<u>&</u>						
and3b1							
- and3b2							
- and3b3	1.1						
and4							
- and4b1							
- and4b2							
and4b3		:::::::::::					
and4b4							
and5 V							

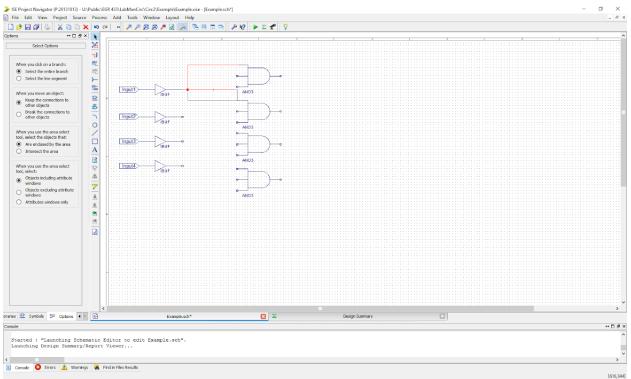
Next, go back to the icon to add symbols, but this time go to logic. Click on the three-input AND gate (and3) put four of them on the schematic.



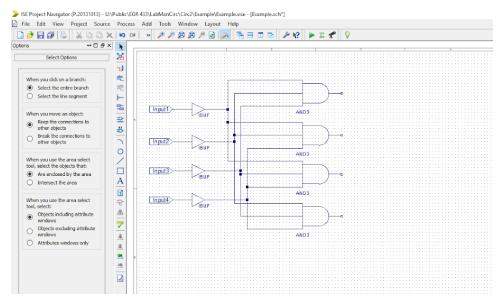
Click on this icon to start adding wires to the circuit.



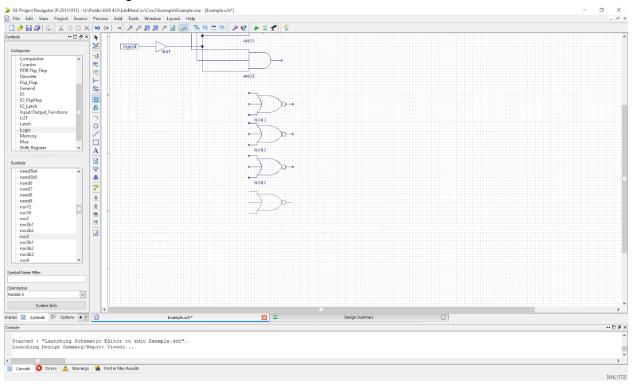
To add a wire to the schematic, either click on two different points or drag the wire from one point to another. These points can either be a blank space, one of the tiny blocks connected to each symbol, or another wire.



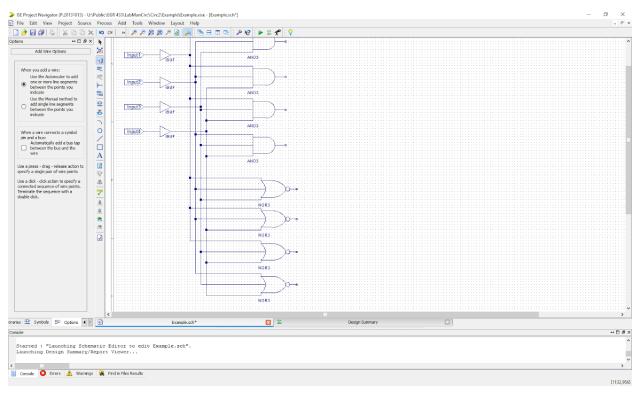
If you do not like how Xilinx automatically placed the wire, you can change its position by clicking on the wire and dragging it.



