# Lab 01: Behavior simulation with Vivado

# Objectives

After completing this lab, you will be able to:

- Create a Vivado project sourcing HDL model(s) and targeting the Zynq Ultrascale+ devices located on the ZCU106 boards.

- Use the provided Xilinx Design Constraint (XDC) file to constrain the pin locations.

- Simulate the design using the Vivado simulator.

# Steps

**Create a Vivado Project**

Launch Vivado and create an empty project targeting Zynq UltraScale+ ZCU106 Evaluation Platform, selecting Verilog as a target language. Use the provided lab1.v, and lab1\_zcu106.xdc (for ZCU106) files.

1. Open Vivado by selecting **Start > Xilinx Design Tools > Vivado 2021.2**.

2. Click **Create New Project** to start the wizard. You will see *Create A New Vivado Project* dialog box. Click **Next**.

3. Click the Browse button of the *Project location* field of the **New Project** form, browse to your desired location.

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

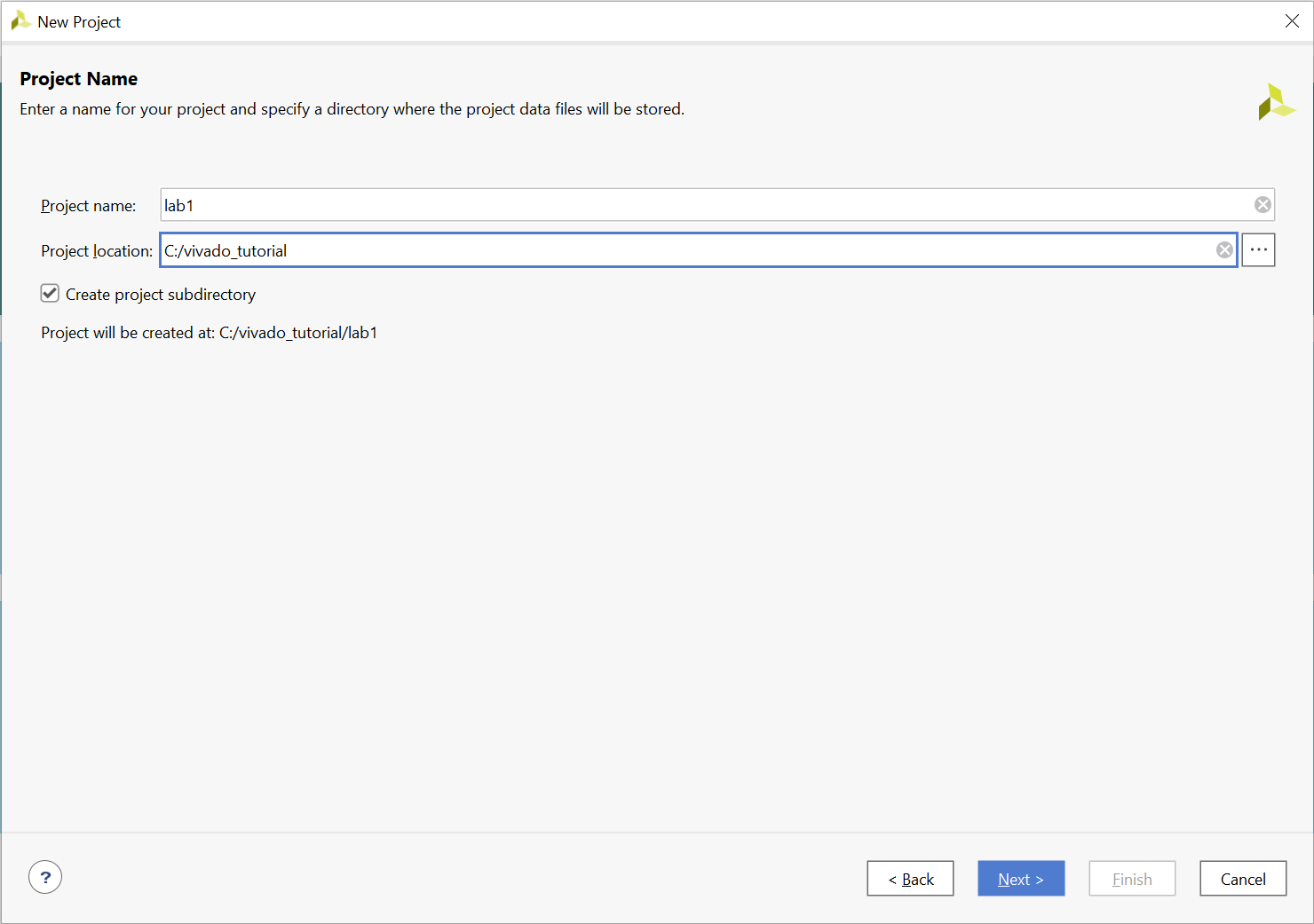


Figure Project Name and Location entry

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

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

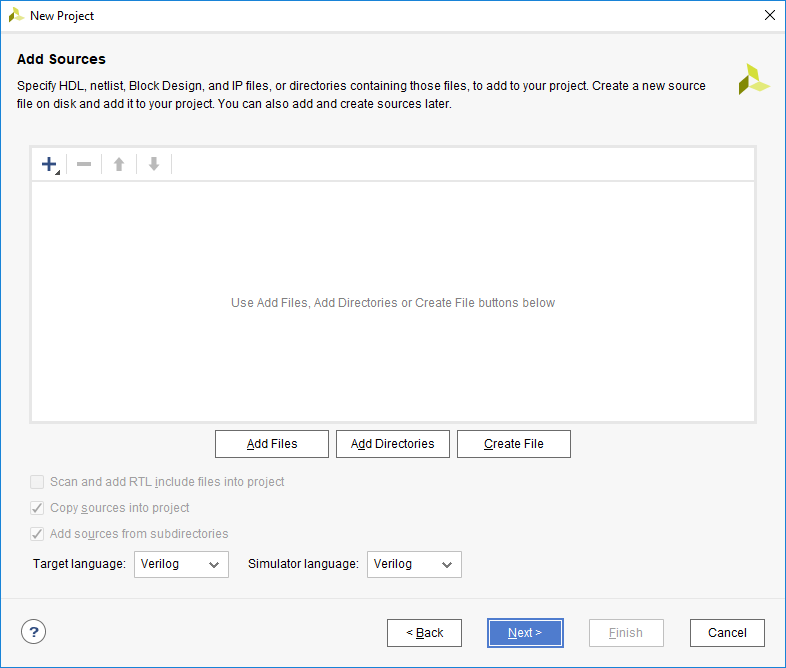


Figure Selecting Target and Simulator language

7. Click on the **Blue Plus** button, then click **Add Files…** and browse to the source directory, select **lab1.v**, and click **OK**.

