# **Vivado Tutorial Using IP Integrator**

## Introduction

This tutorial guides you through the design flow using Xilinx Vivado software to create a simple digital circuit using Vivado IP Integrator (IPI). A typical design flow consists of creating a Vivado project, optionally setting a user-defined IP library settings, creating a block design using various IP, creating a HDL wrapper, creating and/or adding user constraint file(s), optionally running behavioral simulation, synthesizing the design, implementing the design, generating the bitstream, and finally verifying the functionality in the hardware by downloading the generated bitstream file. You will go through the typical design flow targeting the Artix-100t based Nexys4 or Artix-35t based Basys3 board.

# Objectives

After completing this tutorial, you will be able to:

- Create a Vivado project targeting a specific FPGA device located on the Nexys4 or Basys3 board
- Use the provided partially completed Xilinx Design Constraint (XDC) file to constrain some of the pin locations
- Add additional constraints using the Tcl scripting feature of Vivado
- Simulate the design using the XSim simulator
- Synthesize and implement the design
- Generate the bitstream
- Configure the FPGA using the generated bitstream and verify the functionality

## Procedure

This tutorial is broken into steps that consist of general overview statements providing information on the detailed instructions that follow. Follow these detailed instructions to progress through the tutorial.

# **Design Description**

The design consists of some inputs directly connected to the corresponding output LEDs. Other inputs are logically operated on before the results are output on the remaining LEDs as shown in **Figure 1**.



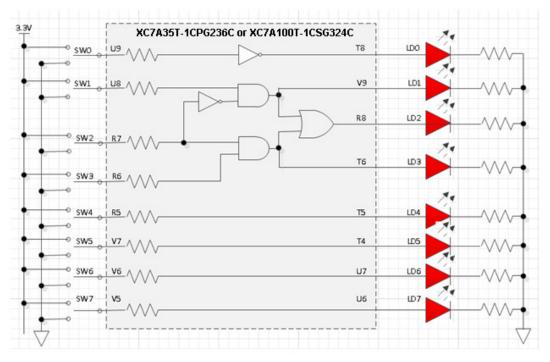


Figure 1. Completed Design

## General Flow for this tutorial

- Create a Vivado project and set IP library setting
- Create a block design
- Create a HDL wrapper and add the provided constraint file
- Simulate the design using XSim simulator
- Synthesize the design
- Implement the design
- Perform the timing simulation
- Verify the functionality in hardware using the target board

# Create a Vivado Project using IDE

## Step 1

1-1. Launch Vivado and create a project targeting either the *Nexys4* or the *Basys3* and using the Verilog HDL. Use the provided Verilog source files and *tutorial\_nexys4.xdc* (for Nexys4) file or *tutorial\_basys3.xdc* (for Basys3) file from the <2014\_2\_artix7\_sources> directory.

References to <2014\_2\_artix7\_labs> means c:\xup\digital\2014\_2\_artix7\_labs and <2014\_2\_artix7\_sources> means c:\xup\digital\2014\_2\_artix7\_sources directories.

- 1-1-1. Open Vivado by selecting Start > All Programs > Xilinx Design Tools > Vivado 2014.2 > Vivado 2014.2
- 1-1-2. Click Create New Project to start the wizard. You will see Create A New Vivado Project dialog box. Click Next.



- 1-1-3. Click the Browse button of the *Project location* field of the **New Project** form, browse to <2014\_2\_artix7\_labs>, and click Select.
- 1-1-4. Enter tutorial in the *Project name* field. Make sure that the *Create Project Subdirectory* box is checked. Click **Next**.

🚴 New Project	
Project Name Enter a name for your project and specify a directory where the project data files will be stored	
Project name: tutorial	3
Project location: C:/xup/digital/2014_2_artix7_labs	
Create project subdirectory	
Project will be created at: C://digital/2014_2_artix7_labs/tutorial	
	_
< <u>Back</u> <u>N</u> ext > <u>Finish</u> Cancel	

Figure 2. Project Name and Location entry

- 1-1-5. Select **RTL Project** option in the *Project Type* form and click **Next**.
- 1-1-6. Select Verilog as the Target language and Simulator language in the Add Sources form.
- 1-1-7. Click Next.
- 1-1-8. Click Next to get to the Add Constraints form.
- 1-1-9. Select constraints file entries, if displayed, and use 'X' button on the right to remove it.

This Xilinx Design Constraints file assigns the physical IO locations on FPGA to the switches and LEDs located on the board. This information can be obtained either through a board's schematic or board's user guide. We will add the file later.

1-1-10. In the *Default Part* form, using the **Parts** option and various drop-down fields of the **Filter** section, select the **XC7A100TCSG324-1** part (for Nexys4) or the **XC7A35TCPG236-1** part (for Basys3). Click **Next**.



efault Part									
Choose a default Xilinx part or board for your project. This can be changed later.									
Specify	Filter								
Parts	Product cat	tegory All		Ŧ	Package cs	a324	-		
Boards		amily Artix	-7	▼ Spe	ed grade -1	-			
U Dourus	S <u>u</u> b-F	amily Artix	-7		mp grade C				
				Reset All Filte					
				Keset All Fille	rs				
Search: Q									
<u>s</u> carem	· ·								
		I/O Pin Count	Available IOBs	LUT Elements	FlipFlops	Block RAMs	DSPs		
Part					FlipFlops 41600		DSPs 90		
Part xc7a35tc xc7a50tc	sg324-1 sg324-1	Count	IOBs	Elements	1	RAMs			
Part xc7a35tc xc7a50tc xc7a75tc	sg324-1 sg324-1 sg324-1	Count 324 324 324 324	IOBs 210 210 210	Elements 20800 32600 47200	41600 65200 94400	RAMs 50 75 105	90 120 180		
Part xc7a35tc xc7a50tc	sg324-1 sg324-1 sg324-1	Count 324 324	IOBs 210 210	Elements 20800 32600	41600 65200	RAMs 50 75	90 120		
Part xc7a35tc xc7a50tc xc7a75tc	sg324-1 sg324-1 sg324-1	Count 324 324 324 324	IOBs 210 210 210	Elements 20800 32600 47200	41600 65200 94400	RAMs 50 75 105	90 120 180		

### Figure 3. Part selection for Nexys4

🚴 New Projec	t						X		
Default Part									
Choose a default Xilinx part or board for your project. This can be changed later.									
Specify Fi	lter								
Parts Pr	odu <u>c</u> t cate	egory All		•	<u>P</u> ackage c	pg236	-		
Boards	_	amily Artix-7	_		ee <u>d</u> grade -:				
	S <u>u</u> b-F	amily Artix-	7	▼ 10	emp grade	;	<b>•</b>		
				Reset All Filt	ers				
<u>S</u> earch: Q <sub>▼</sub>									
Part		I/O Pin Count	Available IOBs	LUT Elements	FlipFlops	Block RAMs	DSPs		
🔷 xc7a35tcpg2		236	106	20800	41600	50	90		
xc7a50tcpg2	36-1	236	106	32600	65200	75	120		
			_						
•	111								
				< <u>B</u> ack	Next >	<u>F</u> inish	Cancel		
						л. —			

Figure 3. Part selection for Basys3

1-1-11. Click Finish to create the Vivado project.

Use the Windows Explorer and look at the <2014\_2\_artix7\_labs>\tutorial directory. You will find that the tutorial.cache directory and the tutorial.xpr (Vivado) project file.





Figure 4. Generated directory structure

## 1-2. Set IP repository path to point to the provided XUP IP library.

1-2-1. In the Flow Navigator window, click on Project Settings under the Project Manager group.

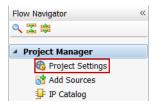


Figure 5. Invoking Project Settings to set IP repository path

- 1-2-2. In the Project Settings window, click on the IP.
- 1-2-3. Click on the Add Repositories button, browse to <2014\_2\_artix7\_sources> and select XUP\_LIB directory, and click Select.

The directory will be scanned and the available IP entries will be displayed.

Review Settings	ζ
IP	
Repository Manager Packager	
General Add directories to the list of repositories. You may then add additional IP to a selected repository. If an IP is disabled then a tool-tip will alert you to the reason	
Simulation IP Repositories	
c:/xup/digital/2014_2_artix7_sources/XUP_LIB (Project)	
Synthesis 7	
Implementation     Add Repository	
IP in Selected Repository	1
Bitstream XUP 2_to_1_mux (xilinx.com:xup:xup_2_to_1_mux:1.0)	
XUP n-bit wide 2_to_1_mux (xilinx.com:xup:xup_2_to_1_mux_vector:1.0)	
XUP 4_to_1_mux (xilinx.com:xup:xup_4_to_1_mux:1.0)           P         XUP a_to_1_mux (xilinx com:xup:xup_4_to_1_mux:1.0)	
IP         XUP n-bit wide 4_to_1_mux (xilinx.com:xup:xup_4_to_1_mux_vector:1.0)           XUP 2-input AND (xilinx.com:xup:xup_and2:1.0)	
P Add IP Ø Refresh Repository	
OK Cancel <u>Apply</u>	

Figure 6. Setting IP Repository

1-2-4. Click OK.



