Experiment 2

Using Vivado

In this lab, we are going to implement a simple combinational circuit given in Fig. 2.1 on our FPGA. For this purpose, we will learn about assigning I/O pins of our modules for synthesizing the designs on the FPGA. After that, we are going to use Vivado to burn our System Verilog code on the FPGA.

Overview of Vivado Design Suite

Vivado design suite software developed by Xilinx for the synthesis and implementation of System Verilog designs. For this lab we will use the web-pack edition of Vivado.

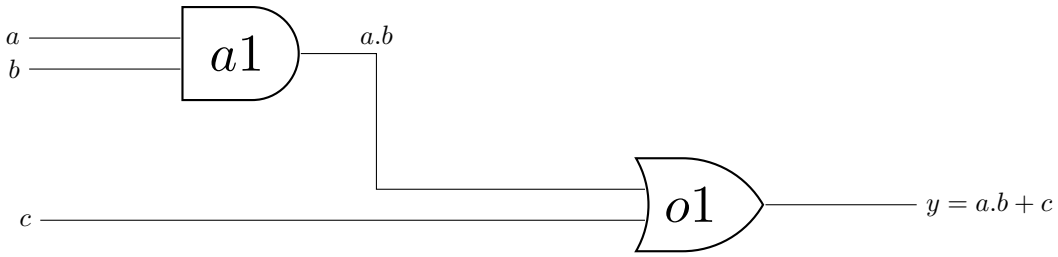

Fig. 2.1: Simple logic circuit implementing $y = a.b + c$

Creating a new project in Vivado

1. From *Quick Start* click on *Create Project*. A new project dialog box will appear. Click on *Next*. (Fig. 2.2)

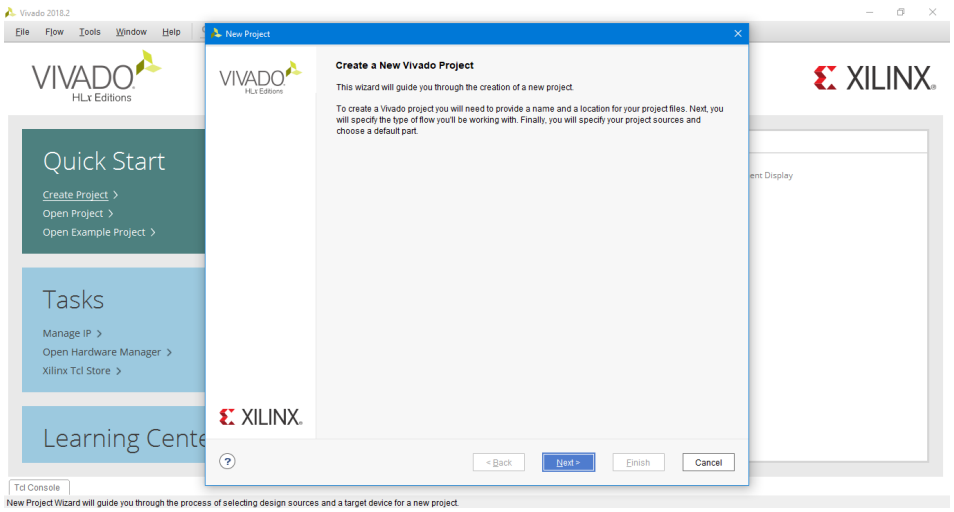

Fig. 2.2: Project Dialog Box

2. Write the project name. Click on *Next*.
3. A *Project Type* dialog box will appear. Select *RTL Project* and check the *Do not specify sources at the time* box. Click *Next* to continue. (Fig. 2.3.)

