Vivado Tutorial Using IP Integrator

Introduction

This tutorial guides you through the design flow using Xilinx Vivado software to create a simple digital circuit using Vivado IP Integrator (IPI). A typical design flow consists of creating a Vivado project, optionally setting a user-defined IP library settings, creating a block design using various IP, creating a HDL wrapper, creating and/or adding user constraint file(s), optionally running behavioral simulation, synthesizing the design, implementing the design, generating the bitstream, and finally verifying the functionality in the hardware by downloading the generated bitstream file. You will go through the typical design flow targeting the Artix-100t based Nexys4 or Artix-35t based Basys3 board.

Objectives

After completing this tutorial, you will be able to:

- Create a Vivado project targeting a specific FPGA device located on the Nexys4 or Basys3 board
- Use the provided partially completed Xilinx Design Constraint (XDC) file to constrain some of the pin locations
- Add additional constraints using the Tcl scripting feature of Vivado
- Simulate the design using the XSim simulator
- Synthesize and implement the design
- Generate the bitstream
- Configure the FPGA using the generated bitstream and verify the functionality

Procedure

This tutorial is broken into steps that consist of general overview statements providing information on the detailed instructions that follow. Follow these detailed instructions to progress through the tutorial.

Design Description

The design consists of some inputs directly connected to the corresponding output LEDs. Other inputs are logically operated on before the results are output on the remaining LEDs as shown in **Figure 1**.

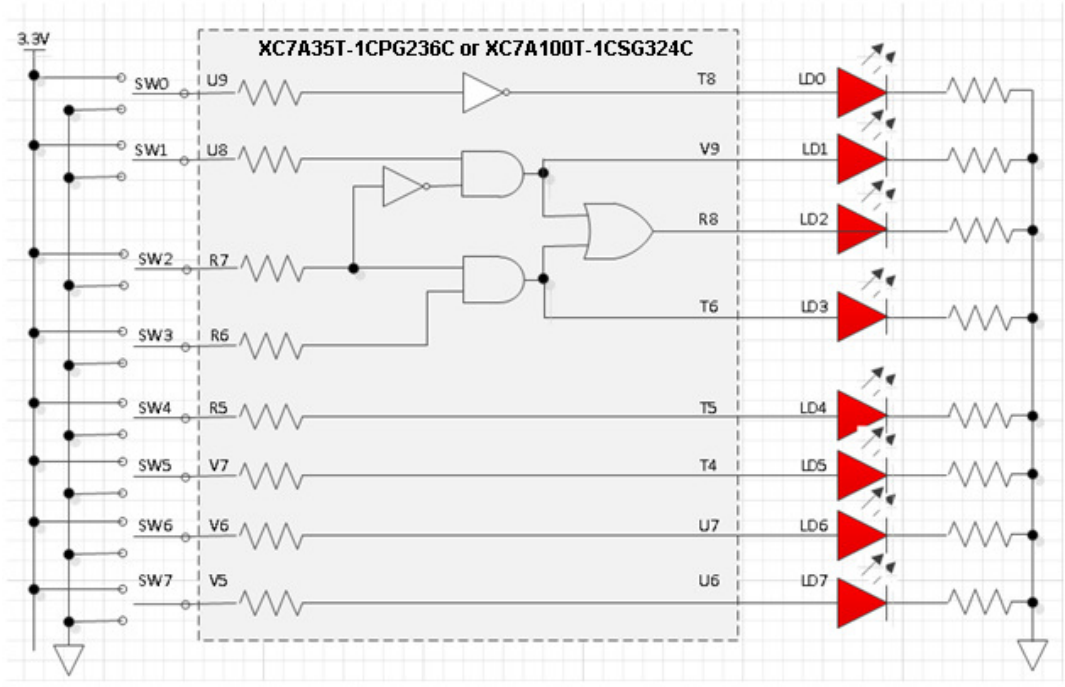

Figure 1. Completed Design

General Flow for this tutorial

- Create a Vivado project and set IP library setting
- Create a block design
- Create a HDL wrapper and add the provided constraint file
- Simulate the design using XSim simulator
- Synthesize the design
- Implement the design
- Perform the timing simulation
- Verify the functionality in hardware using the target board

Create a Vivado Project using IDE — Step 1

1-1. Launch Vivado and create a project targeting either the *Nexys4* or the *Basys3* and using the Verilog HDL. Use the provided Verilog source files and *tutorial_nexys4.xdc* (for Nexys4) file or *tutorial_basys3.xdc* (for Basys3) file from the *<2014_2_artix7_sources>* directory.

References to <2014_2_artix7_labs> means c:\xup\digital\2014_2_artix7_labs and <2014_2_artix7_sources> means c:\xup\digital\2014_2_artix7_sources directories.

1-1-1. Open Vivado by selecting **Start > All Programs > Xilinx Design Tools > Vivado 2014.2 > Vivado 2014.2**

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 **<2014_2_artix7_labs>**, and click **Select**.

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

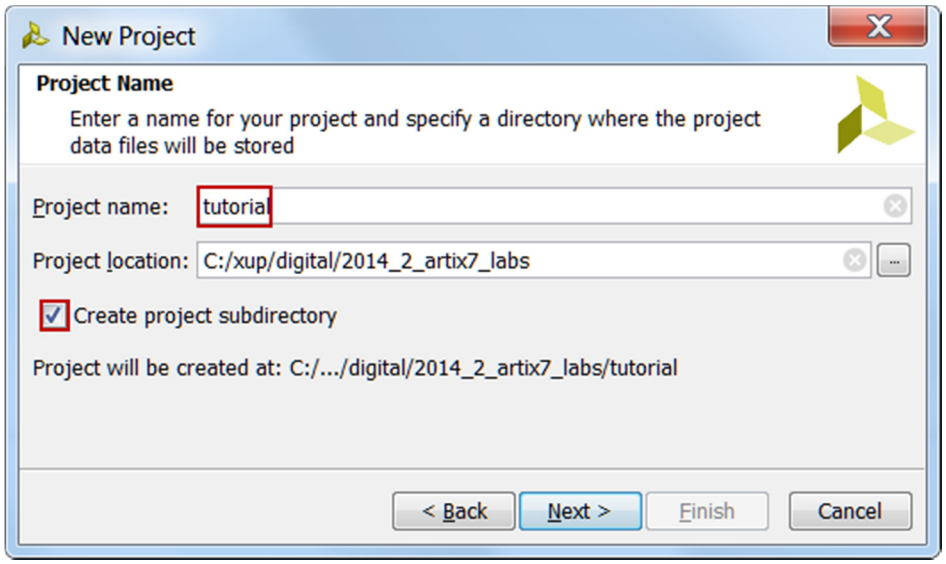

Figure 2. Project Name and Location entry

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

1-1-6. Select **Verilog** as the *Target language* and *Simulator language* in the *Add Sources* form.

1-1-7. Click **Next**.

1-1-8. Click **Next** to get to the *Add Cons*traints form.

1-1-9. Select constraints file entries, if displayed, and use 'X' button on the right to remove it**.**

This Xilinx Design Constraints file assigns the physical IO locations on FPGA to the switches and LEDs located on the board. This information can be obtained either through a board's schematic or board's user guide. We will add the file later.

1-1-10. In the *Default Part* form, using the **Parts** option and various drop-down fields of the **Filter** section, select the **XC7A100TCSG324-1** part (for Nexys4) or the **XC7A35TCPG236-1** part (for Basys3). Click **Next**.

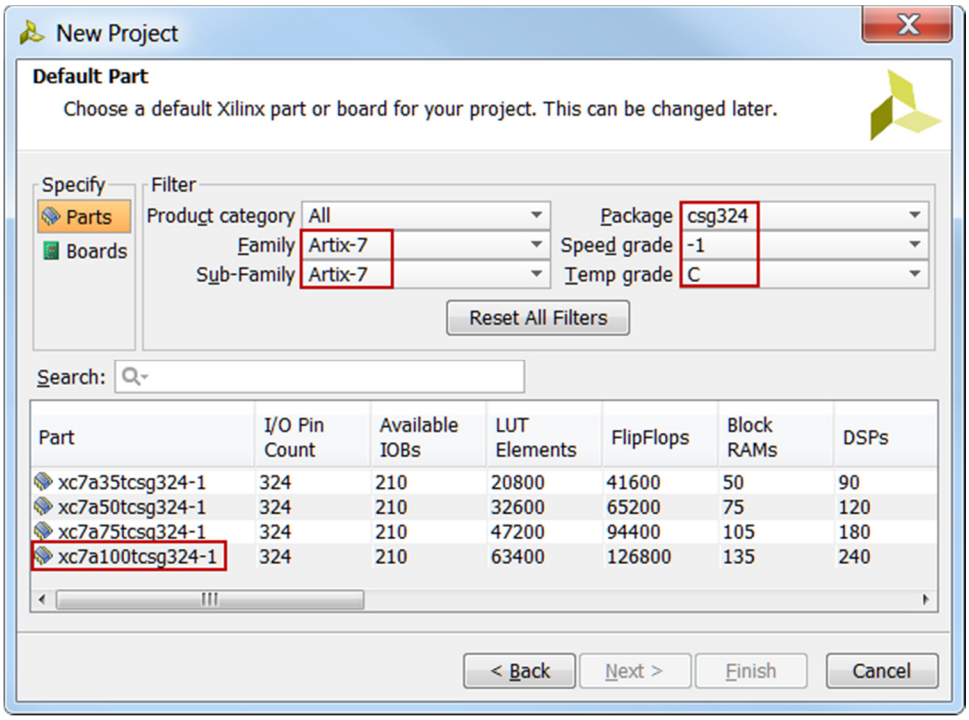

Figure 3. Part selection for Nexys4

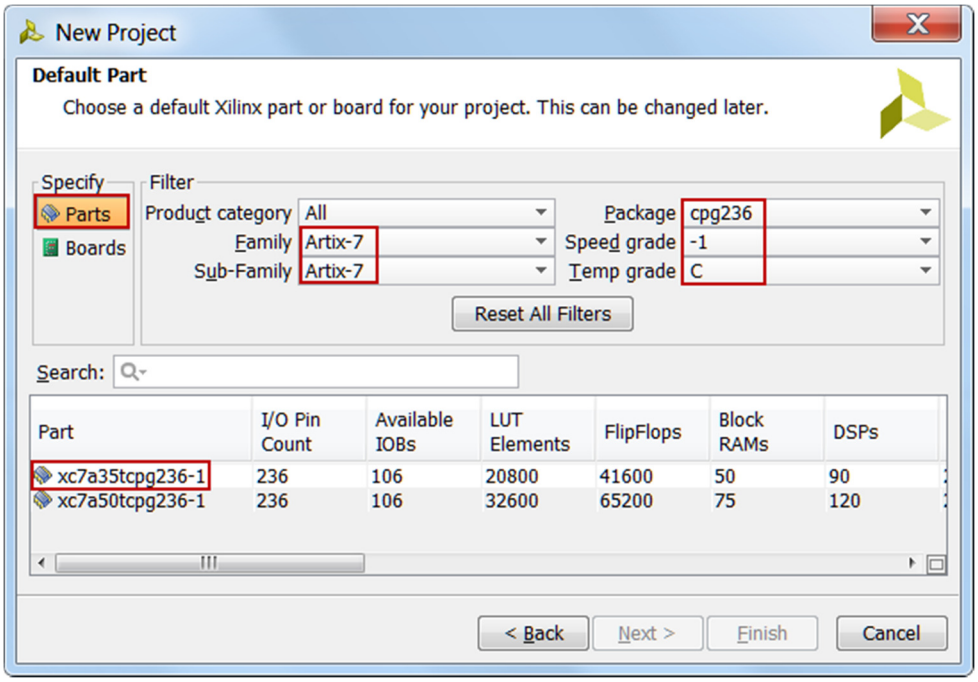

Figure 3. Part selection for Basys3

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

Use the Windows Explorer and look at the **<2014_2_artix7_labs>\tutorial** directory. You will find that the **tutorial.cache** directory and the **tutorial.xpr** (Vivado) project file.