## Create a Block Design

## 2-1. Create a block design.

2-1-1. In the *Flow Navigator* window, click on **Create Block Design** under the IP Integrator block.

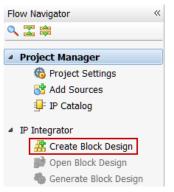


Figure 7. Invoking IP Integrator to create a block design

- **2-1-2.** Click **OK** to create a block design named *design\_1*.
- **2-1-3.** IP from the catalog can be added in different ways. Click on <u>Add IP</u> in the message at the top of the *Diagram* panel, or click the *Add IP icon* in the block diagram side bar, press Ctrl + I, or right-click anywhere in the Diagram workspace and select Add IP

ŀ	Diag	jran	1	×				
₹	🌲 s	yster	m					
•	()	This	s d	esign is emp	ty. To get	started,	<u>Add IP</u>	from the cata
•								
N P P P P P P P P P		R.	5	Properties		(	Ctrl+E	]
<b>₽</b>		>	ĸ	Delete		1	Delete	
ŵ		1	h	Сору		(	Ctrl+C	
\$		4	h	Paste		(	Ctrl+V	
•		•	ι,	Search		(	Ctrl+F	
₽		1	\$	Select All			Ctrl+A	-
•		1	<b>e</b>	Add IP		(	Ctrl+I	]
<b>.</b>		6	ò	IP Settings.				
-		G	2	Validate De	sign		F6	_
6				Create Hier	archy			
				Create Corr	nment			
Q[				Create Port		(	Ctrl+K	
				Create Inte	rface Port.	(	Ctrl+L	
		9	3	Regenerate	Layout			_
		1		Save as PD	F File			

Figure 8. Add IP to Block Diagram

**2-1-4.** Once the IP Catalog is open, type "inv" into the Search bar, find and double click on **XUP 1-input INV** entry, or click on the entry and hit the Enter key to add it to the design.



Search: Qrinv	(2 matches)
Name 1	VLNV
XUP 1-input INV	xilinx.com:xup:xup_inv:1.0
🞐 XUP n-bit wide INV	xilinx.com:xup:xup_inv_vector:1.0
Select and press ENTER or drag a	and drop, ESC to cancel

Figure 9. Add an inverter to the design

- 2-1-5. Similarly, another instance of an inverter.
- 2-1-6. Add two instances of 2-input AND gate and an instance of 2-input OR gate.

You can create an instance of already present IP, by clicking on it, pressing Ctrl key, and dragging the instance with the left mouse button.

2-1-7. Redraw the diagram, by clicking on the re-draw (<sup>™</sup>) button. At this stage the block diagram should look like shown below.

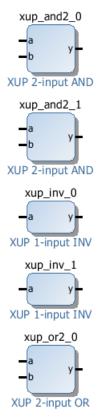


Figure 10. Added necessary instances



## 2-2. Complete the design.

**2-2-1.** Right-click on the **xup\_inv\_0** instance's input port and select **Make External**. Similarly, make the output port of the same instance and make it external.

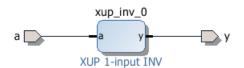


Figure 11. Making ports external

2-2-2. Click on the *a* port, and change the name to SW0 in its properties form.

External Por	rt Properties	-	×
D SW0			
Name: Direction: Net	SW0 Input Input		

General Properties

### Figure 12. Setting input port name to SW0

- **2-2-3.** Similarly, change the output port *y* to **LD0** (as per the diagram in Figure 1).
- **2-2-4.** Arrange OR2 instance such that it is close to the two instances of the AND2.
- **2-2-5.** Arrange the second instance of the inverter on the left of one of the AND2 gate.
- **2-2-6.** Using the left-button of the mouse, draw a connection between the outputs of the AND2 instances and the two input of the OR2.

When you move the mouse closer to a port, the cursor becomes drawing pencil icon. Click the left-button of the mouse and keeping the button pressed draw it towards the destination port. You make a connection this way.

**2-2-7.** Similarly, connect the output of the inverter to one input of one of the AND2 instances.

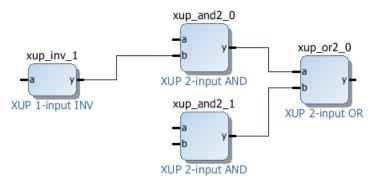


Figure 13. Connecting instances



This diagram is similar to the logic connected between SW1, SW2, SW3, and LD2.

- **2-2-8.** Make input ports of the **xup\_inv\_1**, *a* port of the **xup\_and2\_0**, and *b* port of the **xup\_and2\_1** instances external.
- 2-2-9. Similarly, make the output port of the xup\_or2\_0 instance external.

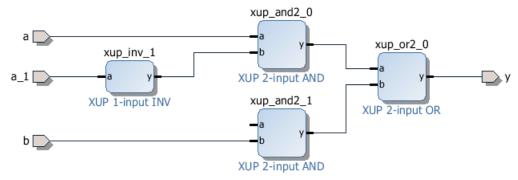


Figure 14. Making ports external

- **2-2-10.** Change the name of *a* to **SW1**, *a*\_1 to **SW2**, *b* to **SW3**, and *y* to **LD2**.
- **2-2-11.** Right-click somewhere on the canvas and select Create Port.

A Create Port form will appear.

2-2-12. Enter LD1 as the port name, using the drop-down button select the type as *output*, and click OK.

🚴 Create Port	×
Create port a	nd connect it to selected pins and ports
Port name:	
Direction:	Output 👻
<u>T</u> ype:	Other 👻
Create vector:	from 31 💭 to 0 💭
Frequency (MHz):	
Connect to mat	ching selected ports
	OK Cancel

Figure 15. Creating an output port

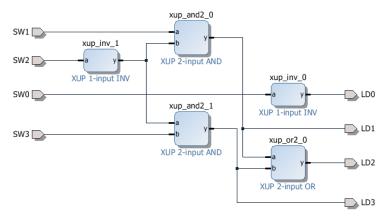
2-2-13. Similarly, create the output port naming it as LD3.

**2-2-14.** Connect the input port *a* of the **xup\_and2\_1** instance to output port of the instance **xup\_inv\_1**.



2-2-15. Connect the output port of the xup\_and2\_0 to LD1 and xup\_and2\_1 to LD3. Click on the redraw button.

The diagram will look similar to shown below.

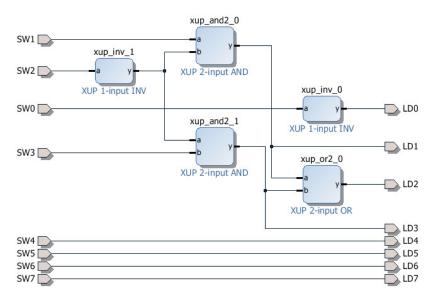


## Figure 16. Partially completed design

## 2-3. Complete the design including rest of the switches and LDs

- **2-3-1.** Right-click on the canvas and create an input port *SW4*.
- **2-3-2.** Similarly, create *SW5*, *SW6*, and *SW7* as input ports, and *LD4*, *LD5*, *LD6*, and *LD7* as output ports.
- 2-3-3. Using wiring tool, connect SW4 to LD4, SW5 to LD5, SW6 to LD6, and SW7 to LD7.
- 2-3-4. Click the re-draw button.

The design should look like as shown below.





#### 2-3-5. Select File > Save Block Design.



## **Create HDL Wrapper and Add a Constraint File**

## Step 3

### 3-1. Create a HDL wrapper and analyze the hierarchy

- **3-1-1.** In the *sources* view, Right Click on the block diagram file, **design\_1.bd**, and select **Create HDL Wrapper** to create the HDL wrapper file. When prompted, select **Let Vivado manage wrapper and auto-update**, click **OK**.
- **3-1-2.** In the *Sources* pane, expand the hierarchy.

Notice the design\_1\_wrapper file instantiates design\_1 which in turn instantiates the inverter twice, and 2 twice, and or2 once.

