

Lab 1 – Introduction to Development Environment

CS1050 - Computer Organization and Digital Design

Dept. of Computer Science and Engineering, University of Moratuwa

Learning Outcomes

In this lab, we will learn how to develop logic circuits using an FPGA (Field-Programmable Gate Array). After completing the lab, you will be able to:

- design a simple logic circuit using VHDL
- connect the inputs and outputs of your circuit to switches and LEDs
- generate the bitstream
- configure the FPGA using the generated bitstream
- verify the functionality on the development board

Introduction

The laboratory assignments will use the Digilent BASYS 3 development board shown in Fig. 1. BASYS 3 is a circuit design and implementation platform that can be used to gain experience in building real digital circuits. Built around a Xilinx Artix-7 FPGA, the BASYS 3 provides complete, ready-to-use hardware suitable for hosting circuits ranging from basic logic devices to complex controllers. The board comes with a large collection of on-board Input/output (I/O) devices such as switches, push buttons, LEDs, 7-segments, VGA port, etc.

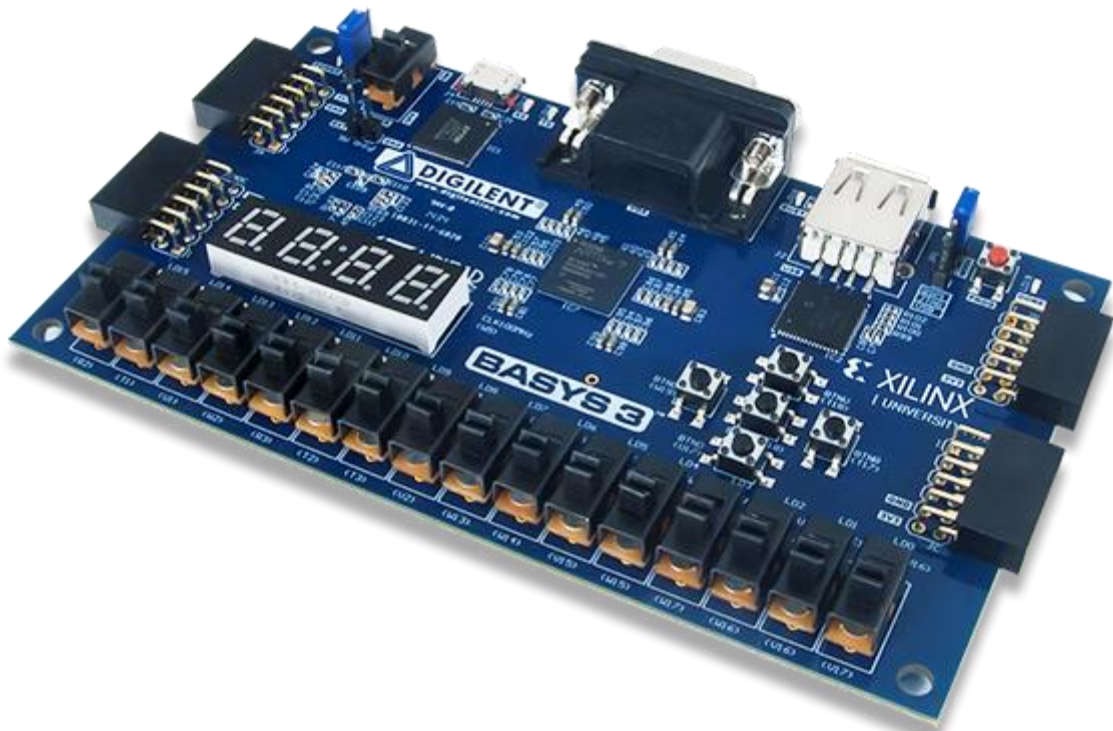


Figure 1 – Digilent BASYS 3 Artix-7 FPGA development board.

An FPGA (Field-Programmable Gate Array) is an integrated circuit designed to be configured by a designer after manufacturing; hence, referred to as “field-programmable”. BASYS 3 uses an Artix-7 35T FPGA from Xilinx. The FPGA configuration is generally

specified using schematics or Hardware Description Language (HDL). In our labs, we will only use the HDL to configure/program the FPGA, as the Vivado development suite supported by BASYS 3 works only with VHDL and Verilog HDLs. While HDLs take a bit more time to learn, much more complex circuits can be built in less time.

We use Xilinx Vivado Design Suite software to program the FPGA using VHDL. The laboratory assignments are based on version 2018.2 of the Xilinx Vivado WebPack edition.

Note As you are new to digital design, do not be surprised if it appears intimidating at first. Do not despair. We will selectively introduce the features as needed while ignoring others that we do not need. As the semester progresses, you will understand more and more features, and by the end of the semester, you will be able to design complex circuits using many features of the Vivado Design Suite.

Tips In addition to using the computers in the lab, you can also install the Vivado WebPack edition on your computer. Size of installation files vary from 6 GB to 20 GB depending on the collection of tools you wish to download, and typical installation requires ~25 GB disk space. Hence, obtain the installation files from the instructor rather than downloading them from the web.

Design Flow

Typical steps involved in programming an FPGA are depicted in Fig. 2. Following is a brief description of steps involved:

- **Create Design** – We first create an Vivado Design Suite project and then, create or add source files to that project. Projects can contain many types of source files and design modules, including HDL, embedded processor, and digital signal processing modules. We will use only the HDL for our laboratory assignments.
- **Simulate Design** – At various points during the design flow, we can verify the functionality of the design using a simulation tool. We use XSim, which is delivered with the Vivado. This step will be covered in a later lab.
- **Synthesize Design** – During synthesis, the synthesis engine compiles the design to transform HDL sources into an architecture-specific design netlist (i.e., connectivity of an electronic design). The Vivado supports the use of Xilinx Design Constraints (XDC), which is delivered with the Vivado.
- **Implement Design** – After synthesis, we implement the design by mapping the logical design to resources on the FPGA device, while meeting logical, physical, and timing constraints.
- **Program Device** – Then the implemented design needs to be transferred to the development board to program the FPGA. For this a *bitstream* that could be downloaded to the selected FPGA (a.k.a., device) is generated as a file. This bitstream tells how to configure the given FPGA to build the desired circuit. Once

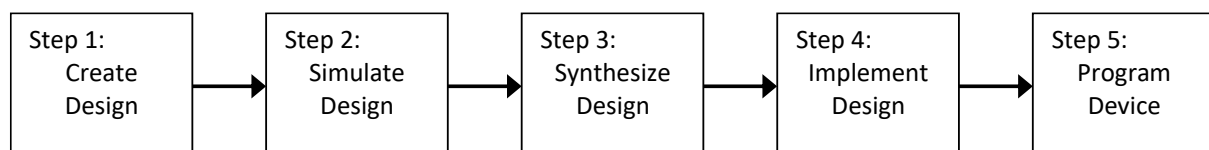


Figure 2 – Design flow overview.

generated, the device is programmed by transferring the programming file (i.e., bitstream) to the device.

