

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 Nexys4 board.

Objectives

After completing this tutorial, you will be able to:

- Create a Vivado project targeting a specific FPGA device located on the Nexys4
- 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**.

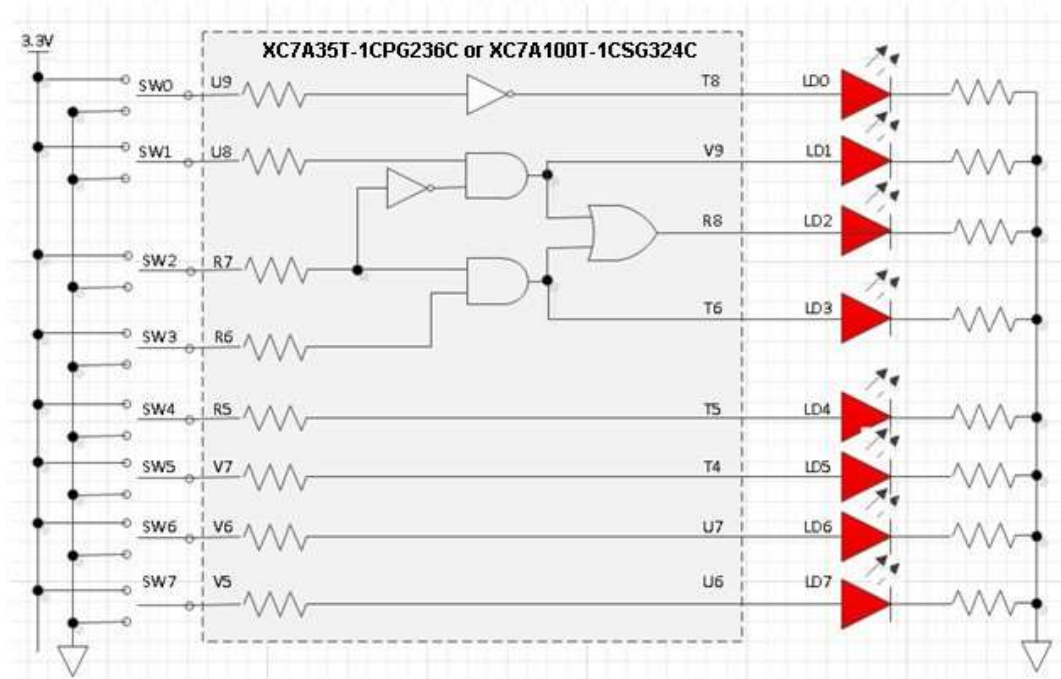


Figure 1. Completed Design

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

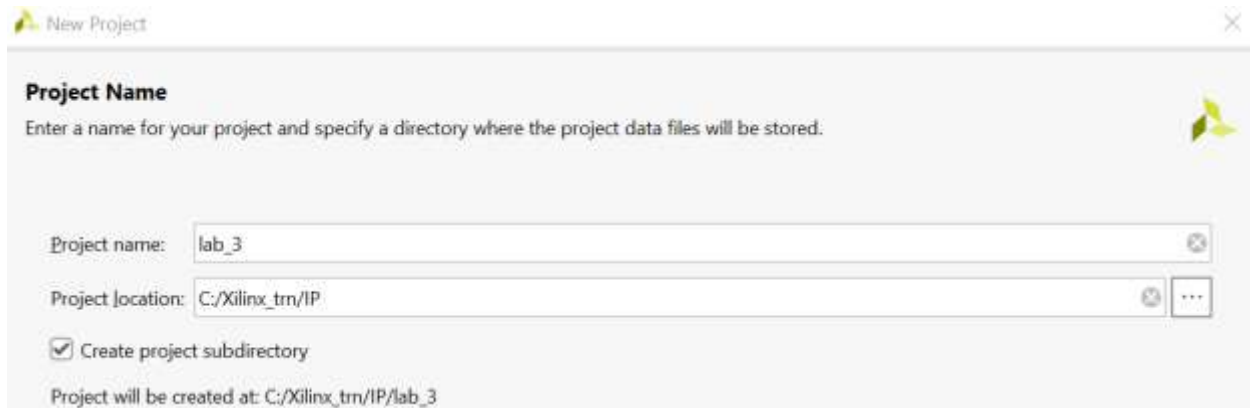
1-1. Launch Vivado and create a project targeting the Nexys4 and using the Verilog HDL. Use the provided *nexys4.xdc* (for Nexys4) file from C:\Xilinx_trn\IP\sources directory.

1-1-1. Open Vivado

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 **C:\Xilinx_trn\IP** and click **Select**.

1-1-4. Enter **lab_3** in the *Project name* field. Make sure that the *Create Project Subdirectory* box is checked. Click **Next**.



The screenshot shows the 'New Project' dialog box. At the top, it says 'Project Name' and 'Enter a name for your project and specify a directory where the project data files will be stored.' Below this, there are two input fields: 'Project name:' with the value 'lab_3' and 'Project location:' with the value 'C:/Xilinx_trn/IP'. There is a checkbox labeled 'Create project subdirectory' which is checked. At the bottom, it says 'Project will be created at: C:/Xilinx_trn/IP/lab_3'.

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. The *Add Constraints* form will be opened

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.

1-1-9. Click **Next**.

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). Click **Next**.

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

Use the Windows Explorer and look at the **C:\Xilinx_trn\IP\lab_3** directory. You will find that the **lab_3.cache** directory and the **lab_3.xpr** (Vivado) project file.

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

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

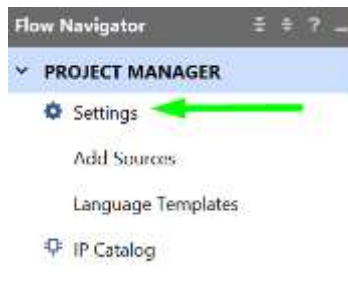


Figure 3. Invoking Project Settings to set IP repository path

1-2-2. In the *Project Settings* window, click on the **IP=>Repository**.

1-2-3. Click on the **Add Repositories** button **+**, browse to **C:\Xilinx_trn\IP\sources** and select **XUP_LIB** directory, and click **Select**. Then click OK.

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

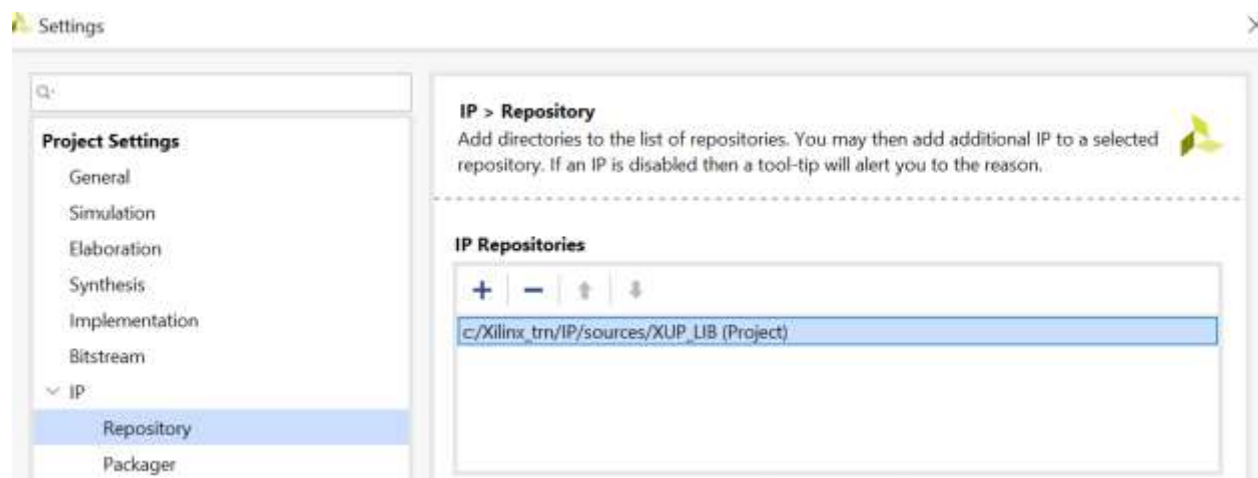


Figure 4. Setting IP Repository

1-2-4. Click **OK**.

Create a Block Design

Step 2

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 5. Invoking IP Integrator to create a block design

2-1-2. Name the design name field as **lab_3** and click **OK** to create a block design named *lab_3*.

