

A Basic SoC

Introduction

This lab guides you through the process of using Vivado IDE to create a simple Cortex-M0 soft core based SoC design targeting the Nexys4 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 attaching to LEDs located on the Nexys4 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.

NOTE: Before proceeding with the lab, please copy the Cortex M0 Design Start files (CORTEXM0DS.v and cortexm0ds_logic.v) in to these locations

1. [P6/Lab Part1/FPGA/Source](#)
2. [P6/Lab Part2/FPGA/Source](#)
3. [P7/Lab/FPGA/Source/CortexM0-DS](#)
4. [P8/Lab/FPGA/Source/CortexM0-DS](#)

Design Description

The design consists of the Cortex-M0 Design Start (DS) core, 1 KB of RAM implemented in FPGA, AHB2LED IP to drive LEDs as shown in **Figure 1**.

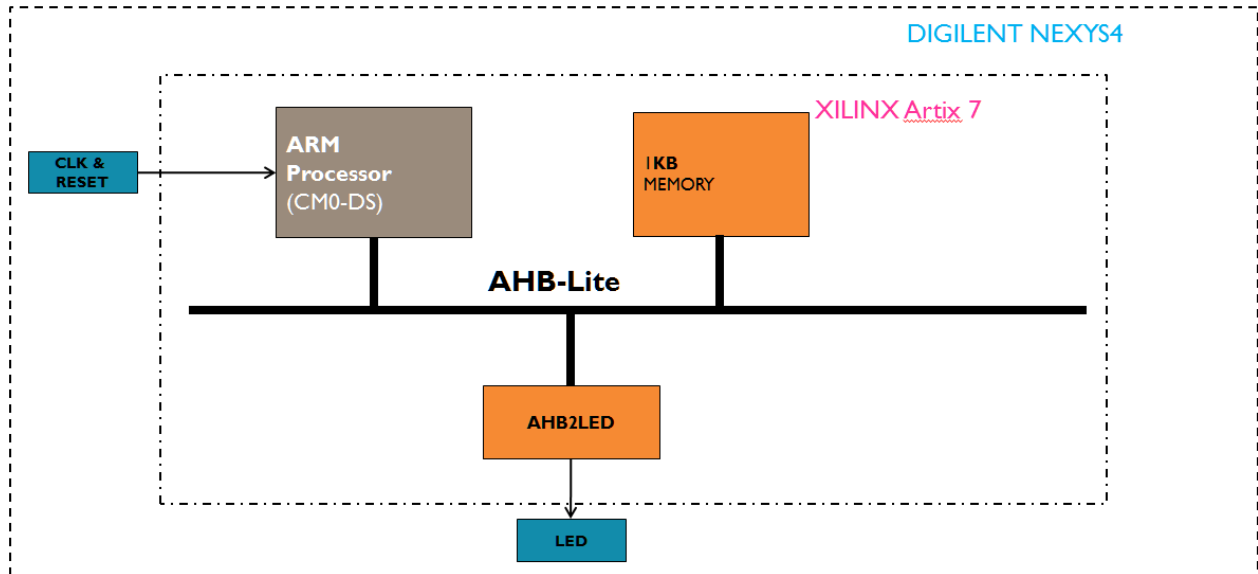
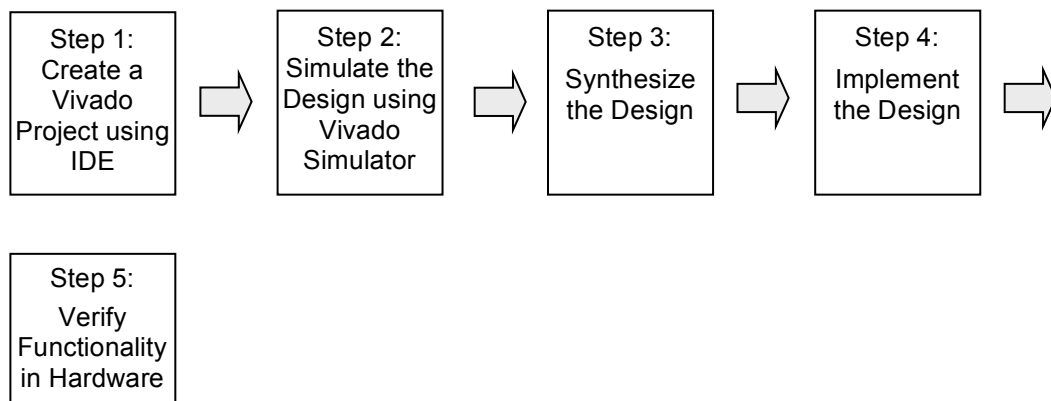


Figure 1. The Completed Design

General Flow



Create a Vivado Project using IDE

Step 1

1-1. Launch Vivado and create a project targeting the XC7A100TCSG324-1 device and using the Verilog HDL. Use the provided files from the *P6\FPGA\Source* directory.

1-1-1. Open Vivado by selecting **Start > All Programs > Xilinx Design Tools > Vivado 2013.4 > Vivado 2013.4**

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 *<labs_install>\P6\FPGA*, 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**.

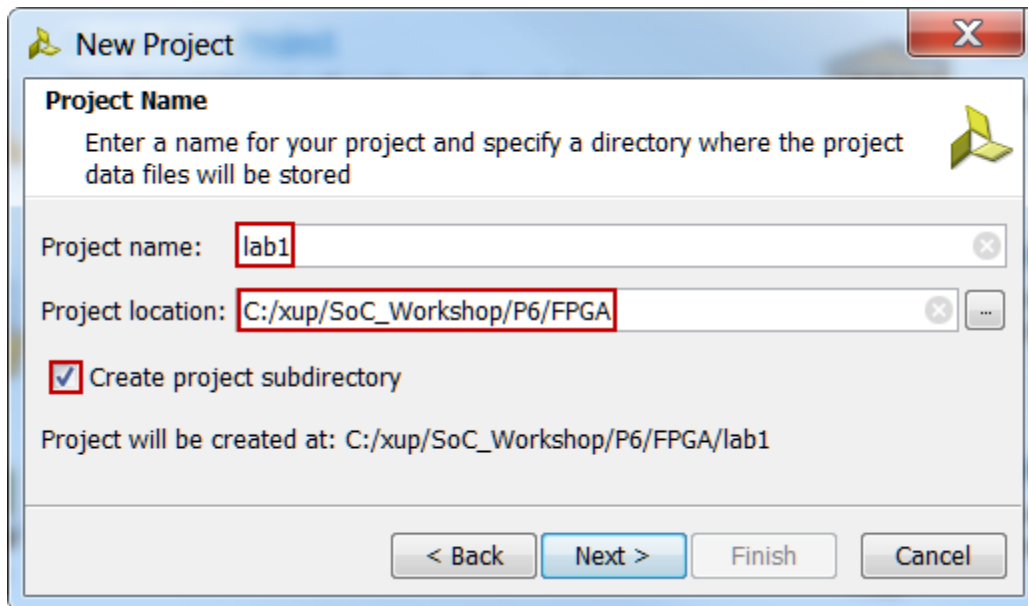


Figure 2. Project Name and Location entry

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

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

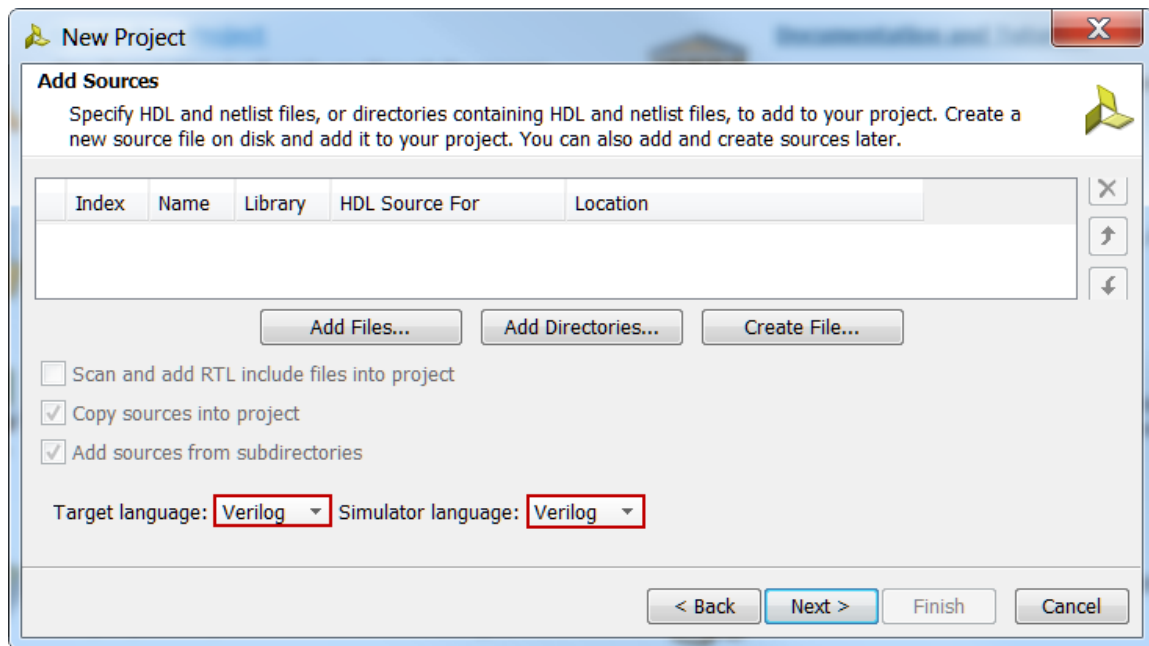


Figure 3. Selecting Target and Simulator language

1-1-7. Click on the **Add Files...** button, browse to the `<labs_install>\P6\FPGA\Source` directory, select `AHB2LED.v`, `AHB2BRAM.v`, `AHBDCD.v`, `AHBLITE_SYS.v`, `AHBMUX.v`, `CORTEXMODS.v`, and `cortexm0ds_logic.v`, (do not select `lab1_tb.v`) and click **OK**, and then click **Next** to get to the *Add Existing IP* form.