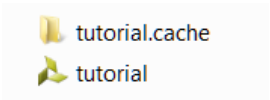

Figure 4. Generated directory structure

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

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

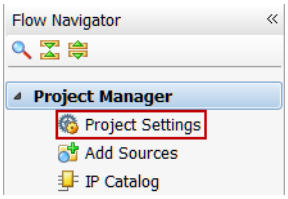

Figure 5. Invoking Project Settings to set IP repository path

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

1-2-3. Click on the **Add Repositories** button, browse to **<2014_2_artix7_sources>** and select **XUP_LIB** directory, and click **Select**.

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

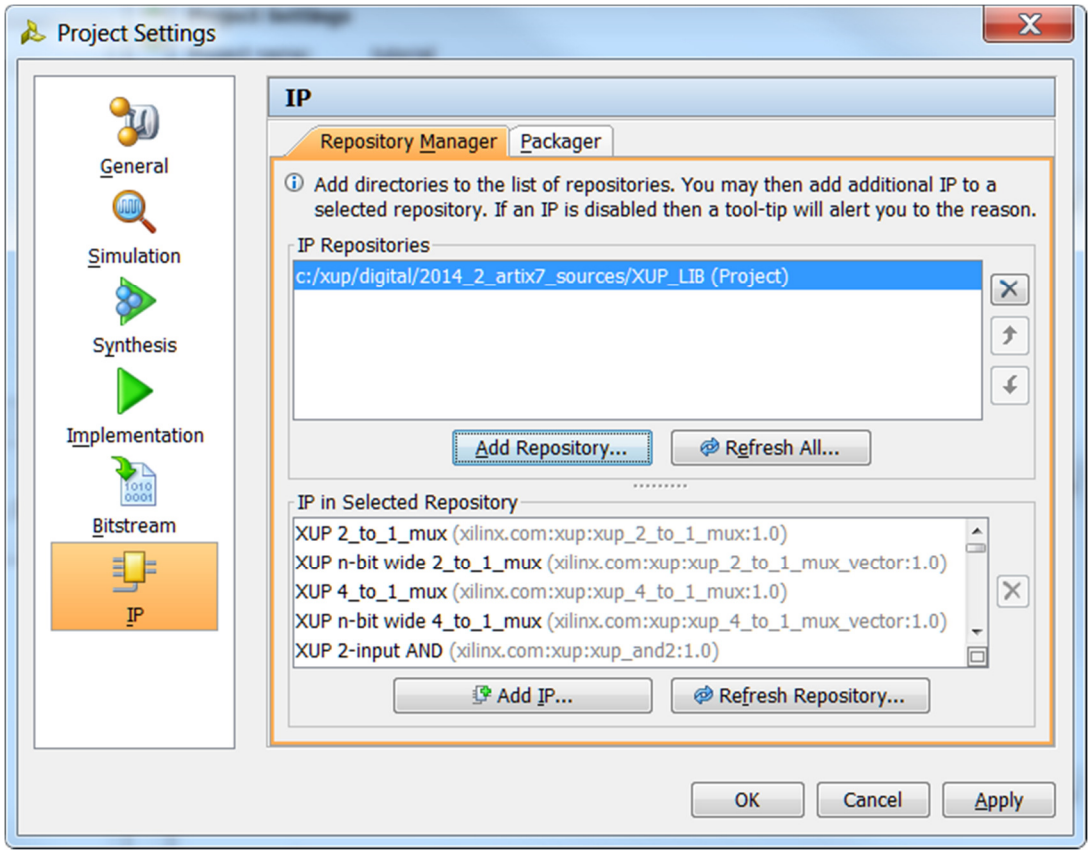

Figure 6. Setting IP Repository

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

Create a Block Design Step 2

2-1. Create a block design.

2-1-1. In the *Flow Navigator* window, click on **Create Block Design** under the IP Integrator block.

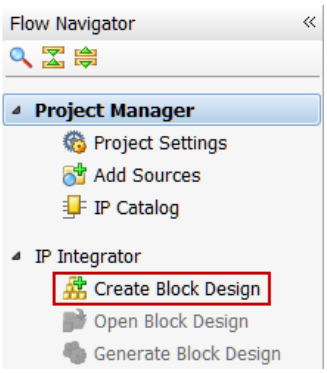

Figure 7. Invoking IP Integrator to create a block design

2-1-2. Click **OK** to create a block design named *design_1*.

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

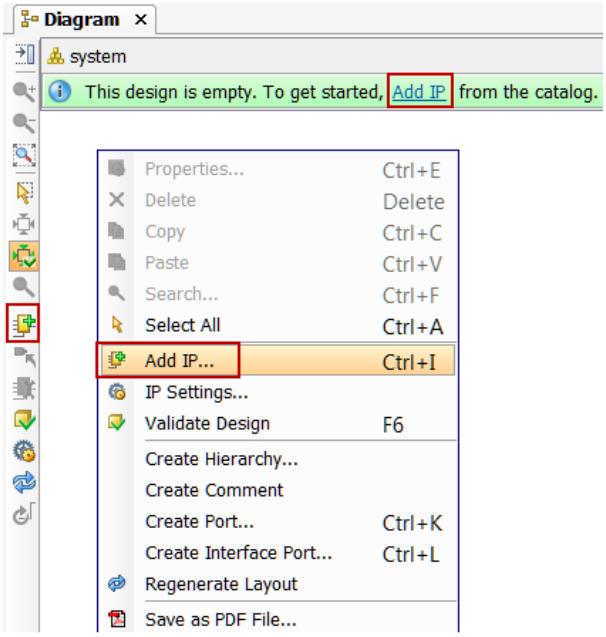

Figure 8. Add IP to Block Diagram

2-1-4. Once the IP Catalog is open, type "inv" into the Search bar, find and double click on **XUP 1-input INV** entry, or click on the entry and hit the Enter key to add it to the design.

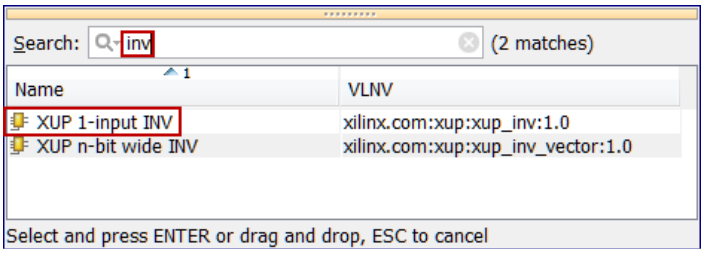

Figure 9. Add an inverter to the design

2-1-5. Similarly, another instance of an inverter.

2-1-6. Add two instances of 2-input AND gate and an instance of 2-input OR gate.

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

2-1-7. Redraw the diagram, by clicking on the re-draw ( ) button. At this stage the block diagram should look like shown below.

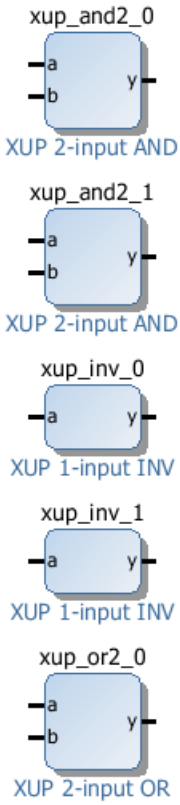

Figure 10. Added necessary instances

2-2. Complete the design.

2-2-1. Right-click on the **xup_inv_0** instance's input port and select **Make External**. Similarly, make the output port of the same instance and make it external.

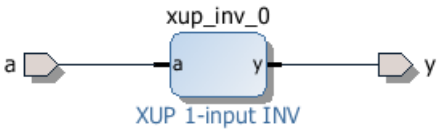

Figure 11. Making ports external

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

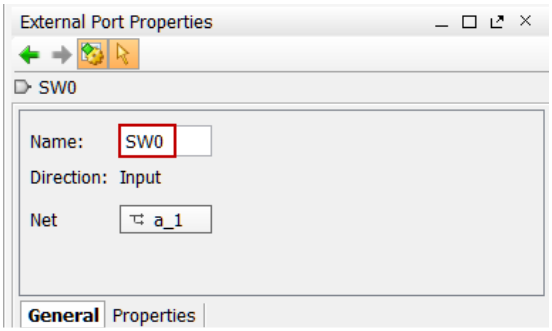

Figure 12. Setting input port name to SW0

2-2-3. Similarly, change the output port *y* to **LD0** (as per the diagram in Figure 1).

2-2-4. Arrange OR2 instance such that it is close to the two instances of the AND2.

2-2-5. Arrange the second instance of the inverter on the left of one of the AND2 gate.

2-2-6. Using the left-button of the mouse, draw a connection between the outputs of the AND2 instances and the two input of the OR2.

When you move the mouse closer to a port, the cursor becomes drawing pencil icon. Click the left-button of the mouse and keeping the button pressed draw it towards the destination port. You make a connection this way.

2-2-7. Similarly, connect the output of the inverter to one input of one of the AND2 instances.

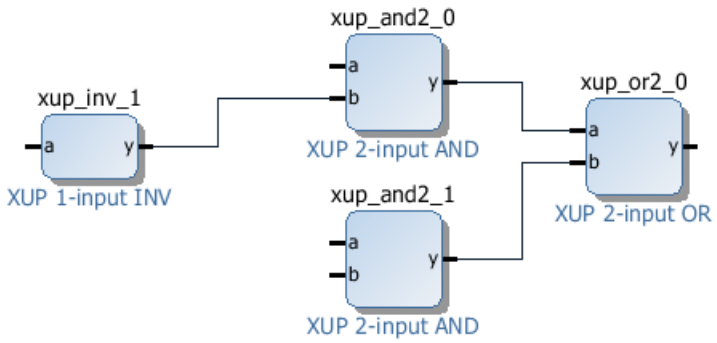

Figure 13. Connecting instances

This diagram is similar to the logic connected between SW1, SW2, SW3, and LD2.

2-2-8. Make input ports of the **xup_inv_1**, *a* port of the **xup_and2_0**, and *b* port of the **xup_and2_1** instances external.

2-2-9. Similarly, make the output port of the **xup_or2_0** instance external.

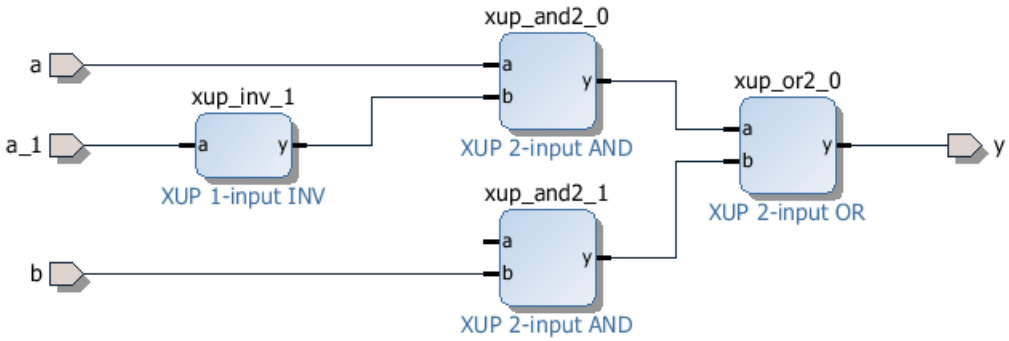

Figure 14. Making ports external

2-2-10. Change the name of *a* to **SW1**, *a_1* to **SW2**, *b* to **SW3**, and *y* to **LD2**.

2-2-11. Right-click somewhere on the canvas and select Create Port.

A Create Port form will appear.

2-2-12. Enter **LD1** as the port name, using the drop-down button select the type as *output*, and click **OK**.