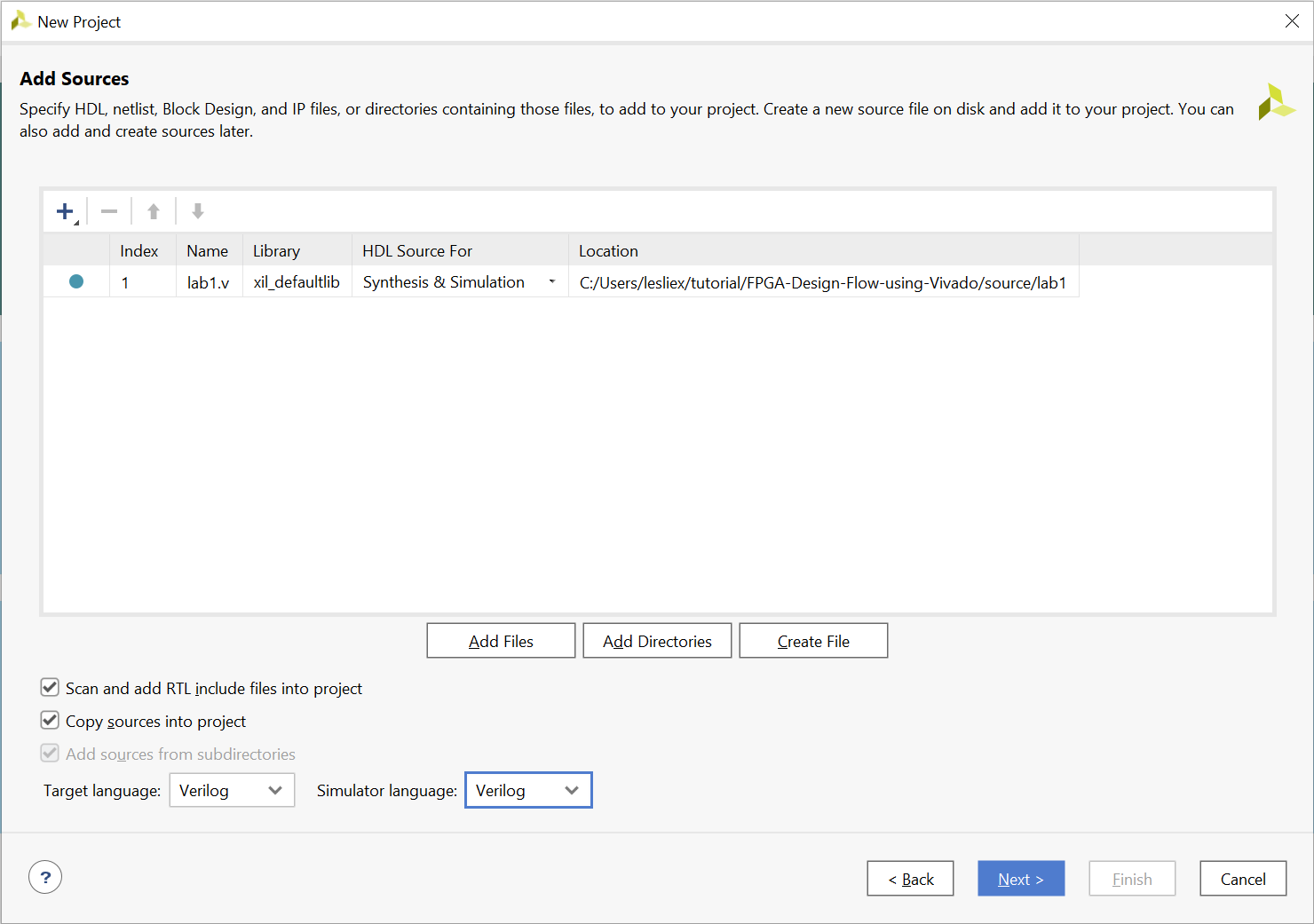


Figure Adding source files

If it is not already checked, check **Copy sources into project**

8. Click Next to get to the *Add Constraints* form.

9. Click on the **Blue Plus** button, then click **Add Files…** and browse to the source directory, select *lab1\_zcu106.xdc* and click **OK** (if necessary), and then click **Next.**

The 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.