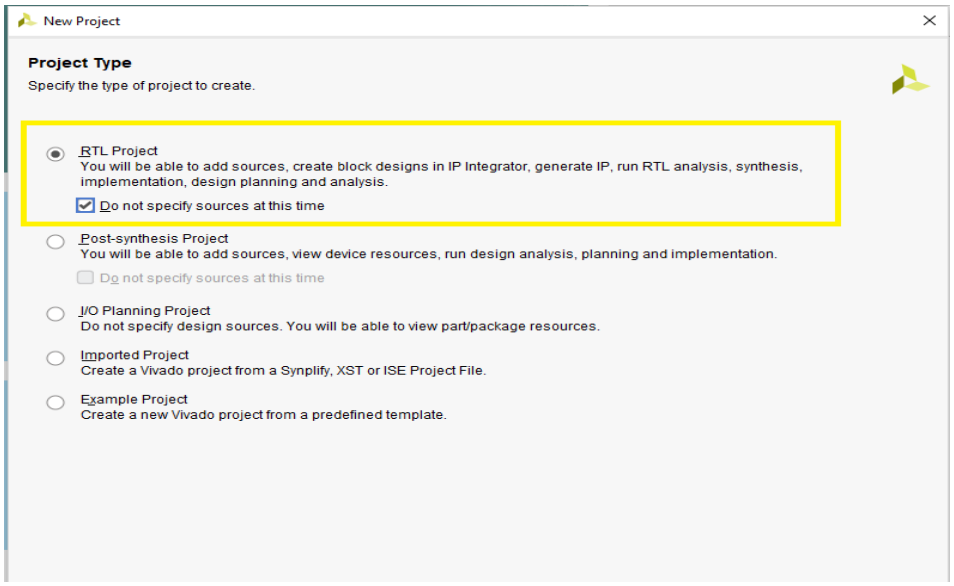

Fig. 2.3: Project Type Dialog Box

4. A *Default Part* dialog box will appear. Select the same family, package and speed of the board. Then, from the parts shown select *xc7a100tcsg324-1*. (Fig. 2.4.) Or select the *Boards* tab and it will display a list of boards. Select the board Nexys A7 100T and Click *Next* to Continue.

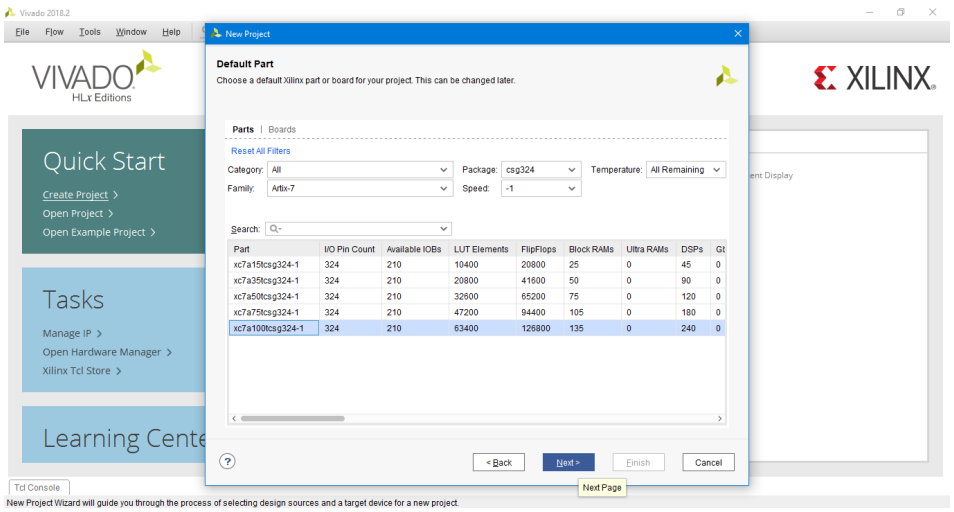

Fig. 2.4: Default Board Dialog Box

5. Then the *Project Summary* will appear. Click on *Finish*. After this, the project will be initialized and window will appear (Fig. 2.5). Project Summary will contain information about the project settings, the board used along with the number of errors and warnings that encountered when synthesis, implementation and bit-stream generation have been completed. Clicking on these will show their details in the *Logs, Messages* window.

