Vivado Design Flow

Introduction

This lab guides you through the process of using Vivado IDE to create a simple HDL design targeting the Nexys4 or the Basys3 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 Nexys4 or the Basys3 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 lab1.

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**.

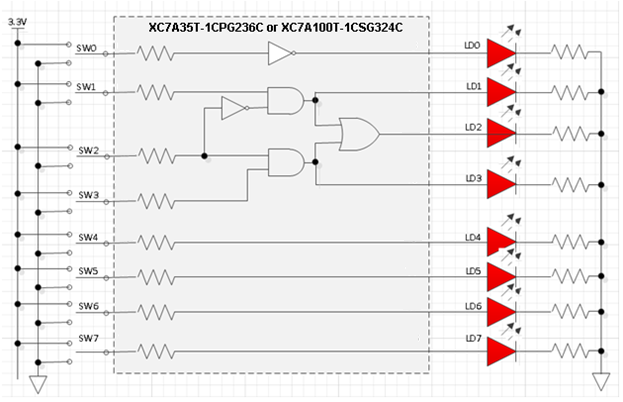


Figure 1. The Completed Design

General Flow

Step 4:

Implement the Design

Step 3:

Synthesize the Design

Step 2: Simulate the Design using Vivado Simulator

Step 1: Create a Vivado Project using IDE

Step 6:

Verify Functionality in Hardware

Step 5: Perform the Timing Simulation

1. Create a Vivado Project using IDE Step 1
   1. Launch Vivado and create a project targeting the XC7A100TCSG324-1 device (Nexys4) or the XC7A35TCPG236-1 (Basys3) and using the Verilog HDL. Use the provided lab1.v and lab1.xdc files from the *2014\_2\_artix7\_sources\lab1* directory.
      1. Open Vivado by selecting **Start > All Programs > Xilinx Design Tools > Vivado 2014.2 > Vivado 2014.2**
      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 **c:\xup\fpga\_flow\2014\_2\_artix7\_labs**, and click **Select**.
      4. Enter **lab1** in the *Project name* field. Make sure that the *Create Project Subdirectory* box is checked. Click **Next**.

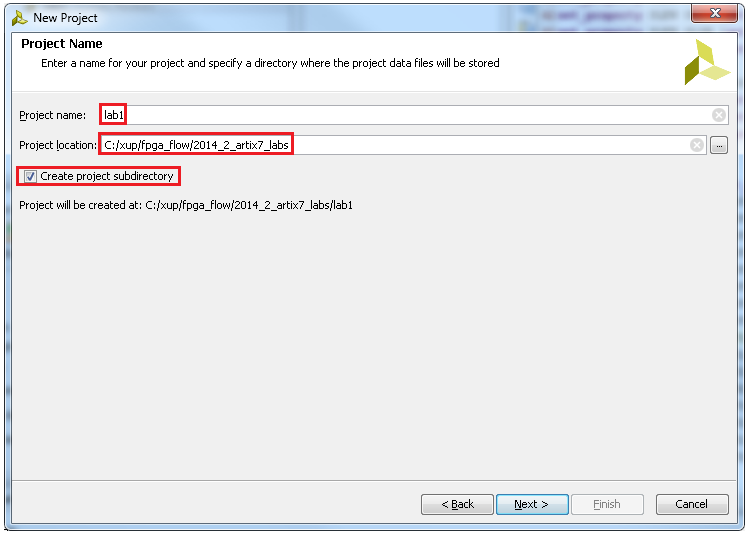


Figure 2. Project Name and Location entry

* + 1. Select **RTL Project** option in the *Project Type* form, and click **Next**.
    2. Using the drop-down buttons, select **Verilog** as the *Target Language* and *Simulator Language* in the *Add Sources* form.

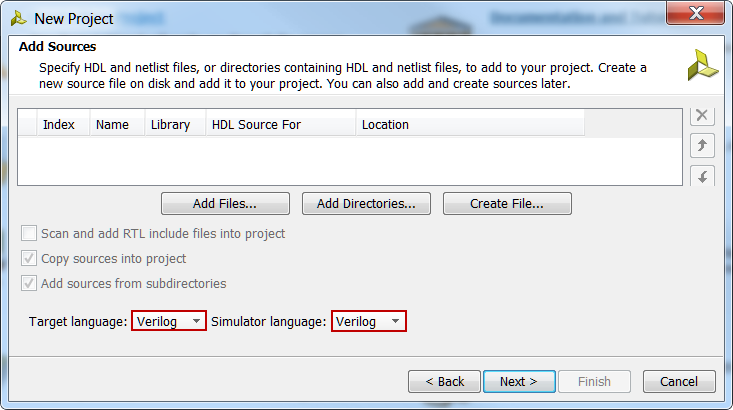


Figure 3. Selecting Target and Simulator language

* + 1. Click on the **Add Files…** button, browse to the **c:\xup\fpga\_flow\2014\_2\_artix7\_sources\lab1** directory, select *lab1.v,* click **OK,** and then click **Next** to get to the *Add Existing IP* form.
    2. Since we do not have any IP to add, click **Next** to get to the *Add Constraints* form.
    3. The constraint file lab1\_basys3.xdc and lab1\_nexys4.xdc are automatically added. Highlight and delete the XDC file not for the target board by clicking on the “X” on the right hand side of the window.

If no files are added, click on the **Add Files…** button, browse to the **c:\xup\fpga\_flow\2014\_2\_artix7\_sources\lab1** directory (if necessary), select *lab1\_basys3.xdc* or *lab1\_nexys4.xdc* and click **OK** (if necessary), 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. In the *Default Part* form, using the **Parts** option and various drop-down fields of the **Filter** section. If using the Nexys4 board, select the **XC7A100TCSG324-1** part. If using the Basys3 board, select the **XC7A35TCPG236-1**. Click **Next**.