10. In the **Boards** tab and then select **Zynq UltraScale+ ZCU106 Evaluation Platform**

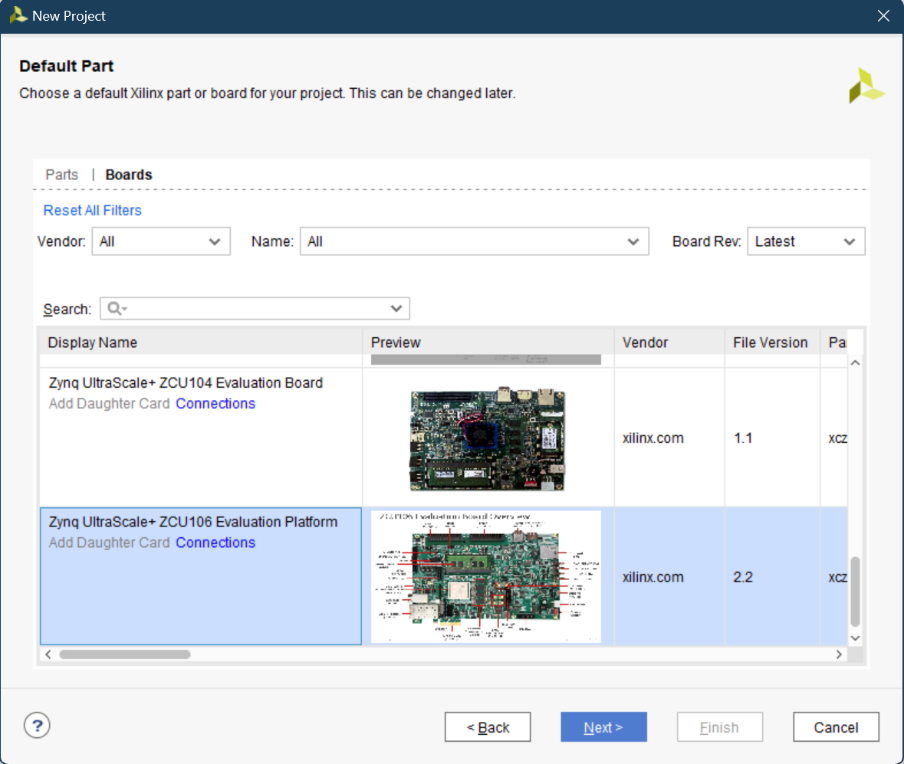


Figure Boards Selection for the ZCU 106

Notice that ZCU106 Board may not be listed as they are not in the tools database. If not listed then you can download the board files for the desired boards from the respective board vendor webpage.

11. Click **Next**.

12. Click **Finish** to create the Vivado project.

**Open the lab1.v source and analyze the content.**

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

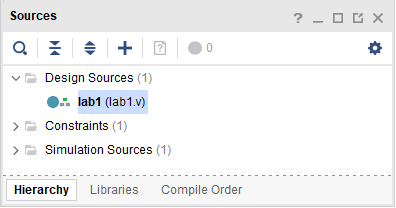


Figure Opening the source file

2. Notice in the Verilog code that the first line defines the timescale directive for the simulator.

3. Line 27 defines the beginning (marked with keyword **module)** and Line 37 defines the end of the module (marked with keyword **endmodule**).

4. Line 28-29 defines the input and output ports whereas lines 32-35 defines the actual functionality.

**Open the lab1 zcu106.xdc source and analyze the content.**

1. In the *Sources* pane, expand the *Constraints* folder and double-click the **lab1\_zcu106.xdc** entry to open the file in text mode.

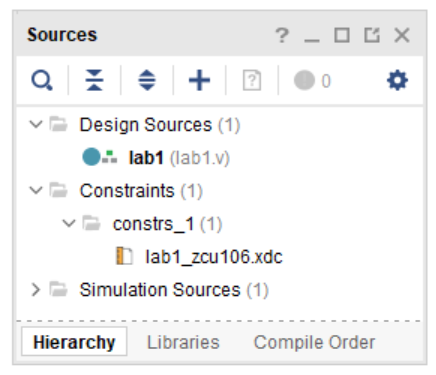


Figure Opening the Constraint file

2. Lines 1-8 define the pin locations for the input buttons and lines 9-16 define pin locations for output LEDs.

**Perform RTL analysis on the source file.**

Expand the *Open Elaborated Design* entry under the *RTL Analysis* tasks of the *Flow Navigator* pane and click on **Schematic**.

