Práctica 1

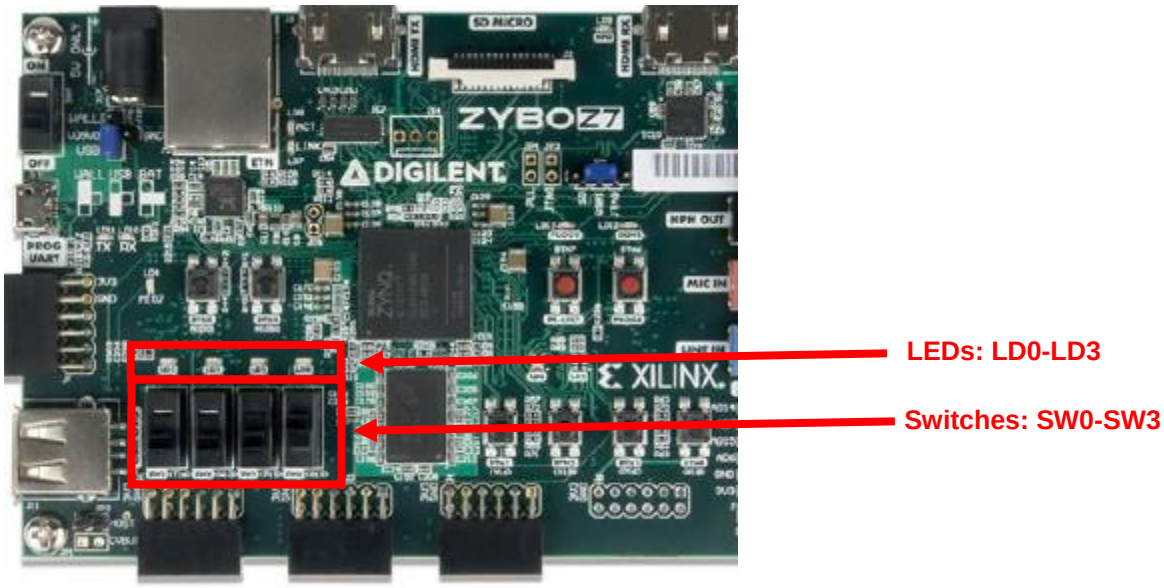
Tutorial

Objetivo

Utilizando un diseño especialmente simple, seguir con él un flujo completo con la herramienta.

Circuito utilizado

Se trata de una lógica muy simple situada entre los interruptores (SW0-SW3) y los leds (LD0-LD3) de la placa Zybo.



Guía para la realización

En las siguientes páginas se reproduce un tutorial del programa universitario de Xilinx, adaptado para esta asignatura.

Antes de continuar:

- Crear una carpeta *DIE* y bajo ella una carpeta *P1*. Será aquí donde se trabaje para esta práctica.
- Bajar de Moodle el archivo *sources.zip* y extraerlo como un subdirectorio *sources* de *P1*.

Vivado Design Flow

Introduction

This lab guides you through the process of using Vivado IDE to create a simple HDL design targeting the ZYBO board. You will simulate, synthesize, and implement the design with default settings. Finally, you will generate the bitstream and download it in to the hardware to verify the design functionality

Objectives

After completing this lab, you will be able to:

- Create a Vivado project sourcing HDL model(s) and targeting a specific FPGA device located on the ZYBO board
- Use the provided Xilinx Design Constraint (XDC) file to constrain the pin locations
- Simulate the design using the Vivado simulator
- Synthesize and implement the design
- Generate the bitstream
- Configure the FPGA using the generated bitstream and verify the functionality

Procedure

This lab 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 lab.

Design Description

The design takes its inputs from the slide switches, operates on them and outputs the results to the LEDs, as shown in **Figure 1**.

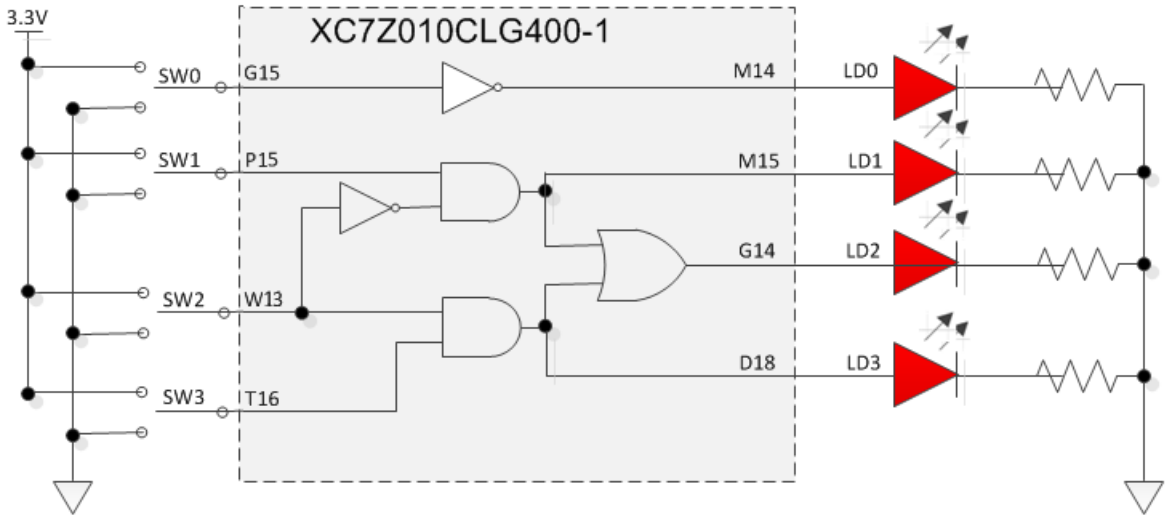
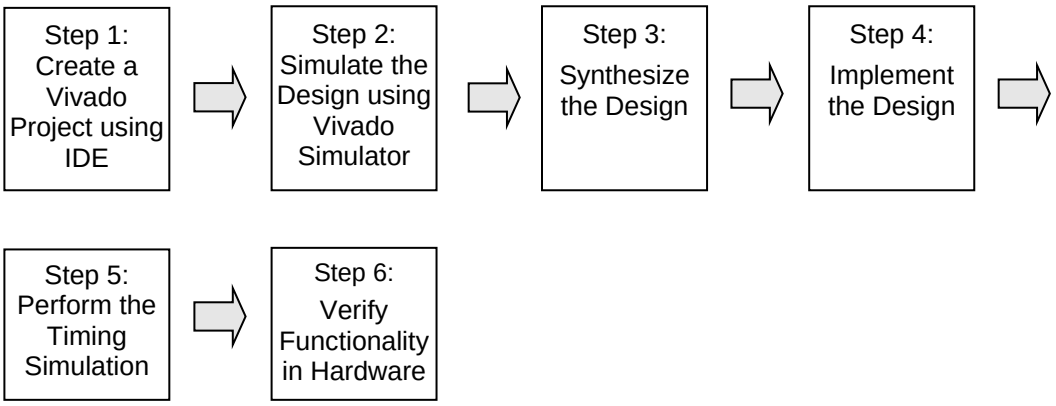


Figure 1. The Completed Design

General Flow



Create a Vivado Project using IDE — Step 1

1-1. Launch Vivado and create a project targeting the XC7Z010CLG400-1 device and using the VHDL language. Use the provided lab1.vhd and lab1.xdc files from the *sources\lab1* directory.

1-1-1. Open Vivado by double clicking in the **Vivado 2020.1** icon.