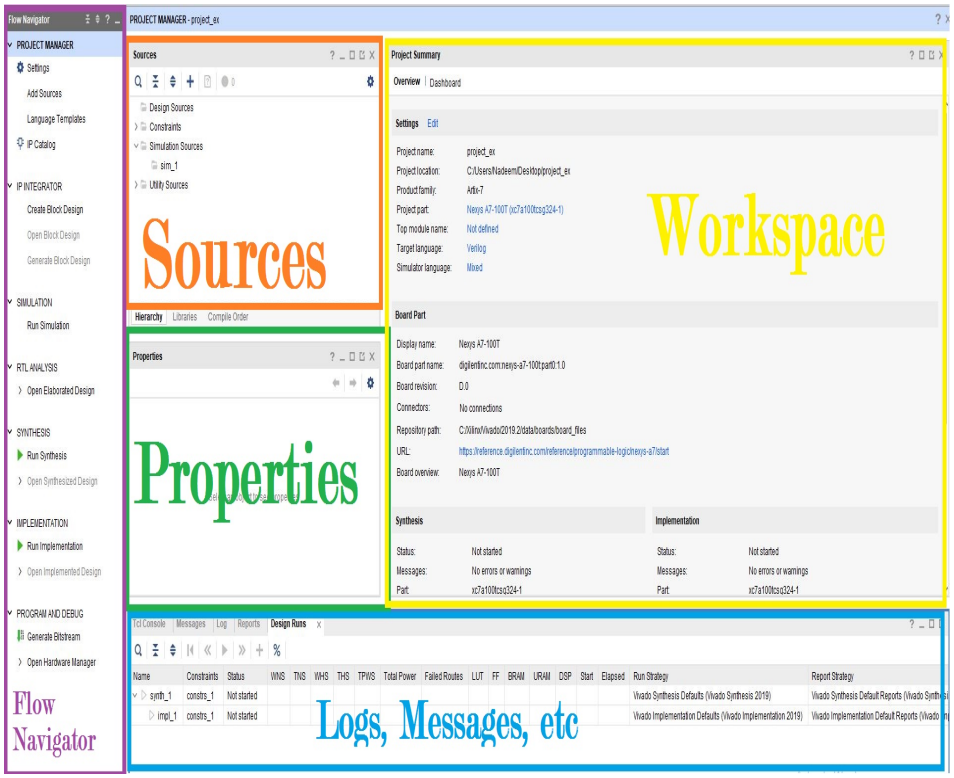

Fig. 2.5: Vivado Design Suite Window

Creating a new RTL File on Vivado

1. Click on *Add sources* from the project manager bar. (Fig. 2.5). A dialog box will appear. Select *Add or create design sources* and click *Next*.

2. *Add sources* dialog box will appear from there click on *Create File*. When prompted, select a *File type, File name, and File location*. Make sure to pick *System Verilog* and *Local to project* for the type and location. Give your file a name ending in **.sv**. Click on *Ok* and then on *Finish*. When *Define Module* dialog box appears, inputs and outputs can be defined. Otherwise they will be defined in the System Verilog file.

3. From the Sources, open the file from *Design Sources* and write the code provided in Listing. 1.

```
module lab2(output logic y,
            input  logic a,b,c
            );
  wire o;
  and a1(o, a, b);
  or  o1(y, o, c);
endmodule
```

Listing 1: System Verilog Code.