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

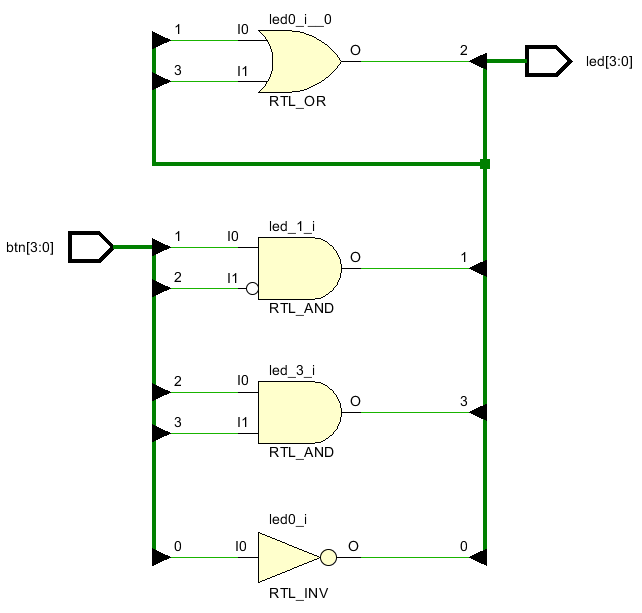


Figure A Logic View of the Design

**Simulate the Design using the Vivado Simulator**

**Add the lab1\_tb.v testbench file.**

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

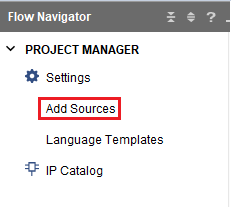


Figure Add sources

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

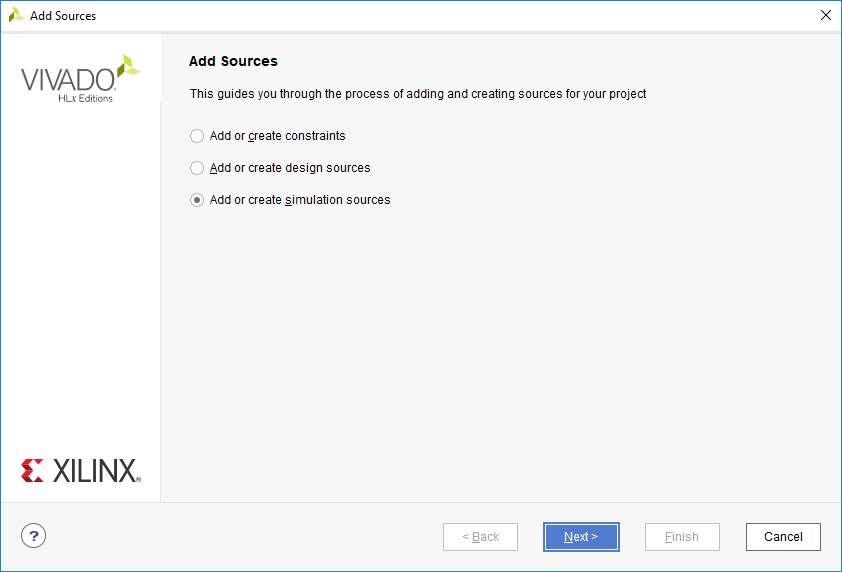


Figure Selecting Simulation Sources option

3. In the *Add Sources Files* form, click the **Blue Plus** button and then then **Add Files….**

4. Browse to the l**ab1** folder and select *lab1\_tb.v* and click **OK**.

5. Click **Finish**.

6. Select the **Sources** tab and expand the *Simulation Sources* group.

The *lab1\_tb.v* file is added under the *Simulation Sources* group, and **lab1.v** is automatically placed in its hierarchy as a dut (device under test) instance.

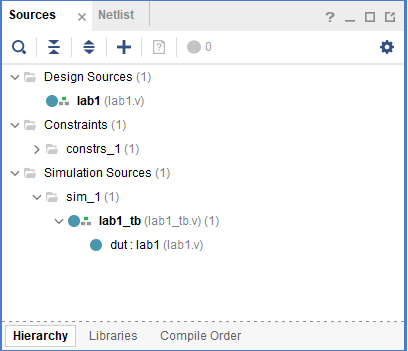


Figure Simulation Sources Hierarchy

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.v is placed under **lab1.srcs > sim\_1 > imports > lab1**.