My First Circuit

Step 1: Starting Vivado Design Suite

Start **Vivado 2018.2** by locating the icon on the Windows **Desktop**, **Start** menu, or **Search** box.

You should see a display similar to the one in Fig. 3. This display consists of several options that provide access to various features of the Vivado software.

Most of the commands provided by the Vivado can be accessed using a set of menus that are located below the title bar.

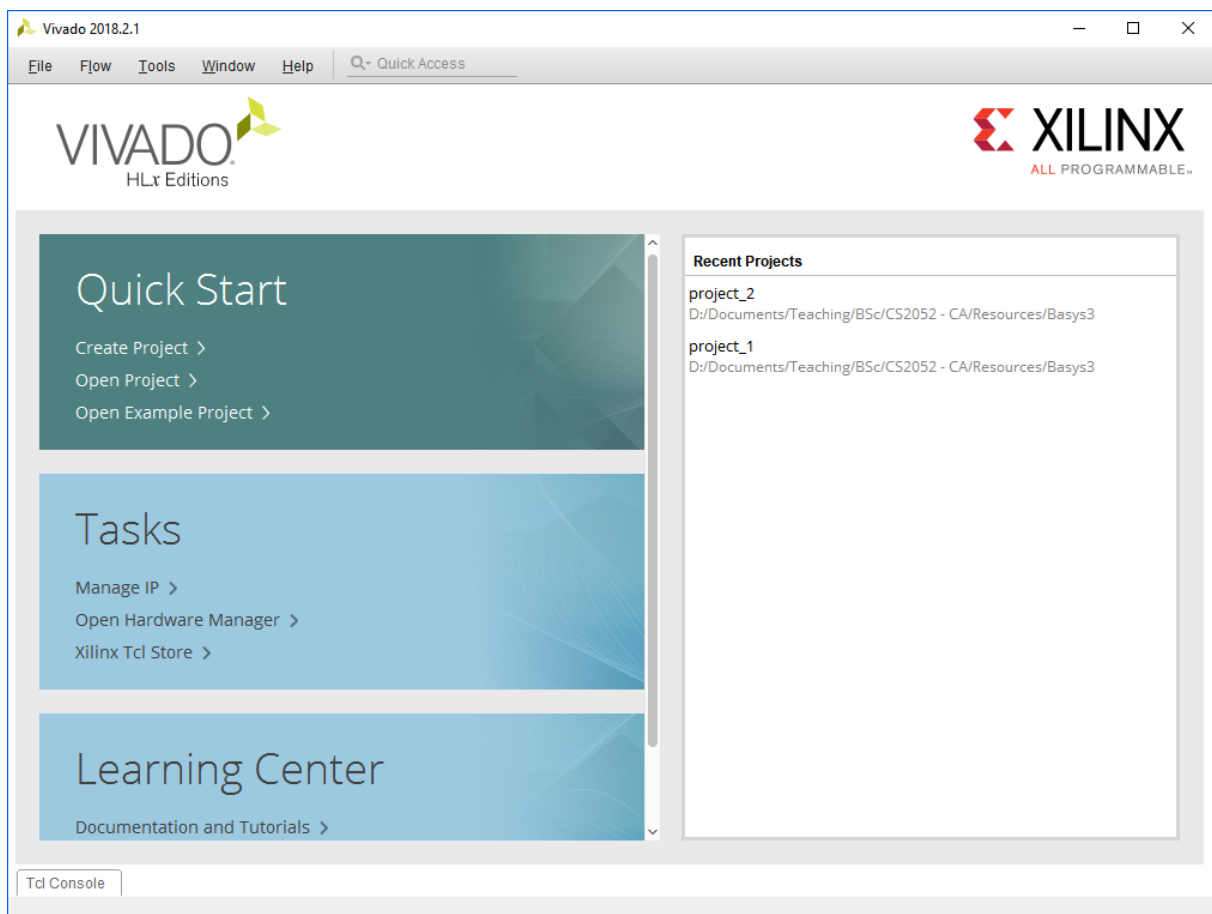


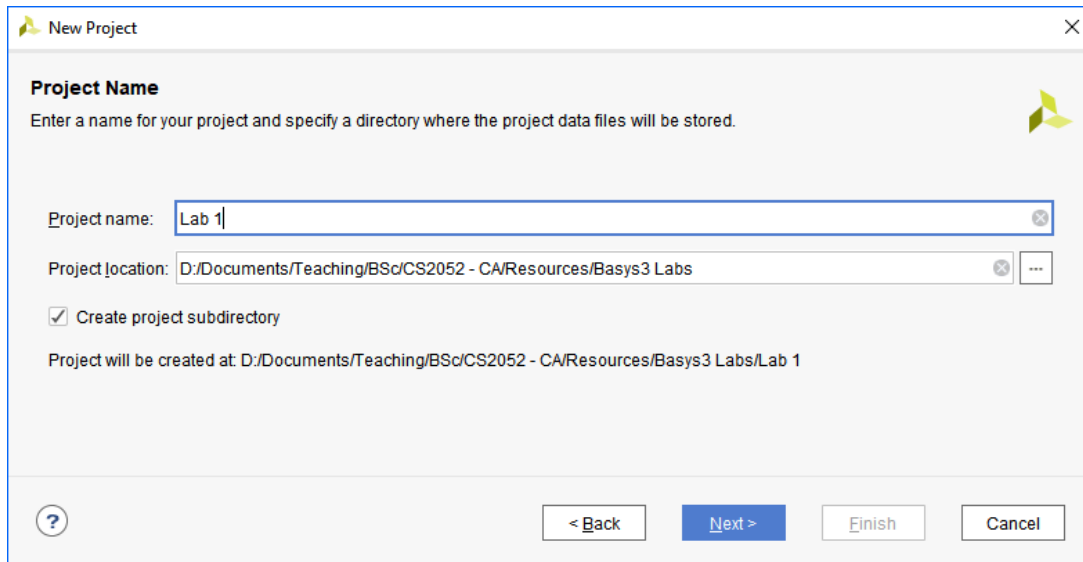
Figure 3 – Vivado Project Navigator window.

Step 2: Starting a New Project

To create a new project either click on the **Create Project >** link on the **Start** panel or **File → Project → New** from the menu.

Then you will see the Create a New Vivado Project wizard dialog box. Click on the **Next** button.