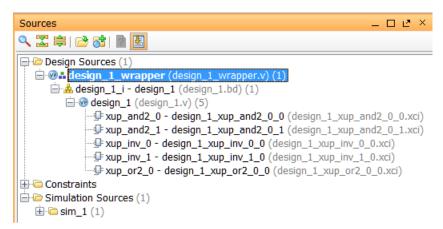


Figure 18. Hierarchical design

- **3-1-3.** Double-click the **design\_1\_wrapper.v** entry to open the file in text mode and observe the instantiation of the *design\_1* module.
- **3-1-4.** Double-click the **design\_1.v** entry to open the file in text mode and observe the instantiation of the lower-level modules.

# 3-2. Add tutorial\_nexys4.xdc (for Nexys4) or tutorial\_basys3 (for Basys3) constraints source and analyze the content.

- 3-2-1. Click on the Add Sources under the Project Manager group in the Flow Navigator window.
- **3-2-2.** Select the **Add or Create Constraints** option and click **Next**.
- 3-2-3. Click Add Files... and browse to <2014\_2\_artix7\_sources>\tutorial.
- 3-2-4. Select tutorial\_nexys4.xdc (for Nexys4) or tutorial\_basys3.xdc (for Basys3) and click OK.
- **3-2-5.** Click **Finish** to close the window and add the constraints file in the project under the Constraints group.



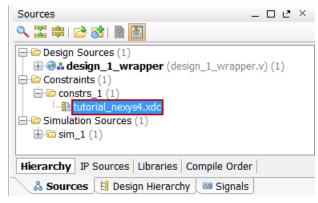


Figure 19. Constraints file added for Nexys4

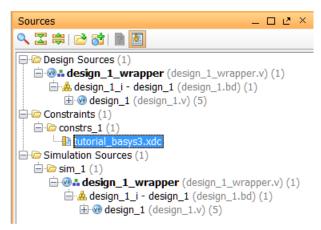


Figure 19. Constraints file added for Basys3

- **3-2-6.** In the *Sources* pane, expand the *Constraints* folder and double-click the **tutorial\_nexys4.xdc** or **tutorial\_basys3.xdc** entry to open the file in text mode.
- **3-2-7.** Lines 2-15 define the pin locations of the input switches [6:0] and lines 17-30 define the pin locations of the output LEDs [6:0]. The SW7 and LD7 are deliberately not defined so you can learn how to enter them using other methods.

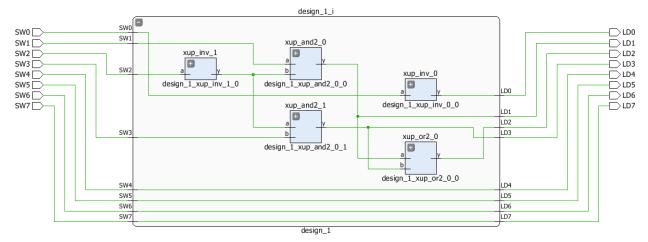
## 3-3. Perform RTL analysis on the source file.

- **3-3-1.** Expand the *Open Elaborated Design* entry under the *RTL Analysis* tasks of the *Flow Navigator* pane and click on **Schematic**.
- 3-3-2. Click Save if asked.

The model (design) will be elaborated and a logic view of the design is displayed.

**3-3-3.** Click on the + sign inside the block to see its content. Use the *Zoom Full* (S) button.





#### Figure 20. A logic view of the design

Notice that some of the switch inputs go through gates before being output to LEDs and the rest go straight through to LEDs as modeled in the file.

## 3-4. Add I/O constraints for the missing LED and switch pins.

**3-4-1.** Once RTL analysis is performed, another standard layout called the *I/O Planning* is available. Click on the drop-down button and select the *I/O Planning* layout.

😬 Default Layout	•
😬 Default Layout	
😬 IP Catalog	
💾 I/O Planning	
😬 Clock Planning	
😬 Floorplanning	
👪 Save As New Layout	
Reset Layout	F5

#### Figure 21. I/O Planning layout selection

Notice that the Package view is displayed in the Auxiliary View area, Device Constraints tab is selected, and I/O ports tab is displayed in the Console View area. Also notice that design ports (LD\* and SW\*) are listed in the I/O Ports tab with both having multiple I/O standards.

Move the mouse cursor over the Package view, highlighting different pins. Notice the pin site number is shown at the bottom of the Vivado GUI, along with the pin type (User IO, GND, VCCO...) and the I/O bank it belongs to.



Elaborated Design - xc7a100tcsg324-1 (active)						×
Device Constraints _ 🗆 🗠	× 🛛 🔠 Package	× 🔷 Device	× 🔄 RTL Schematic 🗴 🔄	RTL Sche	matic (2) ×	∢▶▣ □ ⊵×
a 🔀 🖨 🖪 🕐 🗙		1 2	3 4 5 6 7 8 9 10	11 12 13 1	4 15 16 17 18	1
▼ Internal VREF						1
				+		
	_ <u></u>			T A	-	
- DONE (5)				CICIC		
	E F			- + T		
I/O Bank 16				÷ + +		
I/O Bank 34	Ē Ō H					
Drop I/O banks on voltages or the "NONE" folder to	💾 🔂 🛃					
set/unset Internal VREF.	× K			* *		
🖧 Sources 🛛 😰 RTL Netlist 🏻 🏯 Device C	— 🂫 м			÷ + 🔂		
Properties _ 🗆 🖸				+ + +		
← → N	P <sub>n</sub> P					
	R					•
	Т					
Sector Properties Clock Regions						
I/O Ports						_ 🗆 🖻 ×
Name Direction	Neg Diff Pair Site	Fixed Bank	I/O Std	Vcco	Vref Drive S	trength Slew Typ
🔀 🖃 🕼 All ports (16)						
Scalar ports (16)		(III)				
LD0 Output	T8 V9		34 LVCMOS33*	<ul> <li>3.300</li> <li>3.300</li> </ul>		▼ SLOW
LD1 Output C LD2 Output	R8	V	34 LVCMOS33* 34 LVCMOS33*	<ul> <li>3.300</li> <li>3.300</li> </ul>		✓ SLOW ✓ SLOW
	T6	<u> </u>	34 LVCM0S33*	▼ 3.300		▼ SLOW
Cutout	Т5	V	34 LVCM0S33*	• 3.300		▼ SLOW
LD4 Output	T4	<b>V</b>	34 LVCMOS33*	· 3.300	12	▼ SLOW
✓ UD6 Output	U7	<b>V</b>	34 LVCMOS33*	<ul> <li>3.300</li> </ul>		▼ SLOW
LD7 Output			default (LVCMOS18)	* 1.800		▼ SLOW =
SW0 Input	U9	<u>&lt;</u>	34 LVCMOS33*	<ul> <li>3.300</li> </ul>		
SW1 Input SW2 Input	U8 R7	V	34 LVCMOS33*	* 3.300		
	R/ R6	V	34 LVCMOS33* 34 LVCMOS33*	<ul> <li>3.300</li> <li>3.300</li> </ul>		
SW3 Input	R5	V	34 LVCM0533*	* 3.300		
		the second se				
	V7	V	34 LVCM0S33*	<ul> <li>3.300</li> </ul>		
	V7 V6		34 LVCMOS33* 34 LVCMOS33*	<ul> <li>3.300</li> <li>3.300</li> </ul>		
SW5 Input		✓ ✓		and a second sec		