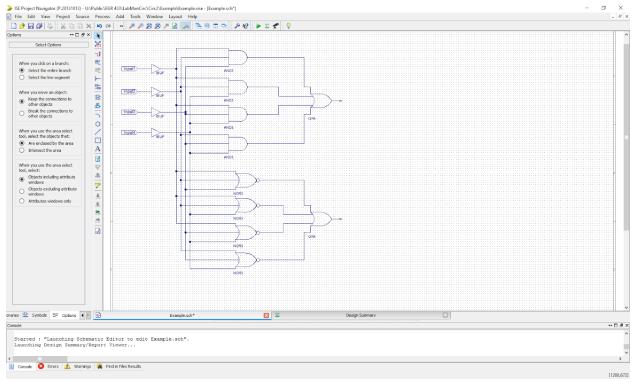
Wire the four AND gates as shown above.



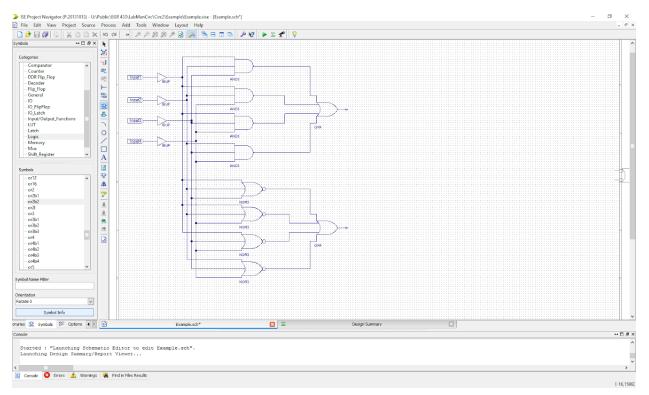
Next, place four three-input NOR gates (nor3).



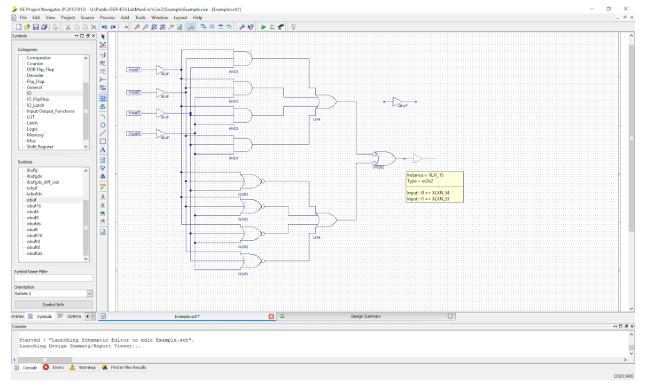
Wire them the same as the four AND gates.



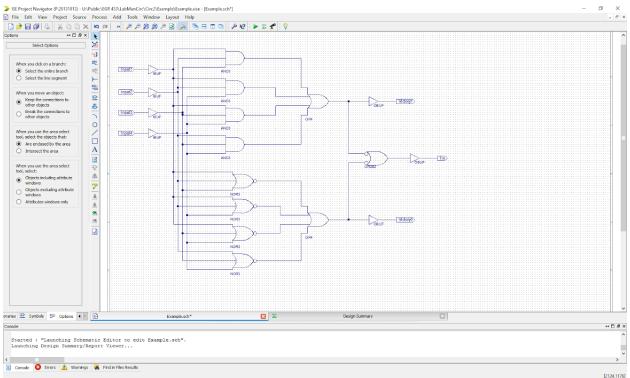
Add two four-input OR gates (or4) and wire each of the AND gates to one and each of the NOR gates to the other.



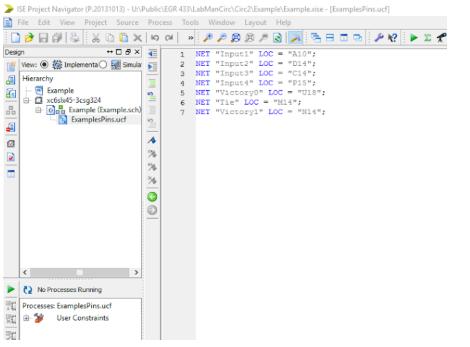
Next, we are going to add a symbol that tells us when it is a tie (neither of the four-input OR gates have a value of 1). Instead of using a two-input NOR gate, let's use the and2b2 gate. To figure out what it is, click on the name of the symbol and then click Symbol Info. Place it on the schematic and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