In the next screen (Fig. 4) set the **Project name** as **Lab 1**. Use **Project location** textbox (or ... button) to set a suitable location to store your project files. Make sure to store your file on a location that can be easily located.



New Project

Project Name
Enter a name for your project and specify a directory where the project data files will be stored.

Project name:

Project location:

☒ Create project subdirectory

Project will be created at: D:/Documents/Teaching/BSc/CS2052 - CA/Resources/Basys3 Labs/Lab 1

? < Back Next > Finish Cancel

Figure 4 – Project name dialog box.

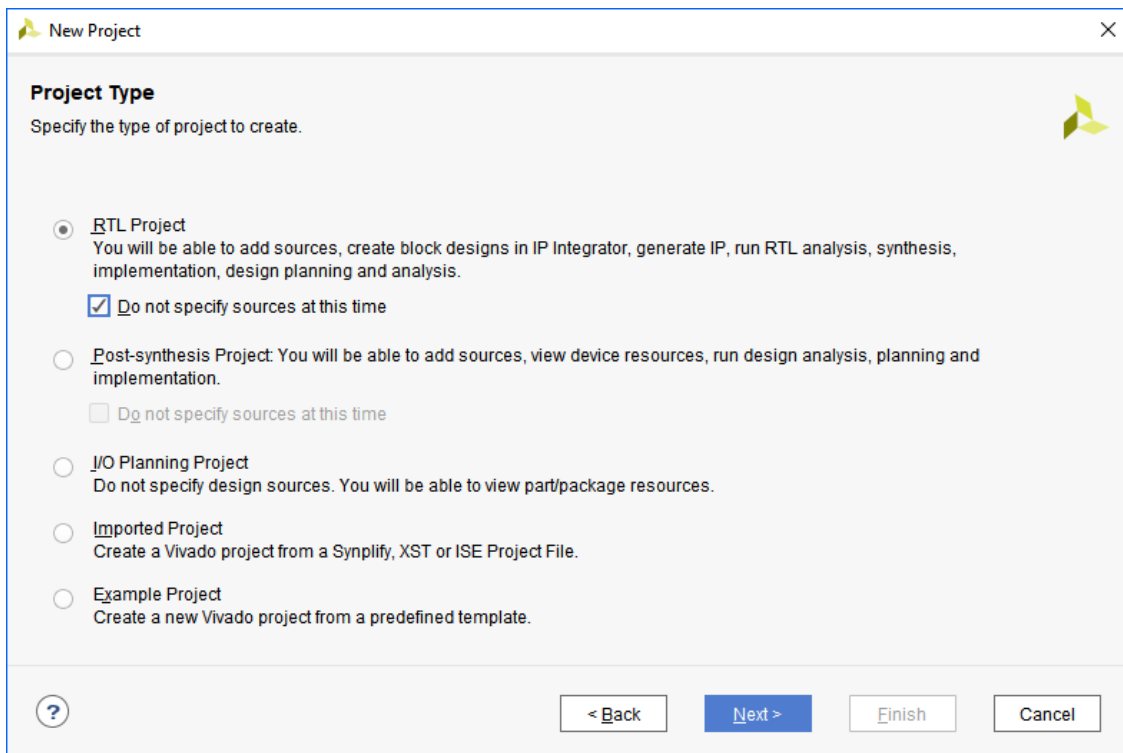
Then click on the **Next** button to continue.

Step 3: Project Type

Next, we need to select the type of project we wish to create. Choose **RTL Project** as wish to build our circuits using RTL (Resistor–Transistor Logic).

Also, tick **Do not specify sources at this time** checkbox as we do not plan to import any existing files into the project (we will do so in a later lab).

Click **Next** button to continue.



New Project

Project Type
Specify the type of project to create.

☒ **RTL Project**
You will be able to add sources, create block designs in IP Integrator, generate IP, run RTL analysis, synthesis, implementation, design planning and analysis.
☒ Do not specify sources at this time

☐ **Post-synthesis Project**: You will be able to add sources, view device resources, run design analysis, planning and implementation.
☐ Do not specify sources at this time

☐ **I/O Planning Project**
Do not specify design sources. You will be able to view part/package resources.

☐ **Imported Project**
Create a Vivado project from a Synplify, XST or ISE Project File.

☐ **Example Project**
Create a new Vivado project from a predefined template.

? < Back Next > Finish Cancel

Figure 5 – Project type dialog box.

Step 4: Development Board

Next, we need to select the development board we plan to use.

Click on the **Boards** tab. Select **Basys3** and click on the **Next** button to continue. If Basys3 does not appear see tip below.