1-1-8. Since we do not have any IP to add, click **Next** to get to the *Add Constraints* form.

1-1-9. Click on the **Add Files...** button, browse to the `<labs_install>\P6\FPGA\Source` directory (if necessary), select `Nexys4_Master.xdc` and click **OK** (if necessary), and then click **Next**.

This Xilinx Design Constraints file assigns the physical IO locations on FPGA to the clk, reset, and LEDs located on the board. This information can be obtained either through the board's schematic or the board's user guide.

1-1-10. In the *Default Part* form, using the **Parts** option and various drop-down fields of the **Filter** section, select the **XC7A100TCSG324-1** part. Click **Next**.

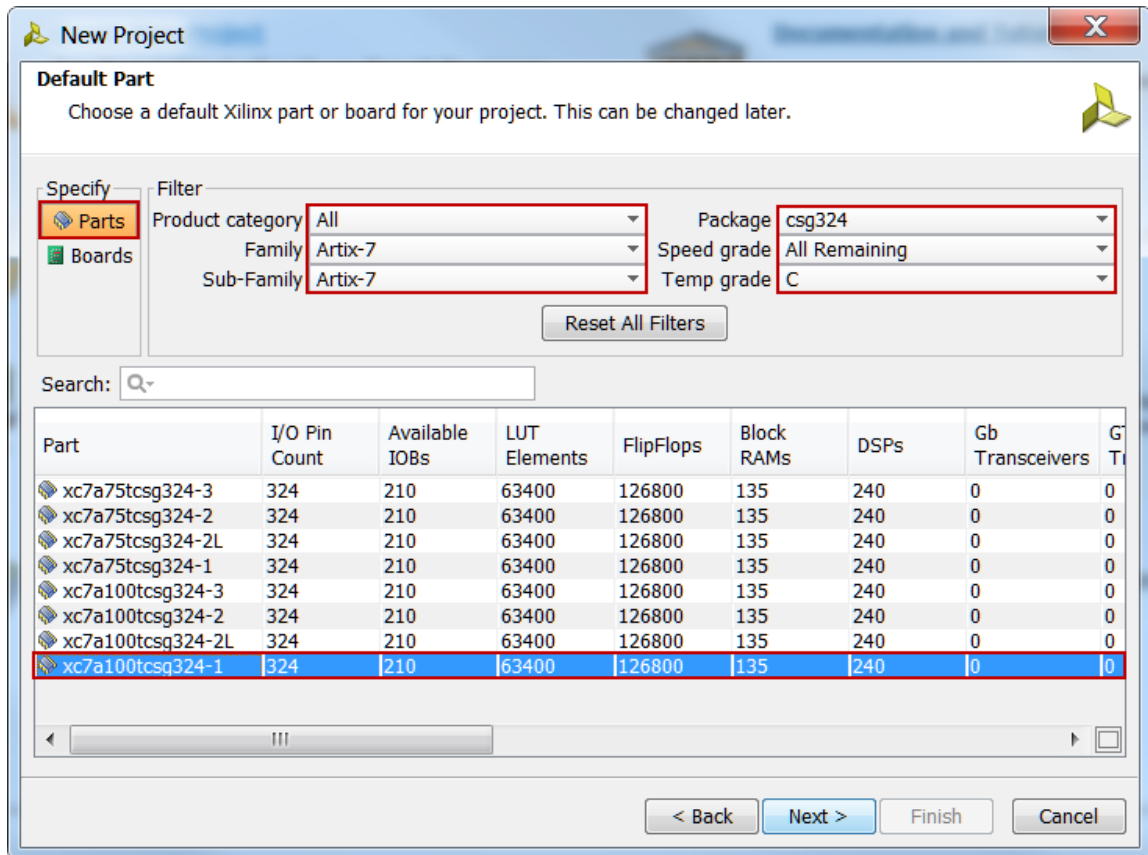


Figure 4. Part Selection

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

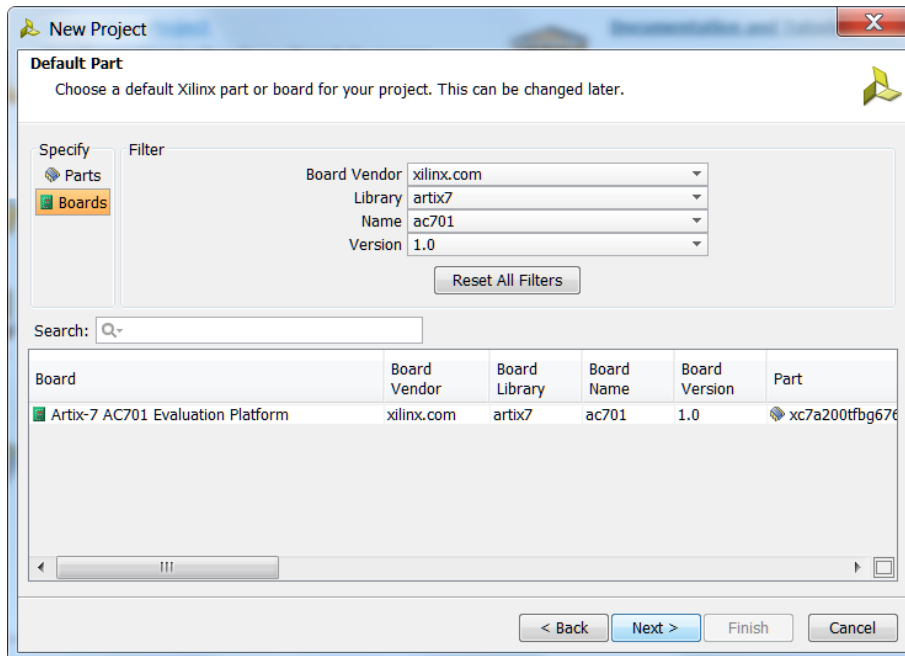


Figure 5. Part Selection using Boards Specify filter

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

Use the Windows Explorer and look at the <labs_install>\P6\FPGA\lab1 directory. You will find that the **lab1.data** and **lab1.srcs** directories and the **lab1.xpr** (Vivado) project file have been created. The **lab1.data** 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 **Nexys4_Master.xdc** (constraint) and **all Verilog** (source) files respectively are placed.

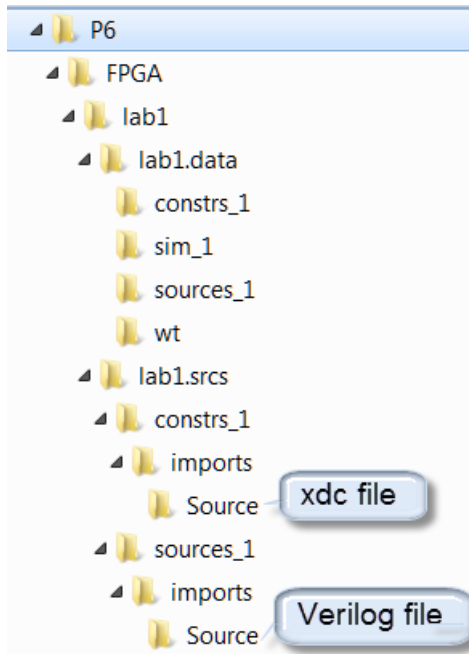


Figure 6. Generated directory structure

1-2. Open the AHBLITE_SYS.v source and analyze the content.

1-2-1. In the *Sources* pane, double-click the **AHBLITE_SYS.v** entry to open the top-level file in text mode.

You can click on the + sign to expand the hierarchy and see lower-level instantiated modules and associated source files.

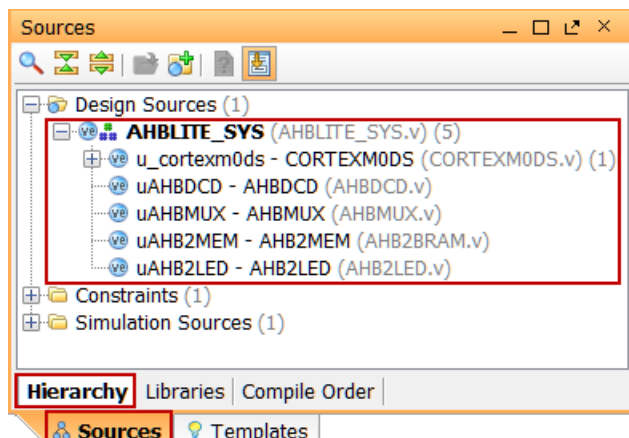


Figure 7. Opening the source file

1-2-2. Line 38 defines the beginning (marked with keyword **module**) and Line 239 defines the end of the module (marked with keyword **endmodule**).