2-1-3. IP from the catalog can be added in different ways. Click on **+** at the top of the *Diagram* panel, or click the *Add IP icon* **+** in the block diagram workspace, or press Ctrl + I, or right-click anywhere in the Diagram workspace and select Add IP


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.

2-1-5. Add another instance of an inverter.

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.

2-1-6. 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  button. At this stage the block diagram should look like shown below.

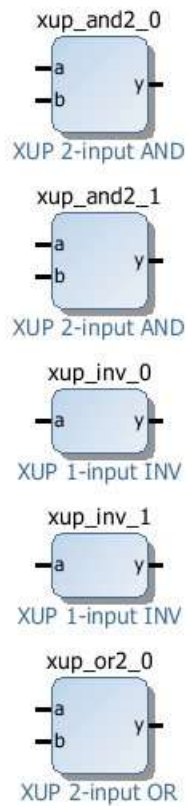


Figure . Added necessary instances

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.

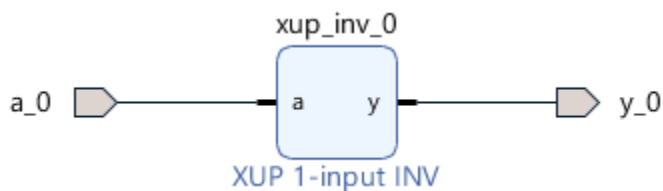


Figure 11. Making ports external

2-2-2. Click on the **a_0** port, and change the name to **SW0** in its properties form.

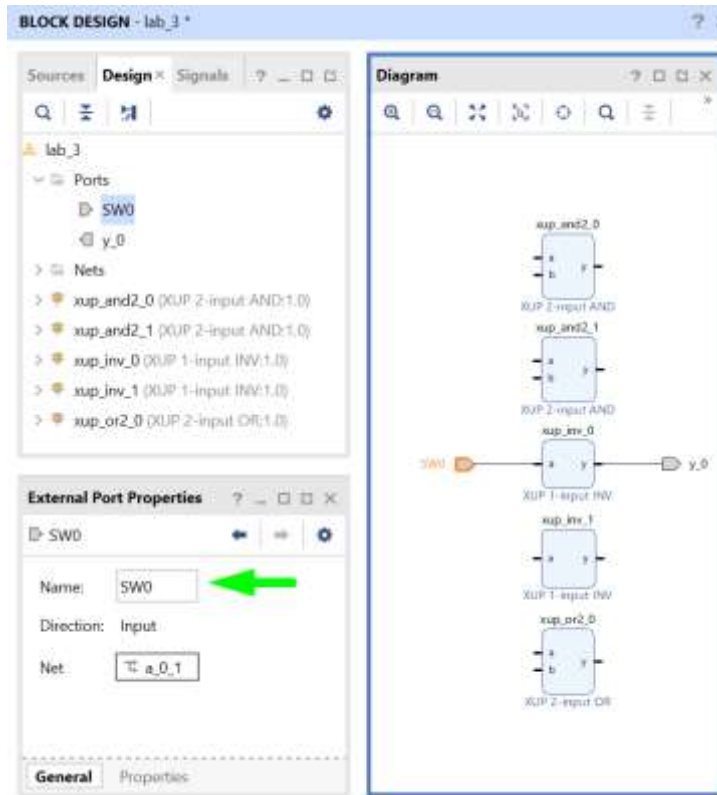
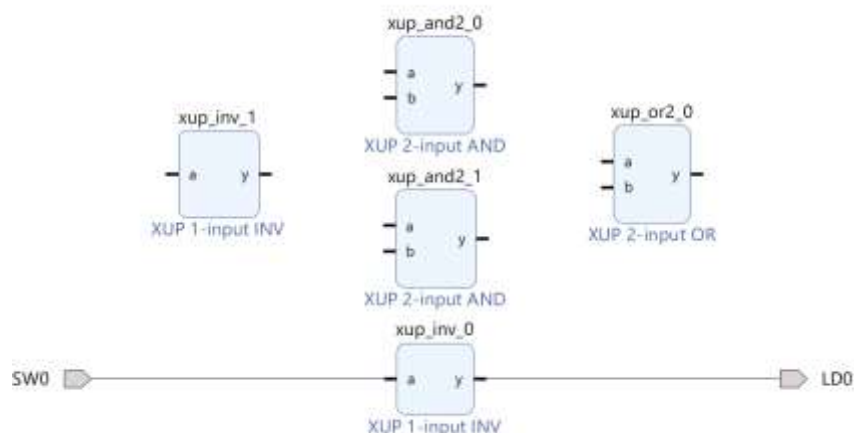


Figure. Setting input port name to SW0

2-2-3. Similarly, change the output port y_0 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.

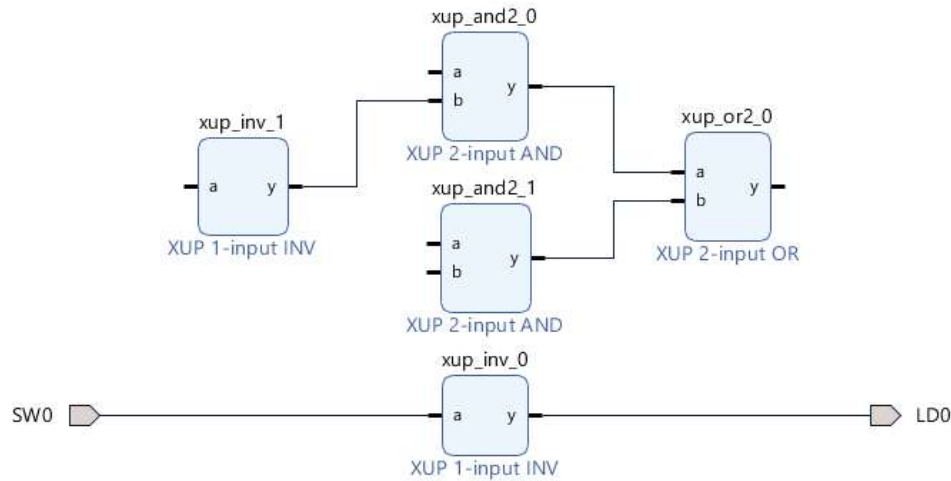


Figure. Connecting instances

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.

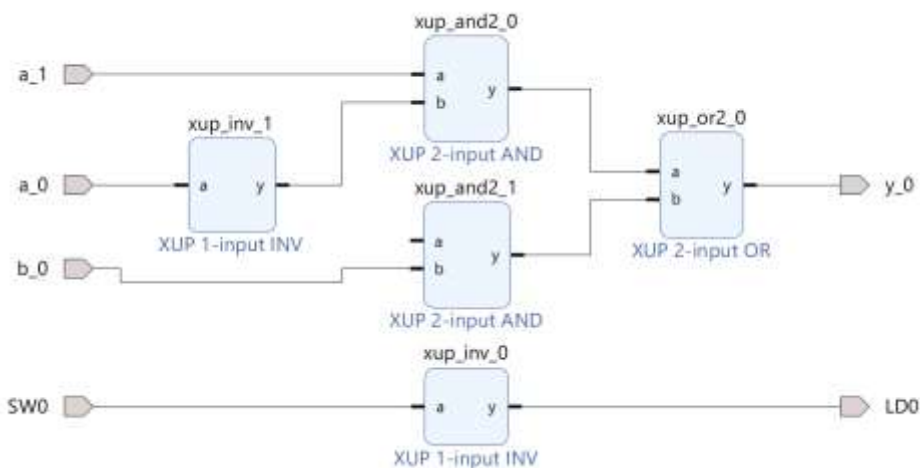
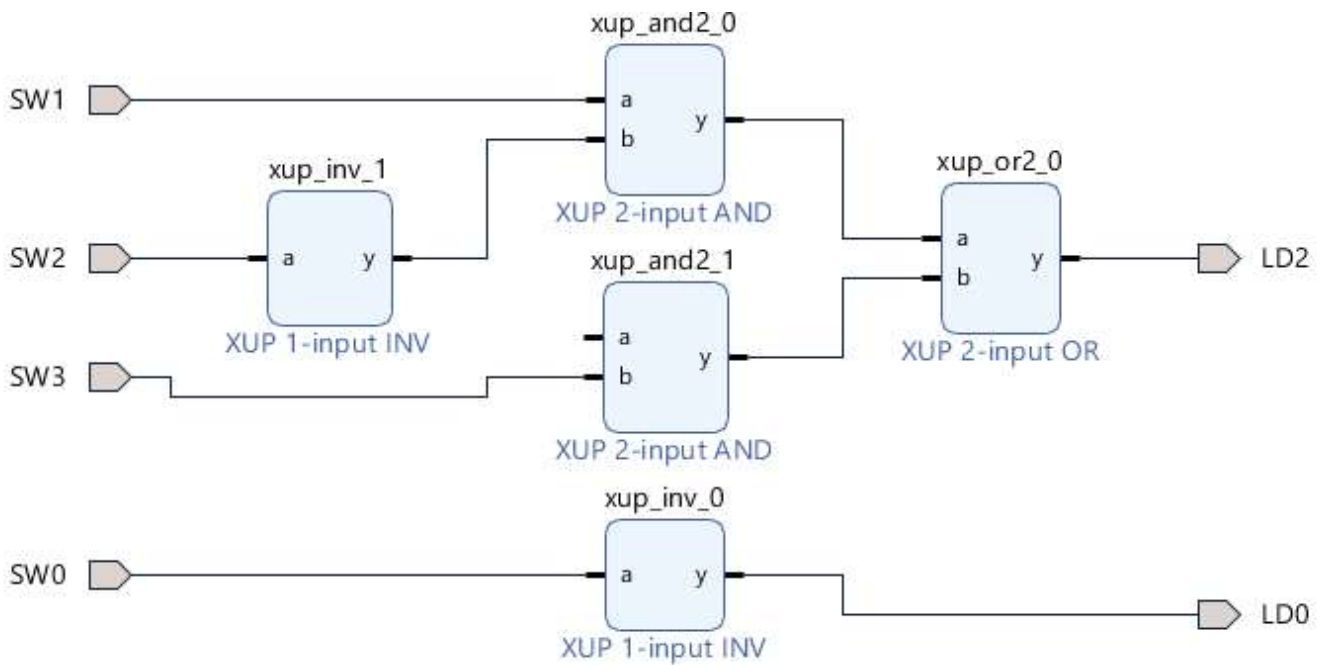