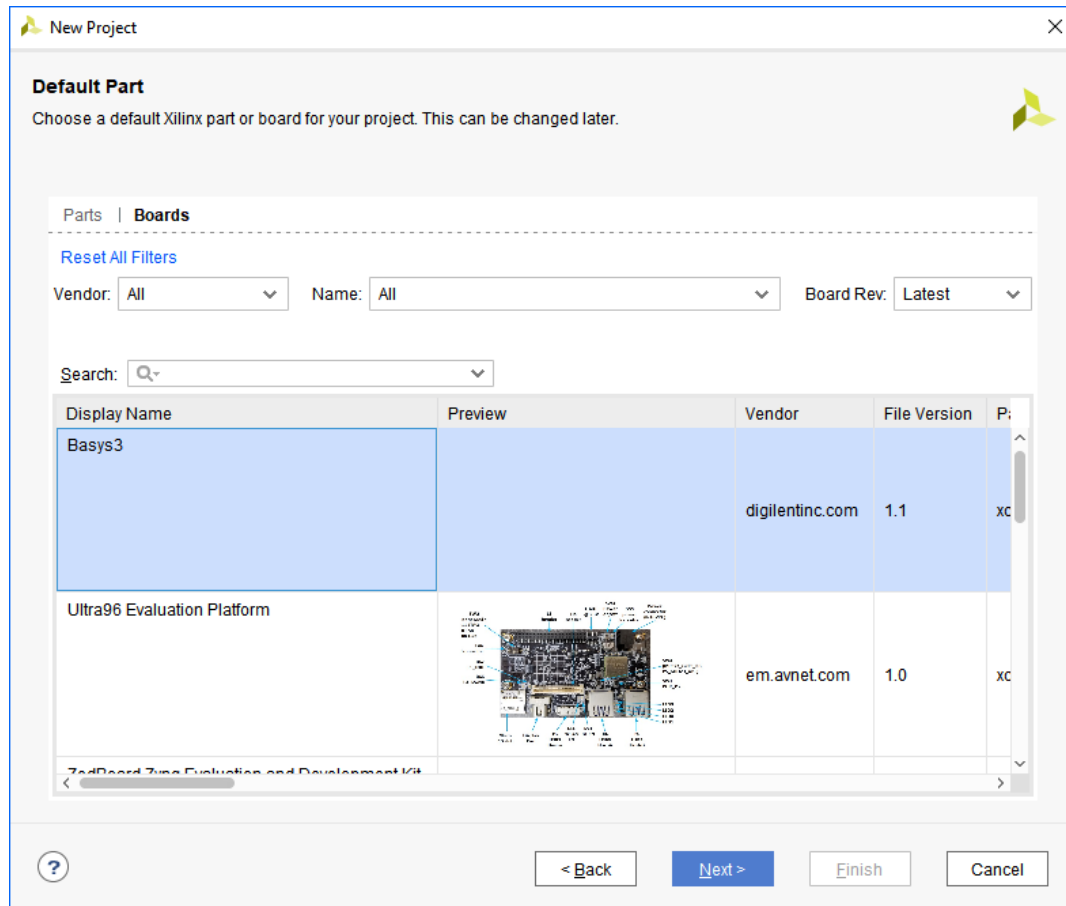


Figure 6 – Project part dialog box.

Tips In case Basys3 does not appear under Boards; you need to add the Board Files to Vivado. For this follow step “3. Installing Digilent Board Files” at <https://reference.digilentinc.com/vivado/installing-vivado/start>. Another option is to download the files from vivado-boards Github repository (link available in above URL or files in LMS/Moodle) and copy the vivado-boards-master\new\board_files\basys3 folder to <Xilinx installation path>\Vivado\2018.2\data\boards\board_files\. After this you need to close and open the Vivado again and follow all steps up to this point.

Then you will see the project summary. Click on the **Finish** button to create the project files.

Once the new project has been created, Vivado opens the project in Project Navigator as in Fig. 7. Navigator can be split into three areas as Flow Navigator (located on the left), Console panel (on the bottom), and Project Manager (on the right). Project Navigator allows us to go through the simulation, synthesis, implementation, and programming processes in sequence. Console panel displays status messages, including error and warning messages. Project Manager displays HDL and simulation code or the schematic corresponding to the given HDL code.

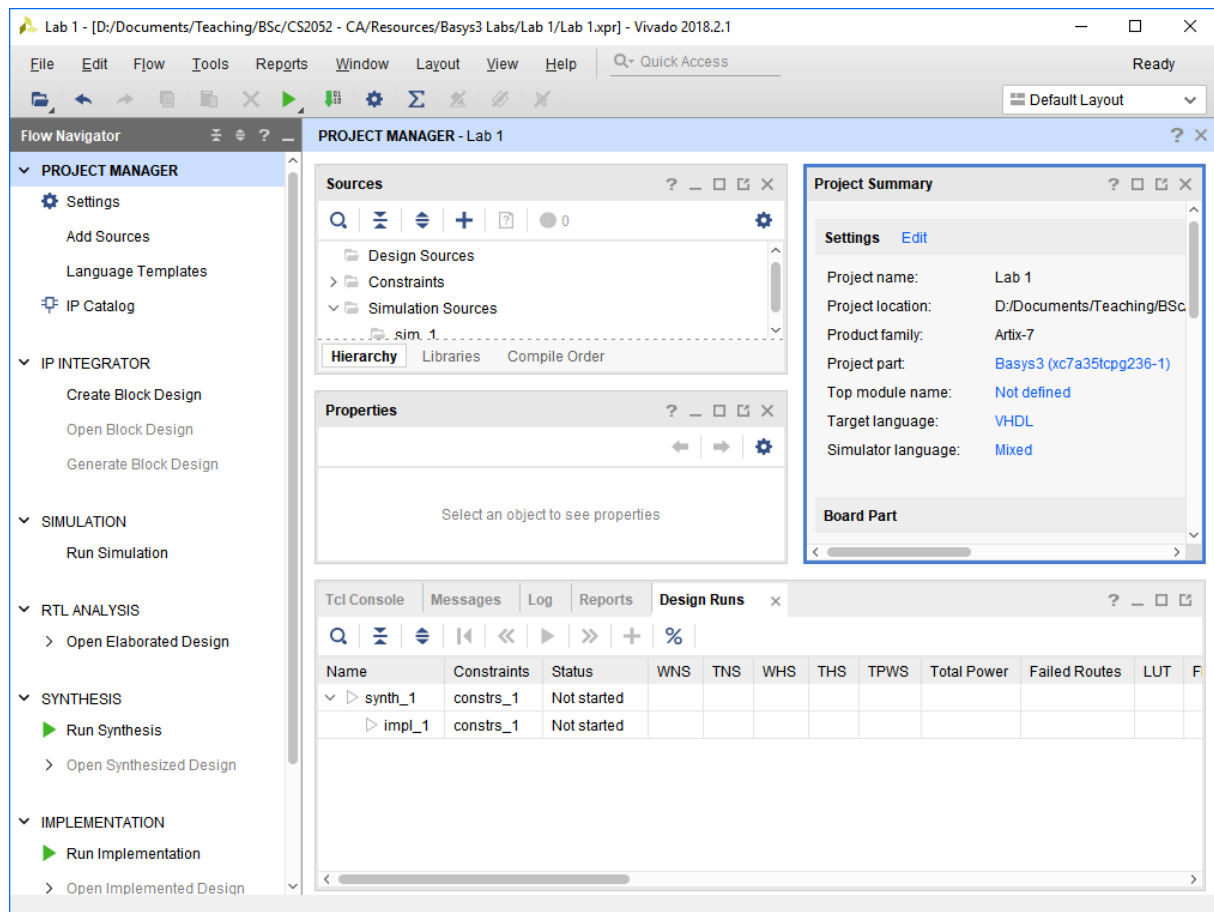


Figure 7 – Project Navigator view.

Step 5: Adding New Source Files

We need to add our circuit design to the project. Circuit design files are referred to as source files. In these labs we will develop our circuits using VHDL (**V**ery **H**igh **S**peed **I**ntegrated **C**ircuit **H**ardware **D**escription **L**anguage). VHDL is a language that is used to describe the behavior of digital circuits. For example, rather than drawing a schematic symbol of a 2-input AND gate, we could describe the same using a statement like `a AND b`. VHDL also supports hierarchical design (e.g., building a Full Adder using 2 Half Adders); hence, enables rapid development of complex digital circuits. The circuits defined in VHDL can be simulated and translated into a form suitable for hardware implementation such as FPGA. We will introduce various VHDL commands and keywords in different labs. Also, explanation of certain VHDL commands and behaviours are left for future labs to make the labs easier to follow.