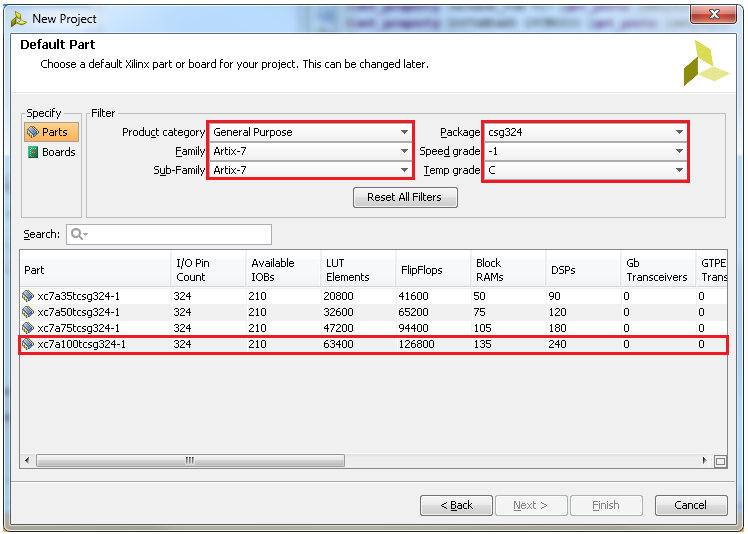


Figure 4. Part Selection for the Nexys4

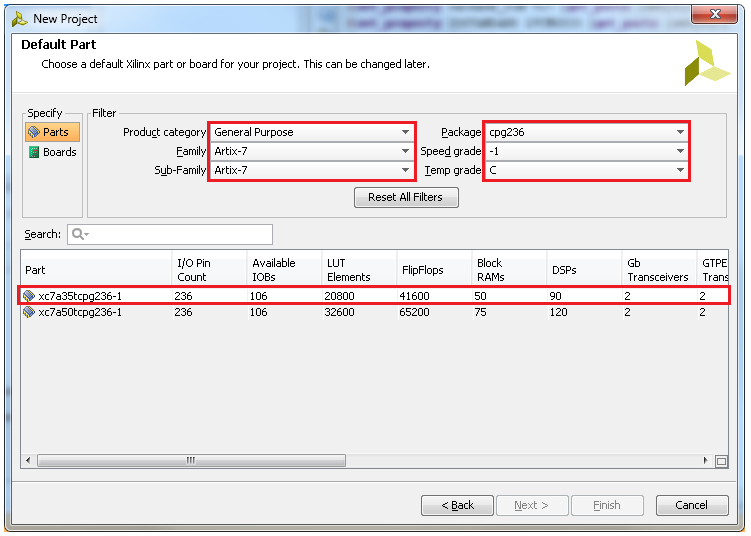


Figure 4. Part Selection for the Basys3

You can select the Boards Specify option, select Artix-7 under the Library filter and select the appropriate board. Notice that Nexys4 and the Basys3 are not listed as they are not in the tools database.

* + 1. Click **Finish** to create the Vivado project.

Use the Windows Explorer and look at the c:\xup\fpga\_flow\2014\_2\_artix7\_labs\lab1 directory. You will find that the lab1.cache and lab1.srcs directories and the lab1.xpr (Vivado) project file have been created. The lab1.cache directory is a place holder for the Vivado program database. 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.v (source) files respectively are placed.

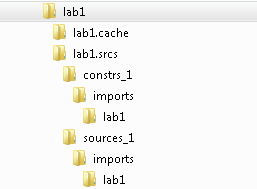


Figure 5. Generated directory structure

* 1. 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.

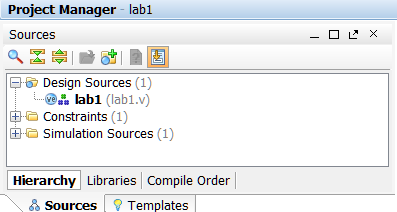


Figure 6. Opening the source file

* + 1. Notice in the Verilog code that the first line defines the timescale directive for the simulator. Lines 2-5 are comment lines describing the module name and the purpose of the module.
    2. Line 7 defines the beginning (marked with keyword **module**) and Line 19 defines the end of the module (marked with keyword **endmodule**).
    3. Lines 8-9 defines the input and output ports whereas lines 12-17 defines the actual functionality.
  1. Open the lab1\_basys3.xdc or lab1\_nexys4.xdc source and analyze the content.
     1. In the *Sources* pane, expand the *Constraints* folder and double-click the **lab1.xdc** entry to open the file in text mode.

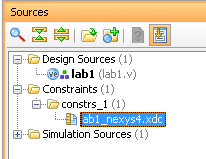


Figure 7. Opening the constraint file

* + 1. For the lines 5-20 defines the pin locations of the input switches [7:0] and lines 25-40 defines the pin locations of the output LEDs [7:0].
  1. Perform RTL analysis on the source file.
     1. 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.

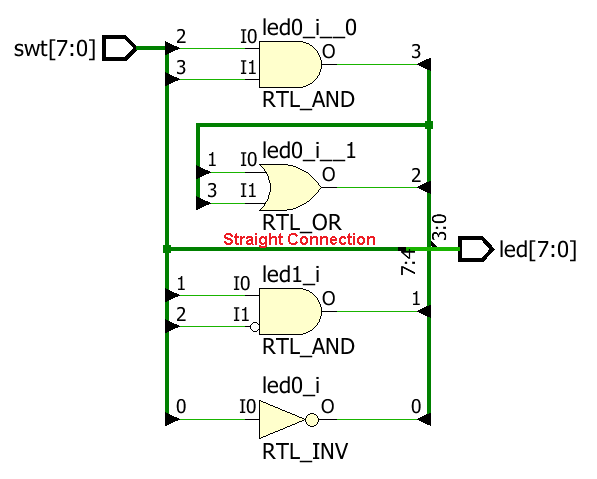


Figure 8. A logic view of the design

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.