1-1-2. Click **Create 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 **[our_location]/P1**, and click **Select**.

1-1-4. Enter **lab1** in the *Project name* field. Make sure that the *Create Project Subdirectory* box is checked. Click **Next**.

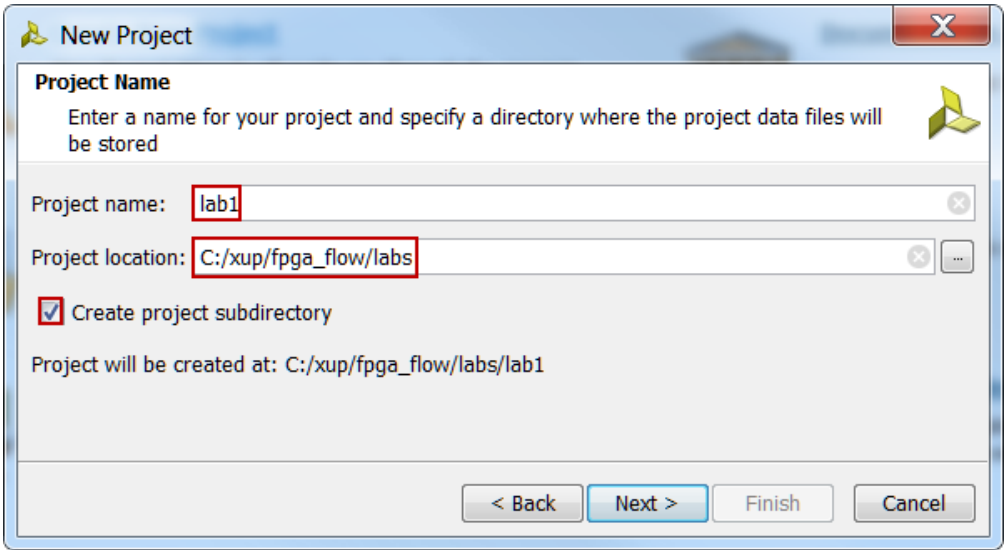


Figure 2. Project Name and Location entry (do not use the exact location shown)

1-1-5. Select **RTL Project** option in the *Project Type* form, and click **Next**.

1-1-6. Using the drop-down buttons, select **VHDL** as the *Target Language* and *Simulator Language* in the *Add Sources* form.

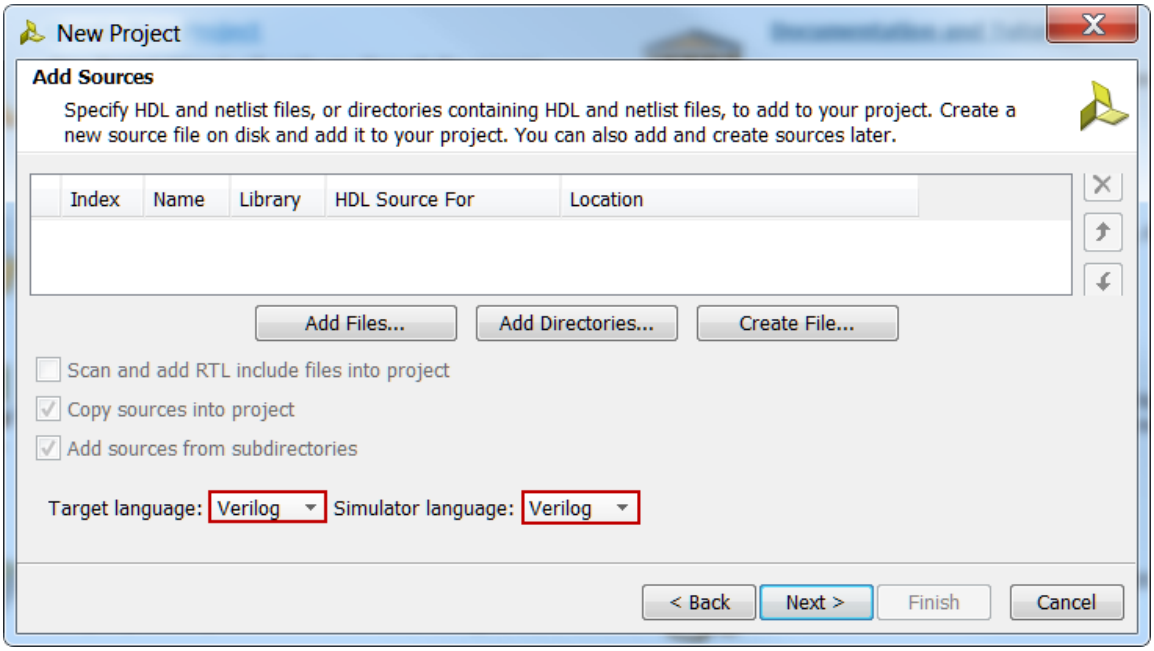


Figure 3. Selecting Target and Simulator language -> Change from Verilog to VHDL

1-1-7. Click on the **Add Files…** button, browse to the **sources** directory, select ***lab1.vhd***, click **Ok**. Select the ***Copy sources into project*** option and then click **Next** to get to the *Add Constraints* form.

1-1-8. Click on the **Add Files…** button, browse to the **sources** directory (if necessary), select *lab1.xdc* and click **OK**. Make sure ***Copy constraints files into project*** is selected and then click **Next.**

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 the board's schematic or the board's user guide.

1-1-9. In the *Default Part* form, using the **Parts** option and various drop-down fields of the **Filter** section, select the **XC7Z010CLG400-1** part: use *Family=Zynq-7000, Package=clg400, Speed=-1* to reduce the list. Click **Next**.