To add a new or existing source file to a project, click on the **+** button or press **Alt + A** in the **Sources** pane of the **Project Manager** panel as seen in Fig. 8.

This will open up the Add Sources wizard dialog box. Select **Add or create design sources** and click **Next >** button.

Then in the next dialog box click on the **Create File** button. From the popup select **VHDL** as the **File type** and set **Lab1** as **File name** as seen in Fig. 9. Then click **Ok** button to close the popup, and then click **Finish** button to close the dialog box.

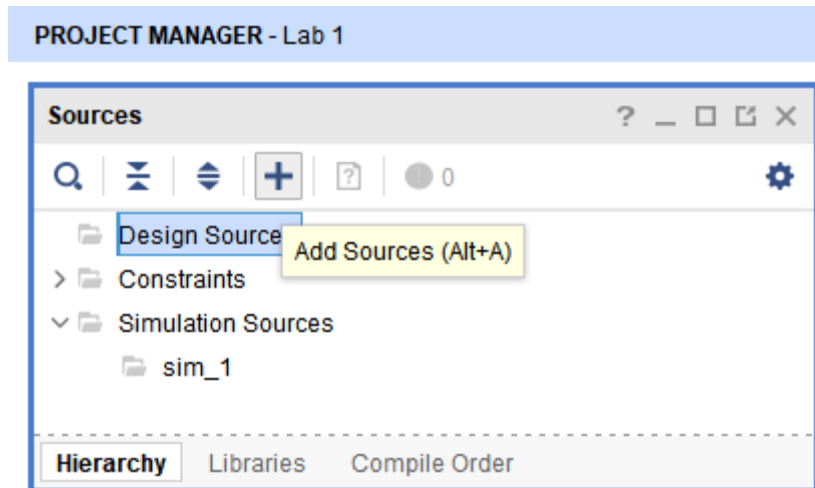


Figure 8 – Adding a new source file.

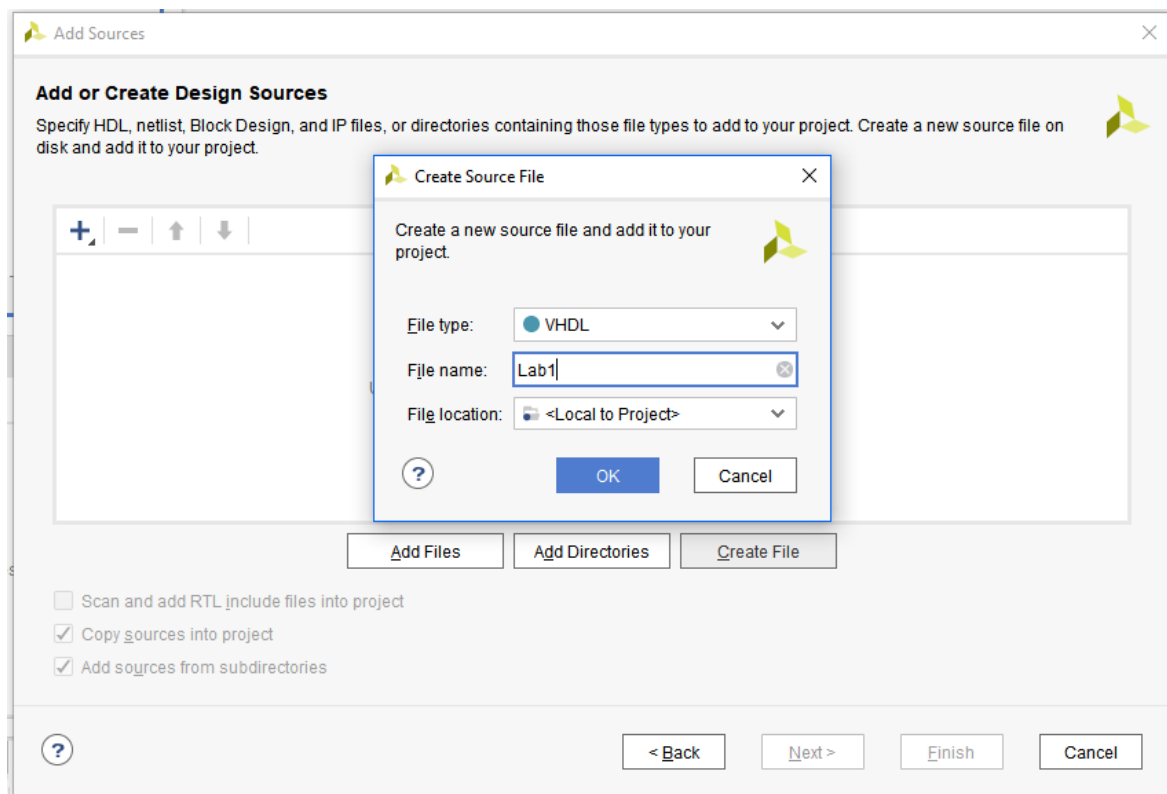


Figure 9 – Source file properties.

Step 6: Building the Circuit

While creating the VHDL file we need to indicate the inputs and outputs of the digital circuit. Let us create the logic circuit depicted in Fig. 10. As the circuit has 3 inputs *A*, *B*, and *C* and an output *X*, fill Define Module dialog box as seen in Fig 11. Ports *A* to *C* are defined as *in* (i.e., inputs) and port *X* is defined as *out* (i.e., output). To add a new line for port *X* click on the + button. Then click **Ok** button.

You can open the new VHDL file by double-clicking **Lab1(Behavioral) (Lab1.vhd)** by expanding **Design Sources** under Project Manager – Sources. Resulting VHDL code would look similar to Fig. 12.

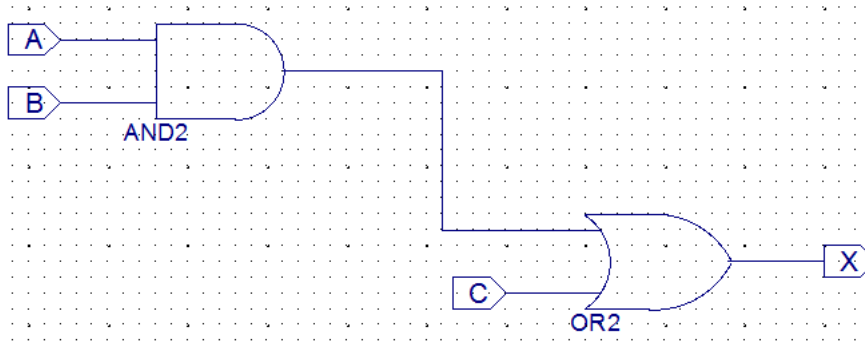


Figure 10 – Schematic view of the circuit.