1. Simulate the Design using the Vivado Simulator Step 2
   1. Add the lab1\_tb.v testbench file.
      1. Click **Add Sources** under the *Project Manager* tasks of the *Flow Navigator* pane.



Figure 9. Add Sources

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



Figure 10. Selecting Simulation Sources option

* + 1. In the *Add Sources Files* form, click the **Add Files…** button.
    2. Browse to the **c:\xup\fpga\_flow\2014\_2\_artix7\_sources\lab1** folder and select *lab1\_tb.v* and click **OK**.
    3. Click **Finish**.
    4. 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.

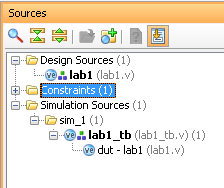


Figure 11. Simulation Sources hierarchy

* + 1. 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**.
    2. Double-click on the **lab1\_tb** in the *Sources* pane to view its contents.

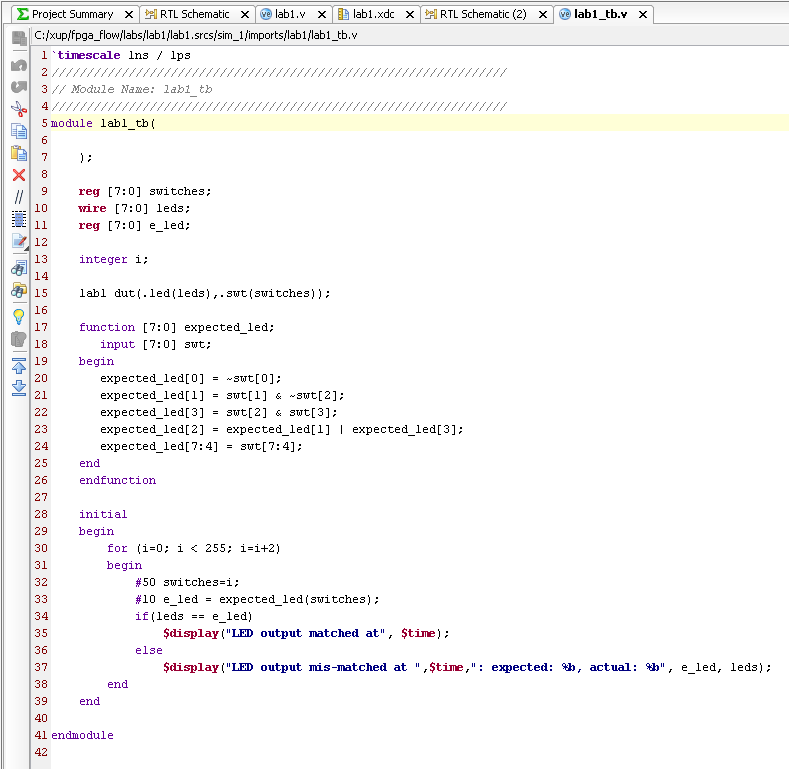


Figure 12. The self-checking testbench

The testbench defines the simulation step size and the resolution in line 1. The testbench module definition begins on line 5. Line 15 instantiates the DUT (device/module under test). Lines 17 through 26 define the same module functionality for the expected value computation. Lines 28 through 39 define the stimuli generation, and compare the expected output with what the DUT provides. Line 41 ends the testbench. The $display task will print the message in the simulator console window when the simulation is run.

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

* + 1. Select the **Simulation** tab, and set the **Simulation Run Time** value to 200 ns and click **OK**.

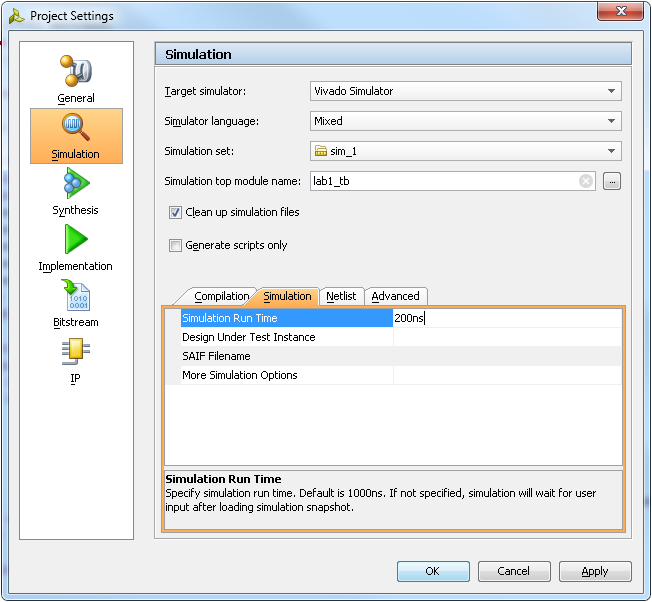


Figure 13. Setting simulation run time

* + 1. 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 Vivado simulator will be run (assuming no errors). You will see a simulator output similar to the one shown below.