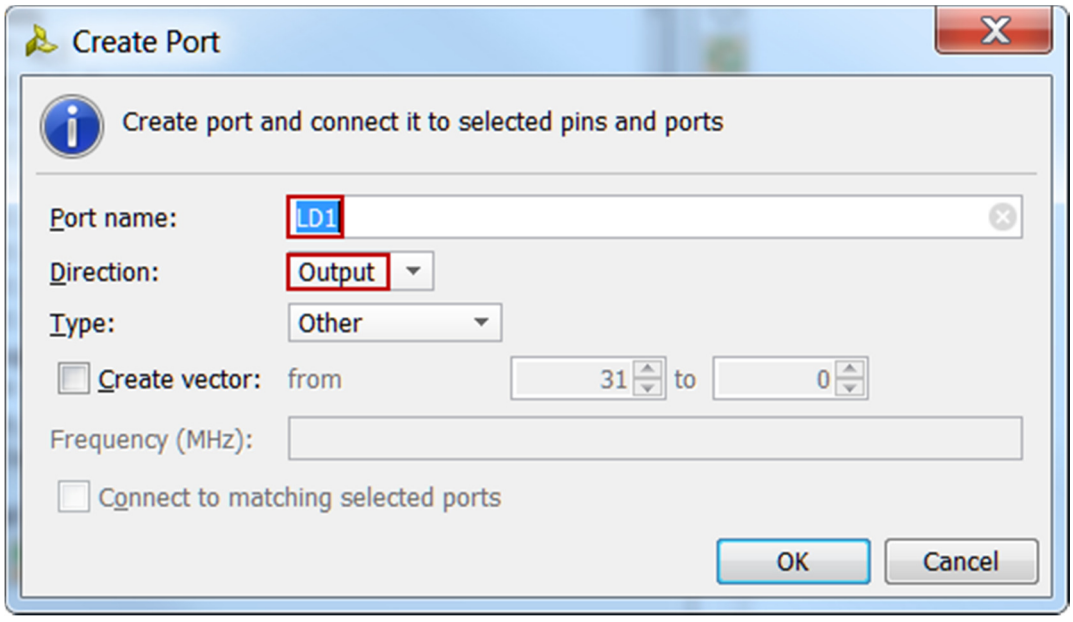

Figure 15. Creating an output port

2-2-13. Similarly, create the output port naming it as **LD3**.

2-2-14. Connect the input port *a* of the **xup_and2_1** instance to output port of the instance **xup_inv_1**.

2-2-15. Connect the output port of the **xup_and2_0** to **LD1** and **xup_and2_1** to **LD3**. Click on the re-draw button.

The diagram will look similar to shown below.

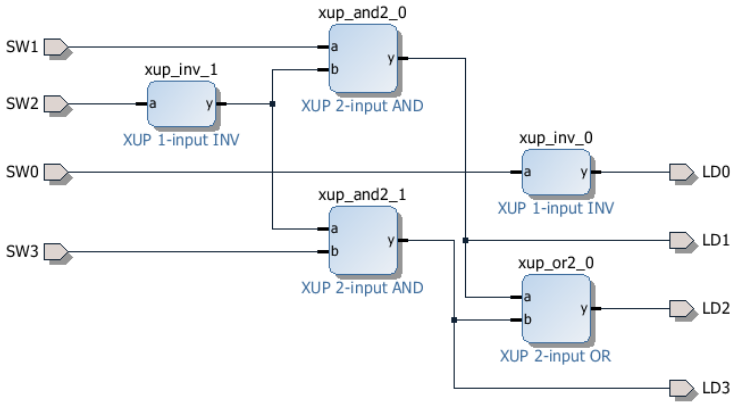

Figure 16. Partially completed design

2-3. Complete the design including rest of the switches and LDs

2-3-1. Right-click on the canvas and create an input port *SW4*.

2-3-2. Similarly, create *SW5*, *SW6*, and *SW7* as input ports, and *LD4*, *LD5*, *LD6*, and *LD7* as output ports.

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

2-3-4. Click the re-draw button.

The design should look like as shown below.

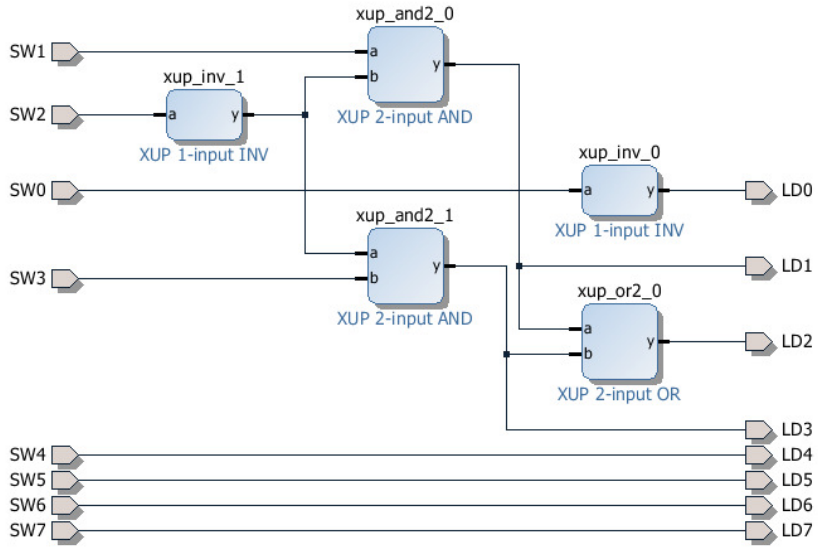

Figure 17. The completed design

2-3-5. Select **File > Save Block Design.**

Create HDL Wrapper and Add a Constraint File Step 3

3-1. Create a HDL wrapper and analyze the hierarchy

3-1-1. In the *sources* view, Right Click on the block diagram file, **design_1.bd**, and select **Create HDL Wrapper** to create the HDL wrapper file. When prompted, select **Let Vivado manage wrapper and auto-update**, click **OK.**

3-1-2. In the *Sources* pane, expand the hierarchy.

Notice the design_1_wrapper file instantiates design_1 which in turn instantiates the inverter twice, and2 twice, and or2 once.

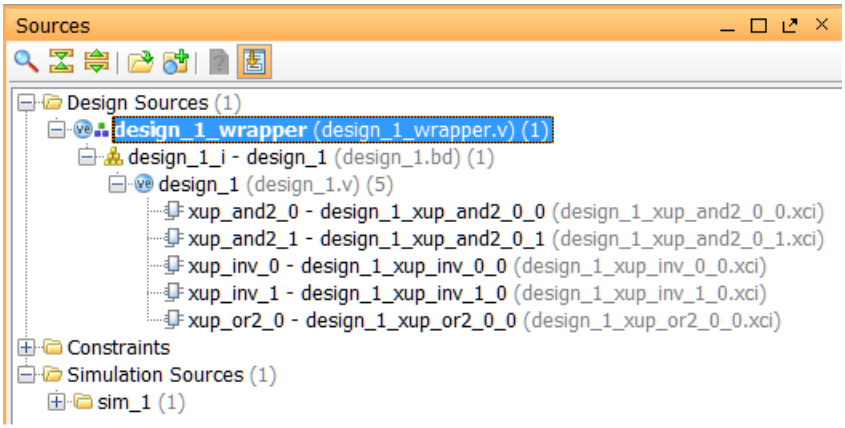

Figure 18. Hierarchical design

3-1-3. Double-click the **design_1_wrapper.v** entry to open the file in text mode and observe the instantiation of the *design_1* module.

3-1-4. Double-click the **design_1.v** entry to open the file in text mode and observe the instantiation of the lower-level modules.

3-2. Add tutorial_nexys4.xdc (for Nexys4) or tutorial_basys3 (for Basys3) constraints source and analyze the content.

3-2-1. Click on the **Add Sources** under the *Project Manager* group in the *Flow Navigator* window.

3-2-2. Select the **Add or Create Constraints** option and click **Next**.

3-2-3. Click **Add Files…** and browse to **<2014_2_artix7_sources>\tutorial.**

3-2-4. Select **tutorial_nexys4.xdc** (for Nexys4) or **tutorial_basys3.xdc** (for Basys3) and click **OK**.

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

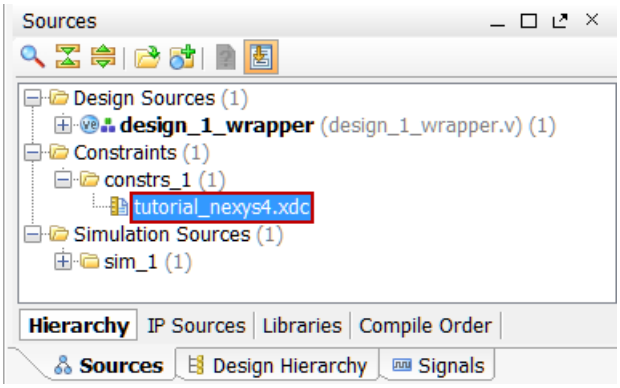

Figure 19. Constraints file added for Nexys4

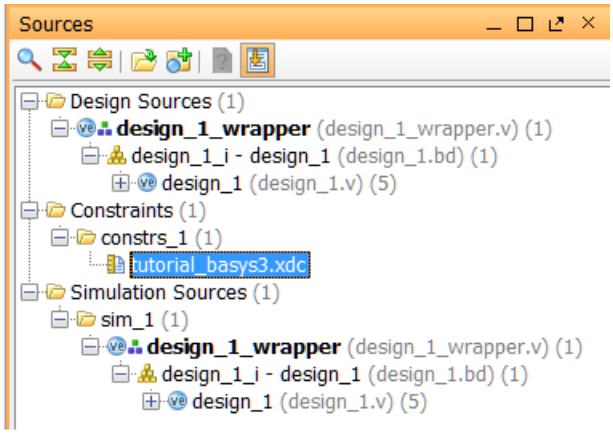

Figure 19. Constraints file added for Basys3

3-2-6. In the *Sources* pane, expand the *Constraints* folder and double-click the **tutorial_nexys4.xdc** or **tutorial_basys3.xdc** entry to open the file in text mode.

3-2-7. Lines 2-15 define the pin locations of the input switches [6:0] and lines 17-30 define the pin locations of the output LEDs [6:0]. The SW7 and LD7 are deliberately not defined so you can learn how to enter them using other methods.

3-3. Perform RTL analysis on the source file.

3-3-1. Expand the *Open Elaborated Design* entry under the *RTL Analysis* tasks of the *Flow Navigator* pane and click on **Schematic**.

3-3-2. Click **Save** if asked.

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

3-3-3. Click on the **+** sign inside the block to see its content. Use the *Zoom Full* ( ) button.

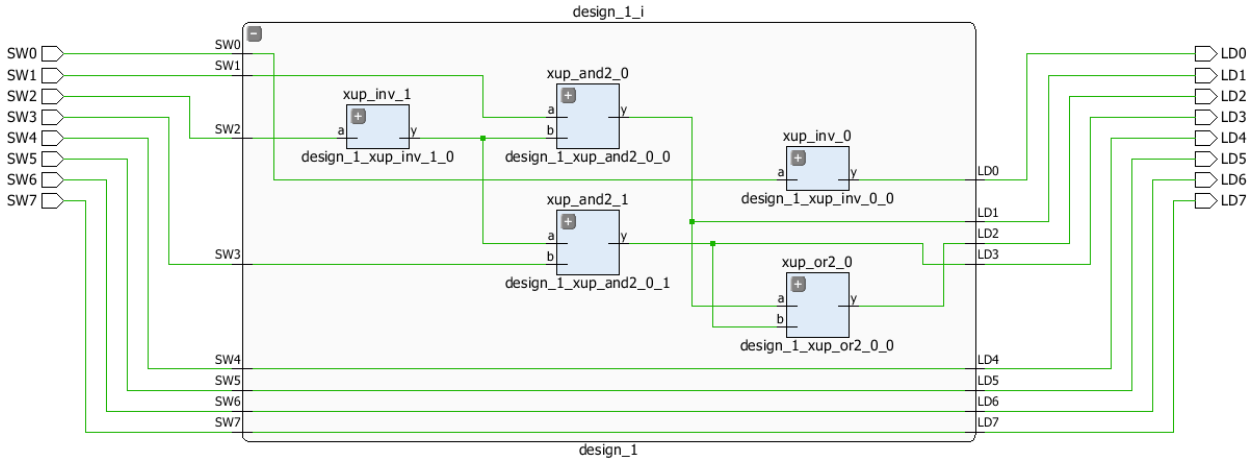

Figure 20. A logic view of the design

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.