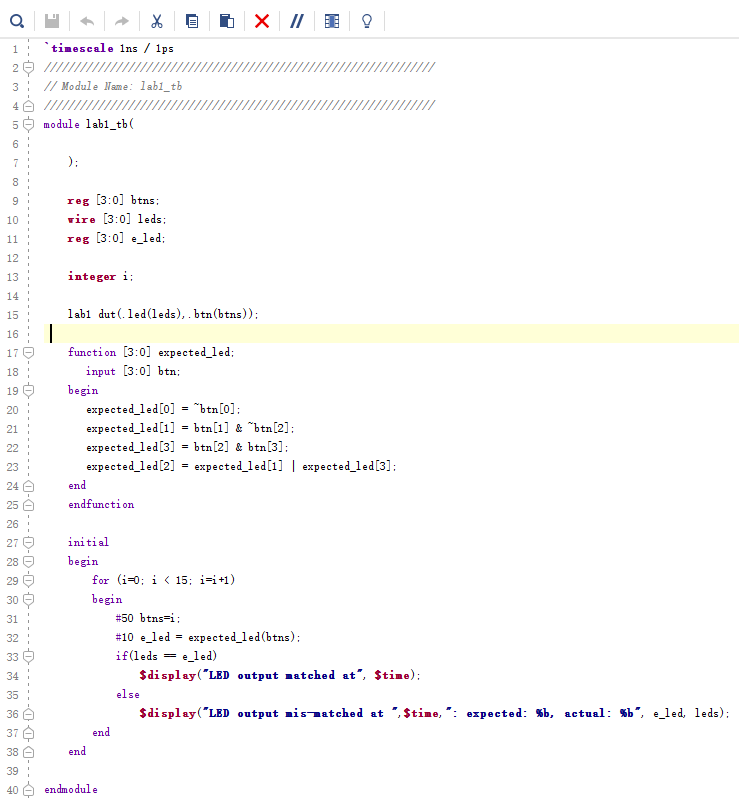
8. Double-click on the **lab1\_tb** in the *Sources* pane to view its contents. ****

Figure The self-checking test bench

The test bench defines the simulation step size and the resolution in line 1. The test bench module definition begins on line 5. Line 15 instantiates the DUT (device/module under test). Lines 17 through 25 define the same module functionality for the expected value computation. Lines 27 through 38 define the stimuli generation, and compare the expected output with what the DUT provides. Line 40 ends the test bench. The **$display** task will print the message in the simulator console window when the simulation is run.

**Simulate the design for 200 ns using the Vivado simulator**

1. Click **Settings** under the **Project Manager** tasks of the **Flow Navigator** pane.

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

2. Select the **Simulation** tab, and set the **Simulation Run Time** value to 200 ns and click **OK**.

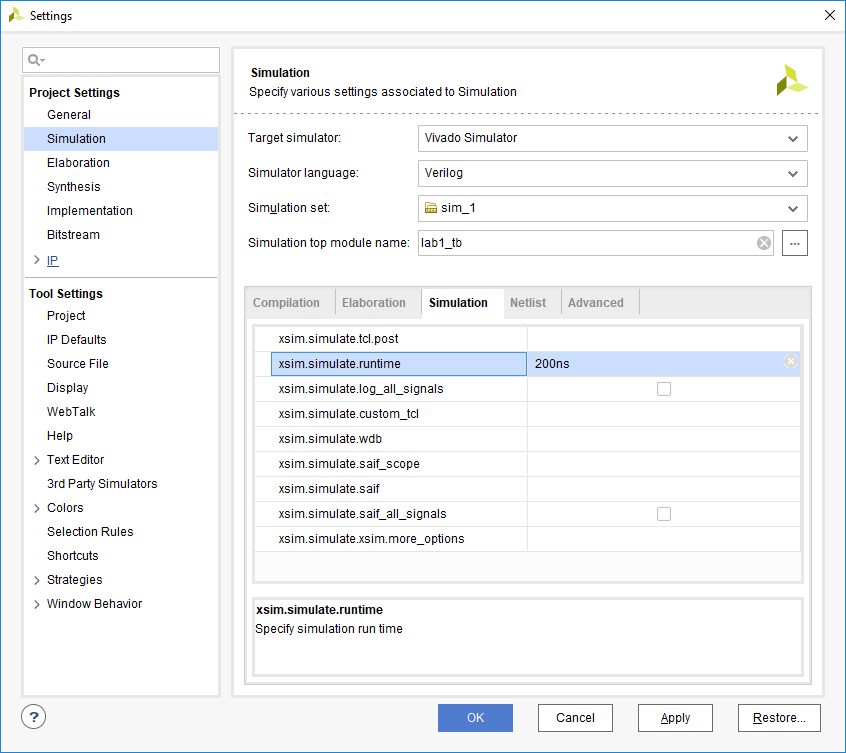


Figure Setting simulation run time

3. Click on **Simulation > 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 Vivado simulator will be run (assuming no errors). You will see a simulator output like the one shown below.