Figure 22. I/O Planning layout view of Nexys4



а UI 🛙	h 🐘 🗙 🦓   🔈 🕨 🚵   🔄	I 🥝 🚳 % 🗡 🧐 🔚	I/O Planning		🔅 🔭 (🕸				F	Rea
- «	Elaborated Design - xc7a3	5tcpg236-1 (active)								
	Sources	_ C	00× 🔳	Package ×	🔷 Device 🗙 🚹 tutorial_b	asys3.xdc 🗙 🔄	RTL Schem	atic ×		e
	🔍 🛣 😂 📾 🔂		₹I		1234567	9 0 10 11	12 13 14	15 16 17 18 1	o	
anager	Design Sources (1)		+			0 9 10 11				
ect Setti Sources	🖃 🕲 🏜 design_1_wrapp	<b>ver</b> (design_1_wrapper.v)	(1)	A	<u> </u>		S S			
		ign_1 (design_1.bd) (1)		в			<b>S S +</b>		P	
atalog	⊕ · · · · · · · · · · · · · · · · · · ·	sign_1.V) (5)	<b>Q</b> t	с	+ + + + + G	C C 🖶 C	SS≱			
tor	📄 ն constrs_1 (1)		Q-	D					4	
Block	tutorial_basys3.x	dc	<u></u>	E	+ + +			÷ o		
ock (	☐-  Simulation Sources (1) ☐-  Sim_1 (1)		<b>I</b> ⊋i	F	÷ ÷ †		100			
e Blo		apper (design_1_wrapper.	r.v)(1)	G	÷ 00 +		4 4·		4	
	🖻 📥 design_1_i -	design_1 (design_1.bd) (1)		н		± ∲ ∲ ±				
	🕀 🐨 design_1	(design_1.v) (5)	*	J			* *			
ion S				к		★	* *			
ulat	Hierarchy IP Sources Lib	raries Compile Order	, i i i i i i i i i i i i i i i i i i i	L		* * * *	<b>₩ ₩</b>			
		list 💧 Device Constraints		м		* * * *	♣ ♣			
ed C			- "	N		* * * *	÷ ÷		4	
	Source File Properties	_ C		Р					4	
rt D										
	← → 100 k			R				+		
ort N				т						
ort N				т U			<b>C</b> + <b>O</b>			
ort N emat				т U V						
ort N emat sis Se				т U		C C C ↓ C C ↓ C C ↓ C C ↓ C C				
ort D ort N emat sis Se nthes	← → <u>™</u> k	Regions		т U V		C C C ♦ C C ♦ C C C C C				
ort N emat sis Se nthes	← → So k	Regions		т U V						
ort N emat sis Se nthes ynthe	← → So k			T U V W						
ort N emat sis Se	← → So h	Regions Direction Neg	g Diff Pair Site	T U V W	k Vo Std		C + C	rength Slew Ty		
is Se thes mthe	♦ ♦ Solution Properties and Clock V/O Ports       V/O Ports       Name       Image: Solution Clock Solution	Direction Neg	g Diff Pair Site	T U V W	k I/O Std		C + O	rength Slew Ty		
t N nat s Se hes nthe	♦ ♦ ♥ Properties ♥ Clock V/O Ports Name Clock 100 Clock V Name Clock 100 Clock 100 Clock Clock 100	Direction Neg	g Diff Pair Site	T U W Fixed Ban	k 1/0 Std		c + C c - C e C - C - C - C - C - C - C - C - C - C	rength Slew Ty		
s Ni Nat Se ies the tat	♦ ♦ ♦ ♦ ♦ Ø Properties I Clock I/O Ports Name All ports (16) I Scalar ports (16) I	Direction Neg Output Output	- U16 E19	T U W Fixed Ban	14 LVCMOS33* 14 LVCMOS33*	Vcco Vrv * 3.300 * 3.300	12 12	<ul> <li>▼ SLOW</li> <li>▼ SLOW</li> </ul>	vpe Pull Typ • NONE • NONE	
t N nat s Se hes ntat em pler	← → Solve h For perties @ Clock J/O Ports J/O Ports Name Context (16) Context (1	) Output Output Output Output	U16 E19 U19	T U W Fixed Ban	14 LVCMOS33* 14 LVCMOS33* 14 LVCMOS33*	<ul> <li>Vcco Vra</li> <li>× 3.300</li> <li>× 3.300</li> <li>× 3.300</li> </ul>	12 12 12	<ul> <li>SLOW</li> <li>SLOW</li> <li>SLOW</li> </ul>	v NONE	
t N nat s Se hes the htal em oler buç se		Direction Neg Output Output Output Output	U16 E19 U19 V19	T U W Fixed Ban	14 LVCM0S33* 14 LVCM0S33* 14 LVCM0S33* 14 LVCM0S33*	<ul> <li>Vcco Vr</li> <li>▼ 3.300</li> <li>▼ 3.300</li> <li>▼ 3.300</li> <li>▼ 3.300</li> </ul>	12 12 12 12	<ul> <li>SLOW</li> <li>SLOW</li> <li>SLOW</li> <li>SLOW</li> </ul>	<ul> <li>Pull Typ</li> <li>NONE</li> <li>NONE</li> <li>NONE</li> <li>NONE</li> <li>NONE</li> </ul>	
t N nat Se nes the ler bug Se Bit		Direction Neg Output Output Output Output Output Output	U16 E19 U19 V19 W18 U15	T U W Fixed Ban	14 LVCMOS33* 14 LVCMOS33* 14 LVCMOS33* 14 LVCMOS33* 14 LVCMOS33* 14 LVCMOS33*	Vcco Vri * 3.300 * 3.300 * 3.300 * 3.300 * 3.300 * 3.300	12 12 12 12 12 12 12	SLOW     SLOW     SLOW     SLOW     SLOW     SLOW     SLOW	<ul> <li>Pull Typ</li> <li>NONE</li> <li>NONE</li> <li>NONE</li> <li>NONE</li> <li>NONE</li> <li>NONE</li> <li>NONE</li> </ul>	
t N nat Se nes the ntat em oler buç Se Bit		Direction Neg Output Output Output Output Output Output Output Output	U16 E19 U19 V19 W18	T U V W Fixed Ban	14 LVCMOS33* 14 LVCMOS33* 14 LVCMOS33* 14 LVCMOS33* 14 LVCMOS33* 14 LVCMOS33* 14 LVCMOS33*	<ul> <li>Vcco Vri</li> <li>3.300</li> <li>3.300</li> <li>3.300</li> <li>3.300</li> <li>3.300</li> <li>3.300</li> <li>3.300</li> <li>3.300</li> </ul>	12 12 12 12 12 12 12 12	<ul> <li>SLOW</li> <li>SLOW</li> <li>SLOW</li> <li>SLOW</li> <li>SLOW</li> <li>SLOW</li> <li>SLOW</li> </ul>	Pull Typ     NONE     NONE     NONE     NONE     NONE     NONE     NONE     NONE     NONE	
t N nat Se nes the ntat em oler buç Se Bit		Output Output Output Output Output Output Output Output Output	U16 E19 U19 V19 W18 U15 U14	T U V W Fixed Ban	14 LVCMOS33* 14 LVCMOS33* 14 LVCMOS33* 14 LVCMOS33* 14 LVCMOS33* 14 LVCMOS33* 14 LVCMOS33* 14 LVCMOS33* default (LVCMOS18)	Vcco Vri * 3.300 * 3.300 * 3.300 * 3.300 * 3.300 * 3.300 * 3.300 * 3.300 * 3.300 * 3.300	12 12 12 12 12 12 12	SLOW     SLOW     SLOW     SLOW     SLOW     SLOW     SLOW	NONE	
t N nat Se nes the ler bug Se Bit		Direction Neg Output Output Output Output Output Output Output Output	U16 E19 U19 V19 W18 U15	T U V W Fixed Ban	14 LVCMOS33* 14 LVCMOS33* 14 LVCMOS33* 14 LVCMOS33* 14 LVCMOS33* 14 LVCMOS33* 14 LVCMOS33*	Vcco Vri 3.300 3.300 3.300 3.300 3.300 3.300 3.300 3.300 1.800	12 12 12 12 12 12 12 12	<ul> <li>SLOW</li> <li>SLOW</li> <li>SLOW</li> <li>SLOW</li> <li>SLOW</li> <li>SLOW</li> <li>SLOW</li> </ul>	Pull Typ     NONE     NONE     NONE     NONE     NONE     NONE     NONE     NONE     NONE	
t N nat s Se hes nthe em pler ebuç n Se Bit		Output Output Output Output Output Output Output Output Output Input Input	U16 E19 U19 V19 W18 U15 U14 V17 V16 W16	T U V W Fixed Ban	14 LVCM0533* 14 LVCM0533* 14 LVCM0533* 14 LVCM0533* 14 LVCM0533* 14 LVCM0533* 14 LVCM0533* 14 LVCM0533* 14 LVCM0533* 14 LVCM0533*	<ul> <li>Vcco Vrn</li> <li>3.300</li> </ul>	12 12 12 12 12 12 12 12	<ul> <li>SLOW</li> <li>SLOW</li> <li>SLOW</li> <li>SLOW</li> <li>SLOW</li> <li>SLOW</li> <li>SLOW</li> </ul>	<ul> <li>NONE</li> </ul>	
ort N emat sis Se hthes ynthe n ental olem npler webug m Se e Bit		Direction Neg Output Output Output Output Output Output Output Output Input Input Input Input Input	U16 E19 U19 W18 U15 U14 V17 V16 W16 W12	T U V V V V V V V V V V V V V V V V V V	14 LVCMOS33* 14 LVCMOS33*	Vcco Vro 3.300 3.300 3.300 3.300 3.300 3.300 1.800 3.300	12 12 12 12 12 12 12 12	<ul> <li>SLOW</li> <li>SLOW</li> <li>SLOW</li> <li>SLOW</li> <li>SLOW</li> <li>SLOW</li> <li>SLOW</li> </ul>	<ul> <li>NONE</li> </ul>	
ort N emat is Se othes ynthe n		Direction Neg Output Output Output Output Output Output Output Output Input Input Input Input Input	U16 E19 U19 V19 W18 U15 U14 V17 V16 W16	T V V W Fixed Ban	14 LVCM0533* 14 LVCM0533* 14 LVCM0533* 14 LVCM0533* 14 LVCM0533* 14 LVCM0533* 14 LVCM0533* 14 LVCM0533* 14 LVCM0533* 14 LVCM0533*	<ul> <li>Vcco Vrn</li> <li>3.300</li> </ul>	12 12 12 12 12 12 12 12	<ul> <li>SLOW</li> <li>SLOW</li> <li>SLOW</li> <li>SLOW</li> <li>SLOW</li> <li>SLOW</li> <li>SLOW</li> </ul>	<ul> <li>NONE</li> </ul>	
rt N mat mat is Se thes rnthe n ental elem npler ebug m Se e Bit		Direction Neg Output Output Output Output Output Output Output Output Input Input Input Input Input	- U16 E19 V19 W18 U15 U14 V17 V17 V17 V16 W16 W16 W17 W15	T U V W Fixed Ban V V V V V V V V V V V V V V V V V V V	14 LVCM0533* 14 LVCM0533*	Vcco Vrn * 3.300 * 3.300	12 12 12 12 12 12 12 12	<ul> <li>SLOW</li> <li>SLOW</li> <li>SLOW</li> <li>SLOW</li> <li>SLOW</li> <li>SLOW</li> <li>SLOW</li> </ul>	Pull Type     Pull Type     NONE	
ort N mat is Se ithes /nthe n ental olem npler webuę m Se e Bit		Output Output Output Output Output Output Output Output Output Input Input Input Input Input Input Input	- U16 E19 U19 V19 U15 U15 U14 V17 V16 W17 V16 W17 V15 V15	T V V W Fixed Ban	14 LVCMOS33* 14 LVCMOS33*	Vcco Vrr • 3.300 • 3.300	12 12 12 12 12 12 12 12	<ul> <li>SLOW</li> <li>SLOW</li> <li>SLOW</li> <li>SLOW</li> <li>SLOW</li> <li>SLOW</li> <li>SLOW</li> </ul>	Pull Typ     NONE     NON	