3-4. Add I/O constraints for the missing LED and switch pins.

3-4-1. Once RTL analysis is performed, another standard layout called the *I/O Planning* is available. Click on the drop-down button and select the *I/O Planning* layout.

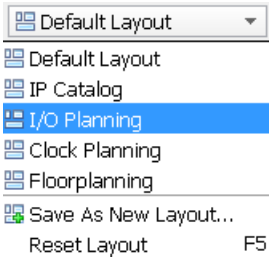

Figure 21. I/O Planning layout selection

Notice that the Package view is displayed in the Auxiliary View area, Device Constraints tab is selected, and I/O ports tab is displayed in the Console View area. Also notice that design ports (LD* and SW*) are listed in the I/O Ports tab with both having multiple I/O standards.

Move the mouse cursor over the Package view, highlighting different pins. Notice the pin site number is shown at the bottom of the Vivado GUI, along with the pin type (User IO, GND, VCCO…) and the I/O bank it belongs to.

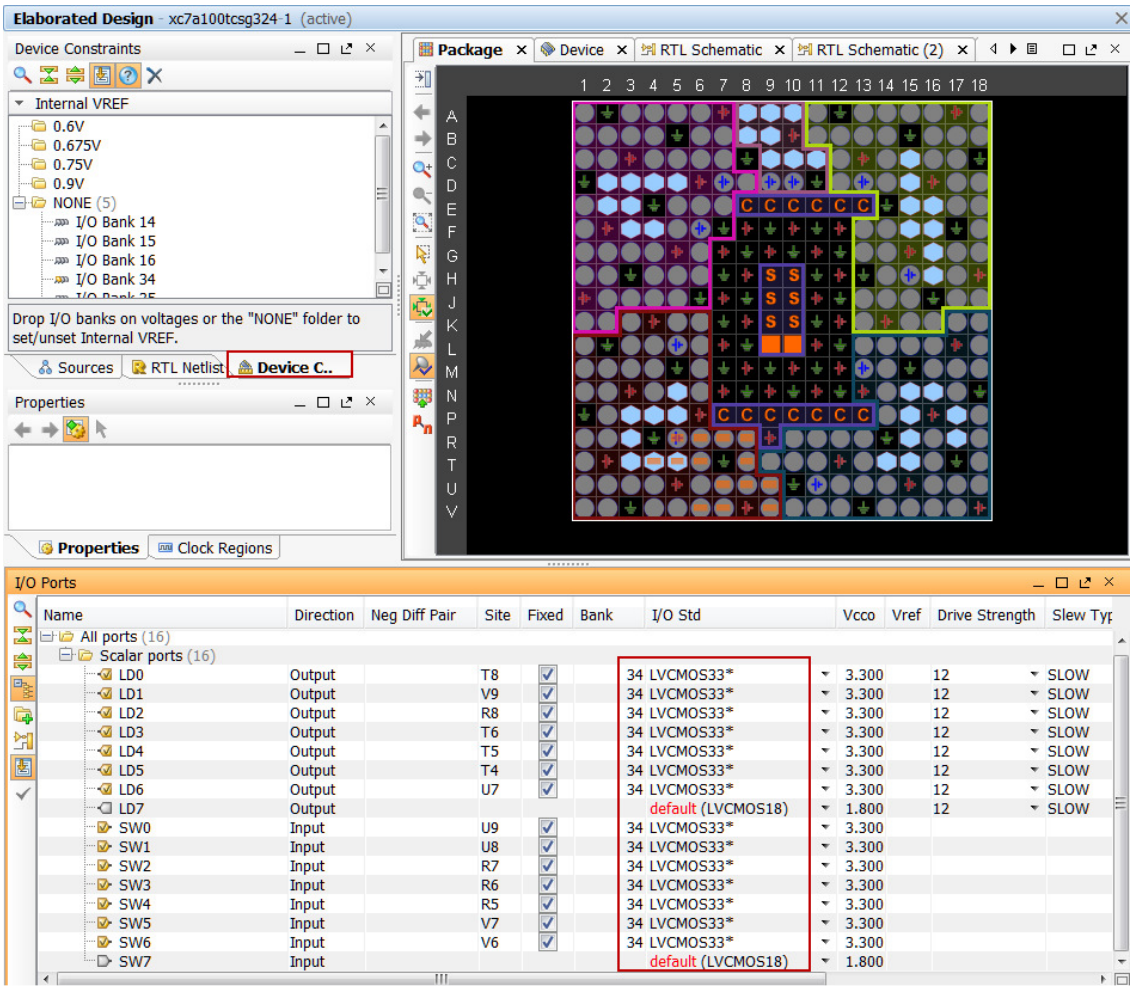

Figure 22. I/O Planning layout view of Nexys4

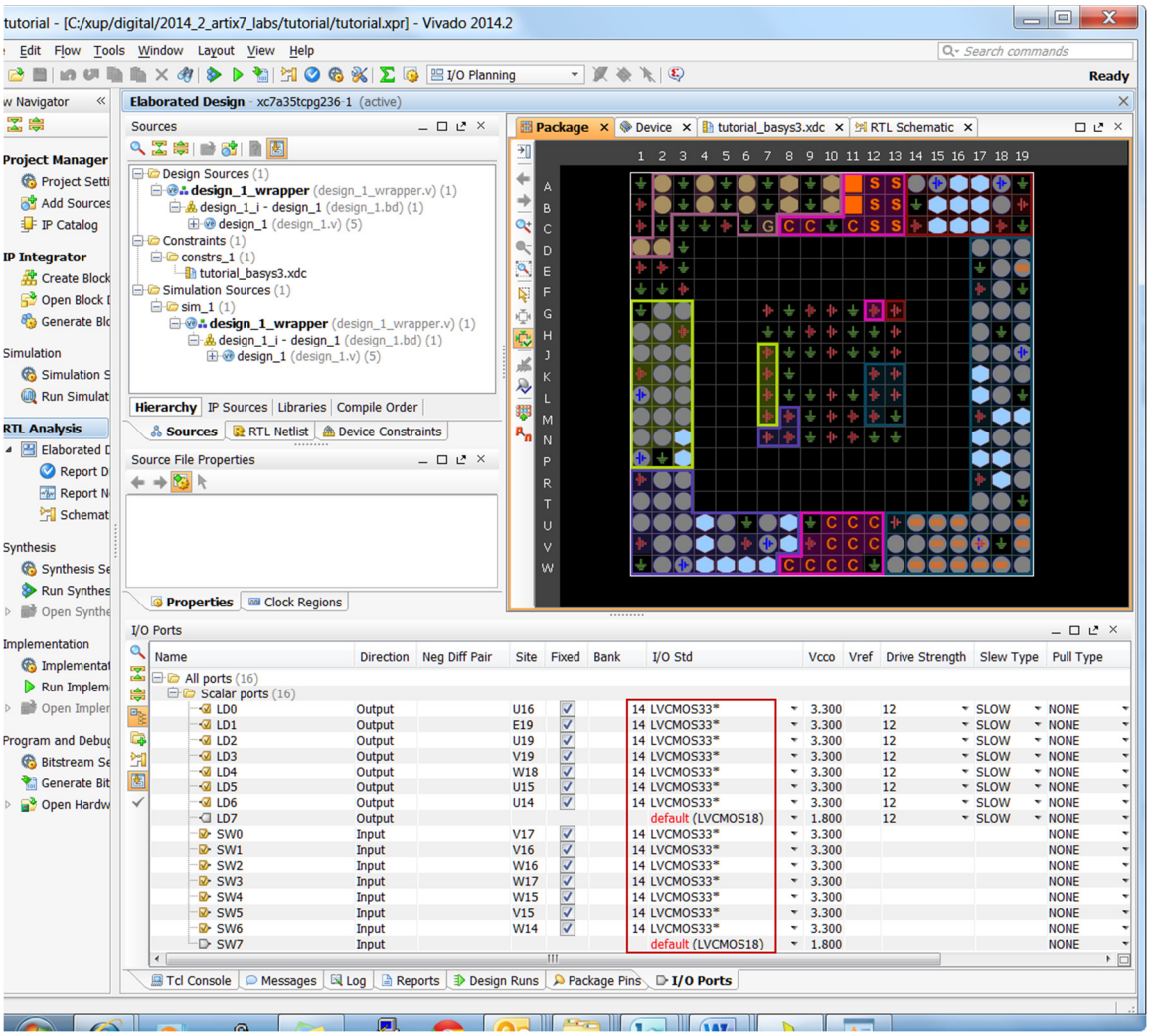

Figure 22. I/O Planning layout view of Basys3

3-4-2. Click under the *I/O Std* column across the **LD7** row and select *LVCOMS33*. This assigns the LVCMOS33 standard to the site.

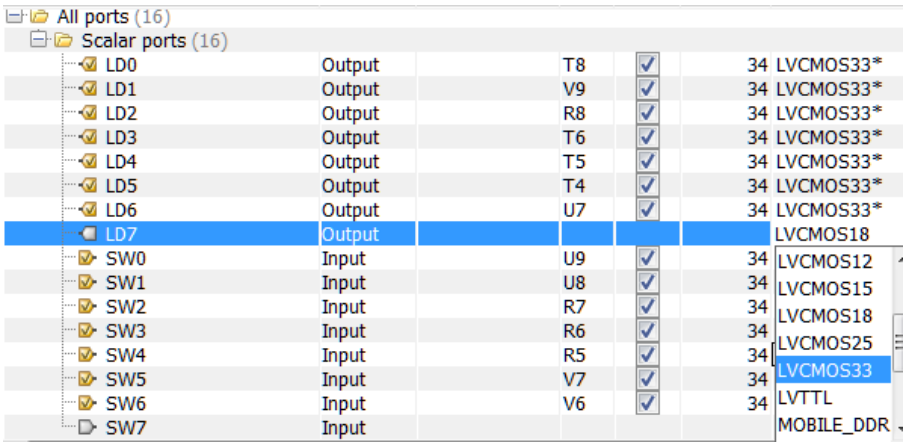

Figure 23. Assigning I/O standard to Nexys4

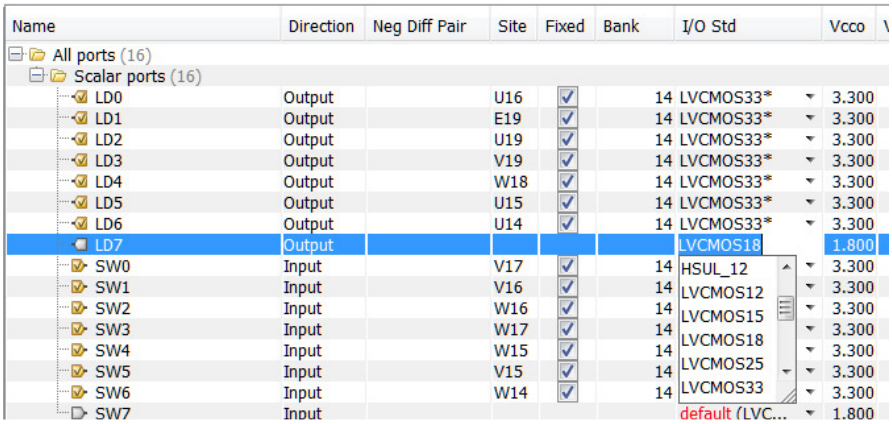

Figure 23. Assigning I/O standard to Basys3

3-4-3. Similarly, click under the *Site* column acrossLD7 row to see a drop-down box appear. Type **U** (for Nexys4) or **V** (for Basys3) in the field to jump to Uxx or Vxx pins, scroll-down until you see U6 (Nexys4) or V14 (Basys3) , select U6 (Nexys4) or V14 (Bsys3) and hit the *Enter* key to assign the pin.

3-4-4. You can also assign the pin constraints using tcl commands. Type in the following two commands in the Tcl Console tab to assign the *V5* (Nexys4) or *W13* (Basys3) pin location and the *LVCSMOS33* I/O standard to **SW7** hitting the Enter key after each command.