Figure 14. Making ports external

2-2-10. Change the name of **a_1** to **SW1**, **a_0** to **SW2**, **b_0** to **SW3**, and **y_0** 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**.

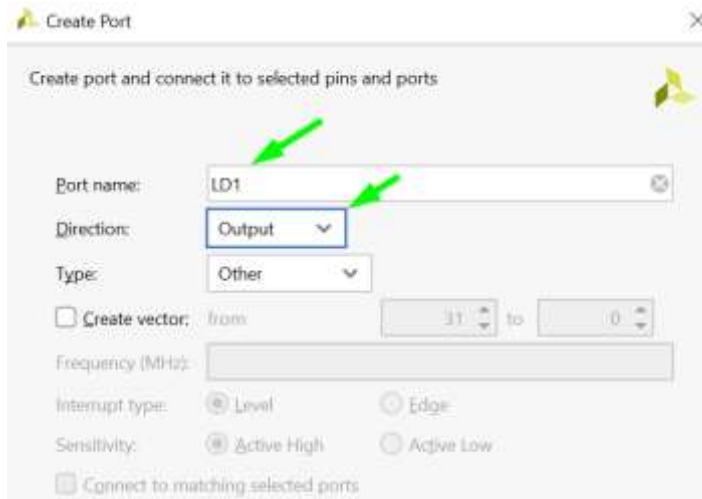



Figure 15. Creating an output port

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**.

2-15. Connect the output port of the **xup_and2_0** to **LD1** and **xup_and2_1** to **LD3**.

2-16. Click on the redraw button  .

The diagram will look similar to shown below.

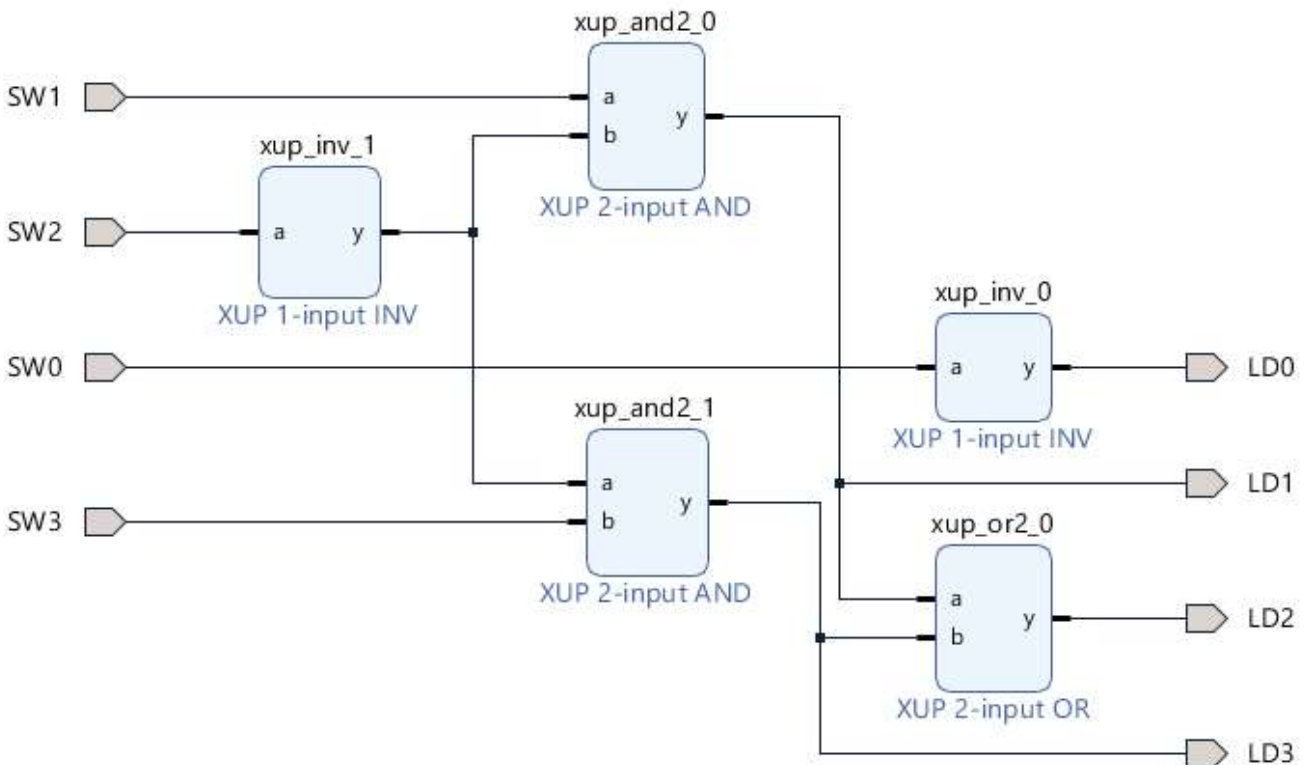


Figure. Partially completed design

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

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

2-3-3. Right-click on the canvas and create an *LD4* as output ports.

2-3-4. Right-click on the canvas and create an *LD4*, *LD5*, *LD6*, and *LD7* as output ports.

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

2-3-4. Using wiring tool, connect *SW4* to *LD4*, *SW5* to *LD5*, *SW6* to *LD6*, and *SW7* to *LD7*.

2-3-5. Click the re-draw button .



The design should look like as shown below.