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

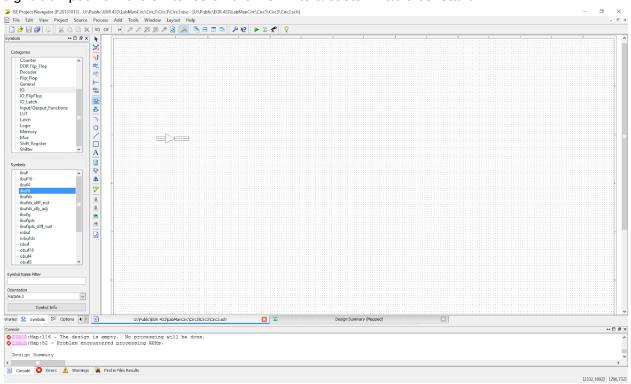


Connect three output markers received from the same location as the input markers on the right side of each output buffer. This is what the completed circuit should look like.

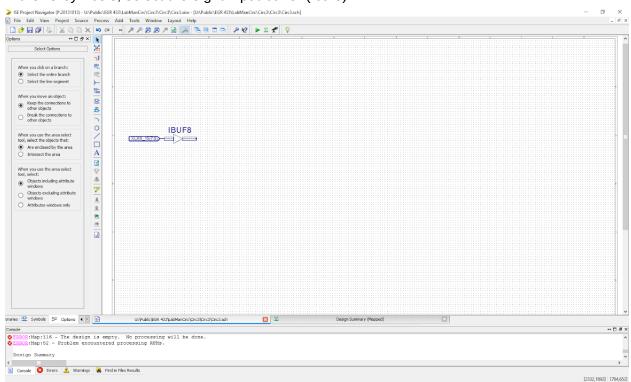


The final step is to create the implementation constraints file and state where the inputs and outputs are to be located on the FGPA. You are now done with the first example circuits.

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.



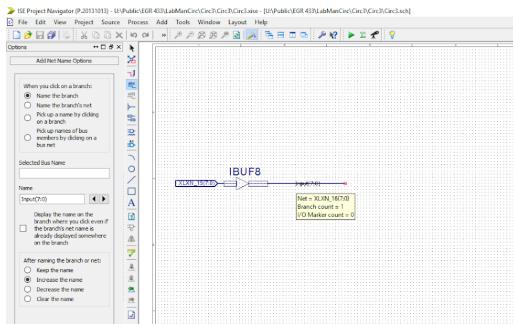
In the IO symbols, select the eight-input buffer (ibuf8).



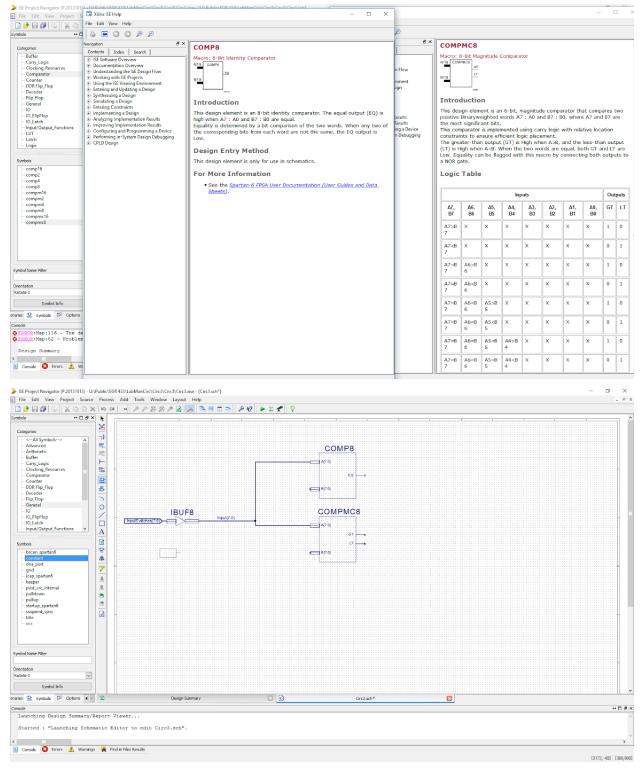
Connect an input marker to it.