Nexys4:

```
set_property package_pin V5 [get_ports SW7]
set_property iostandard LVCMOS33 [get_ports [list SW7]]
```

Basys3:

```
set_property package_pin W13 [get_ports SW7]
set_property iostandard LVCMOS33 [get_ports [list SW7]]
```

Observe the pin and I/O standard assignments in the I/O Ports tab.You can also assign the pin by selecting its entry (SW7) in the I/O ports tab, and dragging it to the Package view, and placing it at the V5 (Nexys4) or W13 (Basys3) location. You can assign the LVCMOS33 standard by selecting its entry (SW7), selecting Configure tab of the I/O Port Properties window, followed by clicking the drop-down button of the I/O standard field, and selecting LVCMOS33.

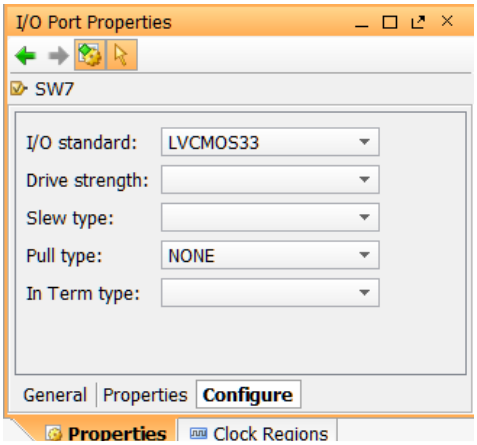

Figure 24. Assigning I/O standard through the I/O Port Properties form

3-4-5. Select **File > Save Constraints** and click **OK** to save the constraints in the **tutorial_nexys4.xdc** or **tutorial_basys3.xdc** file.

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

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

Simulate the Design using the XSim Simulator Step 4

4-1. Add the tutorial_tb.v testbench file.

4-1-1. Click **Add Sources** under the *Project Manager* tasks of the *Flow Navigator* pane.

4-1-2. Select the *Add or Create Simulation Sources* option and click **Next**.

4-1-3. In the *Add Sources Files* form, click the **Add Files…** button.

4-1-4. Browse to the **<2014_2_artix7_labs>\tutorial** folder and select *tutorial_tb.v* and click **OK**.

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

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

The tutorial_tb.v file is added under the *Simulation Sources* group, and **system_wrapper_1.v** is automatically placed in its hierarchy as a tut1 instance.

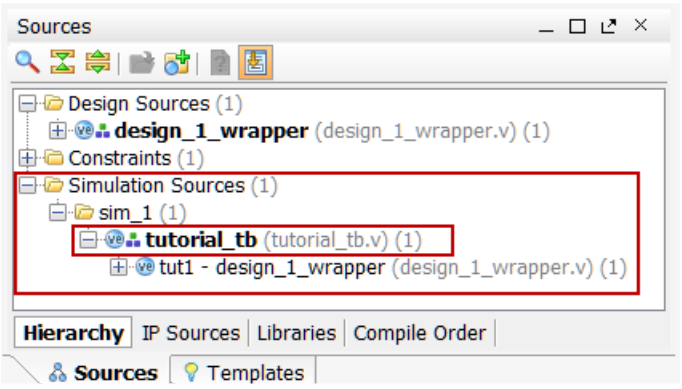

Figure 25. Simulation Sources hierarchy

4-1-7. 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 tutorial.srcs directory, and that a copy of tutorial_tb.v is placed under **tutorial.srcs > sim_1 > imports > tutorial**.

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

C:/xup/digital/tutorial/tutorial.srcs/sim_1/imports/tutorial/tutorial_tb.v

```
1 `timescale 1ns / 1ps
2 //////////////////////////////////////////////////////////////////
3 // Module Name: tutorial_tb
4 //////////////////////////////////////////////////////////////////
5 module tutorial_tb(
6     );
7     reg [7:0] switches;
8     wire [7:0] leds;
9     reg [7:0] e_led;
10    integer i;
11    design_1_wrapper tut1(
12            .LD0(leds[0]),
13            .LD1(leds[1]),
14            .LD2(leds[2]),
15            .LD3(leds[3]),
16            .LD4(leds[4]),
17            .LD5(leds[5]),
18            .LD6(leds[6]),
19            .LD7(leds[7]),
20            .SW0(switches[0]),
21            .SW1(switches[1]),
22            .SW2(switches[2]),
23            .SW3(switches[3]),
24            .SW4(switches[4]),
25            .SW5(switches[5]),
26            .SW6(switches[6]),
27            .SW7(switches[7]));
28
29    function [7:0] expected_led;
30       input [7:0] swt;
31    begin
32       expected_led[0] = ~swt[0];
33       expected_led[1] = swt[1] & ~swt[2];
34       expected_led[3] = swt[2] & swt[3];
35       expected_led[2] = expected_led[1] | expected_led[3];
36       expected_led[7:4] = swt[7:4];
37    end
38    endfunction
39
40    initial
41    begin
42        for (i=0; i < 255; i=i+2)
43        begin
44            #50 switches=i;
45            #10 e_led = expected_led(switches);
46            if(leds == e_led)
47                $display("LED output matched at", $time);
48            else
49                $display("LED output mis-matched at ",$time,": expected: %b, actual: %b", e_led, leds);
50        end
51    end
52
53 endmodule
```

Figure 26. The self-checking testbench

The testbench defines the simulation step size and the resolution in line 1. The testbench module definition begins on line 5. Line 11 instantiates the DUT (device/module under test). Lines 29 through 38 define the same module functionality for the expected value computation. Lines 40 through 51 define the stimuli generation and compares the expected output with what the DUT provides. Line 53 ends the testbench. The $display task will print the message in the simulator console window when the simulation is run.

4-2. Simulate the design for 200 ns using the XSim simulator.

4-2-1. Select **Simulation Settings** under the *Project Manager* tasks of the *Flow Navigator* pane.

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

4-2-2. Select the **Simulation** tab, and set the **Simulation Run Time** value to 200 ns and click **OK**.

4-2-3. Click on **Run Simulation > Run Behavioral Simulation** under the *Project Manager* tasks of the *Flow Navigator* pane.

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

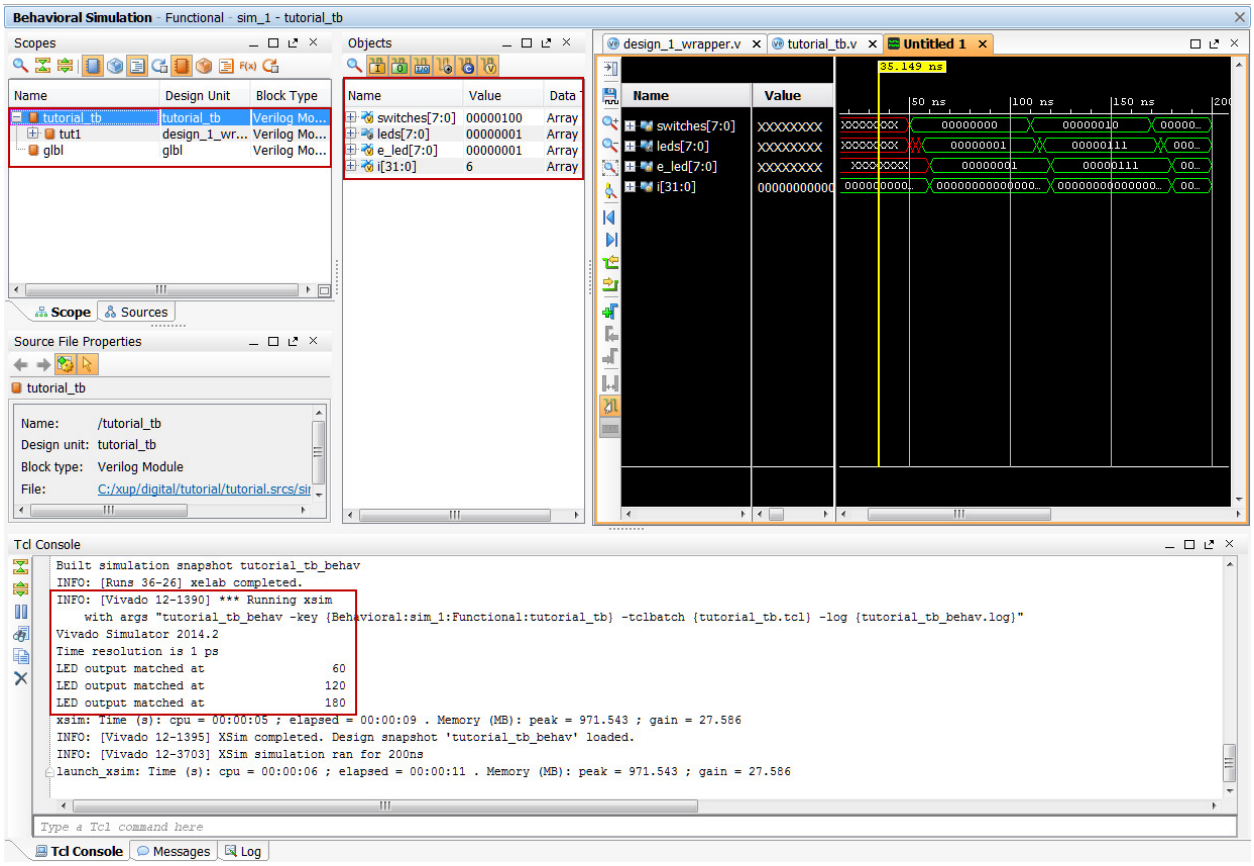

Figure 27. Simulator output

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

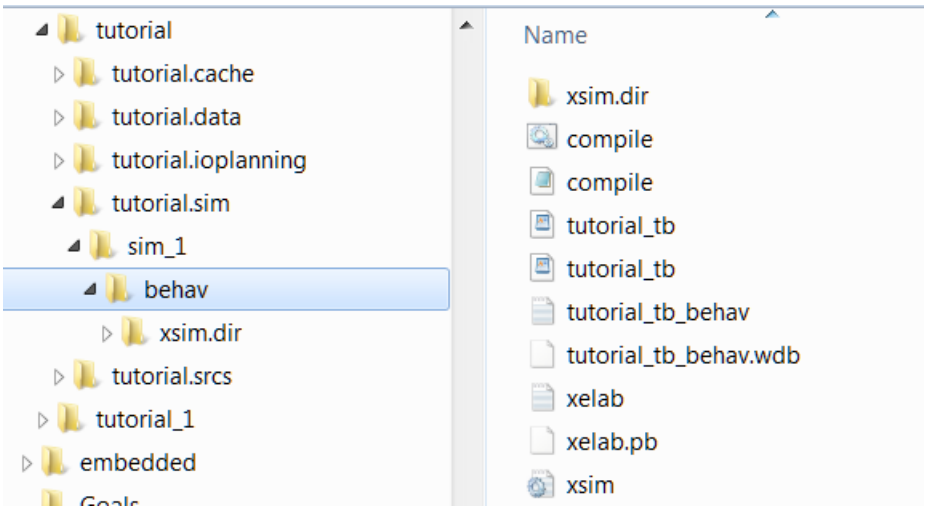

Figure 28. Directory structure after running behavioral simulation

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

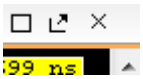

Figure 29. Float Button

Figure 30. Dock Window Button

4-3. Change display format if desired.

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

4-4. Add more signals to monitor lower-level signals and continue to run the simulation for 500 ns.

4-4-1. Expand the **tutorial_tb** instance, if necessary, in the *Scopes* window and select the **tut1** instance.

The SW* (7 to 0)] and LD* (7 to 0) signals will be displayed in the *Objects* window.

