



KIIT, Deemed to be University
School of Electronics Engineering
Digital System Design Laboratory [EC 29005]

EXPERIMENT - 0

Familiarization of Laboratory Equipment and Components

Laboratory experiments based on digital circuits and logic designs provides a hands-on-experience for students with the help of standard integrated circuits (ICs) mounted on breadboards, these digital circuits can be constructed.

The following components/equipments are required for performing these experiments:

Digital logic trainer :

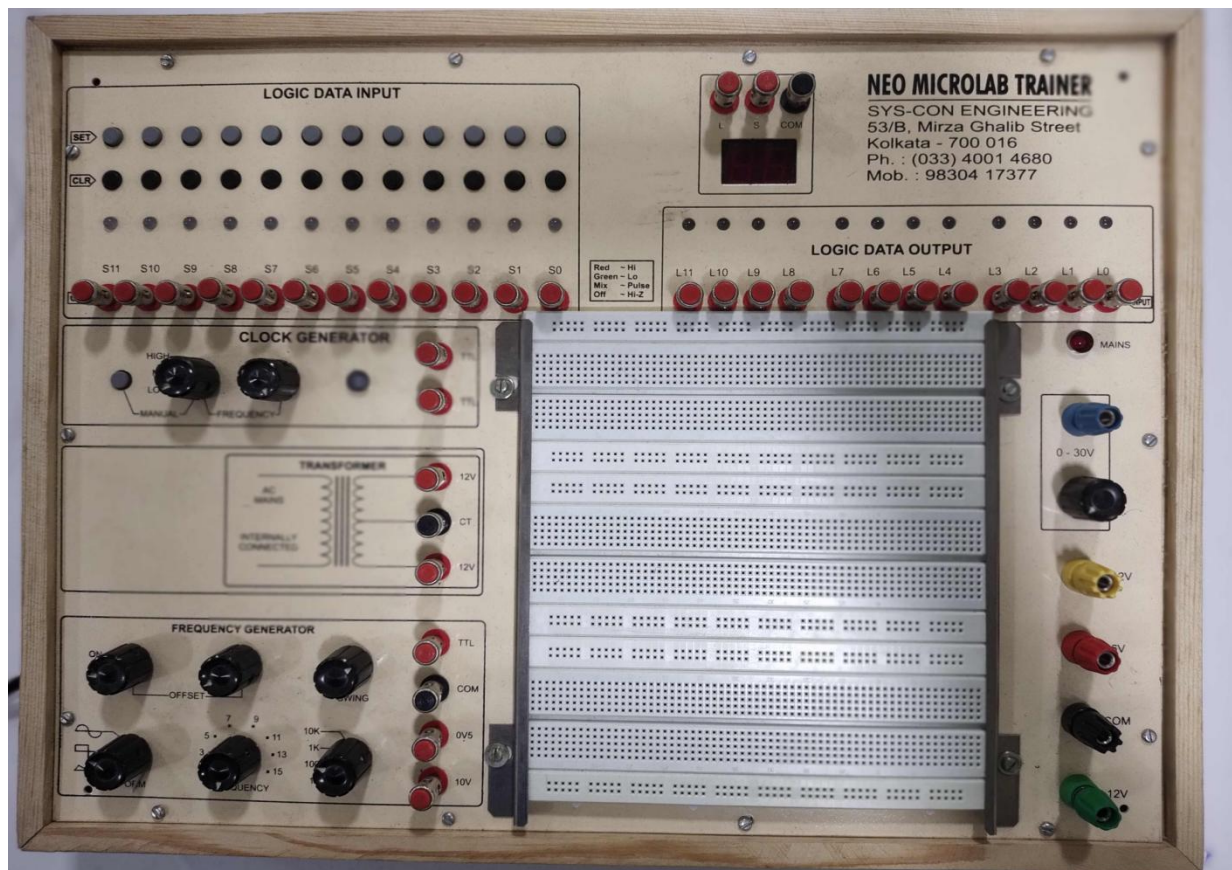


Figure 1:Trainer Kit

Digital logic trainers are tools that provide power supplies, LED lamps, toggle switches, pulsars, clock frequency generators, and IC socket strips for performing experiments on integrated circuits (ICs). The integrated circuits used in the experiments are classified based on their level of integration, such as SSI (small scale integration) and MSI (medium scale integration). SSI ICs contain individual gates or flip-flops, while MSI ICs perform specific digital functions. For the experiments involving two-input gates (OR, AND, NAND, NOR, XNOR), inverters, and four-

input NAND gates, eight SSI gate ICs are required. Additional switches, lamps, IC socket strips, extended breadboards, and plug-in-switches may be necessary for other experiments.

Breadboard :

A breadboard, also known as a protoboard or solderless board, is a device used for prototyping electronic circuits. It allows to build and test circuits quickly and easily without the need for soldering. Breadboards have a grid of holes that are interconnected by metal strips, allowing components to be easily inserted and connected to each other without the need for soldering.

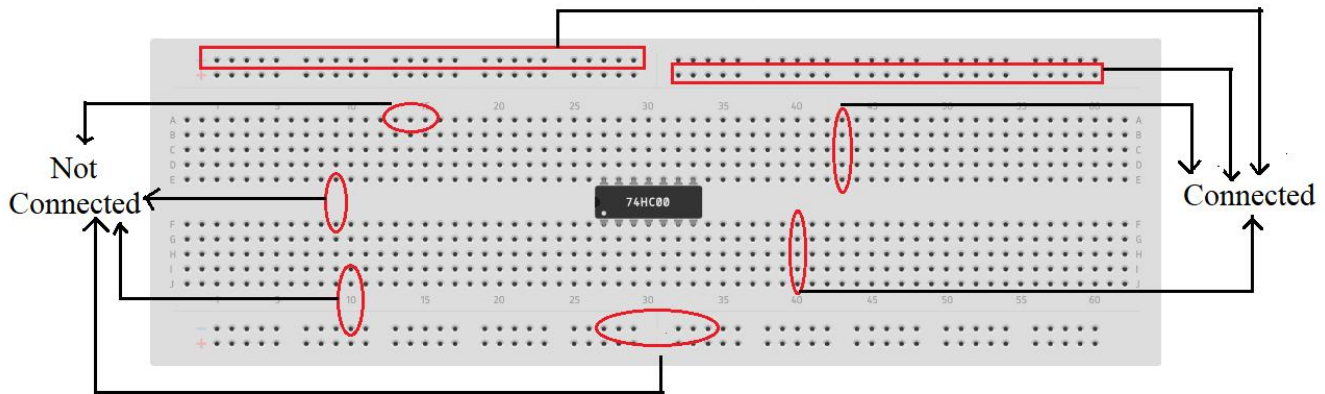


Figure 2: Breadboard Connections

The breadboard has buses across the top that can be connected to logic-1 (+5V DC for TTL family) and logic-0 (ground). Each bus has two halves, and jumper wires (single core) can be installed across the hookup wire to create a continuous bus. Integrated circuits (ICs) can be plugged into the breadboard so that they span the gap, and each pin is accessible via vertical groups of five continuous connections.

Short wires are connected to the power supply connection of each IC to be used. Switches can be used to supply logic-1 (+5V) and logic-0 (ground) to the inputs, or inputs can be wired directly to logic-1 and logic-0 with hookup wire in the inputs of the gate. Outputs are connected to LEDs .

Light Emitting Diode (LED):

It is a semiconductor device that emits light when an electric current passes through it. LED, like a PN junction, conducts current when forward biased and blocks the current when reverse biased. LEDs are not reverse-biased because, with more than a few volts of reverse bias, the LED is aged. The light output of the LED increases with increasing current until the junction gets too hot and burns out. A resistor is, therefore, invariably used in series with the LED to limit the current

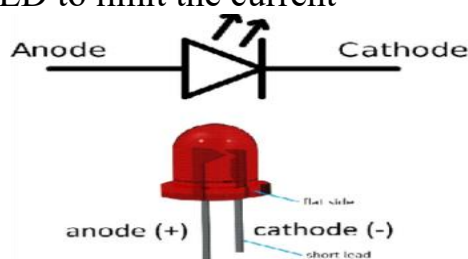


Figure 3:LED identification

Seven-segment display:

The seven-segment display is actually eight separate LEDs (seven segments and one decimal point). The seven-segment display format is used in other types of displays and can display any number from 0 to 9. Figure 4 shows the layout of the seven segment displays. The seven LEDs are labeled A through H.

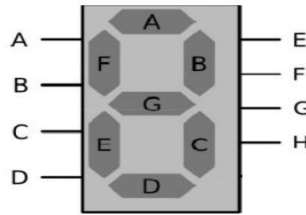


Figure 4: Seven-segment display

We can show the decimal numbers 0 to 9 as well as a few letters of the alphabet by forward-biasing various LEDs. For example, in order to show a 0 we must illuminate parts A, B, C, D, E, and F.

Power Supply:

The power supply has a fixed +5V output for work in TTL or a variable supply for work in CMOS. Be sure that the supply is set to TTL. If your trainer or DC power supply does not have a fixed +5V output, connect the voltmeter across the output of a variable supply and adjust it to +5V. A higher supply voltage can destroy a TTL integrated circuit. Switch off supply voltages before inserting or removing integrated circuits (ICs).

Logic IC :

Commonly used logic IC families are:

1. Standard TTL (type 74XX/54XX)
2. CMOS (type 4XXX)
3. Low power Schottky TTL (type 74LS/54LS)
4. Schottky TTL (type 74S/54S)
5. ECL (type 10,000)

TTL transistor-transistor logic is widely used family of digital devices which was introduced in 1964 by Texas Instruments. TTL is fast, inexpensive, and easy to use. The digital IC7404 is an example of a standard TTL. Over the years, subfamilies of TTL have been developed that have superior characteristics.

The most popular range of commercial TTL devices is the 74 series, which will operate over the temperature range of 0⁰ C to 70⁰ C. The supply voltage required for these gates is:

Maximum: +5.25 V Typical: +5.00 V Minimum: +4.75 V

A pair of digits is used to code the device and these distinguish the logic function of the chip. For example: 7400 is a standard quad 2-input NAND gate.

There are many subfamilies of this group which are distinguished by infix letters in the device coding: e.g. 74LS00 device code-LS are the infix letters. The sub-families are compared by their switching speed and power consumption as shown in the table below.

| Symbol | Type of device | Switching speed | Power consumption |
|-----------------|---------------------------|-----------------|-------------------|
| No infix letter | Standard gate | 10 ns | 10 mW |
| L | Low power | 33 ns | 1 mW |
| LS | Low power Schottky series | 10 ns | 2 mW |
| AS | Advanced Schottky | 1.5 ns | 22 mW |
| ALS | Advanced LS Schottky | 4 ns | 1 mW |
| H | High Speed | 6 ns | 22 mW |
| S | Schottky | 3 ns | 20 mW |

Table 1: Sub-families of the 74 series of TTL gate

Letters following the 74 denote the subfamily as mentioned in the following: 74LS04 indicates a TTL hex inverter in Low-power Schottky technology. 74AL04 indicates Advanced Low-power Schottky TTL technology.

Logic Levels:

There is also a range of voltage that is indeterminate, neither logic-1 nor logic-0. With TTL, logic-1 is represented by +5V and logic-0 by 0V. This, however, is the ideal and, in practice, there is a whole range of voltages that can represent logic-1 and logic-0.

| Logic state | TTL |
|---------------|------------|
| Logic-1 | 2V to 5V |
| Logic-0 | 0V to 0.8V |
| Indeterminate | 0.8V to 2V |

Table 2: Voltage levels for TTL

Voltages present at the inputs and output of the gate must be maintained at the levels shown in table 2 and not be allowed to fall into the intermediate range; otherwise the gate will not give the correct logic function.

Pin arrangements:

The IC pins are arranged in a definite pattern. One end of the top of the IC has a notch or indentation. Starting from the notch, the pin is numbered counter-clockwise. The IC as shown in Fig. 5 is a 14-pin DIP (dual-in-line package).

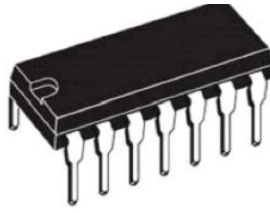


Figure 5: DIP arrangement (14 pin)

IC Identification:

ICs are identified by a number code stamped on the top. The prefix is manufacturer's code. The next two numbers denote the family of ICs such as TTL or CMOS. If letters follow, they indicate the subfamily of the IC. The next numbers indicate the function of the IC, and the last letters indicate the package style.

For example,

74/54 Series Numbering

DM74LS83N

DM

74

LS

83

N

Prefix/ Manufacture's code

Family

Subfamily

Function code

Package

NOT gate (Inverter):

NOT gate is a single-input gate whose output is the complement of the input. It inverts the signal at the input. Figure 6 shows symbols for the inverter. Table 3 summarizes the operation in a truth table by listing all possible inputs and the corresponding output.

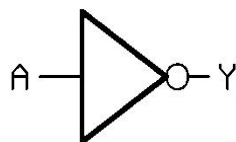


Figure 6: Symbol of NOT gate

| Input | Output |
|-------|---------------|
| A | $Y = \bar{A}$ |
| 0 | 1 |
| 1 | 0 |

Table 3: Truth table of NOT gate

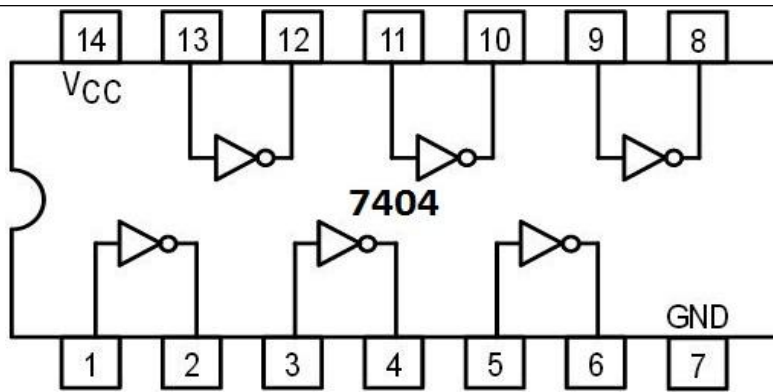


Figure 7: Pin diagram of Hex Inverter (NOT gate IC) 7404



Figure 8: Physical view of IC 7404

AND gate:

The AND gate is a circuit that produces a 1(HIGH) on its output when all of its inputs are 1(HIGH). A two-input AND gate, with inputs A and B and output Y, is shown in Figure 9. The truth table, Table 4, summarizes the operation of the AND gate. All possible input combinations are listed by counting in binary from 00 to 11.