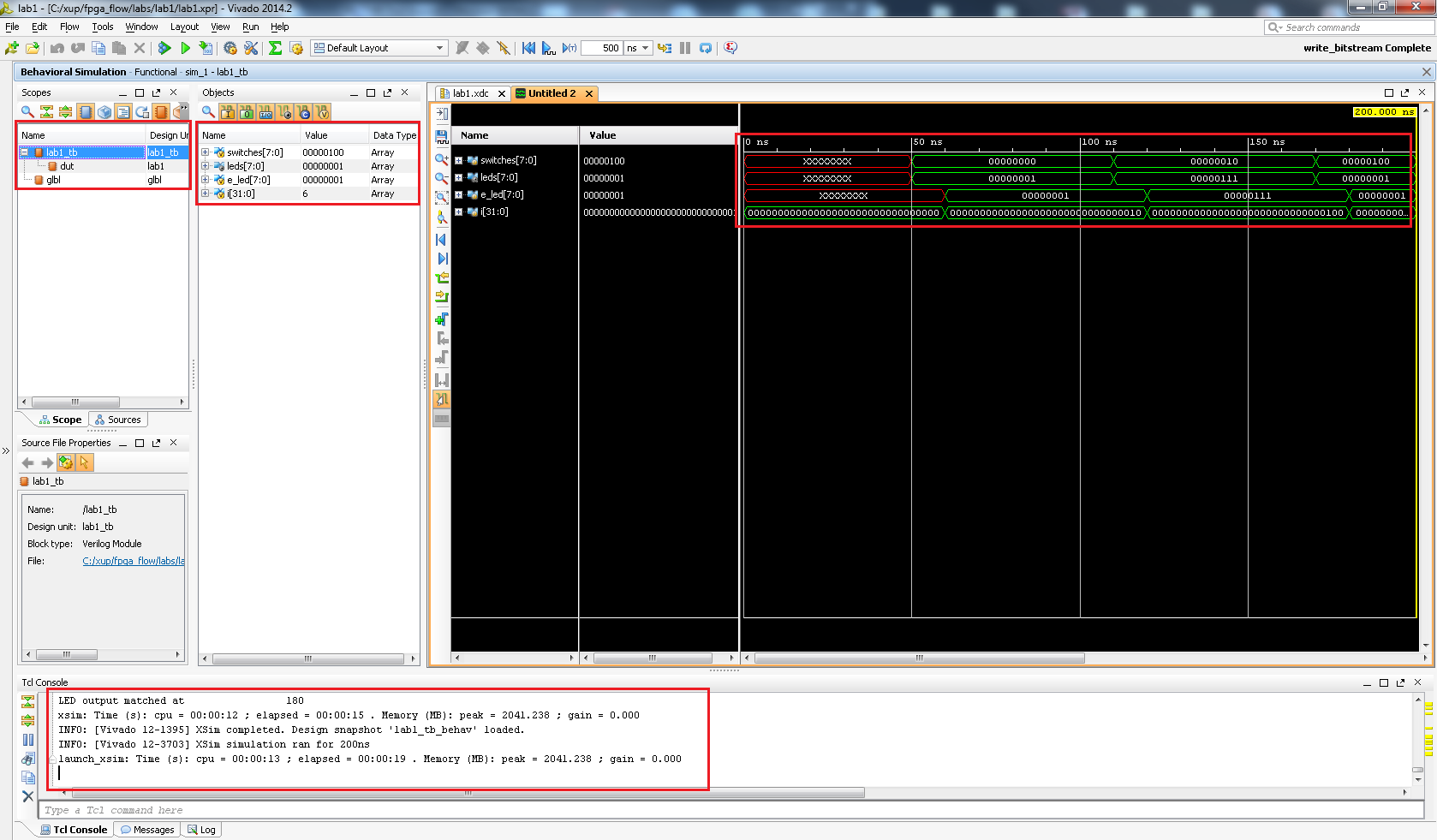


Figure 14. Simulator output

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 **lab1.sim** directory is created under the **lab1** directory, along with several lower-level directories.

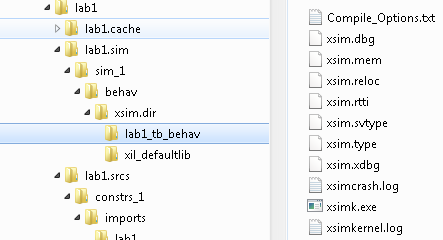


Figure 15. Directory structure after running behavioral simulation

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

Table 1: Various buttons available to view the waveform

|  |  |
| --- | --- |
|  | Waveform options  Save the waveform  Zoom In  Zoom Out  Zoom Fit  Zoom to cursor  Go to Time 0  Go to Last Time  Previous Transition  Next Transition  Add Marker  Previous Marker  Next Marker  Swap Cursors  Snap to Transition  Floating Ruler |

* + 1. 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**

* 1. Change display format if desired.
     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.
  2. 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 swt[7:0] and led[7:0] signals will be displayed in the *Objects* window.

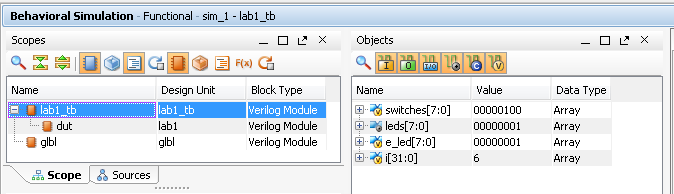


Figure 18. Selecting lower-level signals

* + 1. Select **swt[7:0]** and **led[7:0]** and drag them into the waveform window to monitor those lower-level signals.
    2. 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 () if 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.

* + 1. Click on the *Zoom Fit* button and observe the output.

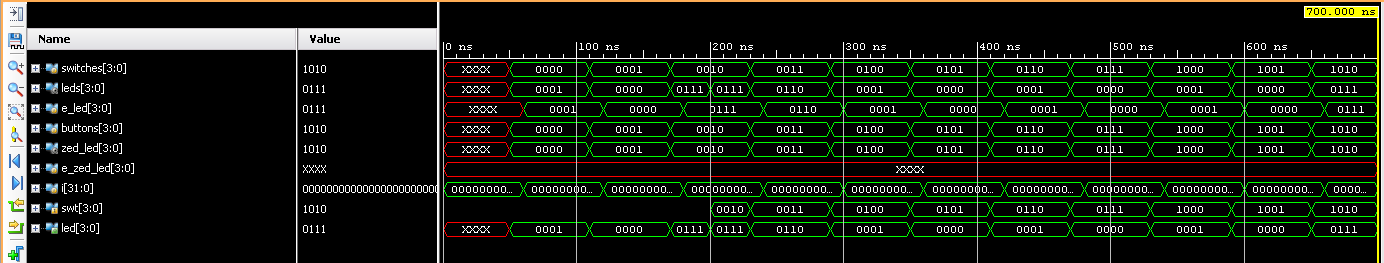


Figure 19. Running simulation for additional 500 ns

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

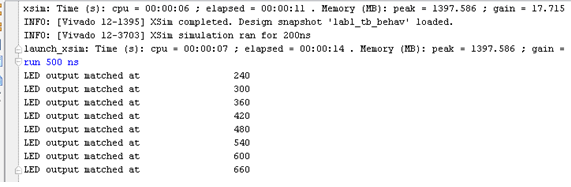


Figure 20. Tcl Console output after running the simulation for additional 500 ns

* + 1. Close the simulator by selecting **File > Close Simulation**.
    2. Click **OK** and then click **No** to close it without saving the waveform.