4. Click on *Schematic* to check your gate level design (Fig. 2.6).

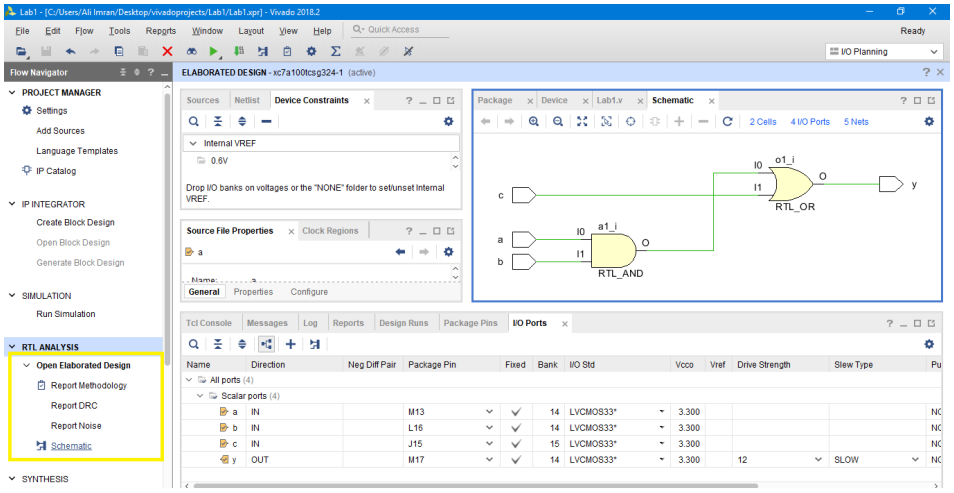

Fig. 2.6: Vivado Gate Level Schematic

Assigning Package Pins

There are two methods for assigning package pins which are given as follows:

Using I/O Planning

1. In the project manager bar, click on *Open Elaborated Design*. A dialog box will appear, click on *Ok*. The I/O ports window will open (Fig. 2.7), (if the I/O ports box does not appear then go to *Layout* and select *I/O planning*). Open the***I/O ports*** tab and assign your input output pin connections under ***Package Pin*** according to Fig. 2.7

2. In the ***I/O Std*** choose your required voltage level (*LVCMOS33.*) then press ctrl S and save the constraints file as XDC file.

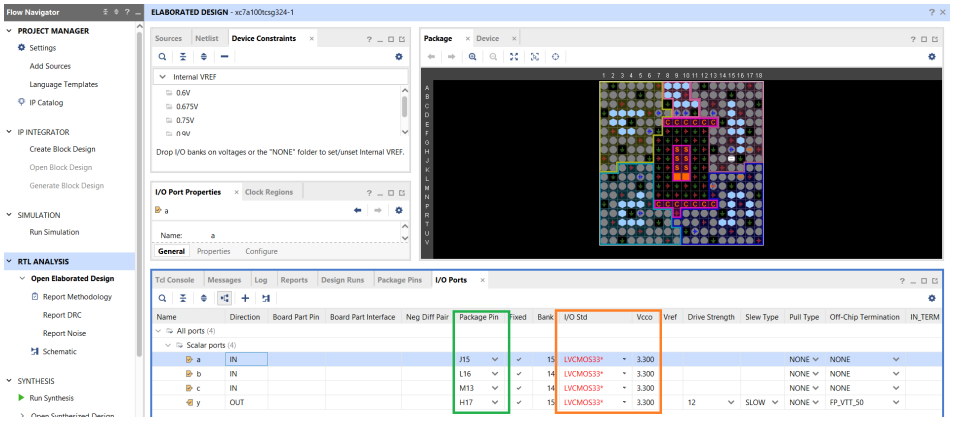

Fig. 2.7: Assigning Package pins on Vivado

Using Constraints File When we save the constraints in the above Section 2, we are indirectly creating a constraint file. But we can directly create that file using the following method:

1. Click on *Sources* in the project manager and then select *add or create constraints* and then select Create File. The constraints file will be created.

2. From the *Nexys A7-100T Master.xdc* file, copy the I/Os that you require. Replace the names in *get_ports* by the name of your respective input and output signals (Listing 2.1) and then save that file. The pins will be assigned in the same manner as done in Section 2.

```
set_property -dict { PACKAGE_PIN H17   IOSTANDARD LVCMOS33 } [get_ports { y }];
set_property -dict { PACKAGE_PIN J15   IOSTANDARD LVCMOS33 } [get_ports { a }];
set_property -dict { PACKAGE_PIN L16   IOSTANDARD LVCMOS33 } [get_ports { b }];
set_property -dict { PACKAGE_PIN M13   IOSTANDARD LVCMOS33 } [get_ports { c }];
```

Listing 2.1: Assigning Package pins using constraints file.

Synthesizing and Implementing the Design

Under the Project Manager bar click on *Run Synthesis*. A dialog box will appear click on *Ok*. After the synthesis is complete click on *Run Implementation* and then repeat the same process as before.