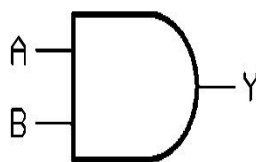


Figure 9: AND gate symbol

| Inputs | | Output |
|--------|---|-------------------|
| A | B | $Y = A \bullet B$ |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

Table 4: Truth table of AND gate

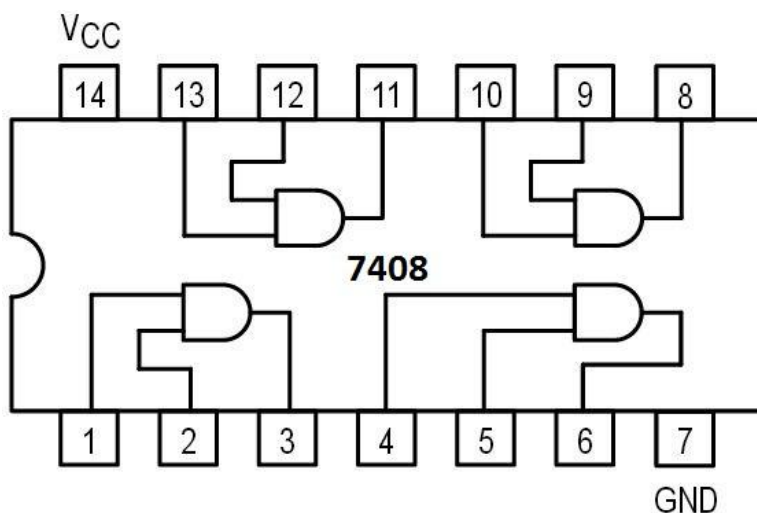


Figure 10: Pin diagram of quad AND gate IC 7408



Figure 11: Physical view of IC 7408

OR gate:

The OR gate is a circuit that produces a 1(HIGH) on its output when any of its inputs are 1(HIGH). Figure 12 shows the symbols for the two-input OR gate with inputs A

and B and output Y. The truth table, Table 5 summarizes the operation of the OR gate. All possible input combinations are listed by counting in binary from 00 to 11.

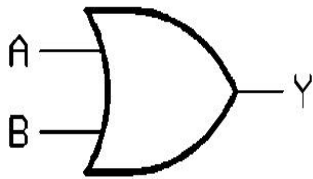


Figure 12: Symbol of OR gate

| Inputs | | Output |
|--------|---|-------------|
| A | B | $Y = A + B$ |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

Table 5: Truth table of AND gate

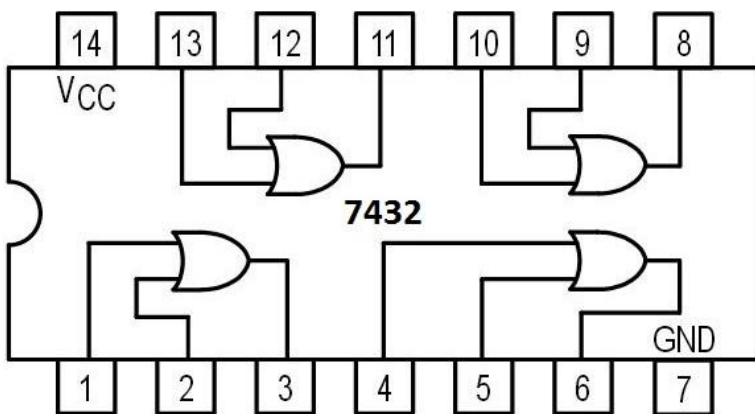


Figure 13: Pin diagram of quad OR gate IC 7432



Figure 14: Physical view of IC7432

NAND gate:

The NAND gate is a circuit that produces a 0 (LOW) at its output when all of its inputs are 1 (HIGH). NAND is the contraction of the words “NOT” and “AND”. Its symbol with two inputs A and B and an output Y, is shown in Figure 15. The truth table for NAND gate is shown in Table 6.

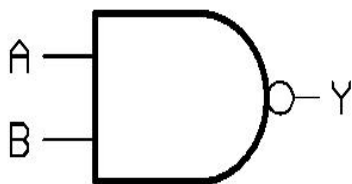


Figure 15: Symbol of NAND gate

| Inputs | | Output |
|--------|---|----------------------------|
| A | B | $Y = \overline{A \cdot B}$ |
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

Table 6: Truth table of NAND gate

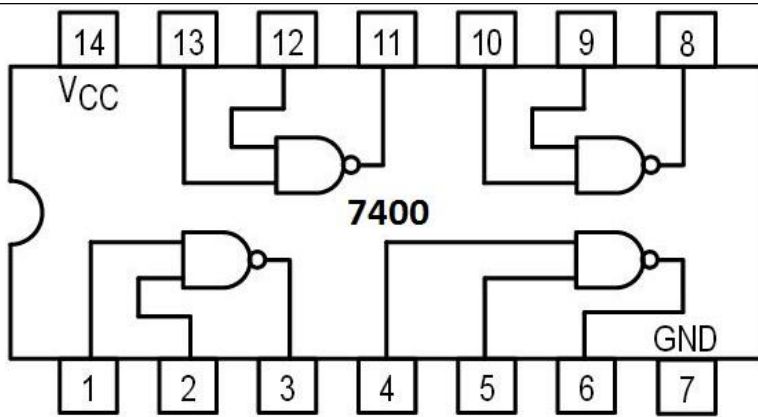


Figure 16: Pin diagram of quad NAND gate IC 7400



Figure 17: Physical view of IC7400

NOR gate:

The NOR gate is a circuit that produces a 0 (LOW) at its output when one or more of its inputs are 1(HIGH). NOR is the contraction of the words “NOT” and “OR”. Its symbol is the OR symbol with an inverted, with two inputs A and B and an output Y, is shown in Figure 18. The truth table for a NOR gate is shown in Table 7.

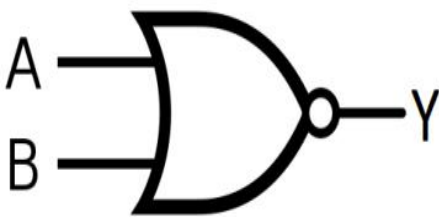


Figure 18: Symbol of NOR gate

| Inputs | | Output |
|--------|---|------------------------|
| A | B | $Y = \overline{A + B}$ |
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

Table 7: Truth table of NOR gate

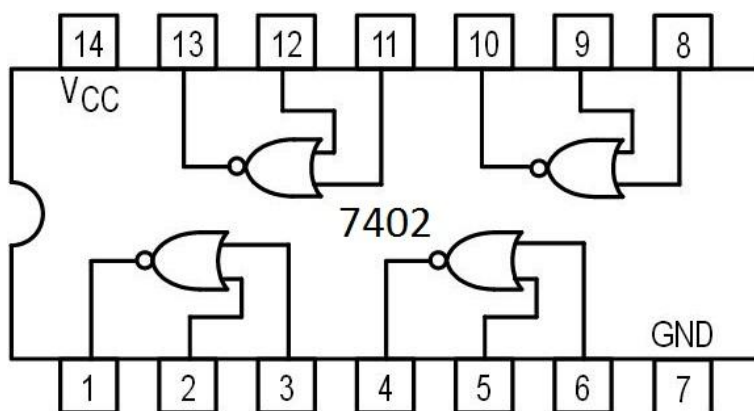


Figure 19: Pin diagram of quad NOR gate IC 7402



Figure 20: Physical view of IC7402

Exclusive-OR gate:

An exclusive-OR (Ex-OR) gate is not one of the basic gates, but is constructed from a combination of the basic gates. The Ex-OR is a two-input gate that produces a 1(HIGH) on its output when its inputs are different and a 0 (LOW) if they are the same. The symbol and truth table for an exclusive-OR are shown in Figure 21 and Table 8.

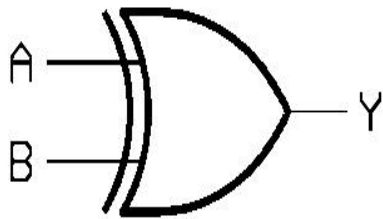


Figure 21: Symbol of XOR gate

| XOR | | |
|-----|---|------------------|
| A | B | $Y = A \oplus B$ |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

Table 8: Truth table of XOR gate

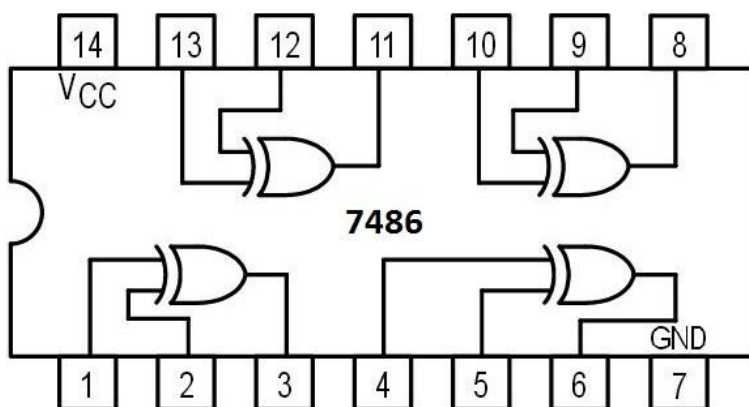


Figure 22: Pin diagram of quad XOR gate IC 7486

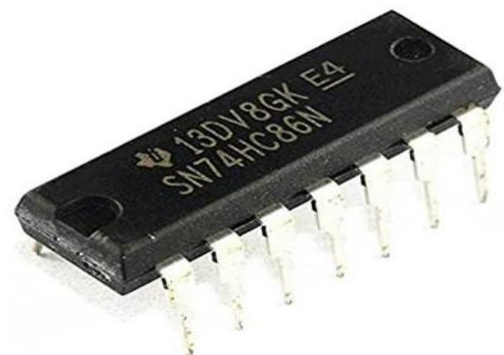


Figure 23: Physical view of IC7486

Common Problems performing the experiment:

1. Not connecting the ground and/or power pins for all chips.
2. Not turning on the power supply before checking the operation of the circuit.
3. Leaving out wires.
4. Plugging wires into the wrong holes.
5. Driving a single gate input with the outputs of two or more gates
6. Modifying the circuit with the power on.

Introduction to Vivado Design Suite Software

Xilinx's Vivado is a comprehensive tool suite used for designing, simulating, and implementing digital systems on FPGA (Field-Programmable Gate Array) devices. Here are some of the key points regarding Xilinx's Vivado FPGA design, simulation, and implementation tool set:

FPGA Design and Implementation: Vivado allows designers to create digital systems using various methods, such as schematic capture or Hardware Description Languages (HDLs) like Verilog or VHDL. These languages enable designers to describe the behavior and structure of digital circuits.

Simulation: The designs created using Vivado can be subjected to simulation to ensure their correctness and functionality. Simulation helps catch errors and bugs before the design is implemented on hardware. The tool provides simulation environments where you can test how your digital circuit behaves under different conditions.

Verification: During simulation, designers can verify whether the behavior of their digital systems matches the intended functionality. This step is crucial to ensure that the design meets the requirements and functions correctly.

Implementation: After simulation and verification, the next step is to implement the design on an FPGA. Implementation involves mapping the logical design onto the physical resources of the FPGA. This includes tasks like placement (deciding where different logic elements will go) and routing (connecting the logic elements).

Pin Assignment: Once the design is implemented, the final step is to assign the inputs and outputs of the design to the physical pins of the FPGA. This step determines how the external world interacts with the digital system.

Vivado simplifies the process of FPGA development by providing a comprehensive set of tools and an integrated development environment that supports the entire design flow, from concept to implementation. It's a powerful tool that enables engineers and designers to create sophisticated digital systems that can be realized on programmable hardware.

Verilog HDL:

Verilog HDL is one of the most used HDLs. It can be used to describe designs at four levels of abstraction:

- Algorithmic level.
- Register transfer level (RTL).
- Gate level.
- Switch level (the switches are MOS transistors inside gates).

Why Verilog ?

- Easy to learn and easy to use, due to its similarity in syntax to that of the C programming language.
- Different levels of abstraction can be mixed in the same design.
- Availability of Verilog HDL libraries for post-logic synthesis simulation.
- Most of the synthesis tools support Verilog HDL.

Verilog HDL Design for Combinational Logic Circuits:

Verilog is a **Hardware Description Language (HDL)**. A hardware description language is a language used to describe a digital system, for example, a network switch, a combinational circuit, a microprocessor or a memory or a simple flip-flop. This just means that, by using a HDL, one can describe any (digital) hardware at any level.

A Verilog input file in the Vivado software environment consists of the following segments:

1. **Header:** module name, list of input and output ports.
2. **Declarations:** input and output ports, registers and wires.
3. **Logic Descriptions:** equations, state machines and logic functions.
4. **End:** endmodule

All the designs for this lab must be specified in the above Verilog input format.

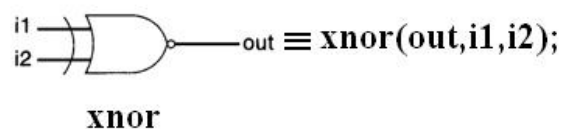
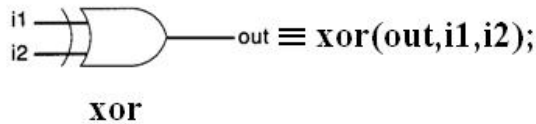
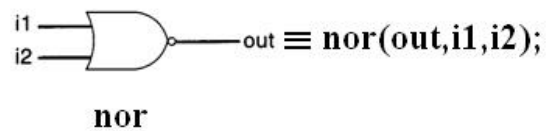
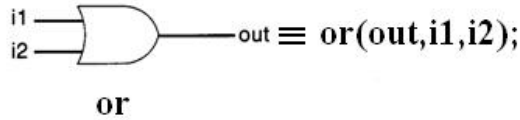
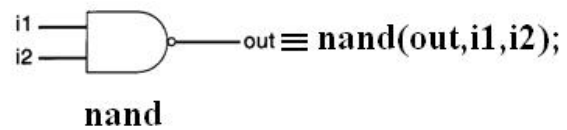
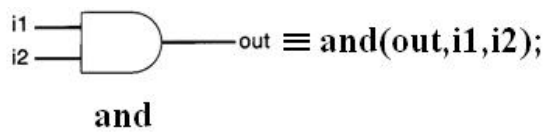
Note: The state diagram segment does not exist for combinational logic designs

Gate level modeling:

In gate level modeling, circuit is described by gates and interconnects. There is one-to-one mapping between the logic circuit diagram and Verilog description.

The module is implemented in terms of logic gates and interconnections between these gates. Design at this level is similar to describing a design in terms of a gate-level logic diagram.

The primitives **not**, and, nand, or, nor, xor, and xnor represent simple logic functions with one or more inputs and one output.



Sample Verilog code :

A verilog code at gate level for a circuit containing **and**, **or** and **not** gate.

Here, module name is sample_circuit. Three input ports are a, b and c. Output ports are x and y. 'e' is declared as a *wire* for internal communication. Primitives are used for corresponding gates.