Figure 22. I/O Planning layout view of Basys3

**3-4-2.** Click under the *I/O Std* column across the **LD7** row and select *LVCOMS33*. This assigns the LVCMOS33 standard to the site.

🖃 🕼 All ports (16)			
🖃 🗁 Scalar ports (16)			
	Output	Т8 🗸	34 LVCMOS33*
	Output	V9 🔽	34 LVCMOS33*
	Output	R8 🗸	34 LVCMOS33*
	Output	Т6 🗸	34 LVCMOS33*
	Output	R8 🗸 T6 🗸 T5 🗸	34 LVCMOS33*
	Output	T4 🔽	34 LVCMOS33*
	Output	U7 🗸	34 LVCMOS33*
🖸 LD7	Output		LVCMOS18
- 🐼 SW0	Input	U9 🗸	34 LVCMOS12 4
	Input	U8 🗸	34 LVCMOS15
	Input	R7 🗸	34 LVCMOS18
	Input	R6 🗸	
	Input	R5 🗸	34 LVCMOS25
- 🕑 SW5	Input	V7 V7 V6 V	34 LVCMOS33 -
SW6	Input	V6 🗸	34 LVTTL
	Input		MOBILE_DDR 1

Figure 23. Assigning I/O standard to Nexys4



Name	Direction	Neg Diff Pair	Site	Fixed	Bank		I/O Std		Vcco
🖃 🗁 All ports (16)									
🖻 🗁 Scalar ports (16)									
	Output		U16	1		14	LVCMOS33*		3.300
	Output		E19	1		14	LVCMOS33*		3.300
	Output		U19	1		14	LVCMOS33*	Ŧ	3.300
	Output		V19	1		14	LVCMOS33*		3.300
	Output		W18	1		14	LVCMOS33*		3.300
	Output		U15	1		14	LVCMOS33*		3.300
	Output		U14	1		14	LVCMOS33*		3.300
	Output						LVCMOS18		1.800
- 🐼 SW0	Input		V17	1		14	HSUL 12		3.300
- 🐼 SW1	Input		V16	-			LVCMOS12	-	3.300
	Input		W16	1		14	LVCMOS15	•	3.300
- SW3	Input		W17	-		14			3.300
- 🐼 SW4	Input		W15	1		14	LVCMOS18		3.300
- SW5	Input		V15	1		14	LVCMOS25 ,		3.300
- SW6	Input		W14	1		14	LVCMOS33	-	3.300
	Input						default (LVC	*	1.800

#### Figure 23. Assigning I/O standard to Basys3

- 3-4-3. Similarly, click under the *Site* column acrossLD7 row to see a drop-down box appear. Type U (for Nexys4) or V (for Basys3) in the field to jump to Uxx or Vxx pins, scroll-down until you see U6 (Nexys4) or V14 (Basys3), select U6 (Nexys4) or V14 (Bsys3) and hit the *Enter* key to assign the pin.
- **3-4-4.** You can also assign the pin constraints using tcl commands. Type in the following two commands in the Tcl Console tab to assign the *V5* (Nexys4) or *W13* (Basys3) pin location and the *LVCSMOS33* I/O standard to **SW7** hitting the Enter key after each command.

#### Nexys4:

set\_property package\_pin V5 [get\_ports SW7] set\_property iostandard LVCMOS33 [get\_ports [list SW7]]

#### Basys3:

set\_property package\_pin W13 [get\_ports SW7] set\_property iostandard LVCMOS33 [get\_ports [list SW7]]

Observe the pin and I/O standard assignments in the I/O Ports tab. You can also assign the pin by selecting its entry (SW7) in the I/O ports tab, and dragging it to the Package view, and placing it at the V5 (Nexys4) or W13 (Basys3) location. You can assign the LVCMOS33 standard by selecting its entry (SW7), selecting Configure tab of the I/O Port Properties window, followed by clicking the drop-down button of the I/O standard field, and selecting LVCMOS33.

I/O Port Propertie	es	_	00	×			
🗲 🔶 🔯 💺							
⊮ SW7							
I/O standard:	LVCMOS33	<b>–</b>					
	Evenioss	_					
Drive strength:		•					
Slew type:		•					
Pull type:	NONE	*					
In Term type:		Ψ.					
General Proper	ties Configure						
🖸 Propertie	Clock Regions						

Figure 24. Assigning I/O standard through the I/O Port Properties form



# **3-4-5.** Select **File > Save Constraints** and click **OK** to save the constraints in the **tutorial\_nexys4.xdc** or **tutorial\_basys3.xdc** file.

3-4-6. Click OK to update the existing constraint file.

Note that the constraints are updated in the tutorial.xdc file under the tutorial project directory and not under the sources directory.

## Simulate the Design using the XSim Simulator Step 4

### 4-1. Add the tutorial tb.v testbench file.

- 4-1-1. Click Add Sources under the Project Manager tasks of the Flow Navigator pane.
- 4-1-2. Select the Add or Create Simulation Sources option and click Next.
- 4-1-3. In the *Add Sources Files* form, click the **Add Files...** button.
- **4-1-4.** Browse to the <**2014\_2\_artix7\_labs**>\**tutorial** folder and select *tutorial\_tb.v* and click **OK**.
- 4-1-5. Click Finish.
- 4-1-6. Select the *Sources* tab and expand the *Simulation Sources* group.

The tutorial\_tb.v file is added under the *Simulation Sources* group, and **system\_wrapper\_1.v** is automatically placed in its hierarchy as a tut1 instance.