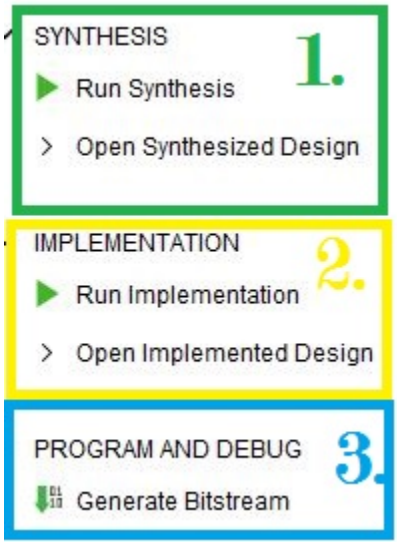

Fig. 2.8: Synthesis, Implementation, Generate BitStream

Generating the bit file and Programming the FPGA

Under the *Program and Debug*, click on *Generate Bitstream*. When the bitstream is generated, FPGA can be programmed in multiple ways.

Using Micro-USB port

1. Connect the micro-USB of the FPGA to your computer with the cable. And under the *Open Hardware Manger* right click on *Open Target* and the click on *Auto Connect* (Fig. 2.9).

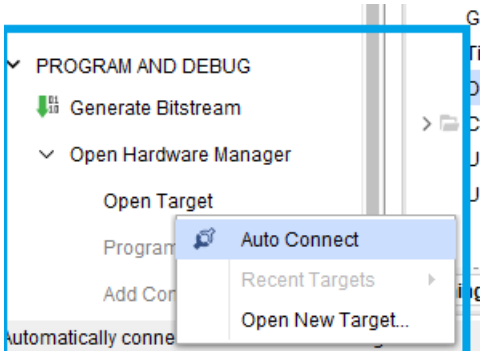

Fig. 2.9: Connecting FPGA to the computer.

2. Vivado will take a few seconds connect to the FPGA. Once done, click on *Program device* from there a dialog box appears and click on *Program* to program your FPGA (Fig. 2.10).

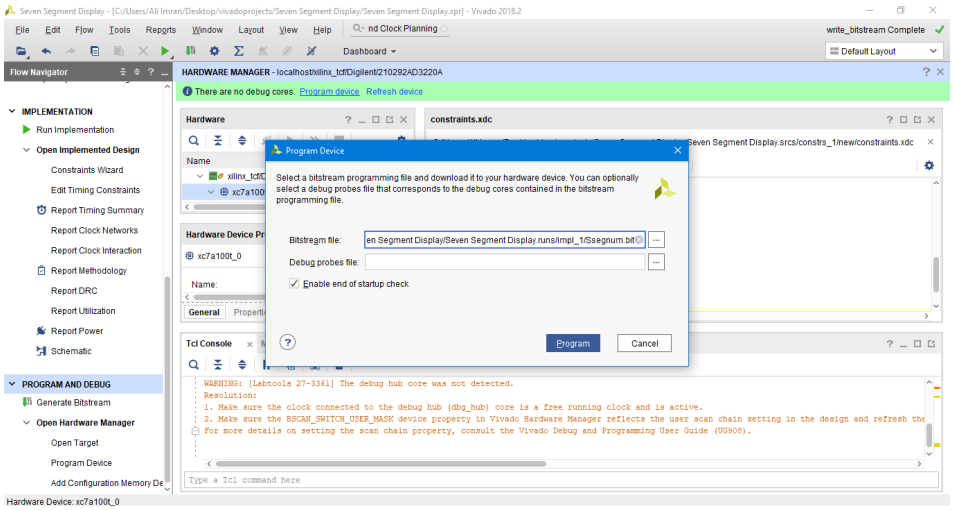

Fig. 2.10: Programming the Bitstream on the FPGA.

Using Type-A USB port

The generated bit file can be found in the project folder under *project_name.runs* $->$ *impl_1*.

1. In the search bar, type .bit. It will be the name of your top module file with *.bit* extension.
2. Copy that file into a **FAT32** format USB.
3. Plug that USB in the type-A USB port of the FPGA, and press the *PROG* button.
4. When the FPGA has been programmed with the bit file, *DONE* LED will turn on.

The RTL written in System Verilog will be implemented on the FPGA. To check the behavior of the LED, make the truth table of $a.b + c$ and check for all the possible combinations of inputs.