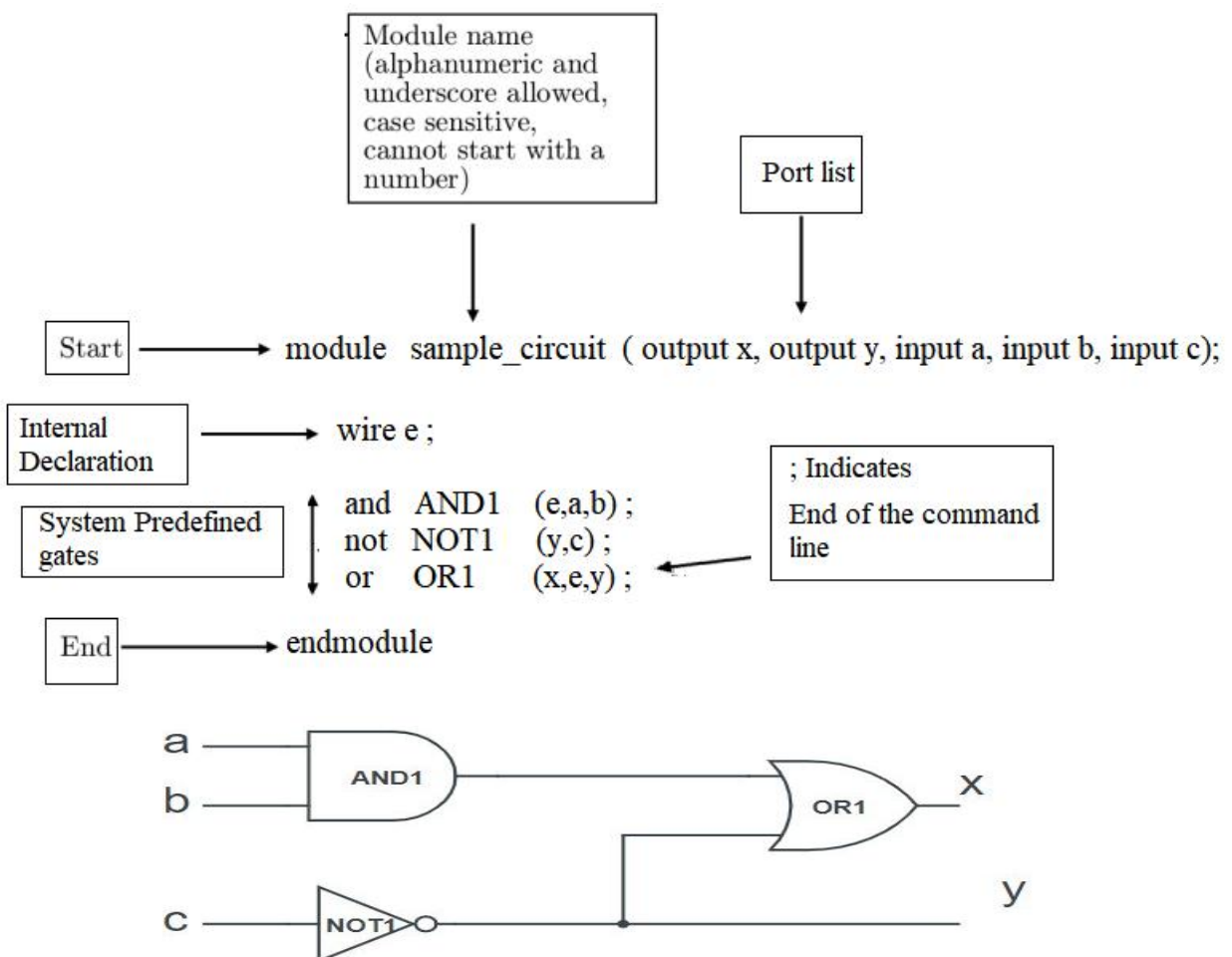


Figure 24: Gate Level Design

Procedure to work with Vivado :

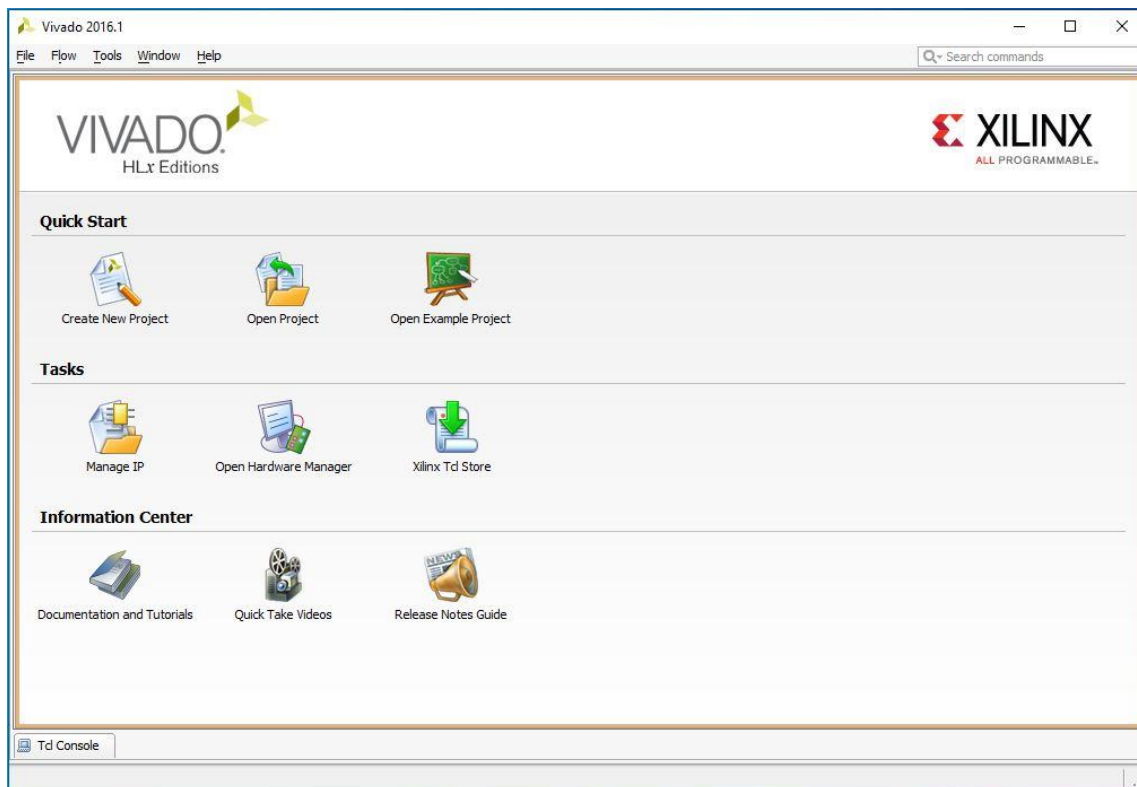
I: Create a new project.

II: Implement the function using Verilog HDL

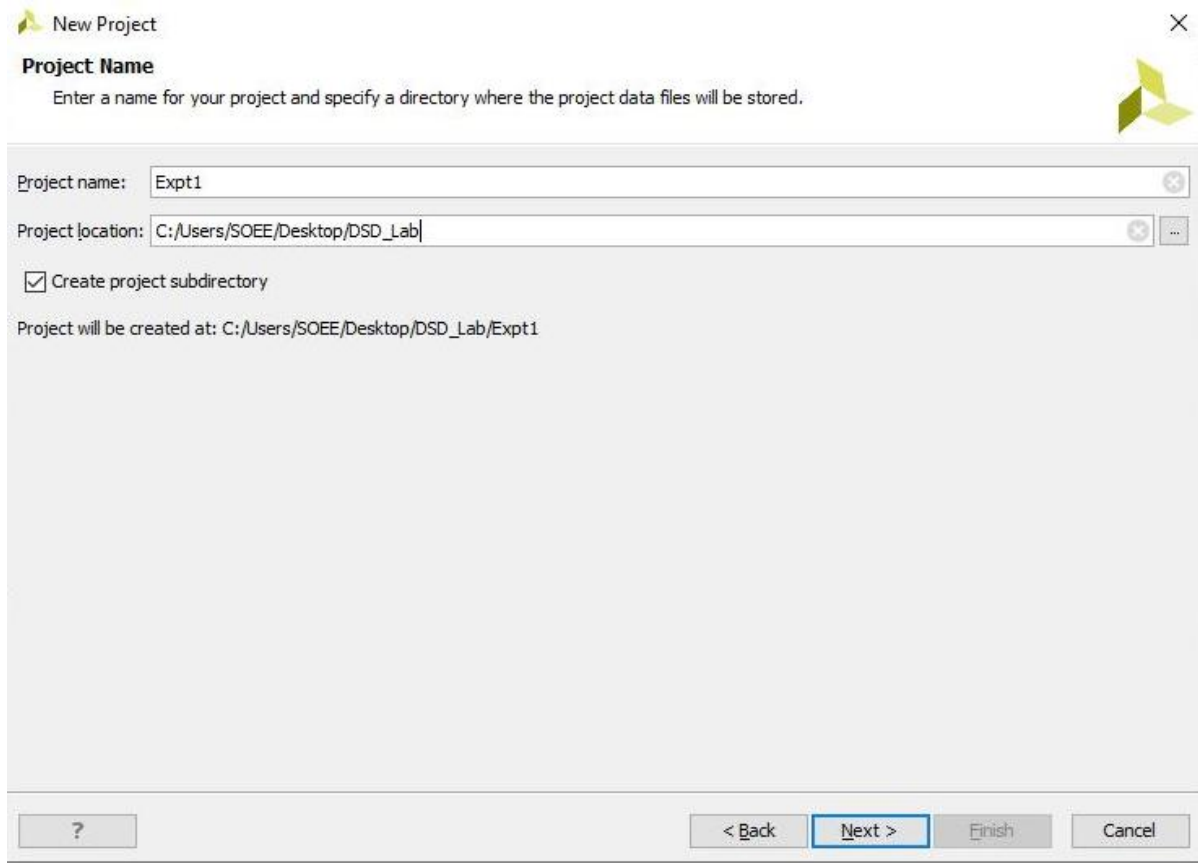
III: Simulate the designed circuit

The above steps for design and simulation of a digital circuit in Vivado is shown with an example of circuit with two outputs x, y and three inputs a,b,c. The boolean expression for outputs are $y = \bar{c}$ and $x = (a \cdot b) + \bar{c}$.

1. Start by creating a folder on the desktop called “DSD_Lab”.
2. Double click on the Vivado icon on your desktop to open up the welcome window of the development tool (as shown below). Three main sections can be observed in this window: “Quick Start”, “Tasks”, and “Learning Center”.



3. Now, click on “Create Project” to create a new project. One has to be careful about where to save your project file in the computer lab. Then, in Xilinx, create a new project inside “DSD_Lab”. Name your project, ‘Expt1’ and it will be in the folder “C:/Users/SOEE/Desktop/DSD_Lab”



New Project

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

Project name:

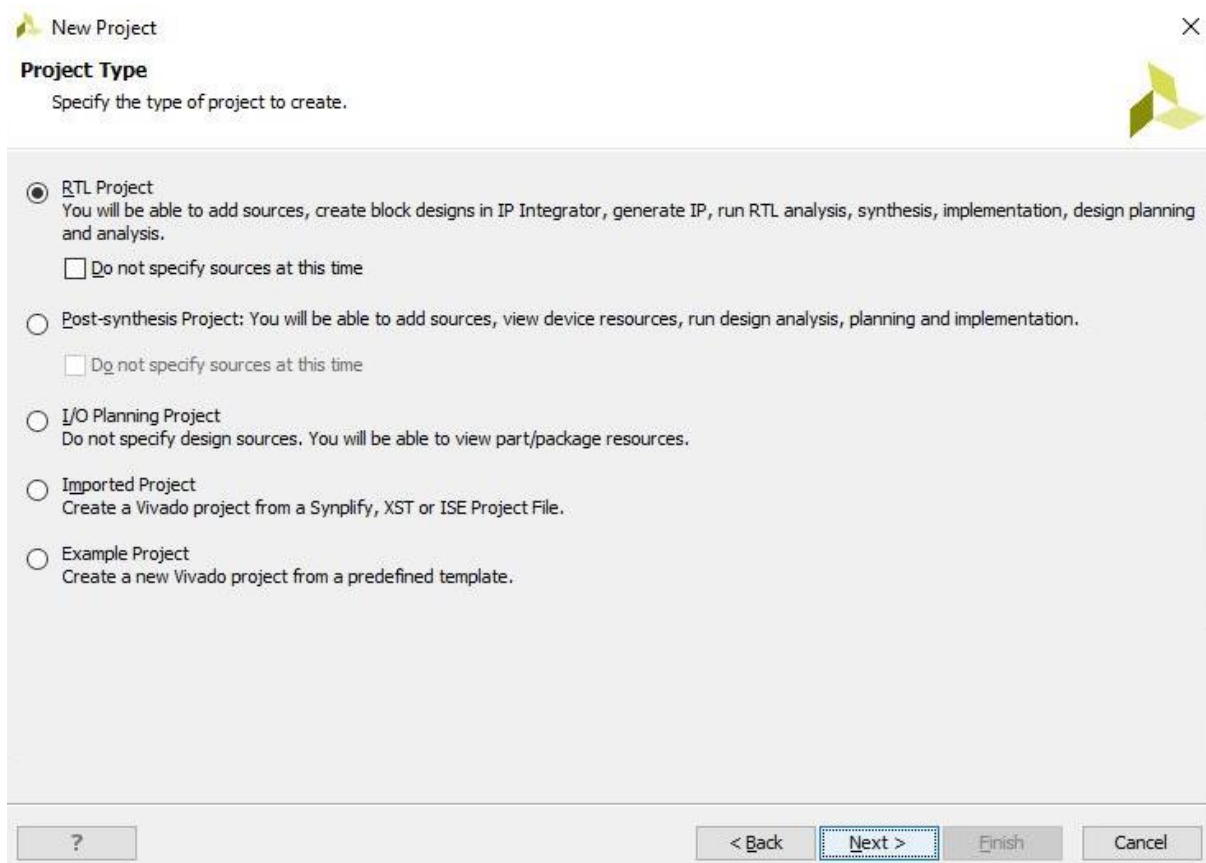
Project location:

☒ Create project subdirectory

Project will be created at: C:/Users/SOEE/Desktop/DSD_Lab/Expt1

? < Back Next > Finish Cancel

4. In the next window, choose “RTL Project” as the project type.



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

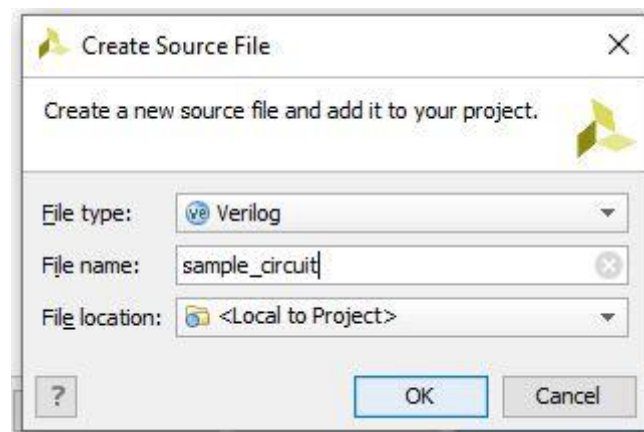
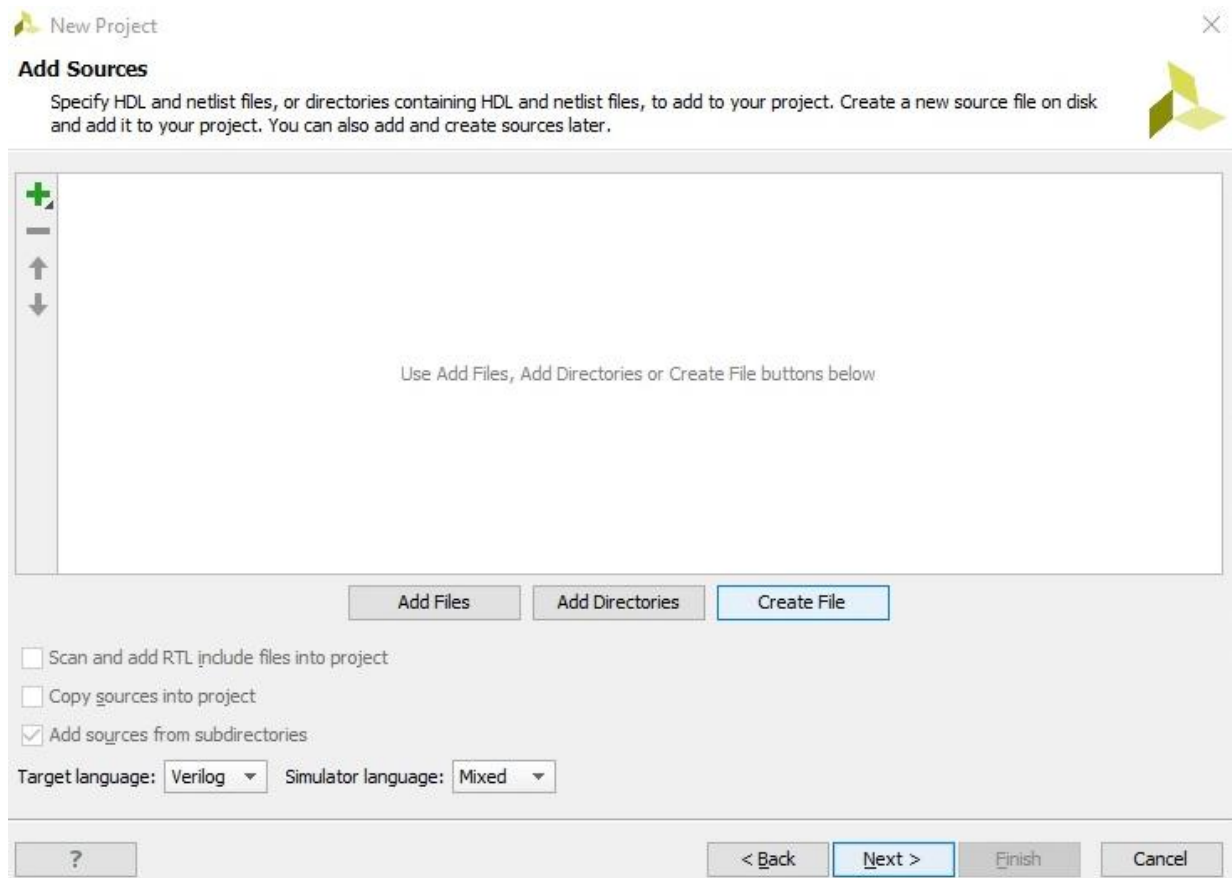
☐ **I/O Planning Project**
Do not specify design sources. You will be able to view part/package resources.

☐ **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

5. In the opened window, create source file (Verilog File) for our new project or can add sources from the existing projects. Click on “Create File”, and in the opened window choose “Verilog” for the “File type”, write a name for file (“Part_1”), and click on “Ok”. Continue clicking on Next until reaching the “Default Part” window.



Add Sources

Specify HDL and netlist files, or directories containing HDL and netlist files, to add to your project. Create a new source file on disk and add it to your project. You can also add and create sources later.



| | Index | Name | Location | Library | HDL Source For | Location |
|---|-------|------------------|--------------------|----------------|------------------------|--------------------|
| + | 1 | sample_circuit.v | <Local to Project> | xil_defaultlib | Synthesis & Simulation | <Local to Project> |

↑
↓

☐ Scan and add RTL include files into project
☐ Copy sources into project
☒ Add sources from subdirectories

Target language: Verilog Simulator language: Mixed

Add Existing IP (optional)

Specify existing configurable IP, DSP composite, and Embedded sub-design files to add to your project.



+

Use Add Files or Add Directories buttons below

☐ Copy sources into project

New Project

Add Constraints (optional)

Specify or create constraint files for physical and timing constraints.

Use Add Files or Create File buttons below

☐ Copy constraints files into project

6. In this window, choose “Artix-7” for the “Family”, “-1” for “Speed grade”, and “cpg236” for “Package”. In the shown parts, select “xc7a50tcpg236-1”.

New Project

Default Part

Choose a default Xilinx part or board for your project. This can be changed later.

Select: ☒ Parts ☐ Boards

Filter

Product category: All Speed grade: -1

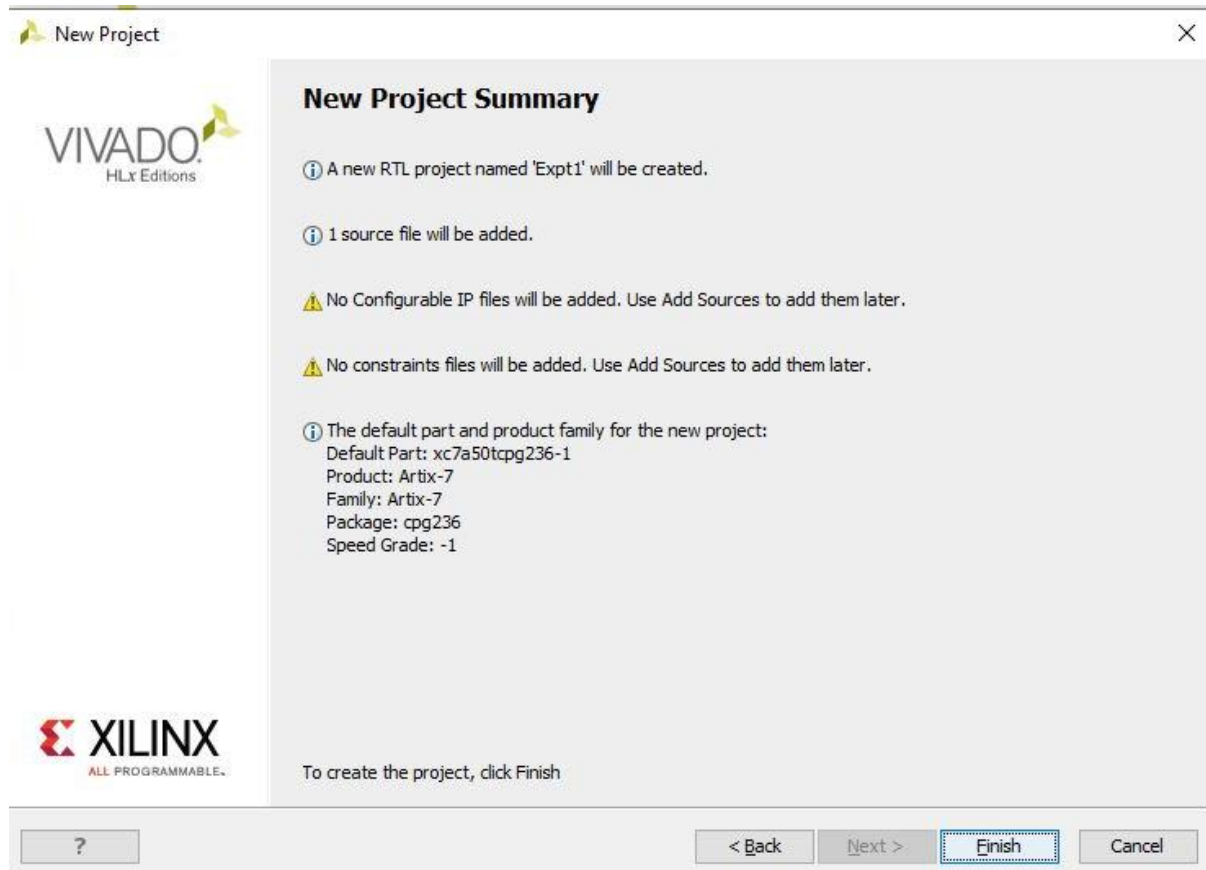
Family: Artix-7 Temp grade: All Remaining

Package: cpg236

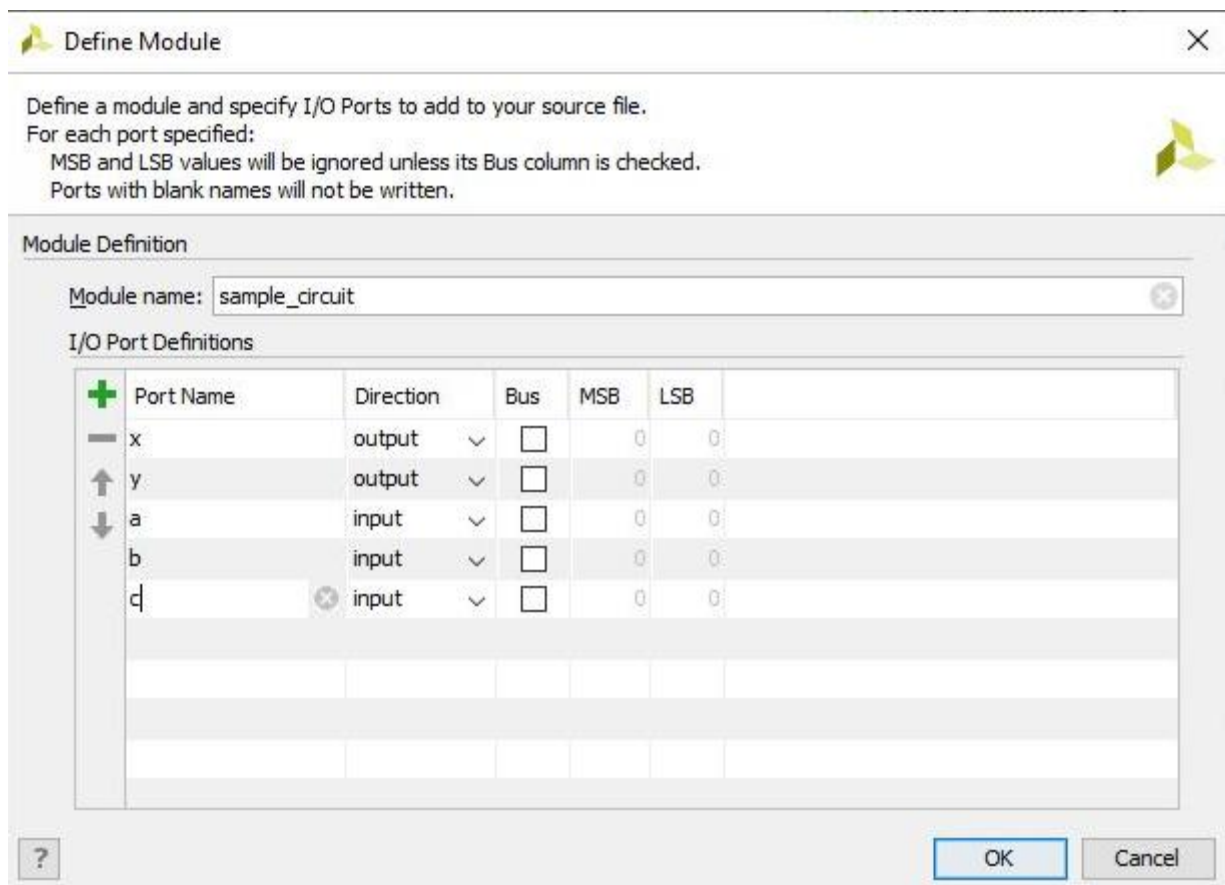
Search: Q

| Part | I/O Pin Count | Block RAMs | DSPs | FlipFlops | GTPE2 Transceivers | Gb Transceivers | Available IOBs | LUT Elements |
|-----------------|---------------|------------|------|-----------|--------------------|-----------------|----------------|--------------|
| xc7a15tcpg236-1 | 236 | 25 | 45 | 20800 | 2 | 2 | 106 | 10400 |
| xc7a35tcpg236-1 | 236 | 50 | 90 | 41600 | 2 | 2 | 106 | 20800 |
| xc7a50tcpg236-1 | 236 | 75 | 120 | 65200 | 2 | 2 | 106 | 32600 |

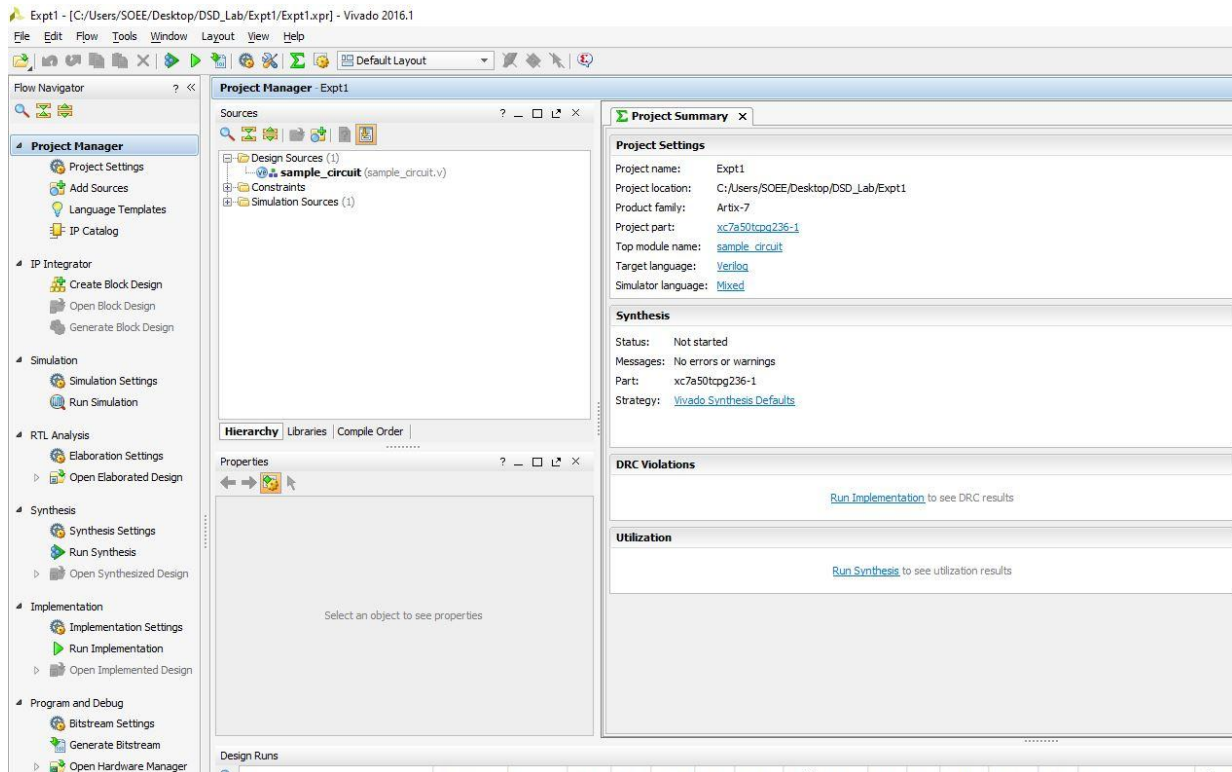
7. Look at your new project summary.