Define a module and specify I/O Ports to add to your source file.
For each port specified:
MSB and LSB values will be ignored unless its Bus column is checked.
Ports with blank names will not be written.

Module Definition

Entity name:

Architecture name:

I/O Port Definitions

Port Name	Direction	Bus	MSB	LSB
A	in	<input type="checkbox"/>	0	0
B	in	<input type="checkbox"/>	0	0
C	in	<input type="checkbox"/>	0	0
X	out	<input type="checkbox"/>	0	0

OK Cancel

Figure 11 – Setting inputs and outputs of the module.

First 20 lines or so on the VHDL are comments. Add **your name, project name, target device, and description** under the comments.

The line `library IEEE` indicates the use of IEEE library which consists of a set of packages, components, and functions that are prebuilt to simplify the task of hardware design. Whereas `use IEEE.STD_LOGIC_1164.ALL;` indicate the use of all data types and subprograms available under the IEEE.STD_LOGIC_1164 package. These 2 lines are required in all our labs.

A VHDL design consists of entities, architectures, and configurations. An *entity* is a specification of a design's/module's external interface. It typically specifies the name and a set of inputs and outputs of the entity. In our case *Lab1* is the name of the entity. Ports A to C are marked as input while port X is marked as output (this is based on the configuration we added in Fig. 11). In `A : in STD_LOGIC` A is the port name, *in* indicates it is an input port, and *STD_LOGIC* indicate input enables single-bit basic logic operations.

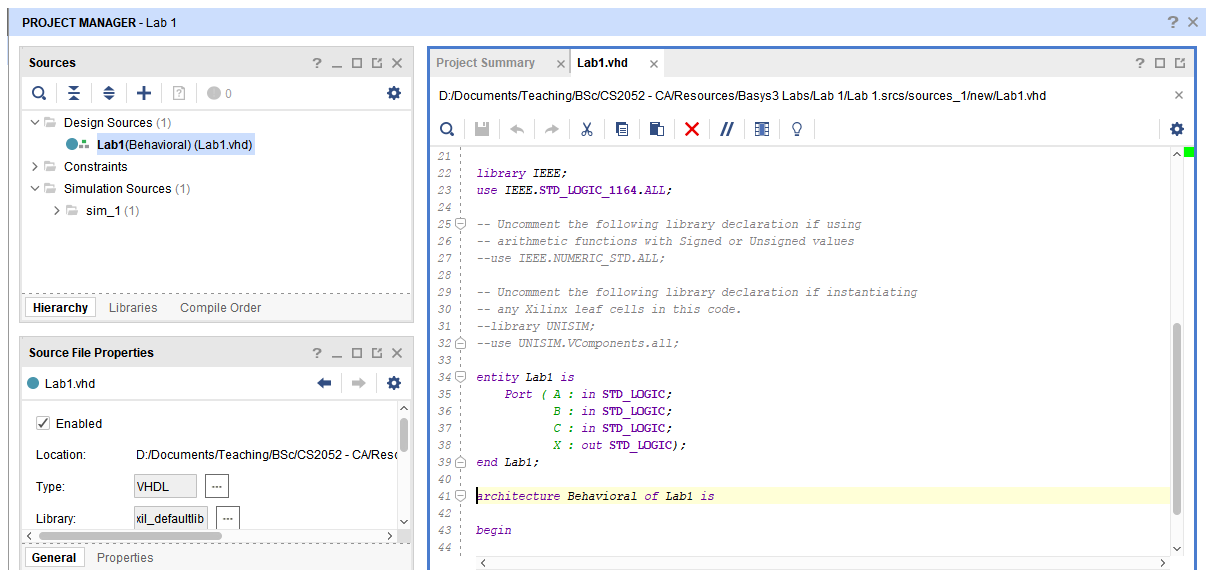


Figure 12 – Contents of Lab1.vhd.

Architecture defines the internal design of the entity/model. Thus, it defines the functionality of the entity. The same entity may have multiple designs based on the number of logic gates used, performance, power consumption, and use of FPGA area. The entity to architecture mapping is achieved using configurations. However, this does not matter to us as our labs focus on building only a single architecture for an entity.

Let us build the circuit in Fig. 10 using two logic gates. For this, we need to keep track of the output from AND gate which needs to be fed into the OR gate. As in any programming language we can use variables in VHDL to keep track of such intermediate values. Declaration of a variable should follow the following format:

```
signal <variable name> : <type of logic>;
```

Note that each VHDL line terminates with “;”. For example, variable could be defined as follows:

```
signal AND_out_sig : std_logic;
```

Variables need to be defined before the `begin` keyword inside the architecture block. Then we can define the $X = (A \cdot B) + C$ behaviour of our circuit as follows:

```
AND_out_sig <= A AND B;
```

```
X <= AND_out_sig OR C;
```

VHDL is case sensitive; therefore, be consistent with the entity, variable, input, and output names. The completed VHDL code should be similar to Fig. 13.

To save your code use shortcut **Ctrl + S** or click on the **Save** icon.

Tips Do not cut and paste VHDL from PDF files, as it may introduce hidden characters which Vivado is not able to interpret correctly. Copying from .vhd and text files is usually ok.

```

34 entity Lab1 is
35     Port ( A : in STD_LOGIC;
36           B : in STD_LOGIC;
37           C : in STD_LOGIC;
38           X : out STD_LOGIC);
39 end Lab1;
40
41 architecture Behavioral of Lab1 is
42     signal AND_out_sig : std_logic;
43
44     begin
45         AND_out_sig <= A AND B;
46         X <= AND_out_sig OR C;
47     end Behavioral;
48

```

Figure 13 – Completed VHDL code.

Step 7: Creating Design Constraints File

A Xilinx Design Constraints (.xdc) file defines user constraints like the physical *pin* to circuit *net* mappings (i.e., mapping the buttons and LEDs on the BASYS 3 board to inputs/outputs in the circuit). It informs the software what physical pins on the FPGA that you plan on use or connect them your VHDL code. The .xdc file can be created and modified inside Vivado using a text editor.

Rather than writing a new .xdc file for every lab, you can use a predeveloped .xdc file with all the BASYS 3 pins and enable and rename only the required pins.

Download the **Basys3Labs.xdc** from LMS (i.e. Moodle) and import it to your project using the following steps.

Click on the **+** button on the Sources pane. Then select **Add or create constraints**. Click **Next >** button.

Click on **Add Files** button and locate the **Basys3Labs.xdc** file that you downloaded. Click **Ok** button. Click **Finish** button.

Expand **Constraints – constrs_1** in the Sources pane and then double click on **Basys3Labs.xdc** to open the file. The entire file is commented (a comment starts with # symbol) and you need to uncomment only the relevant lines and update them as needed.