Maximum Combinational Delay

Click on *Open Synthesized Design*. Click on *Report timing Summary*. A dialog box will appear, click on *Ok*. A timing box will open (Fig. 2.11).

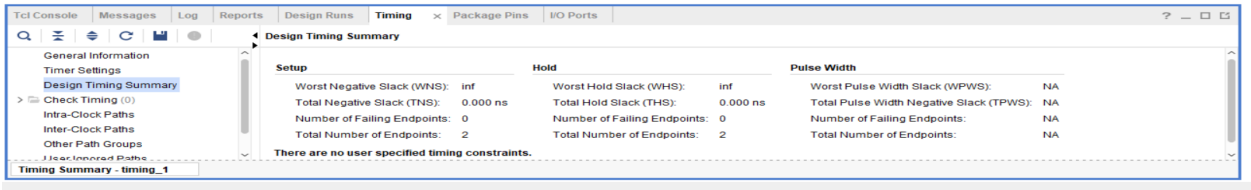

Fig. 2.11: Timing box at the bottom of the screen.

Scroll down the box, click on *Datasheet* and then *Combinational Delays*. The window will show the maximum path along with the name of path in this example it gives maximum delay for c to y with the delay of 3.5 ns (Fig. 2.12).

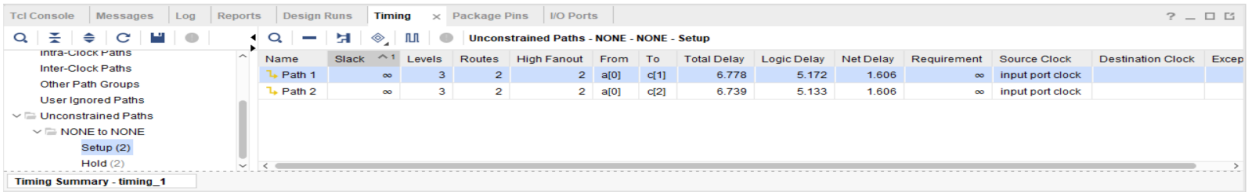

Fig. 2.12: Maximum delay for a[0] to c[1] with the delay of 6.778ns.

Device Utilization Summary

Click on *Reports* in the console tab. Select the *synth_report_utilization*. This will give you the device utilization report.