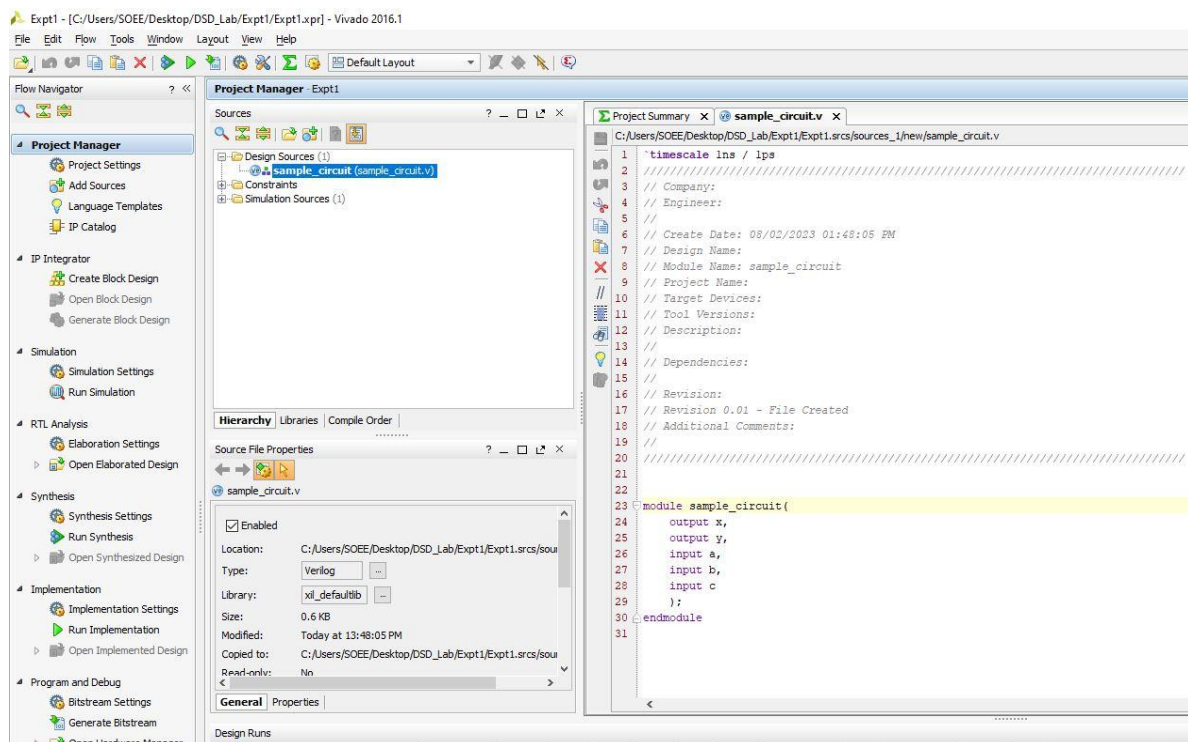
8. Define the input and the output ports of the module according to the shown window.



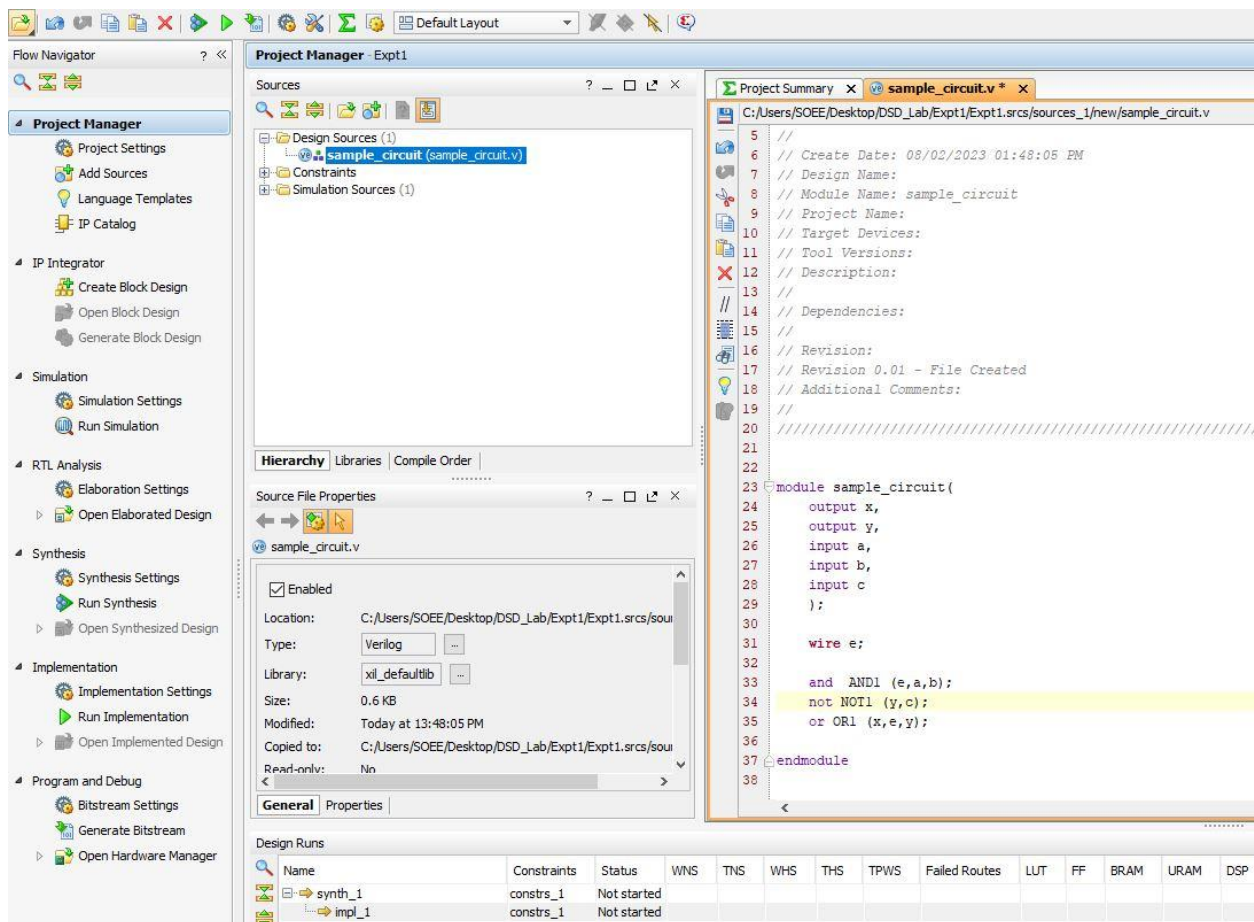
9. The opened window is the main environment for your project that is called “Project Manager”. In the left side, you can see the “Settings”, “Add Sources”, “Language Template”, “IP Catalog”, “IP Integrator”, “Simulation”, “RTL Analysis”, “Synthesis”, “Implementation”, and “Program and Debug”. Each of these serves a part of the digital design flow. In the middle, you can see the windows for “Sources”, “Properties”, “Project Summary”, and the reports and summaries for the execution of the project files.



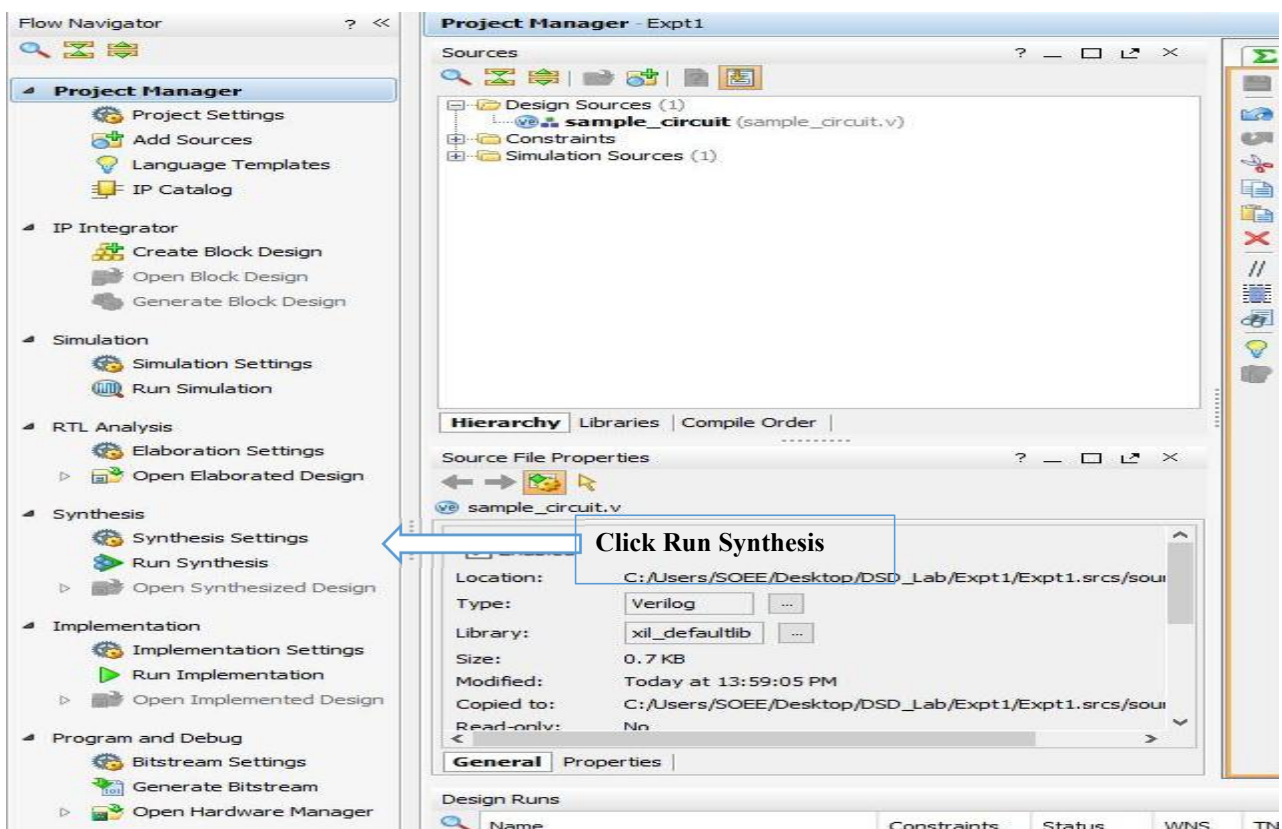
10. Double click on the “sample_circuit.v” file (*.v) in the “Sources” window. The VERILOG source file appears where the window is located in right side. Note that the module shows the defined inputs and outputs that were selected previously.



11. Write the code required to implement the digital circuit.

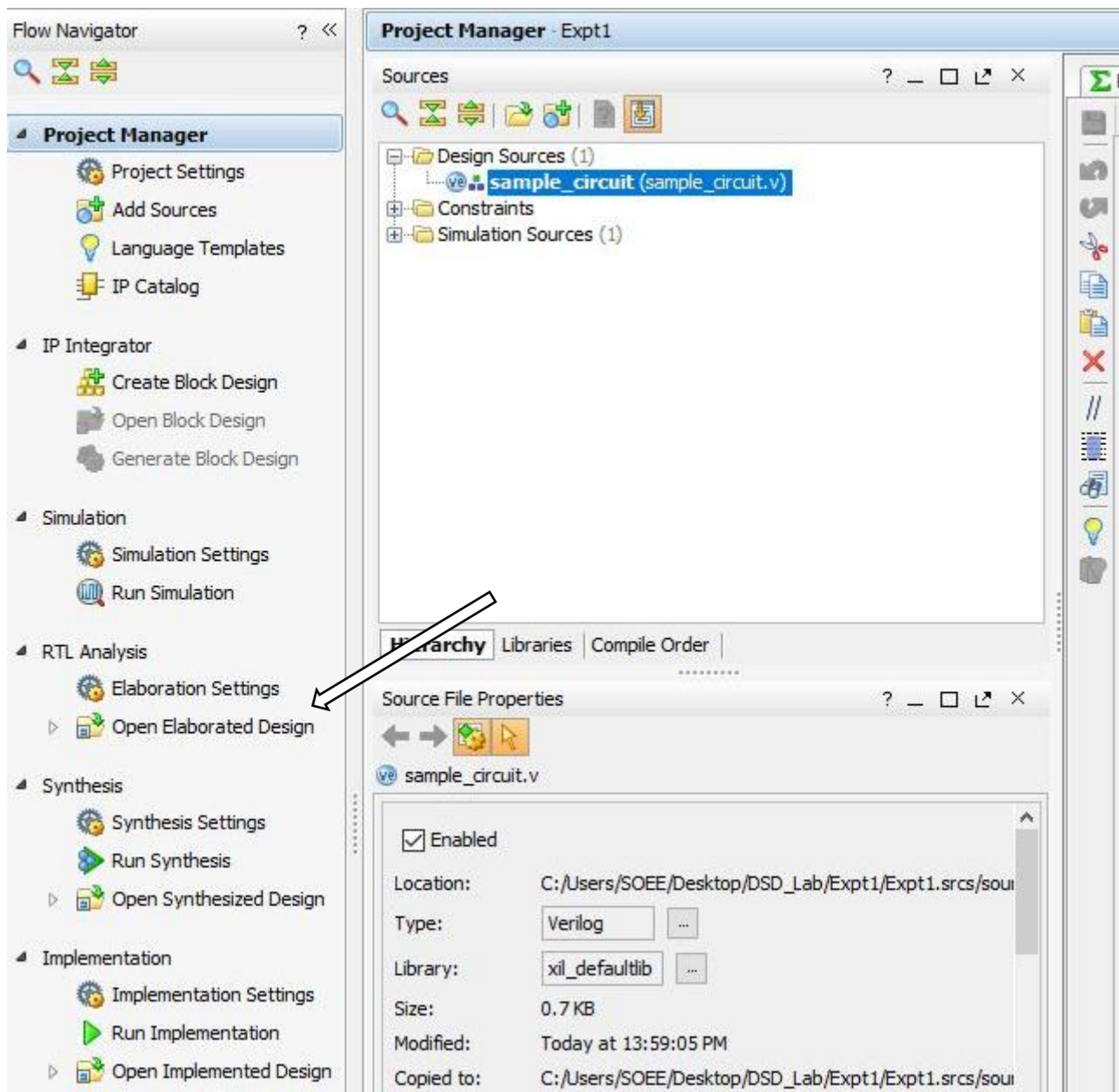


12. Now that the design is finished, next is to build the project. Click **Run Synthesis** on the left hand menu , on successful completion click on Run implementation. Proceed to the next steps.”