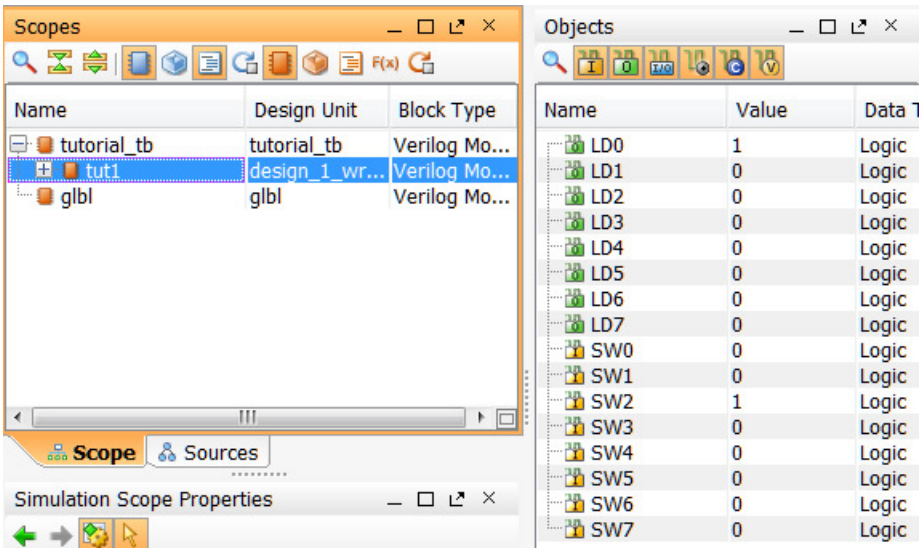

Figure 31. Selecting lower-level signals

4-4-2. Select **SW*** and **LD*** and drag them into the waveform window to monitor those lower-level signals.

4-4-3. On the simulator tool buttons ribbon bar, type 500 in the time window, click on the drop-down button of the units field and select ns, and click on the ( ) button.

The simulation will run for an additional 500 ns.

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

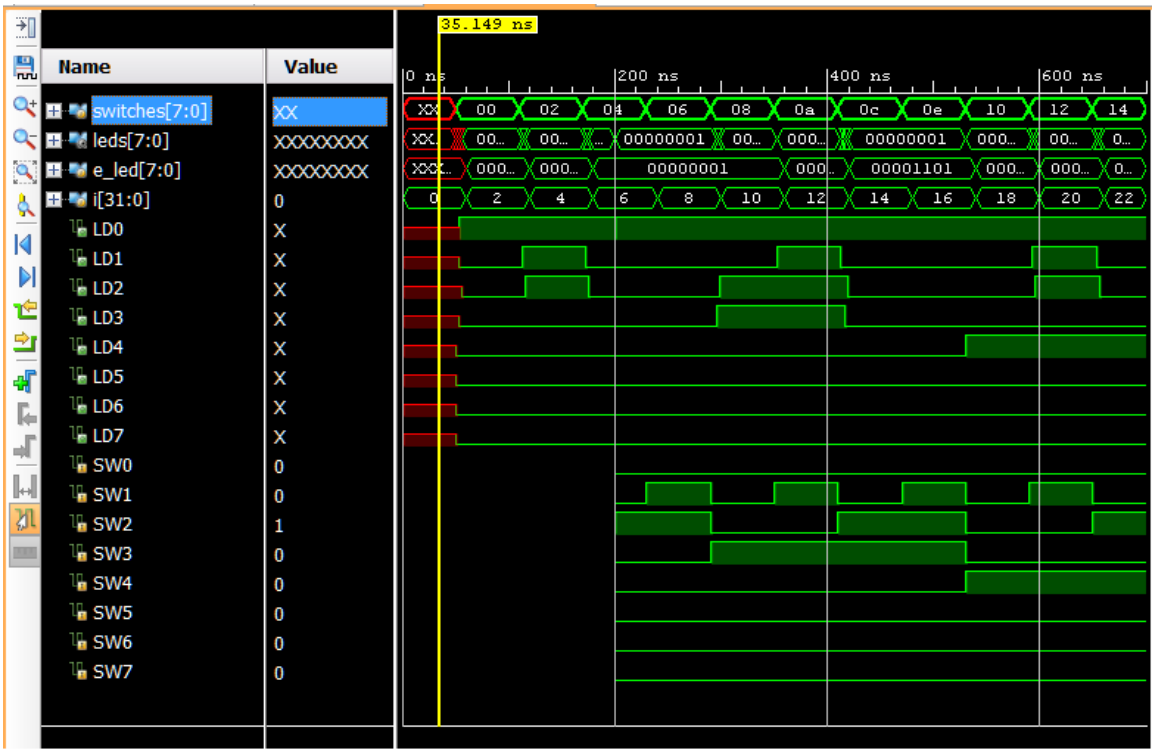

Figure 32. Running simulation for additional 500 ns

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

4-4-6. Click **OK** and then click **No** to close it without saving the waveform.

Synthesize the Design Step 5

5-1. Synthesize the design with the Vivado synthesis tool and analyze the Project Summary output.

5-1-1. Click on **Run Synthesis** under the *Synthesis* tasks of the *Flow Navigator* pane.

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

5-1-2. Select the *Open Synthesized Design* option and click **OK** as we want to look at the synthesis output before progressing to the implementation stage.

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

5-1-3. Select the **Project Summary** tab (Select default layout if the tab is not visible) and understand the various windows.

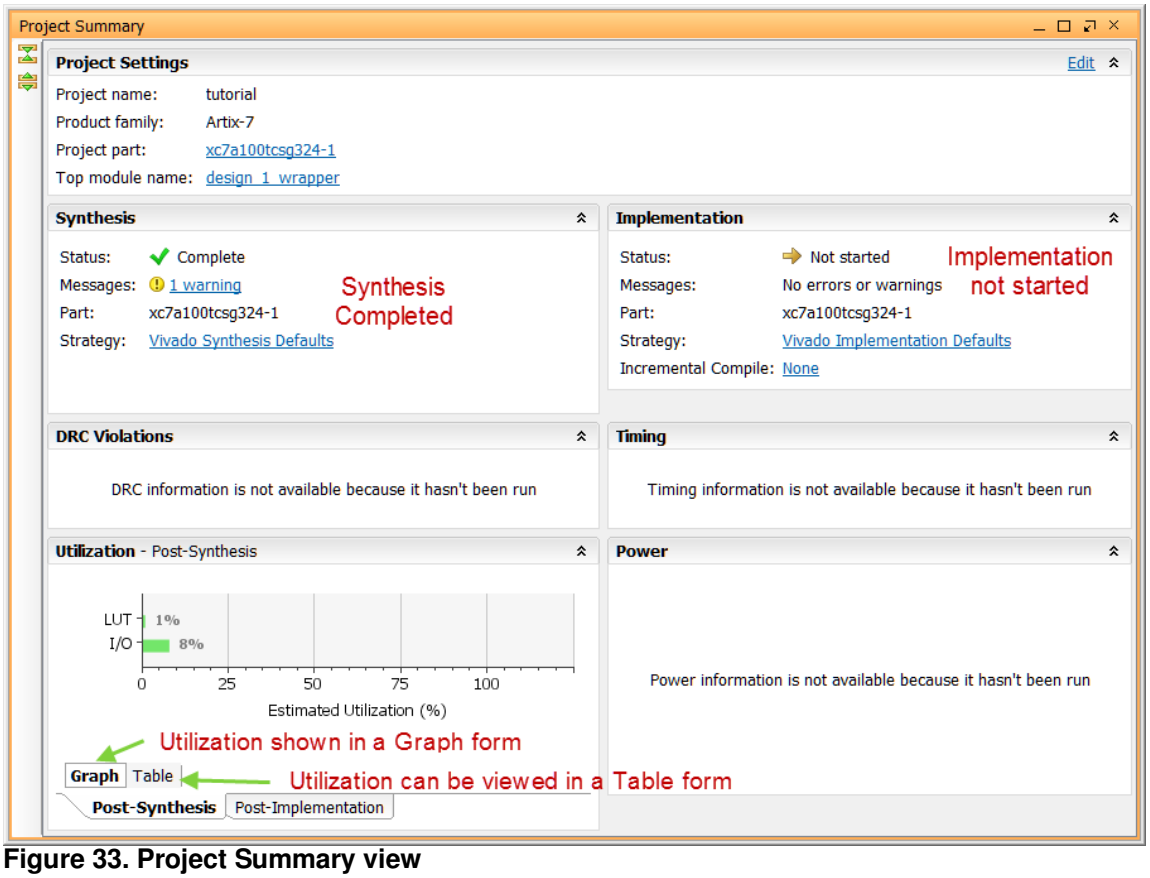

Figure 33. Project Summary view

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

5-1-4. Click on the **Table** tab in the **Project Summary** tab.

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

Resource	Estimation	Available	Utilization...
LUT	5	63400	0.01
I/O	16	210	7.62

Graph **Table**

Figure 34. Resource utilization estimation summary for Nexys4

Utilization - Post-Synthesis

Resource	Estimation	Available	Utilization...
LUT	5	20800	0.02
I/O	16	106	15.09

Graph **Table**

Figure 34. Resource utilization estimation summary for Basys3

5-1-5. Click on **Schematic** under the *Open Synthesized Design* tasks of *Synthesis* tasks of the *Flow Navigator* pane to view the synthesized design in a schematic view.

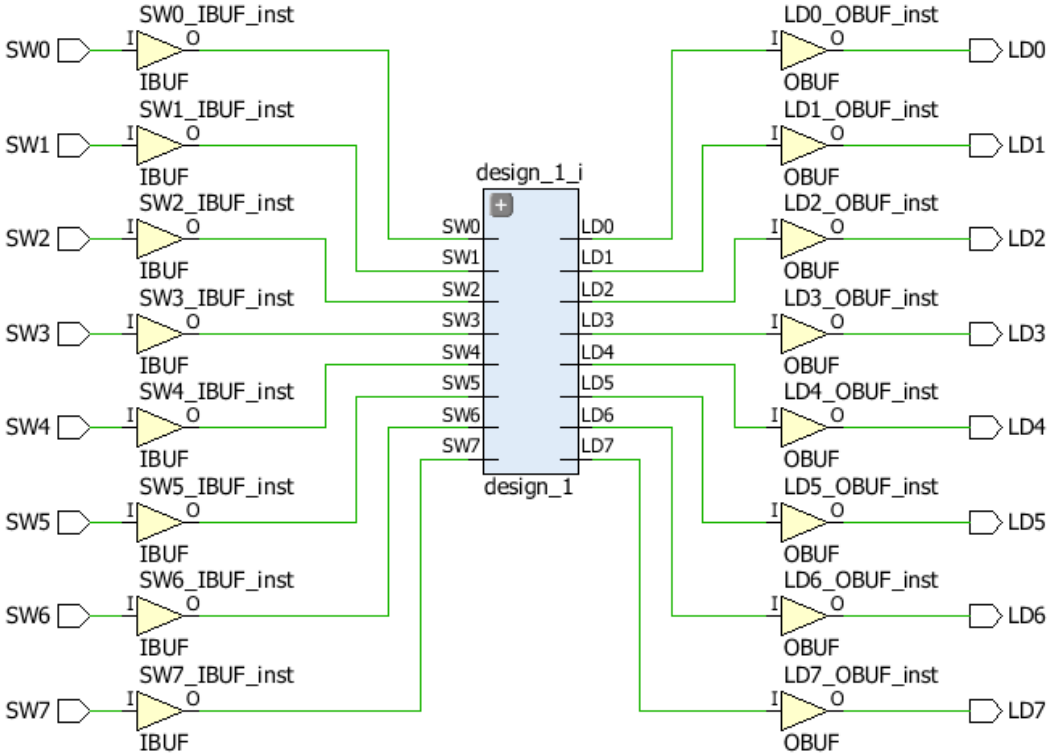

Figure 35. Synthesized design's schematic view

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

5-1-6. Click on the **+** sign within the *design_1* block to see the underlying logic.

5-1-7. Click on the **+** sign of each of the lower-level blocks to see their implementation.