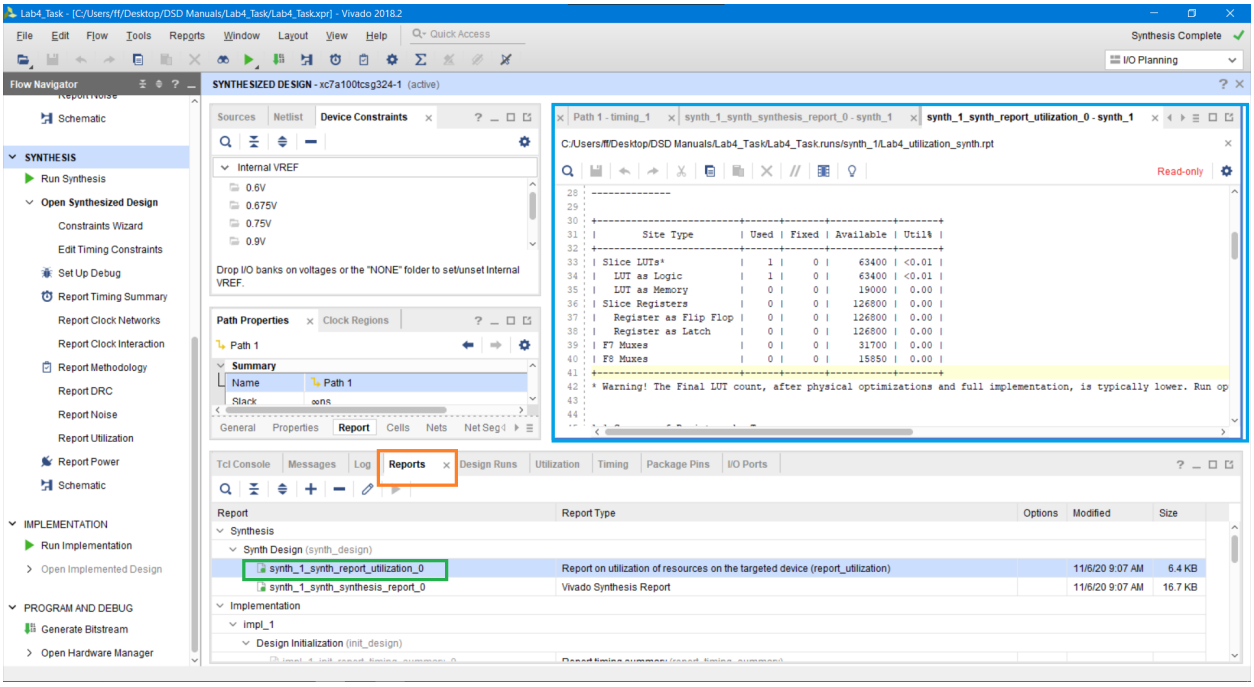

Fig. 2.13: Device Utilization Summary.

Tasks

Implement the circuit shown in Fig. 2.14 on the FPGA and develop it's truth table.

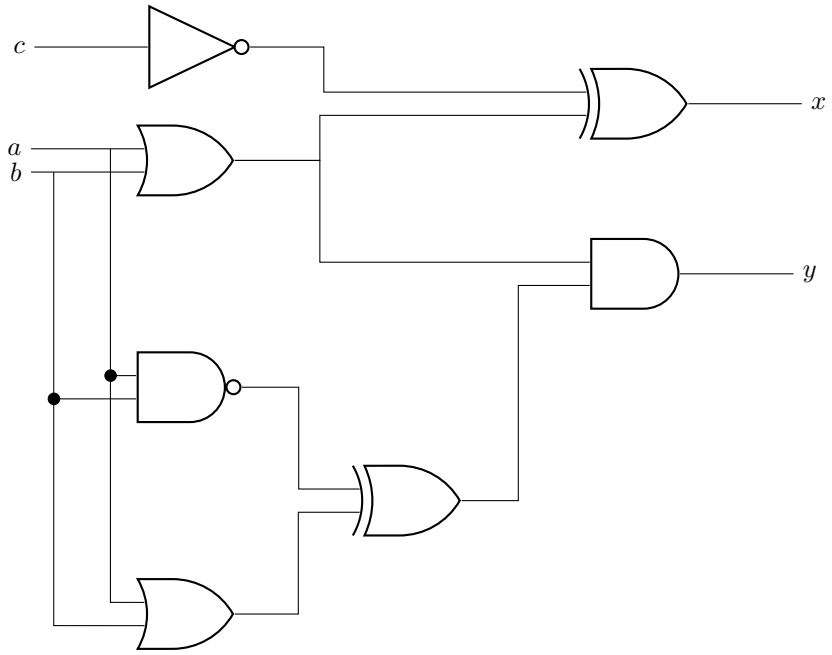

Fig. 2.14: Circuit to be implemented.

Deliverables

1. Prepare the report containing following items:
   (a) Truth table of the circuit.
   (b) Compare the circuit implemented with the schematic under RTL Analysis. Are they same. If not, how are they different?
   (c) Maximum combinational delay in Synthesis: Read the report of your circuit and describe which path has the maximum combinational delay?
   (d) Resource Utilization: Read the synthesis report and identify how many resources in the FPGA such as lookup tables (LUTs), input/output (IOs), etc., has been utilized.
2. System Verilog code for the circuit using structural modeling.
3. Compare the schematic under Synthesis and Implementation.
4. Synthesize the circuit for the starter kit available in the lab. Tie inputs to the switches and outputs to the LEDs available on the board.

The collaboration between students is encouraged, but blind code sharing/copying is not allowed. If you are unable to explain anything in your code, it will be assumed you have copied it. So make sure you know every thing you have written in your code. We are least concerned about how you have learnt something as long as you have learnt it well. Copied assignments will get ZERO marks.

Acknowledgments

The manual has been written by Mr. Ali Imran and Ms. Shehzeen Malik.