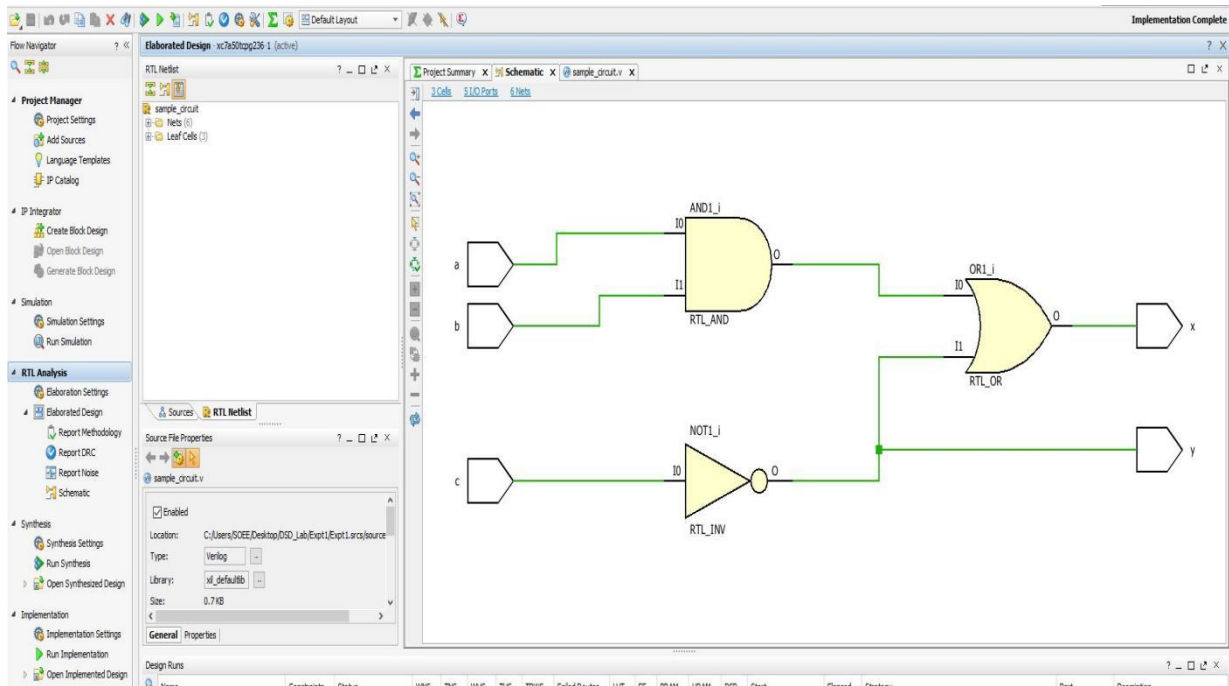




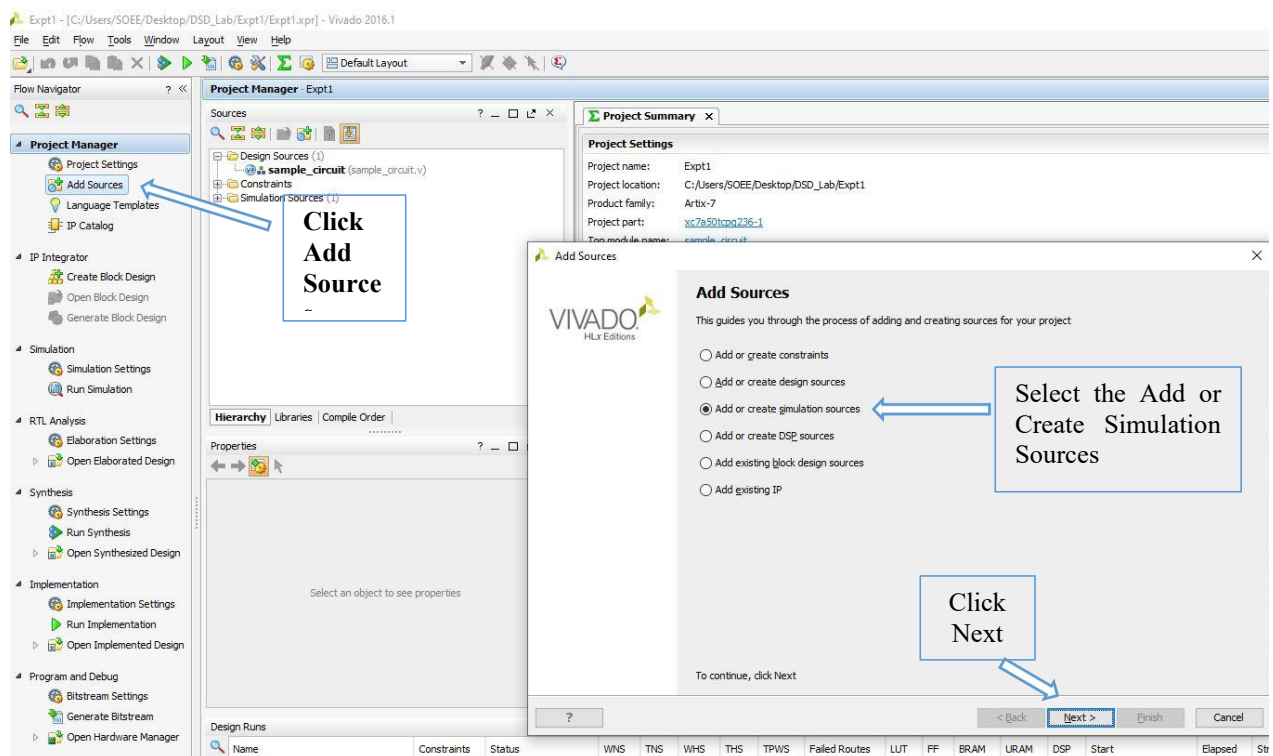
13. Do the “**RTL Analysis**”. Expand the Open Elaborated Design entry under the RTL Analysis tasks of the Flow Navigator pane and click on **Schematic**.

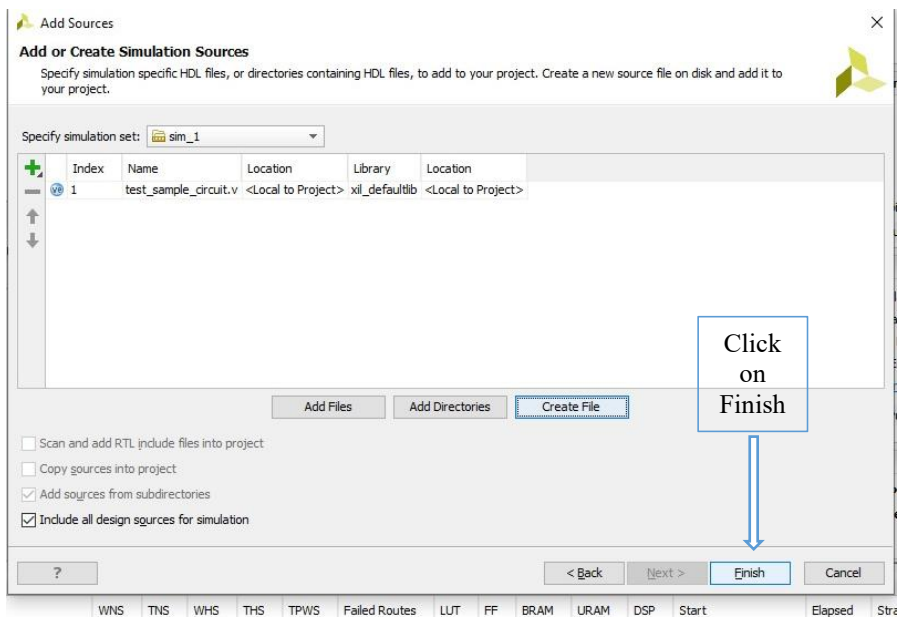
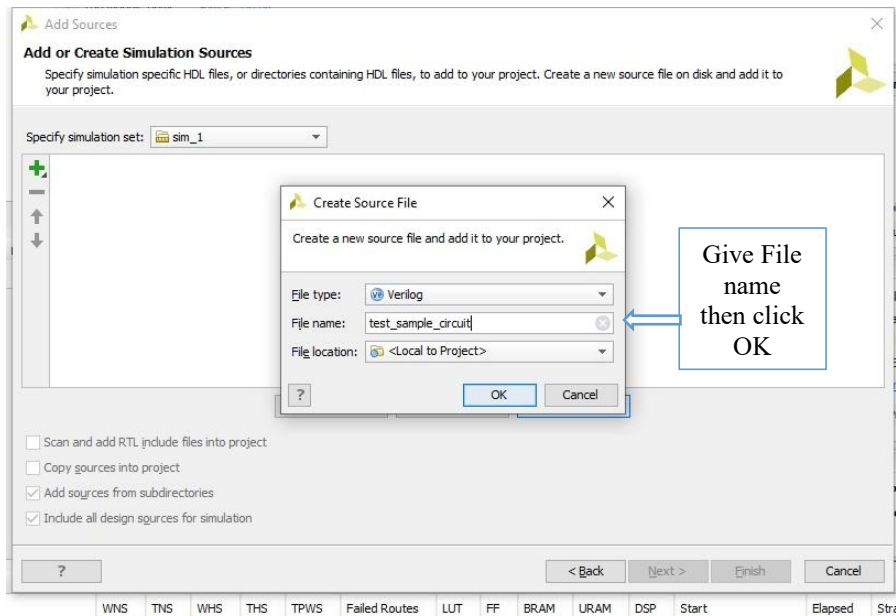
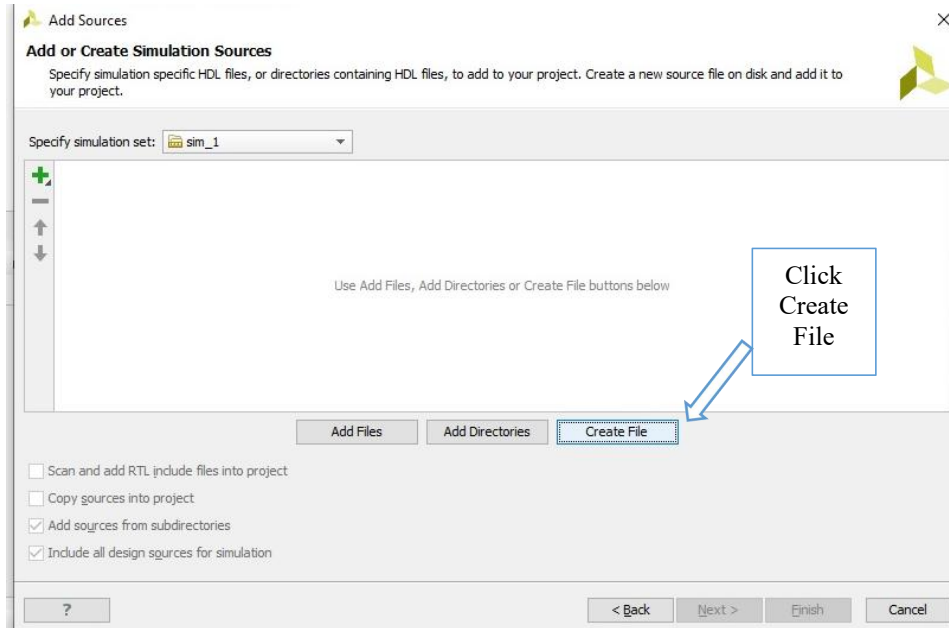


14. The model (design) will be elaborated and a logic view of the design is displayed.

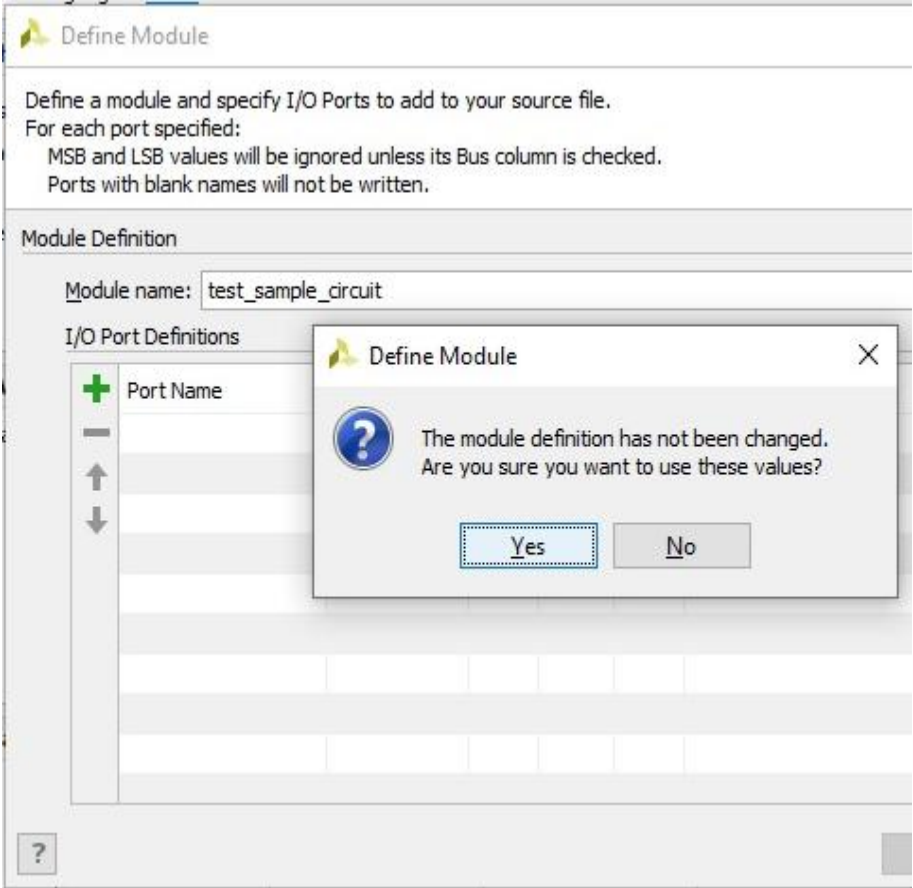
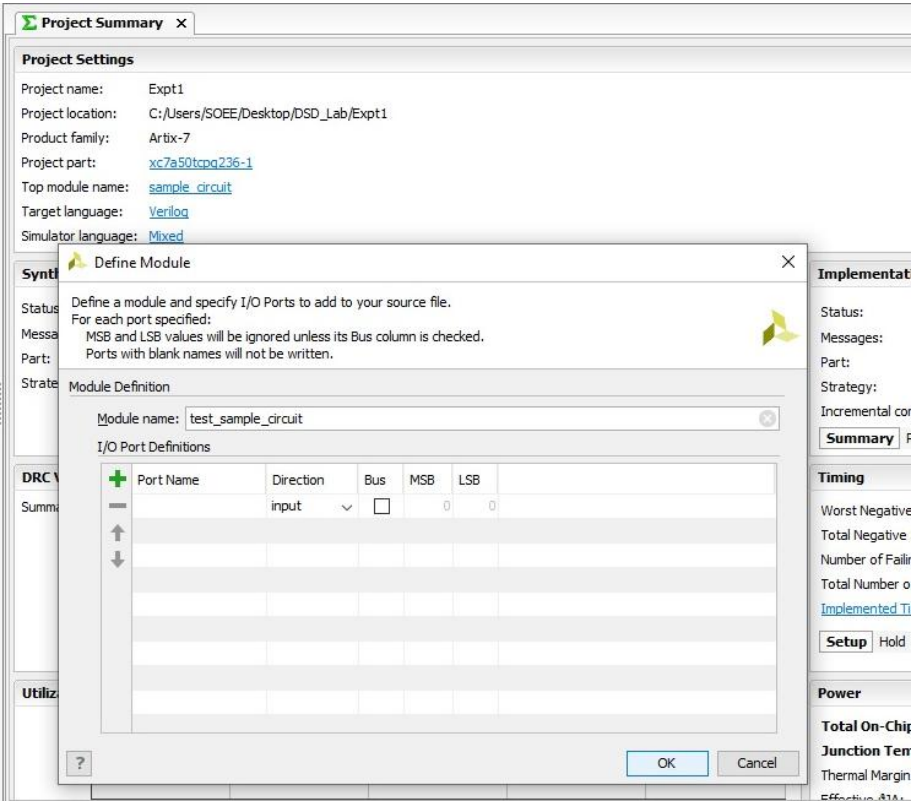


15. Then to simulate the design using the Simulator. Click **Add Sources** under the Project Manager tasks of the Flow Navigator pane. Select the Add or Create Simulation Sources option and click Next. We create a new file for simulation, and name it as “test_sample_circuit”