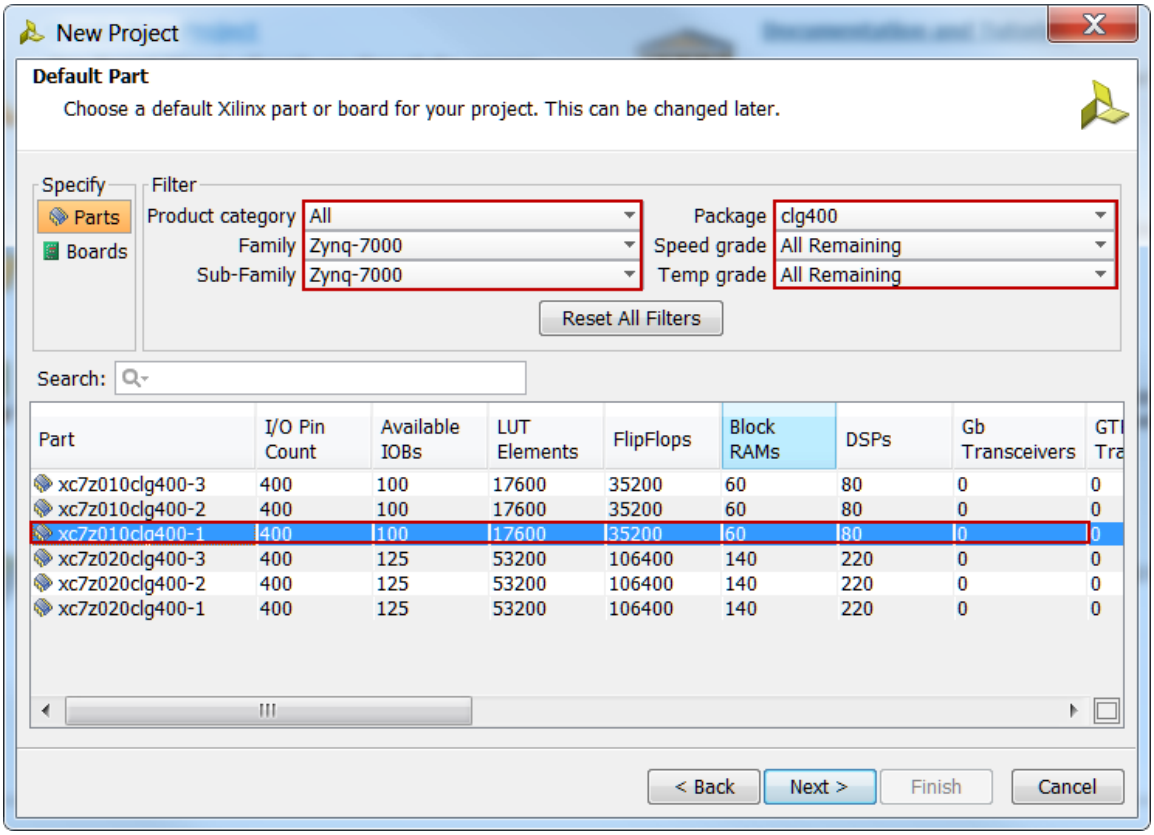


Figure 4. Part Selection

1-1-10. Click **Finish** to create the Vivado project.

Use the file system explorer and look at the **[your_drive_and_dir]/P1/lab1** directory. You will find that the **lab1.srcs** directory and the **lab1.xpr** (Vivado) project file have been created. Two directories, **constrs_1** and **sources_1**, are created under the **lab1.srcs** directory; deep down under them, the copied **lab1.xdc** (constraint) and **lab1.vhd** (source) files respectively are placed.

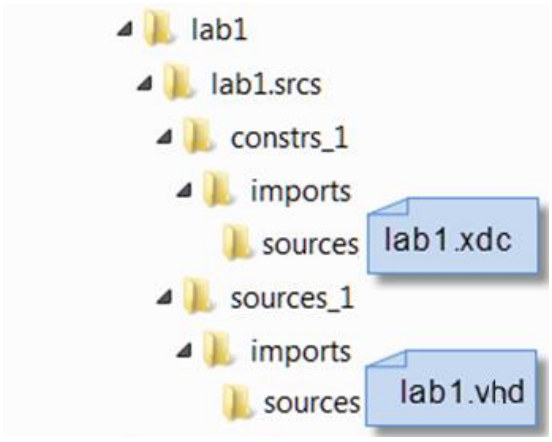


Figure 6. Generated directory structure

1-2. Open the lab1.vhd source and analyze the content.

1-2-1. In the *Sources* pane, double-click the **lab1.vhd** entry to open the file in text mode.

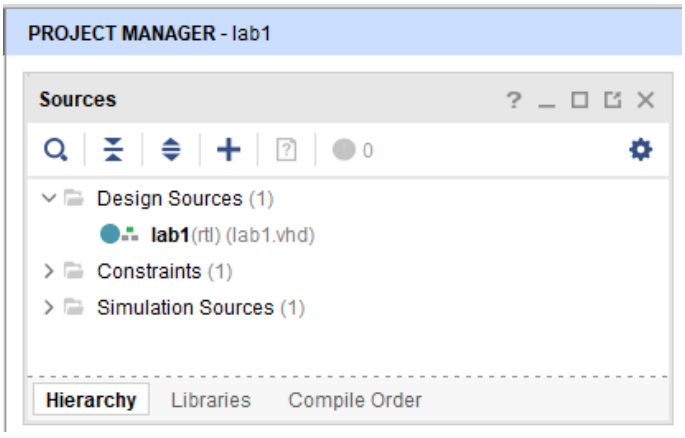


Figure 7. Opening the source file

1-2-2. **Read and understand** the existing, incomplete code.

1-2-3. **Complete the architecture** in order to implement the logic shown in Figure 1.

1-3. Open the lab1.xdc source and analyze the content.

1-3-1. In the *Sources* pane, expand the *Constraints* folder and double-click the **lab1.xdc** entry to open the file in text mode.

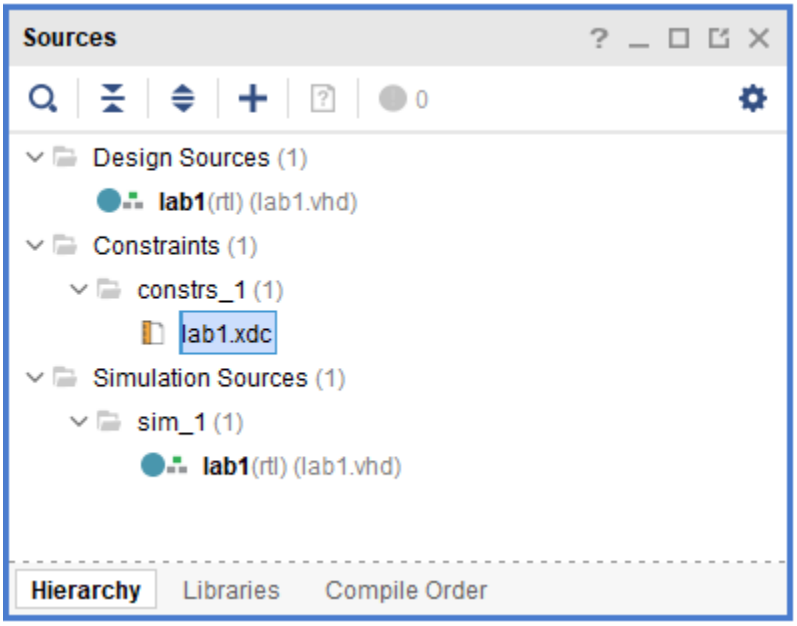


Figure 8. Opening the constraint file

1-3-2. Lines 5-12 define the pin locations of the input switches [3:0] and lines 17-24 define the pin locations of the output LEDs [3:0].

Simulate the Design using the Vivado Simulator Step 2

2-1. Add the lab1_tb.v testbench file.

2-1-1. Click **Add Sources** under the *Project Manager* tasks of the *Flow Navigator* pane.

Figure 10. Add Sources

2-1-2. Select the *Add or Create Simulation Sources* option and click **Next**.

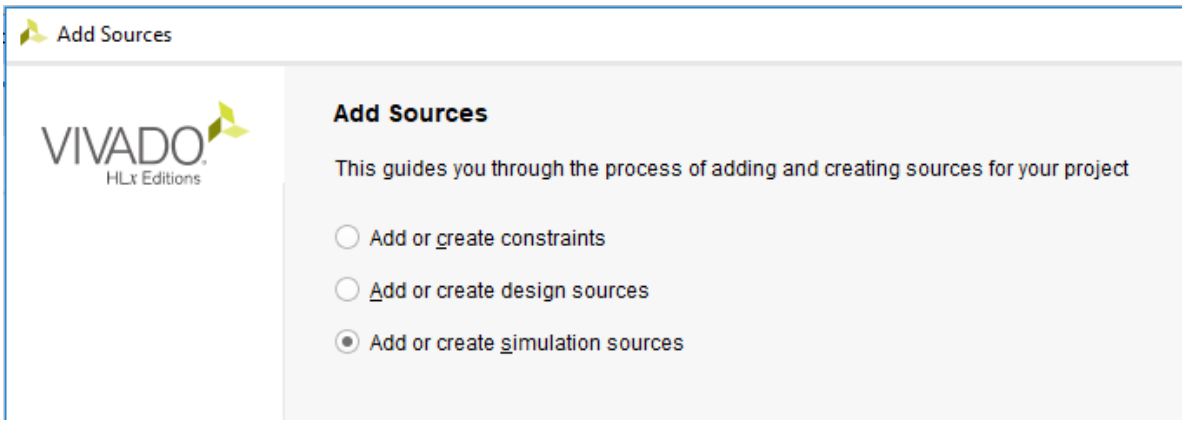


Figure 11. Selecting Simulation Sources option

2-1-3. In the *Add or Create Simulation Sources* form, click the **Add Files…** button.