	Add Tools Window Layout Help	<i>▶</i> k? ► Σ 🛠 💡				
+ C 8 × k			4 1	8 1	2	
Select Options 🔀						
you dick on a branch: Select the entire branch						
Select the line segment						
you move an object:						
eep the connections to						
reak the connections to ther objects						
O	IBUF8					
you use the area select						
re enclosed by the area A	Net = XLX					
ntersect the area	Branch co VO Marke	er count = 0				
you use the area select						
instantia inclusion attribute						
Andows Abjects excluding attribute excluding a						
Attributes windows only						
<u>A</u>						
*						
Symbols \$2 Options 4 > 0	U: \Public\EGR 433\LabManGirc\Grc3\Girc3\Girc3.sch		Design Summary (Mapp	od)	×	
Symbols and Options	or young pak 455 patriancine (pines (Design Summary (Happ	eu)		
	pty. No processing will be done.					
Map:52 - Problem encounter	eu processing RPMS.					
Summary						

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.

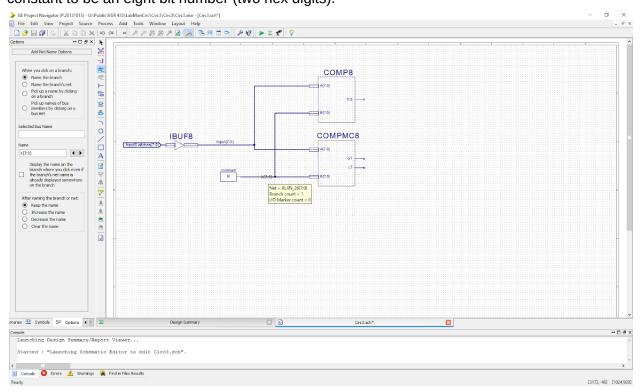


There are two different comparators in Xilinx, one that says whether or not they are equal and one that says whether the first input is greater than or less than the second. We will be using

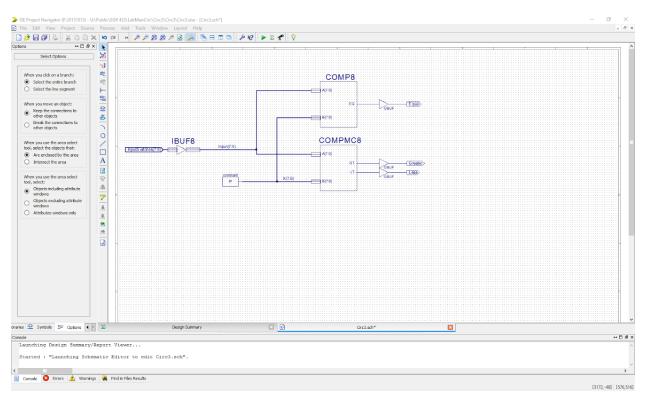
both.

Select Options	<u>×</u>									
hen you dick on a branch:	*			C	OMP8					
Select the entire branch Select the line segment	<u>비한</u>				Instance XLXI_12 Attribu	ter			× 1::::	
States on the augment	<u>له</u>			Object Properties -					^	
nen you move an object:	2			Category	View and edit the at	ttributes of the selec	ted instances			
Keep the connections to other objects				Instances XLXI_12						
Break the connections to other objects					Name	Value		Visible	New	
ound objects	ò				InstName SymbolName	XLXI_12			Edit Traits	
en you use the area select	~	IBUF8			CValue	3F			Delete	
ol, select the objects that: Are enclosed by the area		witches(7:0)	input(7:0)		LEVEL	CXILINX			Symbol Info	
Are enclosed by the area Intersect the area	Ā								Symportuno	
	1									
ien you use the area select sl, select:	9		constant							
Objects including attribute	4									
	7									
Objects excluding attribute windows	A									
Attributes windows only	<u>*</u>									
	<u>م</u>						ОК	Cancel App	ly Help	
	2							Carles rep		
👷 Symbols 🜮 Options 📢	> <u>Σ</u>	Design Summary	×	o	Circ3.sch*		2	1		
				·····				_		+

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.