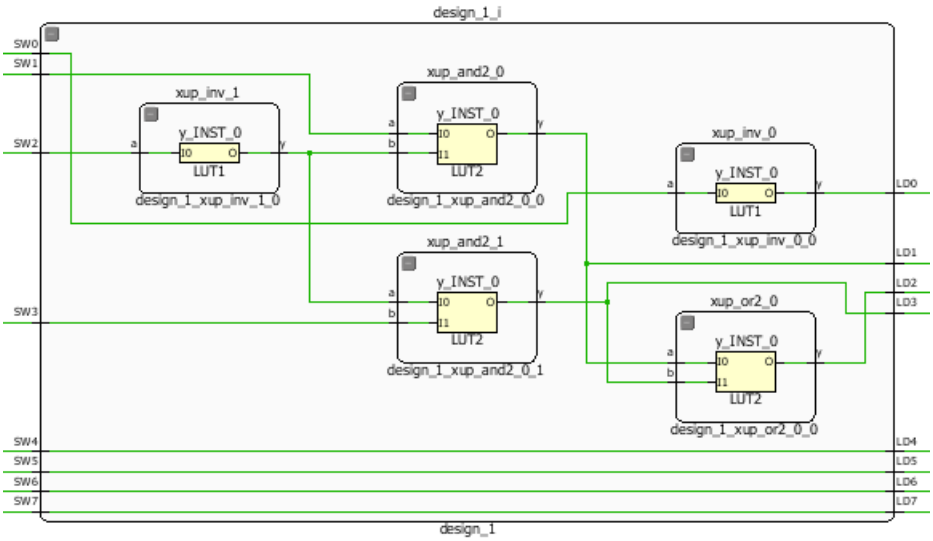

Figure 36. Lower-level logic

The logical gates are implemented in LUTs (1 input is listed as LUT1 and 2 input is listed as LUT2). Five blocks in RTL analysis output are mapped into five LUTs in the synthesized output.

Using the Windows Explorer, verify that **tutorial.runs** directory is created under **tutorial**. Under the **runs** directory, **synth_1** directory is created which holds several temporary sub-directories.

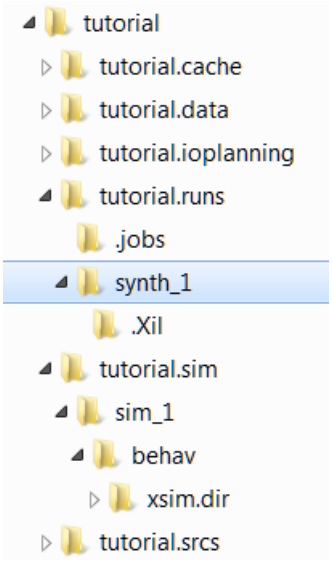

Figure 37. Directory structure after synthesizing the design

Implement the Design Step 6

6-1. Implement the design with the Vivado Implementation Defaults (Vivado Implementation 2014) settings and analyze the Project Summary output.

6-1-1. Click on **Run Implementation** under the *Implementation* tasks of the *Flow Navigator* pane.

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

6-1-2. Select **Open implemented design** and click **OK** as we want to look at the implemented design in a Device view tab.

6-1-3. Click **Yes** to close the synthesized design.

The implemented design will be opened.

6-1-4. In the *Netlist* pane, select one of the nets (e.g. n_0_design_1_i) and notice that the displayed net.

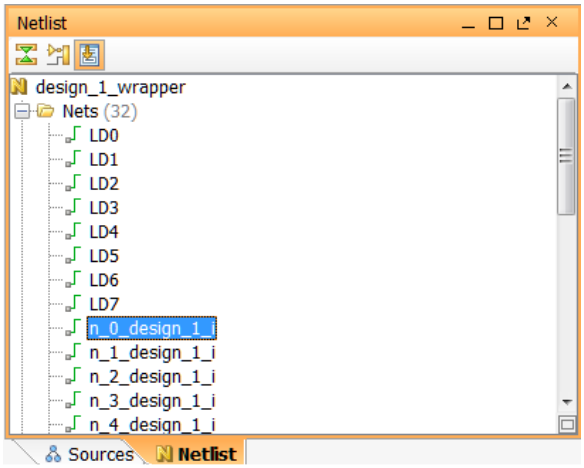

Figure 38: Selecting a net

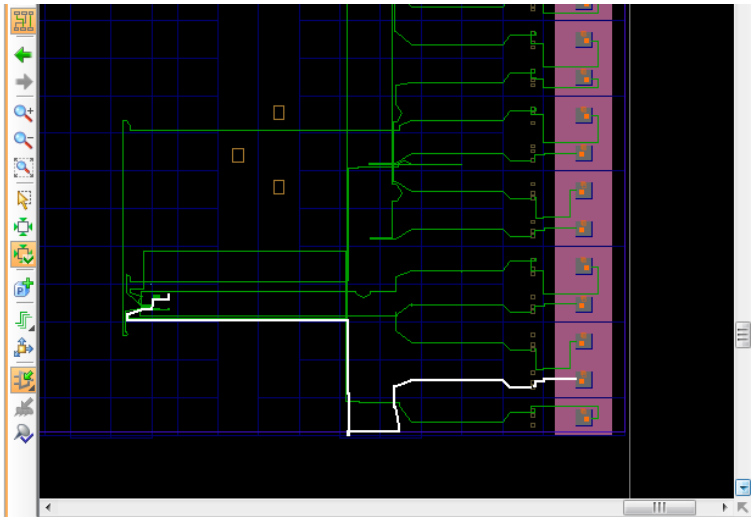

Figure 39. Viewing implemented design for Nexys4

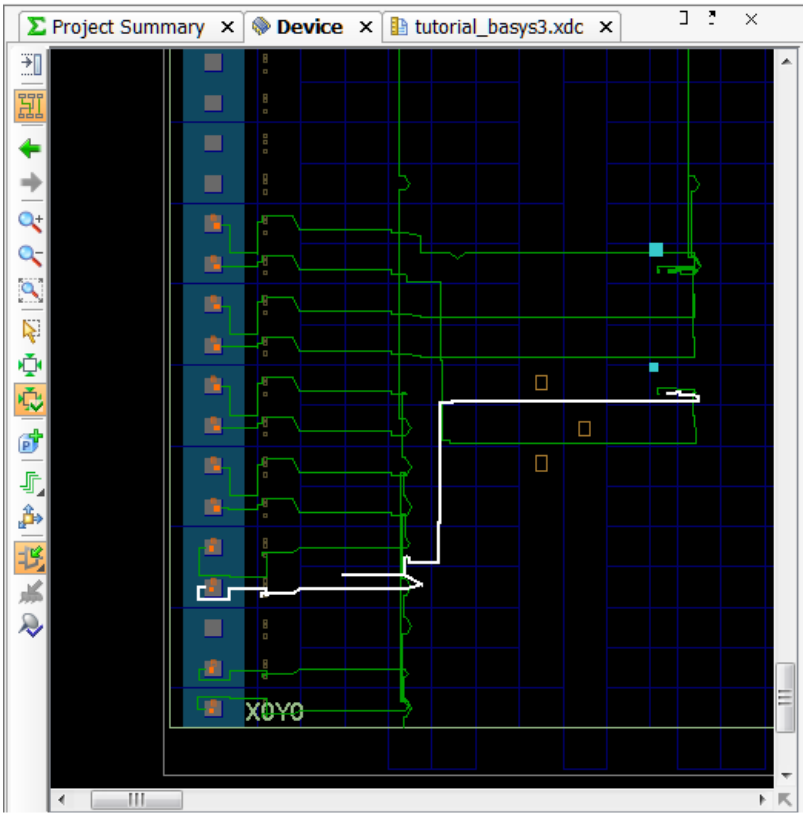

Figure 39. Viewing implemented design for Basys3

6-1-5. 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.

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). Select the **Post-implementation** tabs under the *Timing* and *Utilization* windows.

6-1-6. Using the Windows Explorer, verify that **impl_1** directory is created at the same level as **synth_1** under the **tutorial.runs** directory. The **impl_1** directory contains several files including the report files.

6-1-7. Select the **Reports** tab, and double-click on the *Utilization Report* entry under the *Place Design* section. The report will be displayed in the auxiliary view pane showing resources utilization. Note that since the design is combinatorial no registers are used.

Perform Timing Simulation Step 7

7-1. Run a timing simulation.

7-1-1. Select **Run Simulation > Run Post-Implementation Timing Simulation** process under the *Simulation* tasks of the *Flow Navigator* pane.

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

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

7-1-2. Click on the **Zoom Fit** button to see the waveform window from 0 to 200 ns.

7-1-3. Right-click at 50 ns (where the switch input is set to 0000000b) and select **Markers > Add Marker**.

7-1-4. Similarly, right-click and add a marker at around 55.000 ns where the **leds** changes.

7-1-5. 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 40. 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.

7-1-6. Close the simulator by selecting **File > Close Simulation** without saving any changes.

Generate the Bitstream and Verify Functionality Step 8

8-1. Connect the board and power it ON. Generate the bitstream, open a hardware session, and program the FPGA.

8-1-1. 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.

This process will have **design_1_wrapper.bit** file generated under **impl_1** directory which was generated under the **tutorial.runs** directory.

8-1-2. Make sure that the power supply source is jumper to *USB* and the provided Micro-USB cable is connected between the board and the PC.

Note that you do not need to connect the power jack and the board can be powered and configured via USB alone

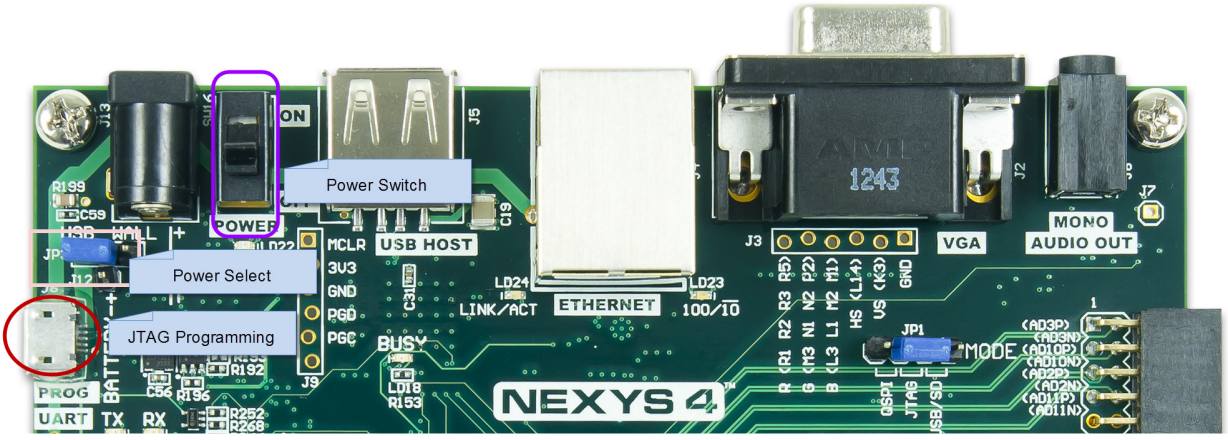

Figure 41. Board settings for Nexys4

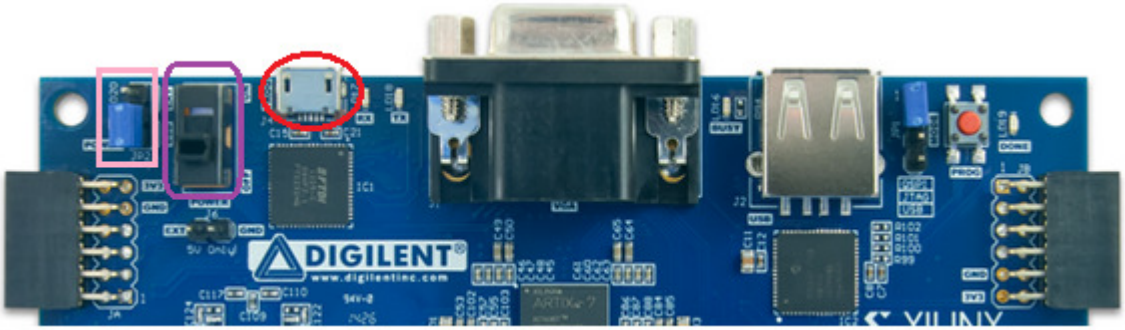

Figure 41. Board settings for Basys3

8-1-3. Power **ON** the switch on the board.