2-1-4. Browse to the **sources** folder, select *lab1_tb.vhd* and click **OK**.

2-1-5. Make sure ***Copy sources into project*** is selected and Click **Finish**.

2-1-6. Select the *Sources* tab and expand the *Simulation Sources* group.

The lab1_tb.vhd file is added under the *Simulation Sources* group, and **lab1.vhd** is automatically placed in its hierarchy as a dut instance.

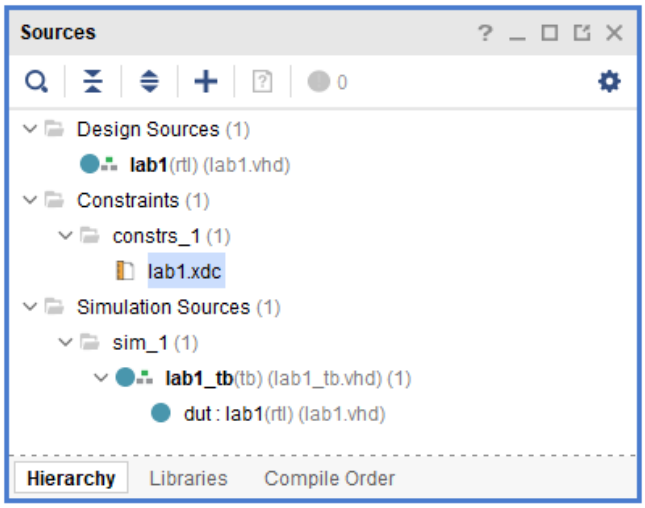


Figure 12. Simulation Sources hierarchy

2-1-7. 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 lab1.srcs directory, and that a copy of lab1_tb.vhd is placed under **lab1.srcs > sim_1 > imports > sources**.

2-1-8. Double-click on the **lab1_tb** in the *Sources* pane to view its contents.

2-1-9. **Read and understand** the existing, complete code.

2-2. Simulate the design for 50 ns using the Vivado simulator.

2-2-1. Select **Settings** under the *Project Manager* tasks of the *Flow Navigator* pan, then **Simulation**.

2-2-2. Select the **Simulation** tab, and set the **Simulation Run Time** value to 50 ns and click **OK** (see below).

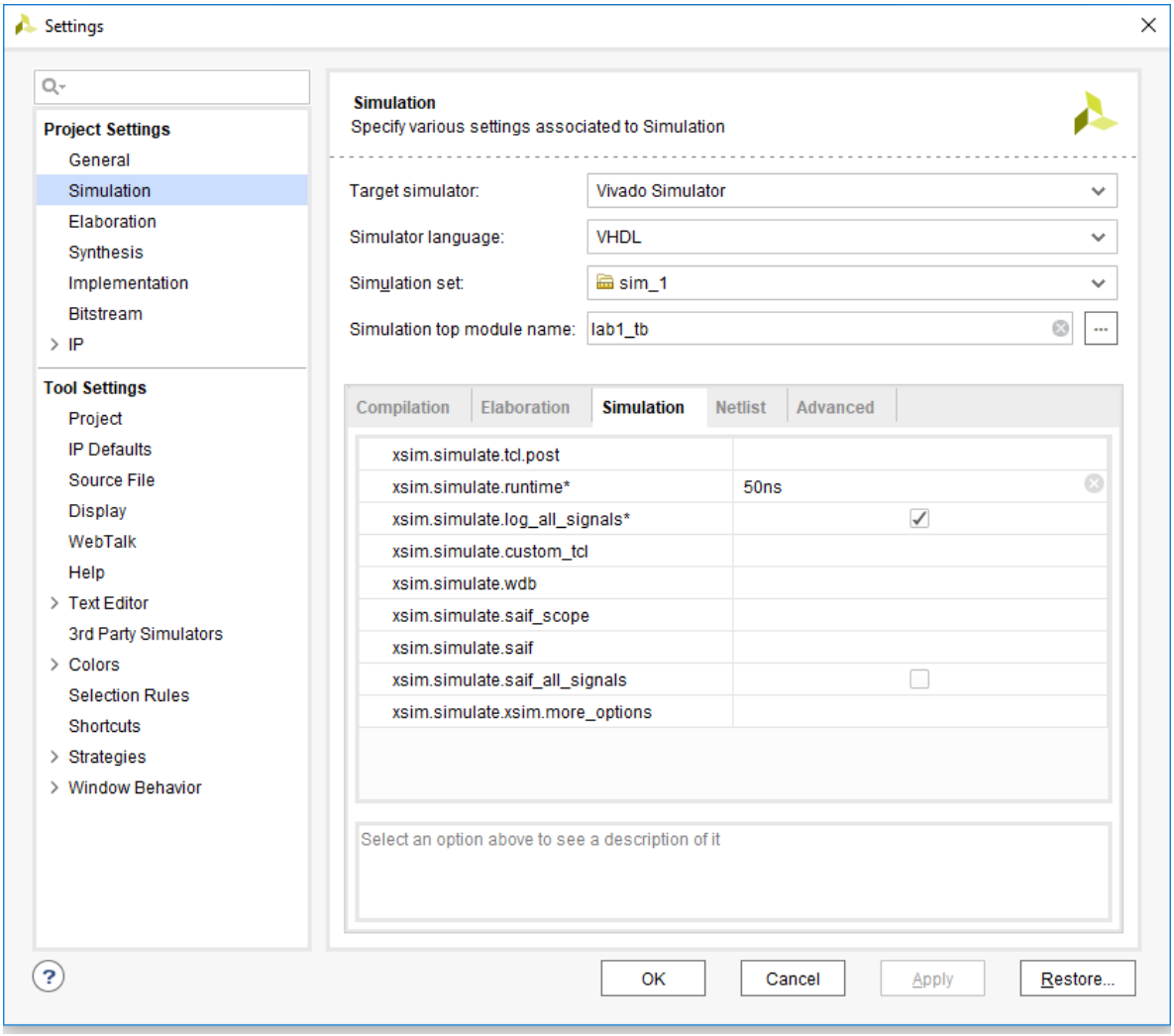


Figure 14. Setting simulation run time

2-2-3. Click on **Run Simulation** under the **SIMULATION** tasks of the *Flow Navigator* pane, then click on **Run Behavioral Simulation**.