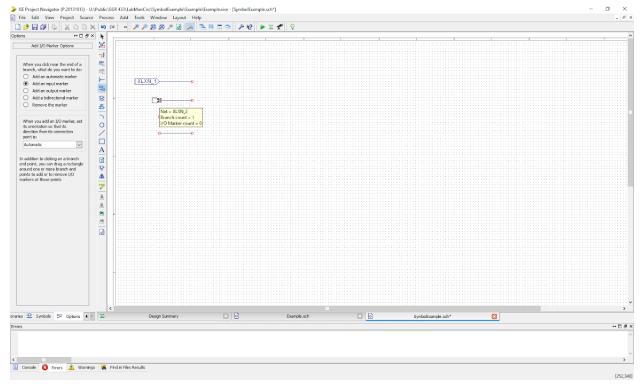
ISE Project Navigator (P.20131013) - U:\	Public\	EGR 433	3\LabMa	nCirc\Circ3\Ci	c3\Circ3.xis	e - [Pins.uc	f]							
File Edit View Project Source	Proc	ess To	ools W	indow Layo	ıt Help									
🗋 ờ 🖬 🕼 😓 🐰 🗅 🗅 🗙	ß	CII	» 🏓	P 👂 🙉 🗸	🤊 🗟 🖻	R 8		<i>⊁</i> k?	🕨 🗵 🕈	۲				
Design ↔ □ ♂ ×	4E	1	NET	"InputSwit	ches(0)	LOC =	"A10";				 	 	 	<i></i>
👔 View: 🖲 👹 Implementa 🔿 🎆 Simulai				"InputSwit										
Hierarchy				"InputSwit										
	≡	4		"InputSwit "InputSwit										
dil xc6slx45-3csq324	*2	5		"InputSwit										
🗄 🖨 💽 📲 Circ3 (Circ3.sch)	\equiv	7		"InputSwit										
Pins.ucf	10 	8	NET	"InputSwit	ches(7)	" LOC =								
<u>4</u>		9		"Greater"										
2	*	10 11		"Equal" LO "Less" LOO										
Image: Second secon	%	11	NE1	Dess. 100	010	1								
-	74													
ш	×4													
	G													
	6													
	\geq													
	ļ l													
No Processes Running														
No single design module is selected	é l													
Design Utilities Update All Schemati Compile HDL Simul View Compilatio View Compilatio														
🕅 🖨 🚺 Compile HDL Simul														
View Compilatio														
Regenerate All Cores														
Check All Core version														

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

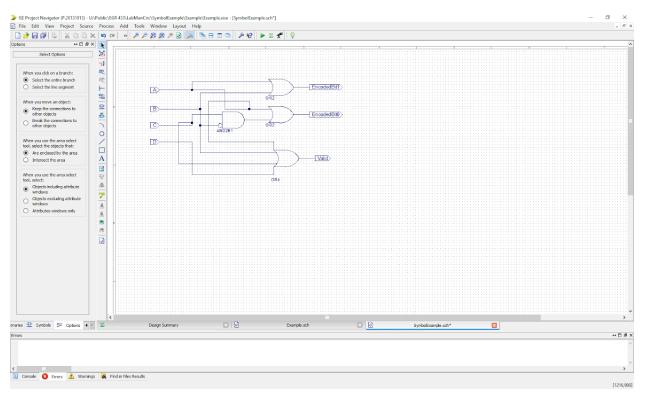
The last example circuit will go over how to create custom symbols. Start by creating two schematics, one will be the main schematic and the other the schematic to create the symbol.