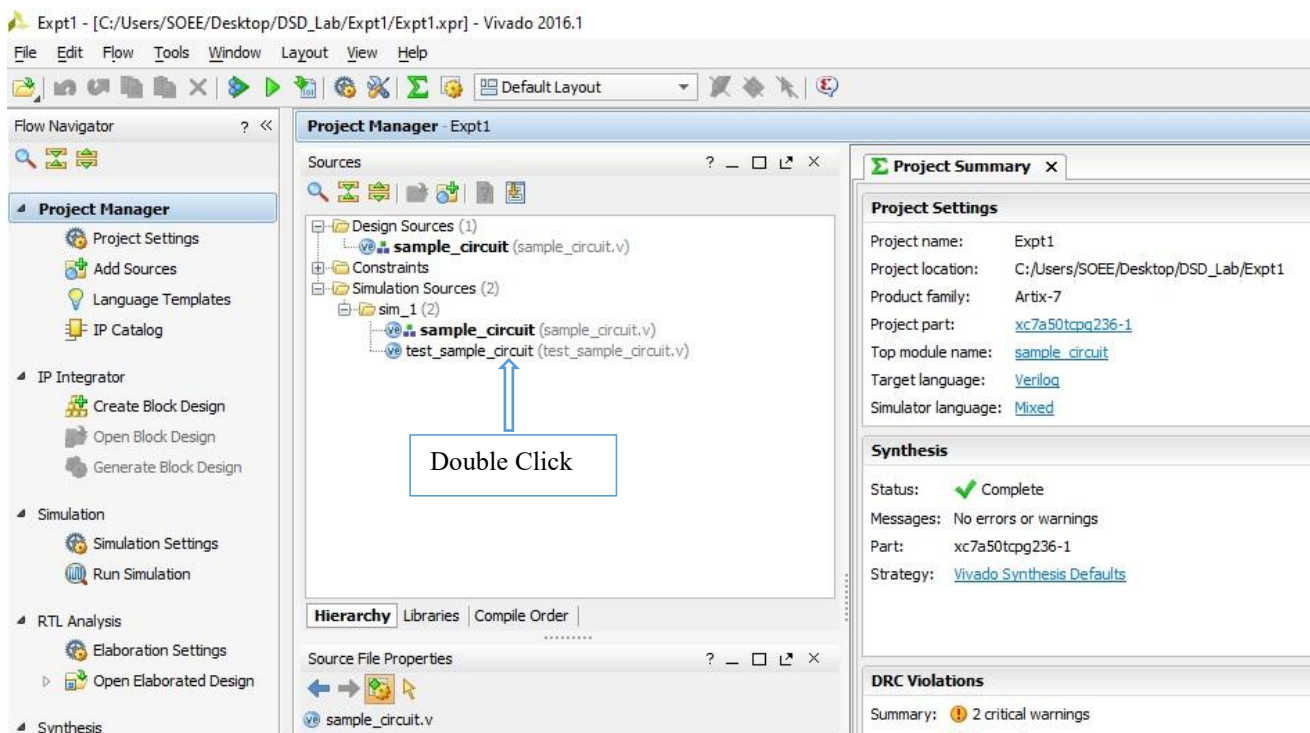




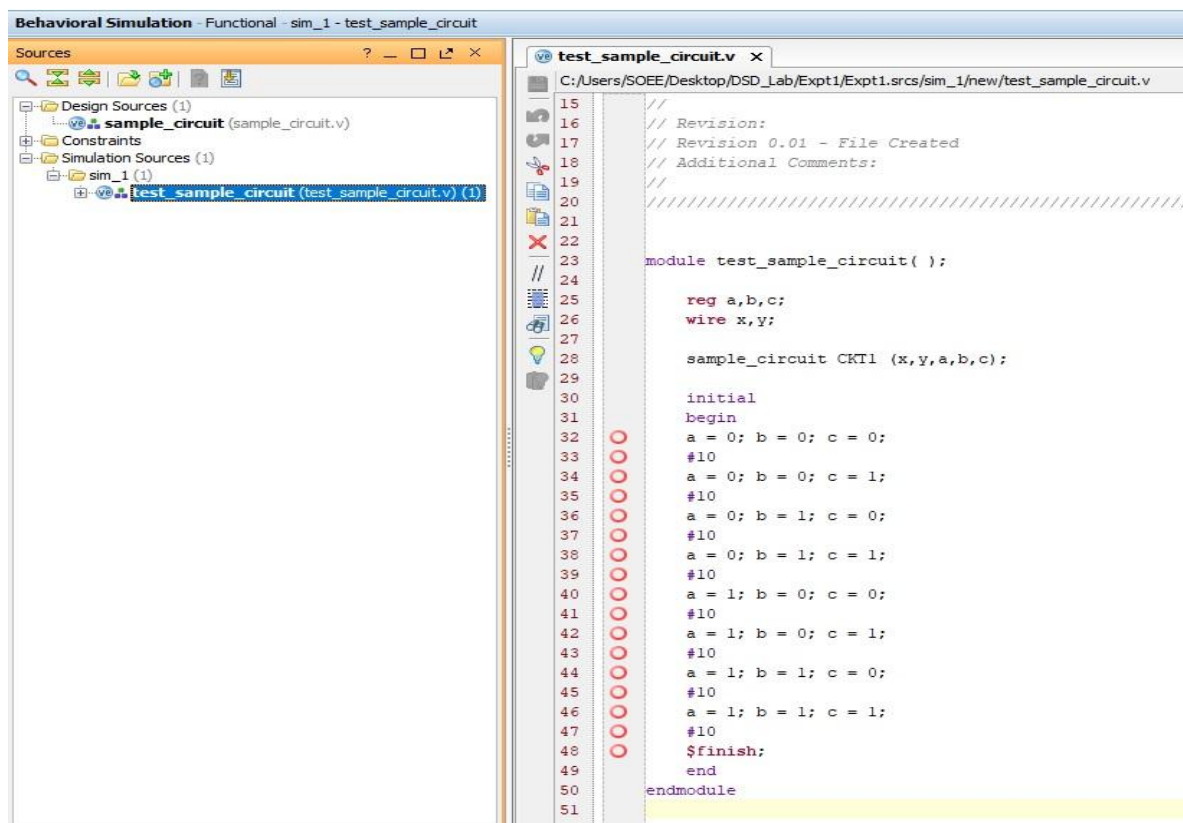
16. For the simulation file, we don't need to set the I/O. Click OK in this step.



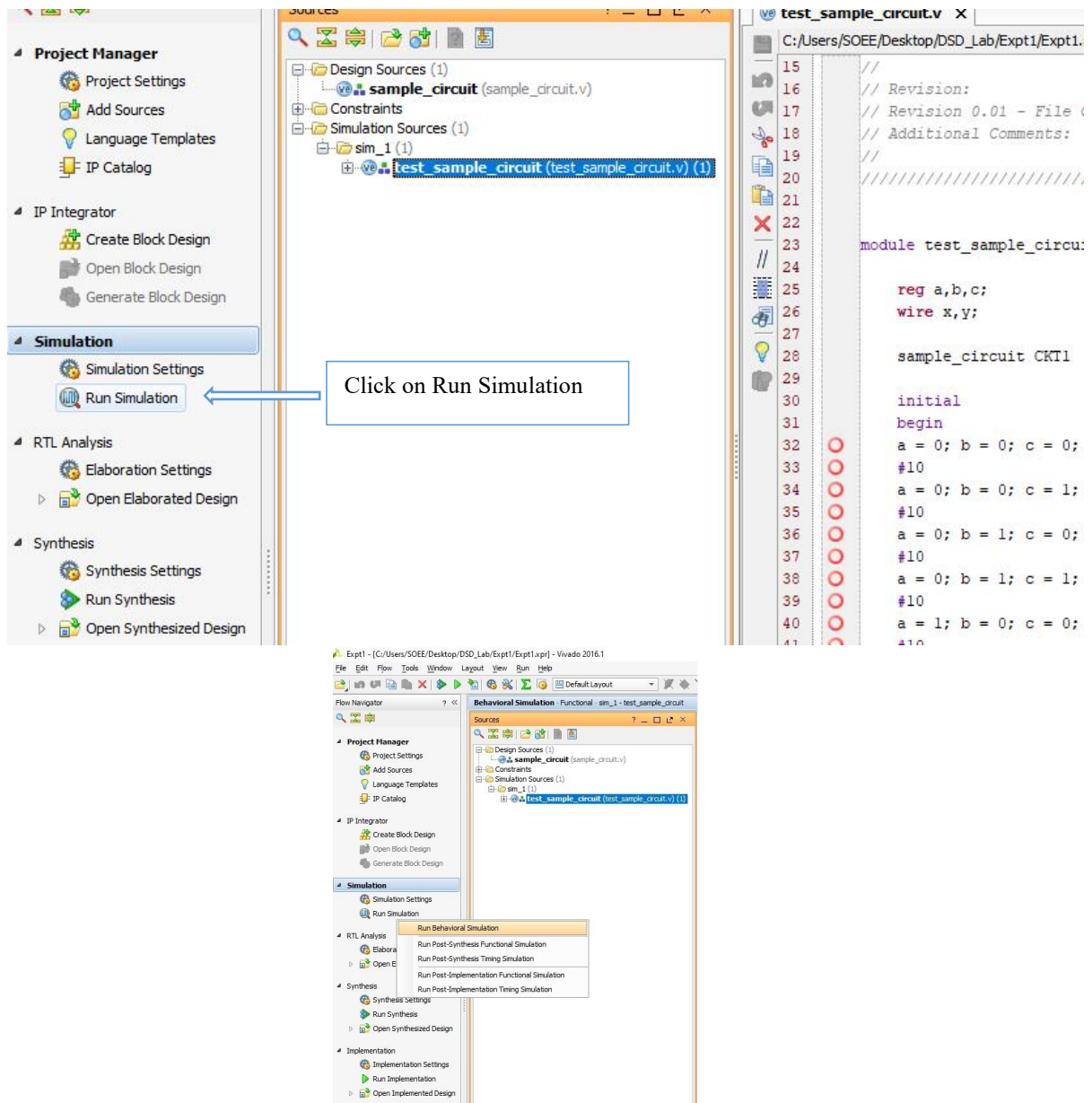
17. Double click on the “test_sample_circuit.v” in the Sources window to type the simulation contents into the file.



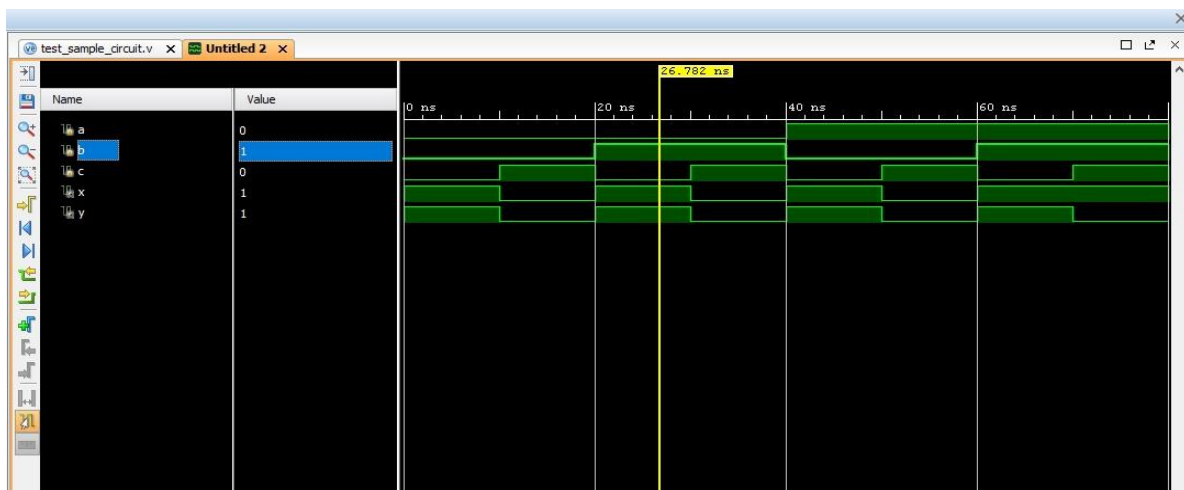
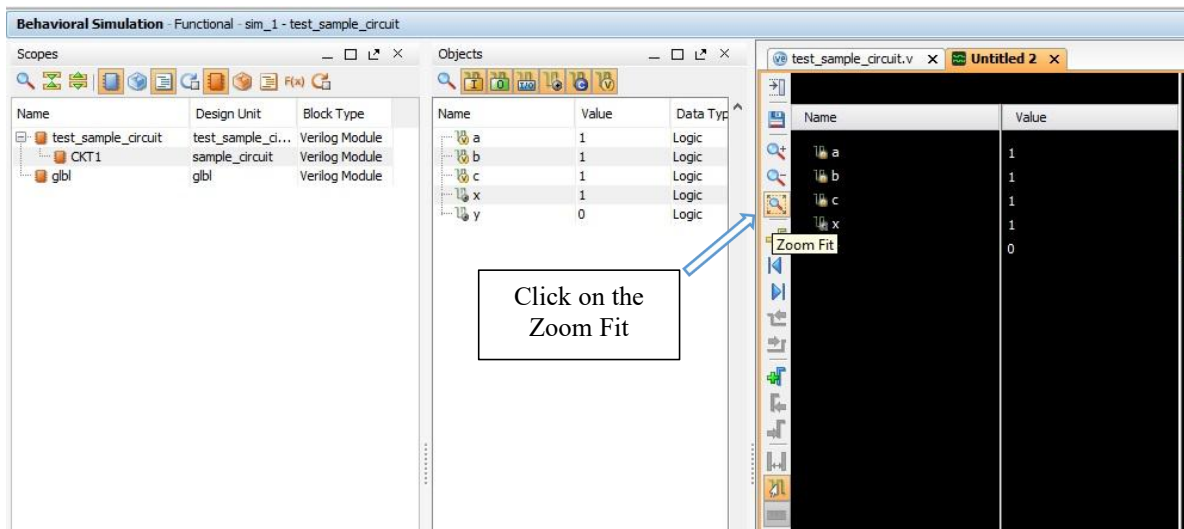
18. Within the Test-bench file, the simulation step size and the resolution can be specified in line 1. The Test-bench module definition begins on line 23. Line 30 instantiates the UUT (unit/module under test). Also, it should be mentioned that the timescale for both the Verilog module and the test-bench module is set to 1 ns/1 ps.