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

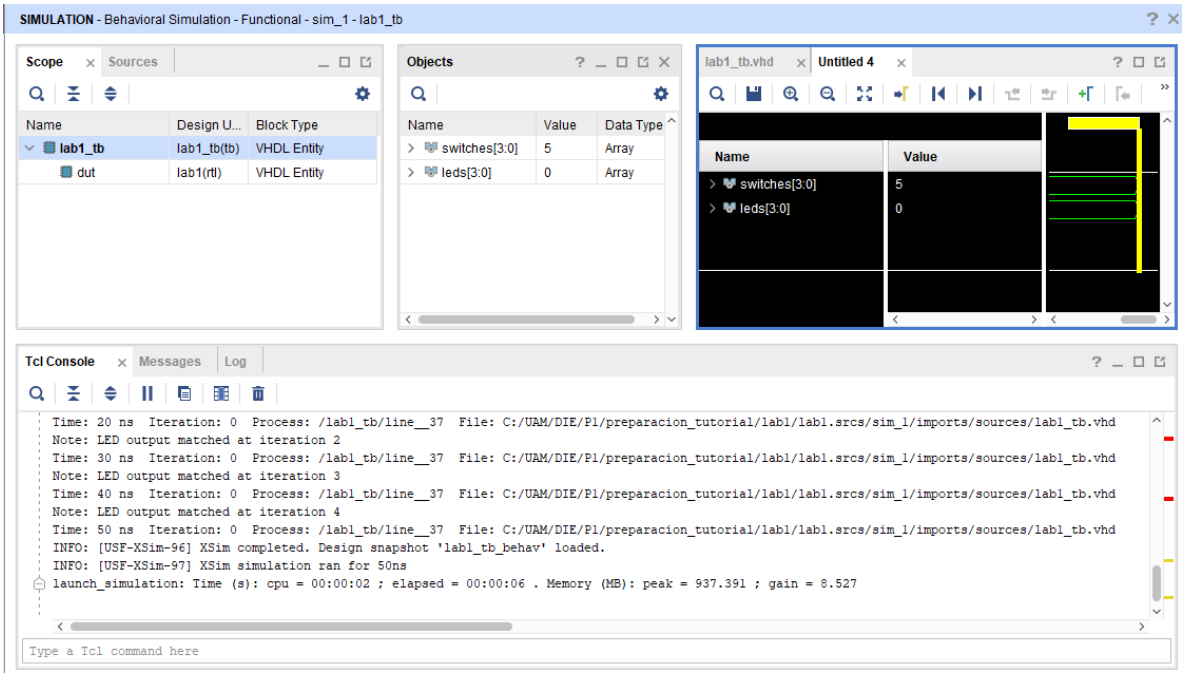


Figure 15. Simulator output

You will see four main views: (i) *Scope,* where the testbench hierarchy is 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 **lab1.sim** directory is created under the **lab1** directory, along with several lower-level directories.

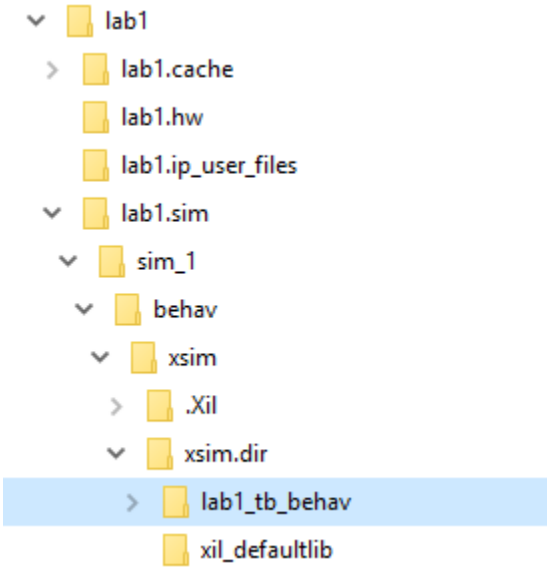


Figure 16. Directory structure after running behavioral simulation

2-2-4. Click on the maximize button of the waveform window ( ). Then click on the *Zoom Fit* button ( ) 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.

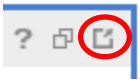


Figure 16. Float Button

Figure 17. Dock Window Button

2-3. Change display format if desired.

2-3-1. Select **switches[3:0]** in the waveform window, right-click, select *Radix*, and then select *Unsigned Decimal* to view this signal in an integer form. Now, using the shift or control key, select both **switches[3:0]** and **leds[3:0]** and change the radix to *binary* as we want to see each output bit.

2-4. Add more signals to monitor the lower-level signals and continue to run the simulation.

2-4-1. De-maximize the waveform window if necessary. Expand the **lab1_tb** instance, if necessary, in the *Scope* window and select the **dut** instance.

The swt[3:0], led[3:0] and ledSig[3:0] signals will be displayed in the *Objects* window.

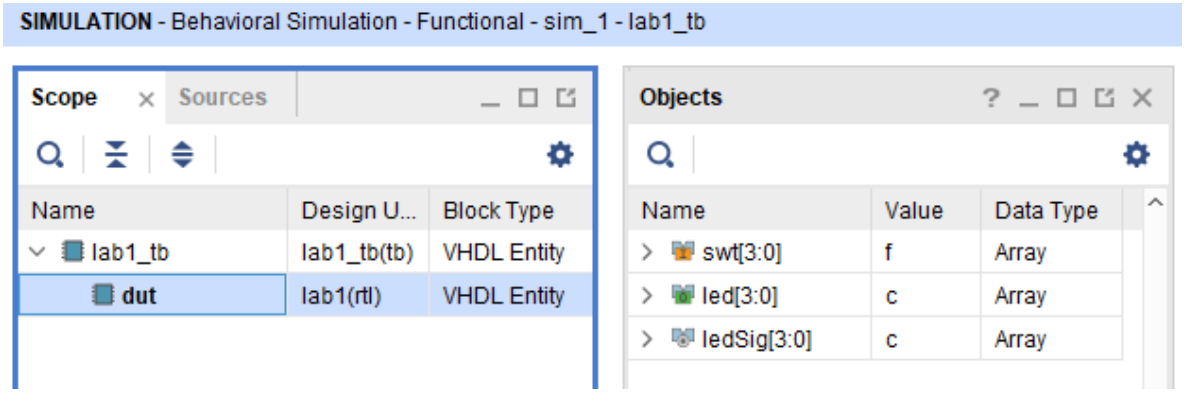
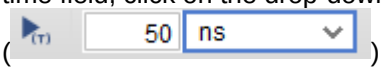


Figure 18. Selecting lower-level signals

2-4-2. Select **swt[3:0]** and **led[3:0],** right-click and select Add to Wave Window to include then into the waveform window to monitor those lower-level signals.

2-4-3. On the simulator tool buttons ribbon bar (below the main menu bar), type 50 in the simulation run time field, click on the drop-down button of the units field and select ns () to run for another 50 ns (total of 100 ns), and click on the () button.

The simulation will run for an additional 50 ns. Observe the new messages in the *Tcl Console* window.

Now click on the *Run All* button (). The simulation will run until the simulator has nothing else to do, i.e., until our simulation is finished.

2-4-4. Maximize the waveform window, click on the *Zoom Fit* button and observe the output.

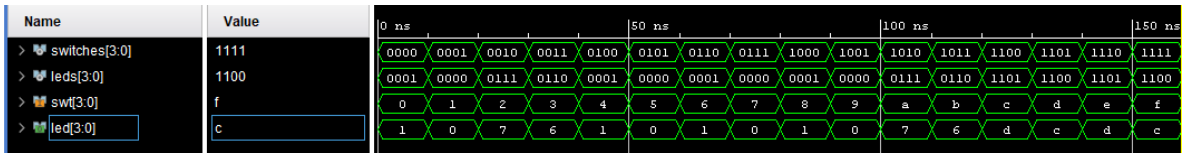


Figure 19. Running simulation until completion

Observe the Tcl Console window and see the output is being displayed as the testbench uses the *report* sentence.

```
Note: LED output matched at iteration 14
Time: 150 ns  Iteration: 0  Process: /lab1_tb/line__37  File:
Note: LED output matched at iteration 15
Time: 160 ns  Iteration: 0  Process: /lab1_tb/line__37  File:
```

Figure 20. Tcl Console output

2-4-5. Close the simulator by selecting **File > Close Simulation**.