a a X & ¶ ⊟ ⊧⊡÷		P & & P 🗟 🚬 🕞 E		· * : *	4	,			
Select Options	3				1	 			
	*								
n you dick on a branch: Select the entire branch	<u>=0</u>								
Select the entire branch Select the line segment									
Select the line segment									
n you move an object:	₩ ₩ ₩								
Keep the connections to	2								
other objects	4		N						
Break the connections to	- <u>-</u>	Input0	IBUE						
other objects									
	0		N						
n you use the area select select the objects that:		Input1	ISUF						
Are enclosed by the area									
Intersect the area	Α		N						
		input2	18UF						
you use the area select									
coloris	₽	[input3]							
Objects including attribute			₩isuf						
Million a	*								
Objects excluding attribute windows	2								
Attributes windows only	A								
	<u>*</u>								
	<u>ع</u>								
	2								
	1								
	J								
Symbols \$1 Options	E E	Design Summary	×	0	Example.sch	×	 	 	
						 			++

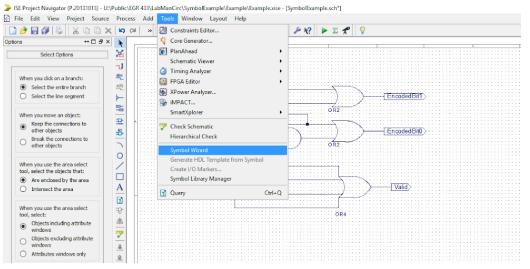
Let's start by creating the inputs for the main schematic.



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.



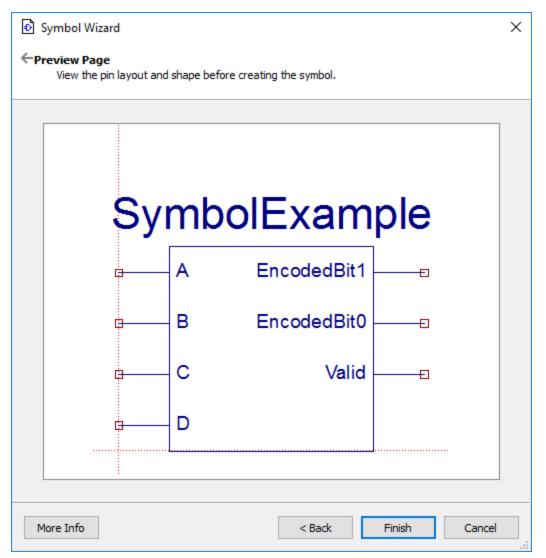
Now, time to create the symbol. Click on Tools on the top of the screen and go down to Symbol Wizard.

Pin name source		
O Specify manually		
Using schematic	SymbolExample	\sim
O Using symbol		\sim
Import symbo	l attributes	
Do not use refere	ince symbol	
Rectangle		
🔘 Square		
O Square	mbol	
		wse

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

	mbol name: mbolExample							
Pin	definitions:							
	Name 🔺	Polarity	_	Side	_	Order	_	Add Pin
1	А	Input	~	Left	\sim	1	~	Remove Pin/Spacer
2	В	Input	~	Left	~	2	~	Insert Spacer
3	с	Input	~	Left	~	3	~	Move Spacer Up
4	D	Input	\sim	Left	~	4	~	Move Spacer Down
5	EncodedBit0	Output	~	Right	~	2	~	
6	EncodedBit1	Output	~	Right	~	1	~	
7	Valid	Output	~	Right	~	3	~	