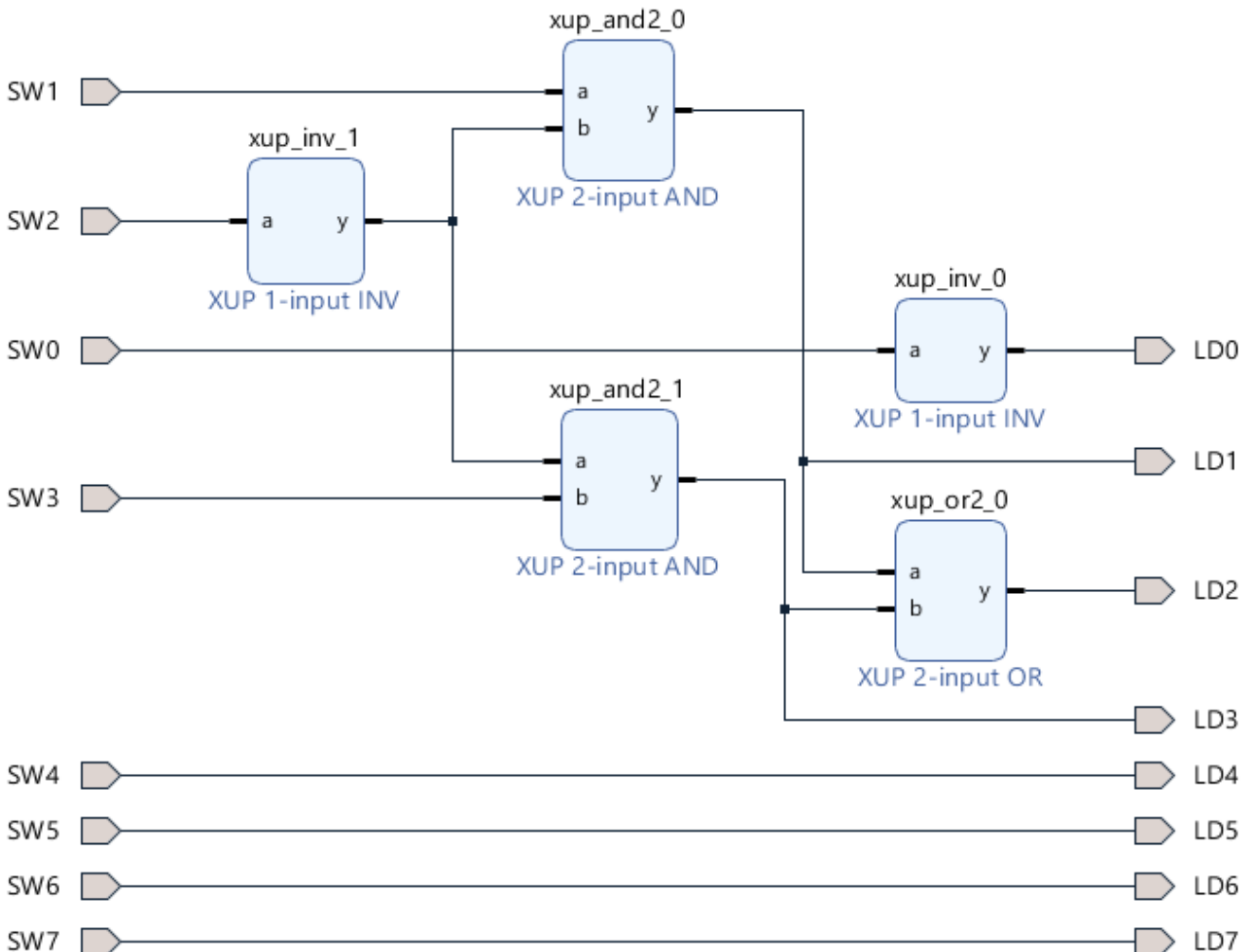


Figure. The completed design

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, **lab_3.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 lab_3_wrapper file instantiates lab_3 which in turn instantiates the inverter twice, and2 twice, and or2 once.

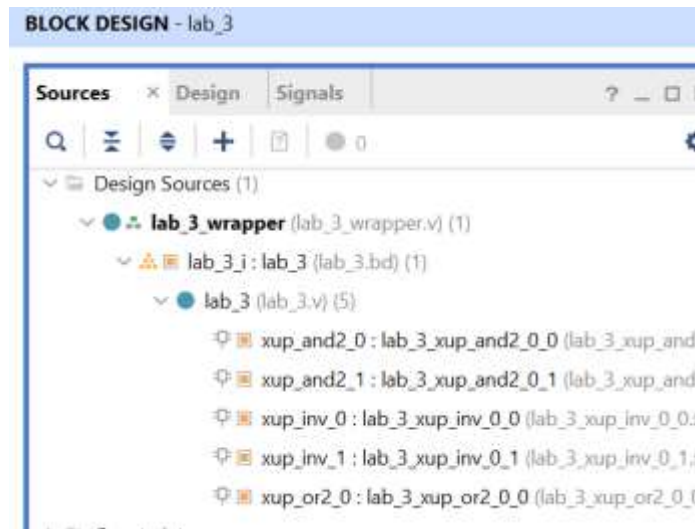


Figure. Hierarchical design

3-1-3. Double-click the **lab_3_wrapper.v** entry to open the file in text mode and observe the instantiation of the *lab_3* module.

3-1-4. Double-click the **lab_3.v** entry to open the file in text mode and observe the instantiation of the lower-level modules.

3-2. Add nexys4.xdc (for Nexys4) 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 **C:\Xilinx_trn\IP\sources**.

3-2-4. Select **nexys4.xdc** (for Nexys4) and click **OK**.

3-2-5. Click **Finish** to close the window and add the constraints file in the project under the Constraints group.

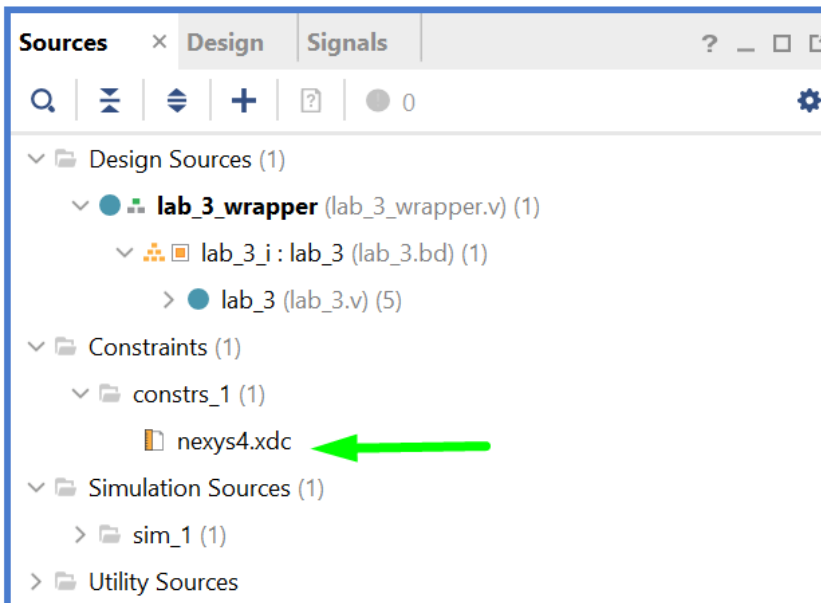
BLOCK DESIGN - lab_3

Figure. Constraints file added for Nexys4

3-2-6. In the *Sources* pane, expand the *Constraints* folder and double-click the **nexys4.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** and OK 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* () button.

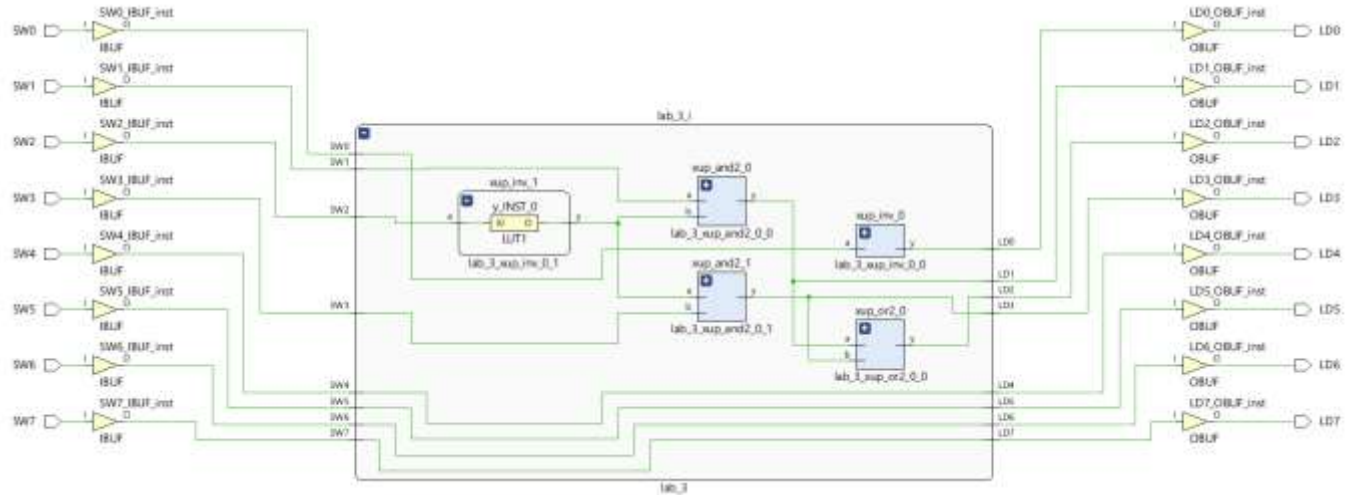


Figure. A logic view of the design

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

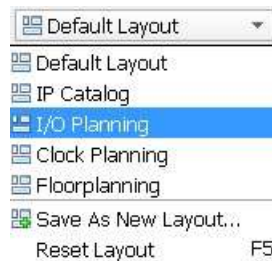


Figure 21. I/O Planning layout selection

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.