Sources _ D L <sup>a</sup> ×	
९ 🔀 🚔 🖬 🔂 📓 🖪	
Design Sources (1)	٦
Image: Weighter Weighter (design_1_wrapper.v) (1)	
🕀 🗁 Constraints (1)	
□· □ Simulation Sources (1)	
🖻 🗁 sim_1 (1)	
🖻 🔞 👪 tutorial_tb (tutorial_tb.v) (1)	
⊞-@ tut1 - design_1_wrapper (design_1_wrapper.v) (1)	
Hierarchy IP Sources Libraries Compile Order	_
& Sources ? Templates	_

#### Figure 25. Simulation Sources hierarchy

- 4-1-7. Using the Windows Explorer, verify that the sim\_1 directory is created at the same level as constrs\_1 and sources\_1 directories under the tutorial.srcs directory, and that a copy of tutorial\_tb.v is placed under tutorial.srcs > sim\_1 > imports > tutorial.
- 4-1-8. Double-click on the tutorial\_tb in the Sources pane to view its contents.



```
C:/xup/digital/tutorial/tutorial.srcs/sim_1/imports/tutorial/tutorial_tb.v
1 `timescale 1ns / 1ps
3 // Module Name: tutorial tb
5 module tutorial_tb(
6
     );
7
     reg [7:0] switches;
8
     wire [7:0] leds;
9
     reg [7:0] e led;
10
     integer i;
11
     design_1_wrapper tut1(
12
            .LD0(leds[0]),
13
            .LD1(leds[1]),
14
            .LD2(leds[2]),
15
            .LD3(leds[3]),
16
            .LD4(leds[4]),
17
            .LD5(leds[5]),
18
            .LD6(leds[6]),
19
            .LD7(leds[7]),
20
            .SWO(switches[0]),
            .SW1(switches[1]),
21
22
            .SW2(switches[2]),
23
            .SW3(switches[3]),
24
            .SW4(switches[4]),
25
             .SW5(switches[5]),
26
             .SW6(switches[6]),
27
            .SW7(switches[7]));
28
29
    function [7:0] expected led;
30
       input [7:0] swt;
31
    begin
32
       expected_led[0] = ~swt[0];
33
      expected_led[1] = swt[1] & ~swt[2];
34
       expected_led[3] = swt[2] & swt[3];
35
       expected_led[2] = expected_led[1] | expected_led[3];
36
       expected_led[7:4] = swt[7:4];
37
     end
38
     endfunction
39
40
     initial
41
     begin
42
        for (i=0; i < 255; i=i+2)
43
        begin
44
            #50 switches=i:
45
            #10 e led = expected led(switches);
            if(leds == e led)
46
47
                $display("LED output matched at", $time);
48
            else
49
                $display("LED output mis-matched at ",$time,": expected: %b, actual: %b", e_led, leds);
50
         end
51
     end
52
53 endmodule
```

#### Figure 26. The self-checking testbench

The testbench defines the simulation step size and the resolution in line 1. The testbench module definition begins on line 5. Line 11 instantiates the DUT (device/module under test). Lines 29 through 38 define the same module functionality for the expected value computation. Lines 40 through 51 define the stimuli generation and compares the expected output with what the DUT provides. Line 53 ends the testbench. The \$display task will print the message in the simulator console window when the simulation is run.



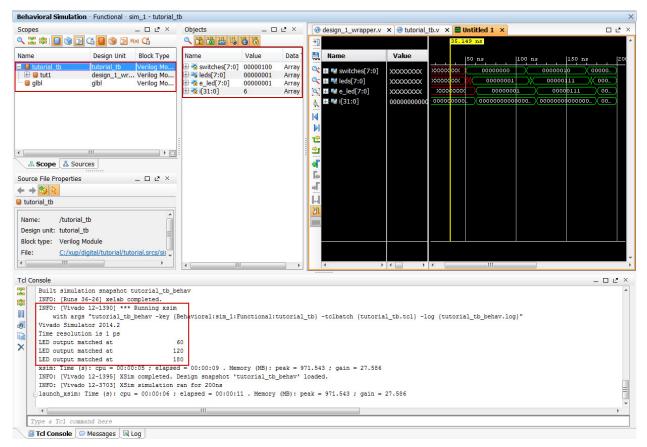
## 4-2. Simulate the design for 200 ns using the XSim simulator.

4-2-1. Select Simulation Settings under the *Project Manager* tasks of the *Flow Navigator* pane.

A **Project Settings** form will appear showing the **Simulation** properties form.

- 4-2-2. Select the Simulation tab, and set the Simulation Run Time value to 200 ns and click OK.
- **4-2-3.** Click on **Run Simulation > Run Behavioral Simulation** under the *Project Manager* tasks of the *Flow Navigator* pane.

The testbench and source files will be compiled and the XSim simulator will be run (assuming no errors). You will see a simulator output similar to the one shown below.



#### Figure 27. Simulator output

You will see four main views: (i) *Scopes*, where the testbench hierarchy as well as glbl instances are displayed, (ii) *Objects*, where top-level signals are displayed, (iii) the waveform window, and (iv) *Tcl Console* where the simulation activities are displayed. Notice that since the testbench used is self-checking, the results are displayed as the simulation is run.

Notice that the **tutorial.sim** directory is created under the **tutorial** directory, along with several lower-level directories.



🛯 👢 tutorial	*	Name
> 👢 tutorial.cache		👢 xsim.dir
Iutorial.data		
		🔍 compile
🛛 📙 tutorial.ioplanning		compile
🔺 👢 tutorial.sim		 R
4 👢 sim_1		tutorial_tb
a 👘 suu't		tutorial_tb
🛯 👢 behav		1023
> 👢 xsim.dir	, 	tutorial_tb_behav
V 👞 XSIII.UII		tutorial_tb_behav.wdb
Interview Parameters		1023
> 📙 tutorial 1		📄 xelab
		xelab.pb
> 1. embedded		1075
Goals		(i) xsim

Figure 28. Directory structure after running behavioral simulation

**4-2-4.** Click on the *Zoom Fit* button (S) located left of the waveform window to see the entire waveform.

Notice that the output changes when the input changes.

You can also float the simulation waveform window by clicking on the Float button on the upper right hand side of the view. This will allow you to have a wider window to view the simulation waveforms. To reintegrate the floating window back into the GUI, simply click on the Dock Window button.





Figure 30. Dock Window Button

## 4-3. Change display format if desired.

4-3-1. Select i[31:0] in the waveform window, right-click, select *Radix*, and then select *Unsigned Decimal* to view the for-loop index in *integer* form. Similarly, change the radix of switches[7:0] to *Hexadecimal*. Leave the leds[7:0] and e\_led[7:0] radix to *binary* as we want to see each output bit.

# 4-4. Add more signals to monitor lower-level signals and continue to run the simulation for 500 ns.

**4-4-1.** Expand the **tutorial\_tb** instance, if necessary, in the *Scopes* window and select the **tut1** instance.

The SW\* (7 to 0)] and LD\* (7 to 0) signals will be displayed in the Objects window.



Scopes		_ 🗆 🖉 ×	Objects	_	□Ľ×		
🔍 🛣 🖨 🚺 🎯 🕻	i 🖬 🕲 📕 i	F(x) 🚰	Q 🕆 👸 🛗	16 18	6 6		
Name	Design Unit	Block Type	Name	Value	Data 1		
🖃 🛢 tutorial_tb	tutorial_tb	Verilog Mo	: 📸 LD0	1	Logic		
🗄 📕 tut1	design_1_wr	Verilog Mo		0	Logic		
🛄 🛢 gibi	glbl	Verilog Mo		0	Logic		
	_			0	Logic		
			🛗 LD4	0	Logic		
				0	Logic		
			LD6	0	Logic		
			🛗 LD7	0	Logic		
				0	Logic		
			- 🛗 SW1	0	Logic		
			🖥 SW2	1	Logic		
•	111		: 🖓 SW3	0	Logic		
👗 Scope 🛛 🖧 So	urces		💾 SW4	0	Logic		
				0	Logic		
Simulation Scope Pro	perties	- 🗆 🖻 ×	💾 SW6	0	Logic		
← → 🚱 📐				0	Logic		

Figure 31. Selecting lower-level signals

- **4-4-2.** Select **SW**<sup>\*</sup> and **LD**<sup>\*</sup> and drag them into the waveform window to monitor those lower-level signals.
- **4-4-3.** On the simulator tool buttons ribbon bar, type 500 in the time window, click on the drop-down button of the units field and select ns, and click on the ( $\boxed{100}$ ) button.

The simulation will run for an additional 500 ns.

**4-4-4.** Click on the *Zoom Fit* button and observe the output.

<b>→</b> □		35.149 ns
Name	Value	0 ns  200 ns  400 ns  600 ns
🕇 🖽 🐨 switches[7:0]	$\times$	
Heds[7:0]	XXXXXXXXXX	🗙 🗶 00 🗶 00 🗶 00000001 🗶 00 🗴 000 🗶 00000001 🗶 00 🗶 00000001 🗶 000 🌋
🕻 🖽 🖏 e_led[7:0]	XXXXXXXXX	xxxx,
⊞ 📲 i[31:0]	0	
LD0	X	
La LDI	x	
1 LD2	X	
🖆 🔓 LD3	x	
🛂 🖓 LD4	x	
🔽 🔓 LD5	x	
1 LD6	x	
LD7	x	
In SWO	0	
📲 🖓 SW1	0	
1 🔓 SW2	1	
💷 🖓 SW3	0	
1 <sup>1</sup> ∎ SW4	0	
1 <sup>1</sup> SW5	0	
1 SW6	0	
₩ SW7	0	

Figure 32. Running simulation for additional 500 ns



- **4-4-5.** Close the simulator by selecting **File > Close Simulation**.
- 4-4-6. Click OK and then click No to close it without saving the waveform.