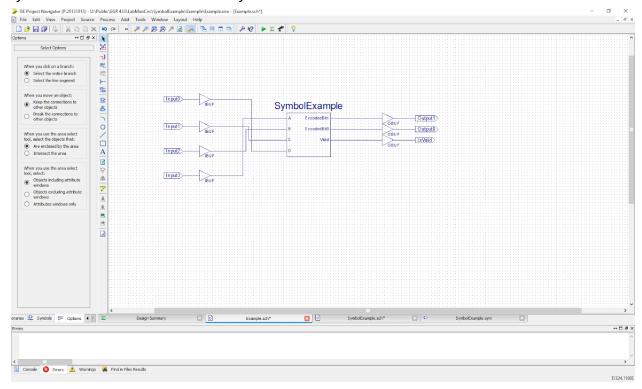
Symbol Wizard			Х
←Layout Page Define the layout fo	or various elements of the symbol.		
Symbol name font size: Pin name font size: Pin length: Pin space: Pin edge: Symbol width: Symbol origin:	56 ✓ 24 ✓ 64 ✓ 64 ✓ 32 ✓ 256 ♀ Left Bottom ✓	symbol name font size	
More Info	<	Back Next > Cancel	



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.

🗲 🔜 🖓 🖓 🖓	00X		* / / / / / /		1 B 🔑 K?	► Σ 📌 🤇	?					
	008×	*										
gories		7										
<all symbols=""> <u:\public\egr 433\l<="" td=""><td>^</td><td>abc</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></u:\public\egr></all>	^	abc										
Advanced		<u>a(0)</u>										
Arithmetic Buffer		⊨										
Carry_Logic												
Clocking_Resources Comparator		l 01 = 10 M										
Counter		#		input0>								
DDR Flip_Flop Decoder		5			P~18UF							
lip_Flop		0			N							
Seneral O		1 -		input1>	18UF							
O_FlipFlop												
O_Latch	~	A		input2>								
ks		2										
SymbolExample		÷		input3								
,		4			L-18UF							
		7										
		<u>.</u>										
		8										
		2										
		魚										
Name Filter												
vame niter		-										
tion												
0	\sim											
Symbol Info												
🔛 Symbols 🞾 C	ptions 🖣 🕨	E	Design Summary	× 0	Exam	ple.sch		SymbolExample.sch*	× •	SymbolExample.sym	×	
												+

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.



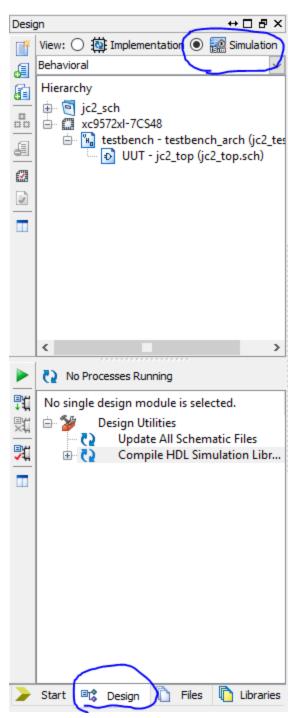
Wire the schematic.

ISE Project Navigator (P.20131013) - U:\Public\EGR 433\LabManCirc\SymbolExample\Example\Example.xise - [Pins.ucf]

File Edit View Project Source	Process Tools Window Layout Help
🗋 ờ 🖬 🕼 😓 🐰 🗅 🗅 🗙	. 📭 🖓 🔑 🔊 🖉 🖉 🛃 🔜 🔜 🚍 🖽 🖽 🖓 🤌 📡 🗶 💡
Design ↔ □ ♂ ×	1 NET "Input0" LOC = "A10";
View: 💿 🙀 Implementa 🔿 🏭 Simulai	
Hierarchy	3 NET "Input2" LOC = "C14"; 4 NET "Input3" LOC = "P15";
👔 🔤 Example	5 NET "Output0" LOC = "U18";
xc6slx45-3csg324 ····· ···· ···· ···· ··· ··· ··· ··· ···	6 NET "Output1" LOC = "M14"; 7 NET "IsValid" LOC = "N14";
	7 NET "IsValid" LOC = "N14";
0	A
	74
—	74
	<u>**</u>
< >	

Add the implementation constraints file.