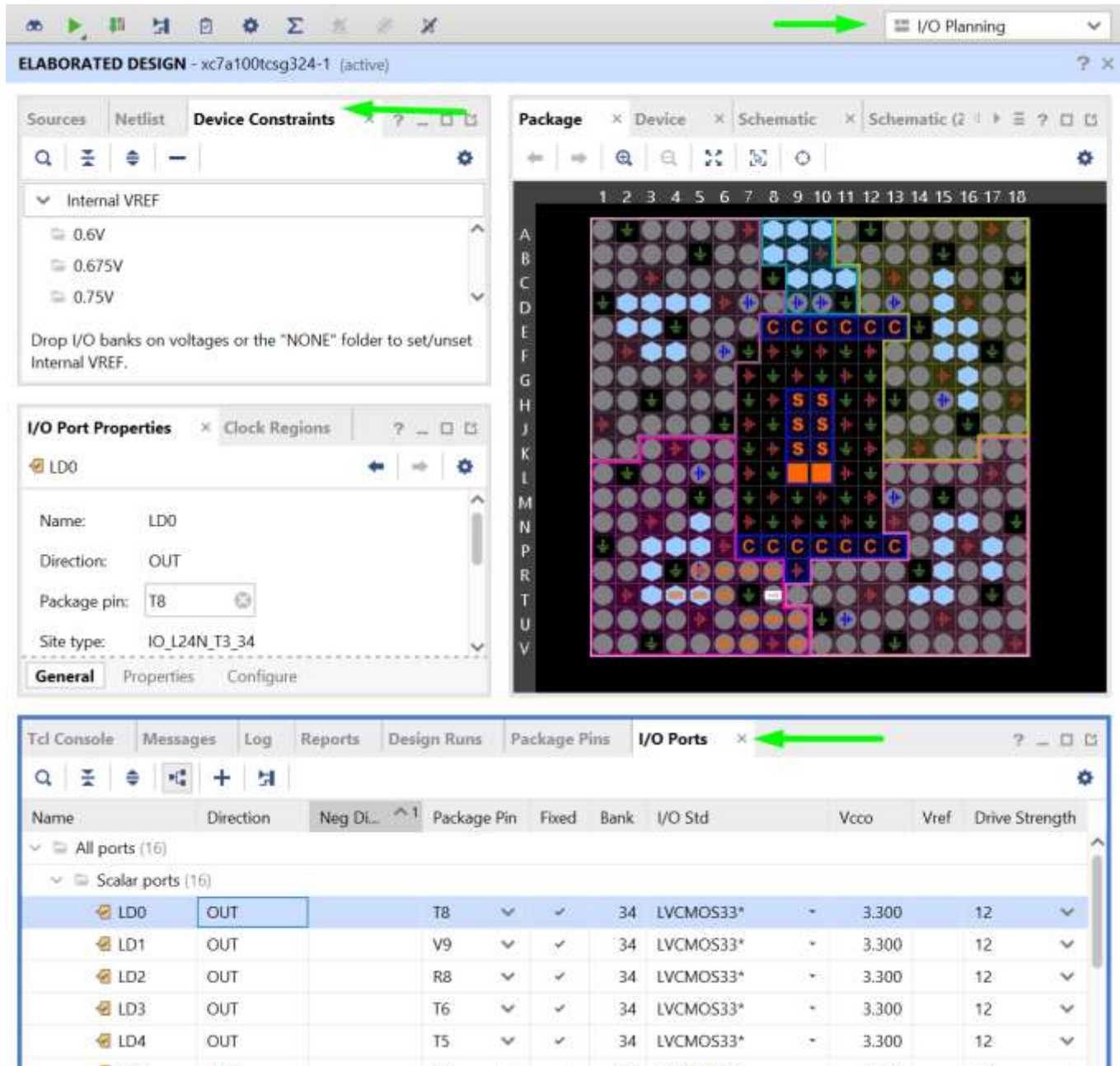


Figure 22. I/O Planning layout view of Nexys4

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 *LVC MOS33*. This assigns the LVC MOS33 standard to the site.

Port	Direction	Pin	Standard
LD0	Output	T8	34 LVC MOS33*
LD1	Output	V9	34 LVC MOS33*
LD2	Output	R8	34 LVC MOS33*
LD3	Output	T6	34 LVC MOS33*
LD4	Output	T5	34 LVC MOS33*
LD5	Output	T4	34 LVC MOS33*
LD6	Output	U7	34 LVC MOS33*
LD7	Output		LVC MOS18
SW0	Input	U9	34 LVC MOS12
SW1	Input	U8	34 LVC MOS15
SW2	Input	R7	34 LVC MOS18
SW3	Input	R6	34 LVC MOS25
SW4	Input	R5	34 LVC MOS33
SW5	Input	V7	34 LVTTL
SW6	Input	V6	34 LVTTL
SW7	Input		MOBILE_DDR

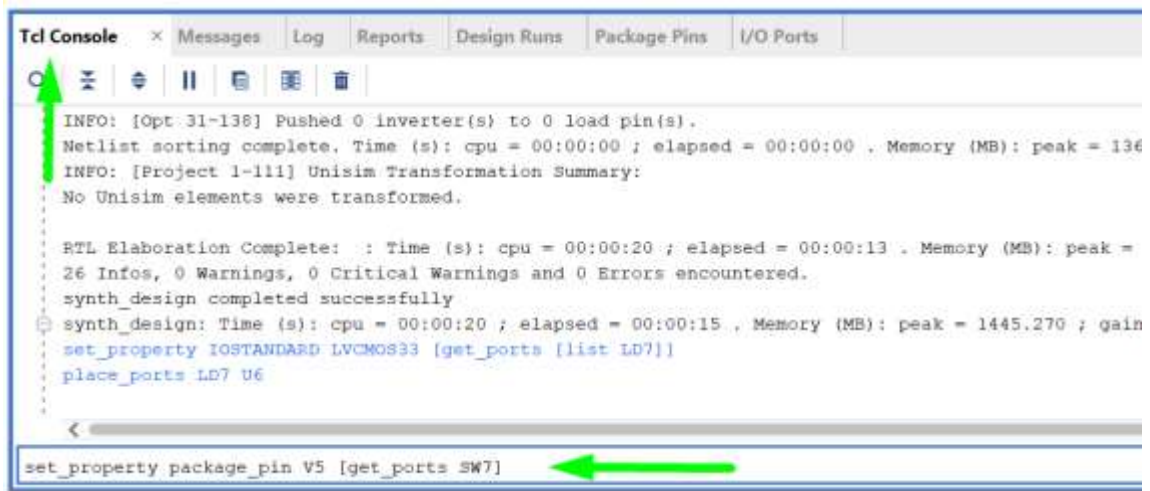
Figure 23. Assigning I/O standard to Nexys4