## Synthesize the Design

Step 5

# 5-1. Synthesize the design with the Vivado synthesis tool and analyze the Project Summary output.

5-1-1. Click on Run Synthesis under the Synthesis tasks of the Flow Navigator pane.

The synthesis process will be run on the tutorial.v file (and all its hierarchical files if they exist). When the process is completed a *Synthesis Completed* dialog box with three options will be displayed.

**5-1-2.** Select the *Open Synthesized Design* option and click **OK** as we want to look at the synthesis output before progressing to the implementation stage.

Click **Yes** to close the elaborated design if the dialog box is displayed.

**5-1-3.** Select the **Project Summary** tab (Select default layout if the tab is not visible) and understand the various windows.

	ject Summary		_ 0 2	ıх
	Project Settings		Edit	*
<b>\$</b>	Project name: tutorial			
	Product family: Artix-7			
	Project part: <u>xc7a100tcsg324-1</u>			
	Top module name: <u>design 1 wrapper</u>			
	Synthesis	*	Implementation	*
	Status: 🗸 Complete		Status:   Not started  Implementatio	on
	Messages: 1 <u>1 warning</u> Synthesis		Messages: No errors or warnings not started	
	Part: xc7a100tcsg324-1 Completed		Part: xc7a100tcsg324-1	
	Strategy: Vivado Synthesis Defaults		Strategy: <u>Vivado Implementation Defaults</u>	
			Incremental Compile: None	
	DRC Violations	*	Timing	*
	DRC information is not available because it hasn't been run		Timing information is not available because it hasn't been run	
	Utilization - Post-Synthesis	*	Power	*
	LUT - 1% I/O - 8% 0 25 50 75 100 Estimated Utilization (%) Utilization shown in a Graph form Graph Table Utilization can be viewed i Post-Synthesis Post-Implementation	n_a	Power information is not available because it hasn't been run Table form	
	Post-Synthesis Post-Implementation			

Figure 33. Project Summary view

Click on the various links to see what information they provide and which allows you to change the synthesis settings.



#### 5-1-4. Click on the Table tab in the Project Summary tab.

Notice that there are an estimated five LUTs and 16 IOs (8 input and 8 output) that are used.

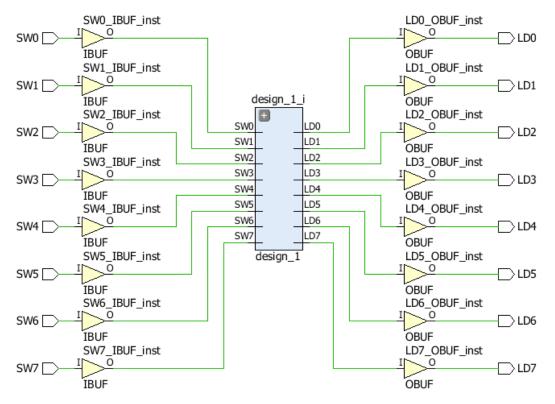
Resource	Estimation	Available	Utilization
LUT	5	63400	0.01
I/O	16	210	7.62

Resource	Estimation	Available	Utilization
LUT	5	20800	0.02
I/O	16	106	15.09

#### Graph Table

#### Figure 34. Resource utilization estimation summary for Basys3

**5-1-5.** Click on **Schematic** under the *Open Synthesized Design* tasks of *Synthesis* tasks of the *Flow Navigator* pane to view the synthesized design in a schematic view.

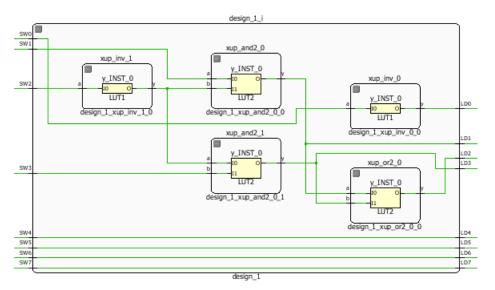


#### Figure 35. Synthesized design's schematic view

Notice that IBUF and OBUF are automatically instantiated (added) to the design as the input and output are buffered.



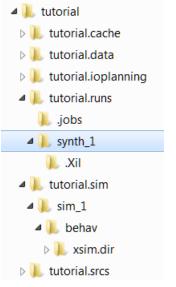
- **5-1-6.** Click on the + sign within the *design\_1* block to see the underlying logic.
- **5-1-7.** Click on the + sign of each of the lower-level blocks to see their implementation.

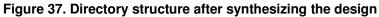


#### Figure 36. Lower-level logic

The logical gates are implemented in LUTs (1 input is listed as LUT1 and 2 input is listed as LUT2). Five blocks in RTL analysis output are mapped into five LUTs in the synthesized output.

Using the Windows Explorer, verify that **tutorial.runs** directory is created under **tutorial**. Under the **runs** directory, **synth\_1** directory is created which holds several temporary sub-directories.







## Implement the Design

Step 6

- 6-1. Implement the design with the Vivado Implementation Defaults (Vivado Implementation 2014) settings and analyze the Project Summary output.
- 6-1-1. Click on Run Implementation under the Implementation tasks of the Flow Navigator pane.

The implementation process will be run on the synthesis output files. When the process is completed an *Implementation Completed* dialog box with three options will be displayed.

- 6-1-2. Select **Open implemented design** and click **OK** as we want to look at the implemented design in a Device view tab.
- 6-1-3. Click Yes to close the synthesized design.

The implemented design will be opened.

6-1-4. In the *Netlist* pane, select one of the nets (e.g. n\_0\_design\_1\_i) and notice that the displayed net.

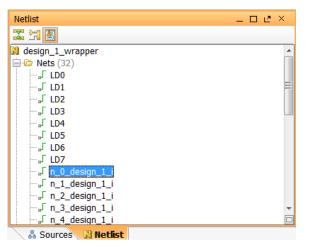


Figure 38: Selecting a net

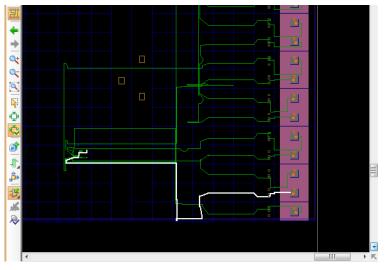


Figure 39. Viewing implemented design for Nexys4



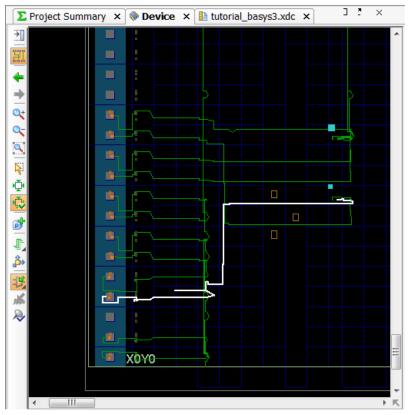


Figure 39. Viewing implemented design for Basys3

**6-1-5.** Close the implemented design view and select the **Project Summary** tab (you may have to change to the Default Layout view) and observe the results.

Notice that the actual resource utilization is three LUTs and 16 IOs. Also, it indicates that no timing constraints were defined for this design (since the design is combinatorial). Select the **Post-implementation** tabs under the *Timing* and *Utilization* windows.

- 6-1-6. Using the Windows Explorer, verify that impl\_1 directory is created at the same level as synth\_1 under the tutorial.runs directory. The impl\_1 directory contains several files including the report files.
- **6-1-7.** Select the **Reports** tab, and double-click on the *Utilization Report* entry under the *Place Design* section. The report will be displayed in the auxiliary view pane showing resources utilization. Note that since the design is combinatorial no registers are used.

## **Perform Timing Simulation**

Step 7

## 7-1. Run a timing simulation.

**7-1-1.** Select **Run Simulation > Run Post-Implementation Timing Simulation** process under the *Simulation* tasks of the *Flow Navigator* pane.

The XSim simulator will be launched using the implemented design and the **tutorial\_tb** as the top-level module.

Vivado IPI Tutorial-26



Using the Windows Explorer, verify that **timing** directory is created under the **tutorial.sim** > **sim\_1** > **impl** directory. The **timing** directory contains generated files to run the timing simulation.