2-4-6. Click **OK** and then click **Discard** to close it without saving the waveform.

Synthesize the Design — Step 3

3-1. Synthesize the design with the Vivado synthesis tool and analyze the Project Summary output.

3-1-1. Click on **Run Synthesis** under the *Synthesis* tasks of the *Flow Navigator* pane, then **OK**.

The synthesis process will be run on the lab1.vhd file (and all its hierarchical files if they existed). When the process is completed a *Synthesis Completed* dialog box with three options will be displayed.

3-1-2. 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.

3-1-3. Select the **Project Summary** tab and understand the various sections.

If you don't see the Project Summary tab then select **Layout > Default Layout,** or click the **Project Summary** icon **.**

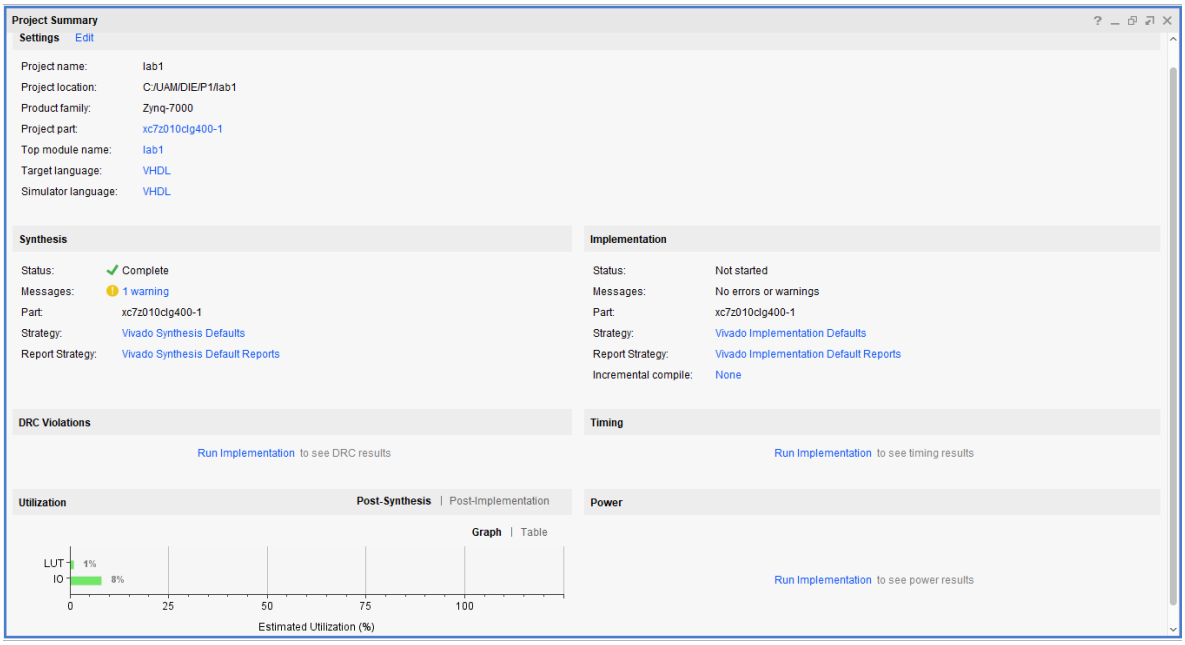
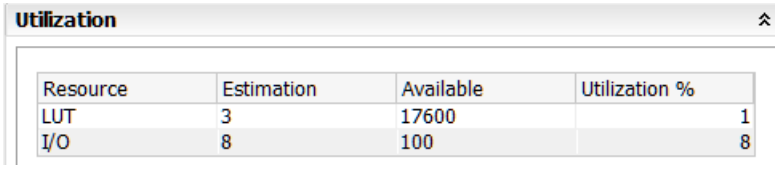


Figure 21. Project Summary view

3-1-4. Click on the **Table** tab under the **Utilization** section.

Notice that there are an estimated three LUTs and 8 IOs (4 input and 4 output) that are used.



Resource	Estimation	Available	Utilization %
LUT	3	17600	1
I/O	8	100	8

Figure 22. Resource utilization estimation summary

3-1-5. In The *Flow Navigator*, under *Synthesis* (expand *Open Synthesized Design* if necessary), click on **Schematic** to view the synthesized design in a schematic view.

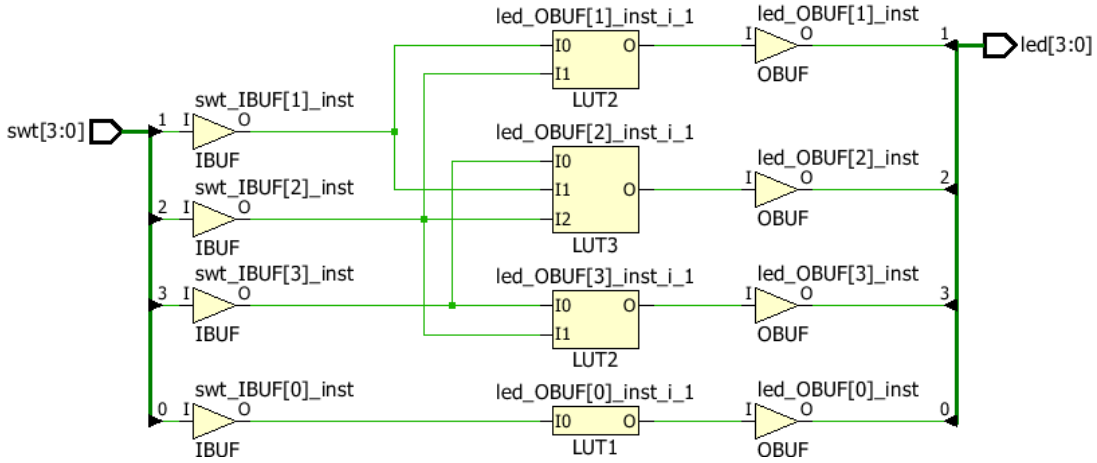


Figure 23. Synthesized design's schematic view

Notice that IBUFs and OBUFs are automatically instantiated (added) to the design as the input and output are buffered. The logical gates are implemented in LUTs (1 input is listed as LUT1, 2 input is listed as LUT2, and 3 input is listed as LUT3). You can see that there is not a one to one relationship between the basic logic elements we described in our RTL and how the circuit is implemented, but the result is the same.

Using the file system explorer, verify that **lab1.runs** directory is created under **lab1**. Under the **runs** directory, **synth_1** directory is created which holds several files related to synthesis.

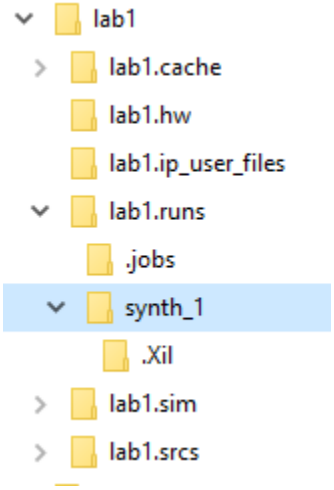


Figure 24. Directory structure after synthesizing the design

Implement the Design Step 4

4-1. Implement the design with the Vivado Implementation default settings and analyze the Project Summary output.