1-2-3. Lines 40-45 define the input and output ports, lines 49-85 define internal connections/nets, whereas lines 144-238 define various modules instantiations.

1-3. Open the Nexys4_Master.xdc source and analyze the content.

1-3-1. In the *Sources* pane, expand the *Constraints* folder and double-click the **Nexys4_Master.xdc** entry to open the file in text mode.

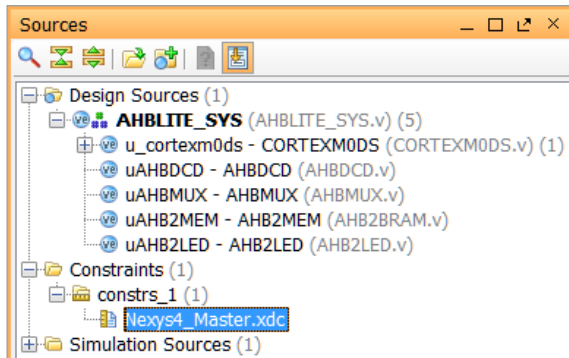



Figure 8. Opening the constraint file

1-3-2. Lines 8-10 define the clock pin location, its I/O property, and 100 MHz frequency constraint. Lines 13-14 define the reset pin location which is controlled by SW15 of the board. Lines 20-42 define the pin locations of the output LEDs [7:0] and lines 44-45 define the LOCKUP signal on LED15 of the board.

1-4. Perform RTL analysis on the source file.

1-4-1. Expand the *Open Elaborated Design* entry under the *RTL Analysis* tasks of the *Flow Navigator* pane and click on **Schematic**.

A warning indicating that the BRAM data file could not be found will be displayed. Click **OK** to close the warning box for now.

The model (design) will be elaborated and a logic view of the design is displayed. Click on the Zoom-In button () in the left vertical toolbar to see zoom in.

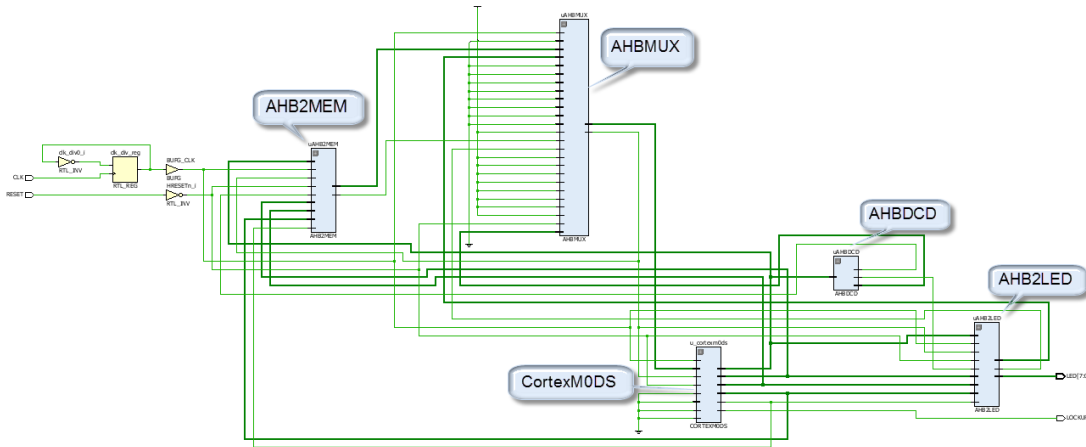


Figure 9. A logical view of the design

1-4-2. Double-click on the **CortexM0DS** block to see the next-level module instance as well as ports.

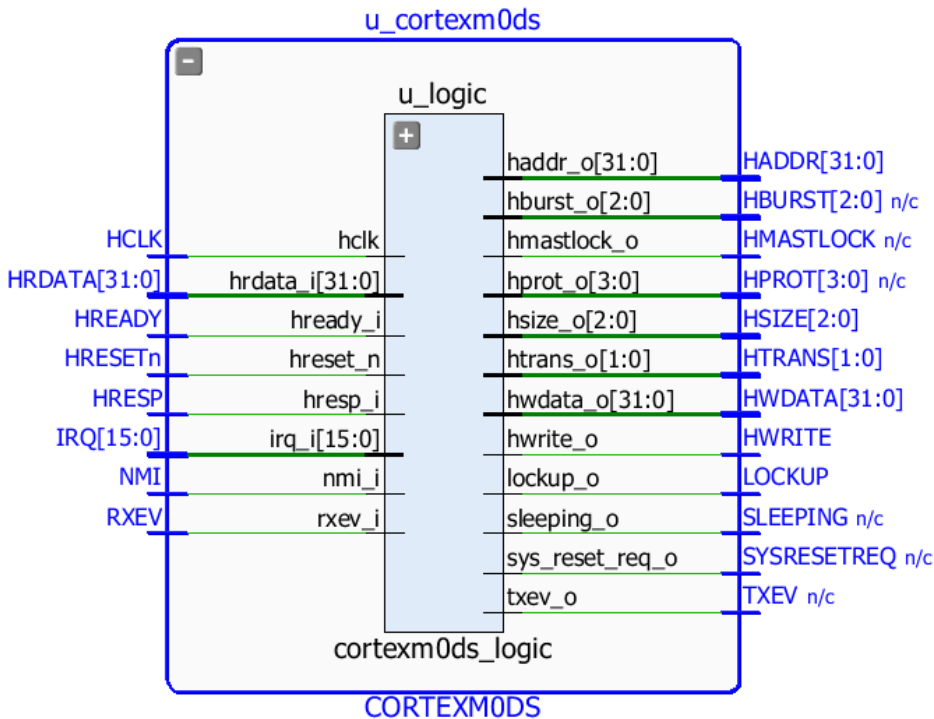


Figure 10. Hierarchy view

1-4-3. Click on the Previous button (←) to see the upper-level view.

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.

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

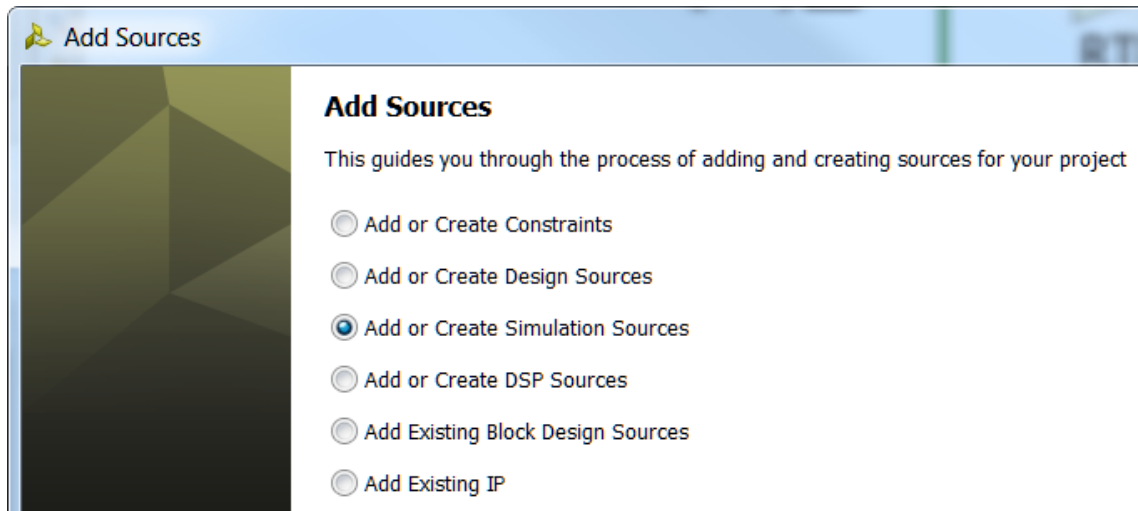


Figure 12. Selecting Simulation Sources option

2-1-3. In the *Add Sources Files* form, click the **Add Files...** button.

2-1-4. Browse to the `<labs_install>\IP6\FPGA\Source` folder and select `lab1_tb.v` and click **OK**.

2-1-5. Click **Finish**.

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

The `lab1_tb.v` file is added under the *Simulation Sources* group, and **AHBLITE_SYS.v** and its lower-level modules are automatically placed in its hierarchy as a *dut* instance.