As we use *A*, *B*, *C*, and *X* in our circuit, port names need to be replaced with those labels. For our project, let us assign three inputs *A*, *B*, and *C* to switches 0 through 2 (SW0 – SW2) and output *X* to LED0 (LD0) on the BASYS 3 board.

Download “Basys 3 FPGA Board Reference Manual” from LMS and note that switch 0 (SW0) is connected to pin A17 on Atrix-7 FPGA on BASYS 3 (see pg. 15). Similarly, SW1 and SW2 are connected to V16 and W16, respectively. LED 0 (LD0) is connected to pin U16. Therefore, you need uncomment the relevant lines under **## Switches** and **## LEDs**. You need to uncomment both the **set_property PACKAGE_PIN** and **set_property IOSTANDARD** lines.

The .xdc file labels the ports as an array of switches and LEDs (which is very useful when working with a set of input or output wires buses). However, as

we use *A*, *B*, *C*, and *X* those labels need to be replaced. Fig. 14 show the changes that you need to do on the .xdc file. Once the changes are over make sure to save the file.

```
## Switches
set_property PACKAGE_PIN V17 [get_ports {A}]
    set_property IOSTANDARD LVCMOS33 [get_ports {A}]
set_property PACKAGE_PIN V16 [get_ports {B}]
    set_property IOSTANDARD LVCMOS33 [get_ports {B}]
set_property PACKAGE_PIN W16 [get_ports {C}]
    set_property IOSTANDARD LVCMOS33 [get_ports {C}]

## LEDs
set_property PACKAGE_PIN U16 [get_ports {X}]
    set_property IOSTANDARD LVCMOS33 [get_ports {X}]
```

Step 8: Generating the Programming File

Now we are ready to create a programming file (a.k.a. bitstream, .bit file) for the FPGA in BASYS 3.

Go to the **Flow Navigator** panel and click **Run Synthesis**. It will popup Launch runs window. Keep the default options as it is and click **Ok** button. This initiates the synthesis process (see Fig. 2) which may take a couple of minutes to complete depending on the complexity of the circuit and performance of your computer.

If the synthesis is successful, you will get a window similar to Fig. 14. Select **Run Implementation** and click **Ok** button.

If the synthesis failed check the message under Tcl Console and Messages tabs of the Console panel at the bottom. VHDL syntax errors usually result in failures at this stage. If there are any errors your VHDL file name(s) will be underlined in Sources panel. Any pending errors need to be corrected. Then reinitiate the synthesis by clicking on **Run Synthesis** button again.

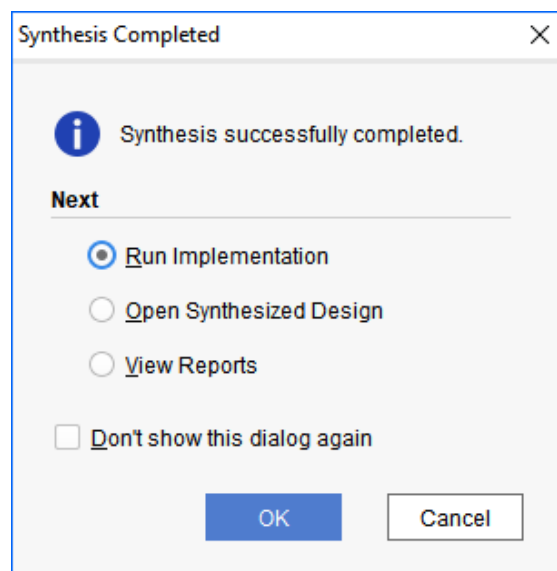


Figure 14 – Results of synthesis phase.

Then Launch runs window will appear again. Click **Ok** button to proceed. This process may also take a couple of minutes to complete. If the implementation is successful, you will get a window similar to Fig. 15. Select **Open**

Implemented Design and click **Ok** button. This will show how your synthesized design is arranged on the FPGA, which is not essential to understand at this stage.

If any errors are reported by implementation process those need to check and corrected. Then rerun the implementation by clicking on **Run Implementation** on Flow Navigator panel.

You can also see the schematics of your circuit by clicking on **Schematic** under Implementation dropdown list on Flow Navigator panel. This should show a logic circuit similar to Fig. 16. We can see A, B, C, and X as inputs and outputs. LUT is our combinational circuit which takes in 3 inputs and produce one output. All inputs and outputs are connected through buffers (IBUF and OBUF), which we will discuss in a later lab.

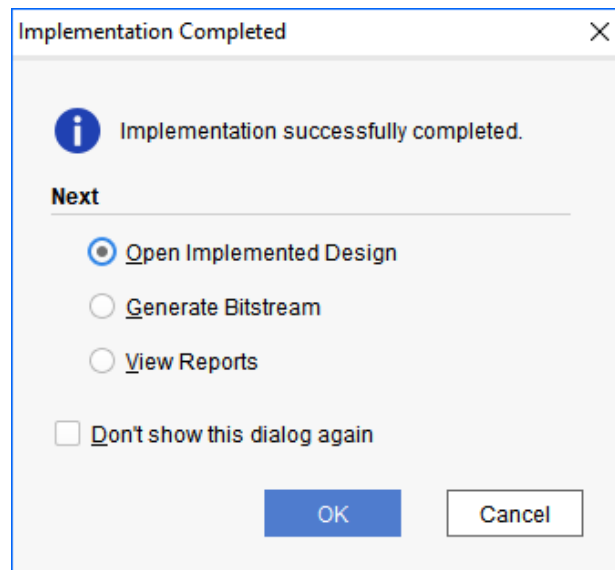


Figure 15 – Results of implementation phase.

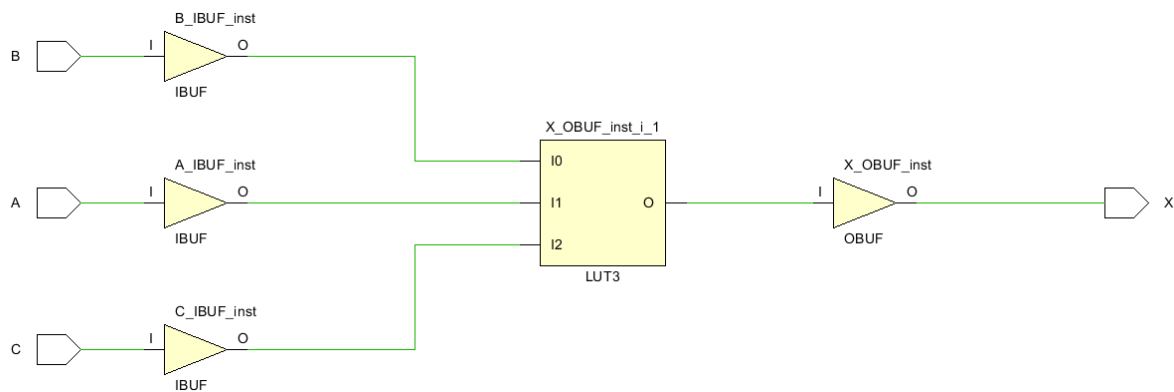


Figure 16 – Schematic representation of the circuit.