4-1-1. Click on **Run Implementation** under the *Implementation* tasks of the *Flow Navigator* pane, then **OK**.

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

4-1-2. Select **Open implemented design** and click **OK** as we want to look at the implemented design in a Device view tab.

4-1-3. Click **Yes,** if prompted, to close the synthesized design.

The implemented design will be opened.

4-1-4. In the *Netlist* pane, select one of the nets (e.g. swt_IBUF[0]) and notice that the net is displayed in the X1Y1 clock region in the Device view tab (you may have to zoom in to see it).

4-1-5. If it is not displayed, click the *Routing Resources* icon  to show routing resources. Take a look to the two end points of the net.

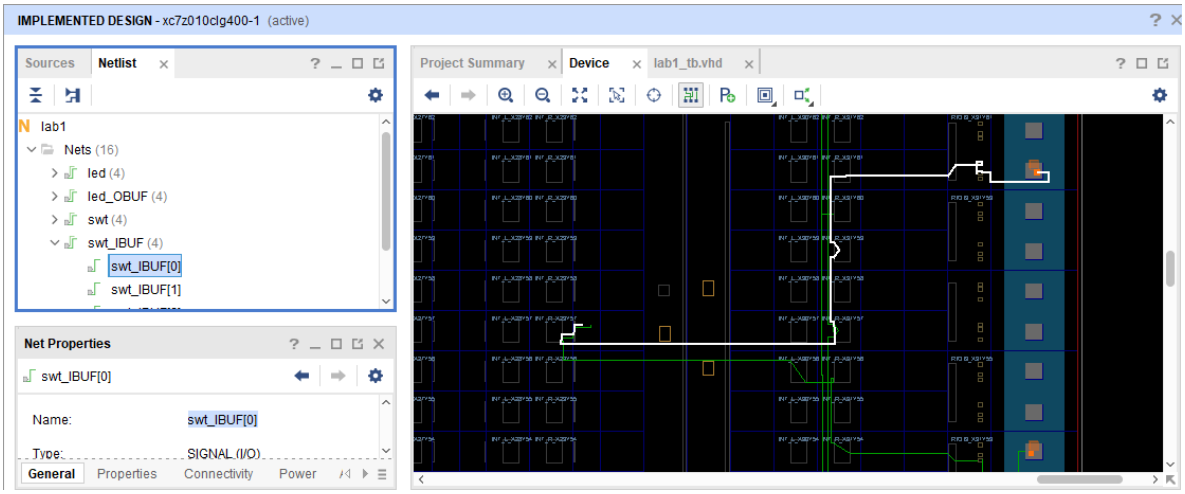


Figure 25. Viewing implemented design

4-1-6. 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.

Select the Post-Implementation tab of *Utilization* if not already selected and the *Table* view.

Notice that the actual resource utilization is three LUTs and 8 IOs.

You can also see that there is no *Timing* information, since no timing constraints were defined for this combinational design.

Using the file system explorer, verify that **impl_1** directory is created at the same level as **synth_1** under the **lab1.runs** directory. The **impl_1** directory contains several files including the implementation report files.

4-1-7. In Vivado, select the **Reports** tab in the bottom panel (if not visible, click *Window* in the menu bar and select **Reports**), and double-click on the utilization report entry under the *Place Design* section. The report will be displayed in the auxiliary view pane showing resource utilization. Note that since the design is combinational no registers are used.

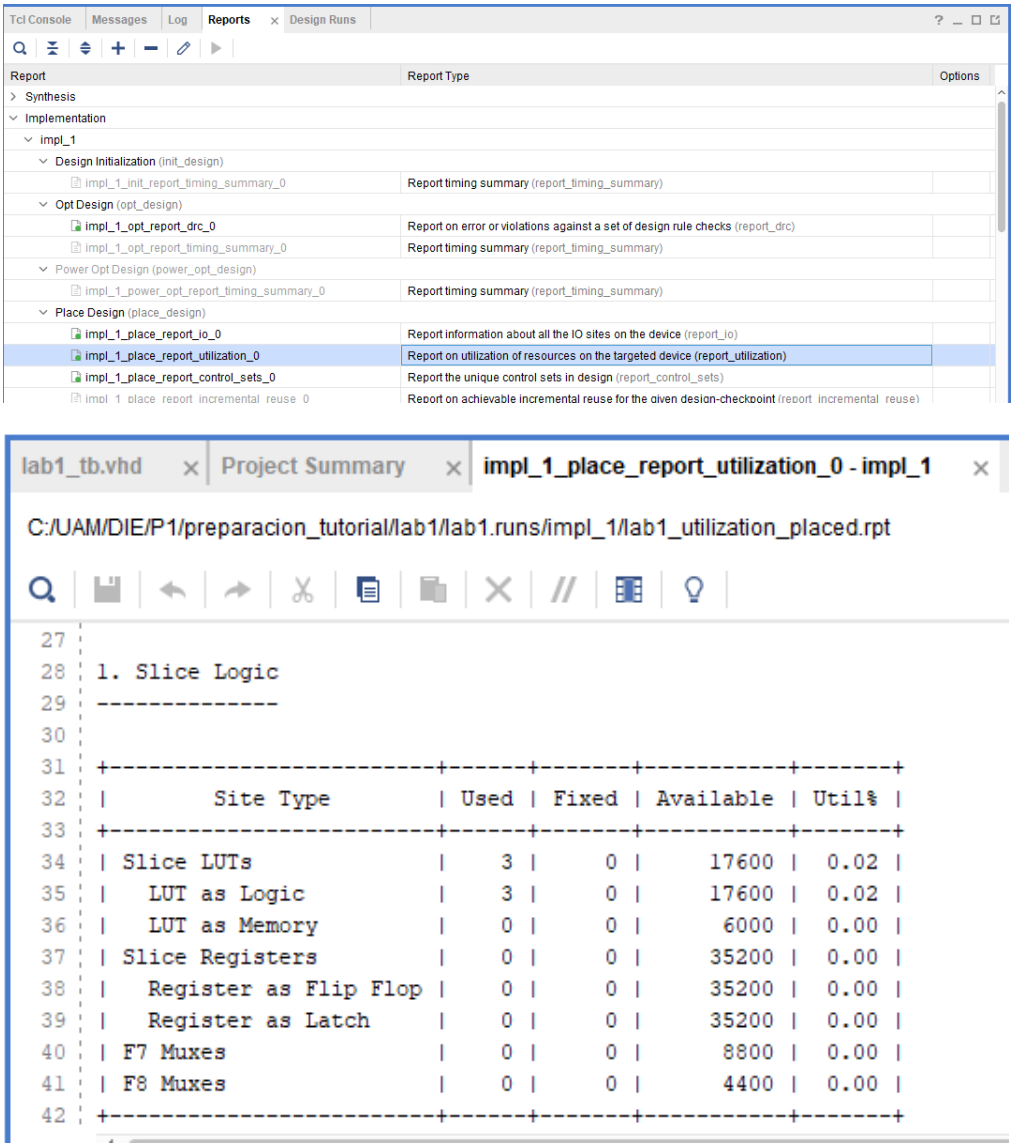


Figure 27. Viewing utilization report

Perform Timing Simulation Step 5

5-1. Run a timing simulation.

5-1-1. Select **Run Simulation > Run Post-Implementation Timing Simulation** process under the *Simulation* tasks of the *Flow Navigator* pane. Disregard a possible warning message.

The Vivado simulator will be launched using the implemented design and **lab1_tb** as the top-level module.