1. Synthesize the Design Step 3
   1. Synthesize the design with the Vivado synthesis tool and analyze the Project Summary output.
      1. Click on **Run Synthesis** under the *Synthesis* tasks of the *Flow Navigator* pane.

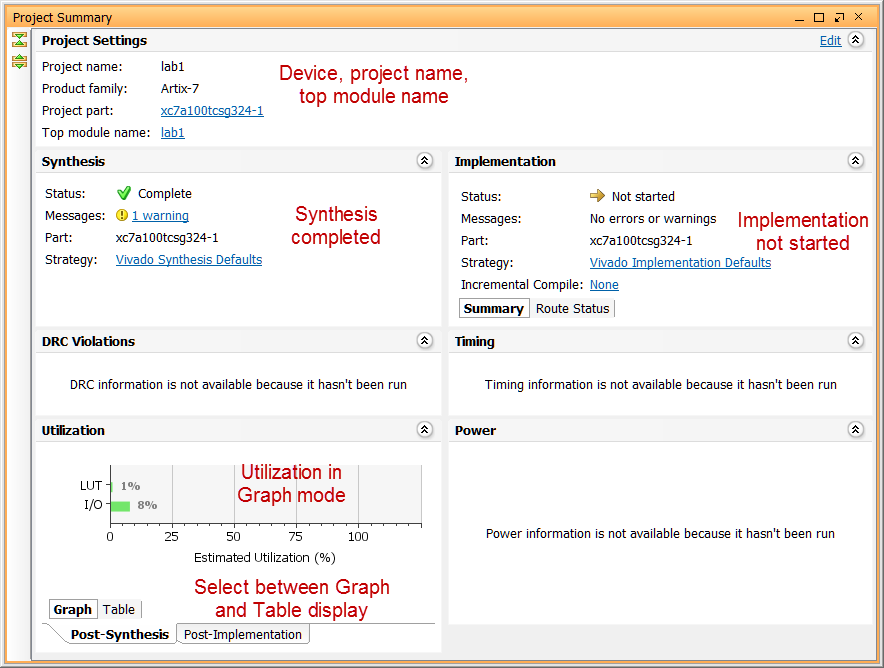
The synthesis process will be run on the lab1.v file (and all its hierarchical files if they exist). When the process is completed a *Synthesis Completed* dialog box with three options will be displayed.

* + 1. 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.

* + 1. Select the **Project Summary** tab and understand the various windows.

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

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

* + 1. Click on the **Table** tab in the **Project Summary** tab.

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

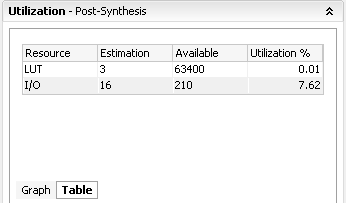


Figure 22. Resource utilization estimation summary for the Nexys4

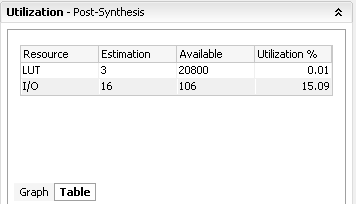


Figure 22. Resource utilization estimation summary for the Basys3

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

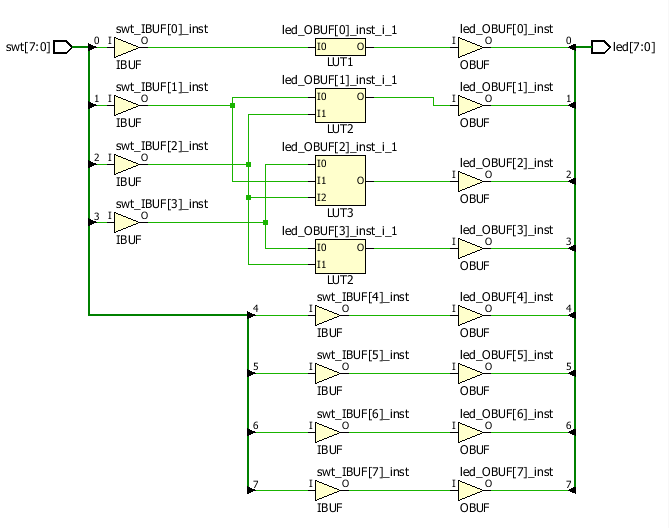


Figure 23. Synthesized design’s schematic view

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). Four gates in RTL analysis output are mapped onto four LUTs in the synthesized output.

Using Windows 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.

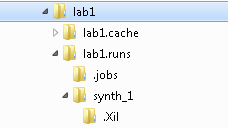


Figure 24. Directory structure after synthesizing the design

1. Implement the Design Step 4
   1. Implement the design with the Vivado Implementation Defaults (Vivado Implementation 2014) settings and analyze the Project Summary output.
      1. Click on **Run Implementation** under the *Implementation* tasks of the *Flow Navigator* pane.

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.

* + 1. Select **Open implemented design** and click **OK** as we want to look at the implemented design in a Device view tab.
    2. Click **Yes,** if prompted, to close the synthesized design.

The implemented design will be opened.