Debugging



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

yn
View: 🔿 🔯 Implementation 💿 🔜 Simulation
Behavioral 🗸
Hierarchy jc2_sch xc9572xl-7CS48 testbench - testbench_arch (jc2_tes UUT - jc2_top (jc2_top.sch)
< >
No Processes Running
Processes: UUT - jc2_top ISim Simulater Simulate Behavioral Model

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

Object Name	Value
L clk	σ
🔓 left	υ
🔓 right	υ
1 stop	σ
> 📷 q[3:0]	0000
l 🔓 dir	0
l q_int0	0
l q_int1	0
l q_int2	0
l q_int3	0
l <mark>a</mark> run	0
l xlxn_10	1
l xlxn_11	1
lo xlxn_12	0
l xixn_18	0
lig xixn_19	x
lig xlxn_20	x
l xlxn_21	Х
la xixn_22	σ
1	

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

🧱 Define Clock	? ×
Enter parameters below to force t alternating pattern (clock). Assign HDL code or any previously applied will be overridden	ments made from within
Signal Name:	/jc2_top/clk
Value Radix	Binary 🗸
Leading Edge Value:	0
Trailing Edge Value:	1
Starting at Time Offset:	0
Cancel after Time Offset:	16us
Duty Cycle (%):	50
Period	2us
OK Cancel	Apply Help

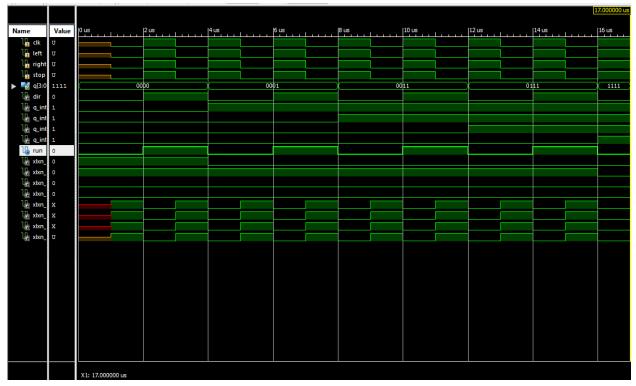
- 4. Force clock with chosen inputs and click apply
- 5. Do this for all inputs

1							
<i>₽</i> №?	1 1	🔊 🥕 🗟 🗠 😒	± † ^ 1			Re-launch	
				-	1,000,000 ps		
Name	Value	999,997 ps	999,998 ps	999,999 ps	1,000,000 ps	1,000,001 ps	1,000,002 ps
🗓 cik	υ						
្រា left	υ						
🔓 right	υ						
🔓 stop	υ						
🕨 📑 d[3:0	0000		0000				
Ug dir	0						
🗓 q_int	0						
🏹 q_int	0						
🏹 q_int	0						
🎼 q_int	0						
퉪 run	0						
🎼 xlxn_	1						
🎼 xlxn_	1						
🗓 xlxn_	0						
] 🖡 vivo	0						

6. When you are finished hit the run arrow

2	P 12	/# <i>F</i> (8 🏓 🗌	3 🗠 🛨 🕇	it 🤺 🚺 🕨	🔉 1.00us 🗸 🐓	🗔 Re-laund	h	
Ð			\sim						
) (2)	Name	Value	1	16,999,992 ps	16,999,993 ps	16,999,994 ps	16,999,995 ps	16,999,996 ps	16,99
•	lla cik	υ							
	l <mark>n</mark> left	U							
€ ∋	ြာ right ြြာ stop								
₩ •	stop g[3:0]						1111		
•r	U _e dir	0							
	ି q_int	1							
1	ີint								
<u>م</u>	le q_int								
1	🍓 q_int	1							
51		0							
ม	u_ xlxn_	0							
	U xlxn_	0							
	1 xlxn_	0							
	1 xlxn_	х							
	Ua xlxn_ Ua xlxn_	x x	_						
	$\frac{1}{16} x x x n_{-}$	ν υ							

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