Using the Windows Explorer, verify that **timing** directory is created under the **lab1.sim > sim_1 > impl** directory. The **timing** directory contains generated files to run the timing simulation.

5-1-2. Click on the **Zoom Fit** button to see the waveform window from 0 to 50 ns.

5-1-3. Right-click at 20 ns (where the **switches** input is set to 2) and select **Markers > Add Marker**.

5-1-4. Similarly, right-click and add a marker at the moment where the **leds** changes to its final value for this input, 7. You can also add a marker by clicking on the Add Marker button ( ).

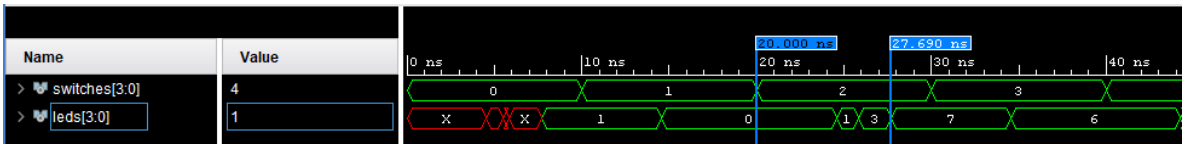


Figure 28. Timing simulation output

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 8 ns.

5-1-5. Close the simulator by selecting **File > Close Simulation** without saving any changes.

Generate the Bitstream and Verify Functionality — Step 6

6-1. Connect the board and power it ON. Generate the bitstream, open a hardware session, and program the FPGA.

6-1-1. Make sure that the Micro-USB cable is connected to the PROG UART connector (next to the power ON/OFF switch).

6-1-2. Make sure that the JP7 is set to select USB power. Connect the USB cable to the PC.

Figure 29. Board connection

6-1-3. Power **ON** the switch on the board.

6-1-4. Click on the **Generate Bitstream** entry under the *Program and Debug* tasks of the *Flow Navigator* pane, then **OK**.

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.

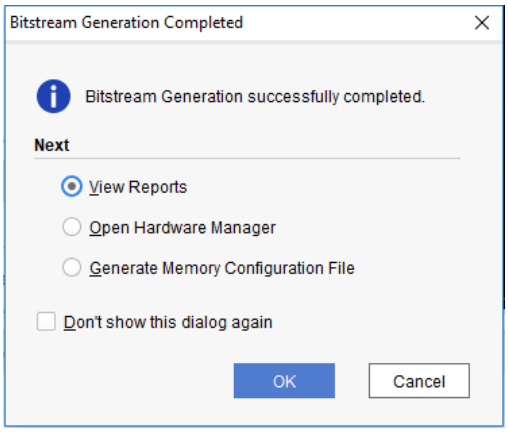


Figure 30. Bitstream generation

This process will have generated a **lab1.bit** file under **impl_1** directory in the **lab1.runs** directory.

6-1-5. Select the *Open Hardware Manager* option and click **OK**.

The Hardware Manager window will open indicating “unconnected” status.

6-1-6. Click on the **Open target** link, then *Auto Connect.*

You can also click on the **Open recent target** link if the board was already targeted before.

Figure 30. Opening new hardware target

6-1-7. The Hardware Session status changes from Unconnected to the server name and the FPGA device is shown. Also notice that the Status indicates that it is not programmed.

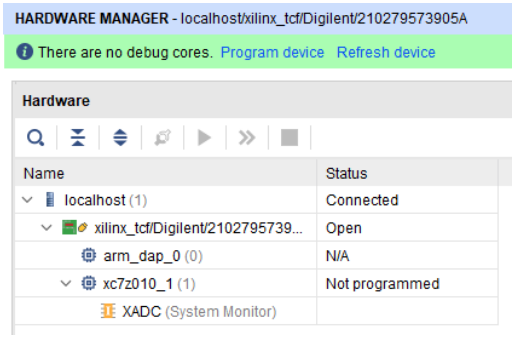


Figure 32. Opened hardware session

6-1-8. Select the device and verify that the lab1.bit is selected as the programming file in the *General* tab of the *Hardware Device Properties* pane.

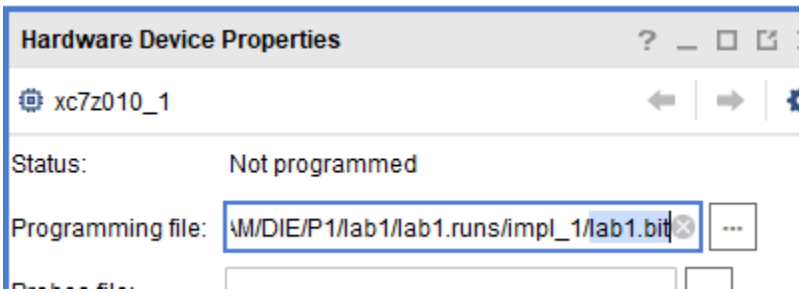


Figure 33. Programming file

6-1-9. **Right-click** on the device and select *Program Device…* or click on the *Program device* link to program the target FPGA device.

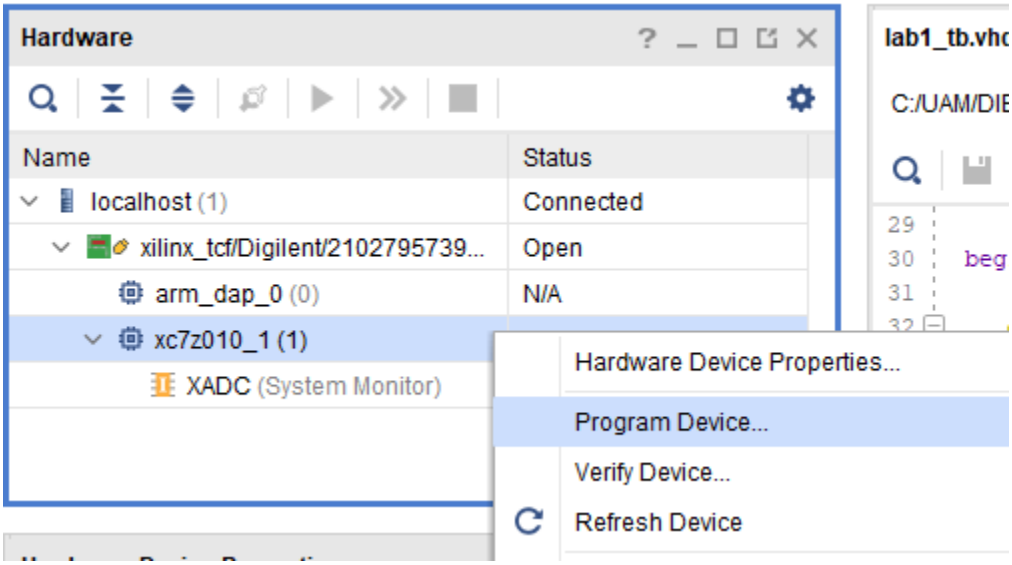


Figure 34. Selecting to program the FPGA

6-1-10. Click **Program** to program the FPGA.

The DONE light will light when the device is programmed. You may see some other LEDs lit depending on switch positions.

6-1-11. Verify the functionality by flipping switches and observing the output on the LEDs (Refer to the earlier logic diagram).

6-1-12. When satisfied, close the hardware session by selecting **File > Close Hardware Manager.**

6-1-13. Power **OFF** the board.

6-1-14. 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 supplied source files (HDL model and user constraint file). A behavioral simulation using the provided testbench 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.