Once both the synthesis and implementation are successfully complete we can generate the bitstream to be downloaded to the device. Use following steps to generate the bitstream and transfer that to device.

Go further down on Flow Navigator panel and click on **Generate Bit Stream**. Click **Ok** button on the popup window. This process may also take a couple of minutes to complete. Once complete a window similar to Fig. 17 will popup. Click **Ok** button.

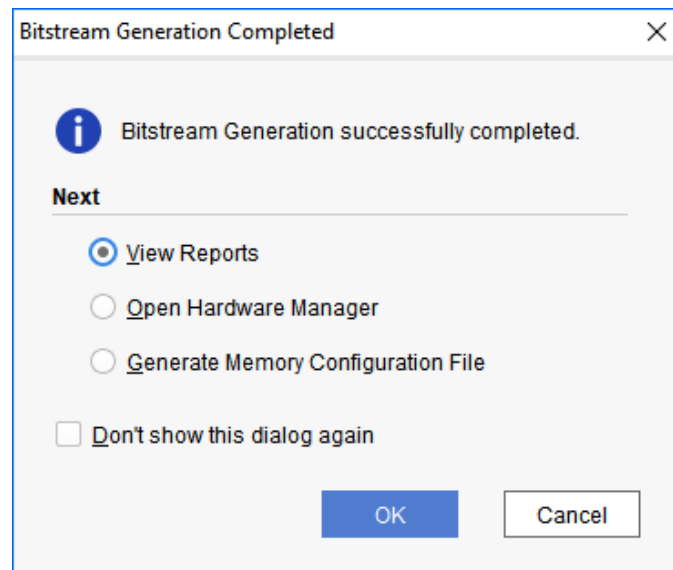


Figure 17 – Results of bitstream generation phase.

Step 9: Configuring the Board

Talk to the instructor to get access to BASYS 3 board. Connect the BASYS 3 to the computer using the USB cable. Following steps should be carried out only on a computer with BASYS 3 attached.

Warning

Never touch/hold the development board from the top, bottom, or middle. Static charges can ruin the small electronic components. Always hold the board from the sides as shown in Fig 18.

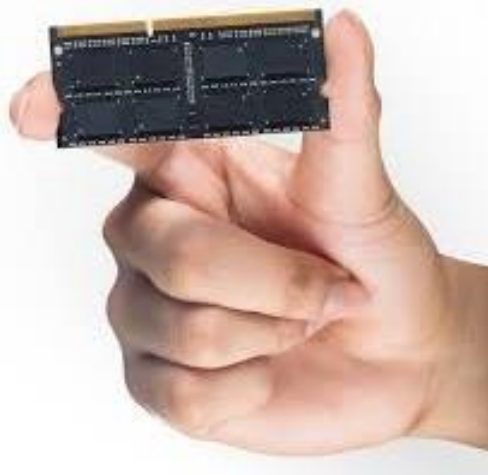


Figure 18 – How to hold a circuit board.

Make sure BASYS 3 is connected to the computer using a USB cable. Also, make sure **JP1** jumper on the board is set to **JTAG**. This tells the program to be loaded from the computer not from ROM.

Set the **POWER** switch to **ON**.

Then click on **Open Hardware Manager** at the bottom of Flow Navigator panel. Then expand the dropdown list and click on **Open Target**. Then from the popup click **Auto Connect**. Now Vivado try to connect to the BASYS 3 board. If successful, it will get listed under Hardware manager as in Fig. 19.

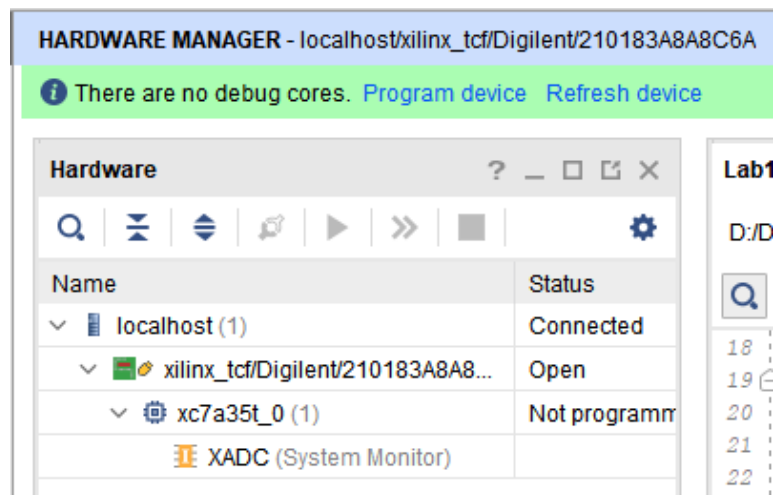


Figure 18 – Hardware manager when successfully connected to the board.

Click on **Program Device** (see the bottom of Flow Navigator panel) and then select **xc7a35t_0** which is the name of the FPGA. Click on **Program** button from the popup window. If any errors are experienced in this process, they will be notified.

Step 10: Change the switches (SW0 – SW2) on the BASYS 3 and verify the functionality of your circuit (check the output on LED LD0).

Note As we did not program the ROM, the board will lose the FPGA configuration once the power is switched off. For the labs, we do not need to permanently program the board. Hence, even in future labs, we will not program the ROM.

Demonstrate the circuit to the instructor and get the Lab Completion Log signed.

Congratulations!!! for successfully completing your first lab!

If you finish a bit early, replace lines 45 and 46 in Fig. 13 with only a single line to perform $X = (A \cdot B) + C$. In this case you do not need the variable in line 42. These lines can be commented using `--`. Return the synthesis, implementation, and bitstream generation and test on the board.

Bibliography

- Digilent Inc., “Basys 3 Programming Guide.
- Digilent Inc., “Basys 3 FPGA Board Reference Manual,” Mar. 23, 2017.
- Xilinx, “Vivado Tutorial – Lab Workbook,” 2015.

Prepared By

- Dilum Bandara, PhD on Jan 26, 2014
- Last Updated on Sep 20, 2018

Note This lab is not compatible with the previous version of this document as this replaces BASYS 2 with BASYS 3, ISE with Vivado, and schematic design with VHDL.