19. Click on Run Simulation > Run Behavioral Simulation under the Project Manager tasks of the Flow Navigator window.

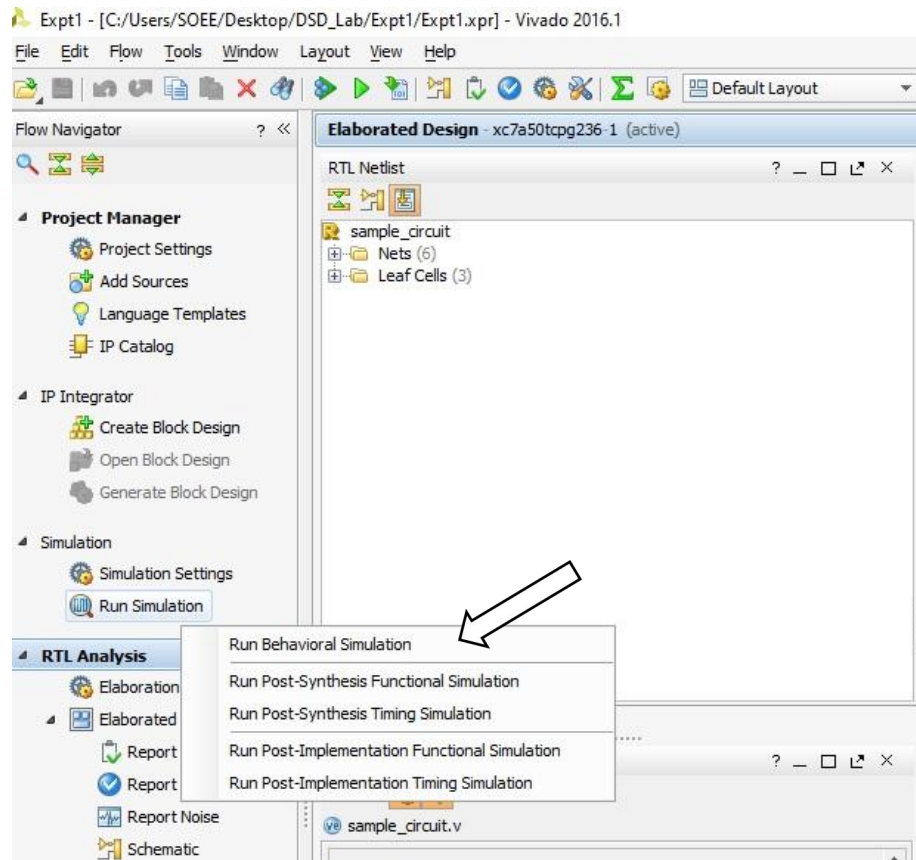


20. The test-bench and source files are compiled and the simulator is run (assuming no errors). Click on the “Zoom Fit” icon to see all the spectrum of simulation.

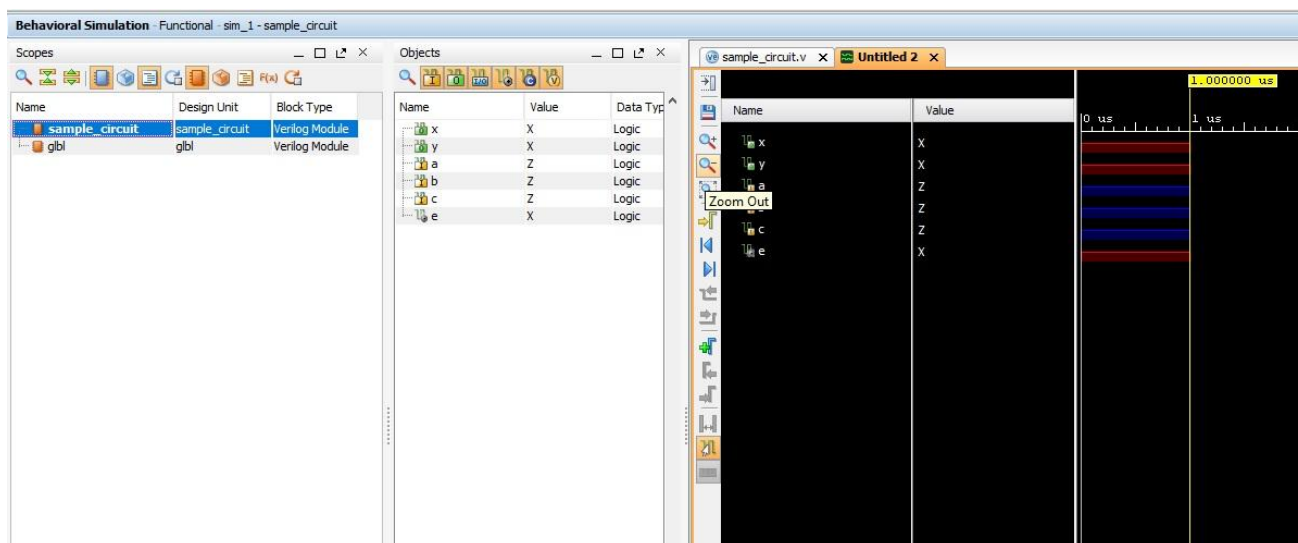


21. To simulate the design could be done by using force clock also .

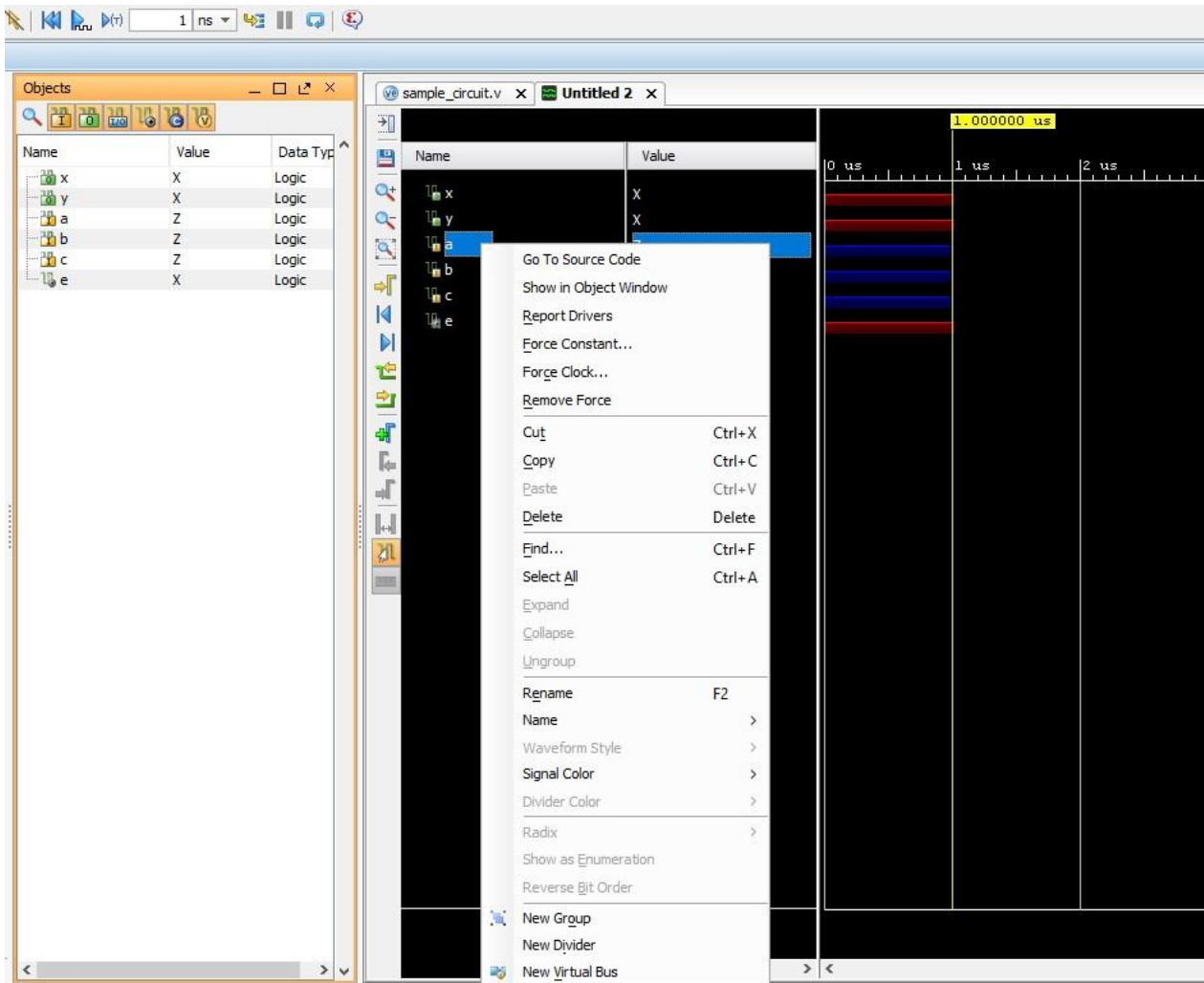
Click on **Run Simulation** > **Run Behavioral Simulation** under the Project Manager tasks of the Flow Navigator window.



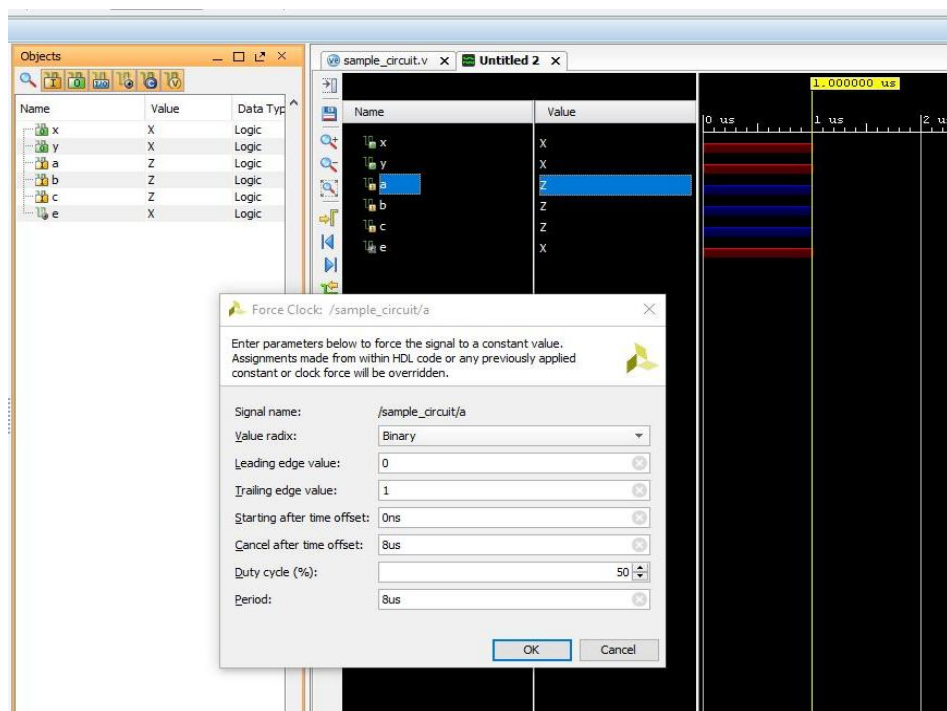
Click on the “Zoom Out” icon to see all the spectrum of simulation.

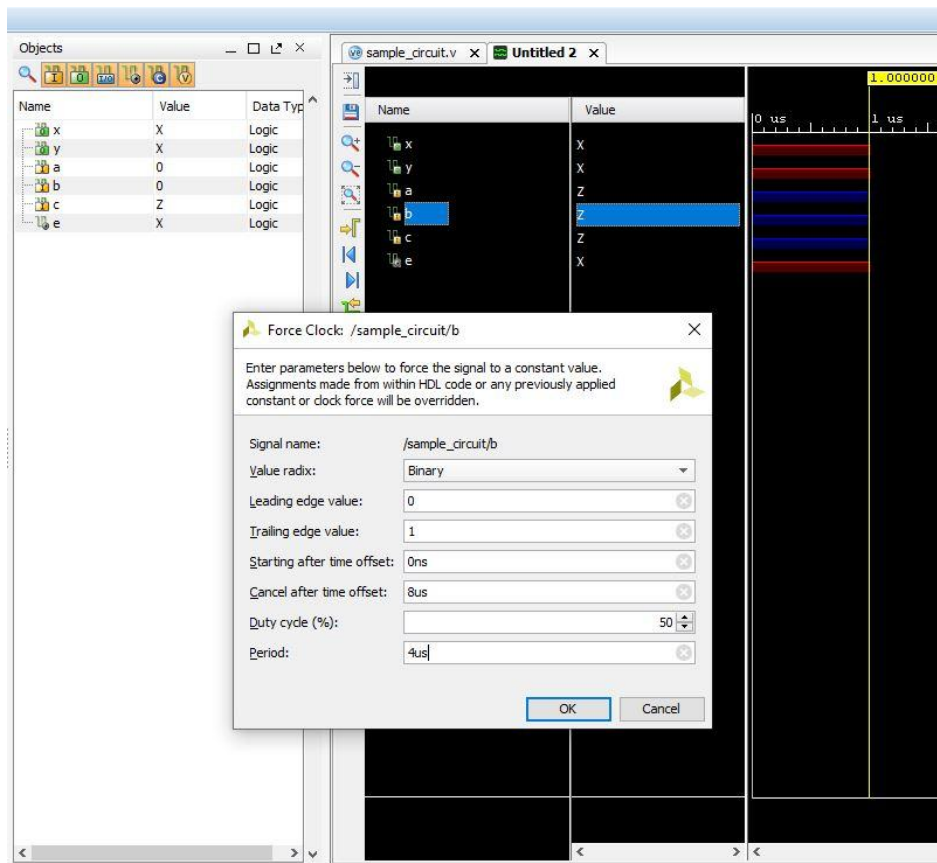


Right click on the input > select Force Clock

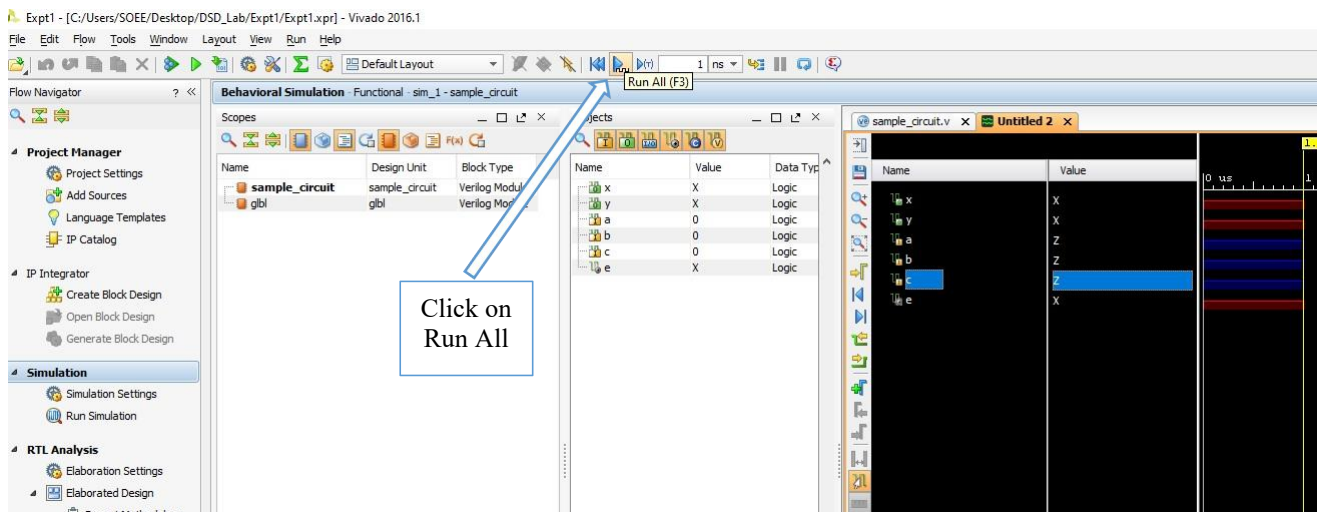


Assign the required values in the Force Clock window for all the inputs in the design.





Click on Run All



Verify that the circuit is working correctly by checking the time diagram with the truth table of the circuit.