* + 1. In the *Netlist* pane, select one of the nets (e.g. led\_OBUF[1]) and notice that the net displayed in the X1Y1 clock region in the Device view tab (you may have to zoom in to see it).
    2. If it is not selected, click the *Routing Resources* icon  to show routing resources.

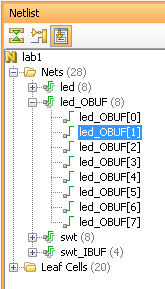


Figure 25. Selecting a net

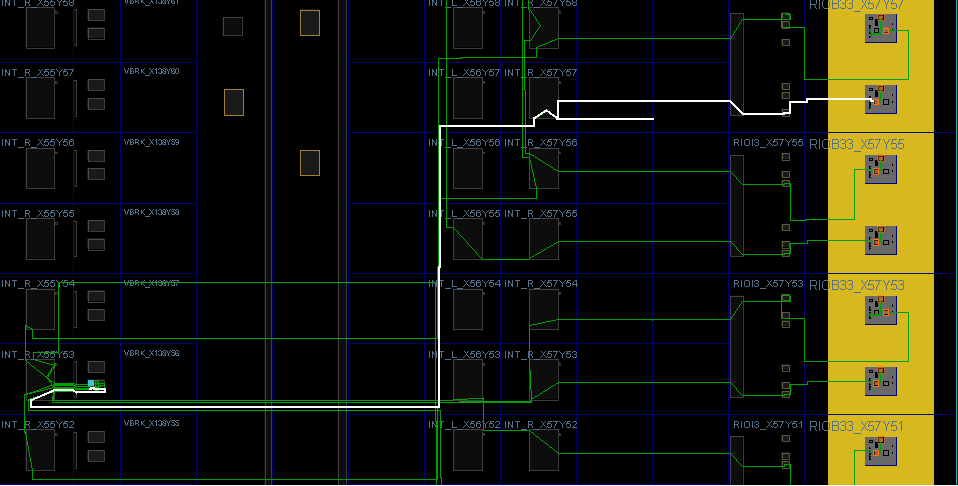


Figure 26. Viewing implemented design for the Nexys4

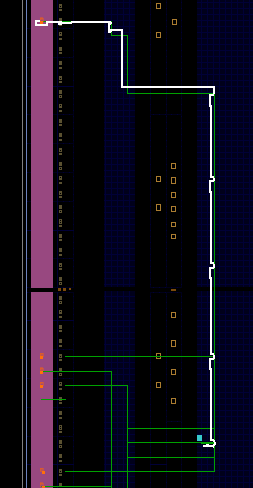


Figure 26. Viewing implemented design for the Basys3

* + 1. 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.

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).

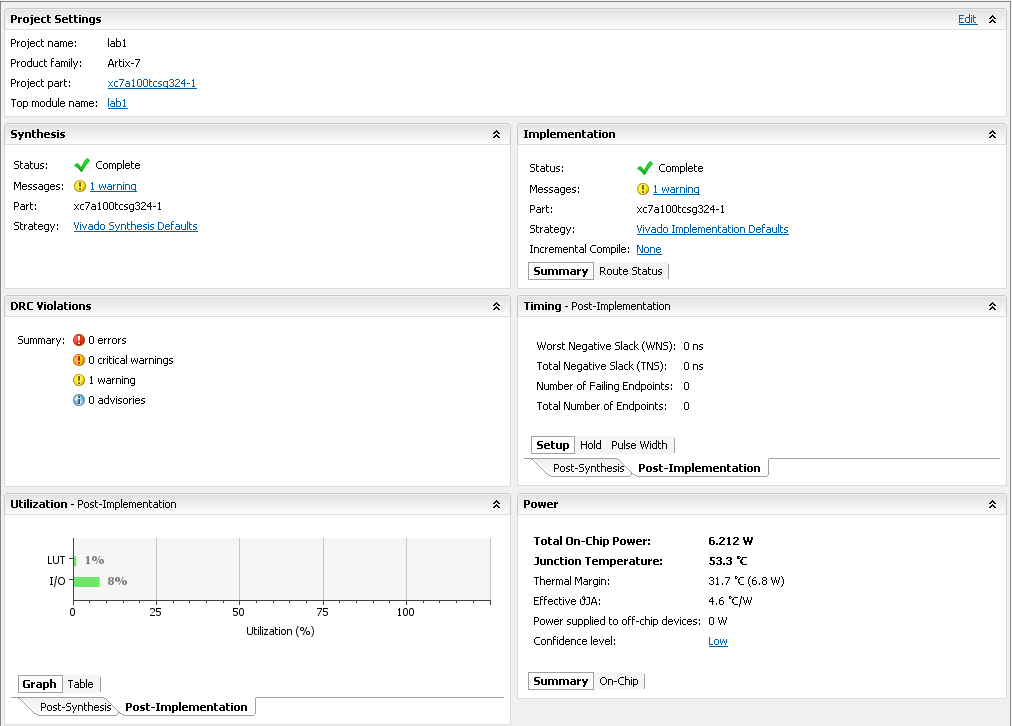
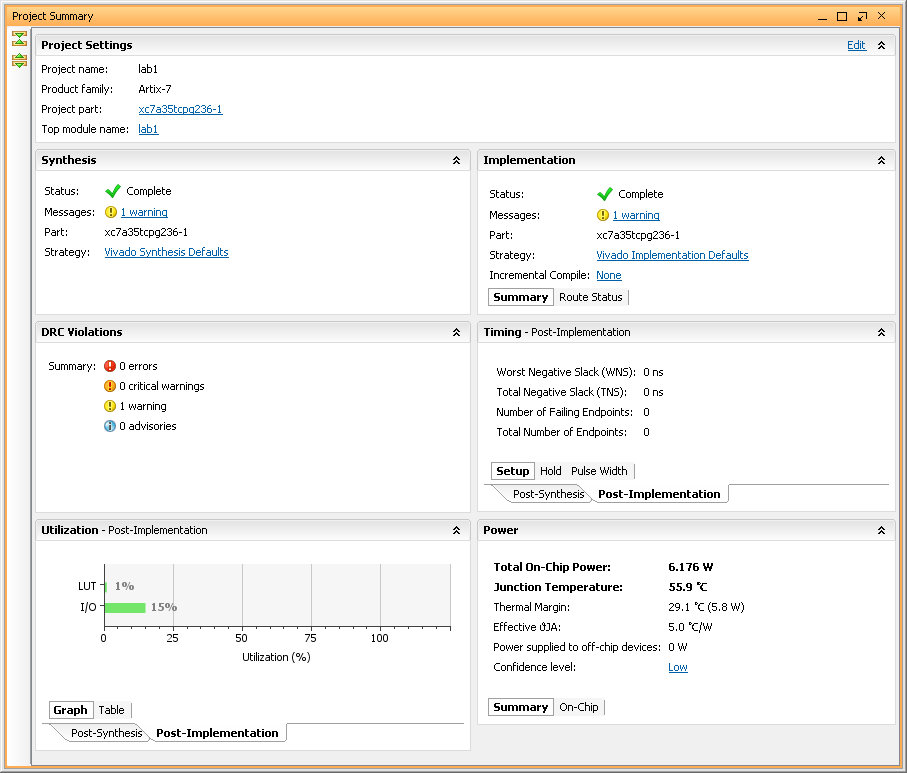