3-4-3. Similarly, click under the *Package Pin* column across LD7 row to see a drop-down box appear. Type **U** (for Nexys4) in the field to jump to Uxx or Vxx pins, scroll-down until you see U6 (Nexys4) 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) pin location and the LVCMOS33 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]]
```



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). 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.

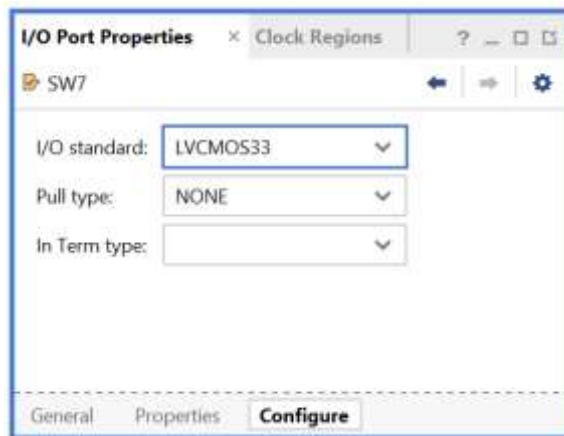


Figure. Assigning I/O standard through the I/O Port Properties form

3-4-5. Select **File > Constraints> Save** and click **OK** to save the constraints in the **nexys4.xdc** file.

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

Note that the constraints are updated in **.xdc** file under the **lab_3** project directory and not under the **sources** directory.

Simulate the Design using the XSim Simulator

Step 4

4-1. Add the lab_3_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 **C:\Xilinx_trn\IP\sources** folder and select **lab_3_tb.v** and click **OK**.

4-1-5. Click **Finish**.

4-1-6. Select the *Sources* tab and expand the *Simulation Sources* group.

The **lab_3_tb.v** file is added under the *Simulation Sources* group, and **lab_3_wrapper.v** is automatically placed in its hierarchy as a **tut1** instance.

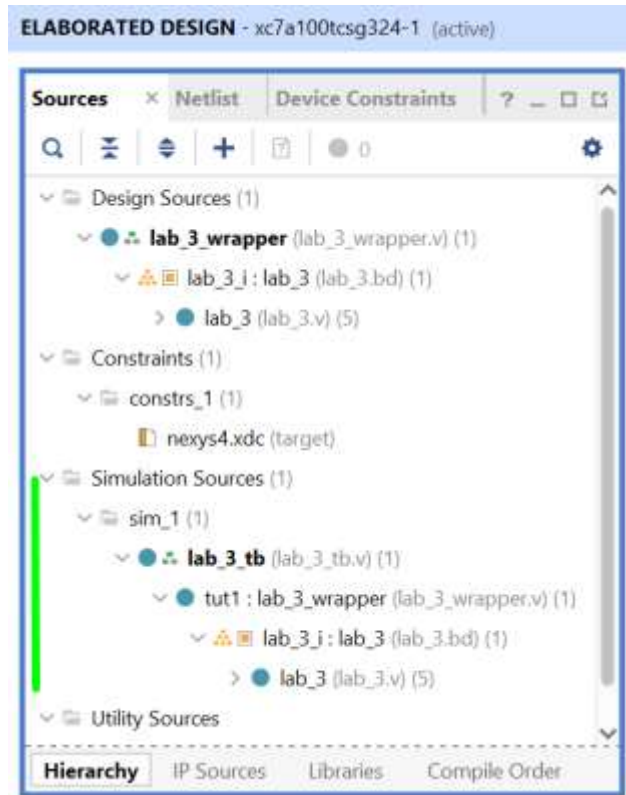


Figure. Simulation Sources hierarchy

4-1-7.

4-1-8. Double-click on the **lab_3_tb** in the *Sources* pane to view its contents.