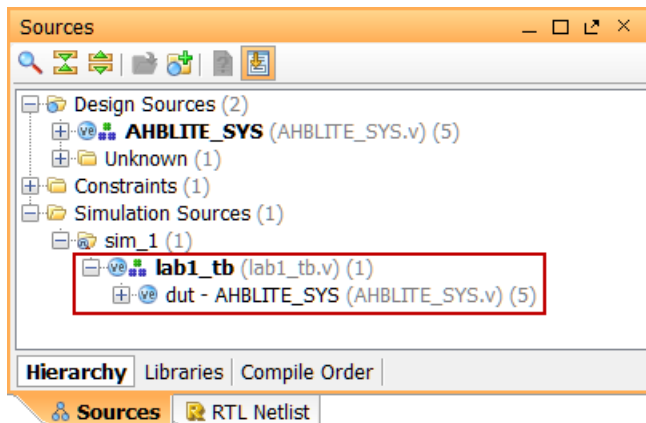


Figure 13. Simulation Sources hierarchy

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.v` 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.

```

1 `timescale 1ns / 1ps
2 ////////////////////////////////////////////////////////////////////
3 // Module Name: lab1_tb
4 ////////////////////////////////////////////////////////////////////
5 module lab1_tb(
6
7 );
8
9     reg RESET, CLK;
10    wire [7:0] LED;
11    wire LOCKUP;
12
13    AHBLITE_SYS dut(.CLK(CLK), .RESET(RESET), .LED(LED), .LOCKUP(LOCKUP));
14
15    initial
16    begin
17        CLK = 0;
18        forever
19        begin
20            #5 CLK = 1;
21            #5 CLK = 0;
22        end
23    end
24
25    initial
26    begin
27        RESET = 0;
28        #30 RESET = 1;
29        #20 RESET = 0;
30    end
31
32 endmodule

```

Figure 14. 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 13 instantiates the DUT (device/module under test). Lines 15 through 23 define the clock signal generation. Lines 25 through 30 define the RESET stimulus generation. Line 32 ends the testbench.

2-2. Add the memory file.

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

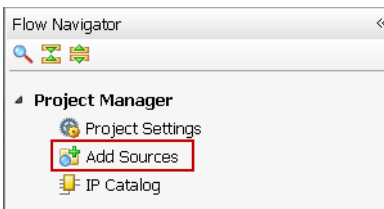
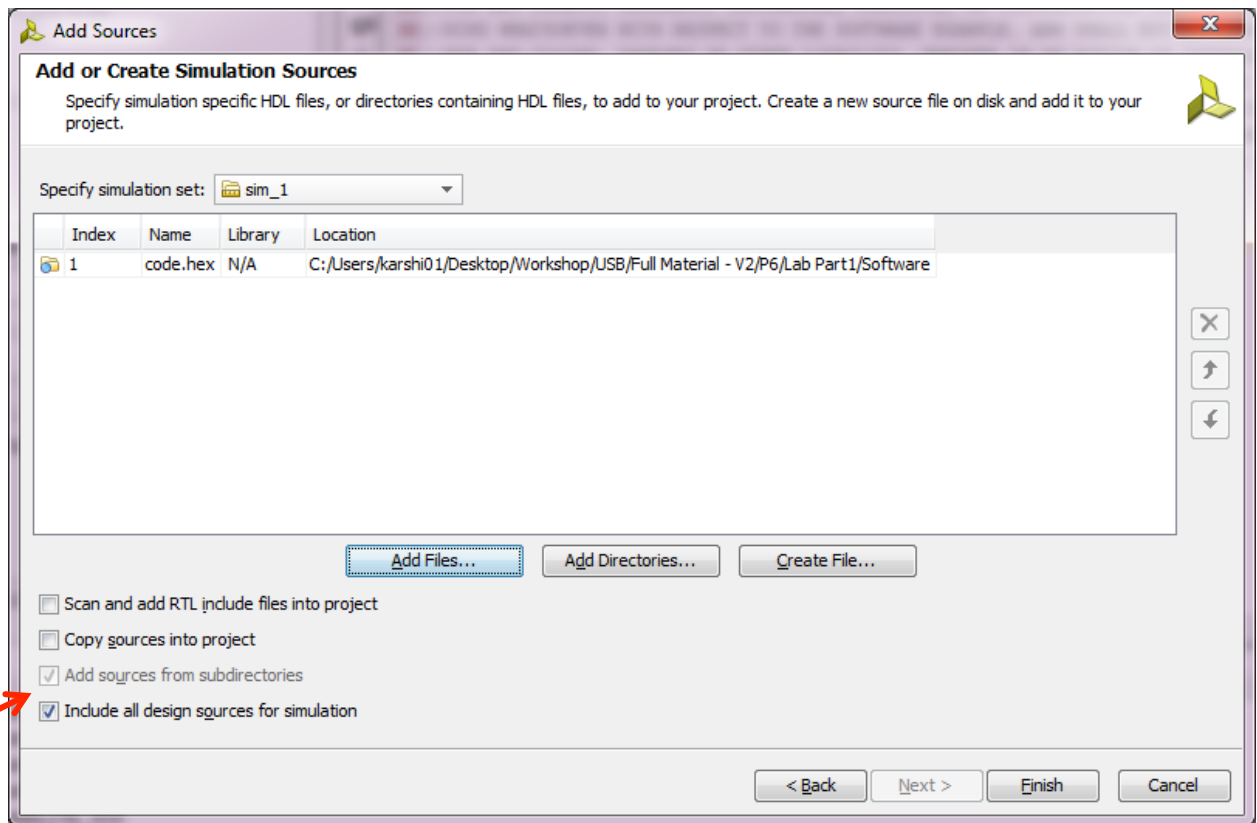


Figure 11. Add Sources

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

2-2-3. Click the **Add Files...** button, select *All Files* in the *Files of type* in the filter field, select code.hex created in the software folder

2-2-4. Make sure you uncheck the box “Copy source into Project” before clicking Finish



2-3. Simulate the design for 1000 ns using the Vivado simulator.

2-3-1. 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.

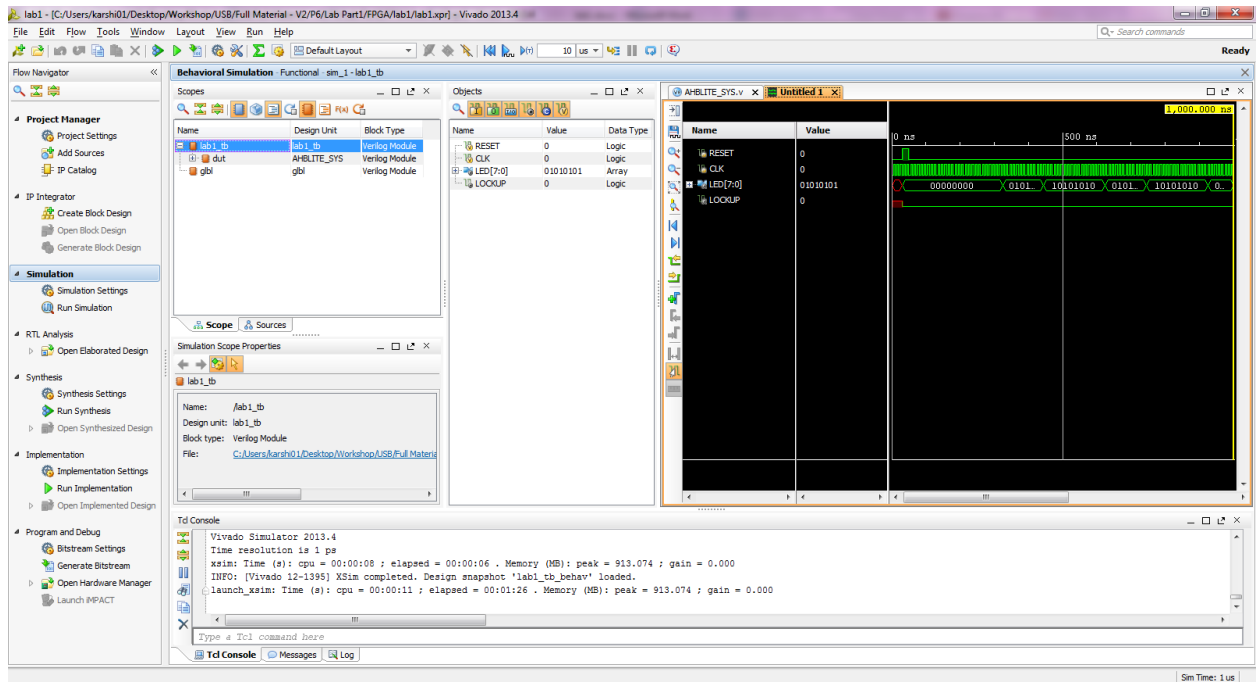


Figure 16. Simulator output

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

2-3-2. Click on the *Zoom Fit* button () to see the entire waveform.

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.

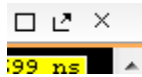


Figure 18. Float Button



Figure 19. Dock Window Button

2-4. Change the display format if desired.

2-4-1. Select the **LED[7:0]** in the waveform window, right-click, select *Radix*, and then select *Hexadecimal*.

2-5. Add more signals to monitor the lower-level signals. and continue to run the simulation for 500 ns.

2-5-1. Expand the **lab1_tb > dut** and select the **u_cortexm0ds** instance in the *Scopes*.

Many signals including AHBLITE interface and internal processor registers will be displayed in the *Objects* window.