Figure 27. Implementation results for the Nexys4

Figure 27. Implementation results for the Basys3

Using the Windows 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.

* + 1. 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 combinatorial no registers are used.

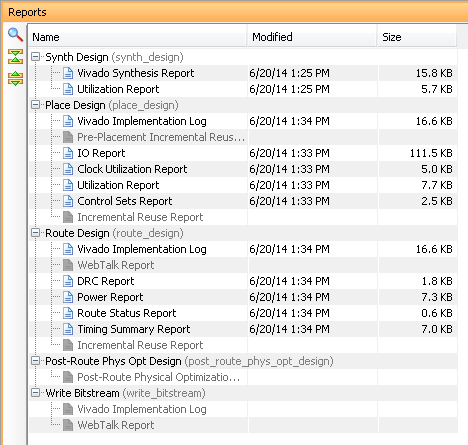


Figure 28. Available reports to view

1. Perform Timing Simulation Step 5
   1. Run a timing simulation.
      1. Select **Run Simulation > Run Post-Implementation Timing Simulation** process under the *Simulation* tasks of the *Flow Navigator* pane.

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.

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

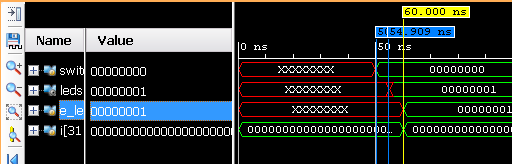


Figure 29. Timing simulation output

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.

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

1. Generate the Bitstream and Verify Functionality Step 6
   1. Connect the board and power it ON. Generate the bitstream, open a hardware session, and program the FPGA.
      1. Make sure that the Micro-USB cable is connected to the JTAG PROG connector (next to the power supply connector).
      2. Make sure that the board is set to use USB power (via the Power Select jumper JP3 on the Nexys4 and JP2 on the Basys3)