```

1  `timescale 1ns / 1ps
2  //////////////////////////////////////////////////
3  // Module Name: lab_3_tb
4  //////////////////////////////////////////////////
5  module lab_3_tb(
6  );
7      reg [7:0] switches;
8      wire [7:0] leds;
9      reg [7:0] e_led;
10     integer i;
11     lab_3_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         .SW0(switches[0]),
21         .SW1(switches[1]),
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);
46             if(leds == e_led)
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

```

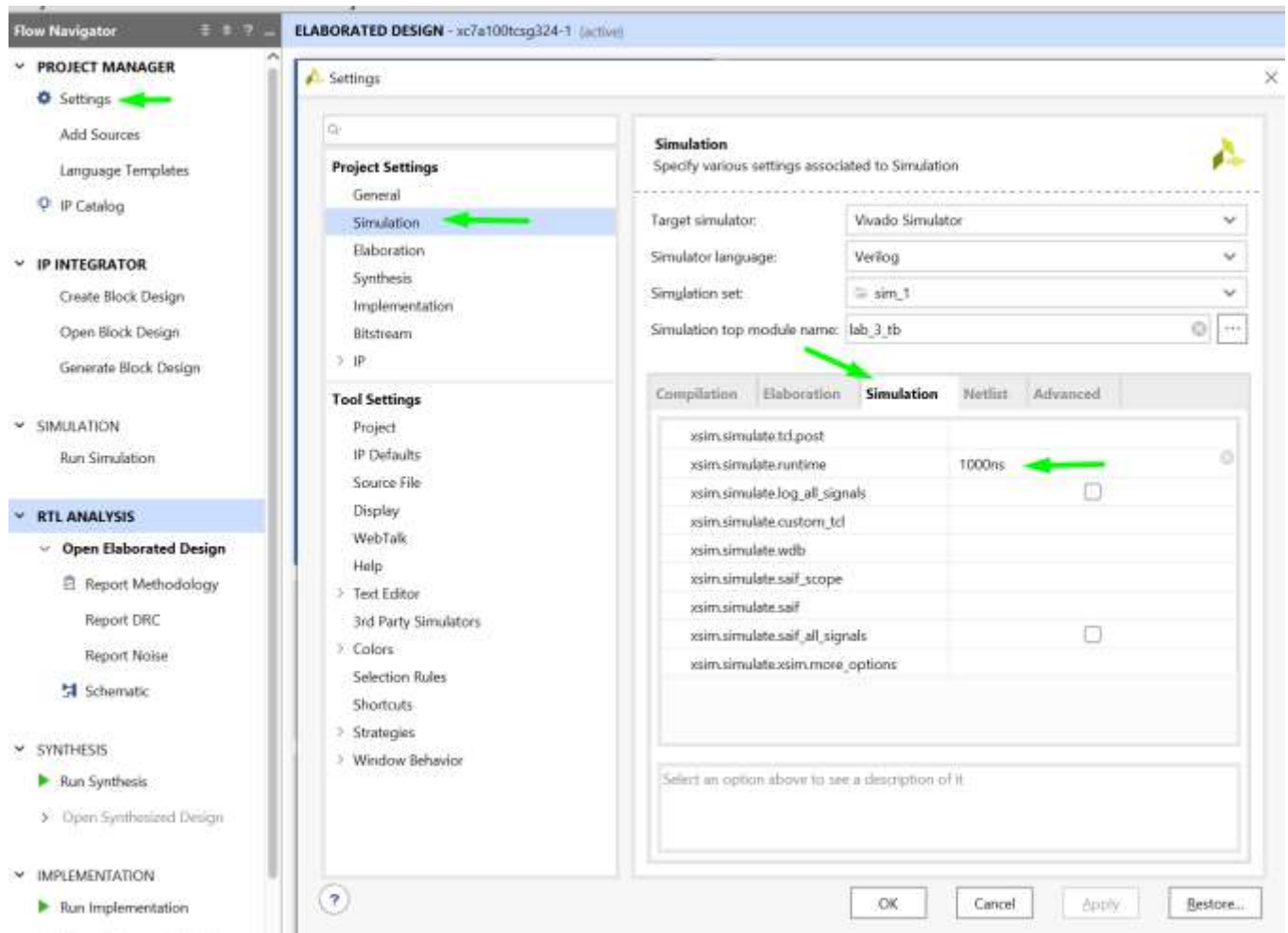
Figure 26. The self-checking testbench

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** under the *Project Manager* tasks > Settings 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).

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

You will see a simulator output similar to the one shown below.

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.

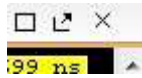


Figure. Float Button



Figure. Dock Window Button

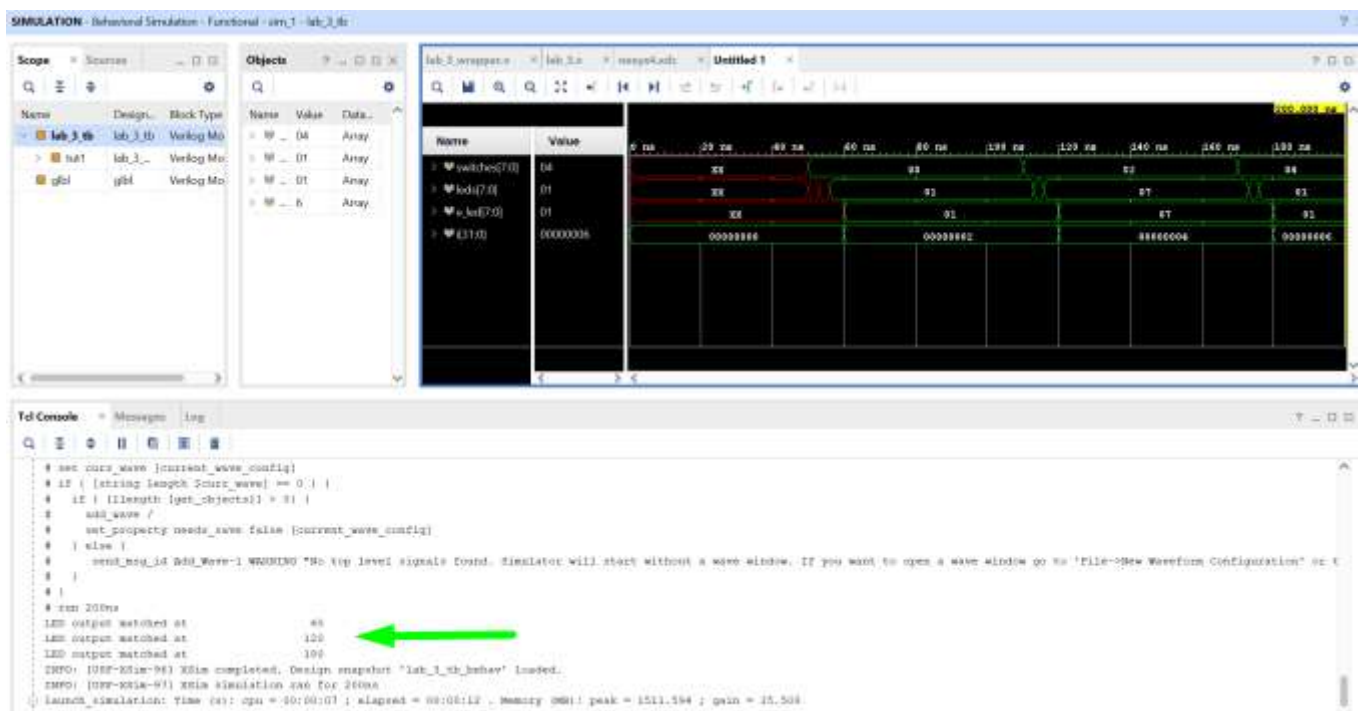


Figure. Simulator output

You will see four main views: (i) *Scopes*, where the testbench hierarchy as well as gbl 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.

4-3. Change display format if desired.

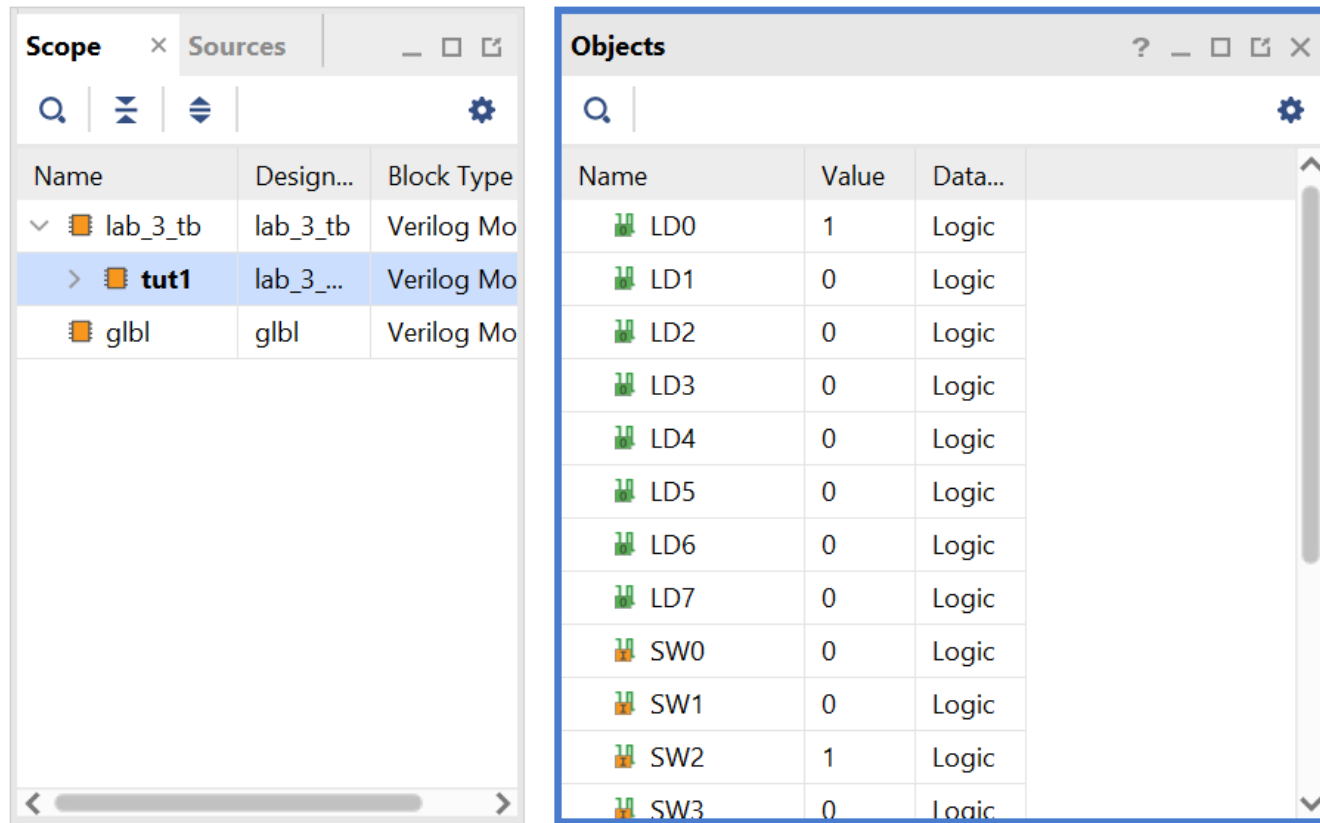
4-3-1. Select **i[31:0]** in the waveform window, right-click, select *Radix*, and then select *Unsigned Decimal* to view the for-loop index in *integer* form. Similarly, change the radix of **switches[7:0]** to *Hexadecimal*. Leave the **leds[7:0]** and **e_led[7:0]** radix to *binary* as we want to 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 **lab_3_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.