- 7-1-2. Click on the Zoom Fit button to see the waveform window from 0 to 200 ns.
- 7-1-3. Right-click at 50 ns (where the switch input is set to 0000000b) and select Markers > Add Marker.
- 7-1-4. Similarly, right-click and add a marker at around 55.000 ns where the leds changes.
- **7-1-5.** You can also add a marker by clicking on the Add Marker button (<sup>44</sup>). Click on the Add Marker button and left-click at around 60 ns where **e\_led** changes.

₹								
10	Name	Value		5	0.000	6	0.000 ns <mark>s</mark>	
nu	Name	value	40 ns			6	) ns	80 ns
Q+	🖽 🐝 switches[7:0]	XXXXXXXXXX	XXXXX	C			0000000	
0-	🖽 📲 leds[7:0]	XXXXXXXXX	XXXXXXX	x	K 💓		000000	901
	🖽 📲 e_led[7:0]	XXXXXXXXXX	>>>>>	x	XX )	C	00000	001
<u>&amp;</u>	⊞•¶i[31:0]	000000000000000000000000000000000000000	0000000	10	000	ſ	000000000000000000000000000000000000000	0000000

Figure 40. Timing simulation output

Notice that we monitored the expected led output at 10 ns after the input is changed (see the testbench) whereas the actual delay is about 5.000 ns.

**7-1-6.** Close the simulator by selecting **File > Close Simulation** without saving any changes.

## Generate the Bitstream and Verify Functionality

### Step 8

# 8-1. Connect the board and power it ON. Generate the bitstream, open a hardware session, and program the FPGA.

8-1-1. Click on the Generate Bitstream entry under the *Program and Debug* tasks of the *Flow Navigator* pane.

The bitstream generation process will be run on the implemented design. When the process is completed a *Bitstream Generation Completed* dialog box with three options will be displayed.

This process will have **design\_1\_wrapper.bit** file generated under **impl\_1** directory which was generated under the **tutorial.runs** directory.

**8-1-2.** Make sure that the power supply source is jumper to *USB* and the provided Micro-USB cable is connected between the board and the PC.

Note that you do not need to connect the power jack and the board can be powered and configured via USB alone



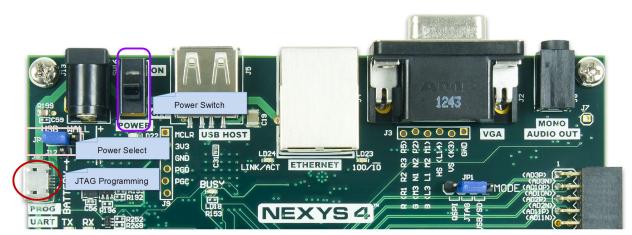


Figure 41. Board settings for Nexys4



Figure 41. Board settings for Basys3

- 8-1-3. Power **ON** the switch on the board.
- 8-1-4. Select the Open Hardware Manager option and click OK.

The Hardware Session window will open indicating "unconnected" status.

8-1-5. Click on the Open a new hardware target link.

You can also click on the Open Recent Hardware Target link if the board was already targeted before.

Hardware Manager - unconnected							
() No hardware target is open. Op	en recent target	<u>Oper</u>	n a new hardware t	target			
Hardware	- 0 0	×	Debug Probes	_ 0			
			🔍 🔀 🖨 🛃				

Figure 42. Opening new hardware target

- 8-1-6. Click Next to see the Vivado CSE Server Name form.
- 8-1-7. Click Next with the localhost port selected.

The JTAG cable will be searched and the Xilinx\_tcf should be detected and identified as e hardware target. It will also show the hardware devices detected in the chain.



🚴 Open N	lew Har	dware	Target			X			
Select a	Select Hardware Target Select a hardware target from the list of available targets, then set the appropriate JTAG clock (TCK) frequency. If you do not see the expected devices, decrease the frequency or								
Hardware T	argets								
Туре	Port	Name	•	JTAG Clock Frequency					
xilinx_tcf		Digiler	nt/2102745524	167A 15000000	·				
Hardware D	evices (	for unk	nown devices,	specify the Instruction Regist	er (IR) length)				
Name	ID	Code	IR Length						
🔷 xc7a100	t_0 136	31093	6						
VCSE server: localhost:60001									
Hardware s	erver: lo	calhos	:3121						
				< <u>B</u> ack N	ext >	Cancel			

Figure 43. New hardware target detection for Nexys4

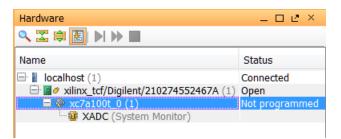
ardware 1 -				
Fype Xilinx_t	Port	Name Digilent/21018356410	JTAG Clock Frequency	
rdware [	Devices (for	unknown devices, spe	ecify the Instruction Register (IR) length)	
	Devices (for ID Co		ecify the Instruction Register (IR) length)	
Vame		de IR Length	ecify the Instruction Register (IR) length)	
lame	ID Co	de IR Length	ecify the Instruction Register (IR) length)	
lame	ID Co	de IR Length	ecify the Instruction Register (IR) length)	

Figure 44. New hardware target detection for Basys3



#### 8-1-8. Click Next and Finish.

The Hardware Session status changes from Unconnected to the server name and the device is highlighted. Also notice that the Status indicates that it is not programmed.



#### Figure 45. Opened hardware session for Nexys4

Hardware	_ 🗆 🖻 ×				
State 1					
Name	Status				
🖃 🚪 localhost (1)	Connected				
🖻 📓 🤌 xilinx_tcf/Digilent/21018356410	Open				
🗏 🚸 xc7a35t_0 (1)	Not programmed				
🦾 🤷 XADC (System Monitor)					

Figure 45. Opened hardware session for Basys3

**8-1-9.** Select the device and verify that the **design\_1\_wrapper.bit** is selected as the programming file in the General tab.

Hardware			- 🗆 🖻 ×
🔍 🛣 🖨 🛃 🕨			
Name		Status	
🖃 📕 localhost (1)		Connected	
	Digilent/210274552467A (1)	Open Not programmed	
🗖 🛞 xc7a100			
- /////	(System Homeory		
Hardware Device Pr	operties		_ D 2 ×
← → 🔁 🦎			
xc7a100t_0			
Name:	xc7a100t 0		
	-		
Part:	xc7a100t		
ID code:	13631093		
IR length:	6		
Status:	Not programmed		
Programming file:	C:/xup/digital/tutorial/tutor	ial.runs/impl_1/design_1	wrapper.bit 🛛 🖳
Probes file:	C:/xup/digital/tutorial/tutor	ial.runs/impl_1/debug_ne	ets.ltx 💿 📖
User chain count:	4		
General Propertie	es		

Figure 46. Programming file for Nexys4



Hardware		_ □ Ľ ×
🔀 🖨 🛃 🕨 🕨 🔳		
Name	Status	
🖃 📕 localhost (1)	Connected	
📄 📓 🖉 xilinx_tcf/Digilent/21018359851		
🔷 xc7a35t_0 (0)	Not programmed	
Hardware Device Properties		_ 🗆 🖻 ×
xc7a35t 0		
F XC/035(_0		
xc7a35t_0		
xc7a35t		
0362D093		
6		
Not programmed		
Not programmed		
g file: C:/xup/digital/2014_2_artix7_labs	/tutorial/tutorial.runs/impl_1/design_1_w	apper.bit 🛛 🖳
	/tutorial/tutorial.runs/impl_1/design_1_wi /tutorial/tutorial.runs/impl_1/debug_nets.	
C:/xup/digital/2014_2_artix7_labs		
C:/xup/digital/2014_2_artix7_labs		
C:/xup/digital/2014_2_artix7_labs		
C:/xup/digital/2014_2_artix7_labs		
C:/xup/digital/2014_2_artix7_labs		

Figure 46. Programming file for Basys3

- 8-1-10. Right-click on the device and select *Program Device…* to program the target FPGA device.
- 8-1-11. Click **Program** to program the FPGA with the selected bitstream.

The DONE light will lit when the device is programmed. You may see some LEDs lit depending on the switches position.

8-1-12. Verify the functionality by flipping switches and observing the output on the LEDs.

8-1-13. Close the hardware session by selecting File > Close Hardware Manager.

- 8-1-14. Click OK to close the session.
- 8-1-15. Power OFF the board.
- 8-1-16. Close the Vivado program by selecting File > Exit and click OK.

## Conclusion

The Vivado software tool can be used to perform a complete design flow. The project was created using the XUP IP library (IPI blocks and user constraint file). A behavioral simulation was done to verify the model functionality. The model was then synthesized, implemented, and a bitstream was generated. The timing simulation was run on the implemented design using the same testbench. The functionality was verified in hardware using the generated bitstream.