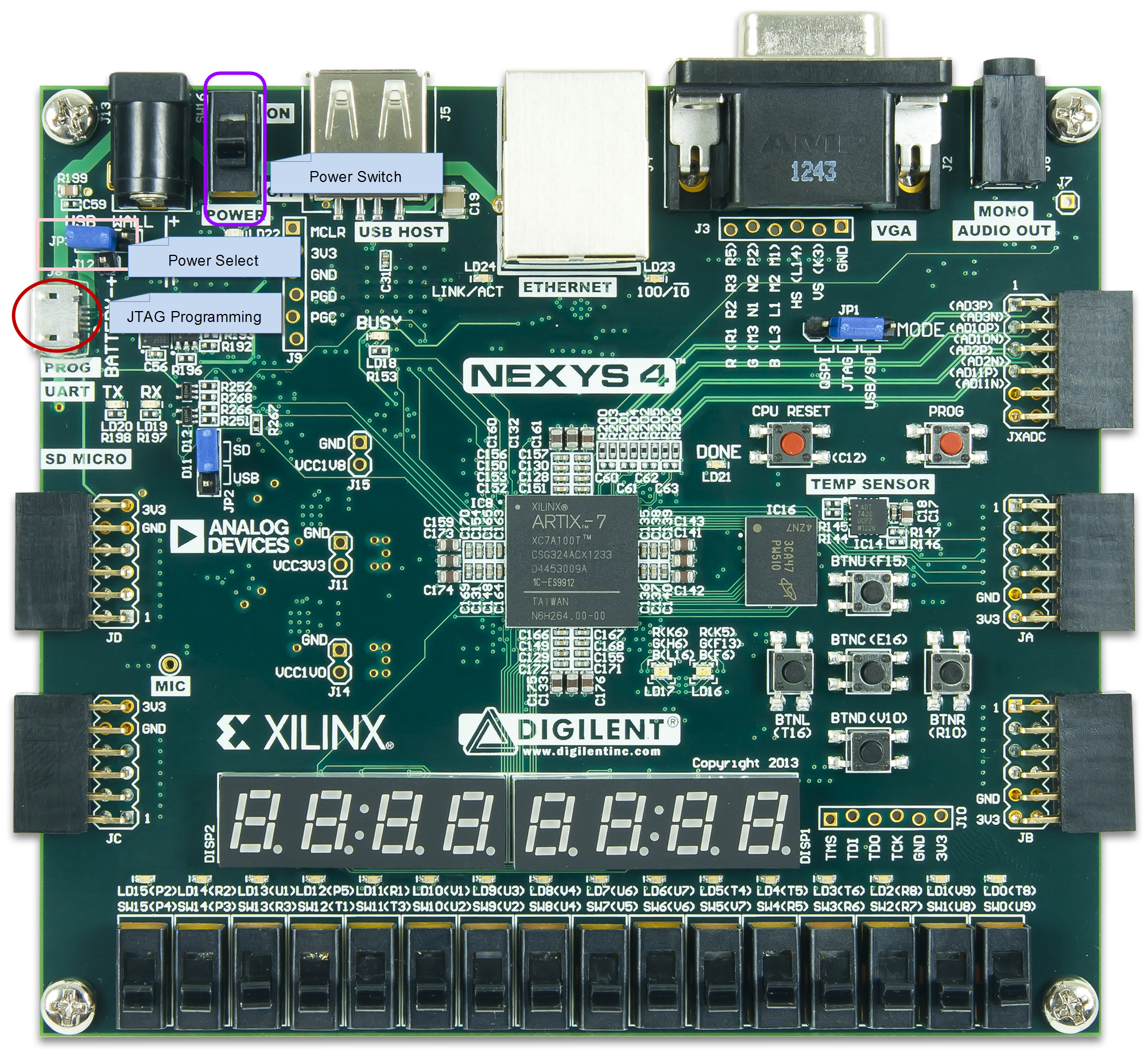


Figure 30. Board connection for the Nexys4



Figure 30. Board connection for the Basys3

* + 1. Power **ON** the board.
    2. 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 two options will be displayed.

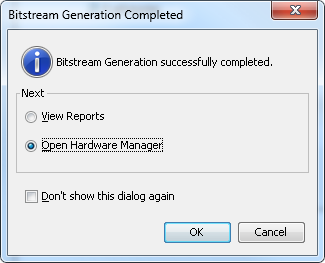


Figure 31. Bitstream generation

This process will have generated a **lab1.bit** file under **impl\_1** directory in the **lab1.runs** directory.

* + 1. Select the *Open Hardware Manager* option and click **OK**.

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

* + 1. Click on the **Open a new hardware target** link.

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



Figure 32. Opening new hardware target

* + 1. Click **Next**  to see the Hardware Server Settings form.
    2. Click **Next** with the Hardware Target selected.

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

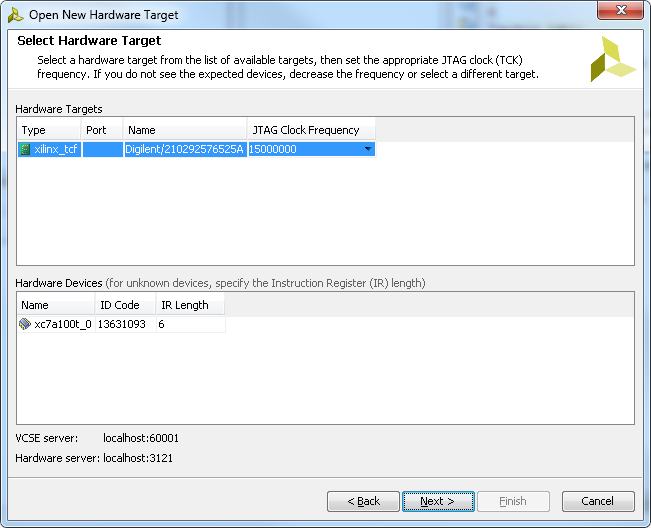
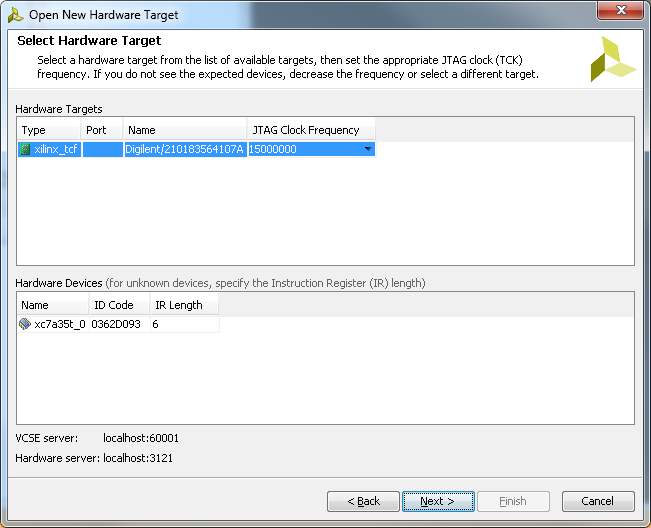


Figure 33. New hardware target detection for the Nexys4

  
Figure 33. New hardware target detection for the Basys3

* + 1. Click **Next** and then **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.

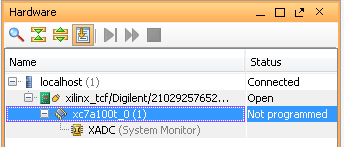


Figure 34. Opened hardware session for the Nexys4

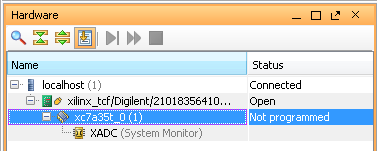


Figure 34. Opened hardware session for the Basys3

* + 1. Select the device and verify that the lab1.bit is selected as the programming file in the General tab.

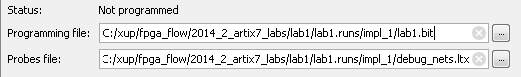


Figure 35. Programming file

* + 1. Click on the *Program device > XC7A100T\_0* or the *XC7A35T\_0* link in the green information bar to program the target FPGA device.

Another way is to right click on the device and select *Program Device…*



Figure 36. Selecting to program the FPGA

* + 1. 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.

* + 1. Verify the functionality by flipping switches and observing the output on the LEDs (Refer to the earlier logic diagram).
    2. When satisfied, power **OFF** the board.
    3. Close the hardware session by selecting **File > Close Hardware Manager.**
    4. Click **OK** to close the session.
    5. 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.