8-1-4. Select the *Open Hardware Manager* option and click **OK**.

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

8-1-5. Click on the **Open a new hardware target** link.

You can also click on the Open Recent Hardware Target link if the board was already targeted before.

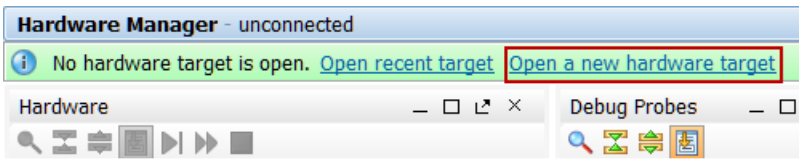

Figure 42. Opening new hardware target

8-1-6. Click **Next** to see the Vivado CSE Server Name form.

8-1-7. Click **Next** with the localhost port selected.

The JTAG cable will be searched and the Xilinx_tcf should be detected and identified as e hardware target. It will also show the hardware devices detected in the chain.

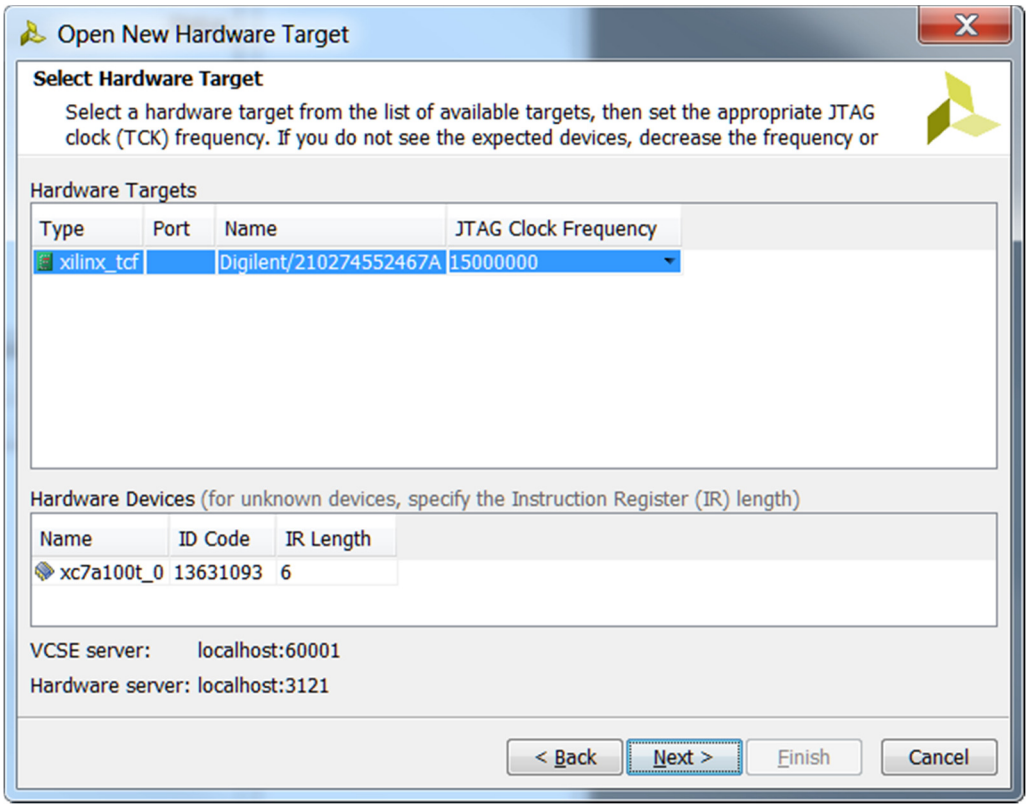

Figure 43. New hardware target detection for Nexys4

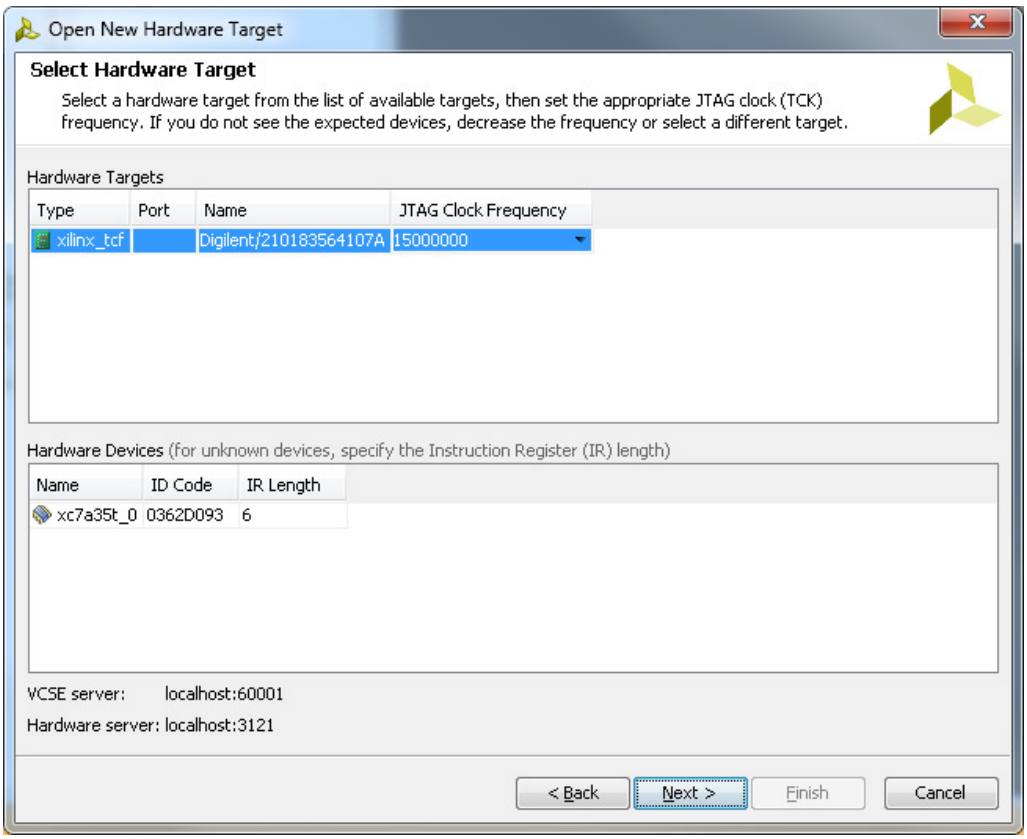

Figure 44. New hardware target detection for Basys3

8-1-8. Click **Next** and **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.

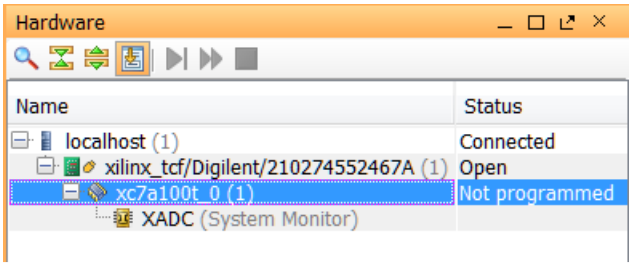

Figure 45. Opened hardware session for Nexys4

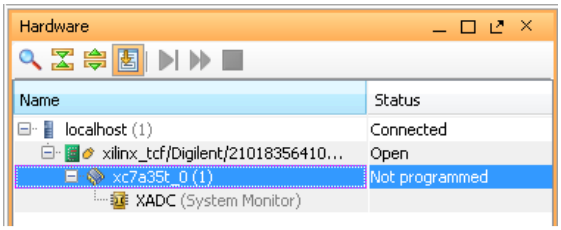

Figure 45. Opened hardware session for Basys3

8-1-9. Select the device and verify that the **design_1_wrapper.bit** is selected as the programming file in the General tab.

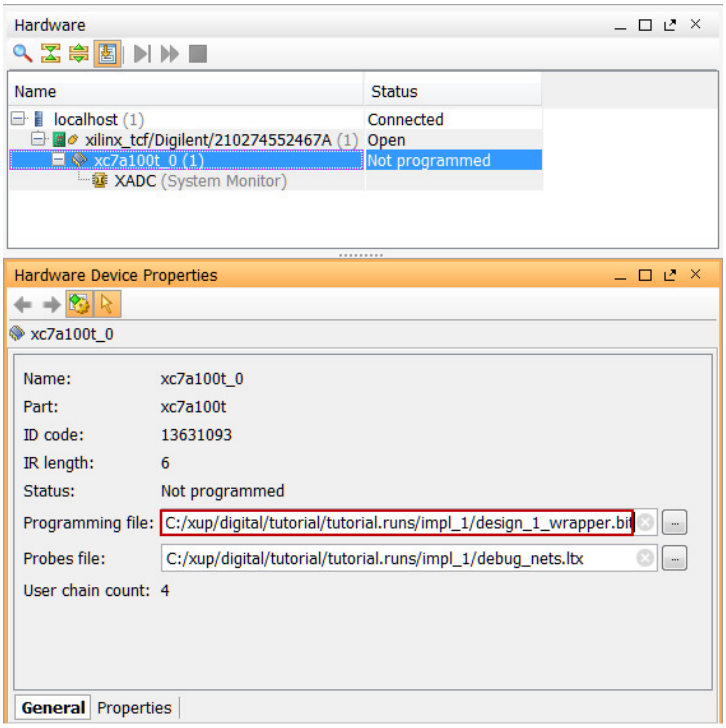

Figure 46. Programming file for Nexys4

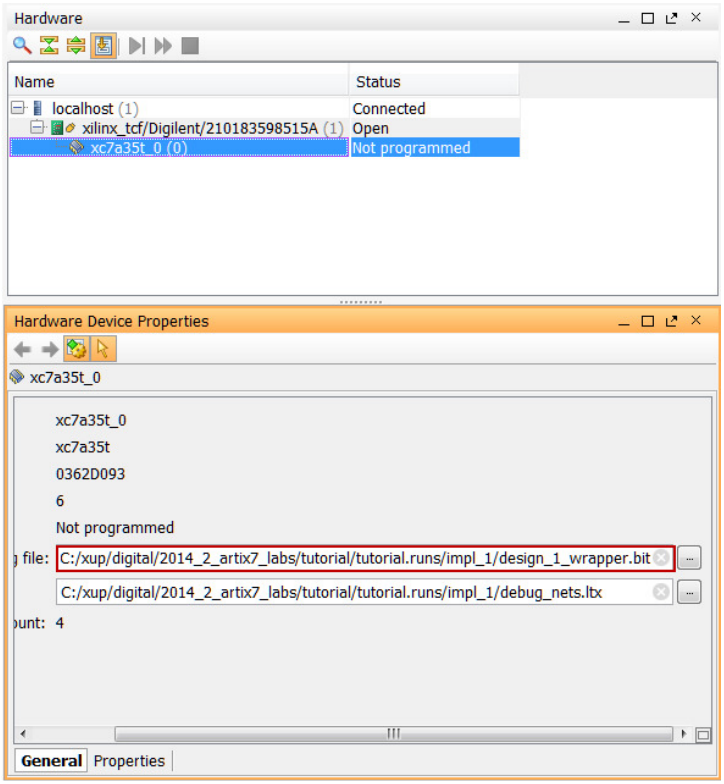

Figure 46. Programming file for Basys3

8-1-10. **Right-click** on the device and select *Program Device…* to program the target FPGA device.

8-1-11. Click **Program** to program the FPGA with the selected bitstream.

The DONE light will lit when the device is programmed. You may see some LEDs lit depending on the switches position.

8-1-12. Verify the functionality by flipping switches and observing the output on the LEDs.

8-1-13. Close the hardware session by selecting **File > Close Hardware Manager.**

8-1-14. Click **OK** to close the session.

8-1-15. Power **OFF** the board.

8-1-16. 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 XUP IP library (IPI blocks and user constraint file). A behavioral simulation was done to verify the model functionality. The model was then synthesized, implemented, and a bitstream was generated. The timing simulation was run on the implemented design using the same testbench. The functionality was verified in hardware using the generated bitstream.