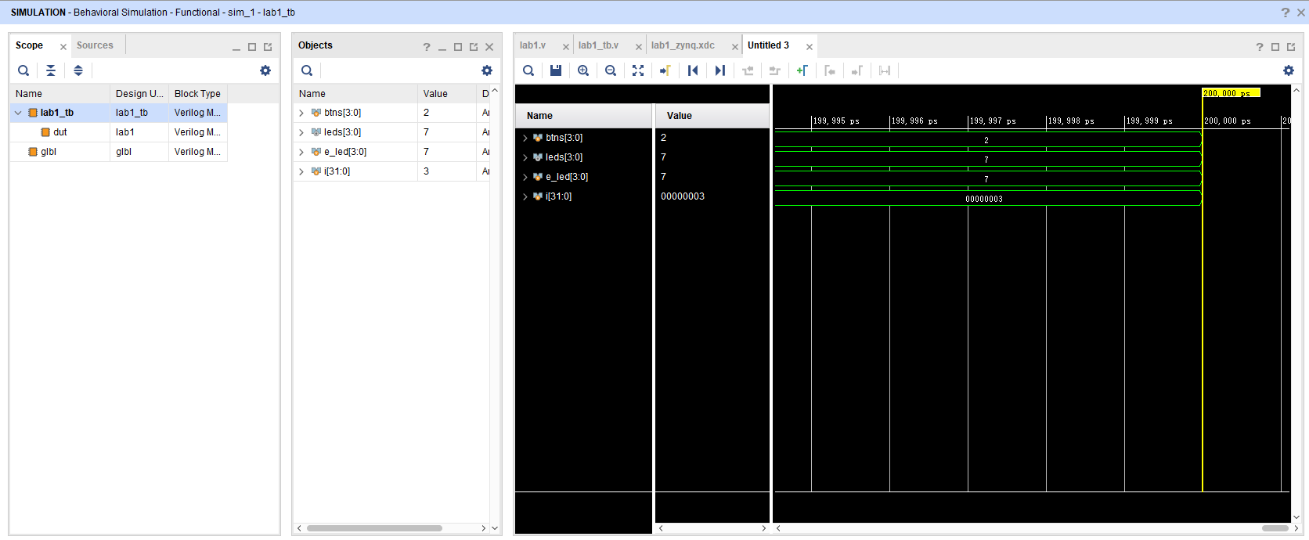


Figure Simulator Output

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.

You will see several buttons next to the waveform window which can be used for the specific purpose as listed in the table below.



Figure Various buttons available to view the waveform

4. Click on the *Zoom Fit* button ([](https://github.com/DucTrungPham/github_fpga_vivado_flow/blob/main/docs/zcu106/images/zcu106/lab1/Fig19.png)) to see the entire waveform.

Notice that the output changes when the input changes.

**Change display format if desired.**

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

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

1. Expand the **lab1\_tb** instance, if necessary, in the *Scopes* window and select the **dut** instance.

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

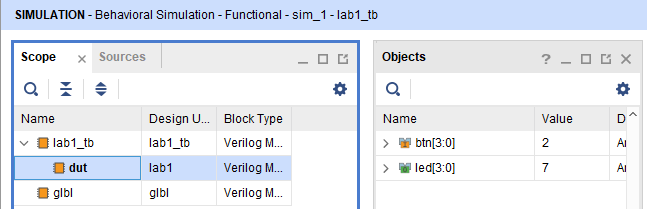


Figure Selecting Lower-Level Signals

2. Select **btn[3:0]** and **led[3:0]** and drag them into the waveform window to monitor those lower-level signals.

3. On the simulator tool buttons ribbon bar , type 500 over in the simulation run time field, click on the drop-down button of the units field and select ns since we want to run for 500 ns (total of 700 ns), and click on the () button. The simulation will run for an additional 500 ns.