SIMULATION - Behavioral Simulation - Functional - sim_1 - lab_3_tb



The **Scopes** window displays the following table:

Name	Design...	Block Type
lab_3_tb	lab_3_tb	Verilog Mo
> tut1	lab_3_...	Verilog Mo
gbl	gbl	Verilog Mo

The **Objects** window displays the following table:

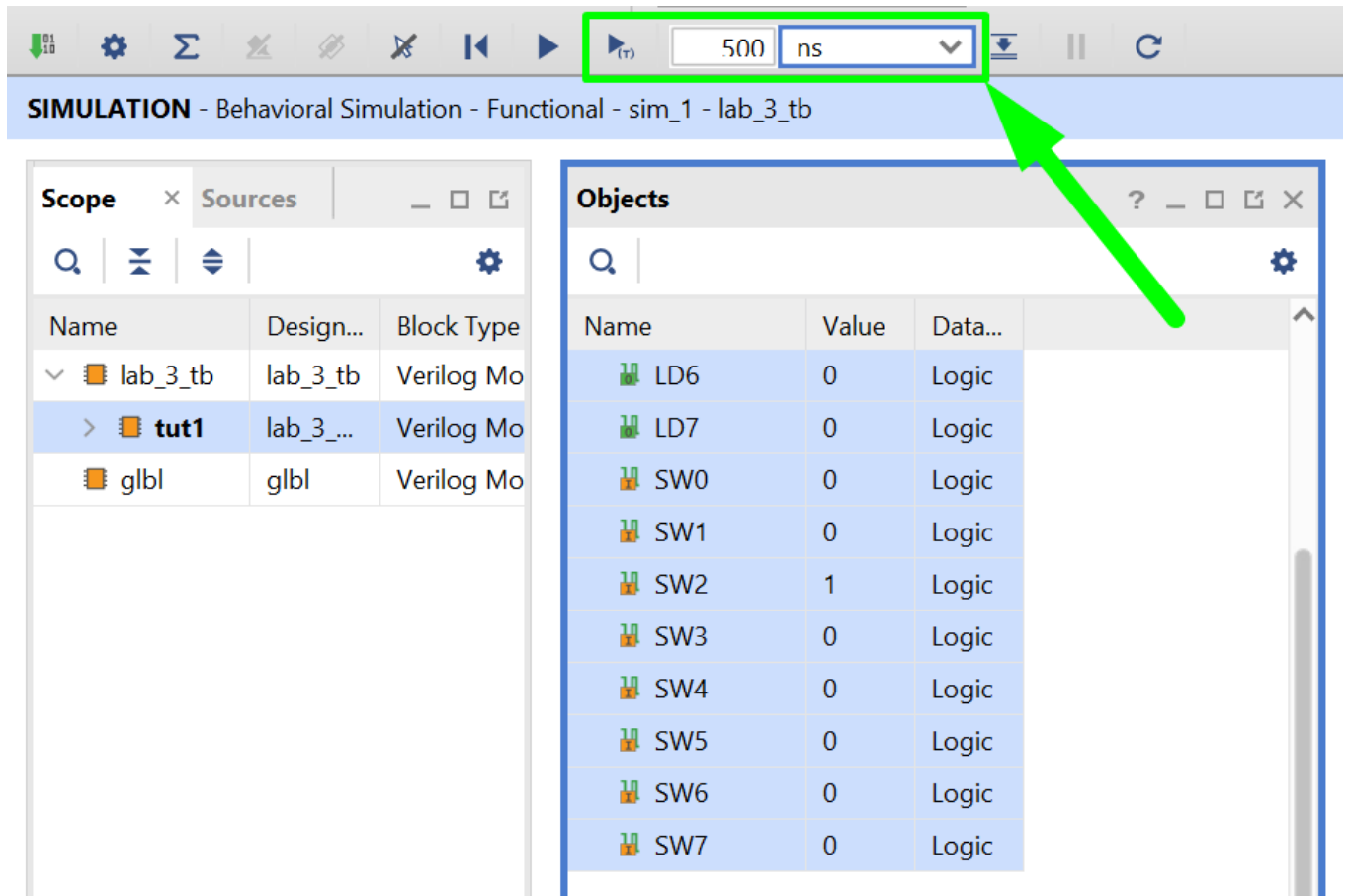
Name	Value	Data...
LD0	1	Logic
LD1	0	Logic
LD2	0	Logic
LD3	0	Logic
LD4	0	Logic
LD5	0	Logic
LD6	0	Logic
LD7	0	Logic
SW0	0	Logic
SW1	0	Logic
SW2	1	Logic
SW3	0	Logic

Figure. Selecting lower-level signals

4-4-2. Select **SW*** and **LD*** 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 (▶) button.

The simulation will run for an additional 500 ns.



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

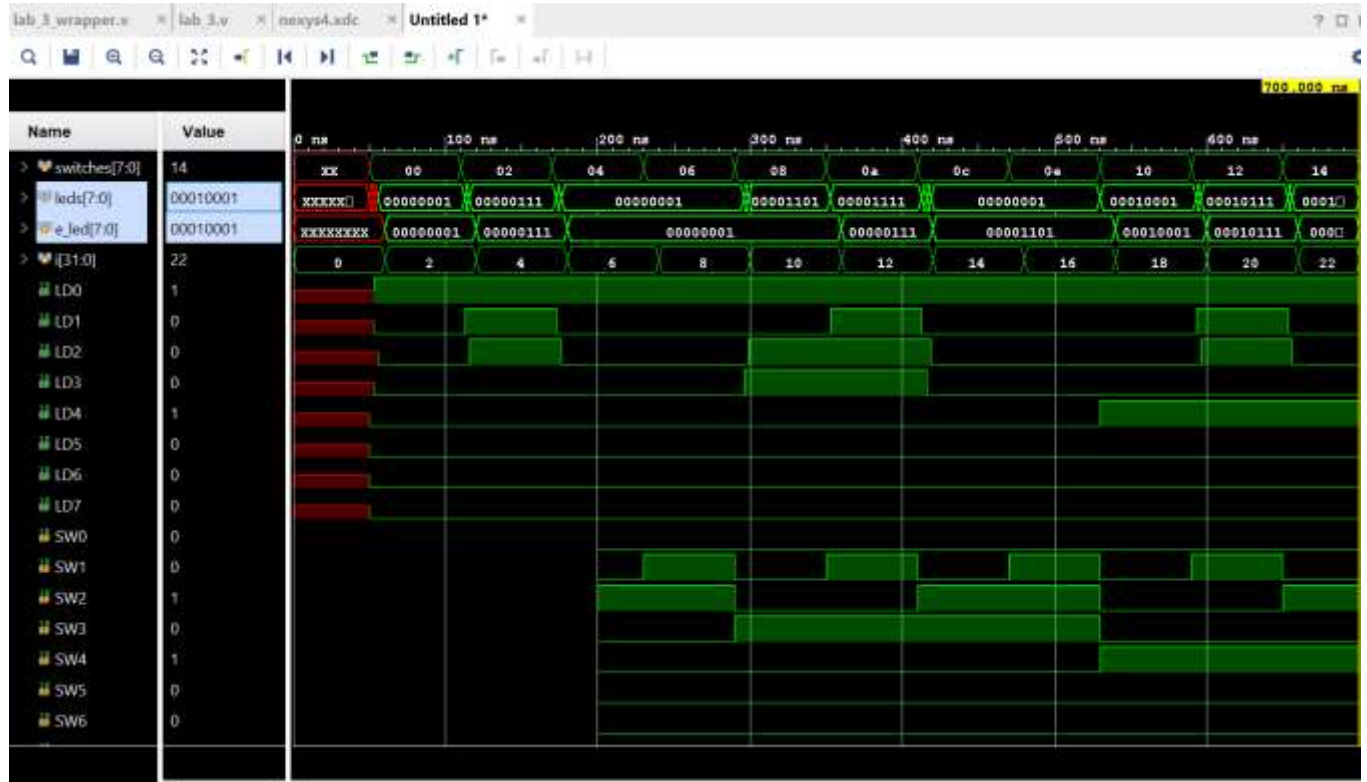


Figure 32. Running simulation for additional 500 ns

4-4-5. Close the simulator by selecting **File > Close Simulation**.

4-4-6. Click **OK** and then click **Discard** 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 lab_3_wrapper.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.

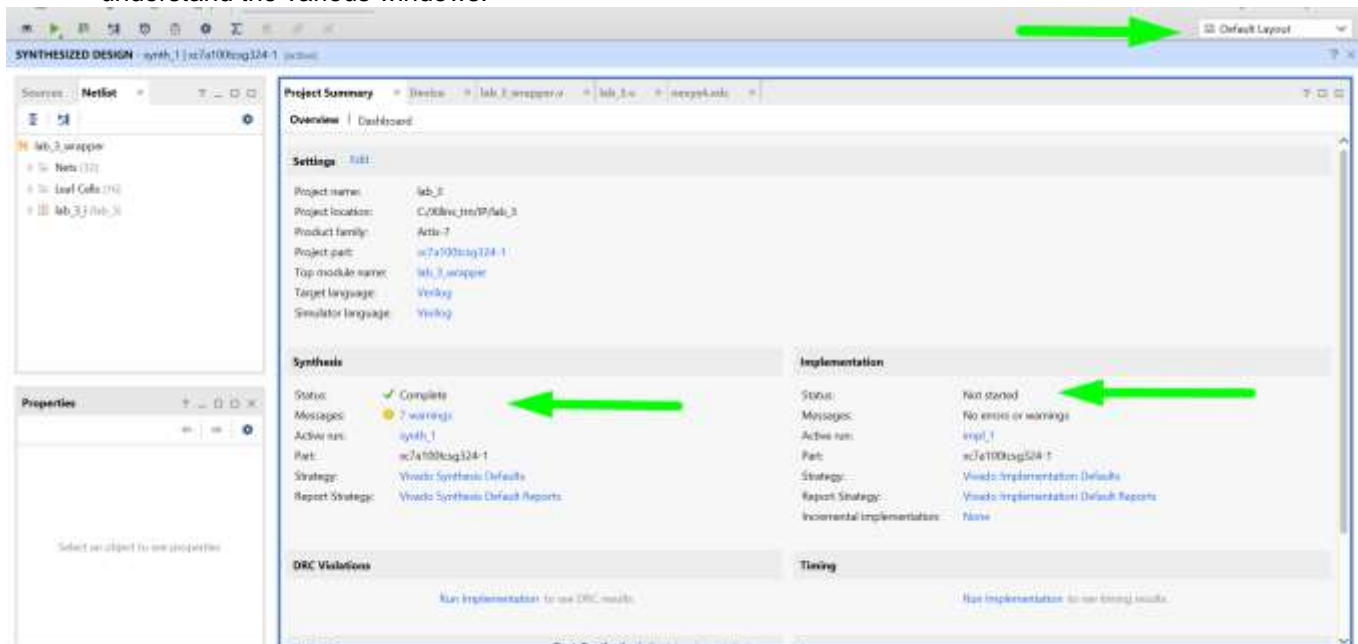


Figure. Project Summary view

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 **Utilization => 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.

Implement the Design

Step 6

- 6-1. Implement the design with the Vivado Implementation Defaults (settings and analyze the Project Summary output.**

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.**

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.