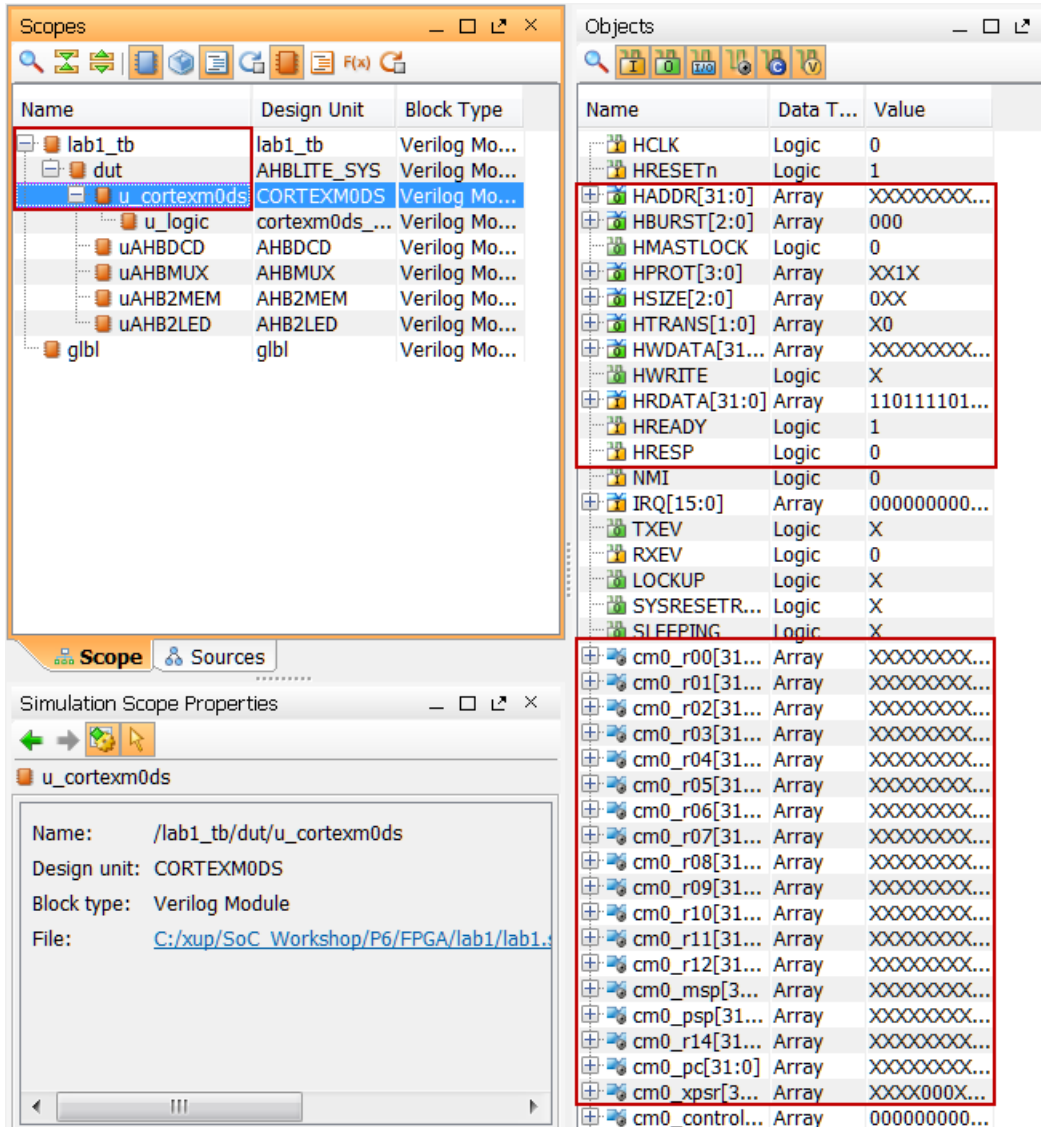


Figure 20. Selecting lower-level signals

- 2-5-2.** Select the objects (H* and cm0_*) as shown in the above figure and drag them into the waveform window to monitor those lower-level signals.
- 2-5-3.** Select the added signals in the waveform window and change their radix to Hexadecimal.
- 2-5-4.** Close the simulator by selecting **File > Close Simulation**.
- 2-5-5.** Click **OK** and then click **Yes** to close it saving the waveform as **untitled_1**.
- 2-5-6.** Click on **Run Simulation > Run Behavioral Simulation**.
- 2-5-7.** Click on the Zoom Fit button and see various signals, bus activities.

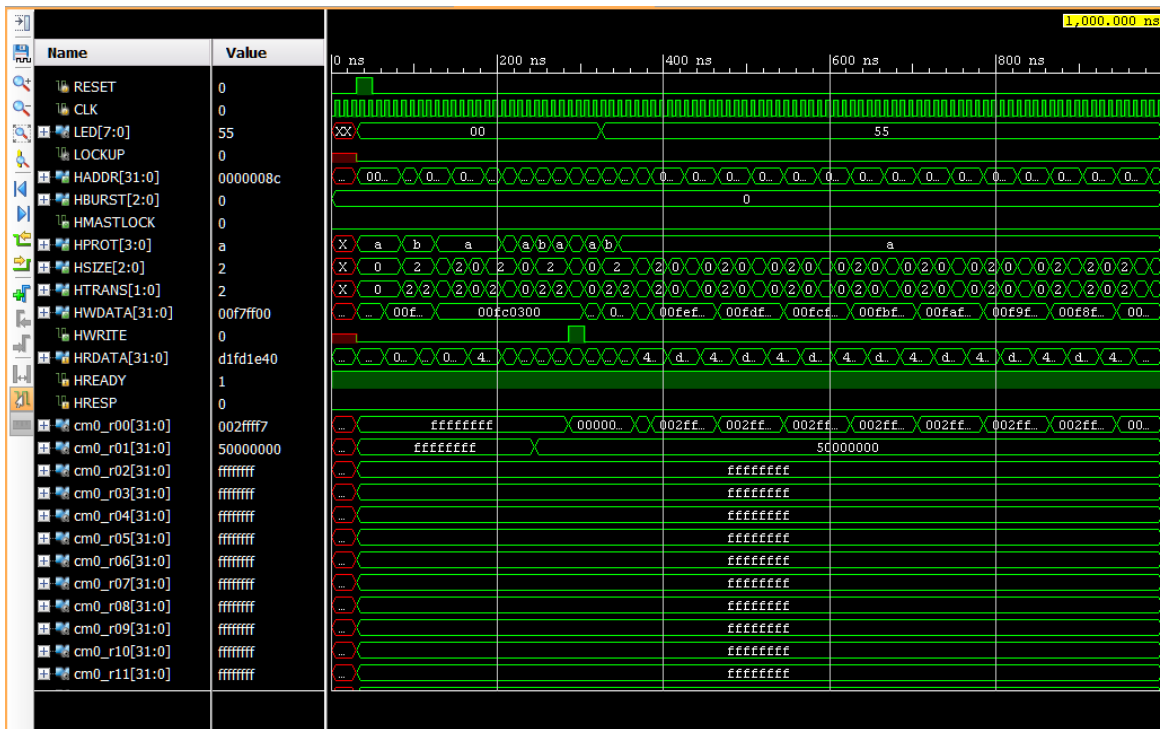
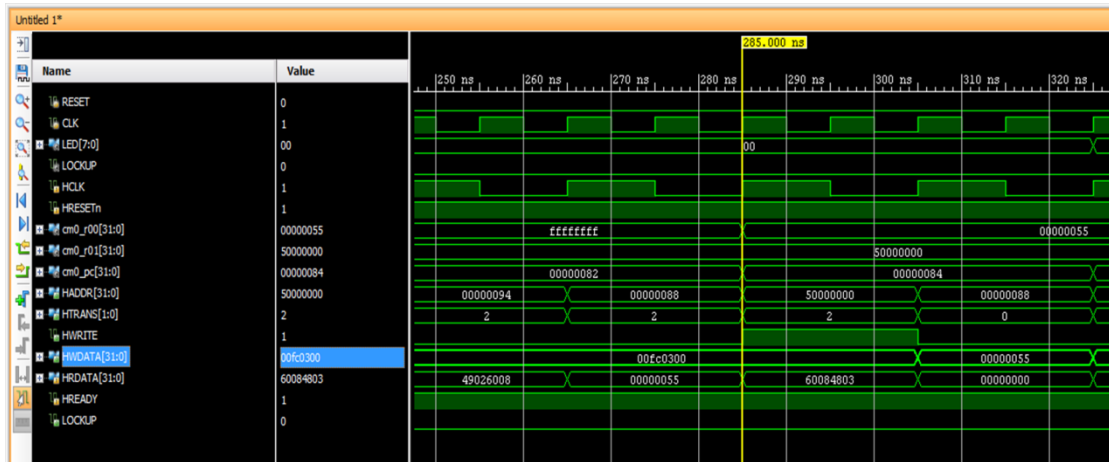


Figure 21. Simulation output

2-5-8. You may want to use zoom in into the simulation window and analyze the output.

2-5-9. Now groups AHB-Lite signals to see the AHB-Lite transactions
 Address & Control: HADDR, HTRANS, HWRITE,
 Write Data: HWDATA,
 Read Data HRDATA,
 Slave Response HREADY

2-5-10. You should now be able to see the AHB-Lite transactions happening like below



Read Transaction Write Transaction

2-5-11. Close the simulator by selecting **File > Close Simulation.**

END OF LAB PART 1

Synthesize the Design

Step 3

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

3-1-1. Make sure you have compiled your software and generated the code.hex in LabPart2/Software folder

3-1-2. Open the project Lab Part2/FPGA/Nexys4/Nexys4.xpr

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

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

3-1-4. Select the *Open Synthesized Design* option and click **OK** as we want to look at the synthesis output before proceeding to the implementation stage.

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

3-1-5. 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 .

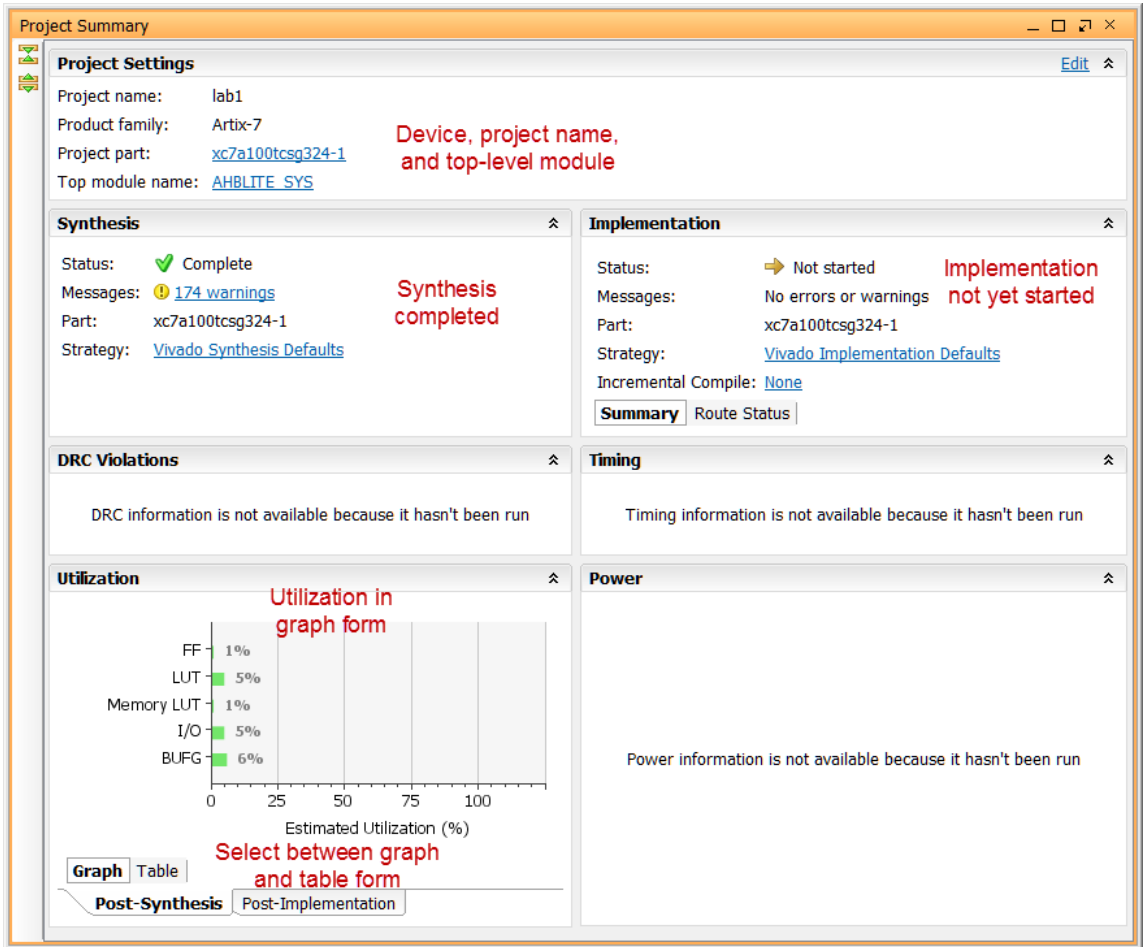


Figure 22. Project Summary view

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

3-1-6. Click on the **Table** tab in the **Project Summary** tab.

Notice that there are an estimated 850 FFs, 3124 LUTs, one BUFG, and 11 IOs (2 input and 9 output) are used.

Resource	Estimation	Available	Utilization %
FF	850	126800	1
LUT	3124	63400	5
Memory LUT	128	19000	1
I/O	11	210	5
BUFG	2	32	6

Graph **Table**

Figure 23. Resource utilization estimation summary

3-1-7. In The *Flow Navigator*, under *Synthesis* (expand *Synthesized Design* if necessary), click on **Schematic** to view the synthesized design in a schematic view.

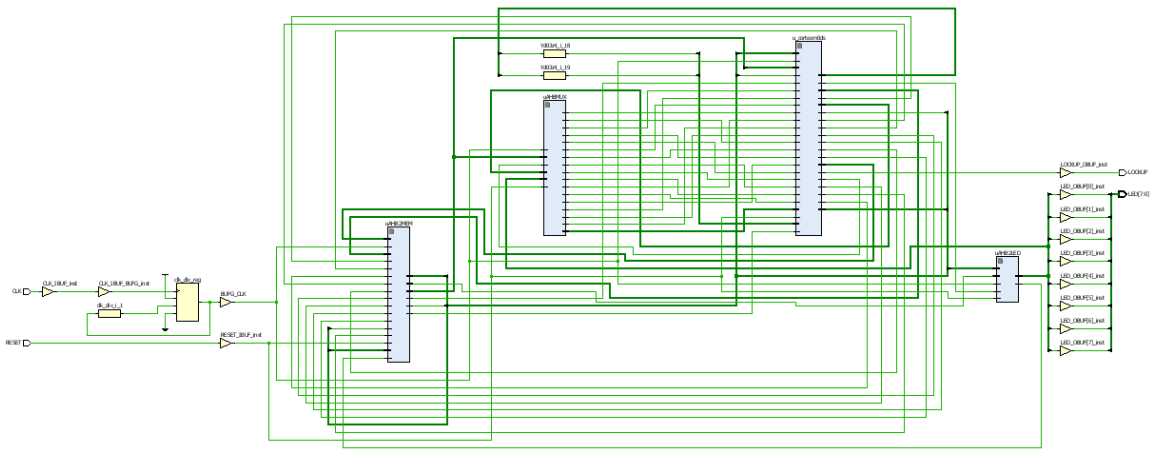


Figure 24. Synthesized design's schematic view

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

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.

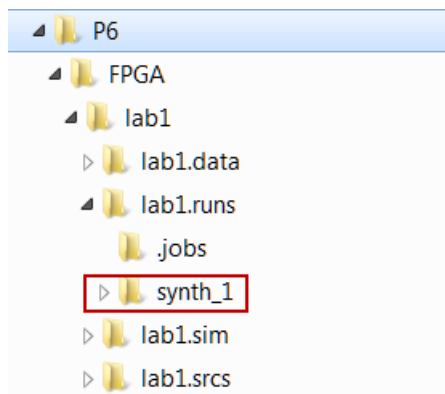


Figure 25. Directory structure after synthesizing the design

Implement the Design

Step 4

4-1. Implement the design with the Vivado Implementation Defaults (Vivado Implementation 2013) settings and analyze the Project Summary output.

4-1-1. 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.

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. Click **OK** to see the device view.

4-1-4. In the *Netlist* pane, select one of the nets (e.g. n_1_uAHB2LED) 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).

4-1-5. If it is not selected, click the *Routing Resources* icon  to show routing resources.

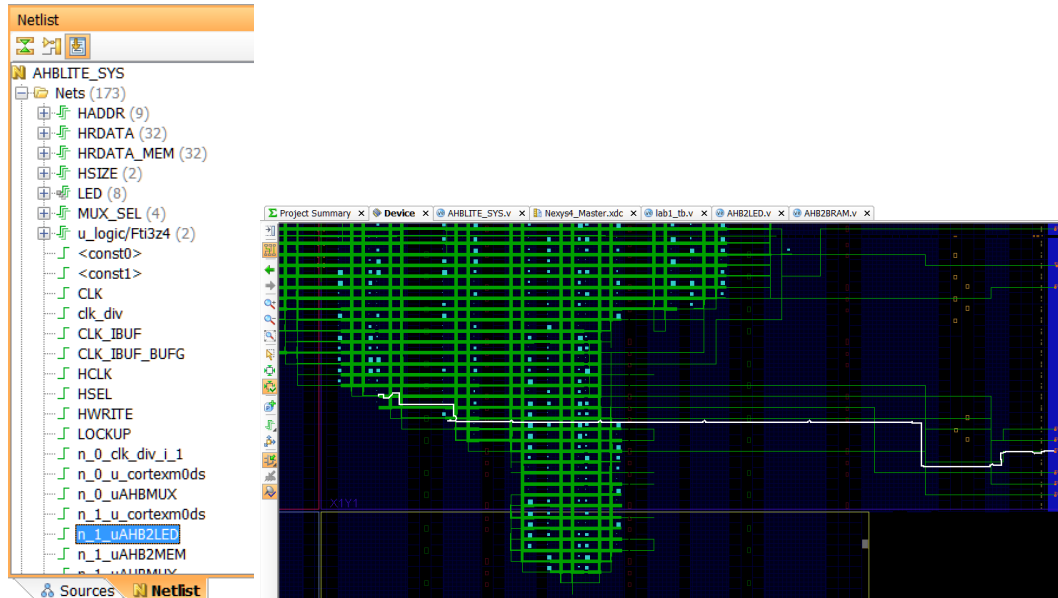


Figure 26. Viewing implemented design

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.

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).

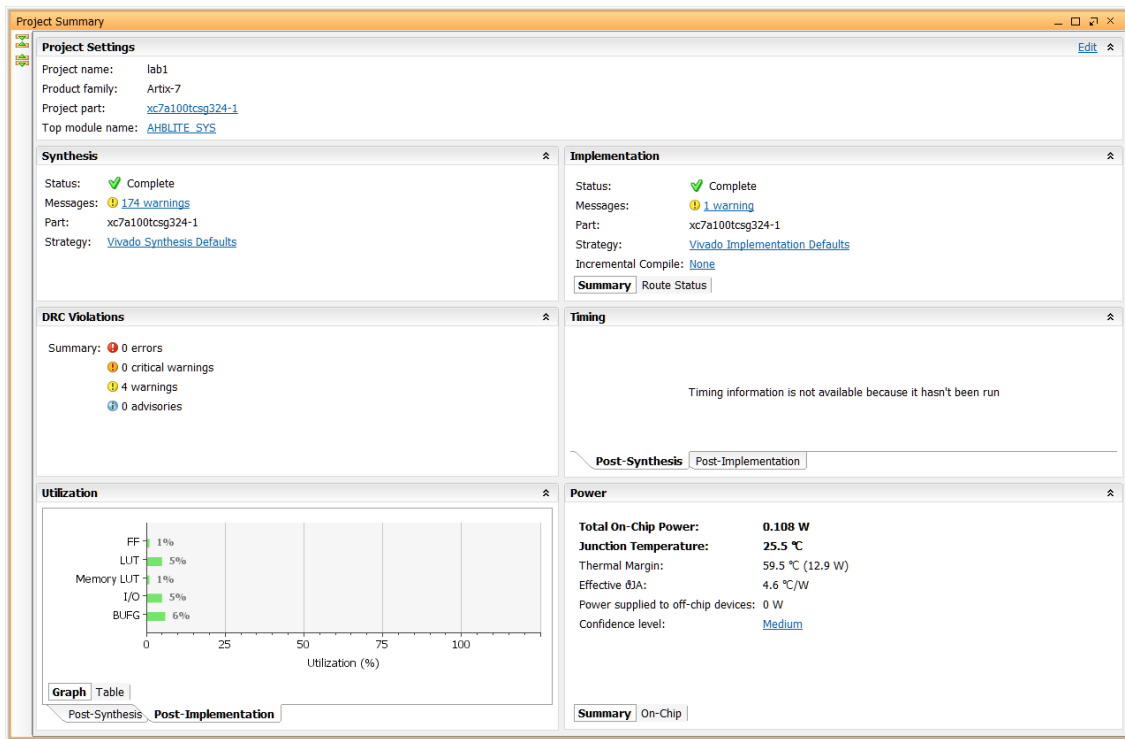
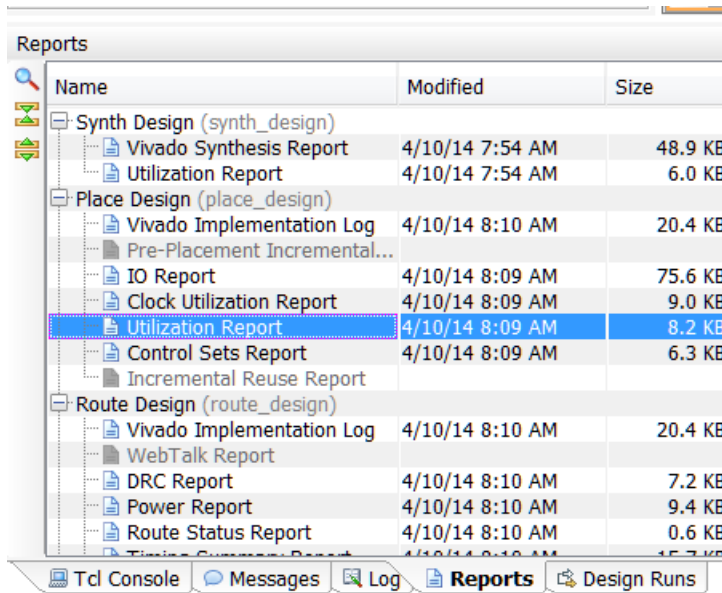


Figure 27. Implementation results

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.

- 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 combinatorial no registers are used.



```

25 10. Instantiated Netlists
26
27 1. Slice Logic
28 -----
29
30 +-----+-----+-----+-----+-----+
31 |           Site Type           | Used | Loded | Available | Util% |
32 +-----+-----+-----+-----+-----+
33 | Slice LUTs                    | 3027 | 0     | 63400    | 4.77  |
34 |   LUT as Logic                | 2899 | 0     | 63400    | 4.57  |
35 |   LUT as Memory                | 128  | 0     | 19000    | 0.67  |
36 |     LUT as Distributed RAM     | 128  | 0     |           |       |
37 |     LUT as Shift Register      | 0    | 0     |           |       |
38 | Slice Registers                | 850  | 0     | 126800   | 0.67  |
39 |   Register as Flip Flop        | 850  | 0     | 126800   | 0.67  |
40 |   Register as Latch            | 0    | 0     | 126800   | 0.00  |
41 | F7 Muxes                       | 73   | 0     | 31700    | 0.23  |
42 | F8 Muxes                       | 32   | 0     | 15850    | 0.20  |
43 +-----+-----+-----+-----+-----+
44

```

Figure 28. Viewing utilization report

Generate the Bitstream and Verify Functionality

Step 5

5-1. Connect the board and power it ON. Generate the bitstream, open a hardware session, and program the FPGA.

5-1-1. Make sure that the Micro-USB cable is connected to the JTAG PROG connector (next to the power supply connector).

5-1-2. Make sure that the JP3 is set to select USB power.

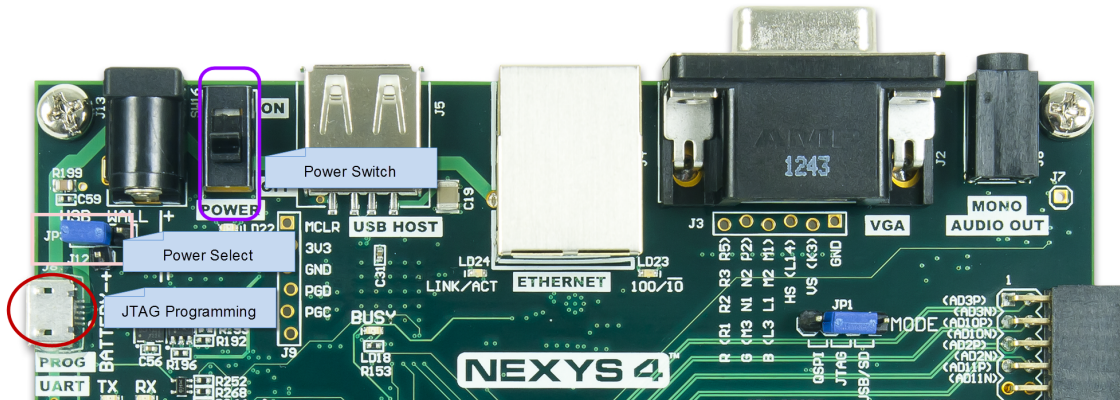


Figure 29. Board connection

5-1-3. Power ON the switch on the board.

5-1-4. Click on the **Generate Bitstream** entry under the *Program and Debug* tasks of the *Flow Navigator* pane.

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

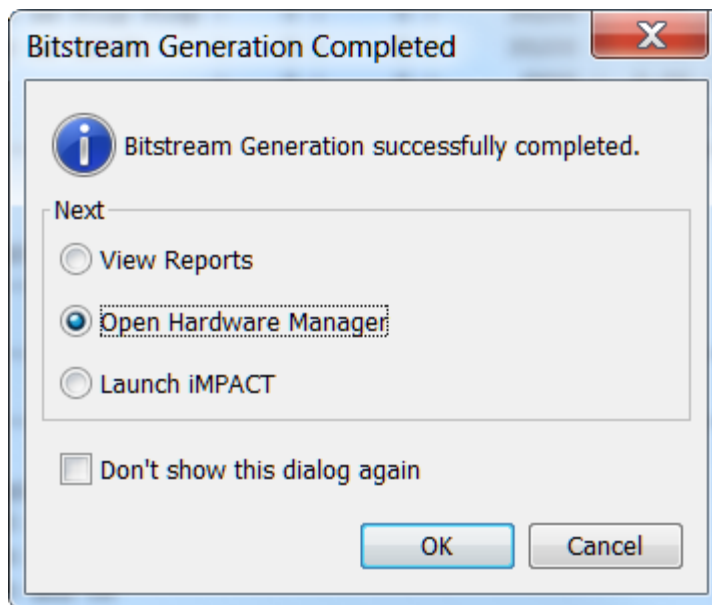


Figure 30. Bitstream generation

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

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

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

5-1-6. 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.

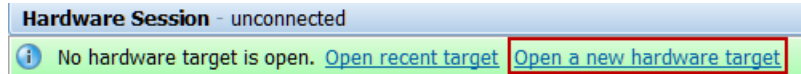


Figure 31. Opening new hardware target

5-1-7. Click **Next** to see the Vivado CSE Server Name form.

5-1-8. Click **Next** with the localhost port 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.

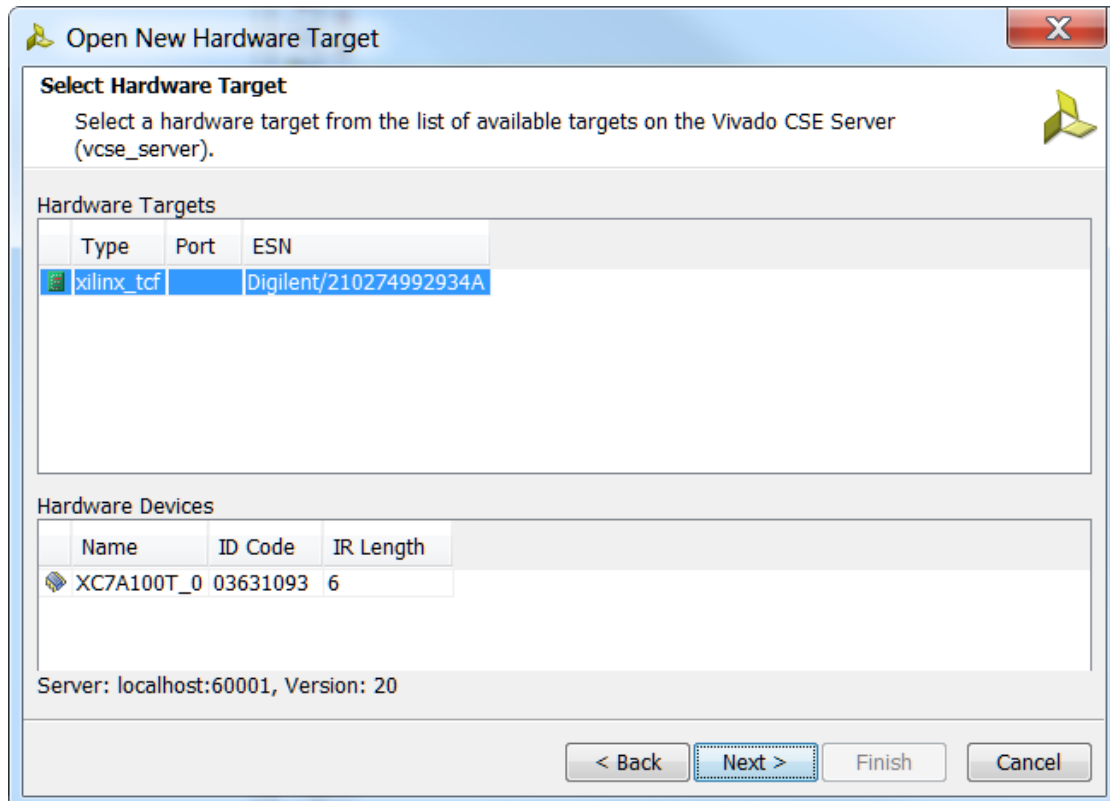


Figure 32. New hardware target detection

5-1-9. Click **Next** twice 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.

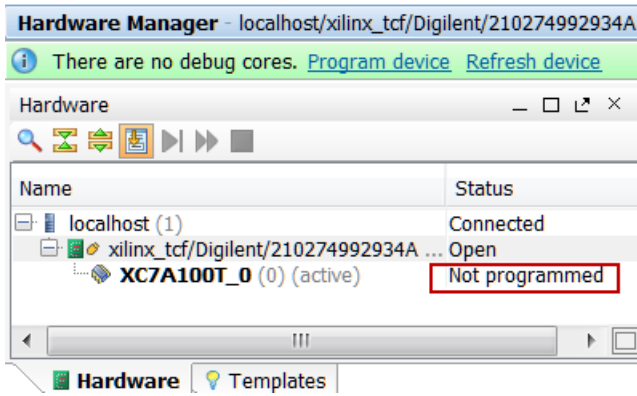


Figure 33. Opened hardware session

5-1-10. Select the device and verify that the AHBLITE_SYS.bit is selected as the programming file in the General tab.

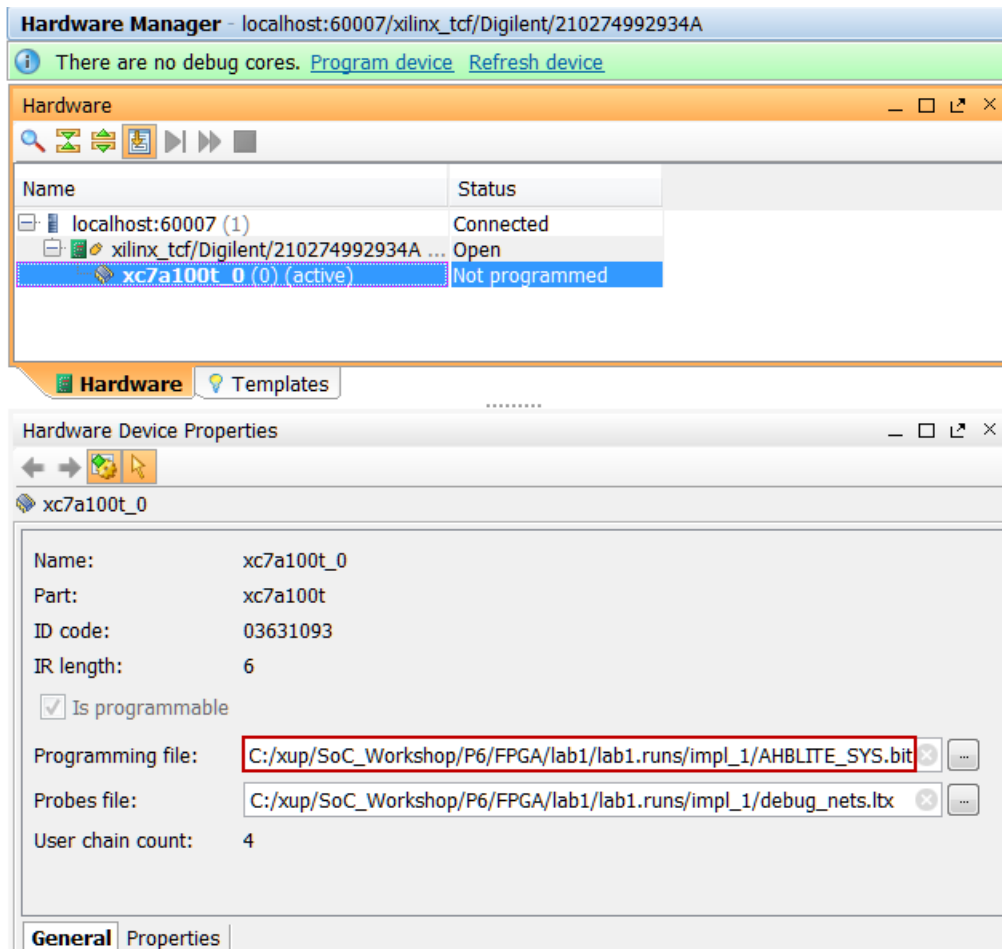


Figure 34. Programming file

5-1-11. Right-click on the device and select *Program Device...* or click on the *Program device* > *XC7A100T_0* link to program the target FPGA device.

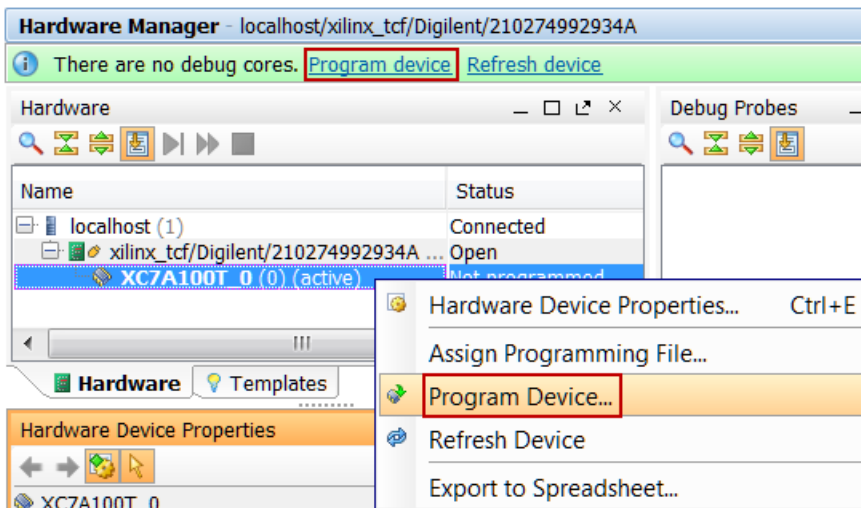


Figure 35. Selecting to program the FPGA

5-1-12. Click **OK** to program the FPGA.

The DONE light will light when the device is programmed. You may see LEDs toggling if the SW15 switch position is OFF (down position).

5-1-13. Turn ON the SW15 switch (up position) which will put the CPU in a reset state and you won't see the LEDs toggling.

5-1-14. Turn OFF the SW15 switch which will put the CPU in a run state and you will see the LEDs toggling.

5-1-15. When satisfied, power **OFF** the board.

5-1-16. Close the hardware session by selecting **File > Close Hardware Manager**.

5-1-17. Click **OK** to close the session.

5-1-18. 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 functionality was verified in hardware using the generated bitstream.