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

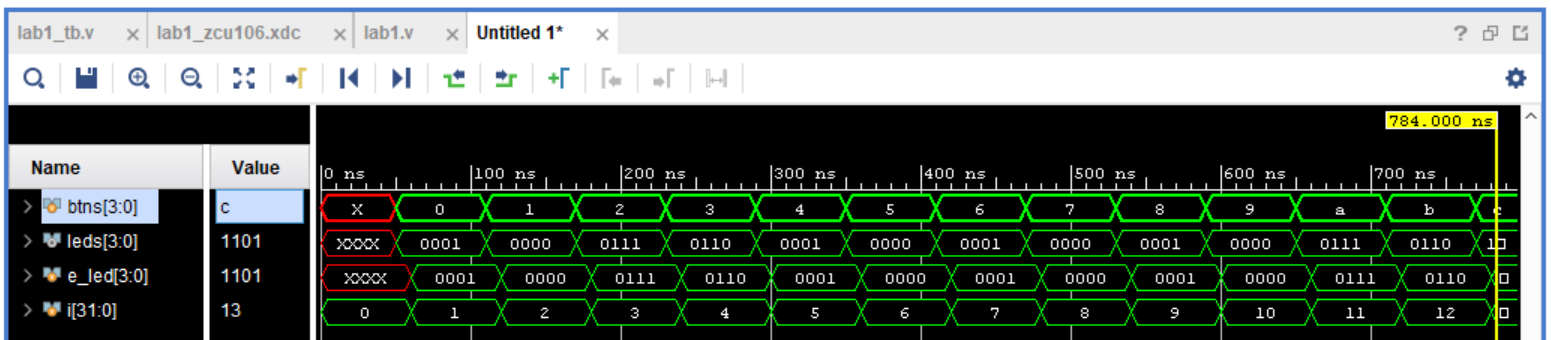


Figure Running simulation for additional 500 ns

Observe the Tcl Console window and see the output is being displayed as the testbench uses the $display task.

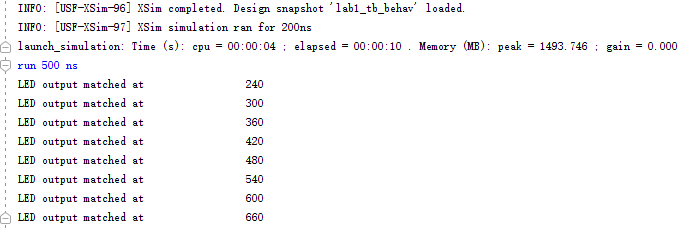


Figure Tcl console output after running the simulation for additional 500 ns

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

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

# Exercises

Make the following changes to the design and testbench files and run the simulation again:

1. Change the design so that it accepts 8 buttons as inputs and 8 leds as outputs. The new 4 outputs should be directly connected to the 4 new inputs.

2. Study the ZCU 106 User Guide to find out which FPGAs pins we should assign to the new 4 inputs and 4 outputs.

3. Change the testbench accordingly to accommodate for the new changes in the design. Run the simulation and make sure it can pass.

4. **(Optional)**

Design a Multiply-and-Accumulator (MAC) module. The inputs to the MAC module are 2 signed, 8-bit integer and the outputs are signed, 19-bit integer. The output is available when the MAC module received 4 pairs of inputs. The output is the sum of the products of the 4 pairs of inputs. For example, given the 2 following matrix

The output is given as the following

The inputs and outputs of the MAC is given as follows

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Signal Name | In/Out | Bus width | Descriptuion |
| 1 | clk | In | 1 | Clock signal |
| 2 | rst\_n | In | 1 | Reset signa, active low signal |
| 3 | ai\_in | In | 8 | Input signal for A |
| 4 | bi\_in | In | 8 | Input signal for B |
| 5 | valid\_in | In | 1 | Valid signal for input  If valid\_in=1, the input is valid  If valid\_in=0, the input is invalid |
| 6 | valid\_out | Out | 1 | Valid signal for the output  If valid\_out=1, we have received 4 inputs, and the output is available  If valid\_out = 0, the output is not available |
| 7 | mac\_out | Out | 19 | The output signal |

a) Crete the design in Verilog

b) Write the testbench and simulate the design in Vivado

**Hint:** A sample block design and hardware architecture for the MAC module is given as the followings:

