PRACTICAL MANUAL

VLSI

Submitted By

RIZWAN YOUSUF DAR (ECE-19-27)

ELECTRONICS AND COMMUNICATION ENGINEERING
ISLAMIC UNIVERSITY OF SCIENCE & TECHNOLOGY
Awantipora, Pulwama – 192122

TABLE OF CONTENTS

INTRODUCTION

VLSI (Very Large-Scale Integration): is a field of electronics engineering that deals with the design and fabrication of integrated circuits (ICs) that contain millions or even billions of transistors on a single chip. VLSI technology allows engineers to pack a large number of electronic components onto a small chip, resulting in smaller, faster, and more powerful devices.

The importance of VLSI in the electronics field cannot be overstated. VLSI has enabled the development of microprocessors, memory chips, and other advanced electronic components that have revolutionized the electronics industry. VLSI has also made it possible to integrate various functions onto a single chip, resulting in smaller, faster, and more power-efficient devices.

Some of the key benefits of VLSI technology include:

1. Miniaturization: VLSI technology allows engineers to pack a large number of electronic components onto a small chip, resulting in smaller devices that are more portable and easier to use.
2. Faster speeds: With more transistors on a single chip, VLSI technology allows for faster processing speeds, which is essential in applications such as high-performance computing and telecommunications.
3. Lower power consumption: VLSI technology allows for more power-efficient devices, which is important in applications such as mobile computing and battery-powered devices.
4. Lower costs: VLSI technology allows for the integration of multiple functions onto a single chip, reducing the cost of production and making electronics more affordable for consumers.

In summary, VLSI technology has played a critical role in the development of advanced electronic devices, and its importance in the electronics field will only continue to grow in the years to come.

VHDL (VHSIC Hardware Description Language): is a language used for the design and verification of digital circuits and systems. VHDL provides a standard syntax and set of constructs for describing the behavior and structure of digital circuits and systems. It allows designers to specify the functionality of a digital system in a textual form, which can then be synthesized into a hardware implementation.

The general structure of VHDL code consists of four main sections:

1. Entity Declaration: The entity declaration defines the input and output ports of the circuit, along with any additional information that describes the behavior of the entity. It describes the external interface of the entity.
2. Architecture Body: The architecture body contains the implementation details of the entity. It describes how the entity functions internally.
3. Configuration Declaration: The configuration declaration is optional and specifies how different entities can be combined to form a larger system.
4. Testbench: The testbench is a simulation environment that allows the designer to verify the correctness of the VHDL code. The testbench typically contains stimulus generators, which generate input signals for the design, and output monitors, which check the output signals.

Here is an example of a VHDL code structure for a simple circuit:

```
entity and_gate is
  port (input1, input2: in std_logic;
      output: out std_logic);
end and_gate;
architecture dataflow of and_gate is
begin
  output <= input1 and input2;
end dataflow;
```

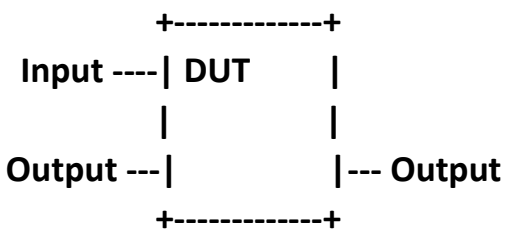
In this example, the entity declaration defines an "and_gate" circuit with two input ports (input1 and input2) and one output port (output). The architecture body defines the implementation of the "and_gate" circuit, which is a logical AND gate. The output port is assigned the value of input1 AND input2. This VHDL code

can be synthesized to produce a digital circuit that performs the logical AND operation.

Key concepts related to digital circuit design:

1. **Design Under Test (DUT):** The DUT is the digital circuit that is being designed or tested. It can be any digital circuit, such as a counter, a decoder, or a controller.
2. **Entity:** In VHDL, an entity is a description of the interface of the DUT. It describes the input and output ports of the circuit, as well as any internal signals. An entity can be thought of as a "black box" that defines the interface of the DUT.
3. **Port:** A port is a connection point between the DUT and the outside world. It can be either an input or an output, and it is defined in the entity declaration. The type of each port can be either a scalar (single bit) or a vector (multiple bits).
4. **Behavior:** The behavior of the DUT is defined in the architecture of the VHDL code. It describes the functionality of the circuit using a combination of Boolean equations and/or conditional statements. The behavior section of the VHDL code can be viewed as the "white box" that defines the internal functionality of the DUT.

Here is a simple block diagram that illustrates these concepts:



In this diagram, the DUT is represented as a black box with input and output ports. The input port is connected to the input signal, and the output port is connected to the output signal. The behavior of the DUT is defined in the VHDL code, which is written in the architecture section of the code. The entity declaration defines the interface of the DUT, including the input and output ports.

Some key concepts in VHDL:

1. **Data Types:** are used to define the characteristics of signals and variables used in the design. VHDL supports a range of data types, including Boolean, integer, real, and character types. These data types can be used to define the type of inputs and outputs, as well as the internal signals and variables used in the design.
2. **Libraries:** A library is a collection of pre-defined functions, procedures, and data types that can be used in a design. Libraries are used to organize VHDL code and make it easier to reuse code across different designs. The standard VHDL library is called the "IEEE" library, which includes a range of pre-defined data types and functions.
3. **Packages:** A package in VHDL is a collection of related functions, procedures, and data types that can be used in a design. Packages are similar to libraries, but they are typically used to group related code together, rather than to organize code for reuse across designs. VHDL includes a number of built-in packages, such as the "std_logic_1164" package, which defines the standard logic data type used in most digital designs.
4. **Objects:** In VHDL, an object is a named item that can be assigned a value. Objects can be either signals or variables. Signals are used to represent physical connections between components in a digital design, while variables are used to store temporary values within the VHDL code.
5. **Predefined Data Types:** VHDL includes a number of predefined data types, such as the standard logic types "std_logic" and "std_logic_vector". These data types are used to represent the values of signals and variables within the design. They can be used to represent binary data, such as 0's and 1's, as well as more complex data types, such as ASCII characters or floating-point numbers.

6. **User-Defined Data Types:** VHDL allows designers to define their own custom data types. This can be done using VHDL packages, which can define new data types, such as records or enumerated types.

Vivado Design Suite: is a powerful design environment from Xilinx, a leading provider of programmable logic solutions. Vivado Design Suite is used to design, implement, and verify digital circuits for FPGAs and other programmable devices.

Some of the key features of Vivado Design Suite include:

1. High-level design: Vivado Design Suite includes a range of high-level design tools that allow designers to quickly and easily create complex digital circuits. These tools include graphical design entry, system-level synthesis, and verification tools.
2. Design automation: Vivado Design Suite includes a range of design automation tools that help to simplify the design process and reduce errors. These tools include automatic placement and routing, design rule checking, and timing analysis.
3. IP integration: Vivado Design Suite includes a wide range of intellectual property (IP) cores that can be easily integrated into designs. These IP cores include processors, memory controllers, communication interfaces, and other commonly used components.
4. Debugging and analysis: Vivado Design Suite includes a range of debugging and analysis tools that help designers to identify and fix issues in their designs. These tools include waveform viewers, debuggers, and static timing analysis.
5. Hardware acceleration: Vivado Design Suite includes tools for hardware acceleration, which allow designers to offload computationally intensive tasks to FPGAs. This can result in significant performance improvements compared to software-based solutions.

Overall, Vivado Design Suite is a comprehensive design environment that provides designers with the tools they need to create high-quality, high-performance digital circuits for a wide range of applications.

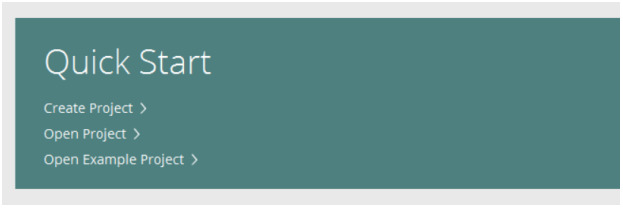
EXPERIMENT 1

AIM: To design and simulate *AND Gate*.

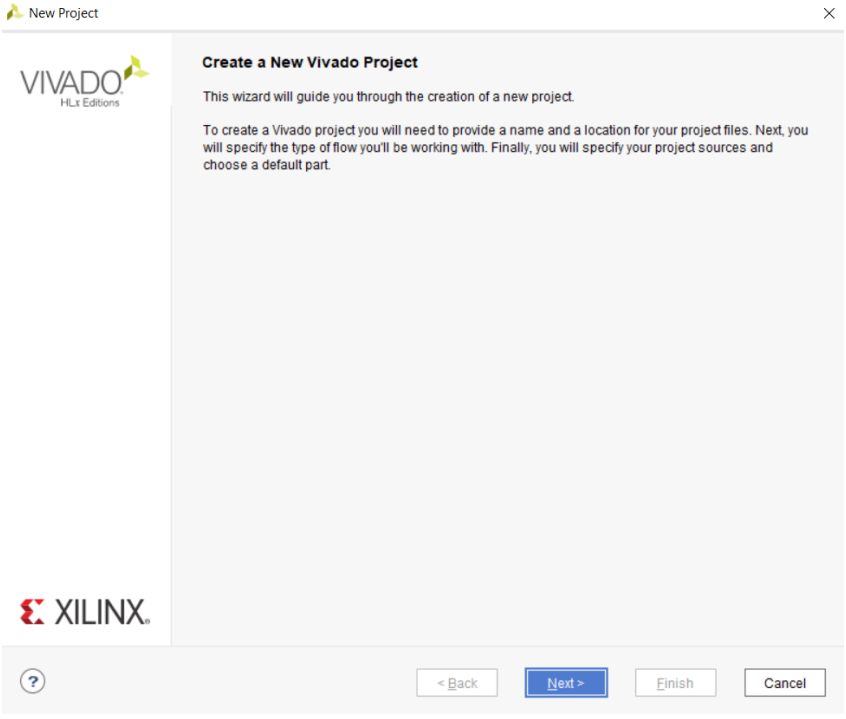
Apparatus Required: Vivado Design Suite.

Procedure:

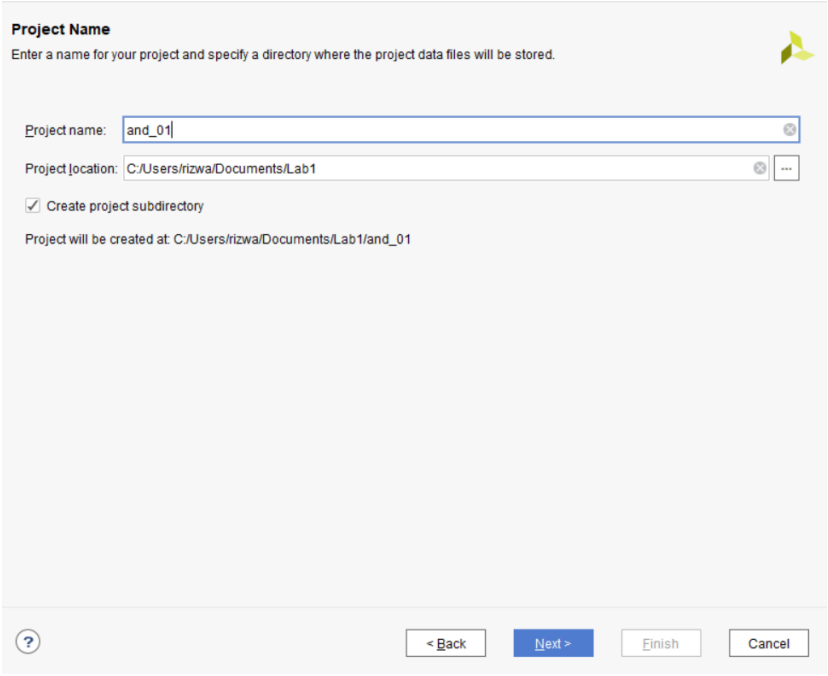
1) **To create an *AND gate*:**
   - Open *Vivado Design Suite*
   - Click on *Create Project* option.



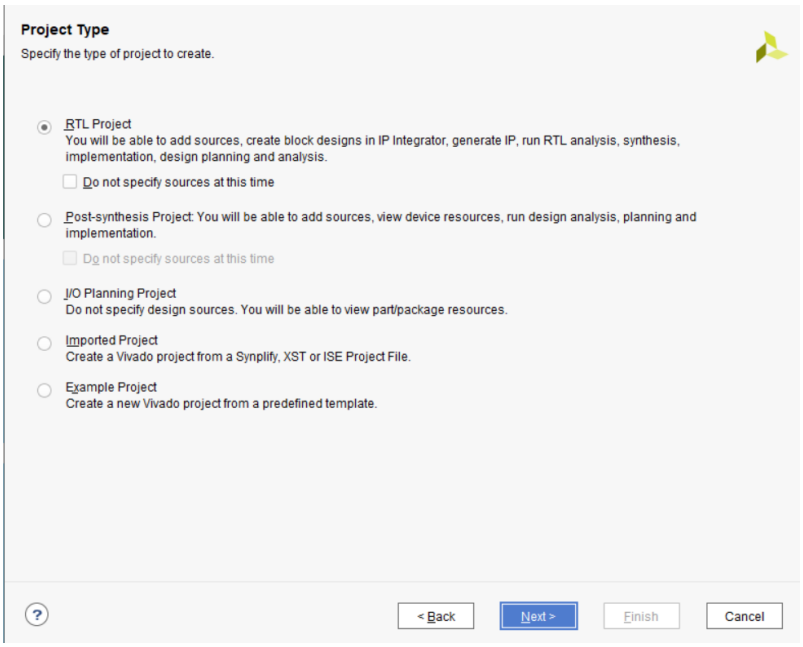
   - Click next on *Create Project* window



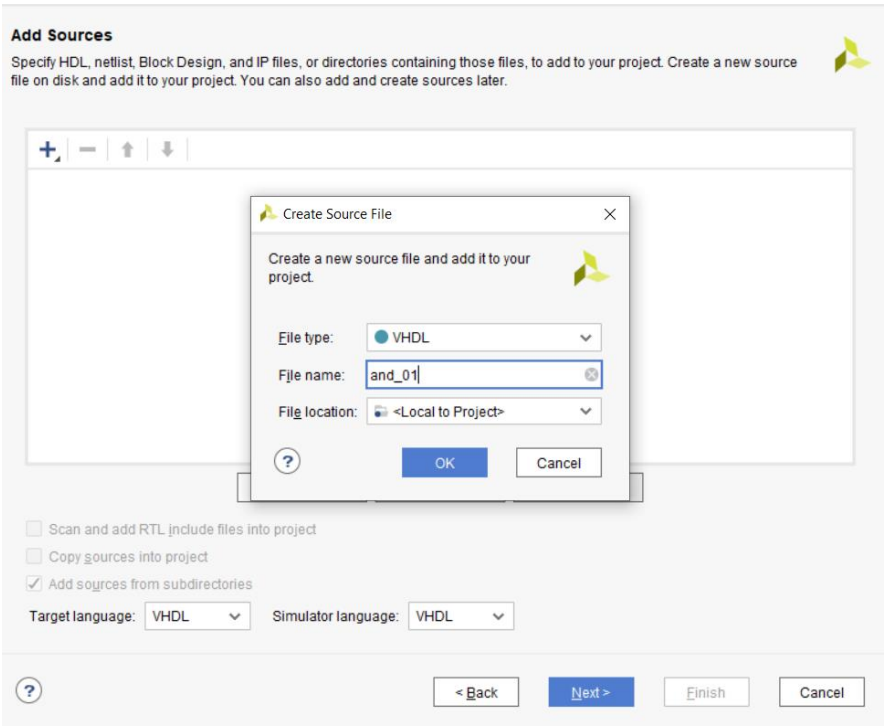
- Name the project *"and_01"* and select the location to save the project, then click next



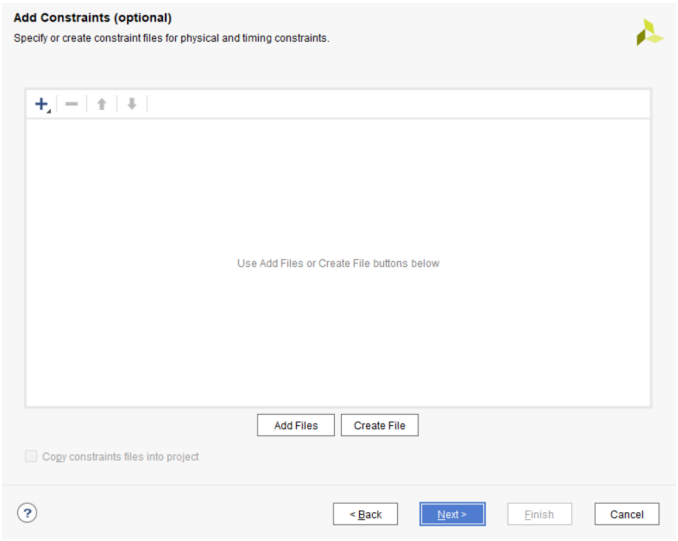
- In Project Type window select *RTL Project* and click next.



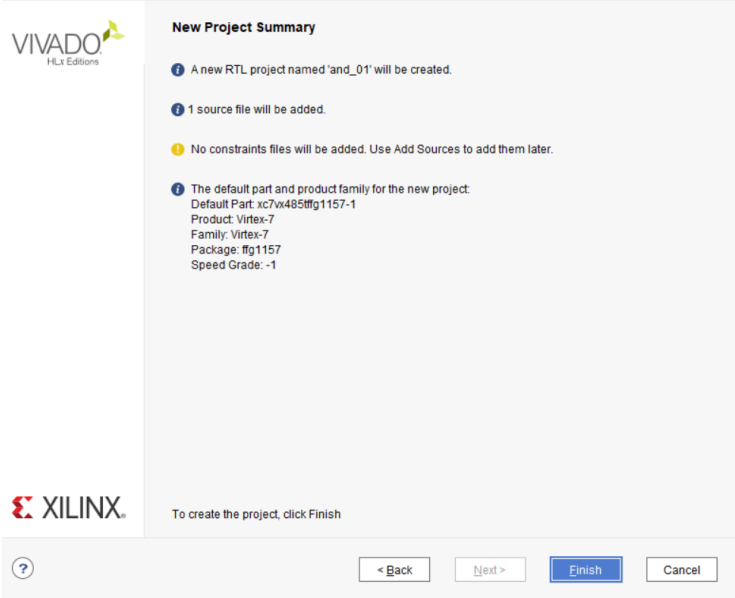
- In Add Sources window click *Create File*, then select file type as *VHDL* and name the file "and_01", click OK. Make sure *VHDL* is selected as Target language and Simulator Language and click Next



- Click next on the Add constraints window and in the Default Part window select the board available.



- Click Finish to create project.



- Define module window appears, in the I/O Definitions tab of the window define the input and output ports for AND gate and click OK.

Port Name	Direction	Bus	MSB	LSB
a	in	☐	0	0
b	in	☐	0	0
c	out	☐	0	0

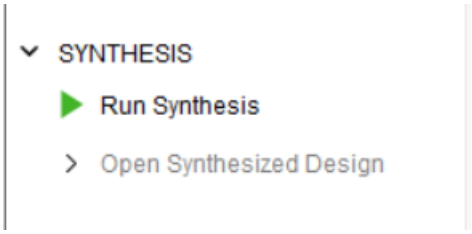
- In the and_01.vhd file add the following code to define (Behavioral) the gate and save it by pressing ctrl + s:

Project Summary | and_01.vhd *

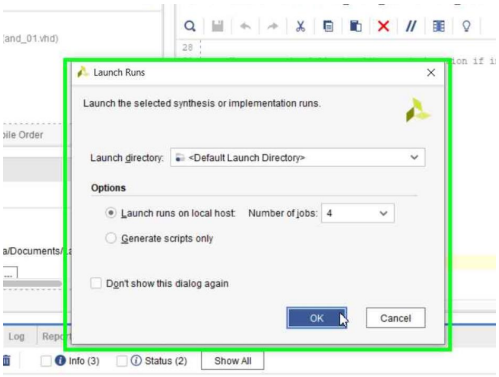
C:/Users/rizwa/Documents/Lab1/and_01/and_01.srcs/sources_1/new/and_01.vhd

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3  entity and_01 is
4      Port ( a : in STD_LOGIC;
5             b : in STD_LOGIC;
6             c : out STD_LOGIC);
7  end and_01;
8
9  architecture Behavioral of and_01 is
10
11 begin
12
13 c<= a and b;
14
15 end Behavioral;
16
```

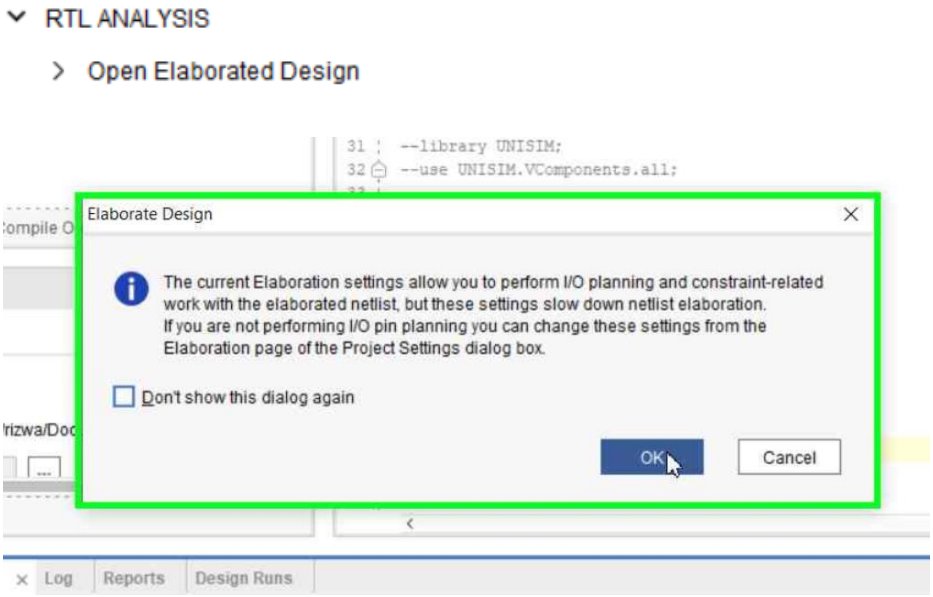
- To synthesize the design, click on Run Synthesis:



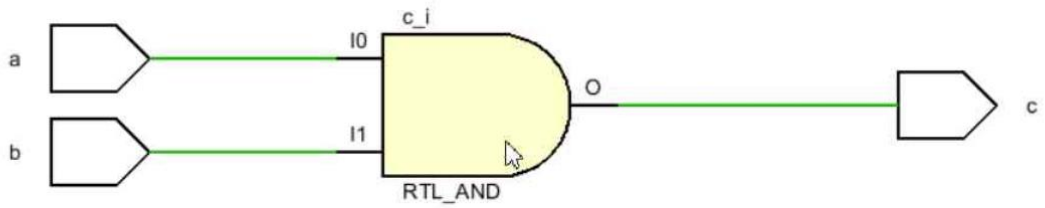
- Then click on OK in Launch Run window



- Close the synthesis completed dialogue, and then Click on Open Elaborated Design and Click OK

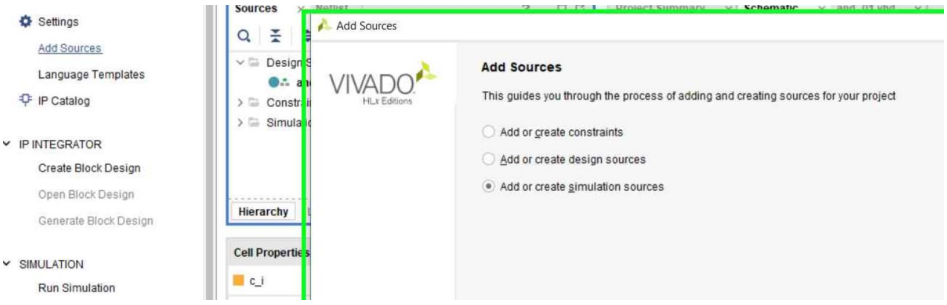


- This shows the schematic of the design

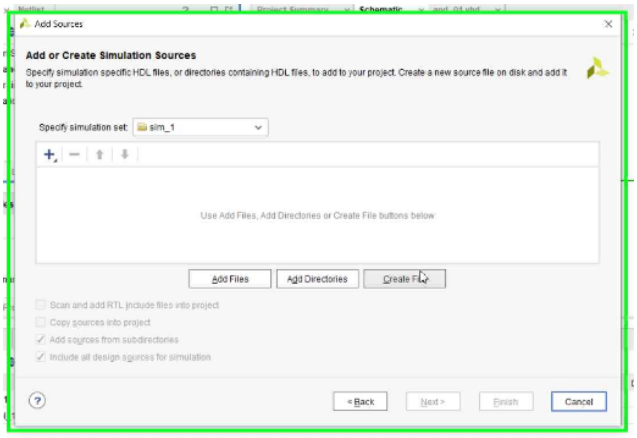


2) To Create a Test Bench:

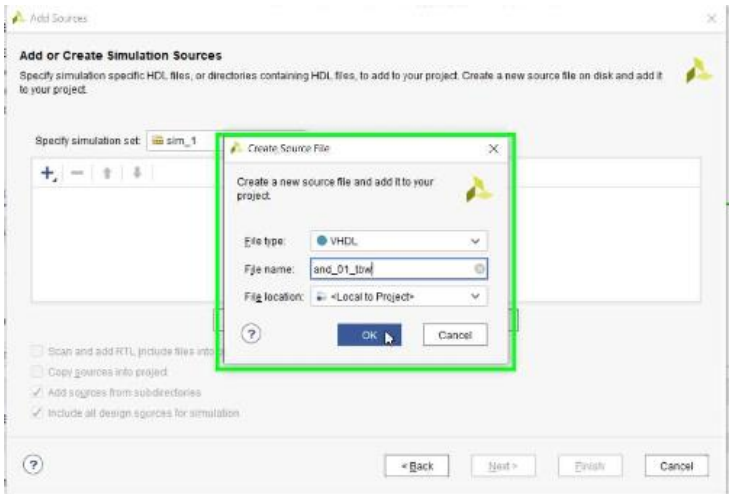
- Click on Add Source and choose Add or Create Simulation Source, then click next.



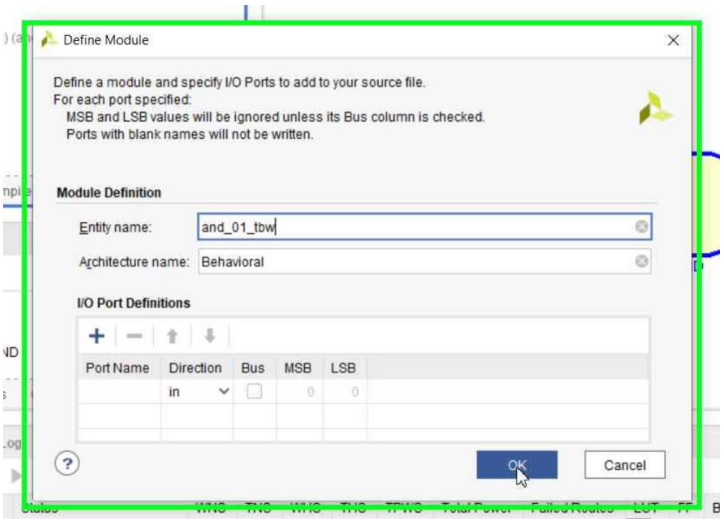
- In Add or Create Simulation Source Window click on Create File



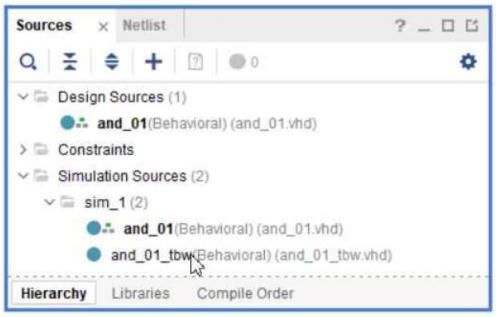
- Select File type as VHDL and name the file “and_01_tbw”. Then click OK and then Finish.



- In Define module window click OK and if the dialogue pops up click on YES.



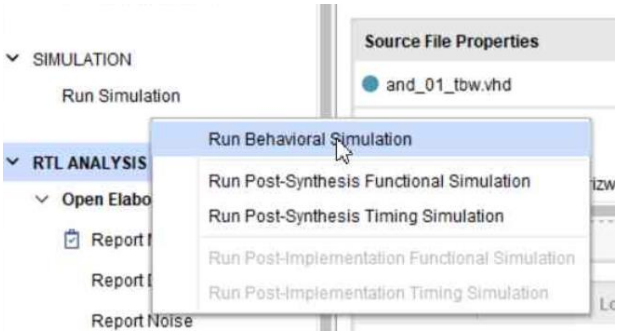
- Under sources window, double click on “and_01_tbw” file.



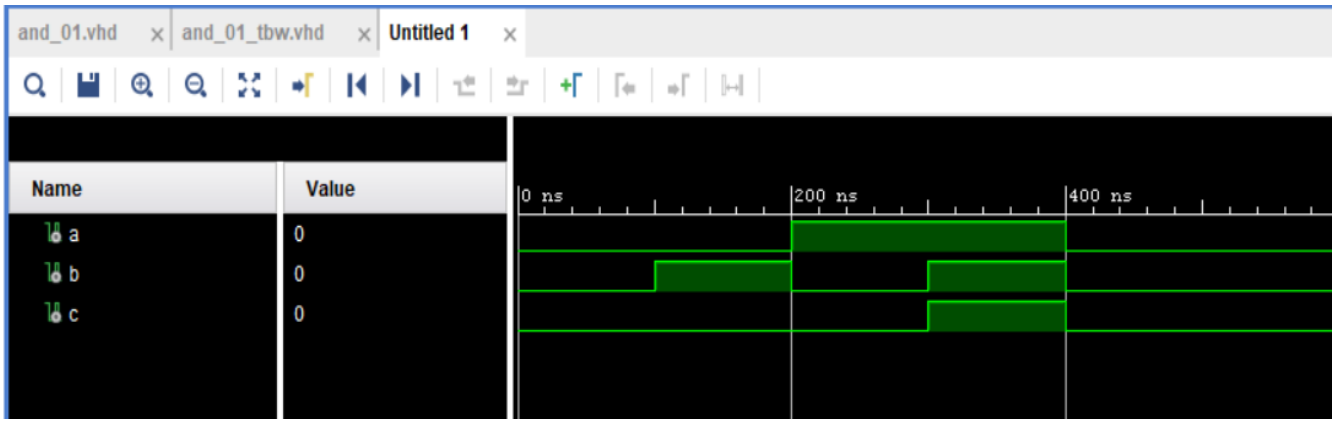
- Then in and_01_tbw.vhd file add the following code:

```
library IEEE;
use IEEE.std_logic_1164.ALL;
ENTITY and_01_tbw IS
END and_01_tbw;
ARCHITECTURE behavior OF and_01_tbw IS
    component and_01 IS
    PORT(
         a : IN  std_logic;
         b : IN  std_logic;
         c : OUT  std_logic
        );
    END COMPONENT;
   signal a : std_logic := '0';
   signal b : std_logic := '0';
   signal c : std_logic;
BEGIN
   uut: and_01 PORT MAP (
          a => a,
          b => b,
          c => c
        );
    stim_proc: process
   begin
      wait for 100 ns;
      a<='0';
      b<='1';
      wait for 100 ns;
      a<='1';
      b<='0';
      wait for 100 ns;
      a<='1';
      b<='1';
      wait for 100 ns;
      a<='0';
      b<='0';
      wait;
   end process;

END;
```

- Save the file and under Simulation option double click on Run Simulation and choose Run Behavioral Simulation



- Behavioral Simulation window appears showing the, waveform of AND gate for different values of input.



EXPERIMENT 2

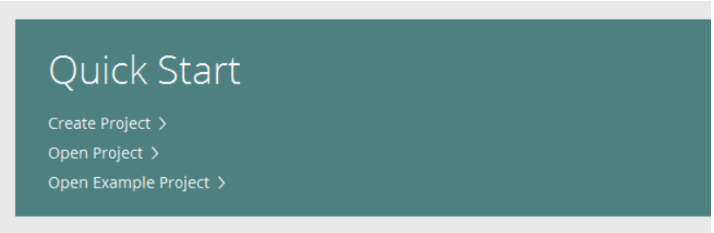
AIM: To design and simulate OR Gate.

Apparatus Required: Vivado Design Suite.

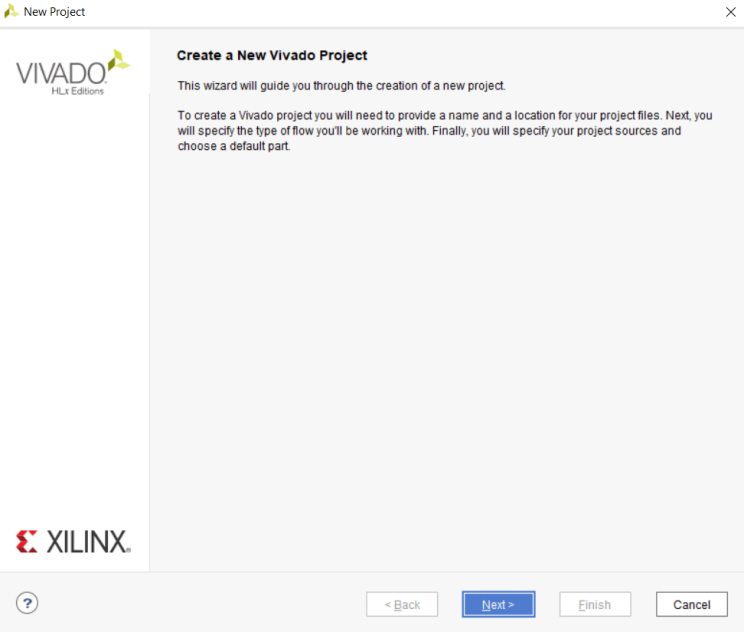
Procedure:

1) **To create an OR gate:**

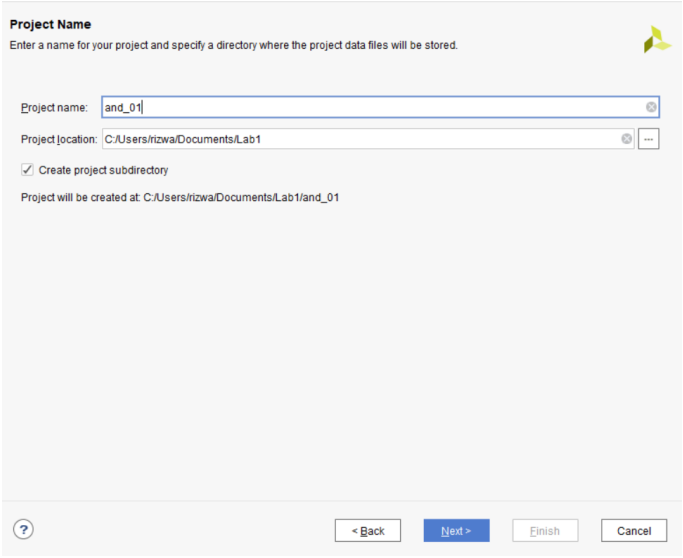
   - Open *Vivado Design Suite*
   - click on *Create Project* option.



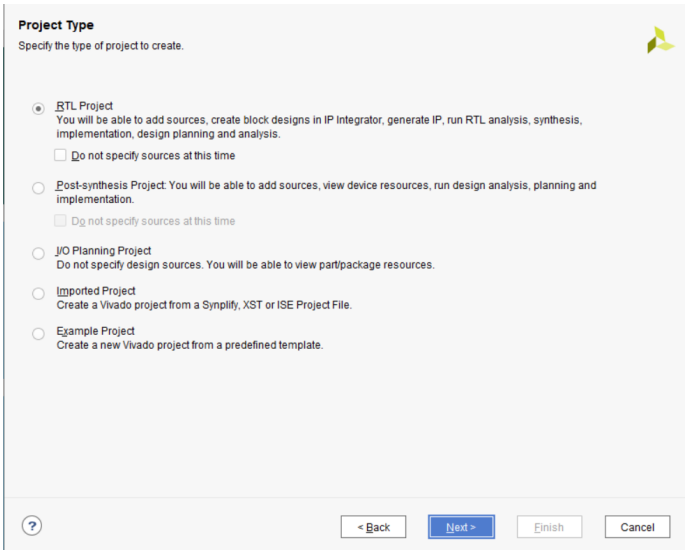
   - Click next on Create Project window



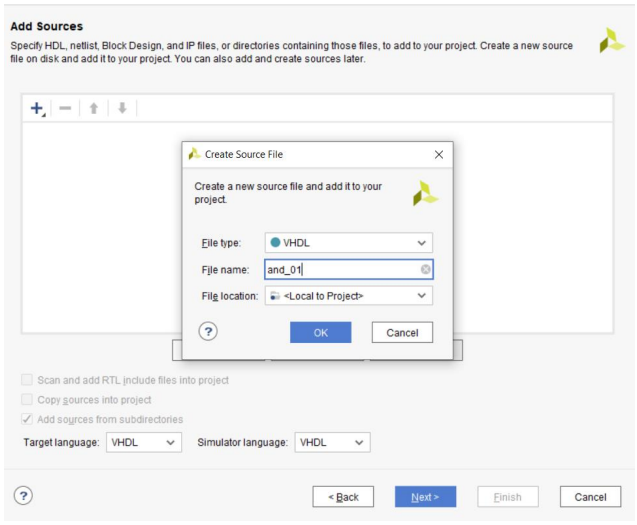
- Name the project “or_01” and select the location to save the project, then click next



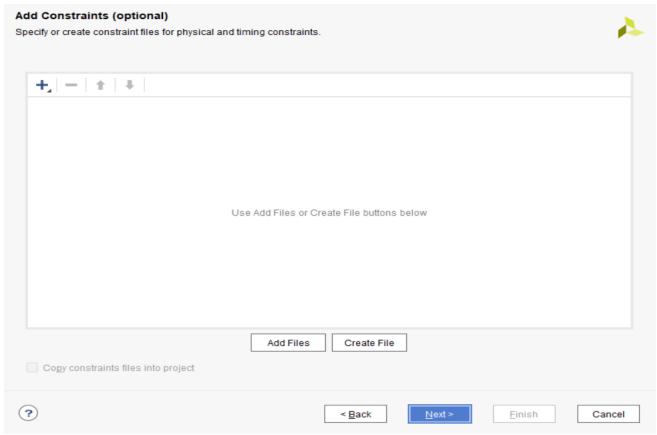
- In Project Type window select *RTL Project* and click next.



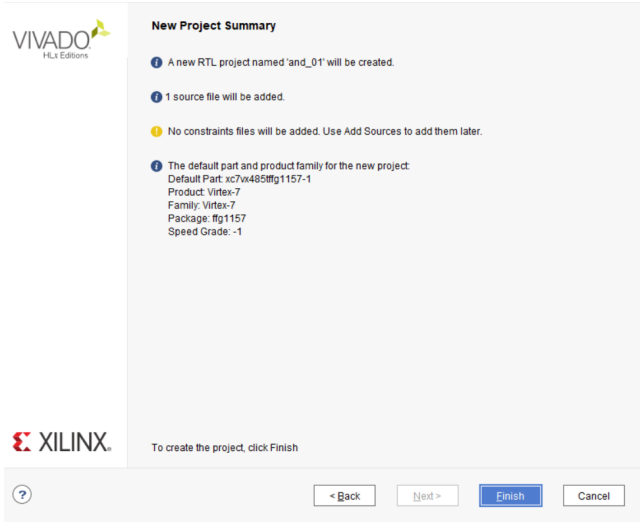
- In Add Sources window click *Create File*, then select file type as *VHDL* and name the file “or_01”, click OK. Make sure *VHDL* is selected as Target language and Simulator Language and click Next



- Click next on the Add constraints window and in the Default Part window select the board available.



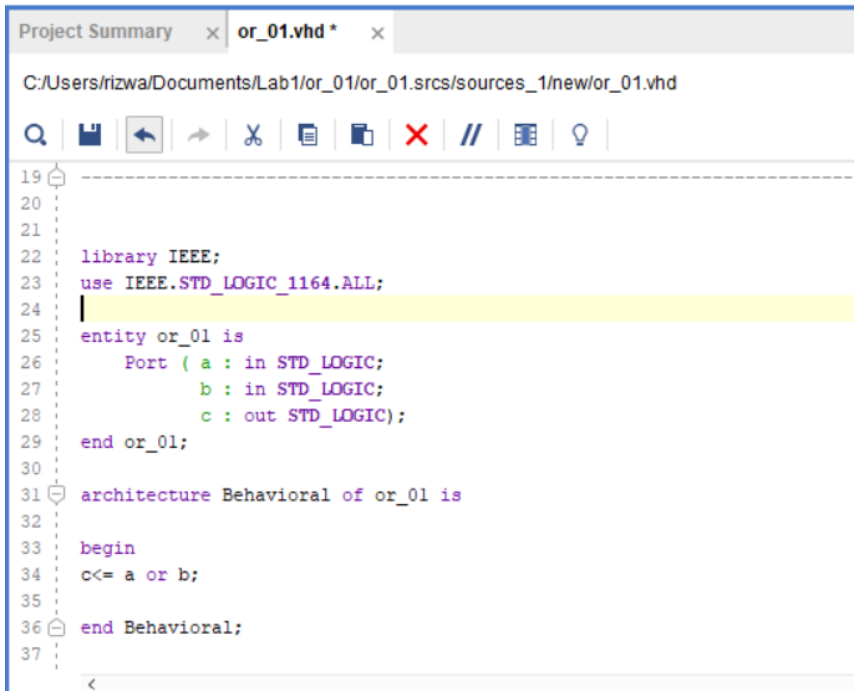
- Click Finish to create project.



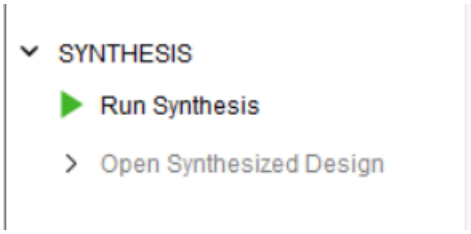
- Define module window appears, in the I/O Definitions tab of the window define the input and output ports for OR gate and click OK.

Port Name	Direction	Bus	MSB	LSB
a	in	☐	0	0
b	in	☐	0	0
c	out	☐	0	0

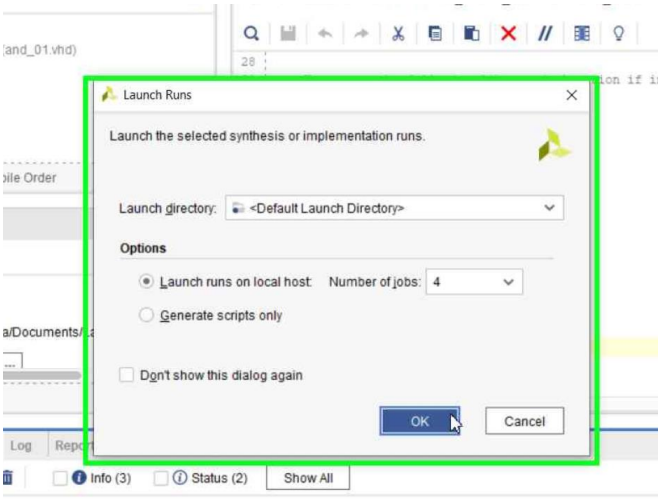
- In the or_01.vhd file add the following code to define (Behavioral) the gate and save it by pressing ctrl + s:



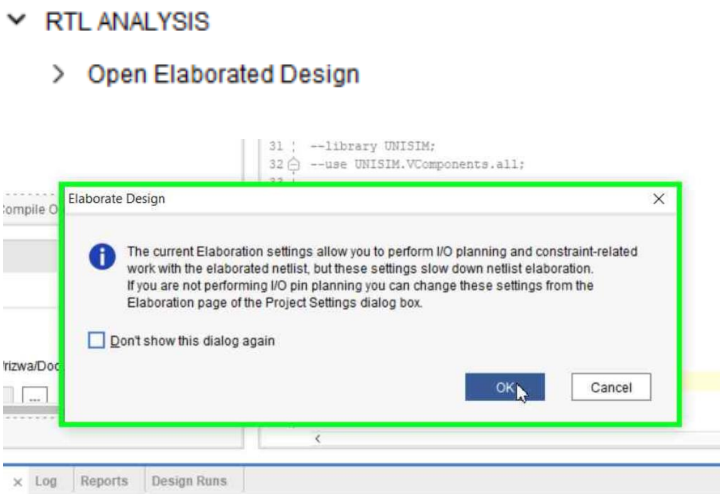
- To synthesize the design, click on Run Synthesis:



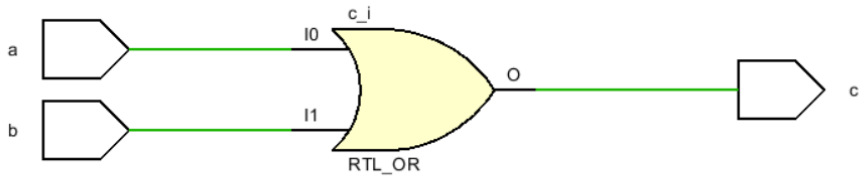
- Then click on OK in Launch Run window



- Close the synthesis completed dialogue, and then Click on Open Elaborated Design and Click OK

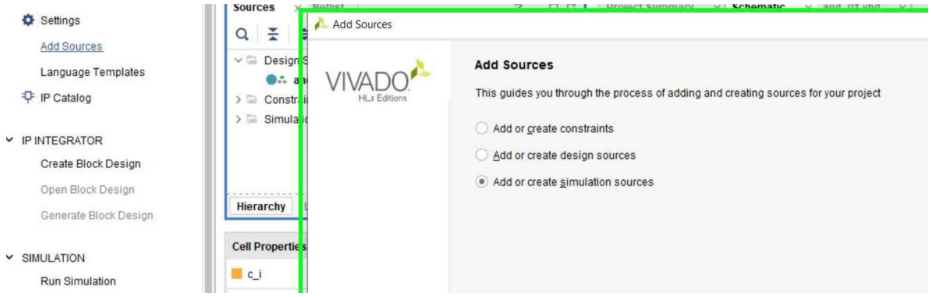


- This shows the schematic of the design

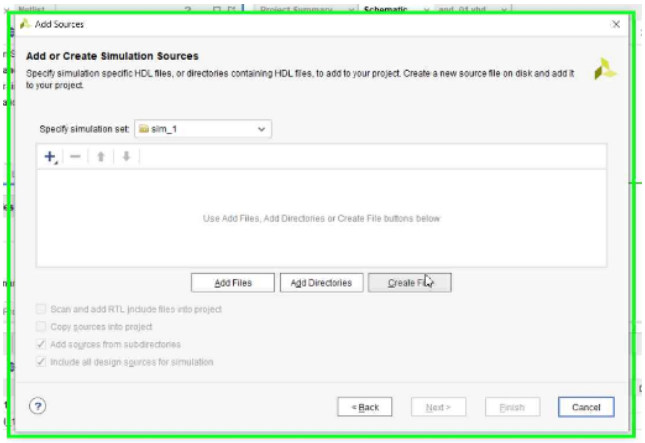


2) To Create a Test Bench:

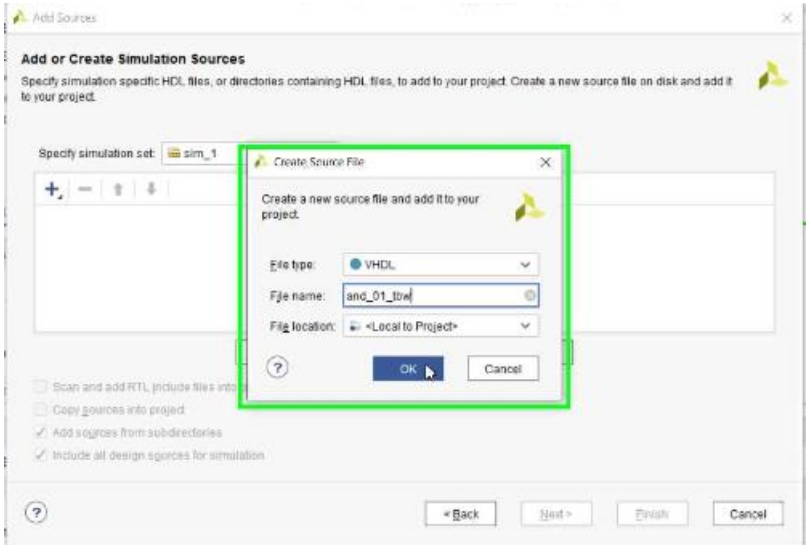
- Click on Add Source and choose Add or Create Simulation Source, then click next.



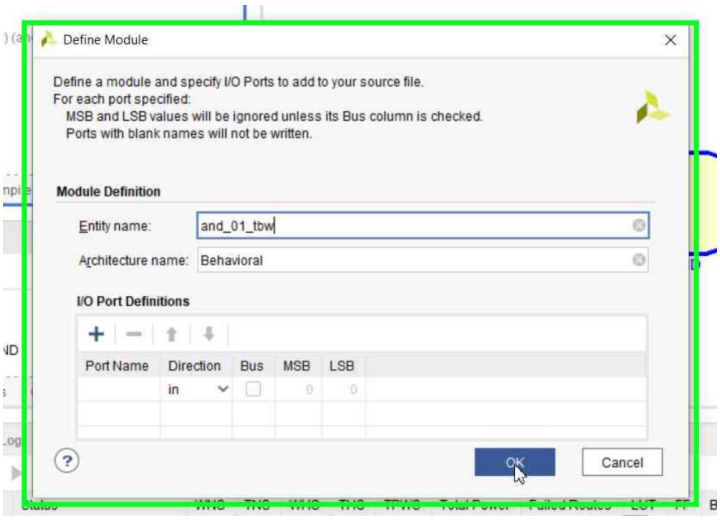
- In Add or Create Simulation Source Window click on Create File



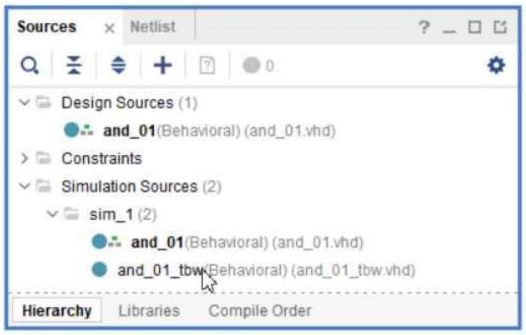
- Select File type as VHDL and name the file “or_01_tbw”. Then click OK and then Finish.



- In Define module window click OK and if the dialogue pops up click on YES.



- Under sources window, double click on “or_01_tbw” file.

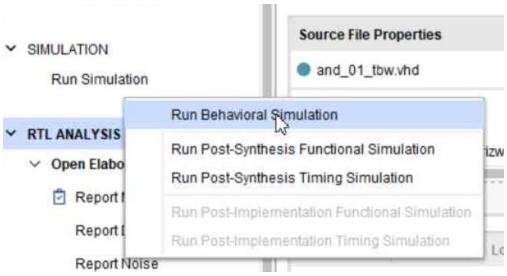


- Then in or_01_tbw.vhd file add the following code:

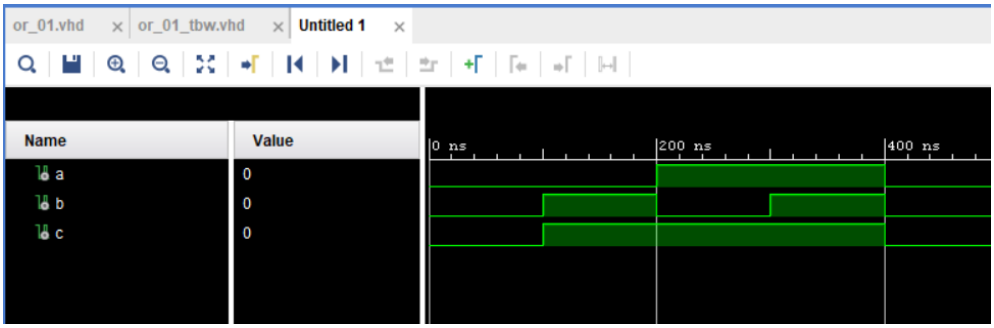
```
library IEEE;
use IEEE.std_logic_1164.ALL;
ENTITY or_01_tbw IS
END or_01_tbw;
ARCHITECTURE behavior OF or_01_tbw IS
   component or_01 IS
   PORT(
        a : IN  std_logic;
        b : IN  std_logic;
        c : OUT  std_logic
       );
   END COMPONENT;
  signal a : std_logic := '0';
  signal b : std_logic := '0';
```

```
            --Outputs
   signal c : std_logic;
BEGIN
    uut: or_01 PORT MAP (
          a => a,
          b => b,
          c => c
        );
   stim_proc: process
   begin
     wait for 100 ns;
     a<='0';
     b<='1';
     wait for 100 ns;
     a<='1';
     b<='0';
     wait for 100 ns;
     a<='1';
     b<='1';
     wait for 100 ns;
     a<='0';
     b<='0';
     wait;
   end process;
END;
```

- Save the file and under Simulation option double click on Run Simulation and choose Run Behavioral Simulation



- Behavioral Simulation window appears showing the, waveform of OR gate for different values of input.



EXPERIMENT 3

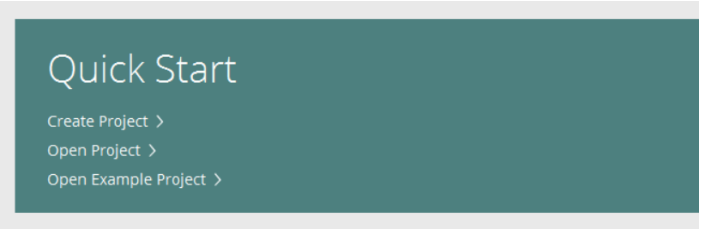
AIM: To design and simulate NAND Gate.

Apparatus Required: Vivado Design Suite.

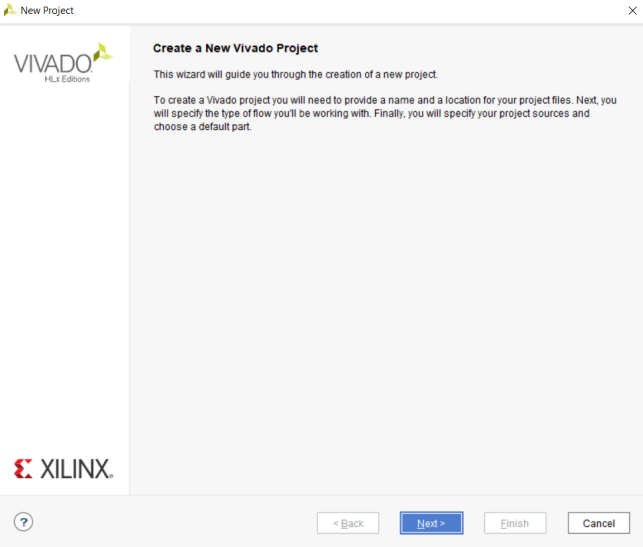
Procedure:

1) **To create a NAND gate:**

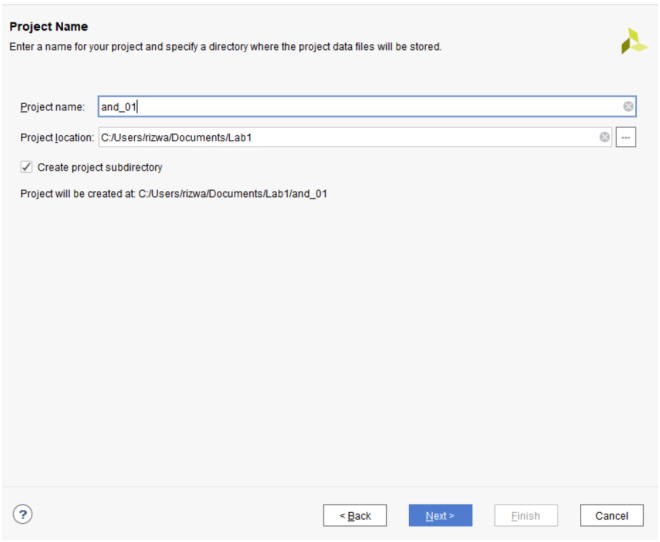
- Open *Vivado Design Suite*
- click on *Create Project* option.



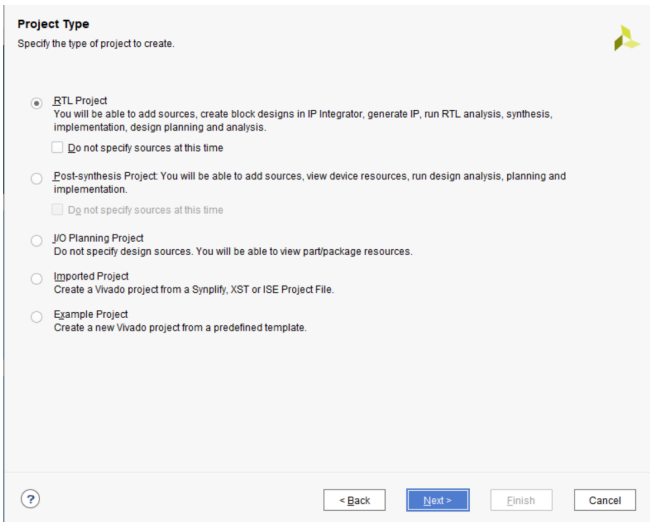
- Click next on Create Project window



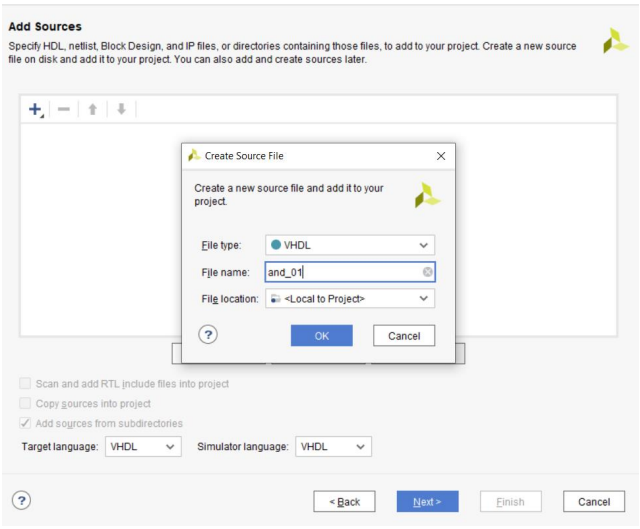
- Name the project “nand_01” and select the location to save the project, then click next



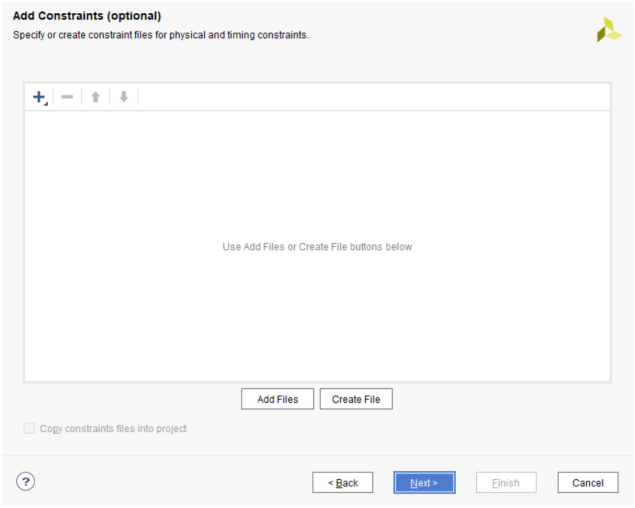
- In Project Type window select *RTL Project* and click next.



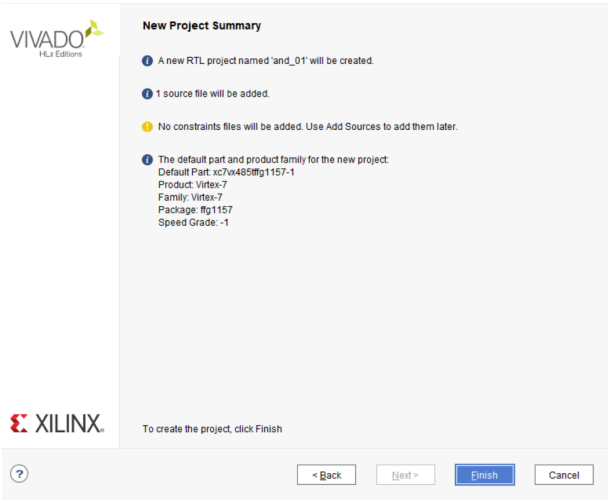
- In Add Sources window click *Create File*, then select file type as *VHDL* and name the file “nand_01”, click OK. Make sure *VHDL* is selected as Target language and Simulator Language and click Next



- Click next on the Add constraints window and in the Default Part window select the board available.



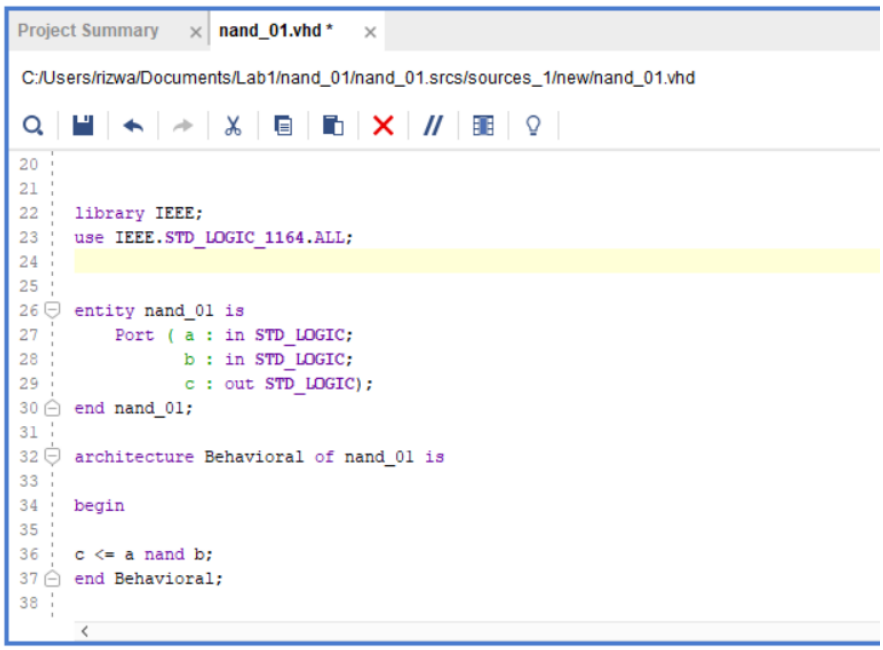
- Click Finish to create project.



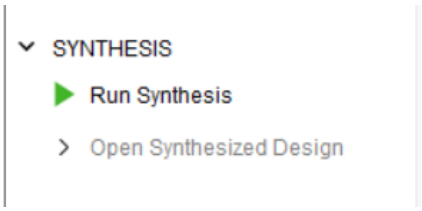
- Define module window appears, in the I/O Definitions tab of the window define the input and output ports for NAND gate and click OK.

Port Name	Direction	Bus	MSB	LSB
a	in	☐	0	0
b	in	☐	0	0
c	out	☐	0	0

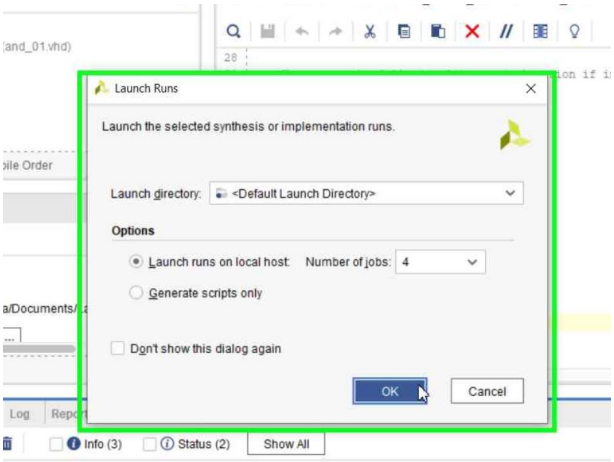
- In the nand_01.vhd file add the following code to define (Behavioral) the gate and save it by pressing ctrl + s:



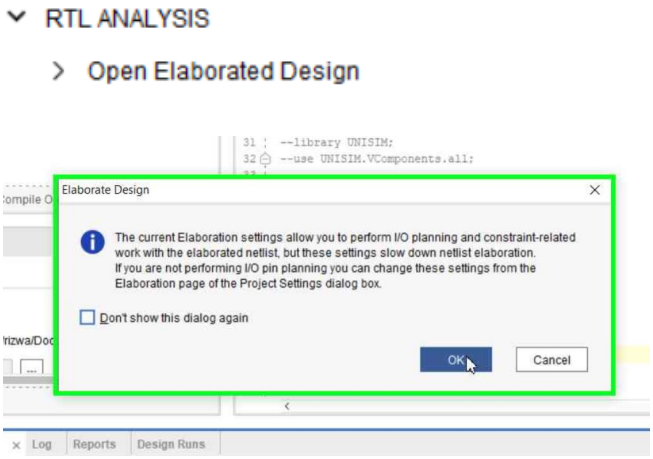
- To synthesize the design, click on Run Synthesis:



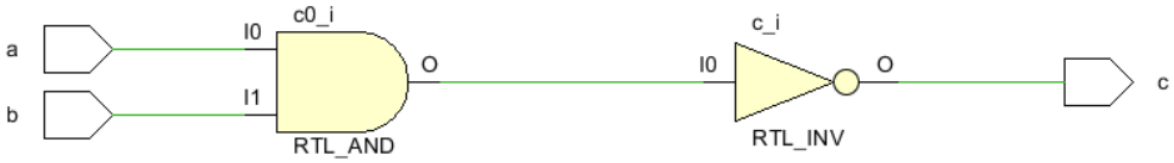
- Then click on OK in Launch Run window



- Close the synthesis completed dialogue, and then Click on Open Elaborated Design and Click OK

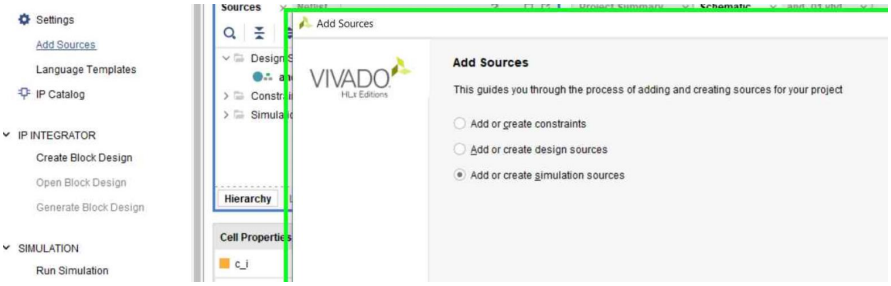


- This shows the schematic of the design

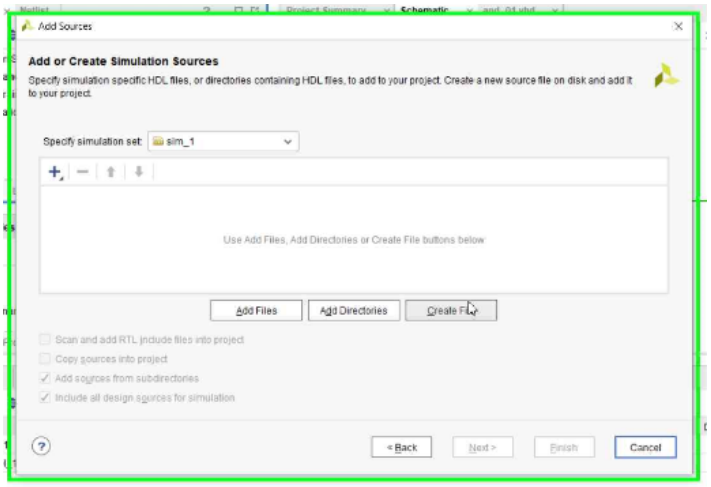


2) To Create a Test Bench:

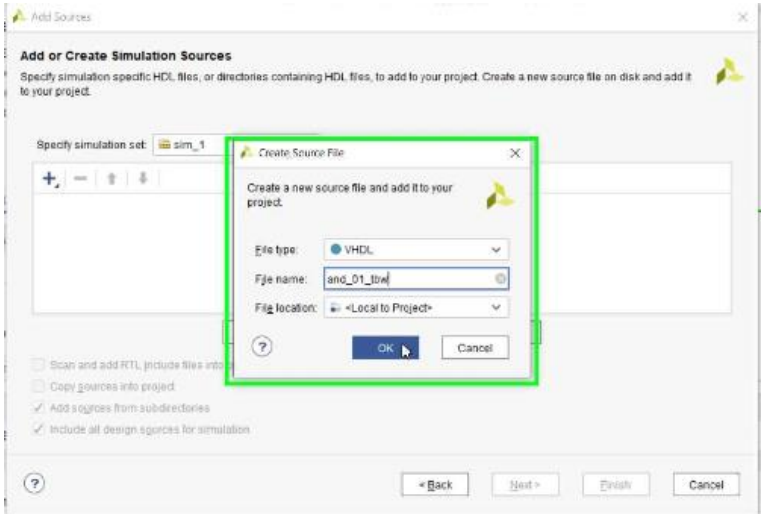
- Click on Add Source and choose Add or Create Simulation Source, then click next.



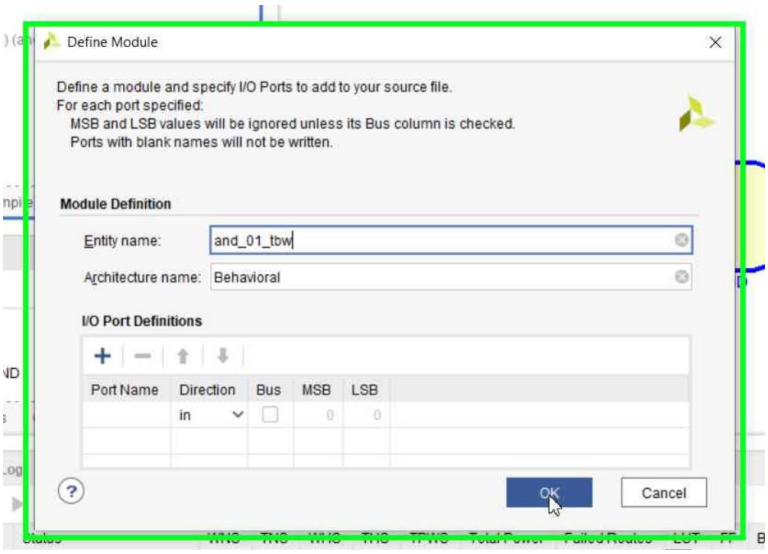
- In Add or Create Simulation Source Window click on Create File



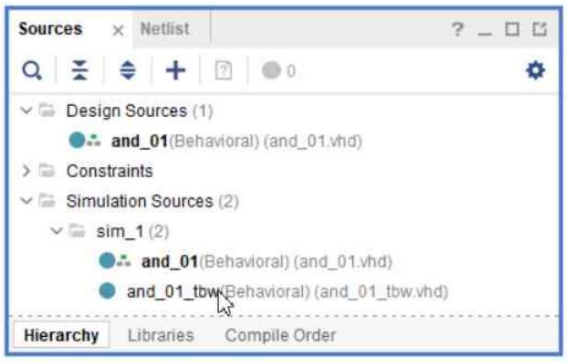
- Select File type as VHDL and name the file “nand_01_tbw”. Then click OK and then Finish.



- In Define module window click OK and if the dialogue pops up click on YES.



- Under sources window, double click on “nand_01_tbw” file.

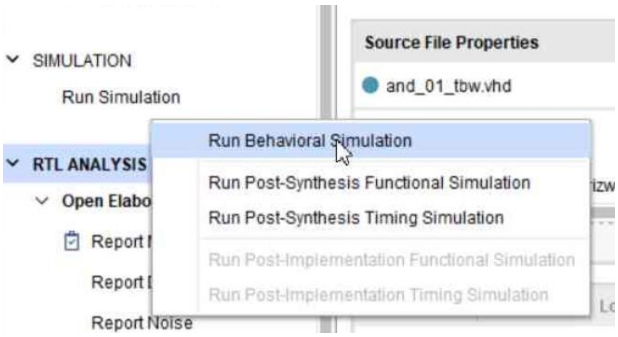


- Then in nand_01_tbw.vhd file add the following code:

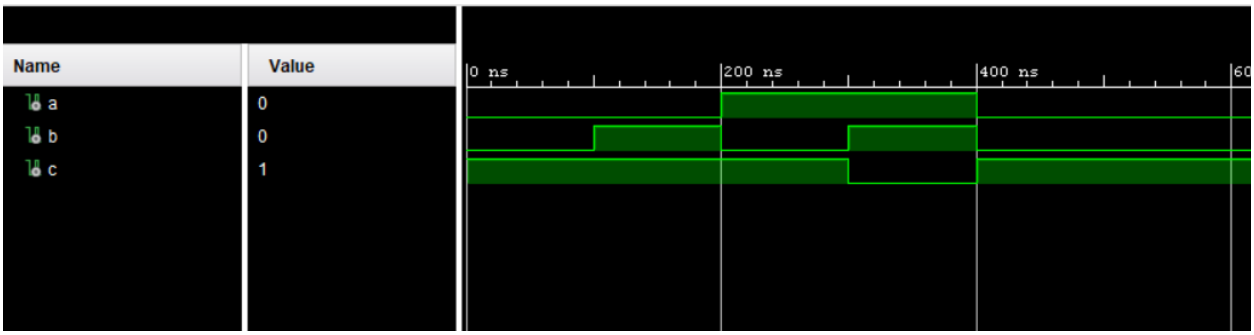
```
library IEEE;
use IEEE.std_logic_1164.ALL;
ENTITY nand_01_tbw IS
END nand_01_tbw;
ARCHITECTURE behavior OF nand_01_tbw IS
   component nand_01 IS
   PORT(
        a : IN  std_logic;
        b : IN  std_logic;
        c : OUT  std_logic
       );
```

```
 END COMPONENT;
  --Inputs
  signal a : std_logic := '0';
  signal b : std_logic := '0';
        --Outputs
  signal c : std_logic;

BEGIN
  uut: nand_01 PORT MAP (
       a => a,
       b => b,
       c => c
     );
  stim_proc: process
  begin
    -- hold reset state for 100 ns.
    wait for 100 ns;
    a<='0';
    b<='1';
    wait for 100 ns;
    a<='1';
    b<='0';
    wait for 100 ns;
    a<='1';
    b<='1';
    wait for 100 ns;
    a<='0';
    b<='0';
    wait;
  end process;

END;
```

- Save the file and under Simulation option double click on Run Simulation and choose Run Behavioral Simulation



- Behavioral Simulation window appears showing the, waveform of NAND gate for different values of input.



EXPERIMENT 4

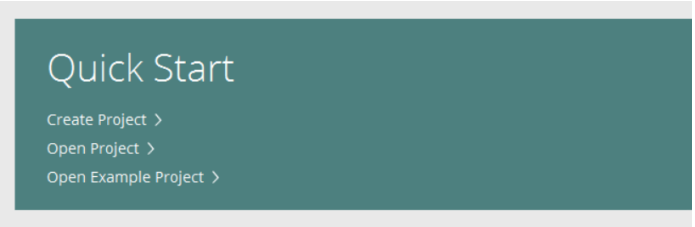
AIM: To design and simulate NOR Gate.

Apparatus Required: Vivado Design Suite.

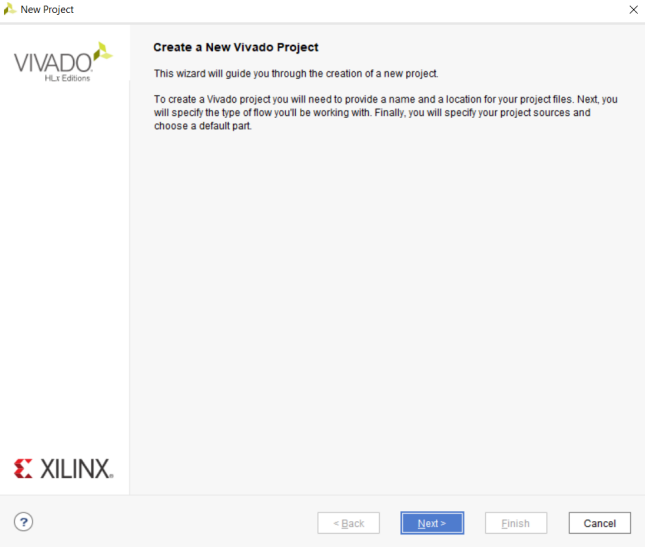
Procedure:

1) **To create a NOR gate:**

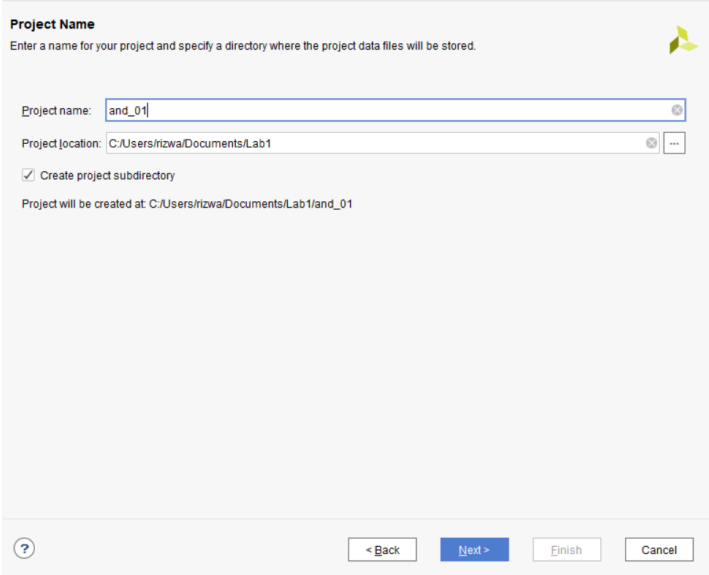
- Open *Vivado Design Suite*
- click on *Create Project* option.



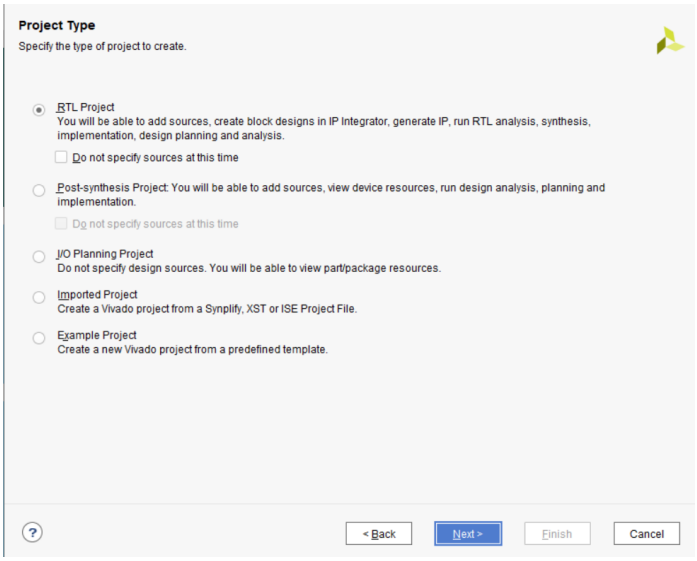
- Click next on Create Project window



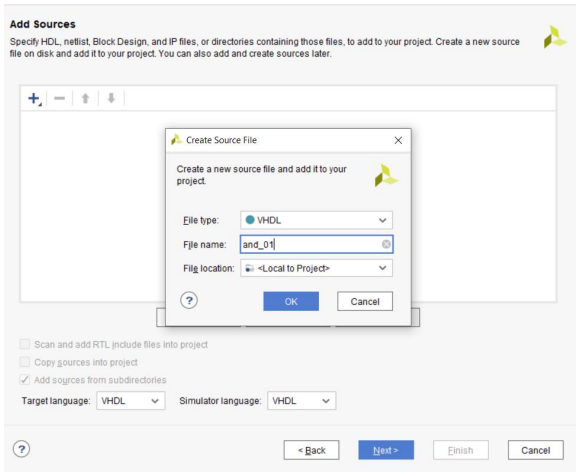
- Name the project “nor_01” and select the location to save the project, then click next



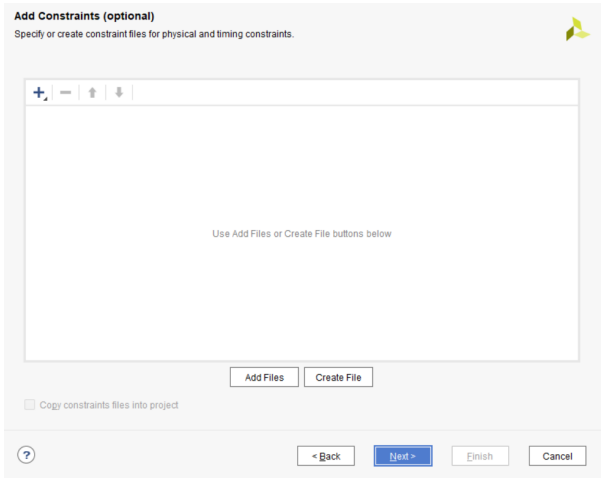
- In Project Type window select *RTL Project* and click next.



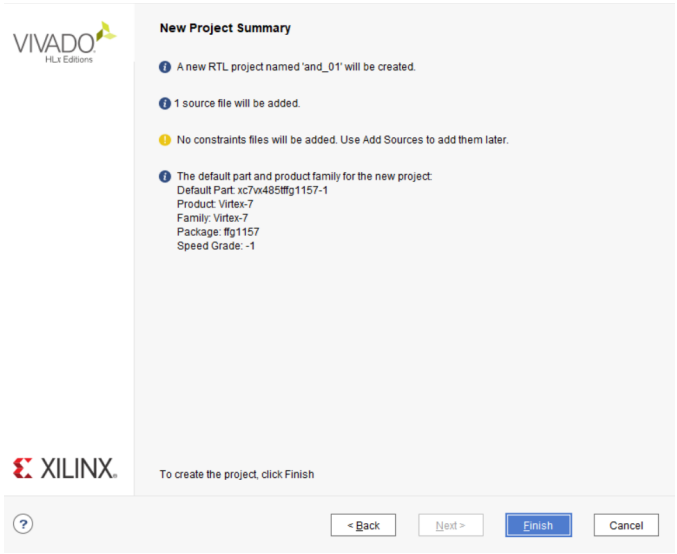
- In Add Sources window click *Create File*, then select file type as *VHDL* and name the file “nor_01”, click OK. Make sure *VHDL* is selected as Target language and Simulator Language and click Next



- Click next on the Add constraints window and in the Default Part window select the board available.



- Click Finish to create project.



- Define module window appears, in the I/O Definitions tab of the window define the input and output ports for NOR gate and click OK.

Port Name	Direction	Bus	MSB	LSB
a	in	☐	0	0
b	in	☐	0	0
c	out	☐	0	0

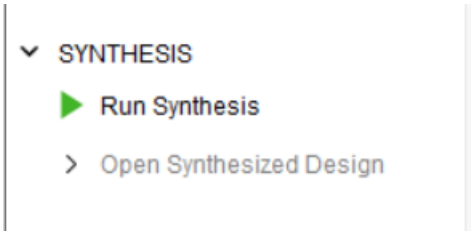
- In the nor_01.vhd file add the following code to define (Behavioral) the gate and save it by pressing ctrl + s:

Project Summary | nor_01.vhd *

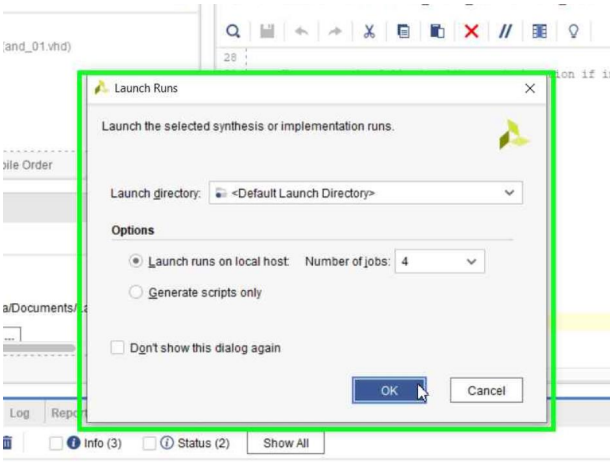
C:/Users/rizwa/Documents/Lab1/nor_01/nor_01.srcs/sources_1/new/nor_01.vhd

```
20
21
22  library IEEE;
23  use IEEE.STD_LOGIC_1164.ALL;
24
25
26  entity nor_01 is
27      Port ( a : in STD_LOGIC;
28             b : in STD_LOGIC;
29             c : out STD_LOGIC);
30  end nor_01;
31
32  architecture Behavioral of nor_01 is
33
34  begin
35
36  c<= a nor b;
37  end Behavioral;
38
```

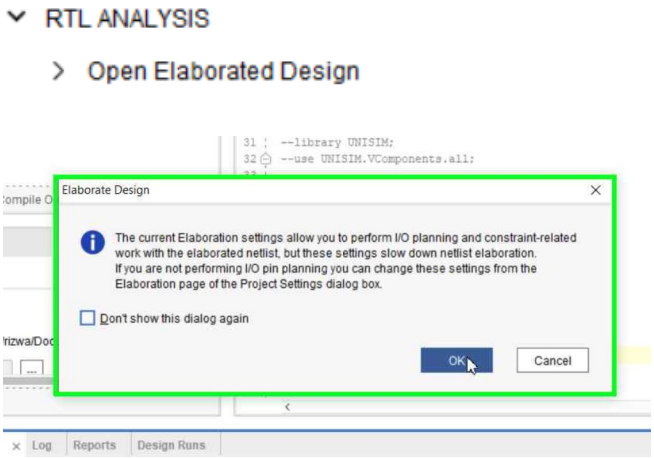
- To synthesize the design, click on Run Synthesis:



- Then click on OK in Launch Run window



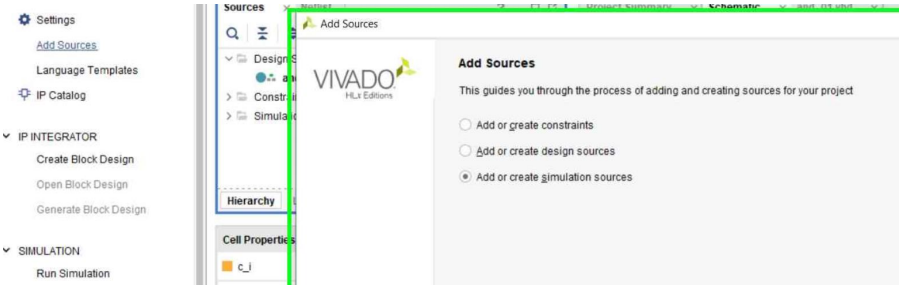
- Close the synthesis completed dialogue, and then Click on Open Elaborated Design and Click OK



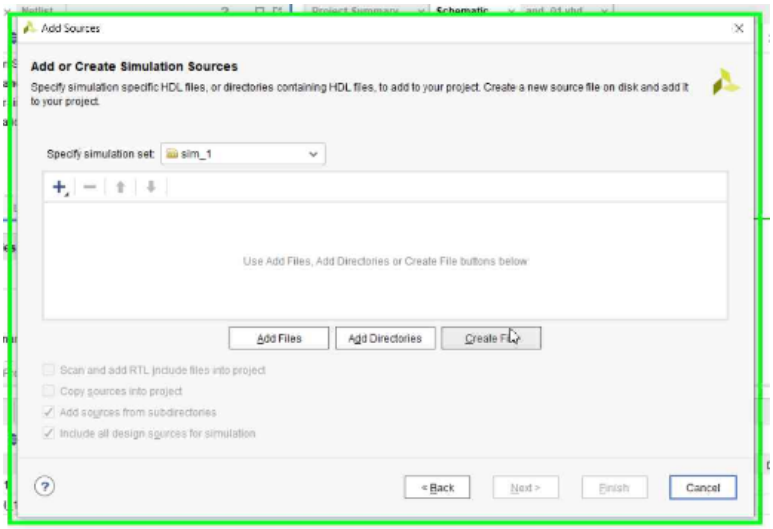
- This shows the schematic of the design

2) To Create a Test Bench:

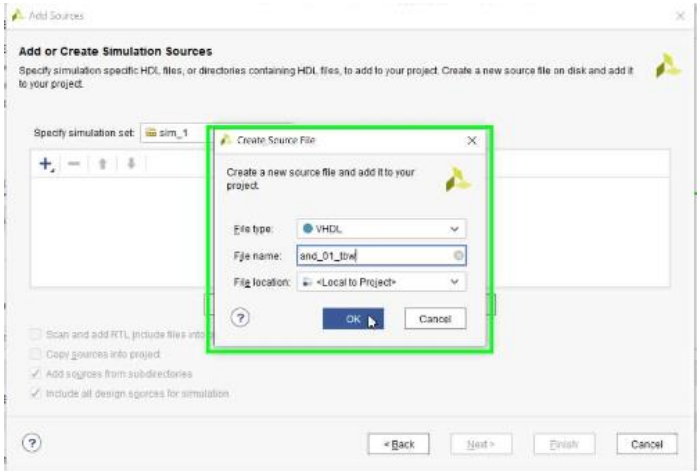
- Click on Add Source and choose Add or Create Simulation Source, then click next.



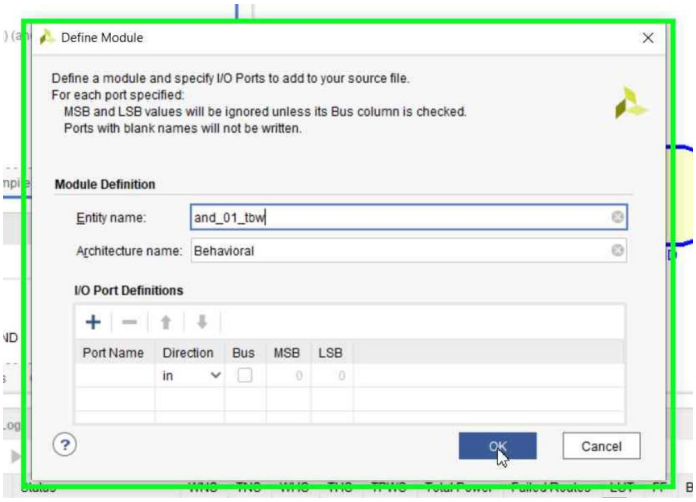
- In Add or Create Simulation Source Window click on Create File



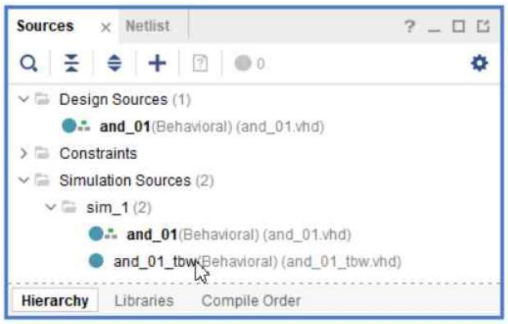
- Select File type as VHDL and name the file “nor_01_tbw”. Then click OK and then Finish.



- In Define module window click OK and if the dialogue pops up click on YES.



- Under sources window, double click on “nor_01_tbw” file.

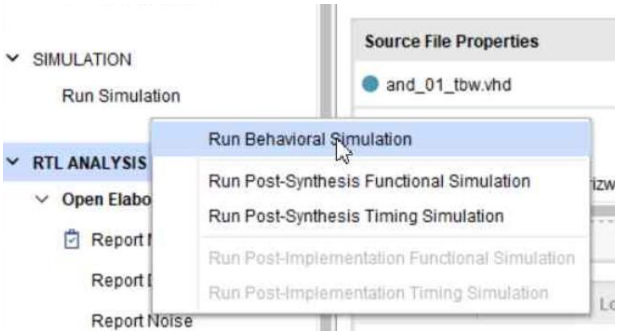


- Then in nor_01_tbw.vhd file add the following code:

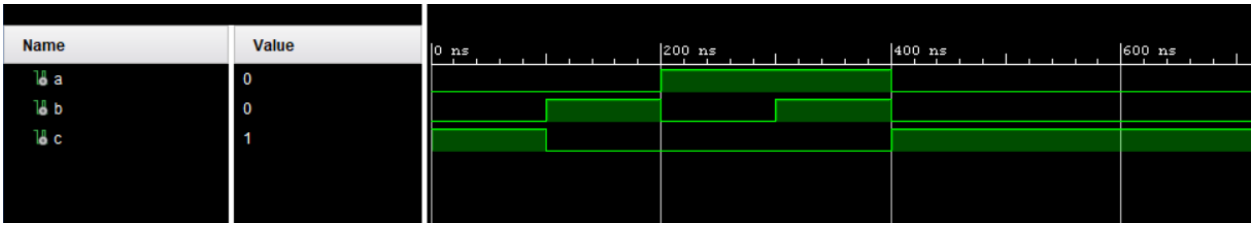
```
library IEEE;
use IEEE.std_logic_1164.ALL;
ENTITY nor_01_tbw IS
END nor_01_tbw;

ARCHITECTURE behavior OF nor_01_tbw IS
   component nor_01 IS
   PORT(
        a : IN  std_logic;
        b : IN  std_logic;
        c : OUT  std_logic
       );
```

```
 END COMPONENT;
signal a : std_logic := '0';
signal b : std_logic := '0';

        --Outputs
signal c : std_logic;

BEGIN
uut: nor_01 PORT MAP (
      a => a,
      b => b,
      c => c
    );

-- Stimulus process
stim_proc: process
begin
  -- hold reset state for 100 ns.
  wait for 100 ns;
  a<='0';
  b<='1';
  wait for 100 ns;
  a<='1';
  b<='0';
  wait for 100 ns;
  a<='1';
  b<='1';
  wait for 100 ns;
  a<='0';
  b<='0';
  wait;
end process;

END;
```

- Save the file and under Simulation option double click on Run Simulation and choose Run Behavioral Simulation



- Behavioral Simulation window appears showing the, waveform of NOR gate for different values of input.



EXPERIMENT 5

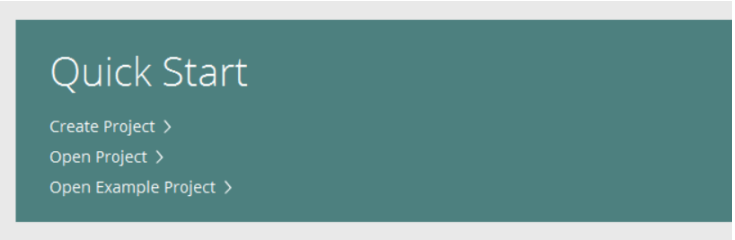
AIM: To design and simulate XOR Gate.

Apparatus Required: Vivado Design Suite.

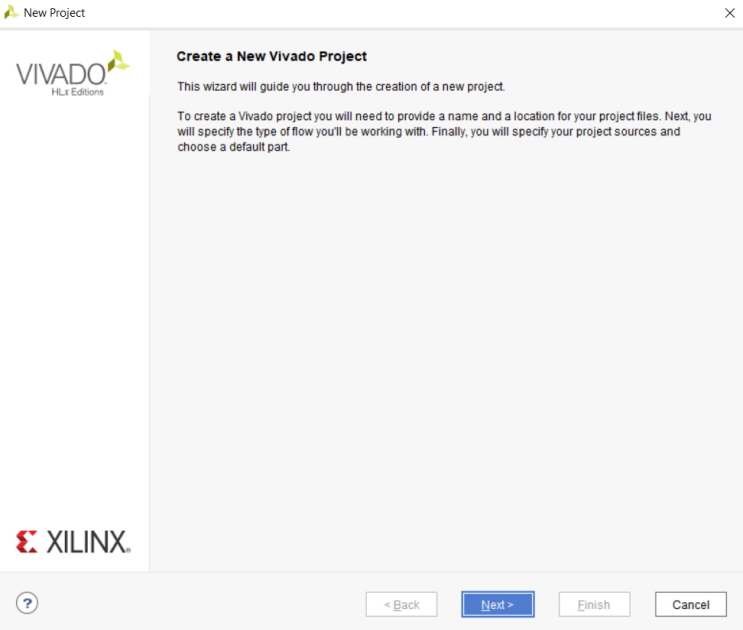
Procedure:

1) **To create a XOR gate:**

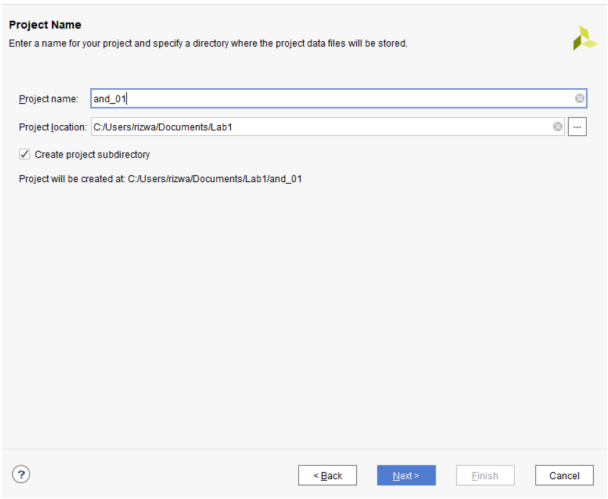
- Open *Vivado Design Suite*
- click on *Create Project* option.



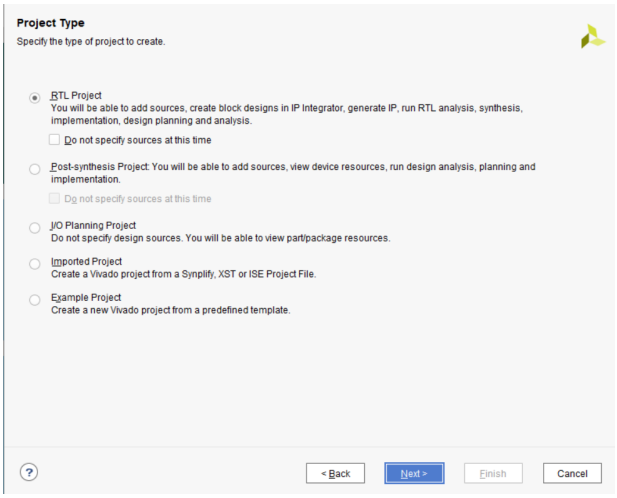
- Click next on Create Project window



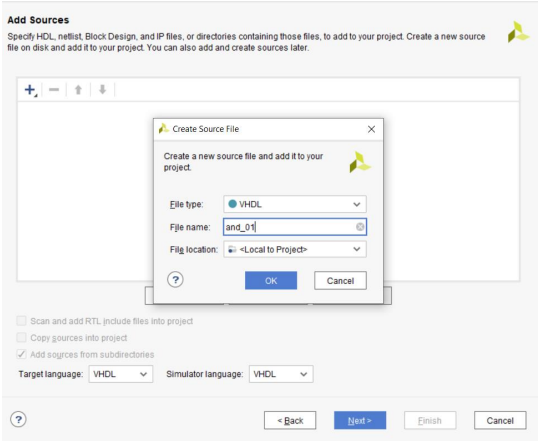
- Name the project “xor_01” and select the location to save the project, then click next



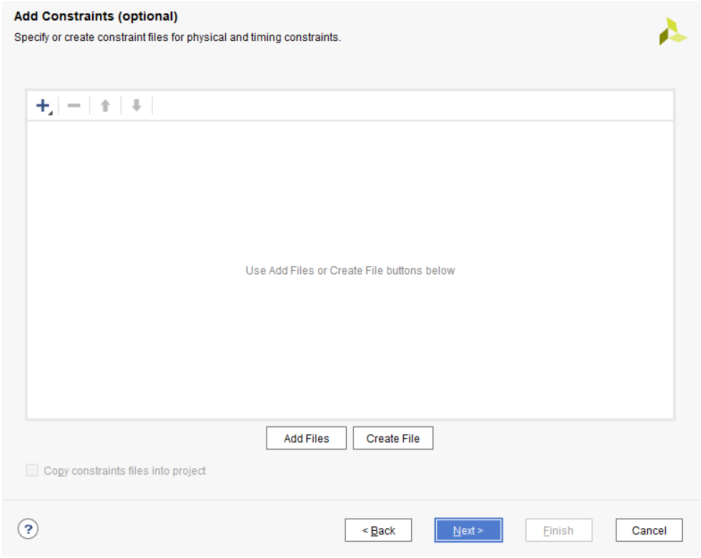
- In Project Type window select *RTL Project* and click next.



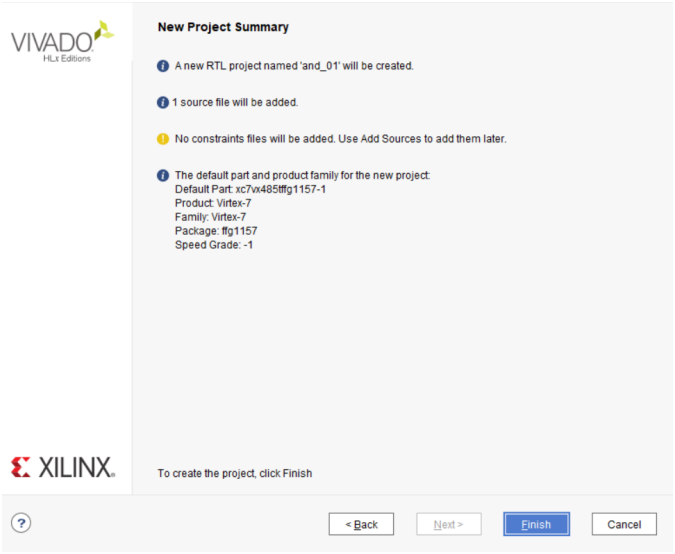
- In Add Sources window click *Create File*, then select file type as *VHDL* and name the file “xor_01”, click OK. Make sure *VHDL* is selected as Target language and Simulator Language and click Next



- Click next on the Add constraints window and in the Default Part window select the board available.



- Click Finish to create project.



- Define module window appears, in the I/O Definitions tab of the window define the input and output ports for XOR gate and click OK.

Port Name	Direction	Bus	MSB	LSB
a	in		0	0
b	in		0	0
c	out		0	0

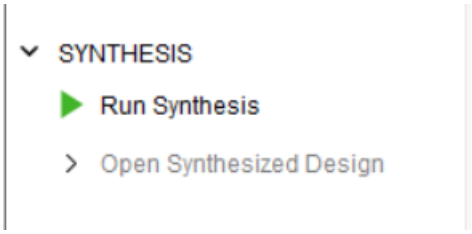
- In the xor_01.vhd file add the following code to define (Behavioral) the gate and save it by pressing ctrl + s:
-

Project Summary | xor_01.vhd

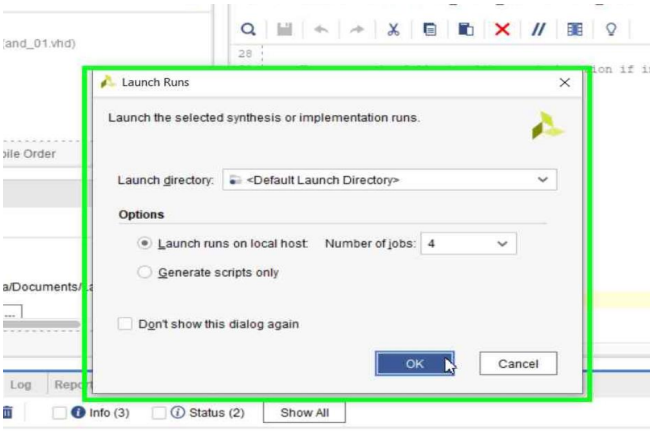
C:/Users/rizwa/Documents/Lab1/xor_01/xor_01.srcs/sources_1/new/xor_01.vhd

```
22  library IEEE;
23  use IEEE.STD_LOGIC_1164.ALL;
24
25
26  entity xor_01 is
27      Port ( a : in STD_LOGIC;
28             b : in STD_LOGIC;
29             c : out STD_LOGIC);
30  end xor_01;
31
32  architecture Behavioral of xor_01 is
33
34  begin
35
36  c<= a xor b;
37  end Behavioral;
38
```

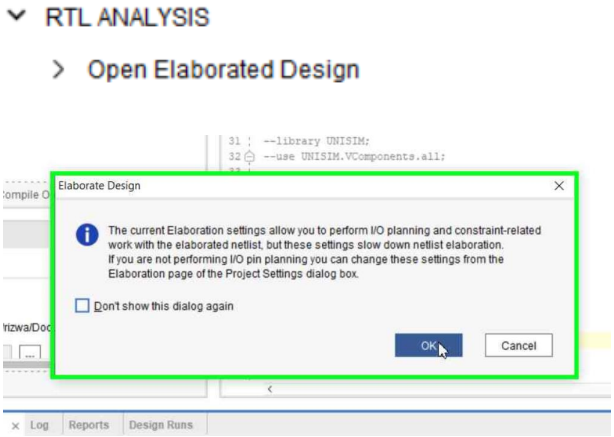
  - To synthesize the design, click on Run Synthesis:



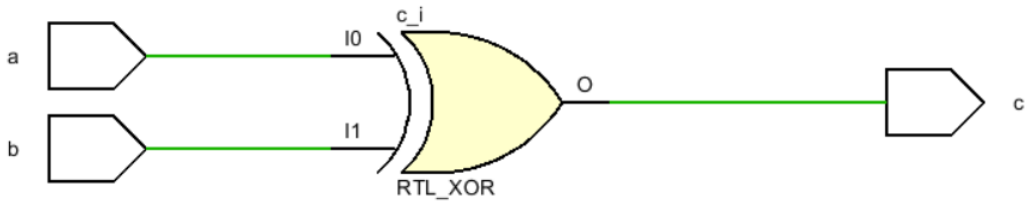
  - Then click on OK in Launch Run window



- Close the synthesis completed dialogue, and then Click on Open Elaborated Design and Click OK

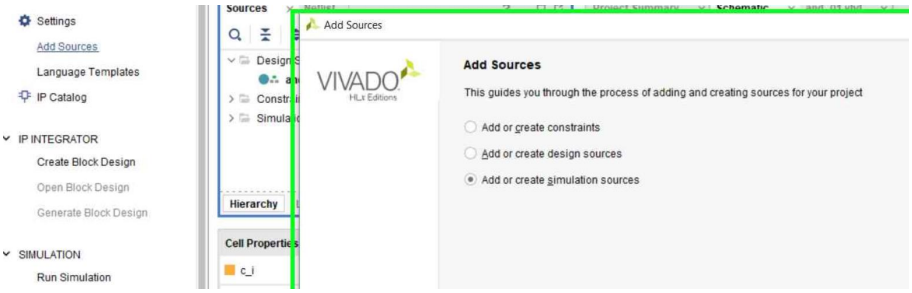


- This shows the schematic of the design

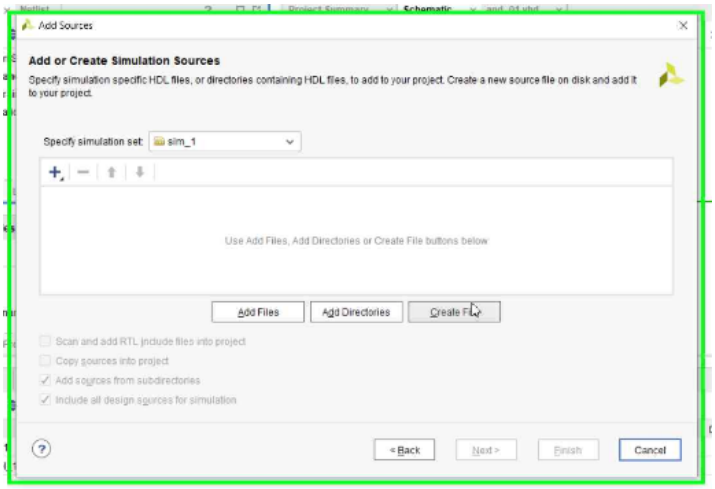


2) To Create a Test Bench:

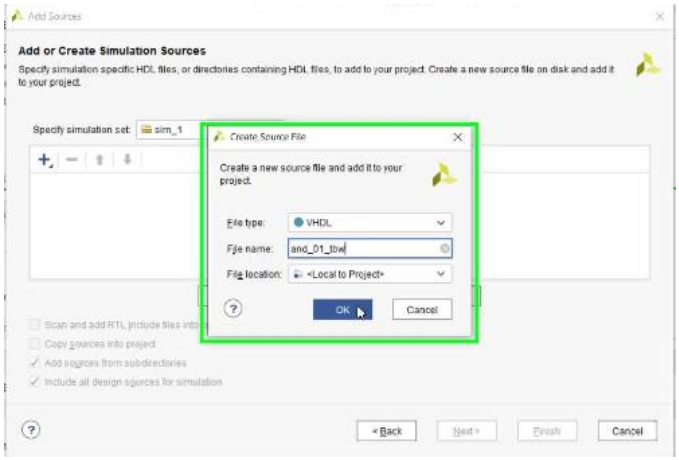
- Click on Add Source and choose Add or Create Simulation Source, then click next.



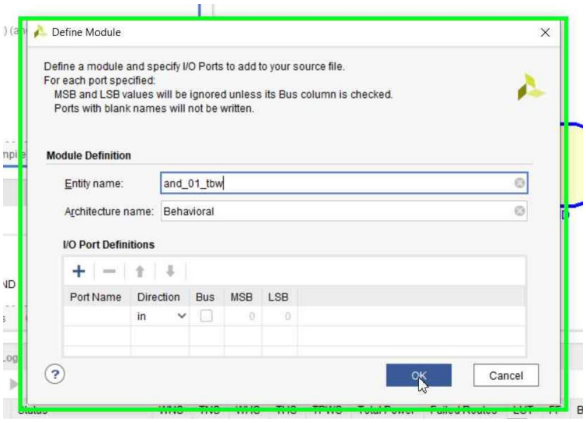
- In Add or Create Simulation Source Window click on Create File



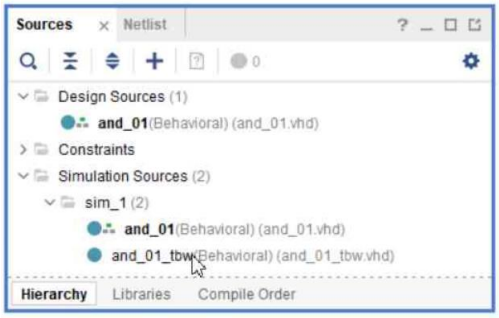
- Select File type as VHDL and name the file “xor_01_tbw”. Then click OK and then Finish.



- In Define module window click OK and if the dialogue pops up click on YES.



- Under sources window, double click on “xor_01_tbw” file.

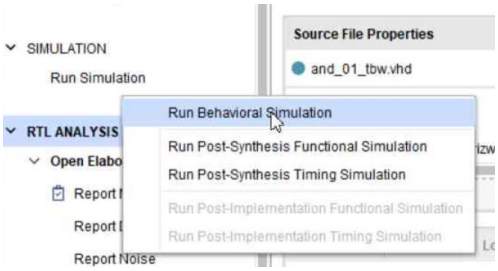


- Then in xor_01_tbw.vhd file add the following code:

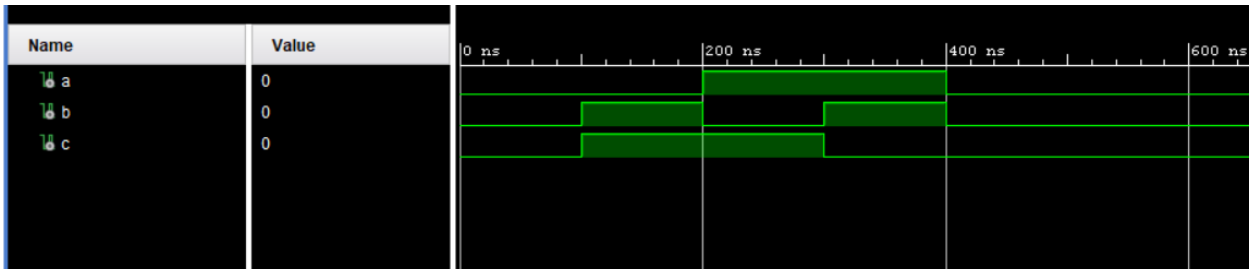
```
library IEEE;
use IEEE.std_logic_1164.ALL;
ENTITY xor_01_tbw IS
END xor_01_tbw;
ARCHITECTURE behavior OF xor_01_tbw IS
    component xor_01 IS
    PORT(
         a : IN  std_logic;
         b : IN  std_logic;
         c : OUT  std_logic
        );
    END COMPONENT;
   signal a : std_logic := '0';
   signal b : std_logic := '0';
   signal c : std_logic;
BEGIN
   uut: xor_01 PORT MAP (
```

```
        a => a,
        b => b,
        c => c
      );
  stim_proc: process
  begin
    -- hold reset state for 100 ns.
    wait for 100 ns;
    a<='0';
    b<='1';
    wait for 100 ns;
    a<='1';
    b<='0';
    wait for 100 ns;
    a<='1';
    b<='1';
    wait for 100 ns;
    a<='0';
    b<='0';
    wait;
  end process;

END;
```

- Save the file and under Simulation option double click on Run Simulation and choose Run Behavioral Simulation



- Behavioral Simulation window appears showing the, waveform of XOR gate for different values of input.



EXPERIMENT 6

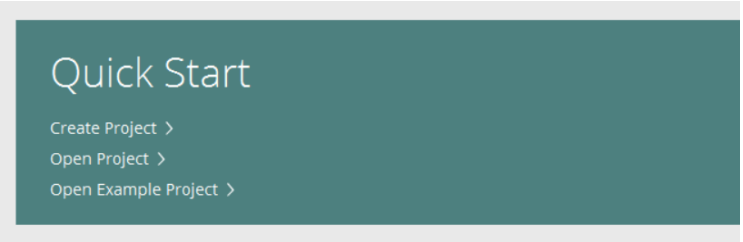
AIM: To design and simulate NOT Gate.

Apparatus Required: Vivado Design Suite.

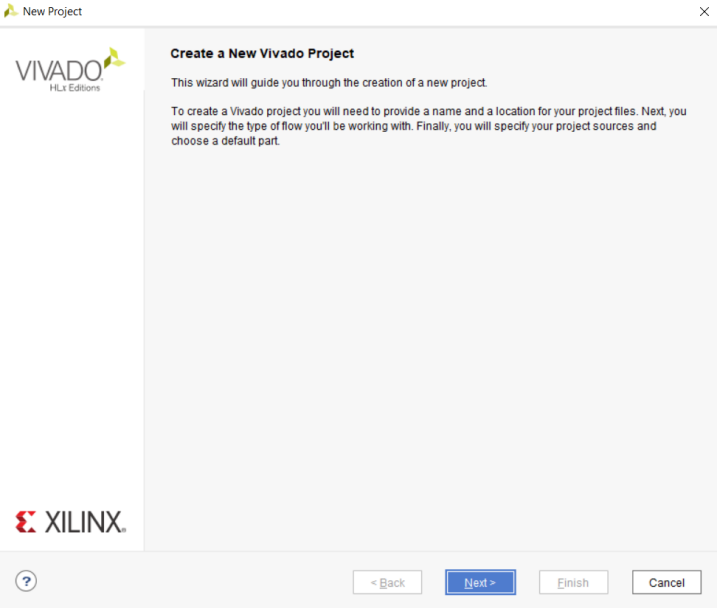
Procedure:

1) **To create a NOT gate:**

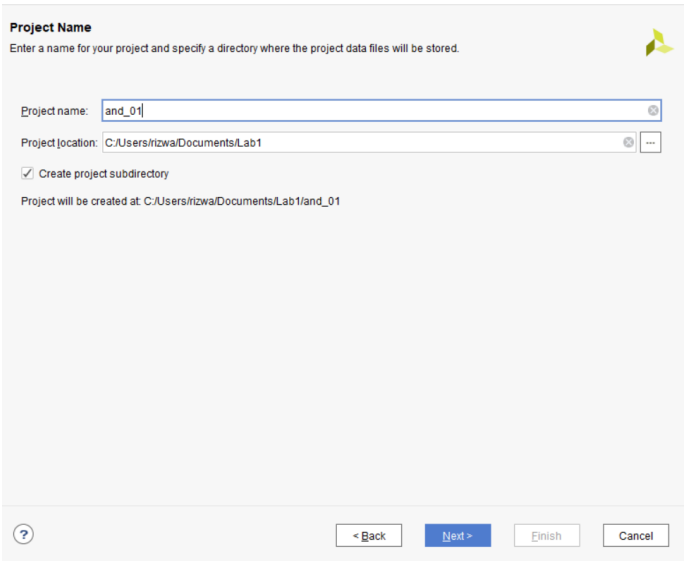
- Open *Vivado Design Suite*
- click on *Create Project* option.



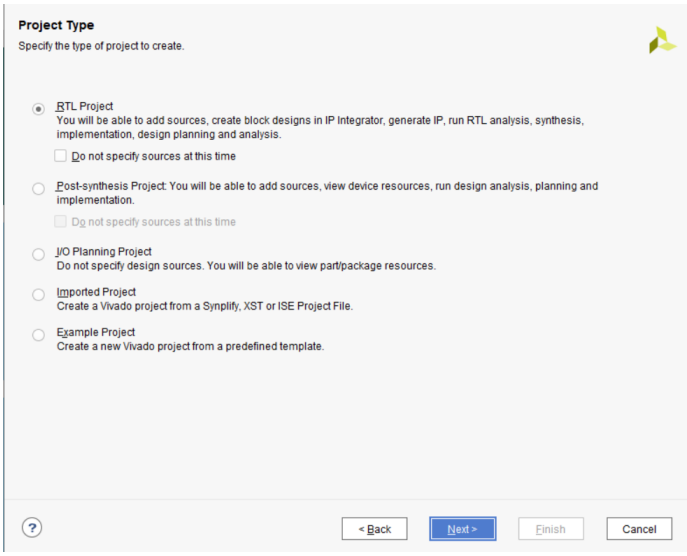
- Click next on Create Project window



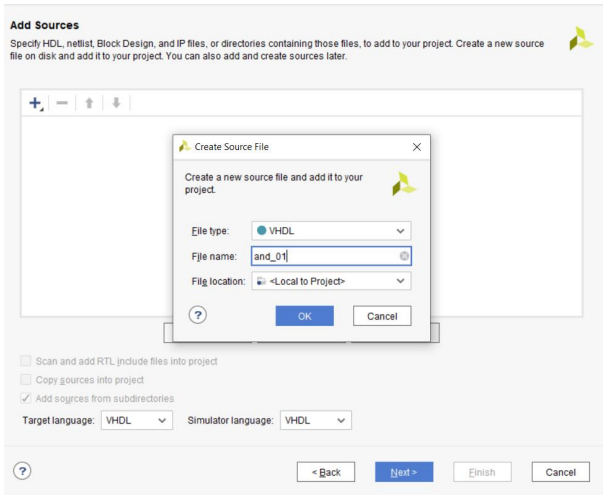
- Name the project “not_01” and select the location to save the project, then click next



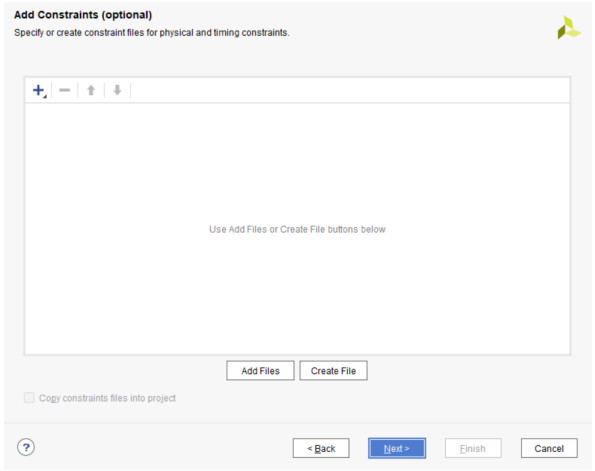
- In Project Type window select *RTL Project* and click next.



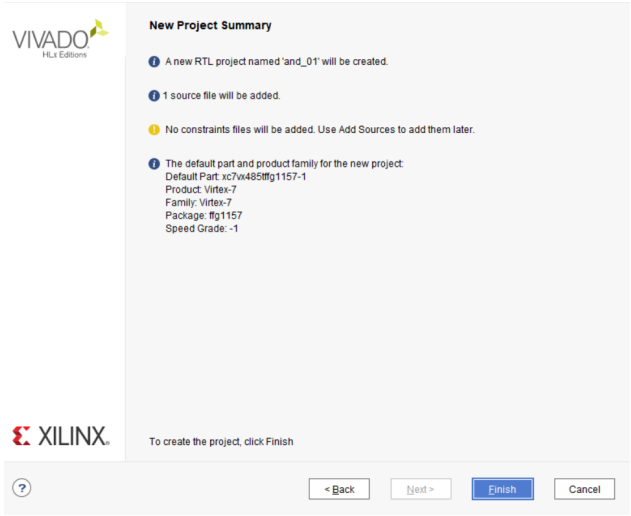
- In Add Sources window click *Create File*, then select file type as *VHDL* and name the file “not_01”, click OK. Make sure *VHDL* is selected as Target language and Simulator Language and click Next



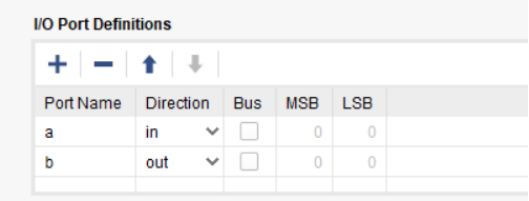
- Click next on the Add constraints window and in the Default Part window select the board available.



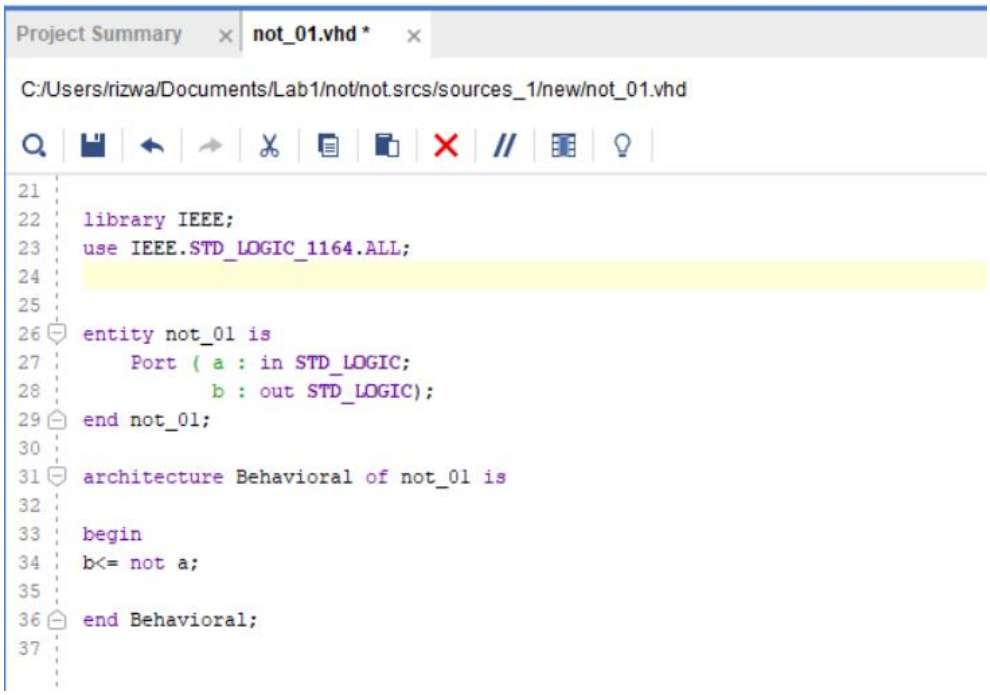
- Click Finish to create project.



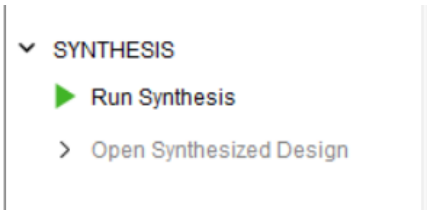
- Define module window appears, in the I/O Definitions tab of the window define the input and output ports for NOT gate and click OK.



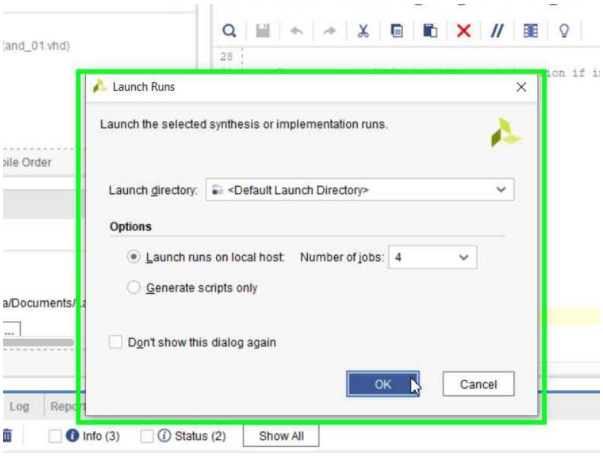
- In the not_01.vhd file add the following code to define (Behavioral) the gate and save it by pressing ctrl + s:



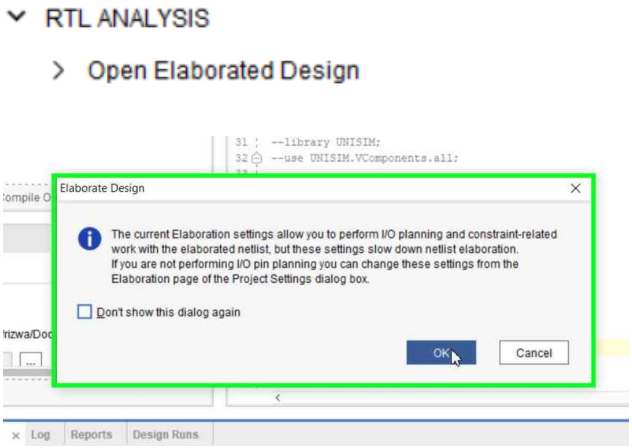
- To synthesize the design, click on Run Synthesis:



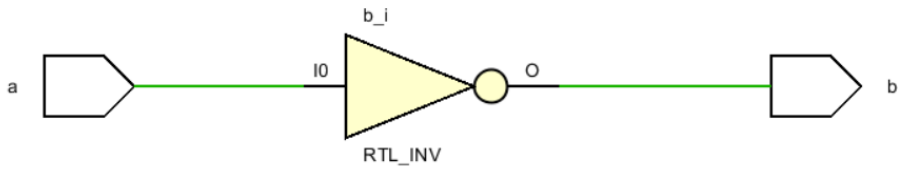
- Then click on OK in Launch Run window



- Close the synthesis completed dialogue, and then Click on Open Elaborated Design and Click OK



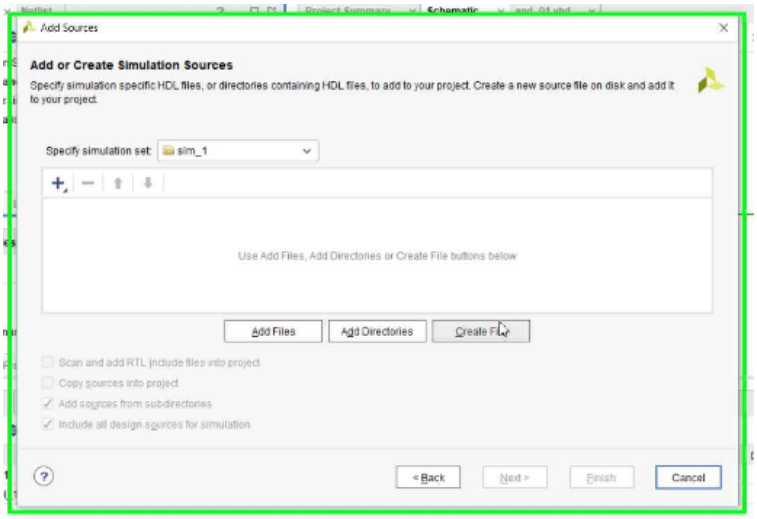
- This shows the schematic of the design



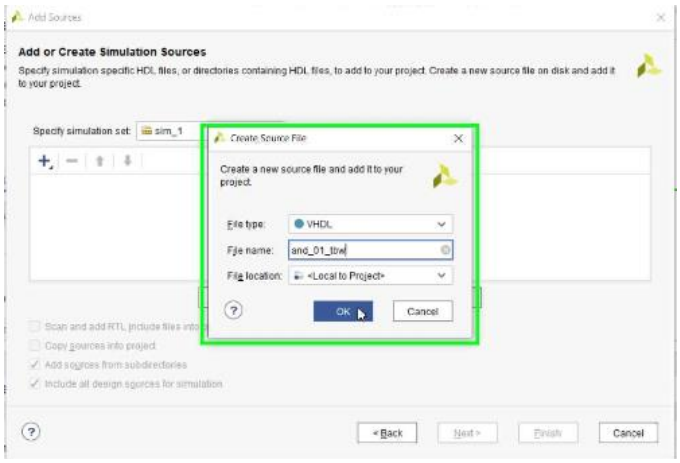
2) To Create a Test Bench:

- Click on Add Source and choose Add or Create Simulation Source, then click next.

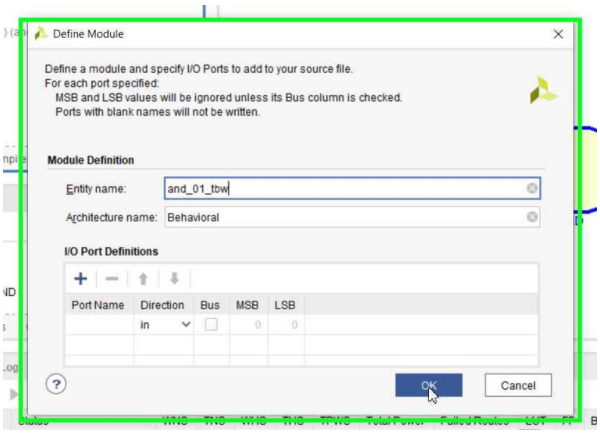
- In Add or Create Simulation Source Window click on Create File



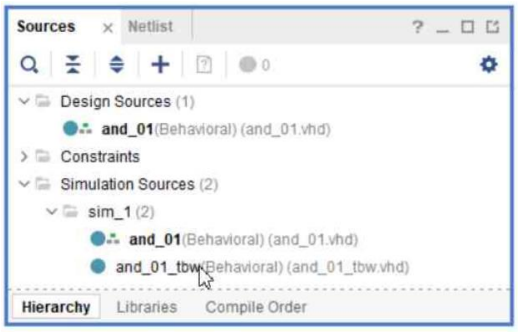
- Select File type as VHDL and name the file “not_01_tbw”. Then click OK and then Finish.



- In Define module window click OK and if the dialogue pops up click on YES.



- Under sources window, double click on “not_01_tbw” file.

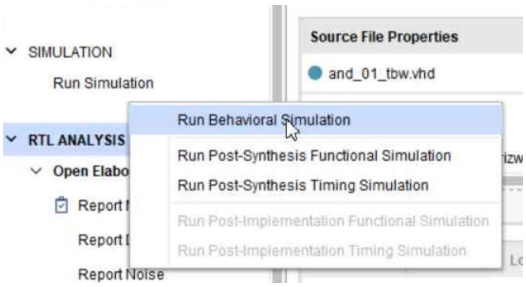


- Then in not_01_tbw.vhd file add the following code:

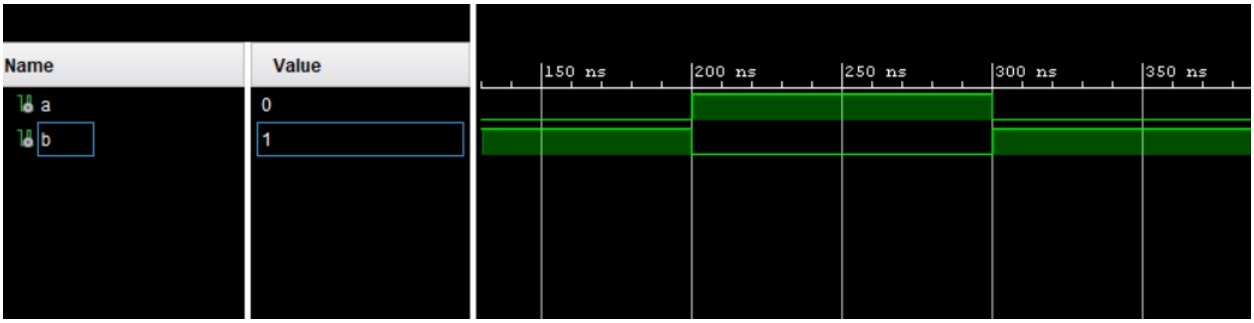
```
library IEEE;
use IEEE.std_logic_1164.ALL;
ENTITY not_01_tbw IS
END not_01_tbw;

ARCHITECTURE behavior OF not_01_tbw IS
    component not_01 IS
    PORT(
         a : IN  std_logic;
         b : OUT  std_logic
        );
    END COMPONENT;
   signal a : std_logic := '0';
   signal b : std_logic;
BEGIN
   uut: not_01 PORT MAP (
```

```
        a => a,
        b => b
      );

  stim_proc: process
  begin
    -- hold reset state for 100 ns.
    wait for 100 ns;
    a<='0';
    wait for 100 ns;
    a<='1';
    wait for 100 ns;
    a<='0';
    wait;
  end process;

END;
```

- Save the file and under Simulation option double click on Run Simulation and choose Run Behavioral Simulation



- Behavioral Simulation window appears showing the, waveform of NOT gate for different values of input.



EXPERIMENT 7

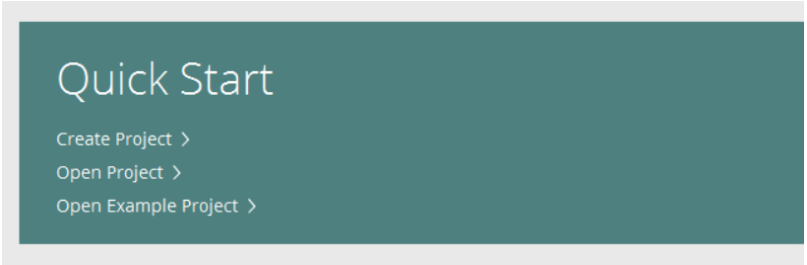
AIM: To design and simulate multi-bit AND Gate.

Apparatus Required: Vivado Design Suite.

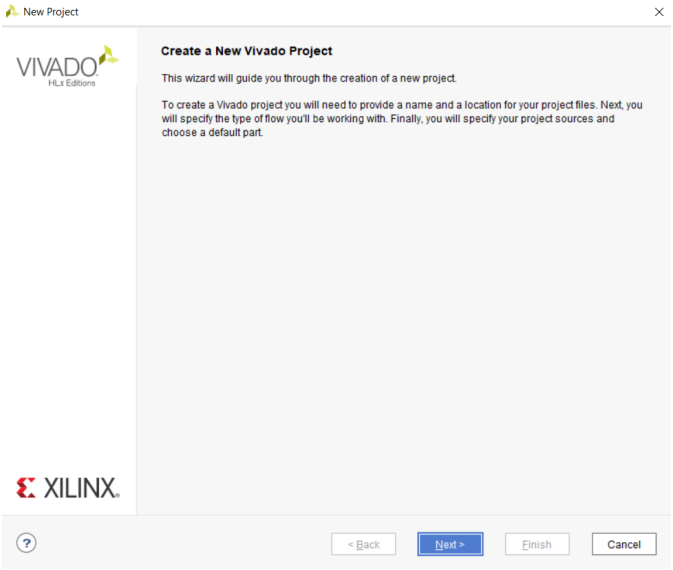
Procedure:

1) **To create a Multibit (8bit) AND gate:**

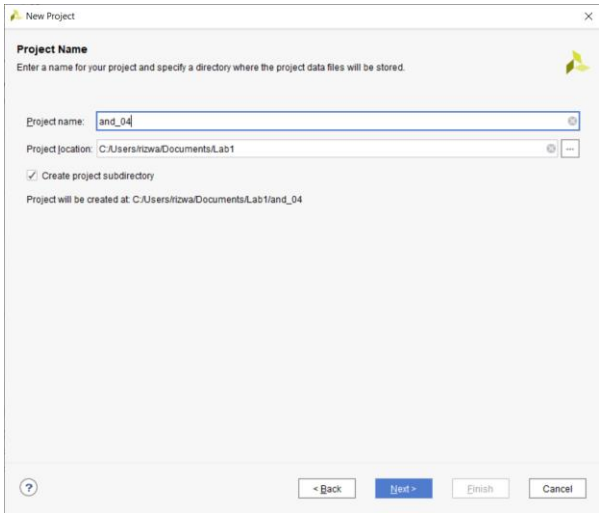
- Open *Vivado Design Suite*
- click on *Create Project* option.



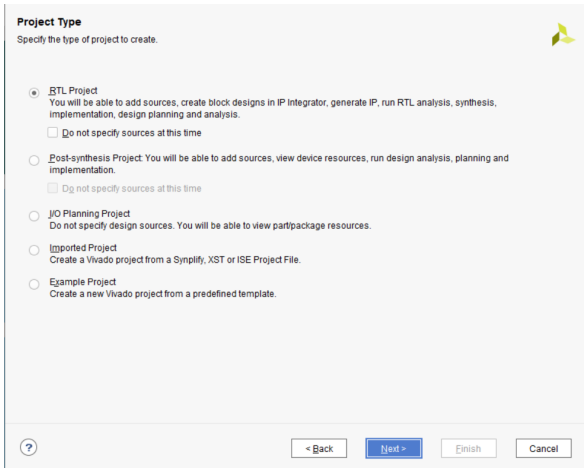
- Click next on Create Project window



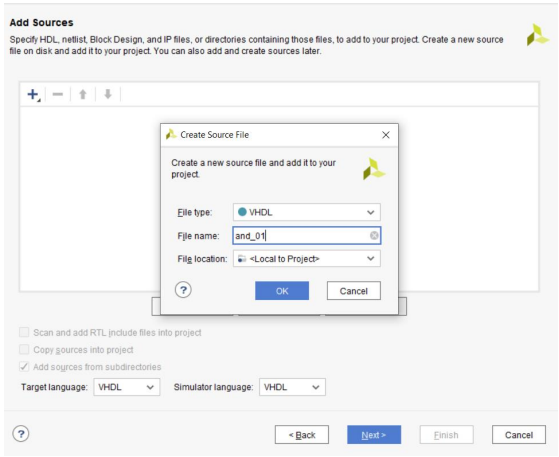
- Name the project “and_04” and select the location to save the project, then click next



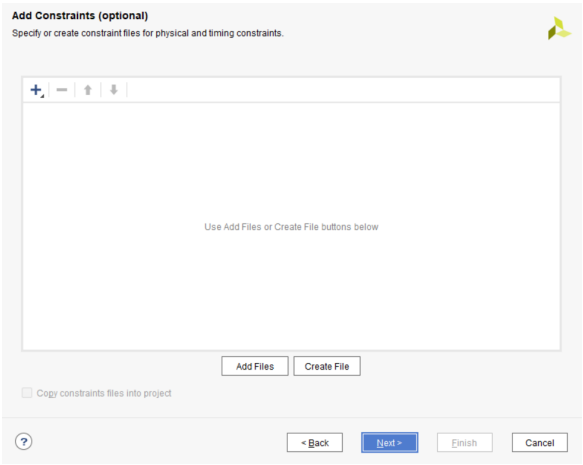
- In Project Type window select *RTL Project* and click next.



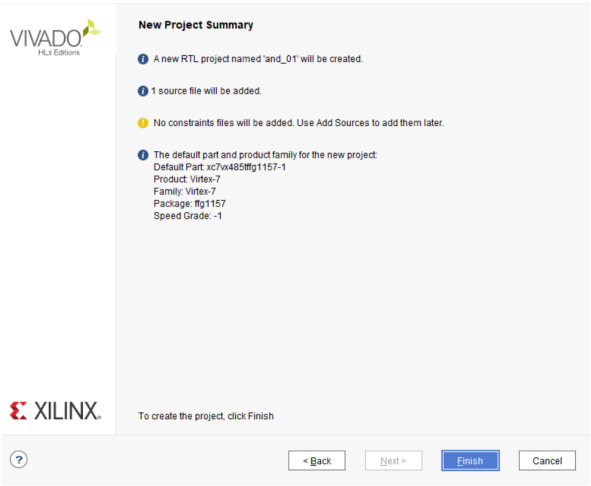
- In Add Sources window click *Create File*, then select file type as *VHDL* and name the file “and_04”, click OK. Make sure *VHDL* is selected as Target language and Simulator Language and click Next



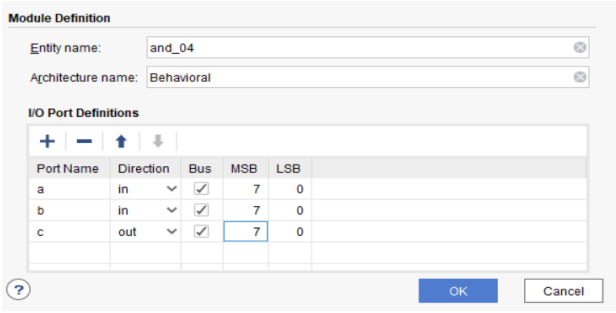
- Click next on the Add constraints window and in the Default Part window select the board available.



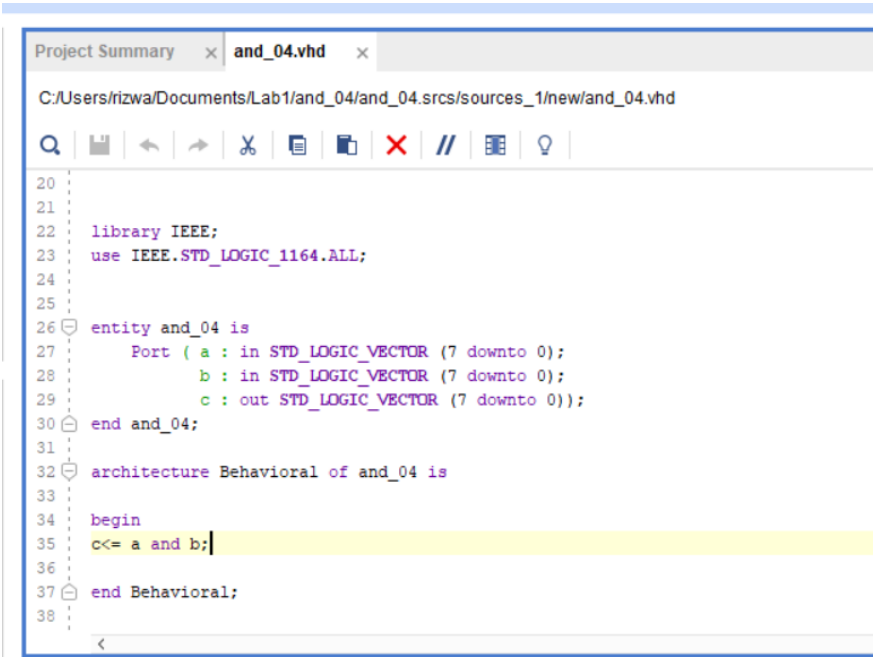
- Click Finish to create project.



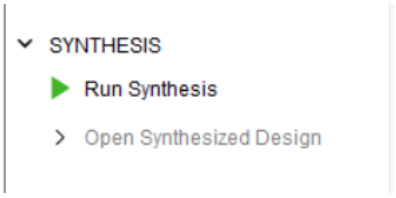
- Define module window appears, in the I/O Definitions tab of the window define the input and output ports for multibit AND gate and click OK. For multibit AND input and output ports will be bus. Here the input and output are of 8 bits therefore, MSB is 7 and LSB 0^{th} bit.



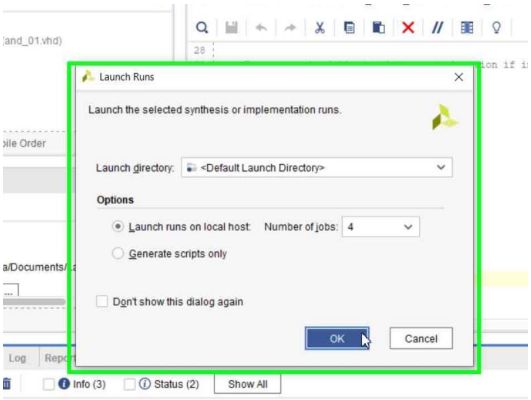
- In the and_04.vhd file add the following code to define (Behavioral) the gate and save it by pressing ctrl + s:



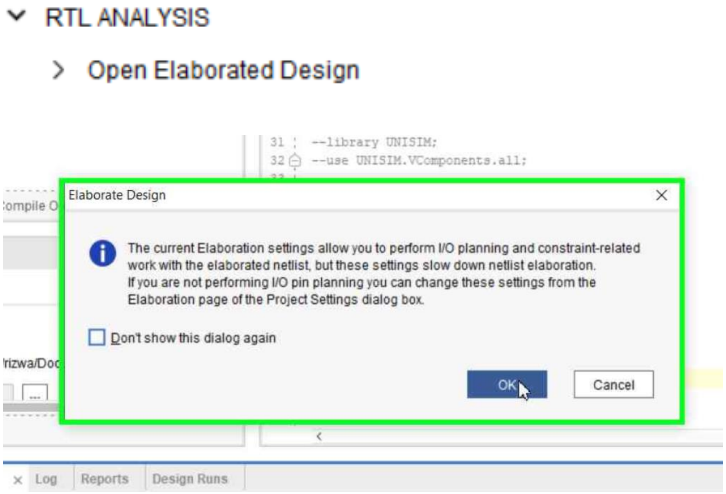
- To synthesize the design, click on Run Synthesis:



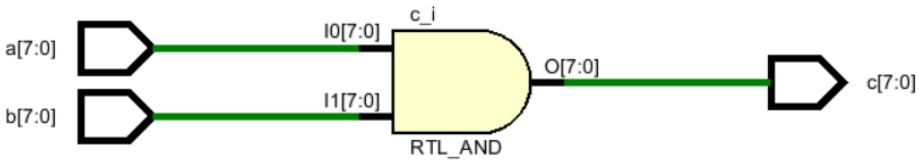
- Then click on OK in Launch Run window



- Close the synthesis completed dialogue, and then Click on Open Elaborated Design and Click OK

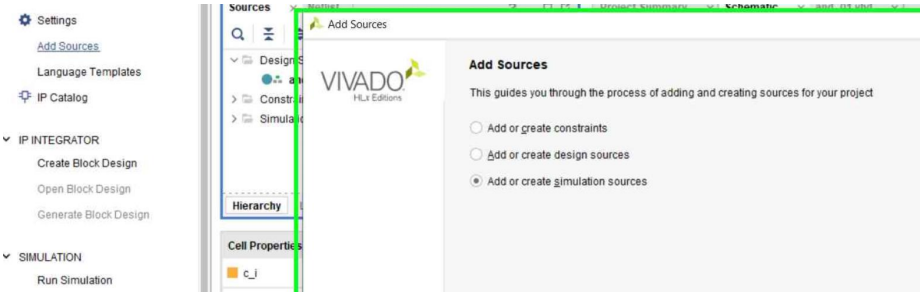


- This shows the schematic of the design

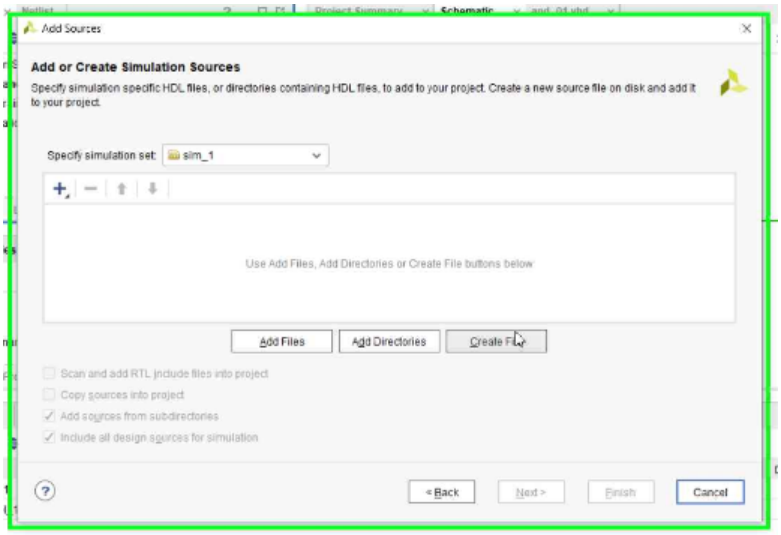


2) To Create a Test Bench:

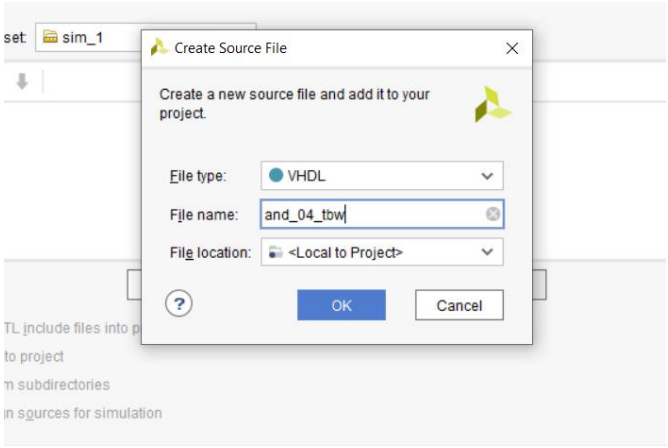
- Click on Add Source and choose Add or Create Simulation Source, then click next.



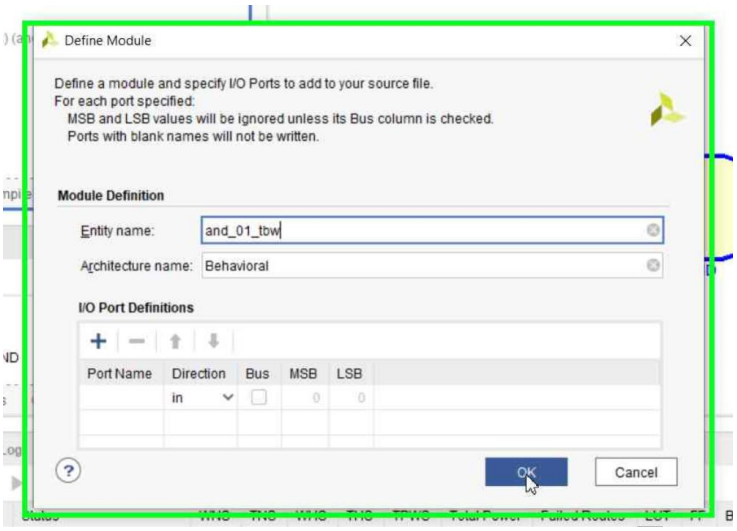
- In Add or Create Simulation Source Window click on Create File



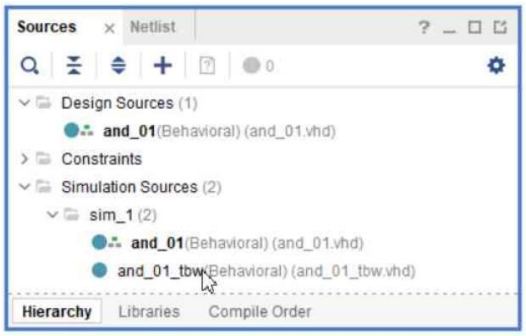
- Select File type as VHDL and name the file “and_04_tbw”. Then click OK and then Finish.



- In Define module window click OK and if the dialogue pops up click on YES.



- Under sources window, double click on “and_04_tbw” file.

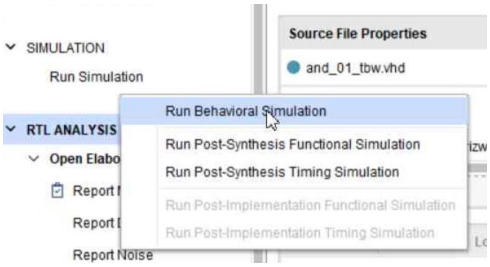


- Then in and_04_tbw.vhd file add the following code:

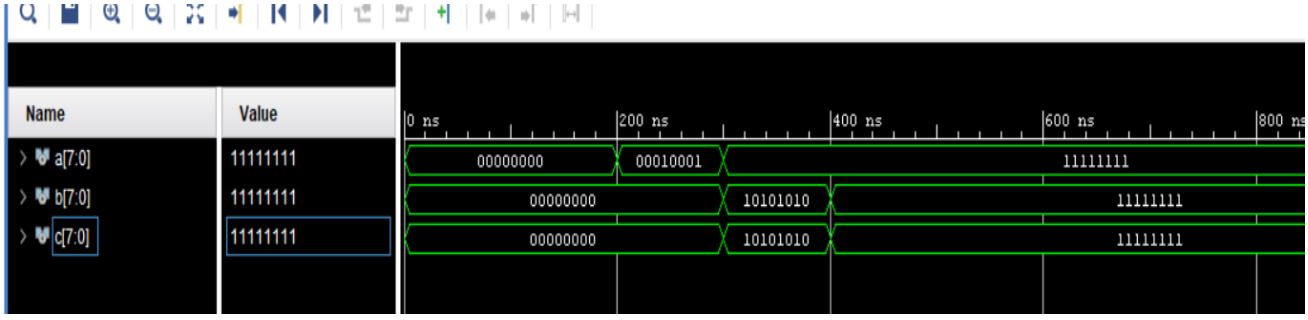
```
library IEEE;
use IEEE.std_logic_1164.ALL;
ENTITY and_03_tbw IS
END and_03_tbw;

ARCHITECTURE behavior OF and_03_tbw IS
   component and_03 IS
   PORT(
        a : IN  std_logic_vector(7 downto 0);
        b : IN  std_logic_vector (7 downto 0);
        c : OUT  std_logic_vector (7 downto 0)
       );
   END COMPONENT;
  --Inputs
```

```
    signal a : std_logic_vector(7 downto 0) := "00000000";
    signal b : std_logic_vector(7 downto 0) := "00000000";
          --Outputs
    signal c : std_logic_vector(7 downto 0);
  BEGIN
    uut: and_03 PORT MAP (
         a => a,
         b => b,
         c => c);
    stim_proc: process
    begin
      wait for 100 ns;
      a<="00000000";
      b<="00000000";
      wait for 100 ns;
      a<="00010001";
      b<="00000000";
      wait for 100 ns;
      a<="11111111";
      b<="10101010";
      wait for 100 ns;
      a<="11111111";
      b<="11111111";
      wait;
    end process;
  END;
```

- Save the file and under Simulation option double click on Run Simulation and choose Run Behavioral Simulation



- Behavioral Simulation window appears showing the, waveform of multibit AND gate for different values of input.



EXPERIMENT 8

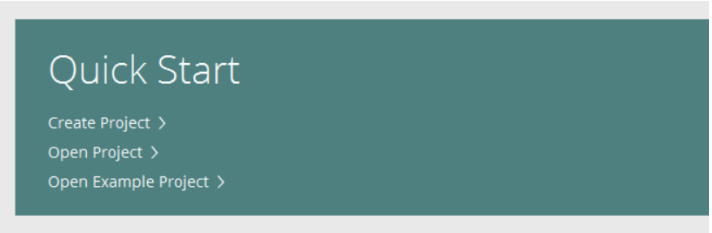
AIM: To design and simulate multi-bit OR Gate.

Apparatus Required: Vivado Design Suite.

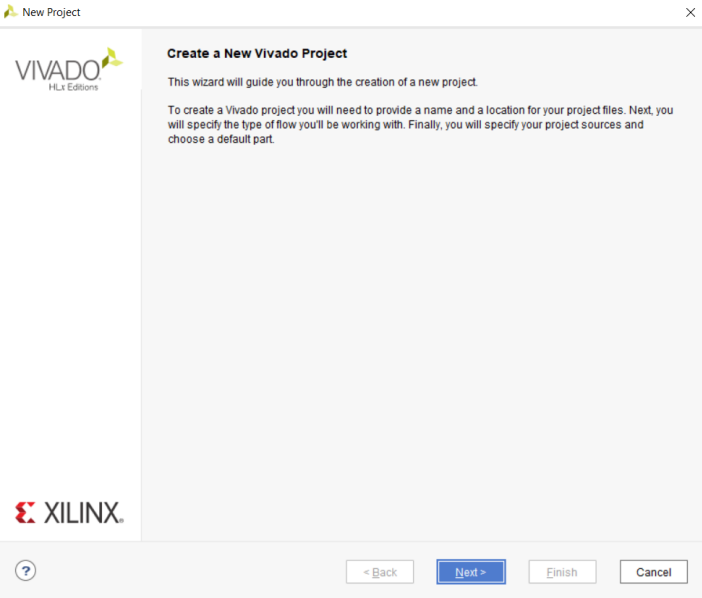
Procedure:

1) **To create a Multibit (8bit) OR gate:**

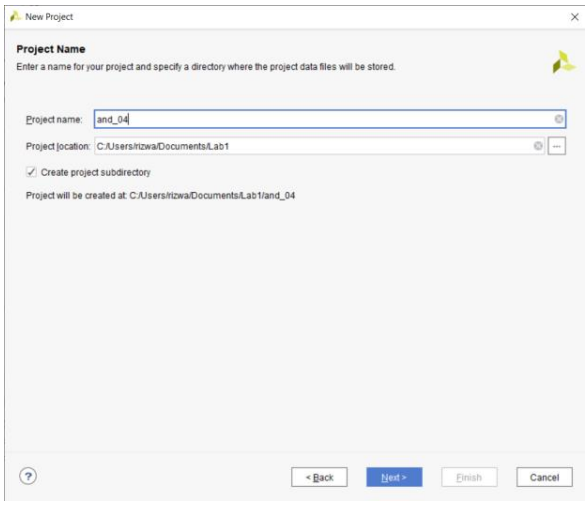
   - Open *Vivado Design Suite*
   - click on *Create Project* option.



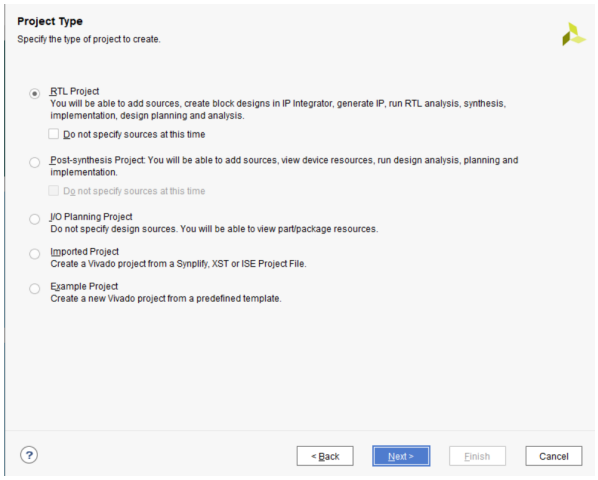
   - Click next on Create Project window



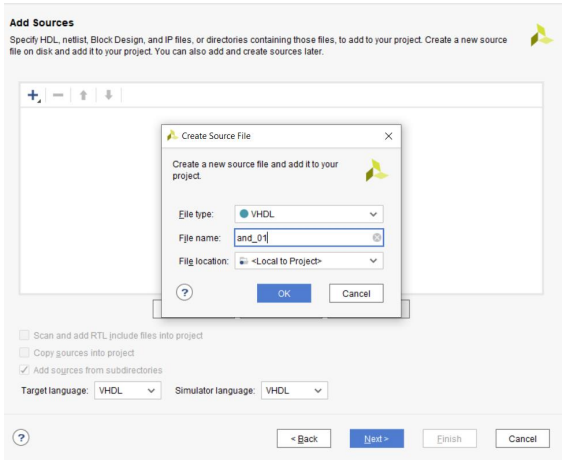
   - Name the project “or_04” and select the location to save the project, then click next



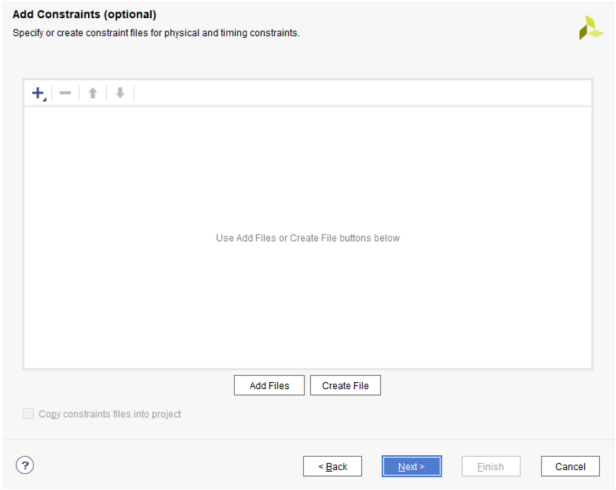
- In Project Type window select *RTL Project* and click next.



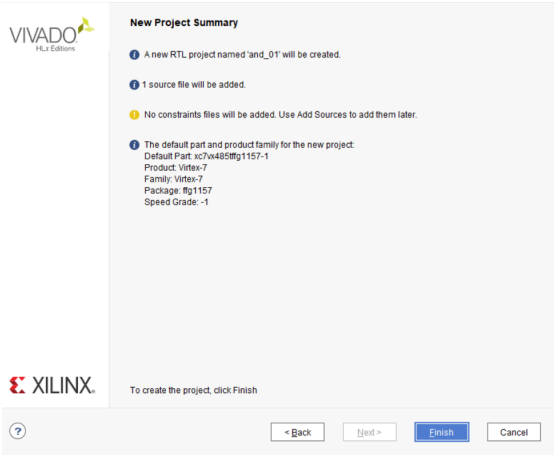
- In Add Sources window click *Create File*, then select file type as *VHDL* and name the file “or_04”, click OK. Make sure *VHDL* is selected as Target language and Simulator Language and click Next



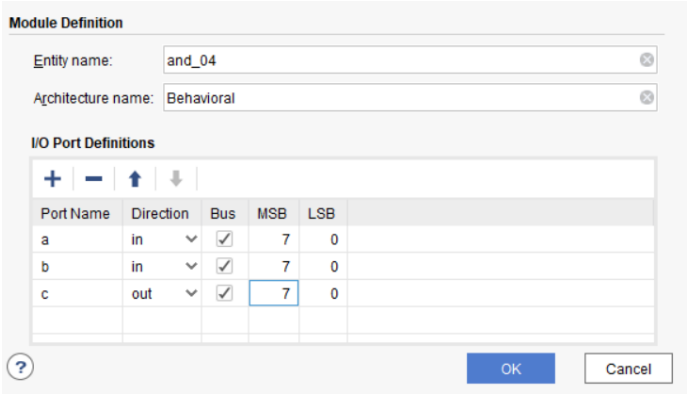
- Click next on the Add constraints window and in the Default Part window select the board available.



- Click Finish to create project.



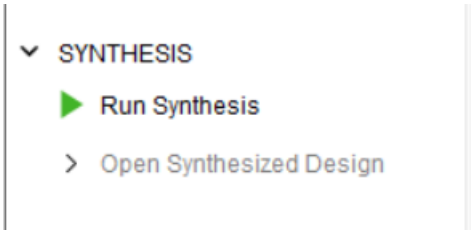
- Define module window appears, in the I/O Definitions tab of the window define the input and output ports for multibit OR gate and click OK. For multibit OR, input and output ports will be bus. Here the input and output is of 8 bits therefore, MSB is 7 and LSB 0^{th} bit.



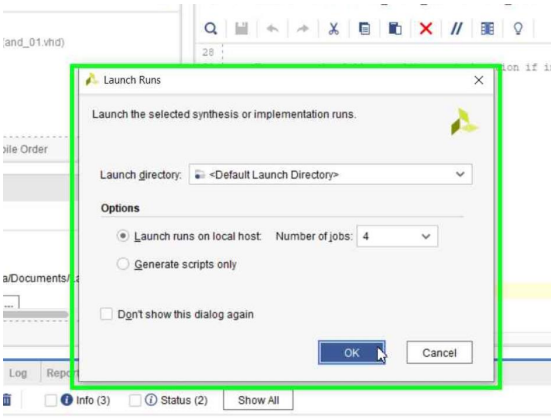
- In the or_04.vhd file add the following code to define (Behavioral) the gate and save it by pressing ctrl + s:

```
19  ----------------------------------------------------------------------------------
20
21
22  library IEEE;
23  use IEEE.STD_LOGIC_1164.ALL;
24
25  entity or_04 is
26      Port ( a : in STD_LOGIC_VECTOR (7 downto 0);
27             b : in STD_LOGIC_VECTOR (7 downto 0);
28             c : out STD_LOGIC_VECTOR (7 downto 0));
29  end or_04;
30
31  architecture Behavioral of or_04 is
32
33  begin
34
35  c <= a or b;
36  end Behavioral;
37
```

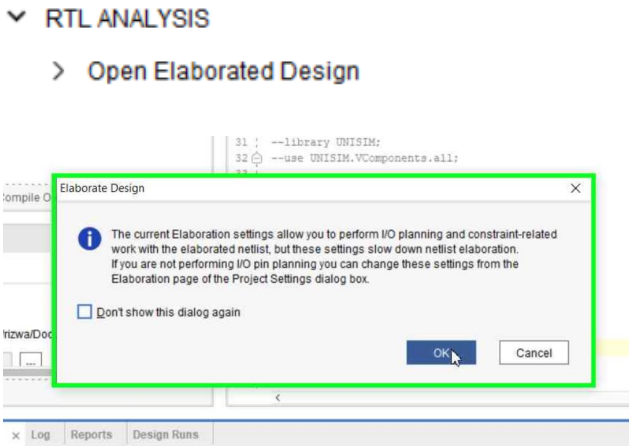
- To synthesize the design, click on Run Synthesis:



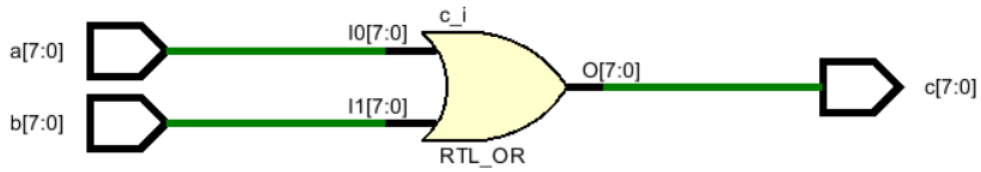
- Then click on OK in Launch Run window



- Close the synthesis completed dialogue, and then Click on Open Elaborated Design and Click OK

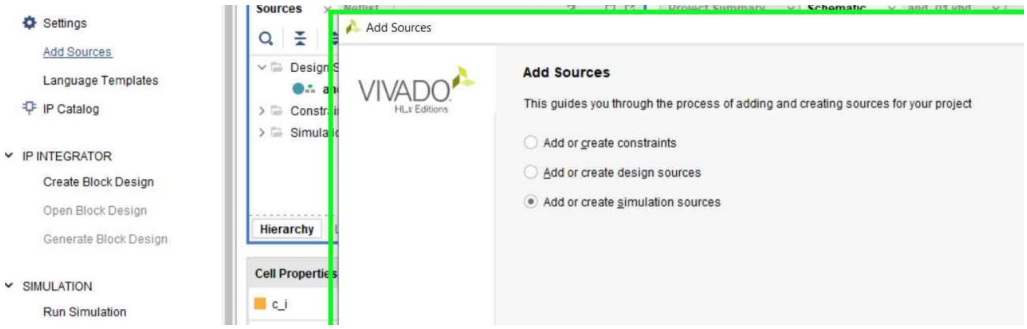


- This shows the schematic of the design

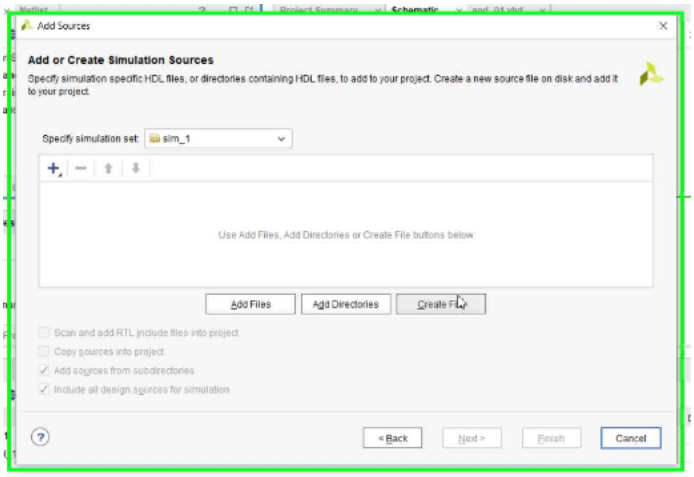


2) To Create a Test Bench:

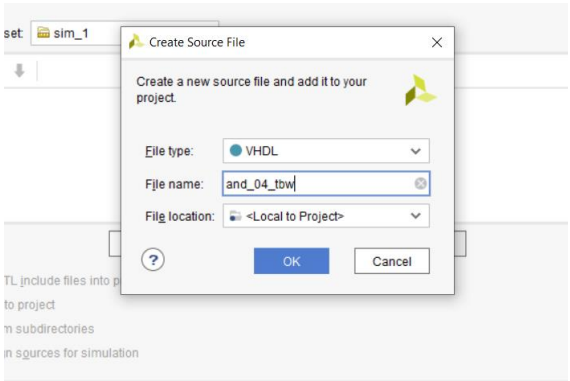
- Click on Add Source and choose Add or Create Simulation Source, then click next.



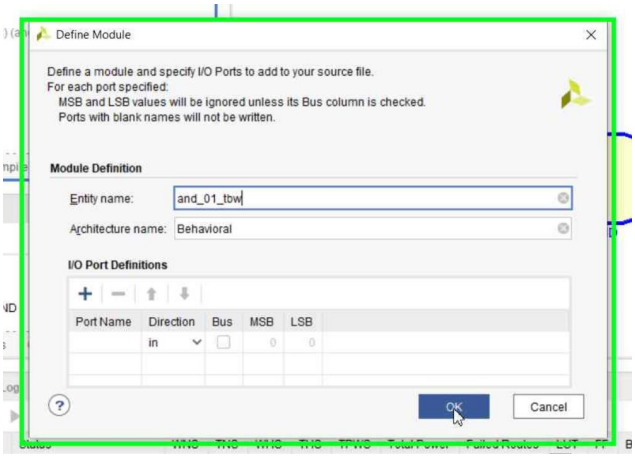
- In Add or Create Simulation Source Window click on Create File



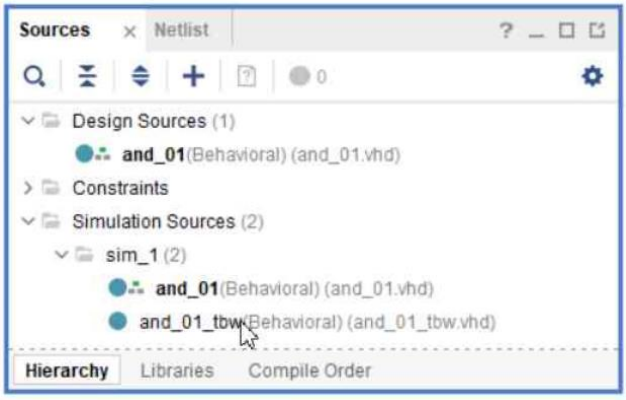
- Select File type as VHDL and name the file “or_04_tbw”. Then click OK and then Finish.



- In Define module window click OK and if the dialogue pops up click on YES.



- Under sources window, double click on “or_04_tbw” file.

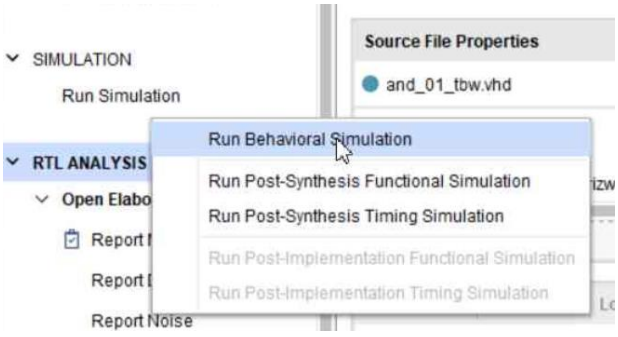


- Then in or_04_tbw.vhd file add the following code:

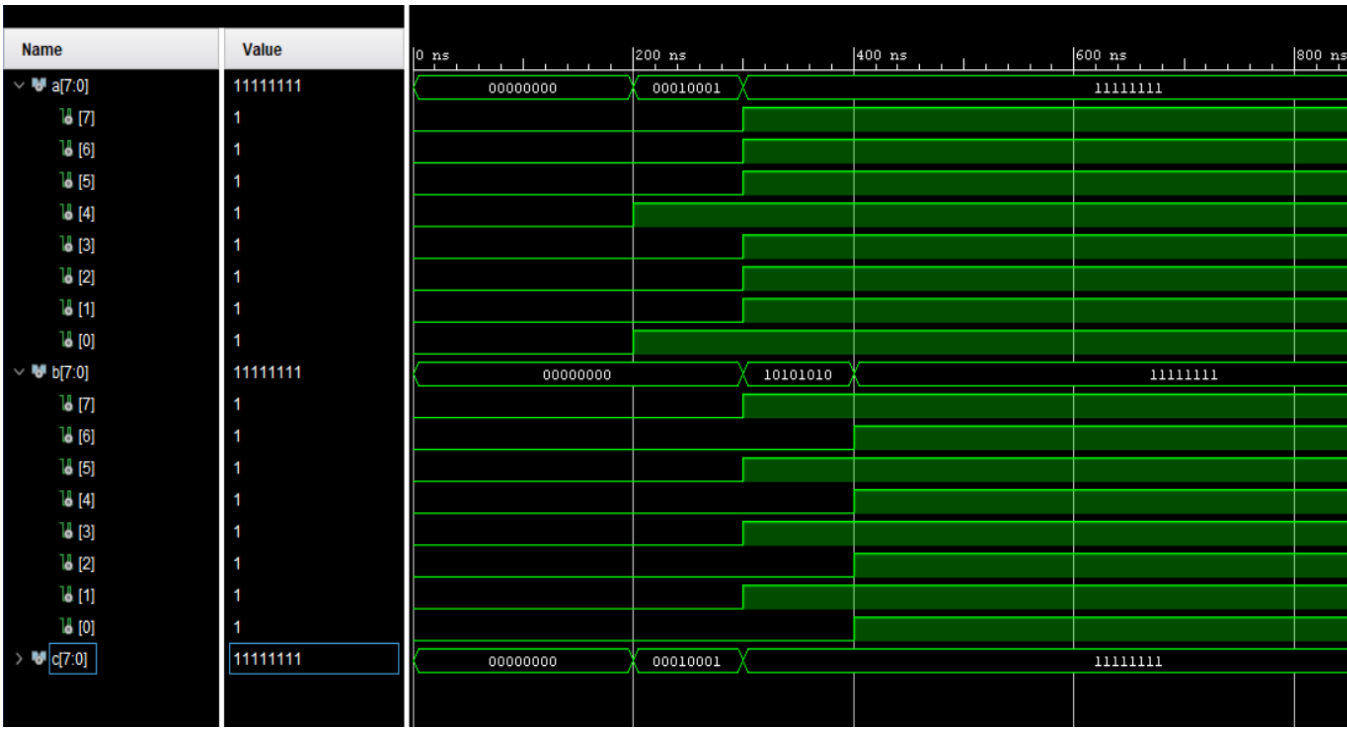
```
library IEEE;
use IEEE.std_logic_1164.ALL;
ENTITY or_04_tbw IS
END or_04_tbw;

ARCHITECTURE behavior OF or_04_tbw IS
   component or_04 IS
   PORT(
        a : IN  std_logic_vector(7 downto 0);
        b : IN  std_logic_vector (7 downto 0);
        c : OUT  std_logic_vector (7 downto 0)
       );
   END COMPONENT;
```

```
  --Inputs
  signal a : std_logic_vector(7 downto 0) := "00000000";
  signal b : std_logic_vector(7 downto 0) := "00000000";

        --Outputs
  signal c : std_logic_vector(7 downto 0);
BEGIN
  uut: or_04 PORT MAP (
       a => a,
       b => b,
       c => c);
  stim_proc: process
  begin
    -- hold reset state for 100 ns.
    wait for 100 ns;
    a<="00000000";
    b<="00000000";
    wait for 100 ns;
    a<="00010001";
    b<="00000000";
    wait for 100 ns;
    a<="11111111";
    b<="10101010";
    wait for 100 ns;
    a<="11111111";
    b<="11111111";
    wait;
  end process;
END;
```

- Save the file and under Simulation option double click on Run Simulation and choose Run Behavioral Simulation



- Behavioral Simulation window appears showing the, waveform of multibit OR gate for different values of input.



EXPERIMENT 9

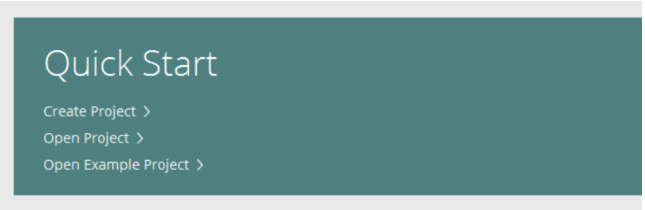
AIM: To design and simulate multi-bit NAND Gate.

Apparatus Required: Vivado Design Suite.

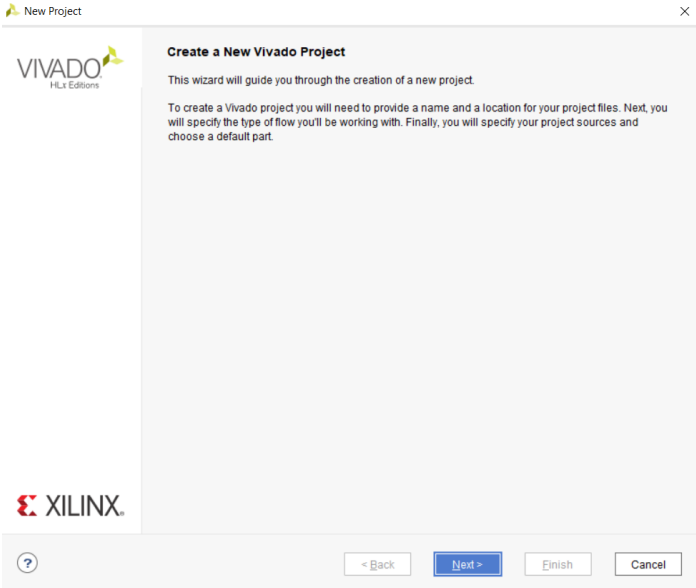
Procedure:

1) **To create a Multibit (8bit) NAND gate:**

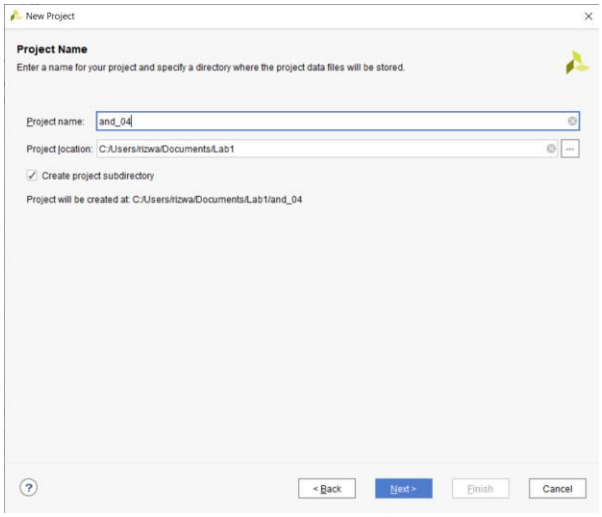
- Open *Vivado Design Suite*
- click on *Create Project* option.



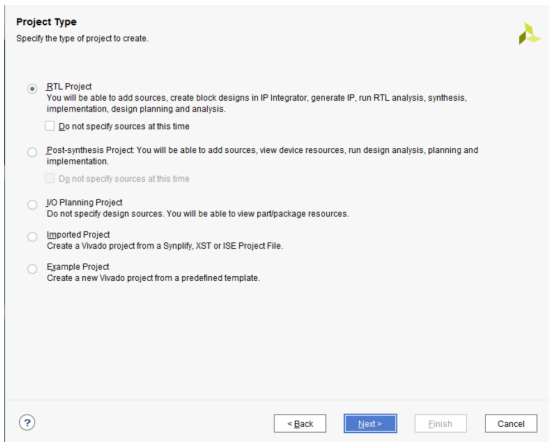
- Click next on Create Project window



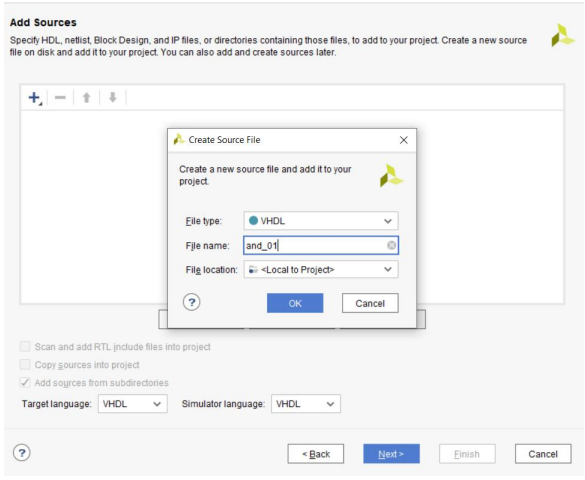
- Name the project “nand_04” and select the location to save the project, then click next



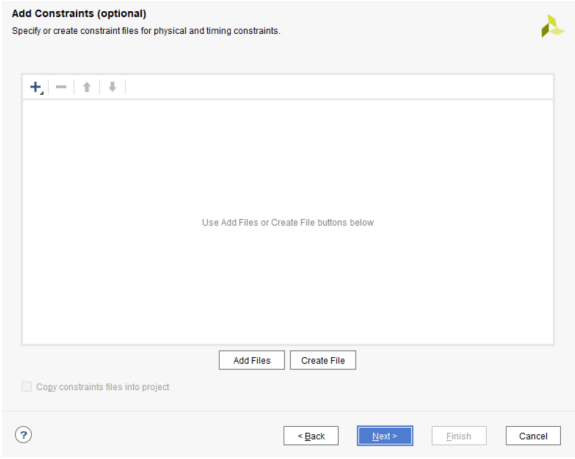
- In Project Type window select *RTL Project* and click next.



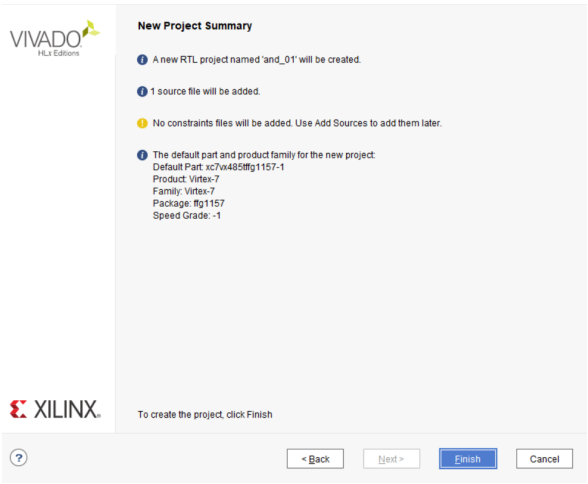
- In Add Sources window click *Create File*, then select file type as *VHDL* and name the file “nand_04”, click OK. Make sure *VHDL* is selected as Target language and Simulator Language and click Next



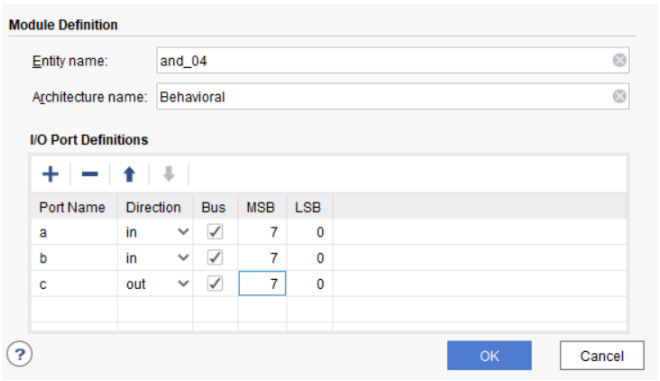
- Click next on the Add constraints window and in the Default Part window select the board available.



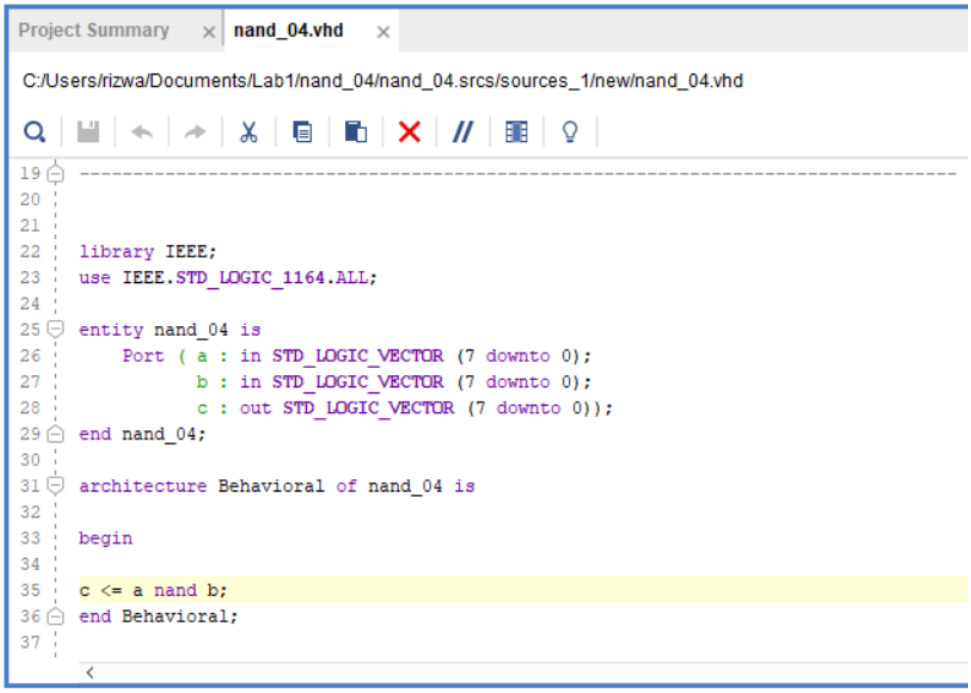
- Click Finish to create project.



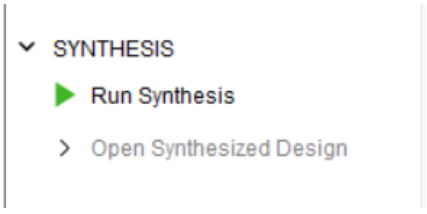
- Define module window appears, in the I/O Definitions tab of the window define the input and output ports for multibit NAND gate and click OK. For multibit NAND, input and output ports will be bus. Here the input and output is of 8 bits therefore, MSB is 7 and LSB 0^{th} bit.



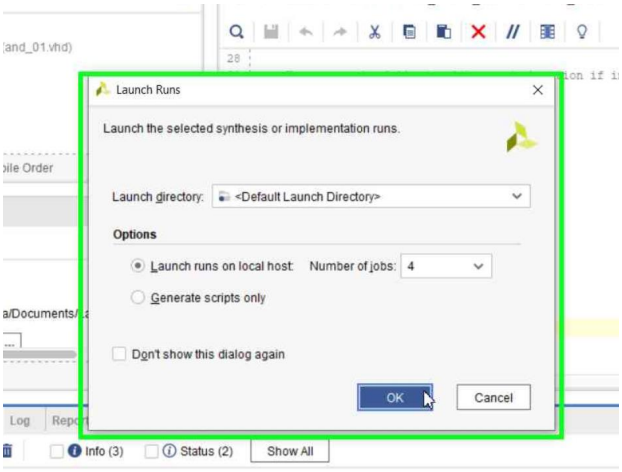
- In the nand_04.vhd file add the following code to define (Behavioral) the gate and save it by pressing ctrl + s:



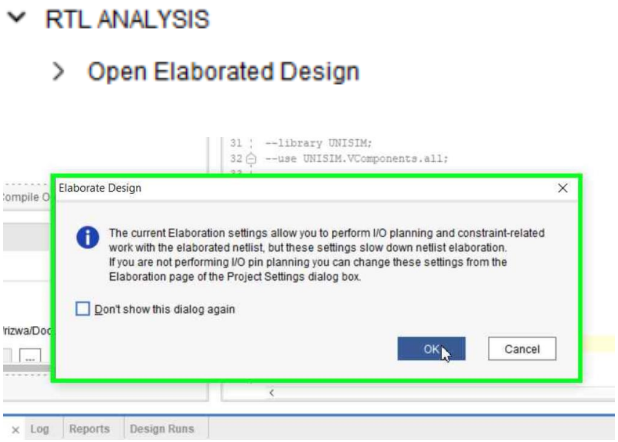
- To synthesize the design, click on Run Synthesis:



- Then click on OK in Launch Run window



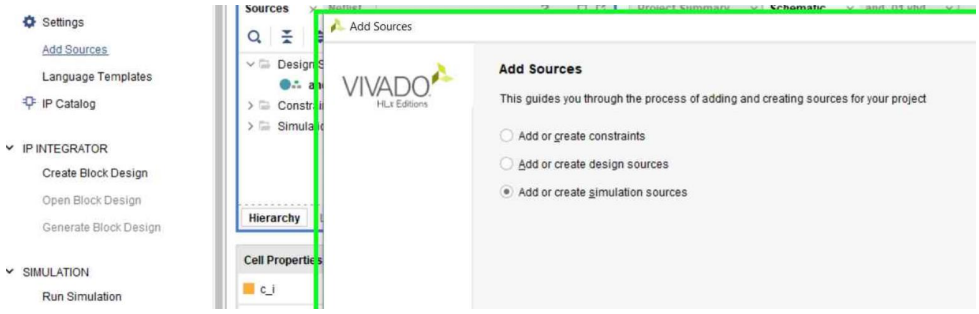
- Close the synthesis completed dialogue, and then Click on Open Elaborated Design and Click OK



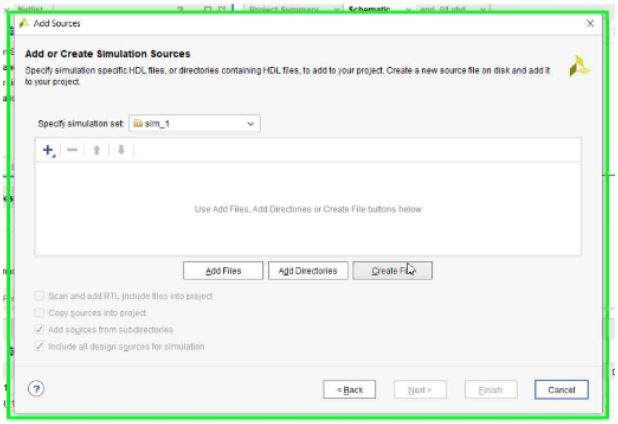
- This shows the schematic of the design

2) To Create a Test Bench:

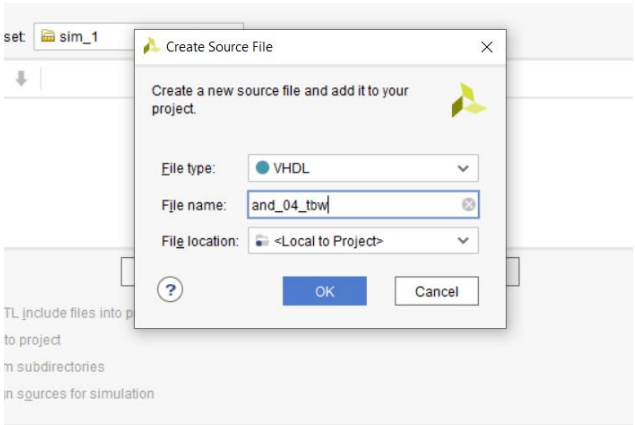
- Click on Add Source and choose Add or Create Simulation Source, then click next.



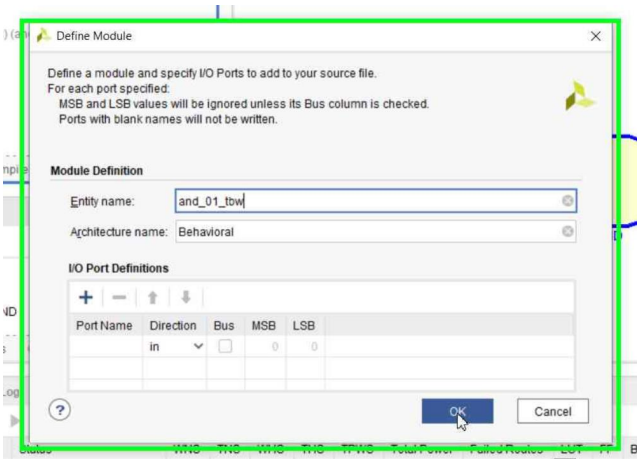
- In Add or Create Simulation Source Window click on Create File



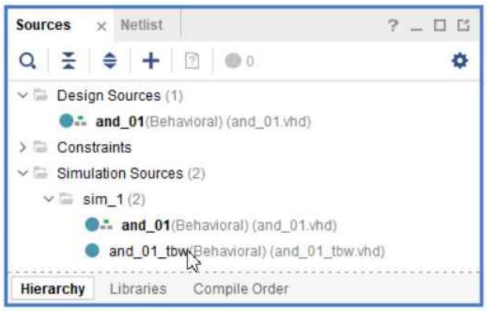
- Select File type as VHDL and name the file “nand_04_tbw”. Then click OK and then Finish.



- In Define module window click OK and if the dialogue pops up click on YES.



- Under sources window, double click on “nand_04_tbw” file.

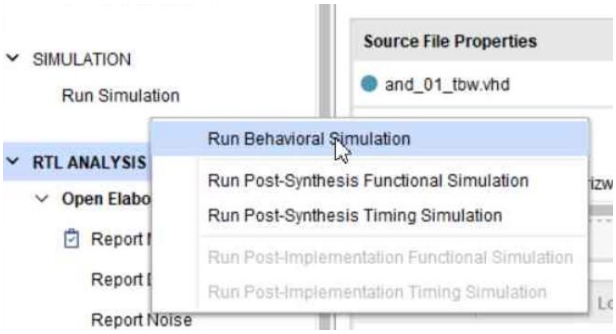


- Then in nand_04_tbw.vhd file add the following code:

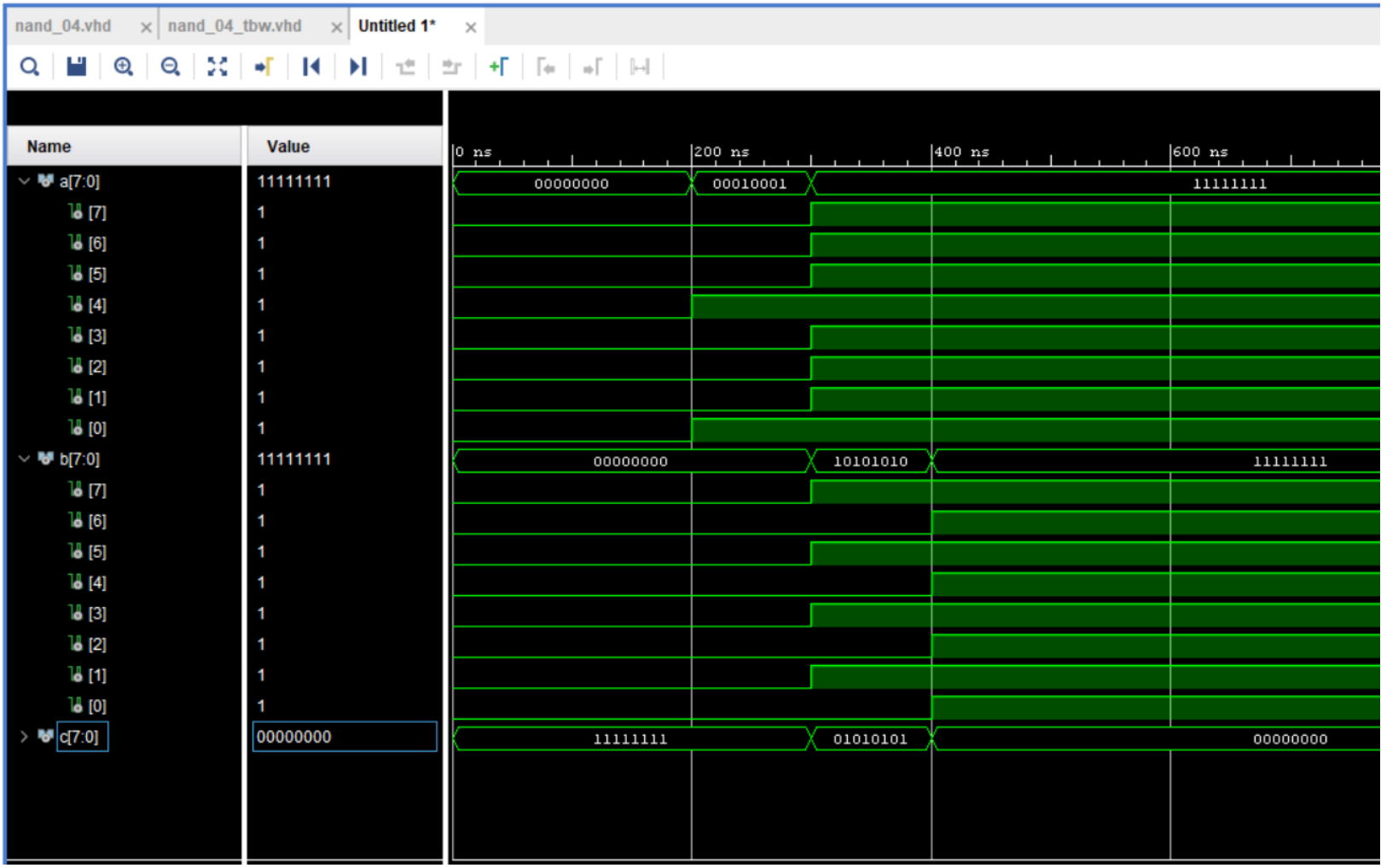
```
library IEEE;
use IEEE.std_logic_1164.ALL;
ENTITY nand_04_tbw IS
END nand_04_tbw;

ARCHITECTURE behavior OF nand_04_tbw IS
   component nand_04 IS
   PORT(
        a : IN  std_logic_vector(7 downto 0);
        b : IN  std_logic_vector (7 downto 0);
        c : OUT  std_logic_vector (7 downto 0)
       );
   END COMPONENT;
  signal a : std_logic_vector(7 downto 0) := "00000000";
  signal b : std_logic_vector(7 downto 0) := "00000000";

          --Outputs
  signal c : std_logic_vector(7 downto 0);
```

```
BEGIN

        -- Instantiate the Unit Under Test (UUT)
  uut: nand_04 PORT MAP (
        a => a,
        b => b,
        c => c);
  stim_proc: process
  begin
    -- hold reset state for 100 ns.
    wait for 100 ns;
    a<="00000000";
    b<="00000000";
    wait for 100 ns;
    a<="00010001";
    b<="00000000";
    wait for 100 ns;
    a<="11111111";
    b<="10101010";
    wait for 100 ns;
    a<="11111111";
    b<="11111111";
    wait;
  end process;

END;
```

- Save the file and under Simulation option double click on Run Simulation and choose Run Behavioral Simulation



- Behavioral Simulation window appears showing the, waveform of multibit NAND gate for different values of input.



EXPERIMENT 10

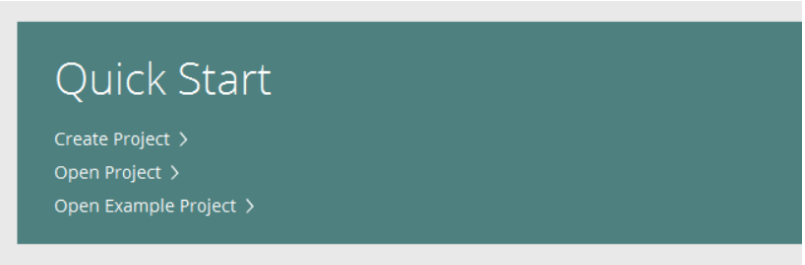
AIM: To design and simulate multi-bit NOR Gate.

Apparatus Required: Vivado Design Suite.

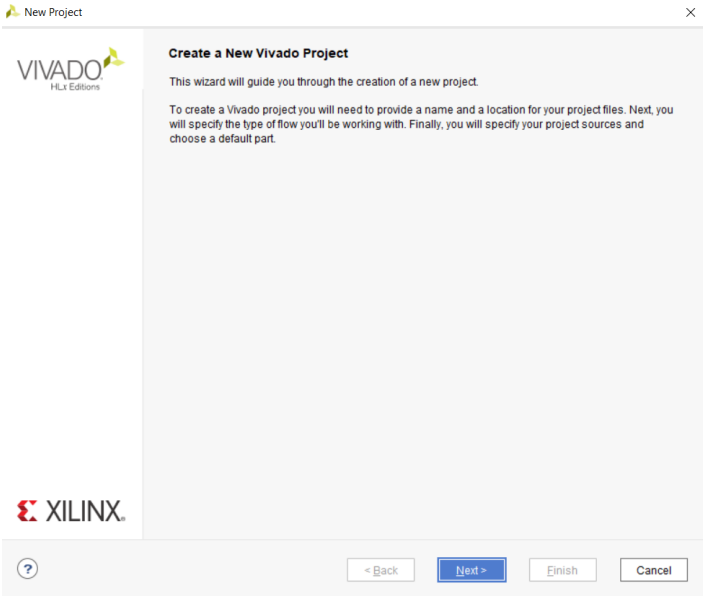
Procedure:

1) **To create a Multibit (8bit) NOR gate:**

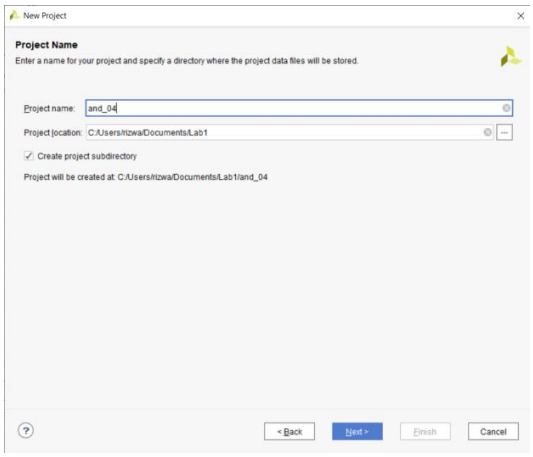
   - Open *Vivado Design Suite*
   - click on *Create Project* option.



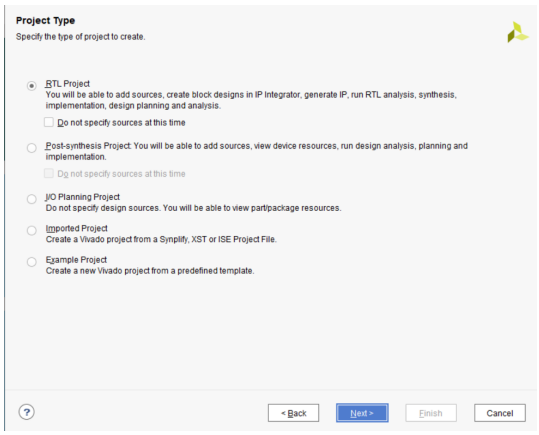
   - Click next on Create Project window



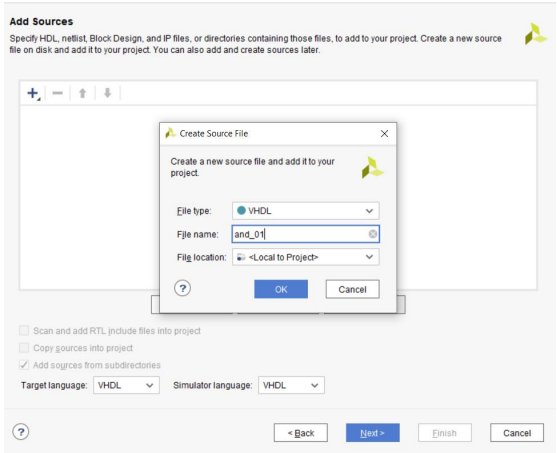
   - Name the project “nor_04” and select the location to save the project, then click next



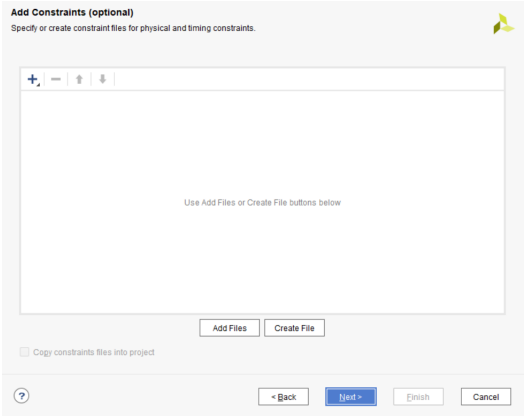
- In Project Type window select *RTL Project* and click next.



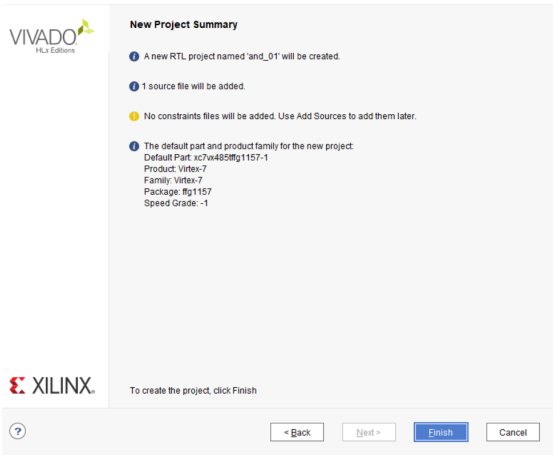
- In Add Sources window click *Create File*, then select file type as *VHDL* and name the file “nor_04”, click OK. Make sure *VHDL* is selected as Target language and Simulator Language and click Next



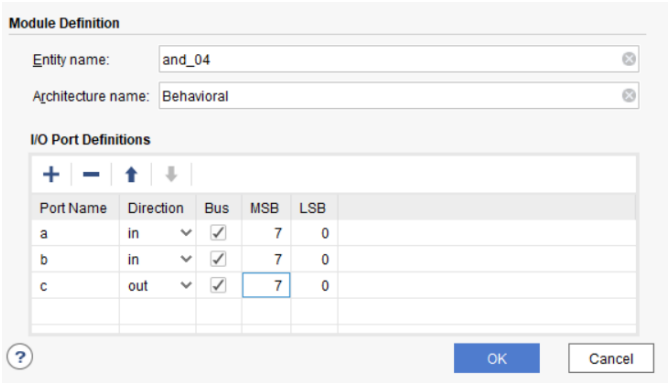
- Click next on the Add constraints window and in the Default Part window select the board available.



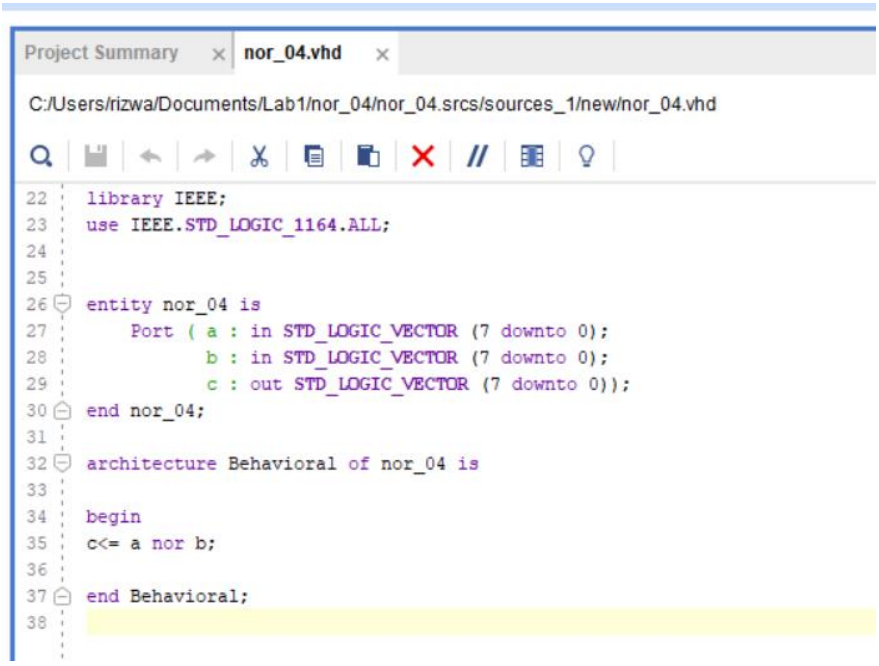
- Click Finish to create project.



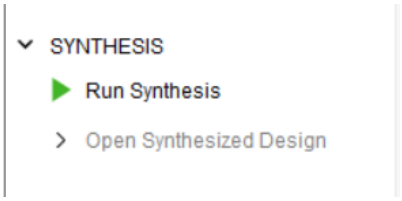
- Define module window appears, in the I/O Definitions tab of the window define the input and output ports for multibit NOR gate and click OK. For multibit NOR, input and output ports will be bus. Here the input and output is of 8 bits therefore, MSB is 7 and LSB 0th bit.



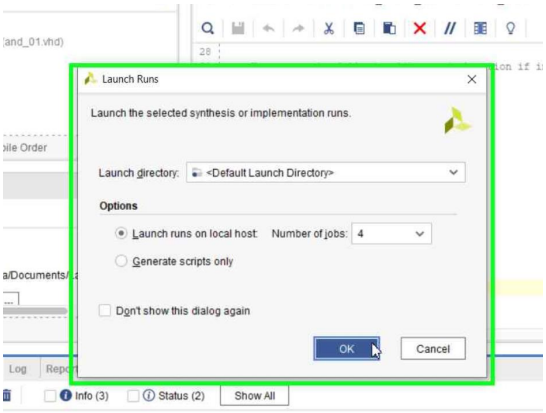
- In the nor_04.vhd file add the following code to define (Behavioral) the gate and save it by pressing ctrl + s:



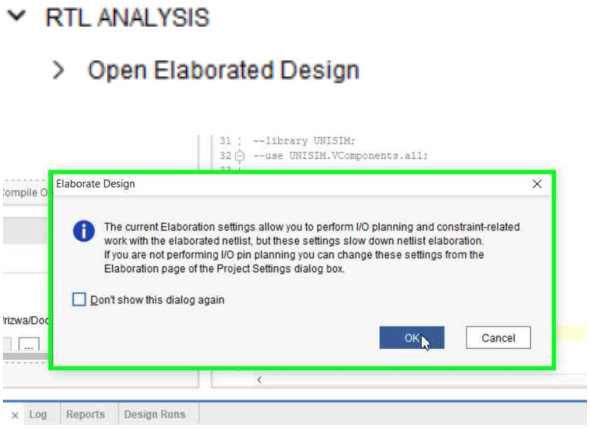
- To synthesize the design, click on Run Synthesis:



- Then click on OK in Launch Run window



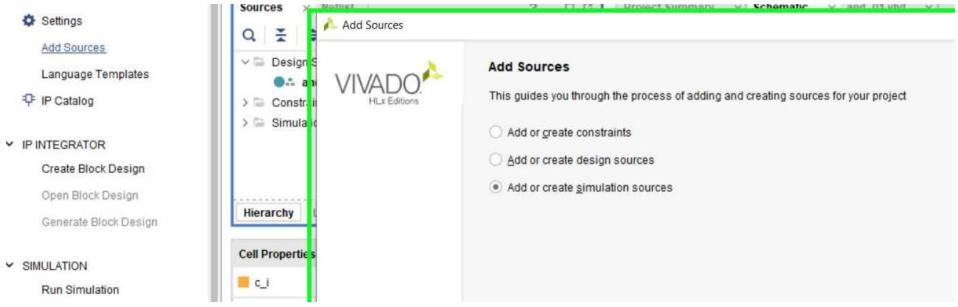
- Close the synthesis completed dialogue, and then Click on Open Elaborated Design and Click OK



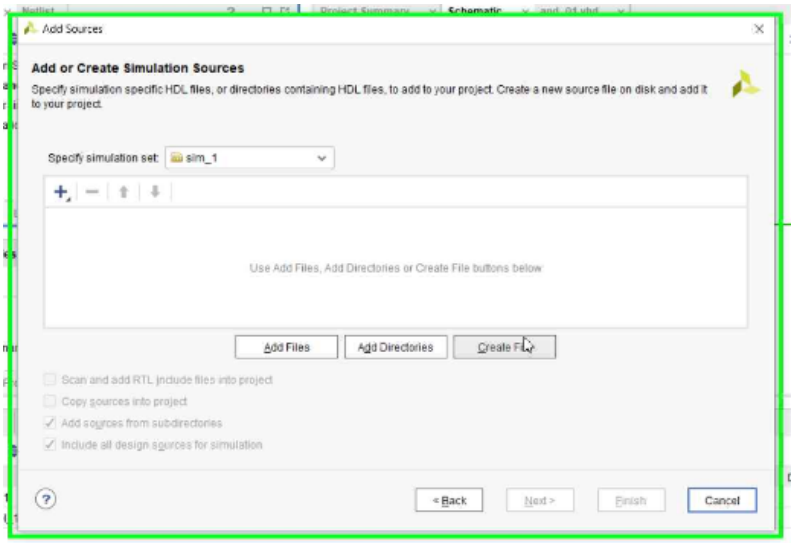
- This shows the schematic of the design

2) To Create a Test Bench:

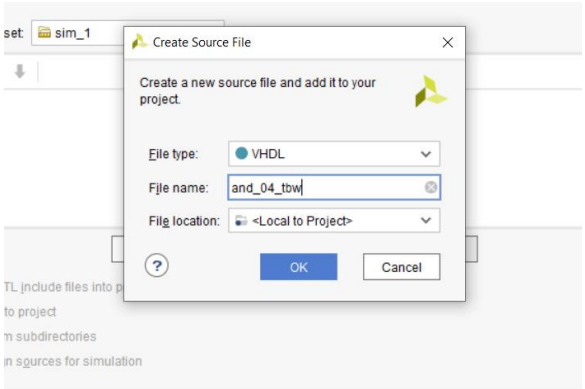
- Click on Add Source and choose Add or Create Simulation Source, then click next.



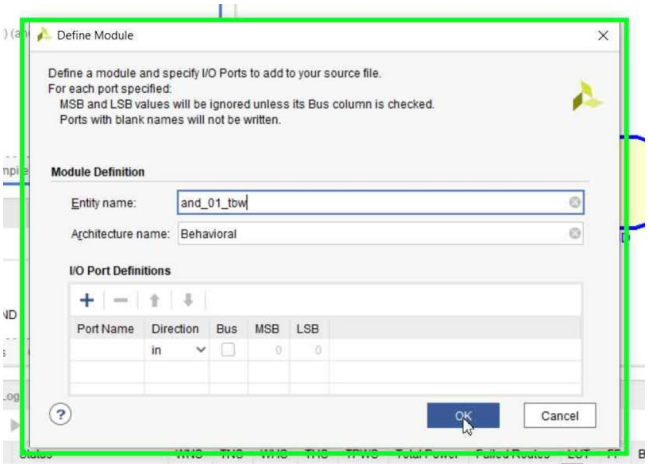
- In Add or Create Simulation Source Window click on Create File



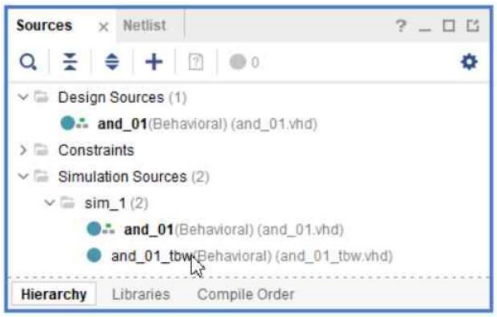
- Select File type as VHDL and name the file “nor_04_tbw”. Then click OK and then Finish.



- In Define module window click OK and if the dialogue pops up click on YES.



- Under sources window, double click on “nor_04_tbw” file.

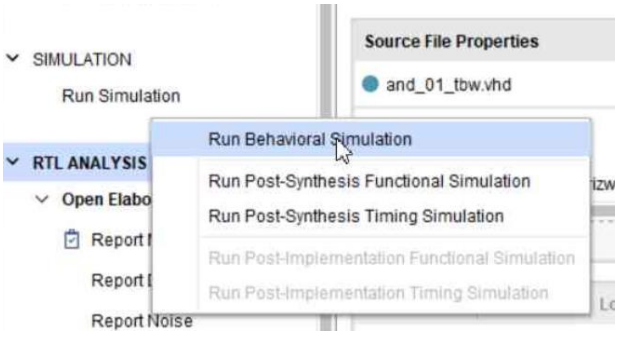


- Then in nor_04_tbw.vhd file add the following code:

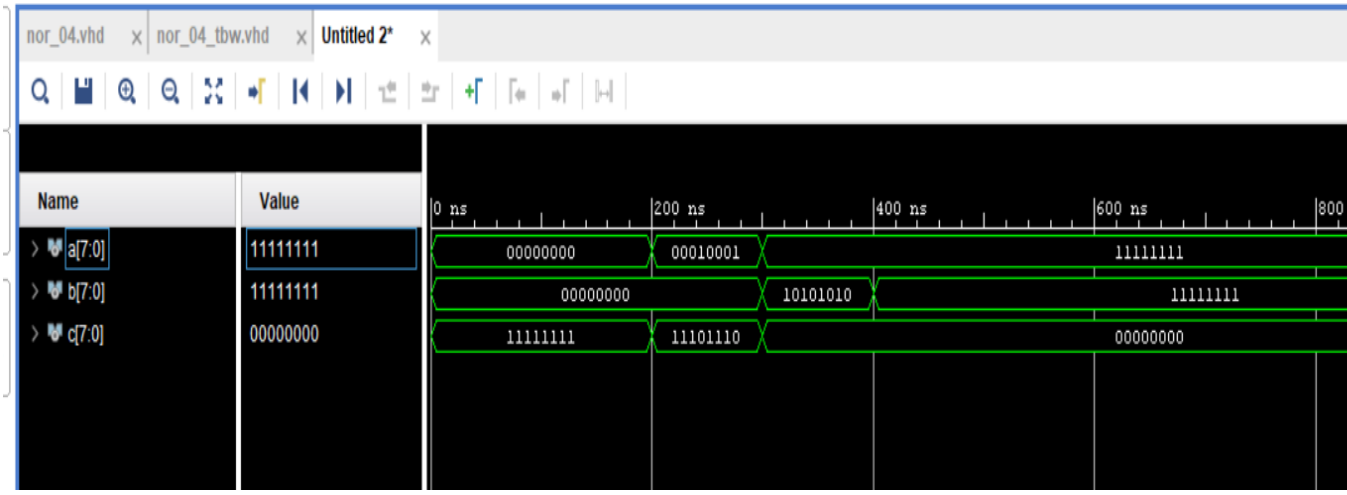
```
library IEEE;
use IEEE.std_logic_1164.ALL;

ENTITY nor_04_tbw IS
END nor_04_tbw;
ARCHITECTURE behavior OF nor_04_tbw IS
   component nor_04 IS
   PORT(
        a : IN  std_logic_vector(7 downto 0);
        b : IN  std_logic_vector (7 downto 0);
        c : OUT  std_logic_vector (7 downto 0)
       );
   END COMPONENT;

  --Inputs
  signal a : std_logic_vector(7 downto 0) := "00000000";
```

```
  signal b : std_logic_vector(7 downto 0) := "00000000";

        --Outputs
  signal c : std_logic_vector(7 downto 0);

BEGIN

  uut: nor_04 PORT MAP (
       a => a,
       b => b,
       c => c);
  stim_proc: process
  begin
    -- hold reset state for 100 ns.
    wait for 100 ns;
    a<="00000000";
    b<="00000000";
    wait for 100 ns;
    a<="00010001";
    b<="00000000";
    wait for 100 ns;
    a<="11111111";
    b<="10101010";
    wait for 100 ns;
    a<="11111111";
    b<="11111111";
    wait;
  end process;

END;
```

- Save the file and under Simulation option double click on Run Simulation and choose Run Behavioral Simulation



- Behavioral Simulation window appears showing the, waveform of multibit NOR gate for different values of input.



EXPERIMENT 11

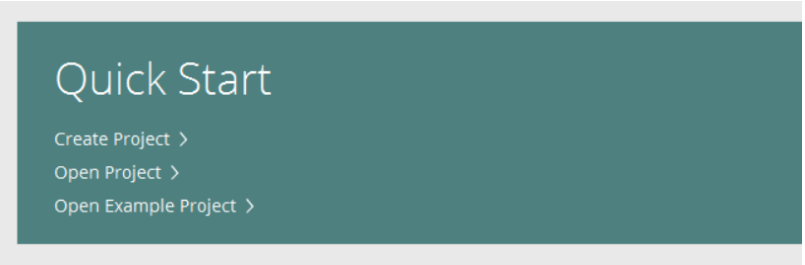
AIM: To design and simulate multi-bit XOR Gate.

Apparatus Required: Vivado Design Suite.

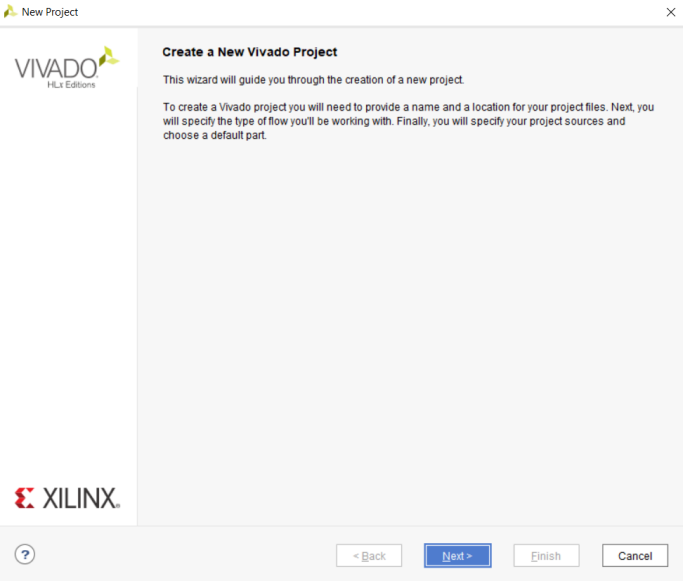
Procedure:

1) **To create a Multibit (8bit) XOR gate:**

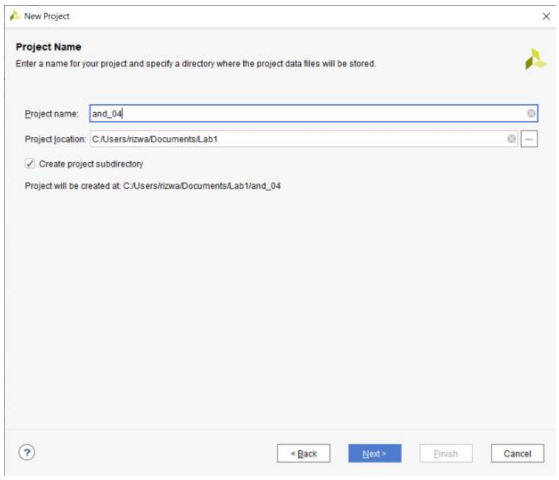
   - Open *Vivado Design Suite*
   - click on *Create Project* option.



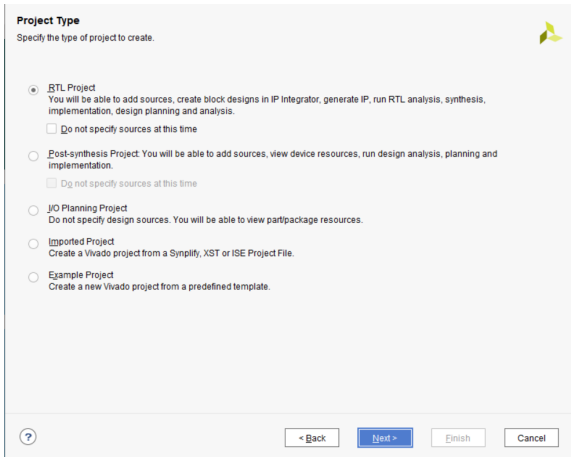
   - Click next on Create Project window



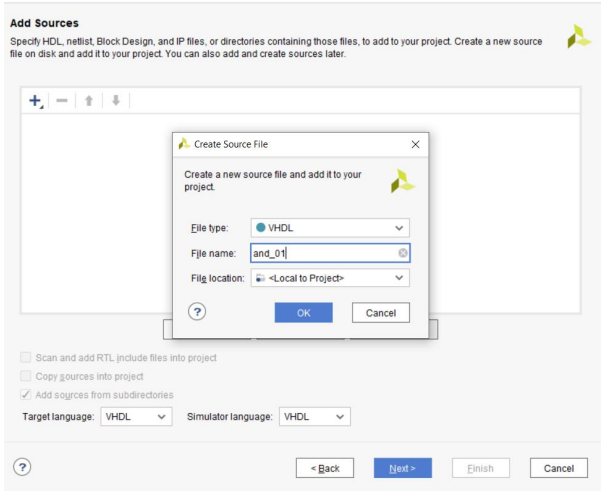
   - Name the project “Xor_04” and select the location to save the project, then click next



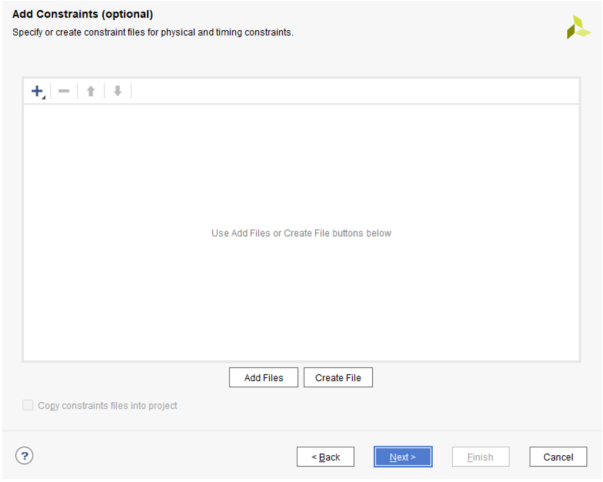
- In Project Type window select *RTL Project* and click next.



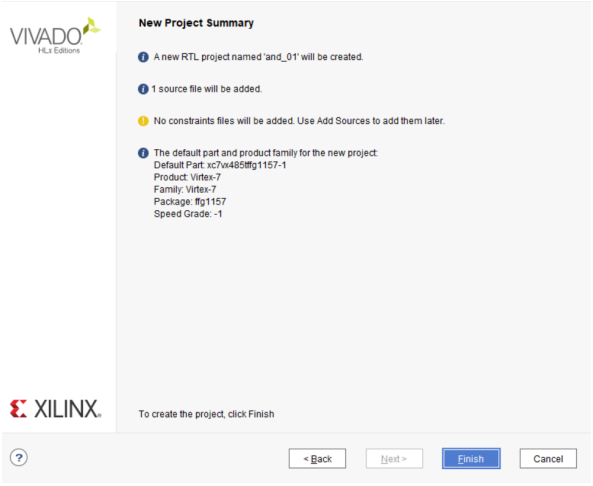
- In Add Sources window click *Create File*, then select file type as *VHDL* and name the file “xor_04”, click OK. Make sure *VHDL* is selected as Target language and Simulator Language and click Next



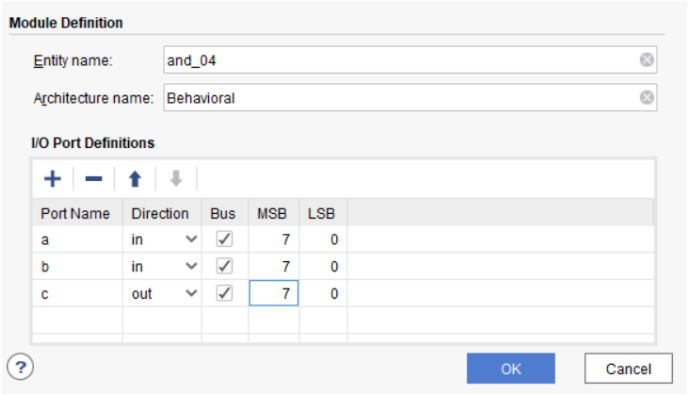
- Click next on the Add constraints window and in the Default Part window select the board available.



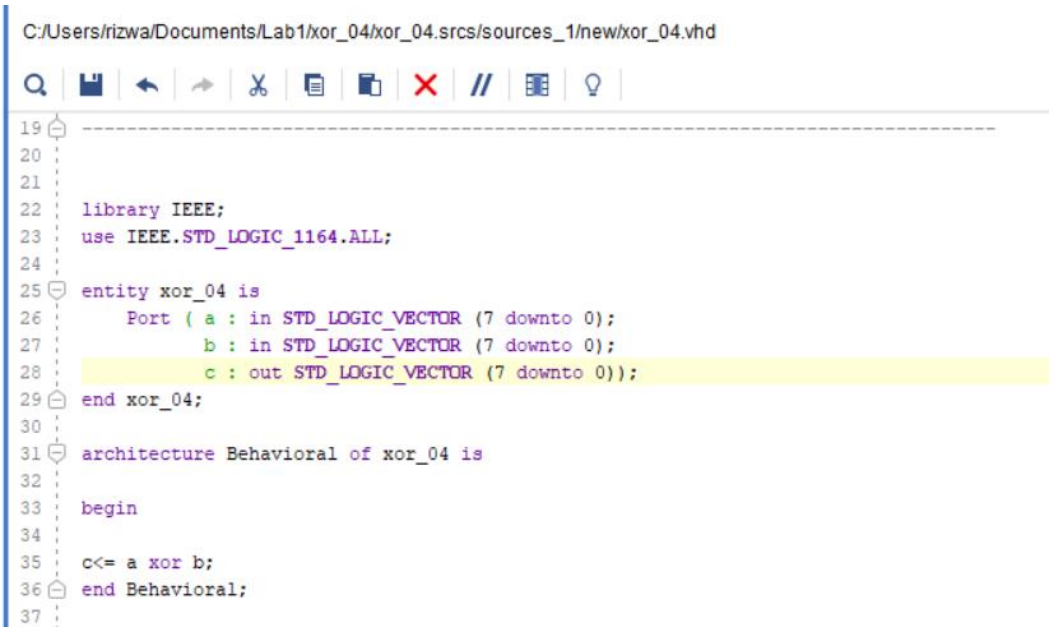
- Click Finish to create project.



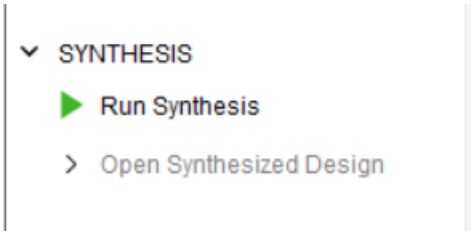
- Define module window appears, in the I/O Definitions tab of the window define the input and output ports for multibit XOR gate and click OK. For multibit XOR, input and output ports will be bus. Here the input and output is of 8 bits therefore, MSB is 7 and LSB 0^{th} bit.



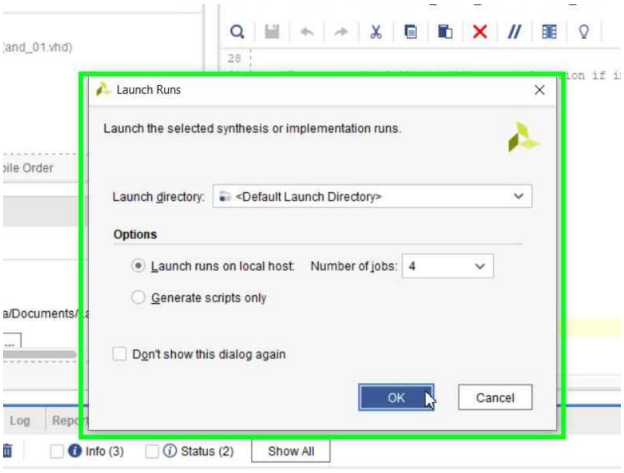
- In the xor_04.vhd file add the following code to define (Behavioral) the gate and save it by pressing ctrl + s:



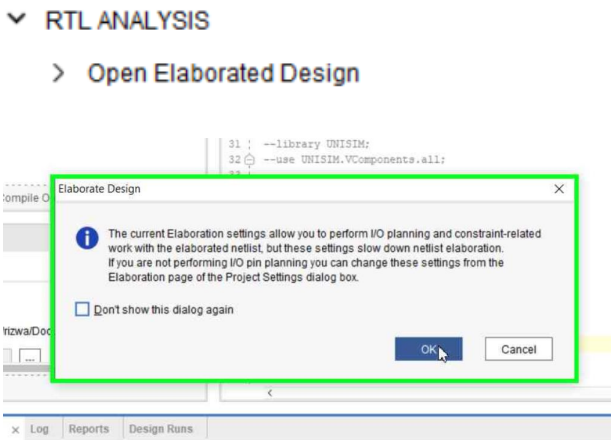
- To synthesize the design, click on Run Synthesis:



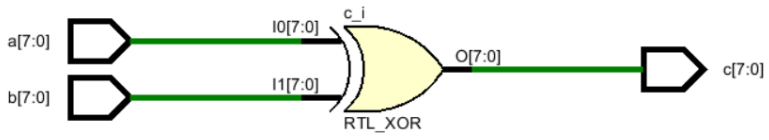
- Then click on OK in Launch Run window



- Close the synthesis completed dialogue, and then Click on Open Elaborated Design and Click OK



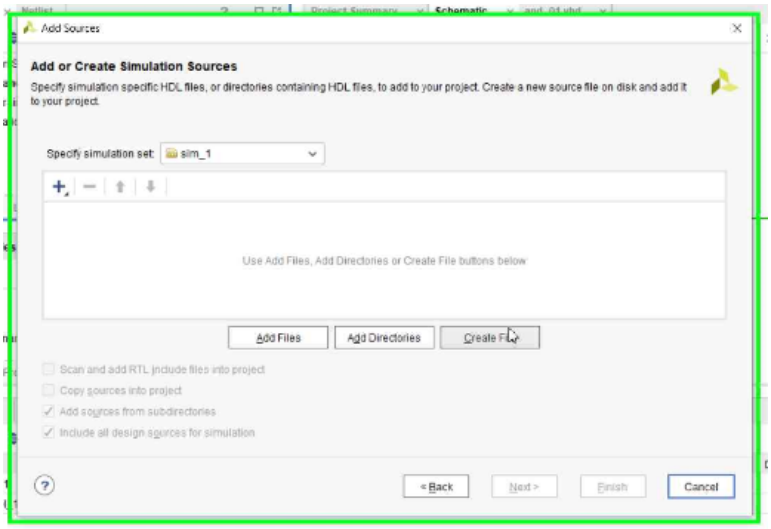
- This shows the schematic of the design



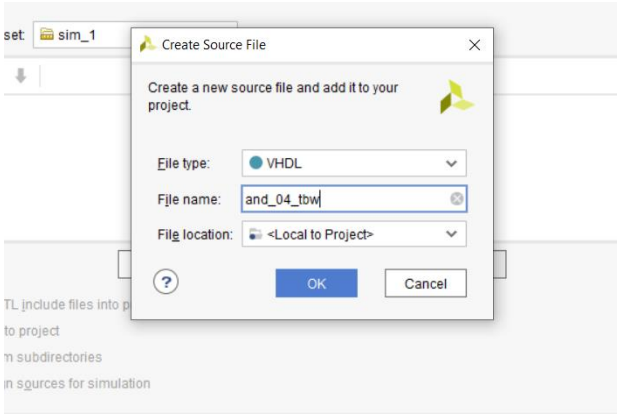
2) To Create a Test Bench:

- Click on Add Source and choose Add or Create Simulation Source, then click next.

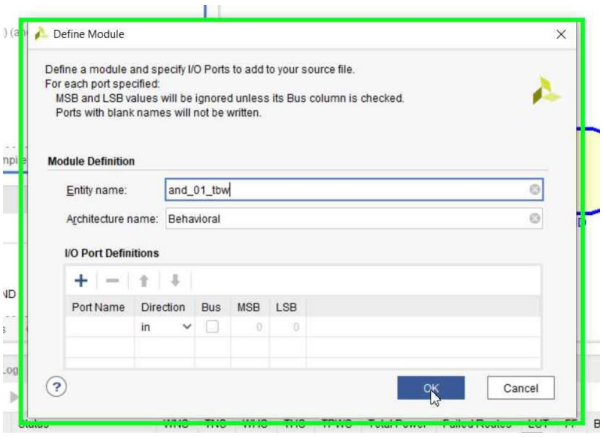
- In Add or Create Simulation Source Window click on Create File



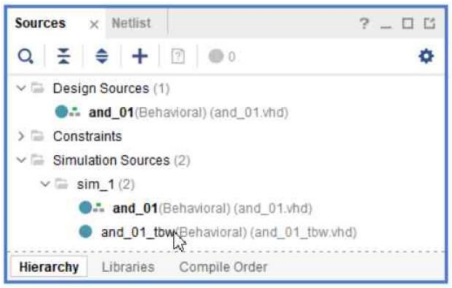
- Select File type as VHDL and name the file “xor_04_tbw”. Then click OK and then Finish.



- In Define module window click OK and if the dialogue pops up click on YES.



- Under sources window, double click on “xor_04_tbw” file.

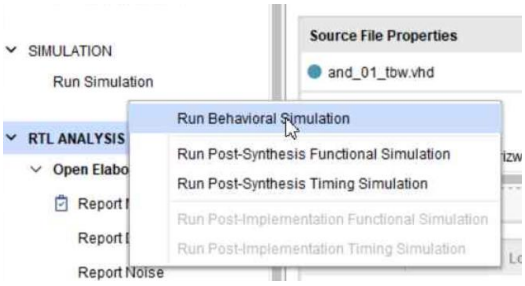


- Then in xor_04_tbw.vhd file add the following code:

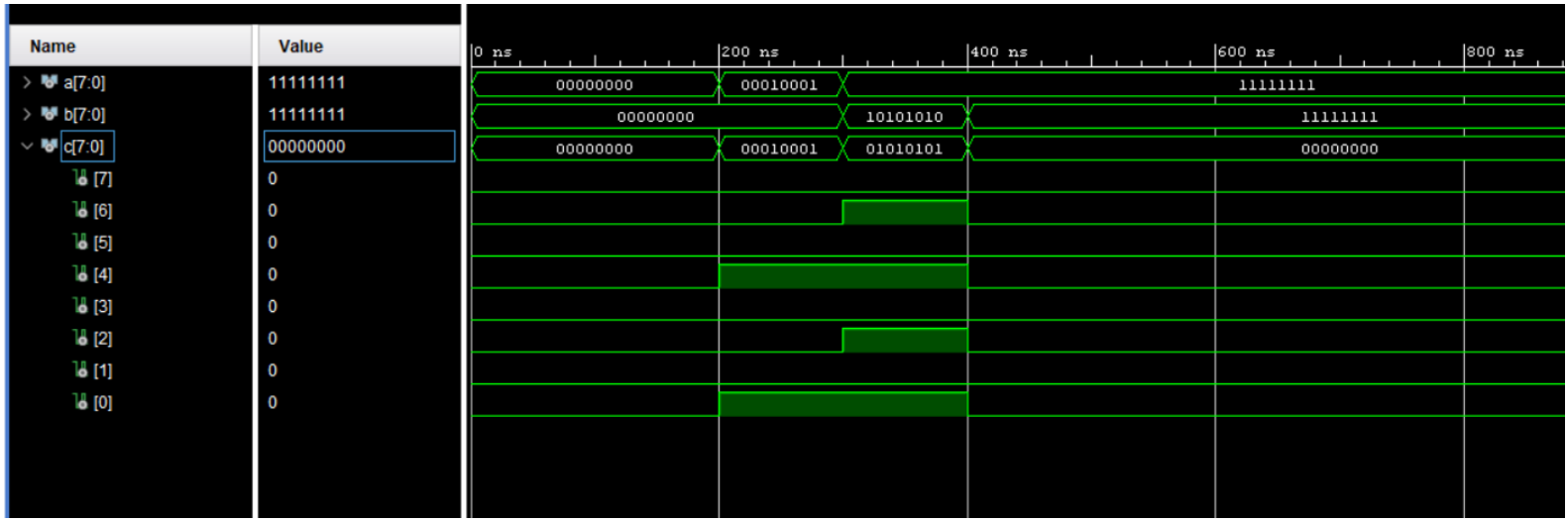
```
library IEEE;
use IEEE.std_logic_1164.ALL;
ENTITY xor_04_tbw IS
END xor_04_tbw;

ARCHITECTURE behavior OF xor_04_tbw IS
   component xor_04 IS
   PORT(
        a : IN  std_logic_vector(7 downto 0);
        b : IN  std_logic_vector (7 downto 0);
        c : OUT  std_logic_vector (7 downto 0)
       );
   END COMPONENT;

  signal a : std_logic_vector(7 downto 0) := "00000000";
  signal b : std_logic_vector(7 downto 0) := "00000000";
  signal c : std_logic_vector(7 downto 0);
```

```
BEGIN
  uut: xor_04 PORT MAP (
        a => a,
        b => b,
        c => c);
  stim_proc: process
  begin
    -- hold reset state for 100 ns.
    wait for 100 ns;
    a<="00000000";
    b<="00000000";
    wait for 100 ns;
    a<="00010001";
    b<="00000000";
    wait for 100 ns;
    a<="11111111";
    b<="10101010";
    wait for 100 ns;
    a<="11111111";
    b<="11111111";
    wait;
  end process;

END;
```

- Save the file and under Simulation option double click on Run Simulation and choose Run Behavioral Simulation



- Behavioral Simulation window appears showing the, waveform of multibit XOR gate for different values of input.



EXPERIMENT 12

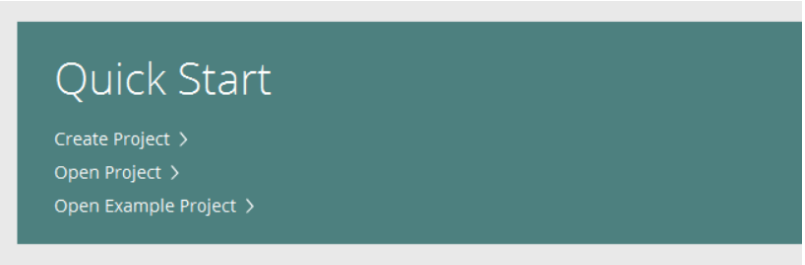
AIM: To design and simulate multi-bit NOT Gate.

Apparatus Required: Vivado Design Suite.

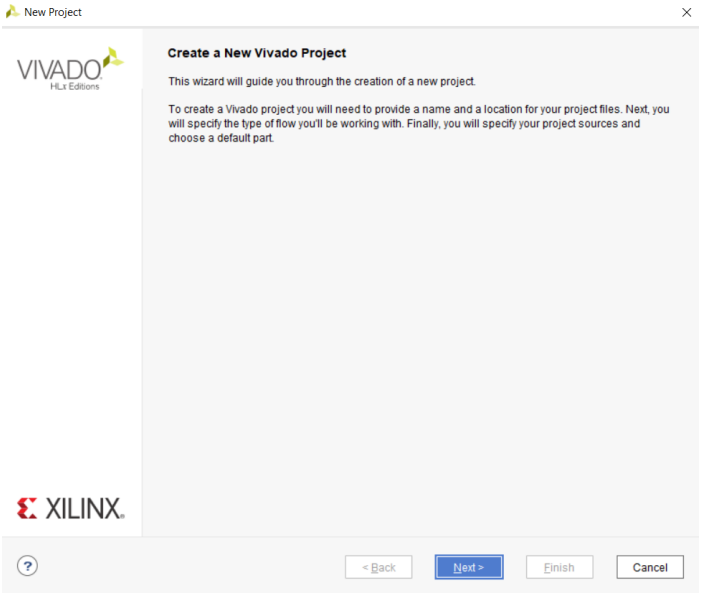
Procedure:

1) **To create a Multibit (8bit) NOT gate:**

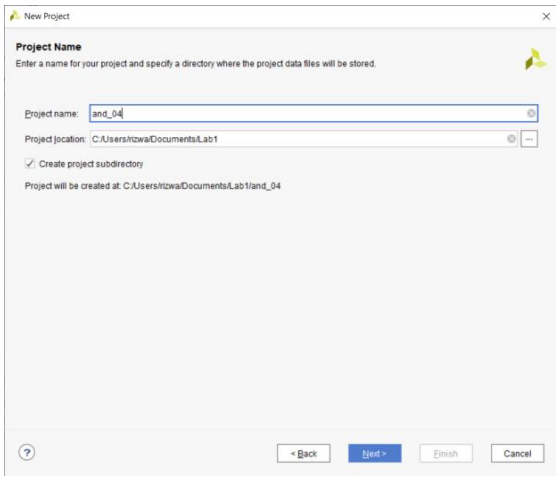
   - Open *Vivado Design Suite*
   - click on *Create Project* option.



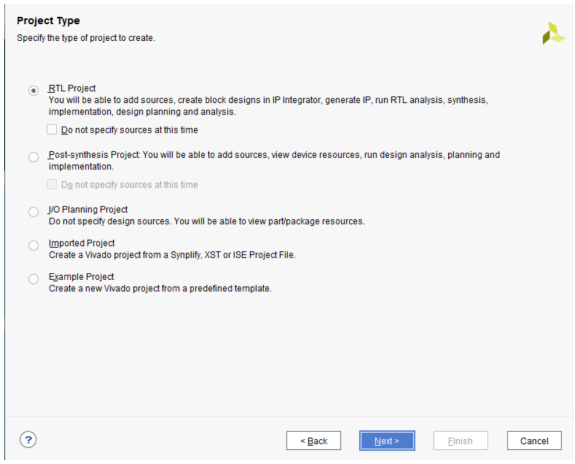
   - Click next on Create Project window



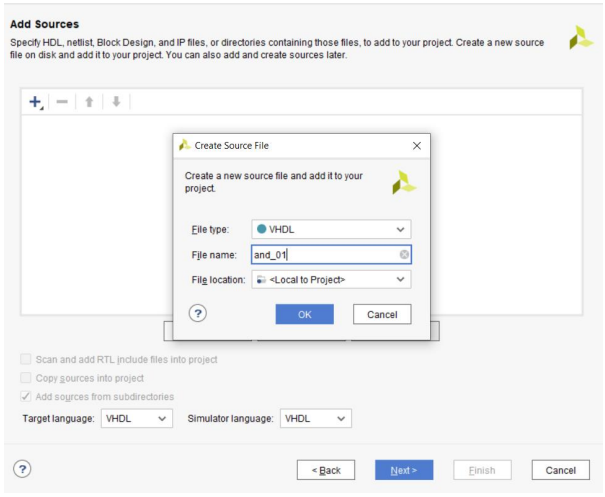
   - Name the project “not_02” and select the location to save the project, then click next



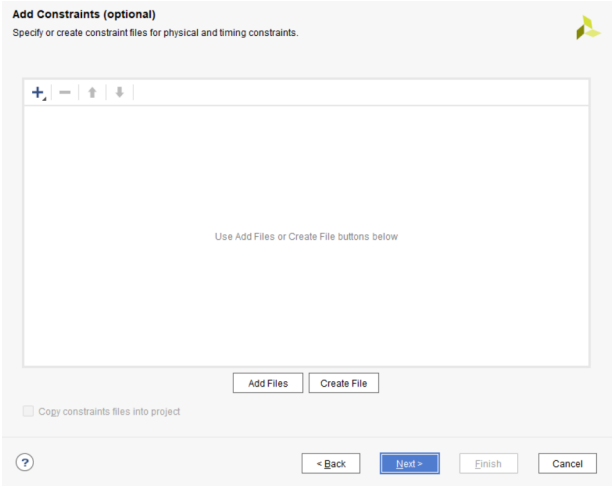
- In Project Type window select *RTL Project* and click next.



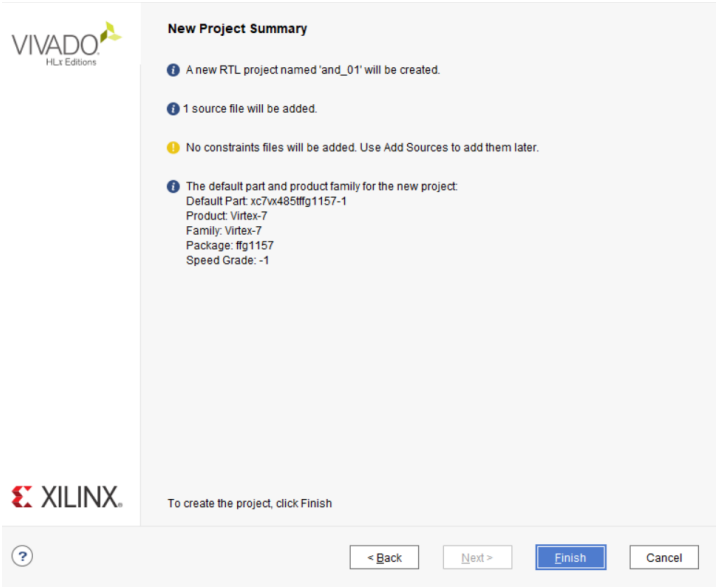
- In Add Sources window click *Create File*, then select file type as *VHDL* and name the file “not_02”, click OK. Make sure *VHDL* is selected as Target language and Simulator Language and click Next



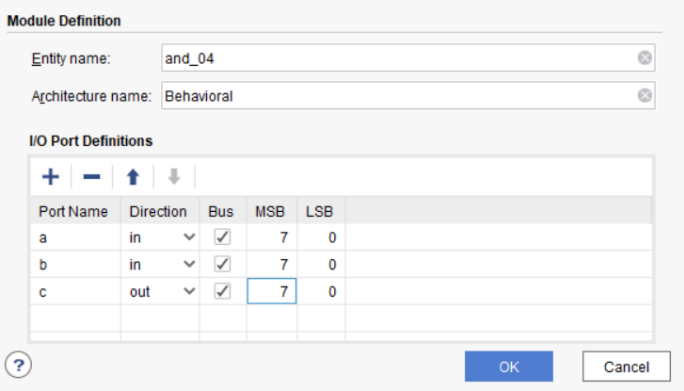
- Click next on the Add constraints window and in the Default Part window select the board available.



- Click Finish to create project.



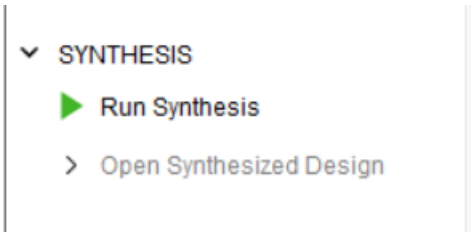
- Define module window appears, in the I/O Definitions tab of the window define the input and output ports for multibit NOT gate and click OK. For multibit NOT, input and output ports will be bus. Here the input and output is of 8 bits therefore, MSB is 7 and LSB 0^{th} bit.



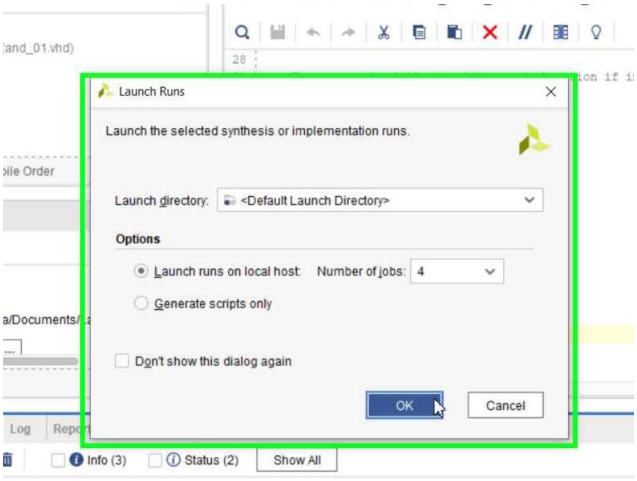
- In the not_02.vhd file add the following code to define (Behavioral) the gate and save it by pressing ctrl + s:

```
18  --
19  ----------------------------------------------------------------------------------
20
21
22  library IEEE;
23  use IEEE.STD_LOGIC_1164.ALL;
24
25  entity not_02 is
26      Port ( a : in STD_LOGIC_VECTOR (7 downto 0);
27             b : out STD_LOGIC_VECTOR (7 downto 0));
28  end not_02;
29
30  architecture Behavioral of not_02 is
31
32  begin
33  b<= not a;
34
35  end Behavioral;
36
```

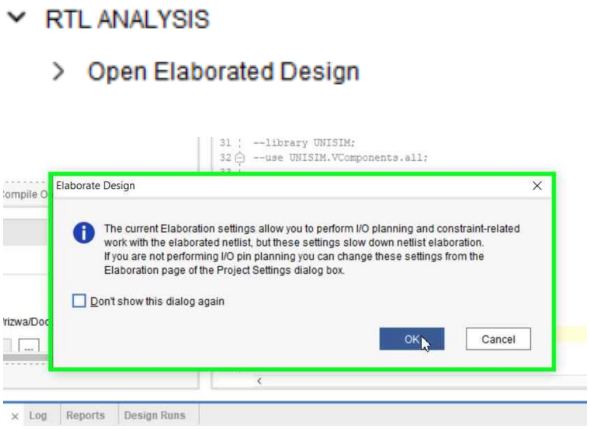
- To synthesize the design, click on Run Synthesis:



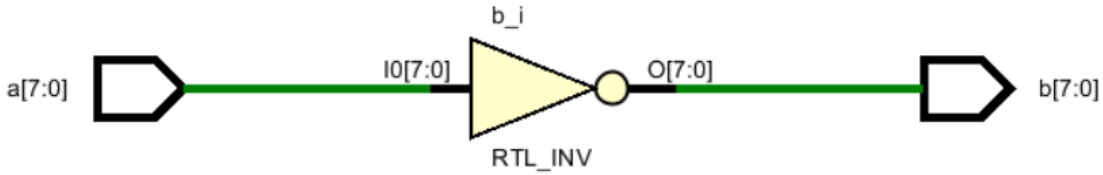
- Then click on OK in Launch Run window



- Close the synthesis completed dialogue, and then Click on Open Elaborated Design and Click OK

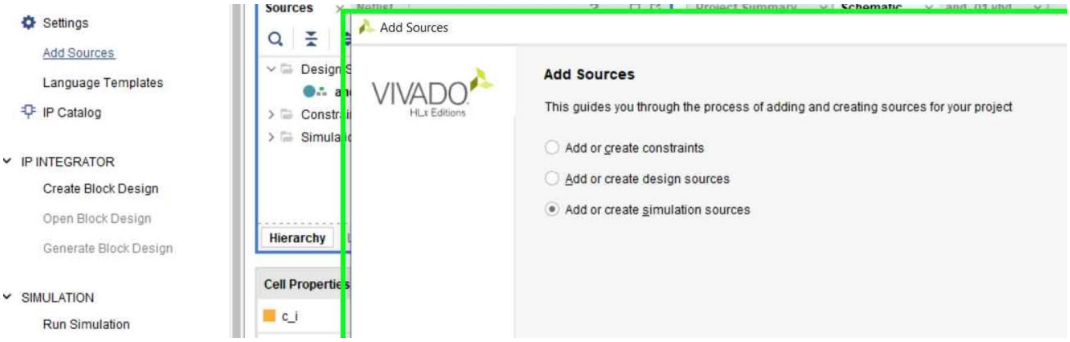


- This shows the schematic of the design

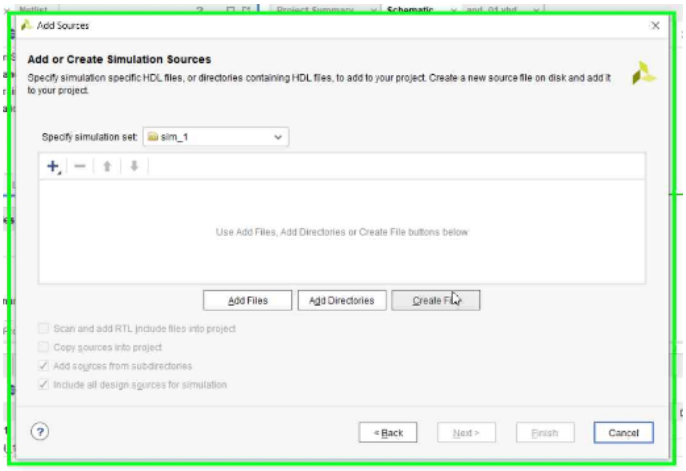


2) To Create a Test Bench:

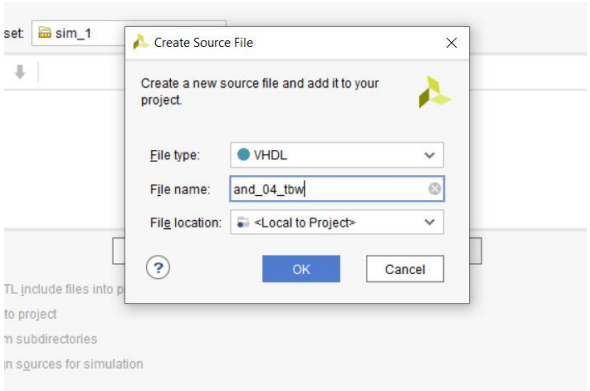
- Click on Add Source and choose Add or Create Simulation Source, then click next.



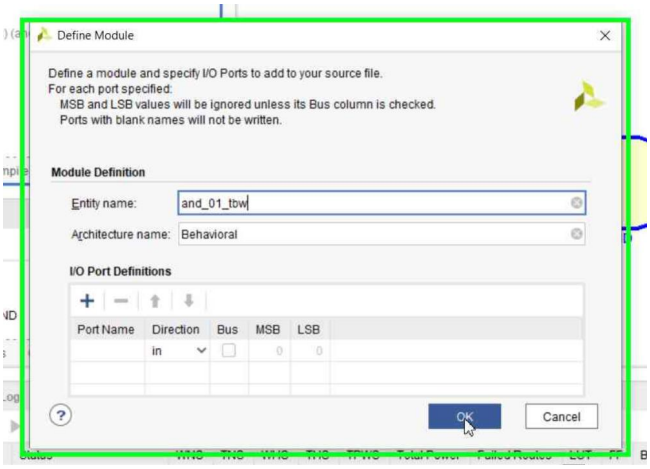
- In Add or Create Simulation Source Window click on Create File



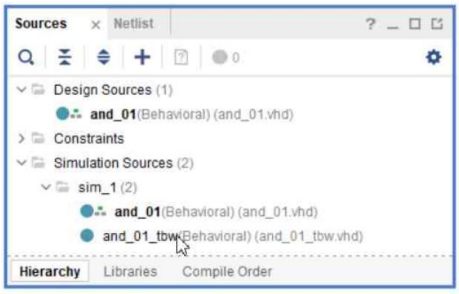
- Select File type as VHDL and name the file “not_02_tbw”. Then click OK and then Finish.



- In Define module window click OK and if the dialogue pops up click on YES.



- Under sources window, double click on “not_02_tbw” file.

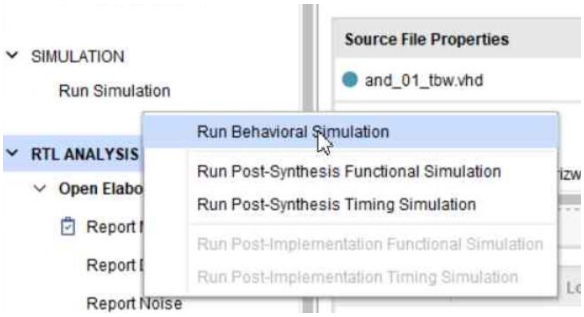


- Then in not_02_tbw.vhd file add the following code:

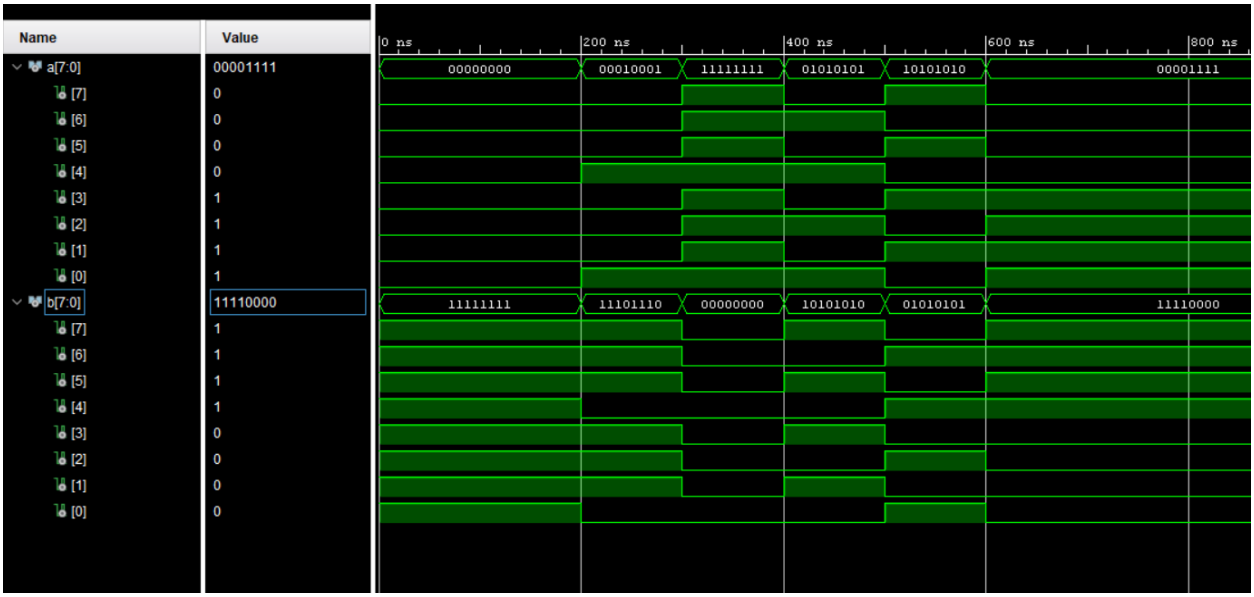
```
library IEEE;
use IEEE.std_logic_1164.ALL;
ENTITY not_02_tbw IS
END not_02_tbw;

ARCHITECTURE behavior OF not_02_tbw IS
   component not_02 IS
   PORT(
        a : IN  std_logic_vector(7 downto 0);
        b : OUT  std_logic_vector (7 downto 0)
       );
   END COMPONENT;
  signal a : std_logic_vector(7 downto 0) := "00000000";

          --Outputs
```

```
  signal b : std_logic_vector(7 downto 0);
BEGIN

        -- Instantiate the Unit Under Test (UUT)
  uut: not_02 PORT MAP (
       a => a,
       b => b
      );
  stim_proc: process
  begin
    -- hold reset state for 100 ns.
    wait for 100 ns;
    a<="00000000";

    wait for 100 ns;
    a<="00010001";

    wait for 100 ns;
    a<="11111111";

    wait for 100 ns;
    a<="01010101";
   wait for 100 ns;
      a<="10101010";
         wait for 100 ns;
       a<="00001111";
    wait;
  end process;

END;
```

- Save the file and under Simulation option double click on Run Simulation and choose Run Behavioral Simulation



- Behavioral Simulation window appears showing the, waveform of multibit NOT gate for different values of input.



EXPERIMENT 13

AIM: To design 2x1 Single bit MUX using logical operators

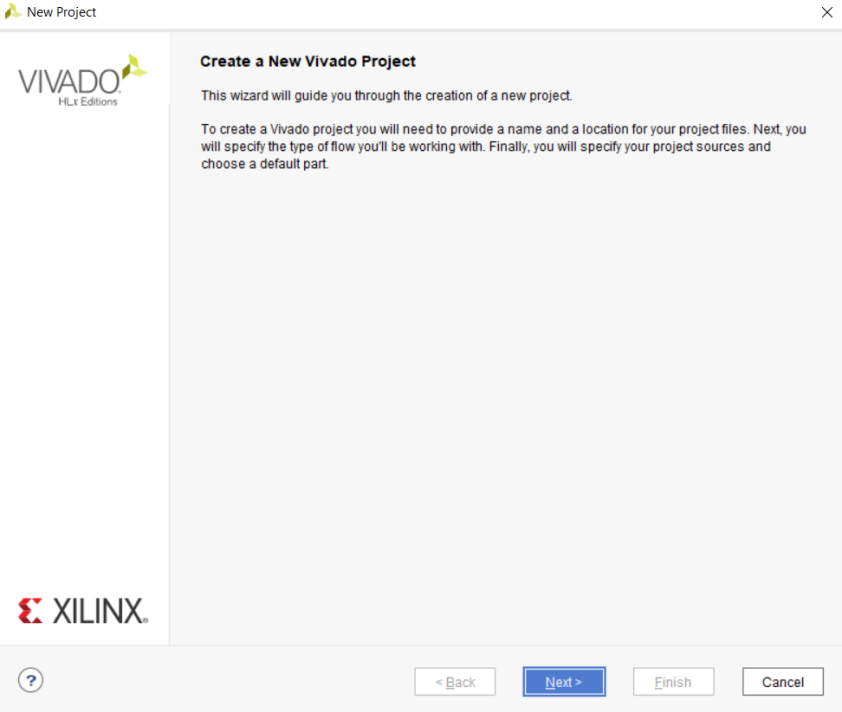
Apparatus Required: Vivado Design Suite.

Procedure:

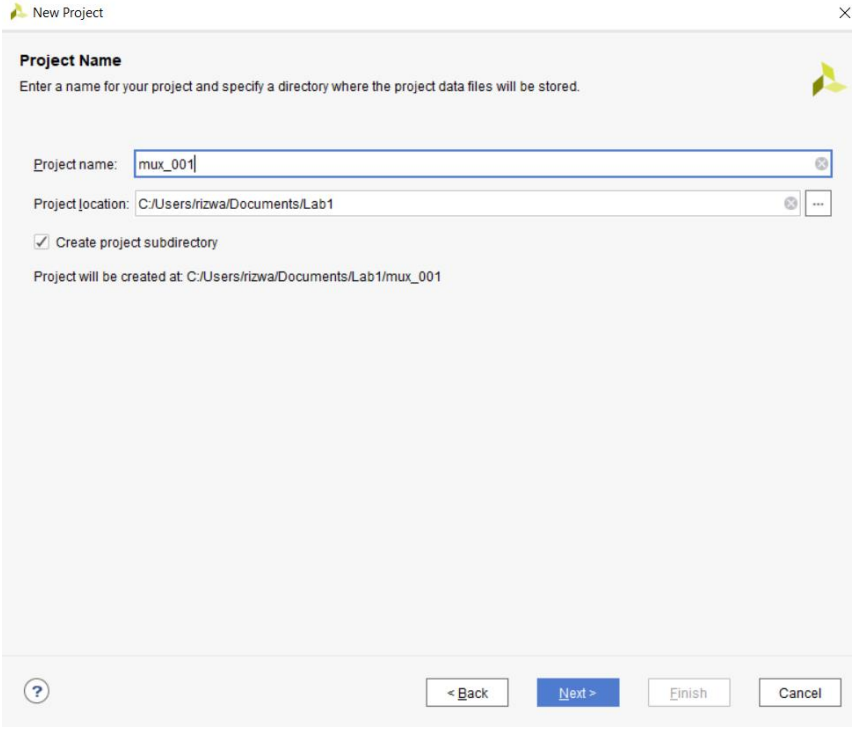
1) **To create a 2x1 MUX:**

   - Open *Vivado Design Suite*
   - click on *Create Project* option.

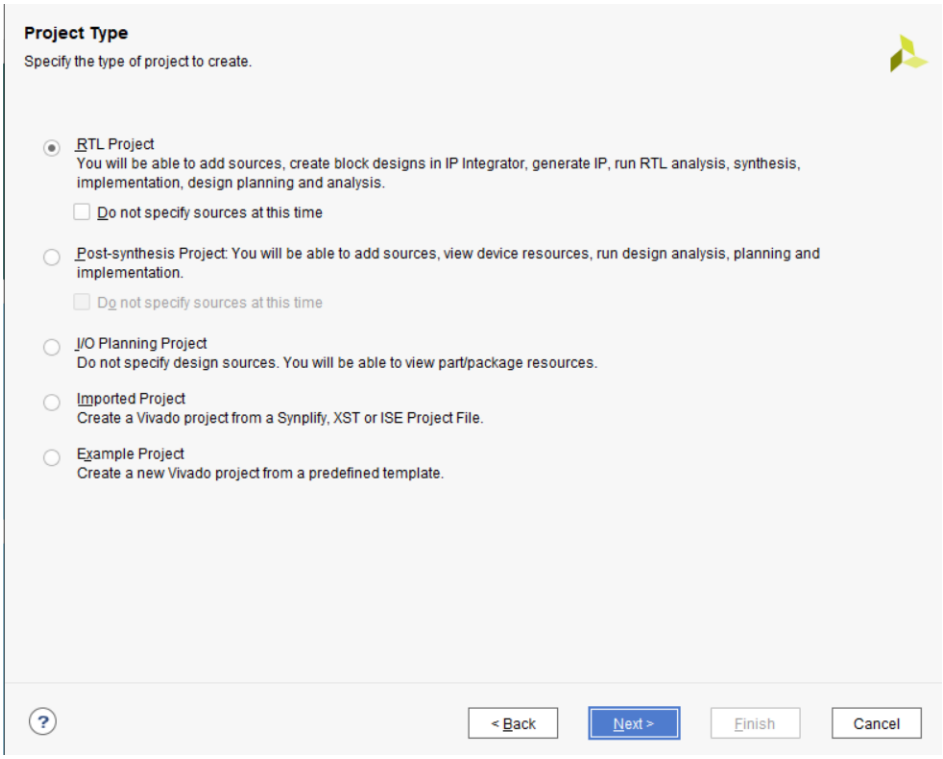
   - Click next on Create Project window



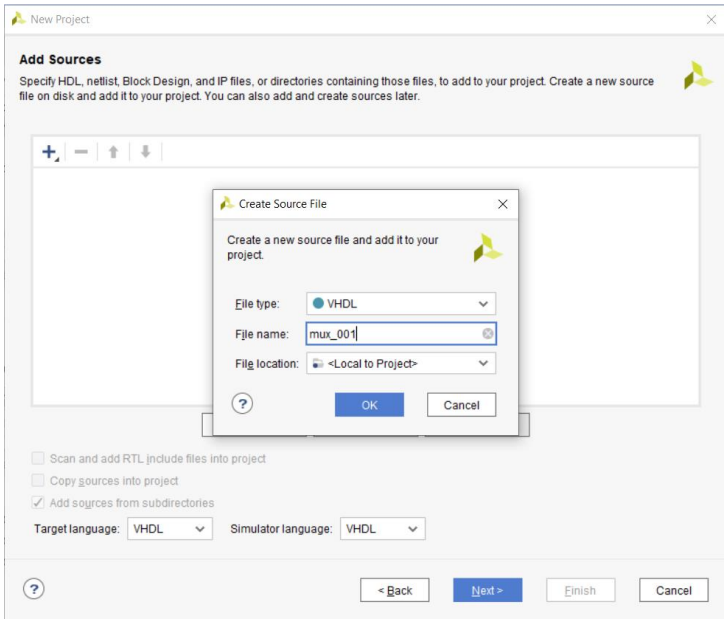
- Name the project “mux_001” and select the location to save the project, then click next



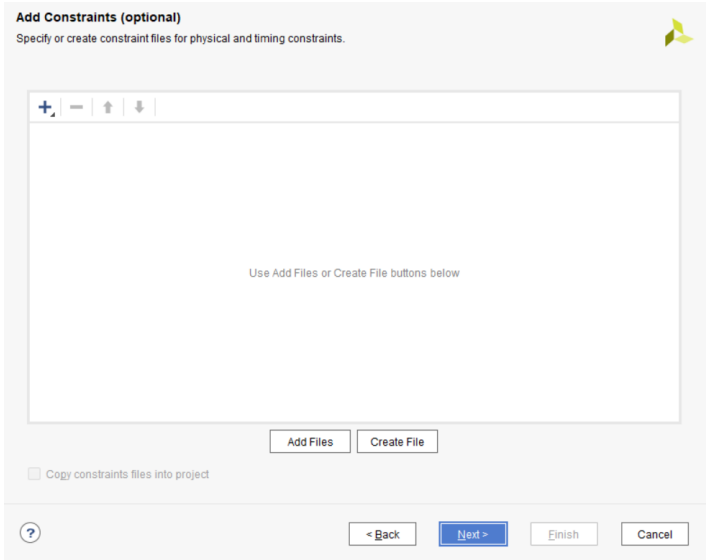
- In Project Type window select *RTL Project* and click next.



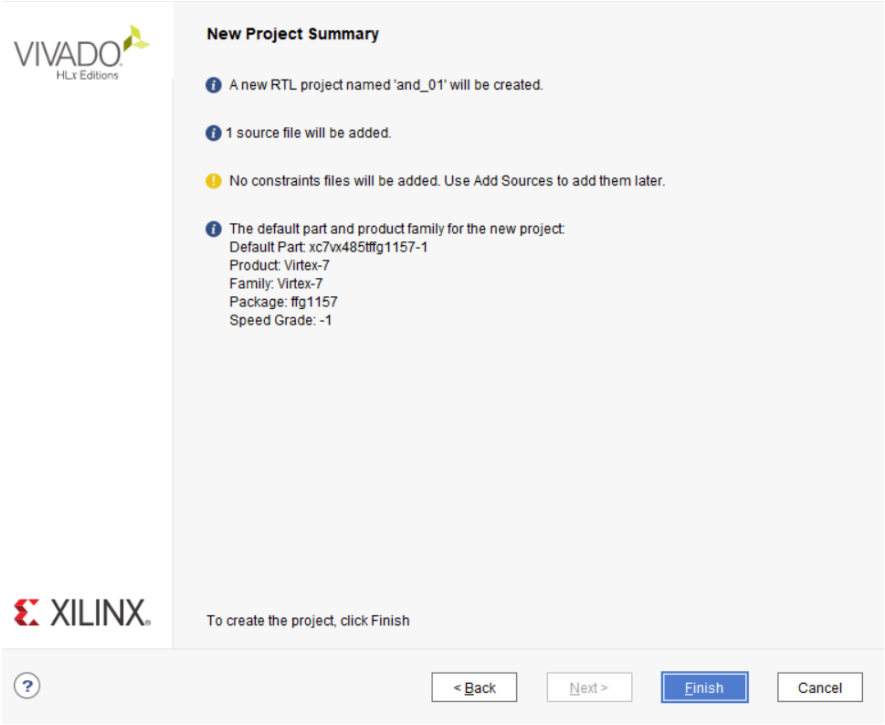
- In Add Sources window click *Create File*, then select file type as *VHDL* and name the file “mux_001”, click OK. Make sure *VHDL* is selected as Target language and Simulator Language and click Next



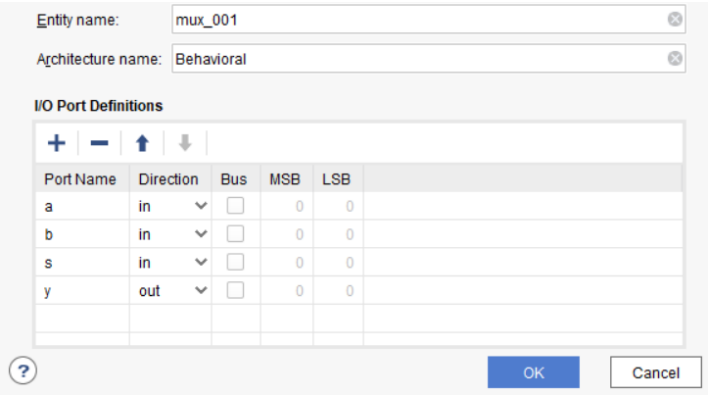
- Click next on the Add constraints window and in the Default Part window select the board available.



- Click Finish to create project.



- Define module window appears, in the I/O Definitions tab of the window define the input and output ports for 2x1 MUX and click OK.



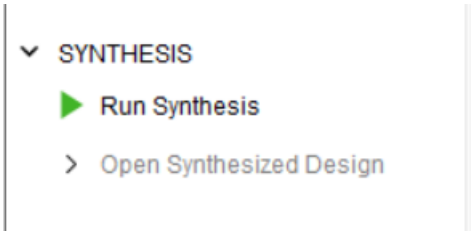
- In the mux_001.vhd file add the following code to define (Behavioral) the gate and save it by pressing ctrl + s:

Project Summary | mux_001.vhd *

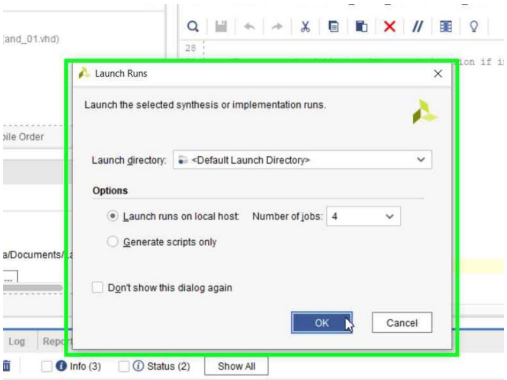
C:/Users/rizwa/Documents/Lab1/mux_001/mux_001.srcs/sources_1/new/mux_001.vhd

```
20
21
22  library IEEE;
23  use IEEE.STD_LOGIC_1164.ALL;
24  entity mux_001 is
25      Port ( a : in STD_LOGIC;
26             b : in STD_LOGIC;
27             s : in STD_LOGIC;
28             y : out STD_LOGIC);
29  end mux_001;
30
31  architecture Behavioral of mux_001 is
32
33  begin
34   y<= (a and (not s)) or (b and s);
35
36  end Behavioral;
37
```

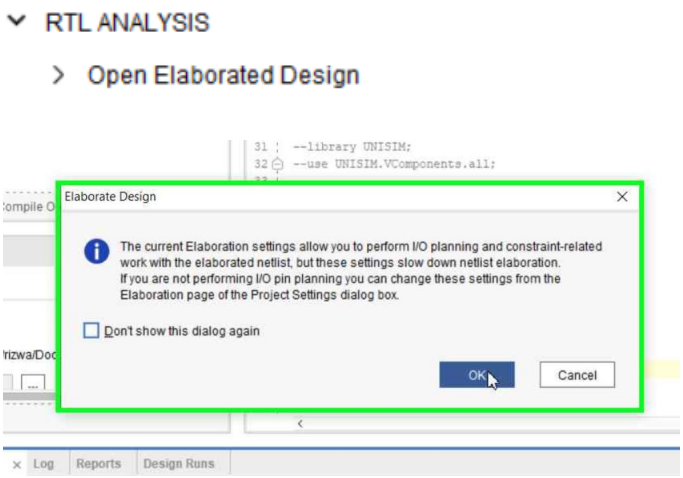
- To synthesize the design, click on Run Synthesis:



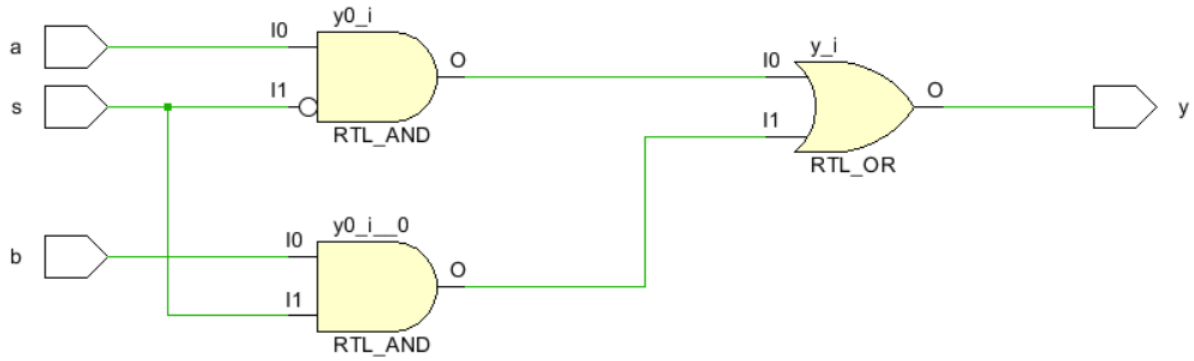
- Then click on OK in Launch Run window



- Close the synthesis completed dialogue, and then Click on Open Elaborated Design and Click OK

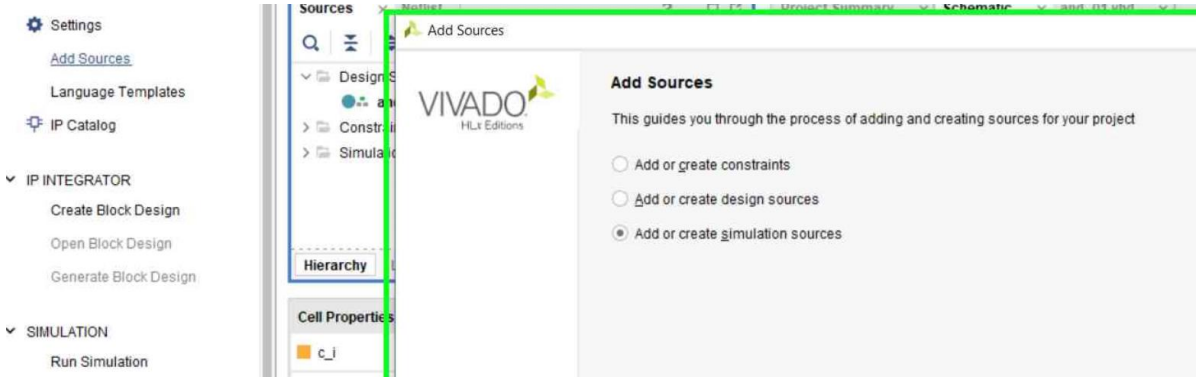


- This shows the schematic of the design

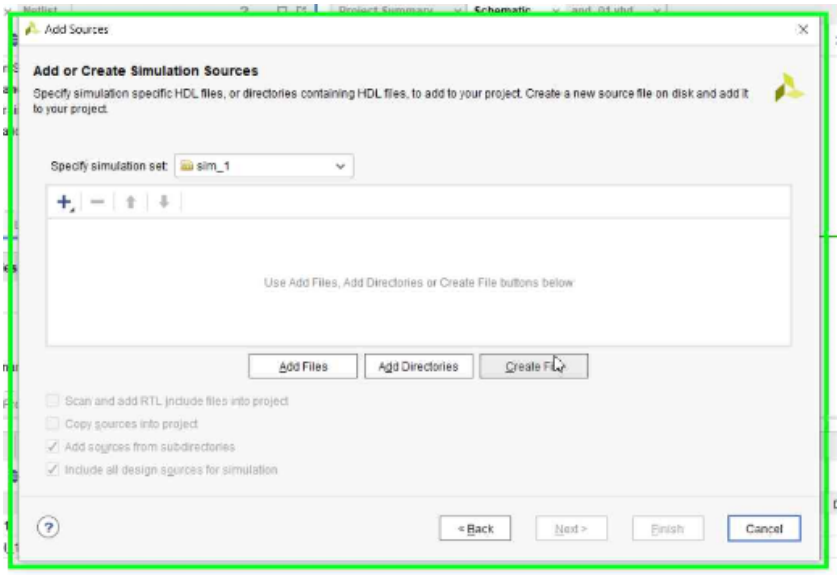


2) To Create a Test Bench:

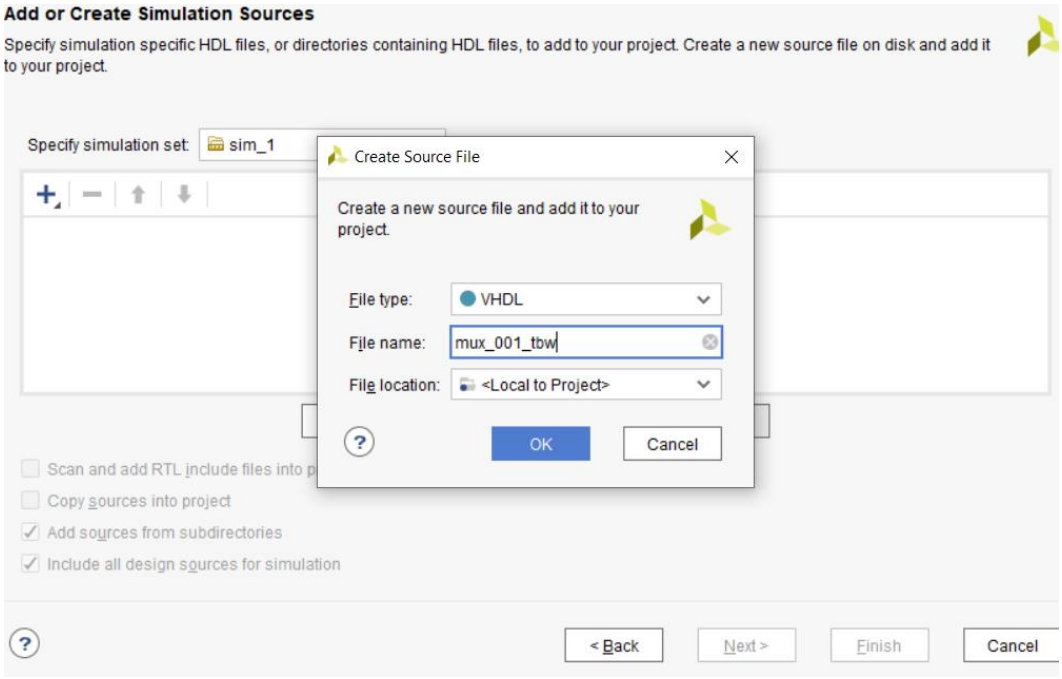
- Click on Add Source and choose Add or Create Simulation Source, then click next.



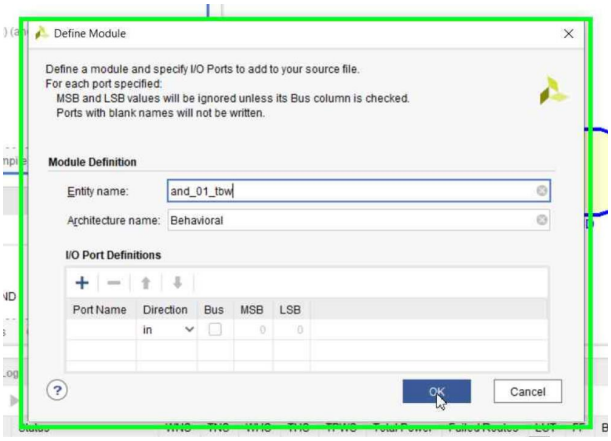
- In Add or Create Simulation Source Window click on Create File



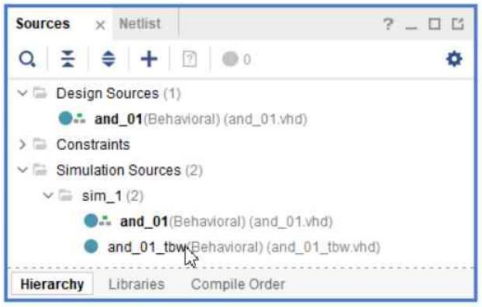
- Select File type as VHDL and name the file “mux_001_tbw”. Then click OK and then Finish.



- In Define module window click OK and if the dialogue pops up click on YES.



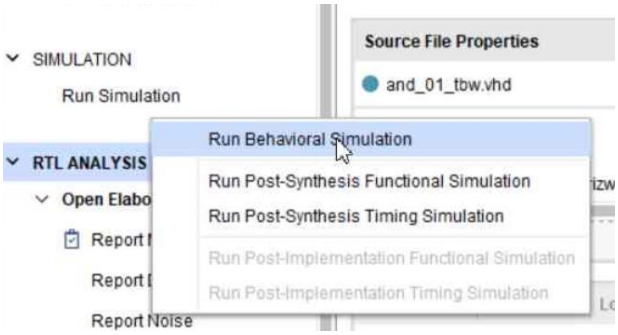
- Under sources window, double click on “mux_001_tbw” file.



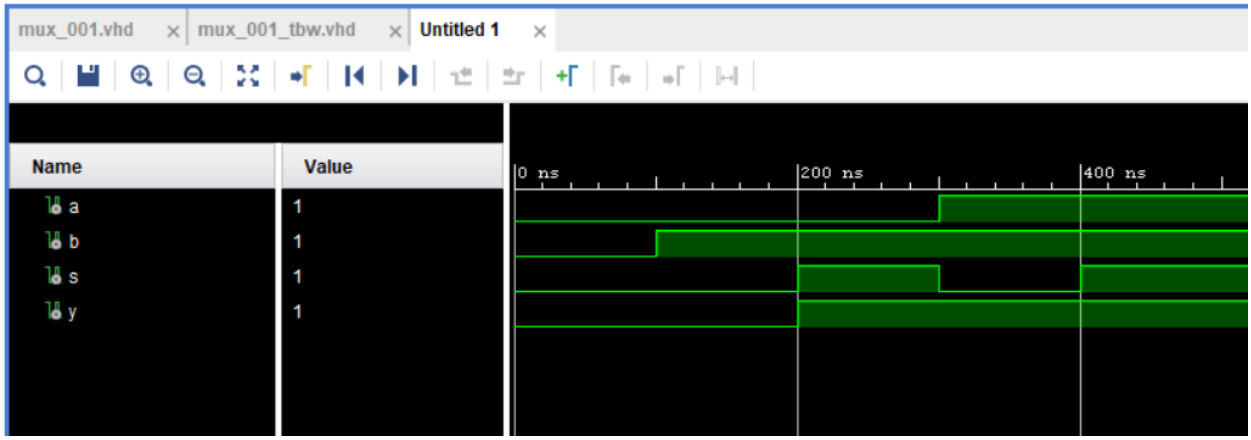
- Then in mux_001_tbw.vhd file add the following code:

```
library IEEE;
use IEEE.std_logic_1164.ALL;
ENTITY mux_001_tbw IS
END mux_001_tbw;
ARCHITECTURE behavior OF mux_001_tbw IS
    component mux_001 IS
    PORT(
         a : IN  std_logic;
         b : IN  std_logic;
         s : IN  std_logic;
         y : OUT  std_logic
        );
    END COMPONENT;
   signal a : std_logic := '0';
   signal b : std_logic := '0';
   signal s : std_logic:= '0';
   signal y : std_logic;
BEGIN
   uut: mux_001 PORT MAP (
          a => a,
          b => b,
          s =>s,
          y => y
        );
   stim_proc: process
   begin
      wait for 100 ns;
      a<='0';
      b<='1';
      s<='0';
      wait for 100 ns;
      a<='0';
      b<='1';
      s<='1';
      wait for 100 ns;
      a<='1';
      b<='1';
      s<='0';
      wait for 100 ns;
      a<='1';
      b<='1';
      s<='1';
      wait;
   end process;

END;
```

- Save the file and under Simulation option double click on Run Simulation and choose Run Behavioral Simulation



- Behavioral Simulation window appears showing the, waveform of 2x1 MUX for different values of input.



EXPERIMENT 14

AIM: To design 2x1 Single bit MUX using 'SELECT when' statement.

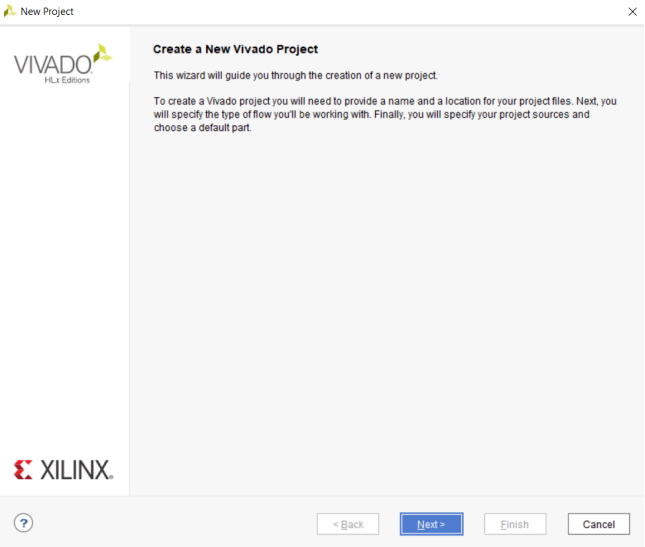
Apparatus Required: Vivado Design Suite.

Procedure:

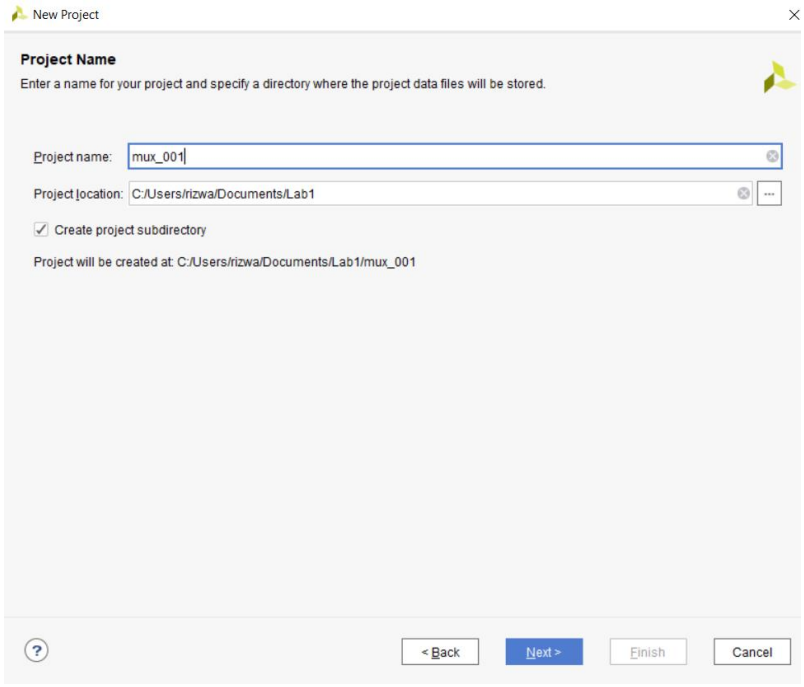
1) **To create a 2x1 MUX:**

- Open *Vivado Design Suite*
- click on *Create Project* option.

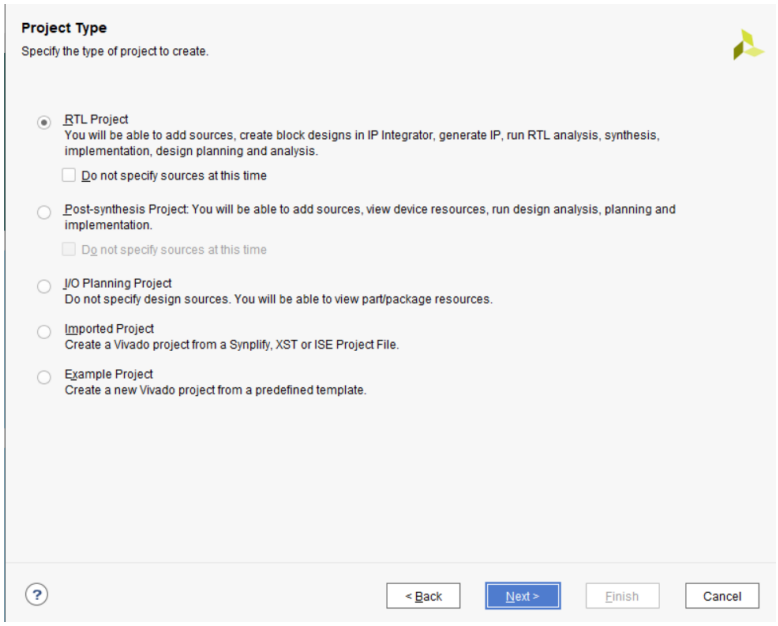
- Click next on Create Project window



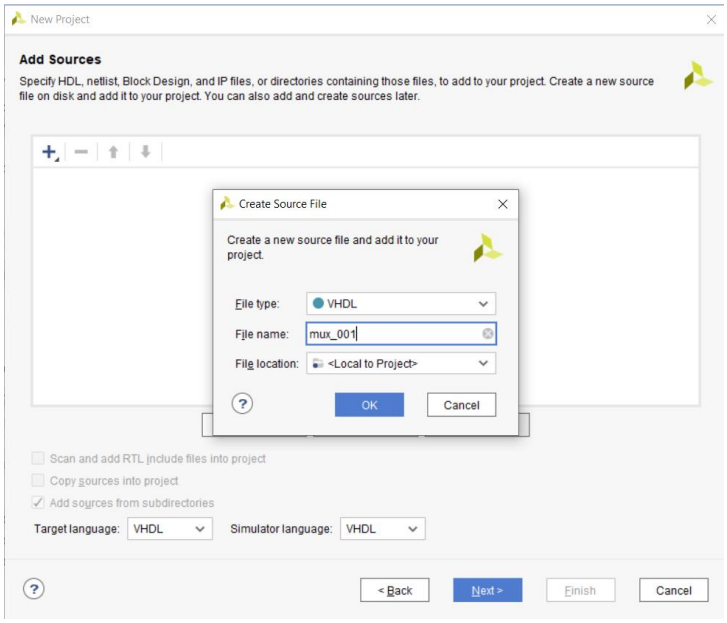
- Name the project "mux_02" and select the location to save the project, then click next



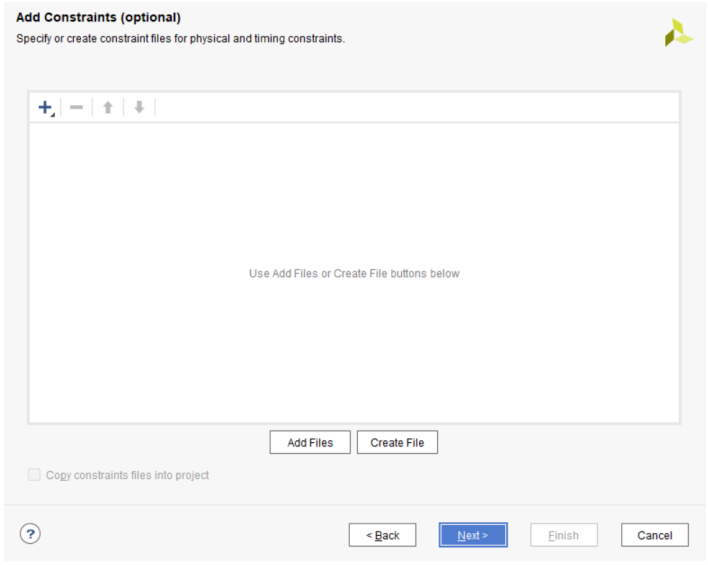
- In Project Type window select *RTL Project* and click next.



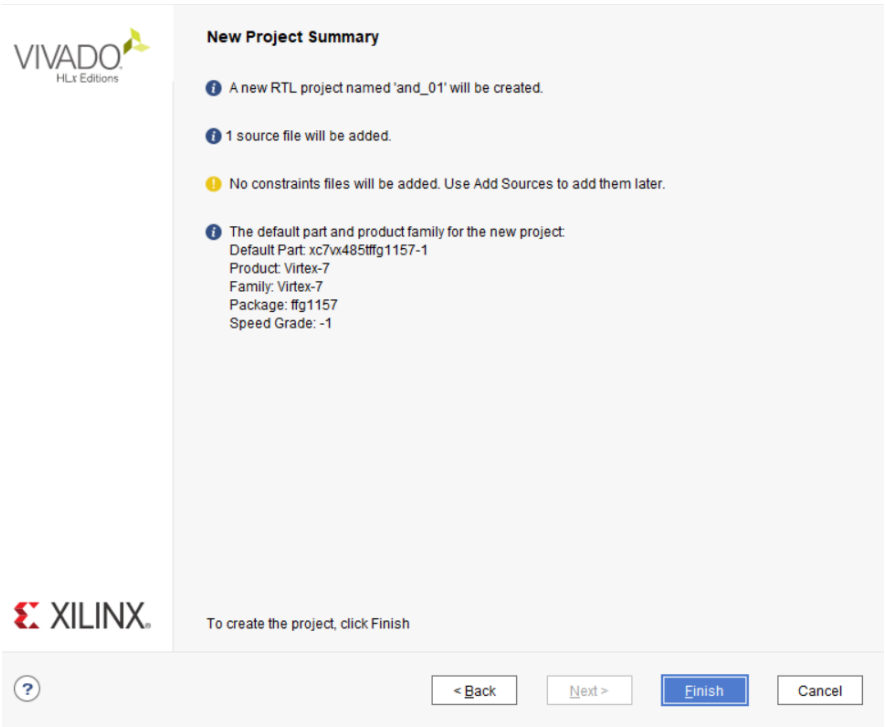
- In Add Sources window click *Create File*, then select file type as *VHDL* and name the file “mux_02”, click OK. Make sure *VHDL* is selected as Target language and Simulator Language and click Next



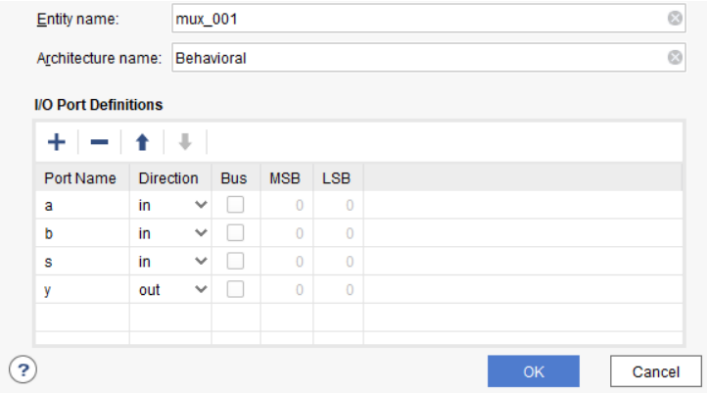
- Click next on the Add constraints window and in the Default Part window select the board available.



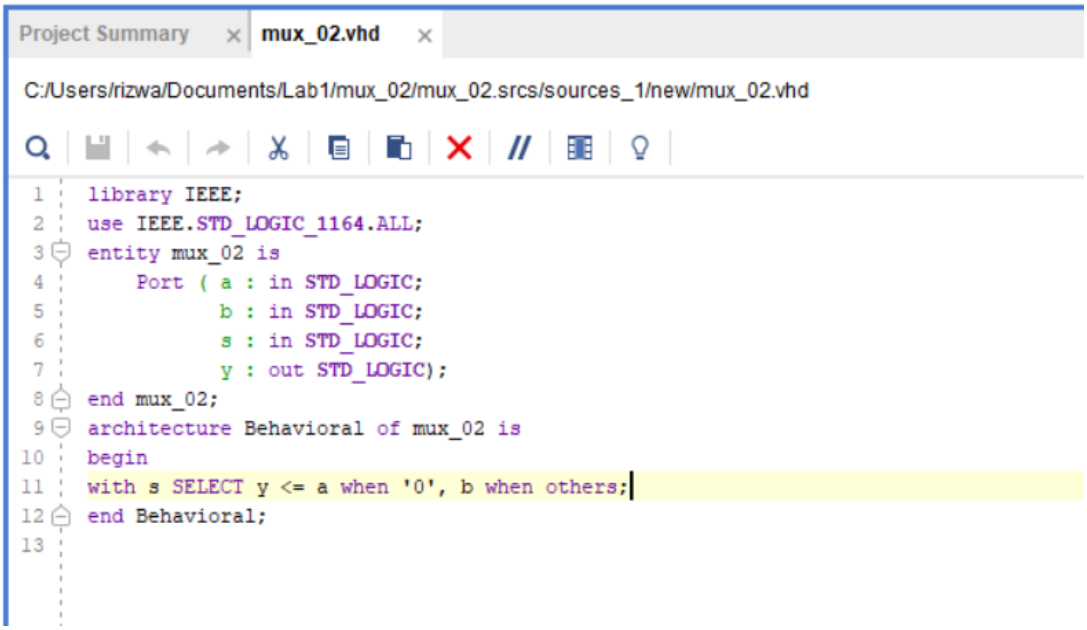
- Click Finish to create project.



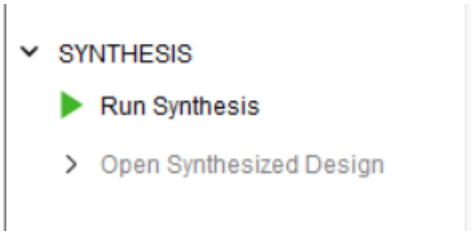
- Define module window appears, in the I/O Definitions tab of the window define the input and output ports for 2x1 MUX and click OK.



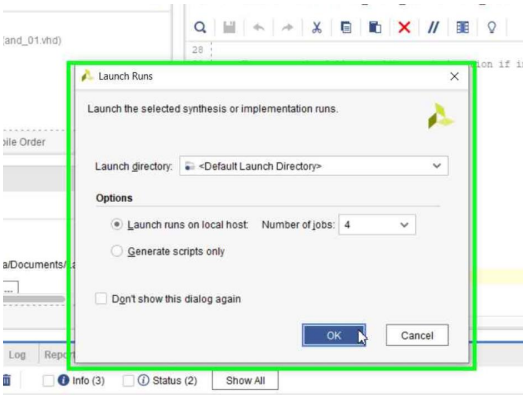
- In the mux_02.vhd file add the following code to define (Behavioral) the gate and save it by pressing ctrl + s:



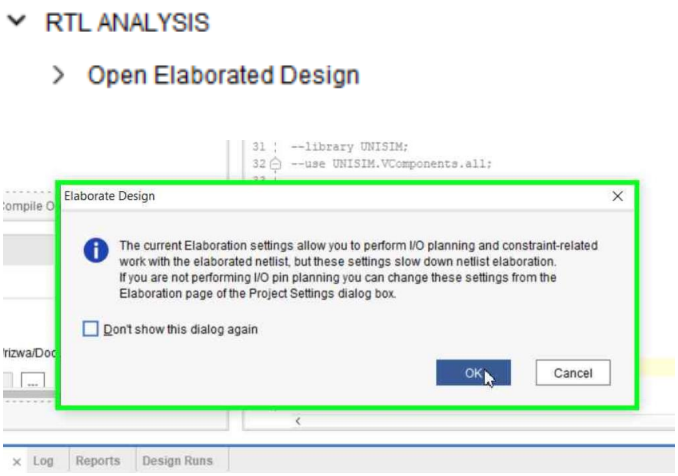
- To synthesize the design, click on Run Synthesis:



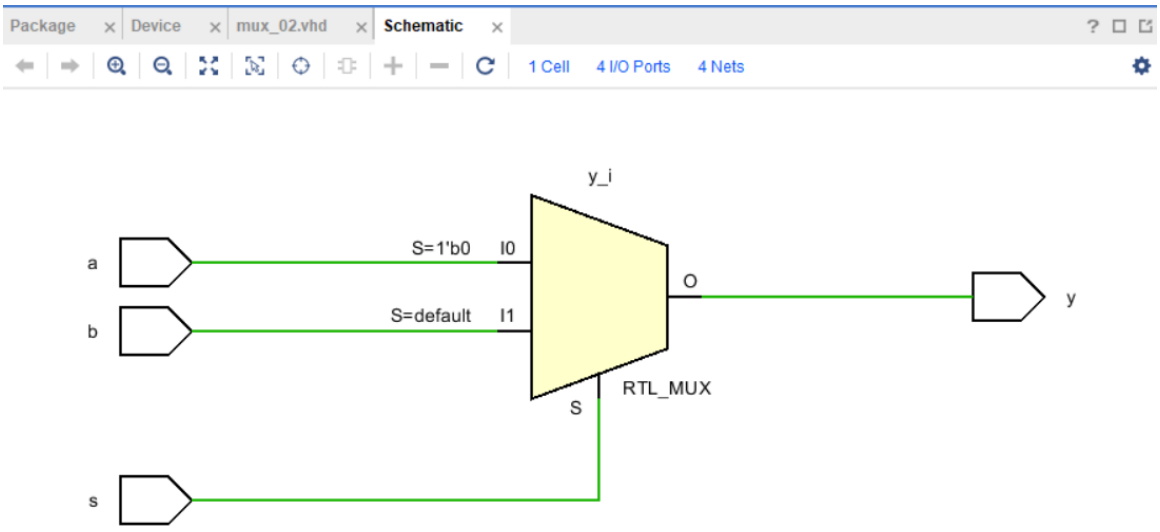
- Then click on OK in Launch Run window



- Close the synthesis completed dialogue, and then Click on Open Elaborated Design and Click OK

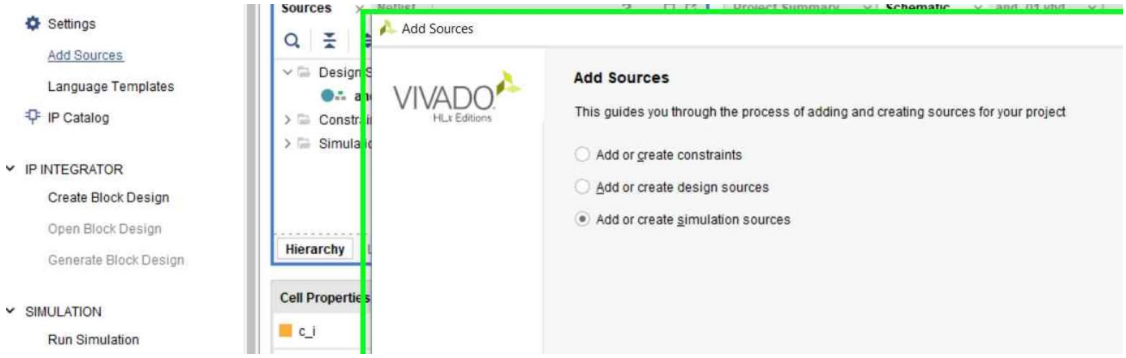


- This shows the schematic of the design

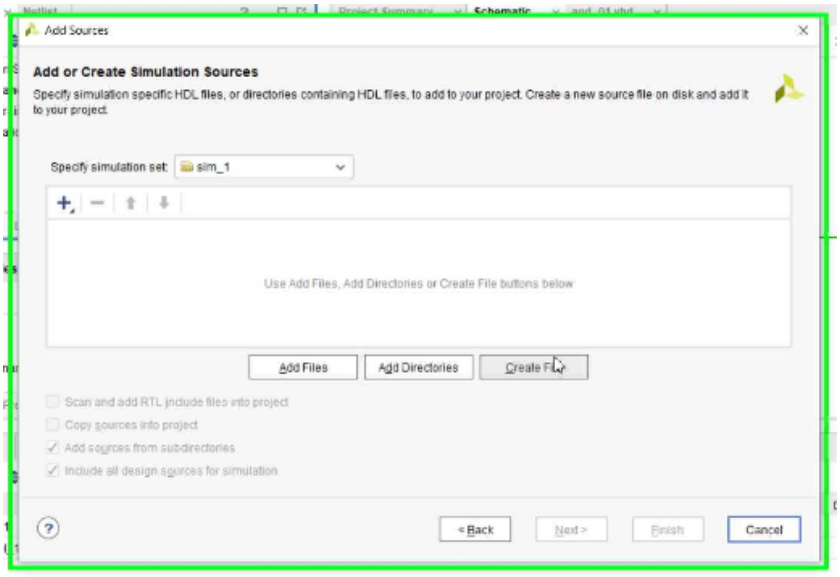


2) To Create a Test Bench:

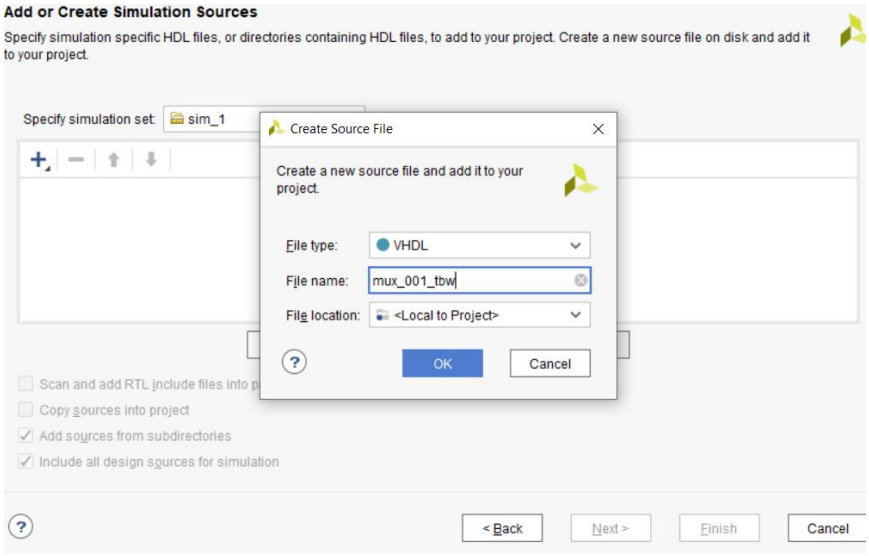
- Click on Add Source and choose Add or Create Simulation Source, then click next.



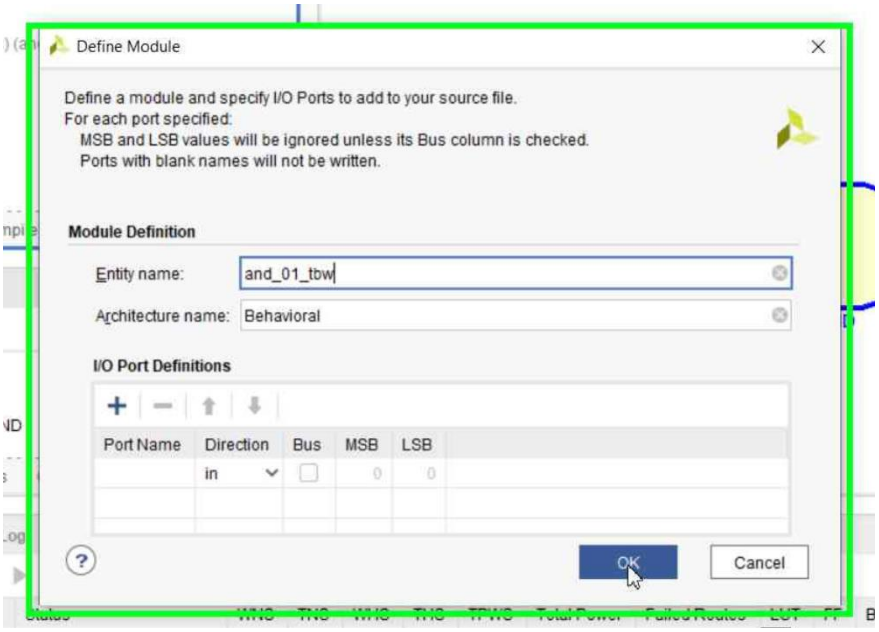
- In Add or Create Simulation Source Window click on Create File



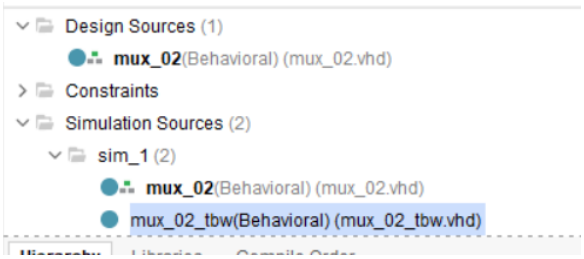
- Select File type as VHDL and name the file “mux_02_tbw”. Then click OK and then Finish.



- In Define module window click OK and if the dialogue pops up click on YES.



- Under sources window, double click on “mux_02_tbw” file.

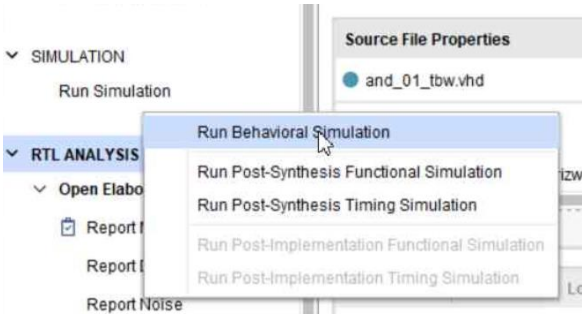


- Then in mux_02_tbw.vhd file add the following code:

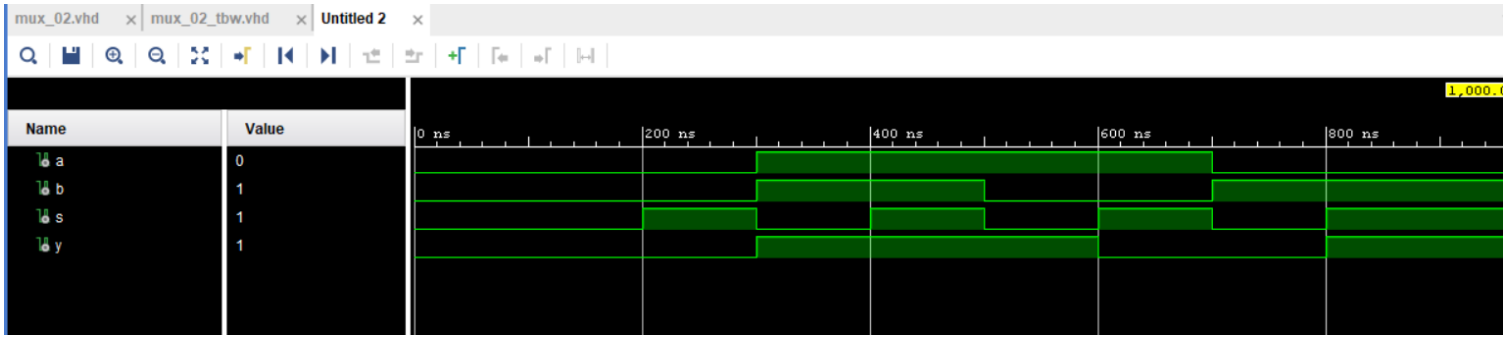
```
library IEEE;
use IEEE.std_logic_1164.ALL;
ENTITY mux_02_tbw IS
END mux_02_tbw;
ARCHITECTURE behavior OF mux_02_tbw IS
    -- Component Declaration for the Unit Under Test (UUT)
    component mux_02 IS
    PORT(
         a : IN  std_logic;
         b : IN  std_logic;
         s : IN  std_logic;
         y : OUT  std_logic
        );
    END COMPONENT;
   --Inputs
   signal a : std_logic := '0';
   signal b : std_logic := '0';
   signal s : std_logic:= '0';
     --Outputs
   signal y : std_logic;
BEGIN
      -- Instantiate the Unit Under Test (UUT)
   uut: mux_02 PORT MAP (
          a => a,
          b => b,
          s =>s,
          y => y
        );
   -- Stimulus process
   stim_proc: process
   begin
     -- hold reset state for 100 ns.
     wait for 100 ns;
     a<='0';
     b<='0';
     s<='0';
     wait for 100 ns;
     a<='0';
     b<='0';
     s<='1';
     wait for 100 ns;
     a<='1';
     b<='1';
     s<='0';
     wait for 100 ns;
```

```
    a<='1';
    b<='1';
    s<='1';
    wait for 100 ns;
    a<='1';
    b<='0';
    s<='0';
    wait for 100 ns;
    a<='1';
    b<='0';
    s<='1';
    wait for 100 ns;
    a<='0';
    b<='1';
    s<='0';
    wait for 100 ns;
    a<='0';
    b<='1';
    s<='1';
    wait for 100 ns;
    wait;
  end process;

END;
```

- Save the file and under Simulation option double click on Run Simulation and choose Run Behavioral Simulation



- Behavioral Simulation window appears showing the, waveform of 2x1 MUX for different values of input.



EXPERIMENT 15

AIM: To design 4x1 MUX using 'SELECT when' statement.

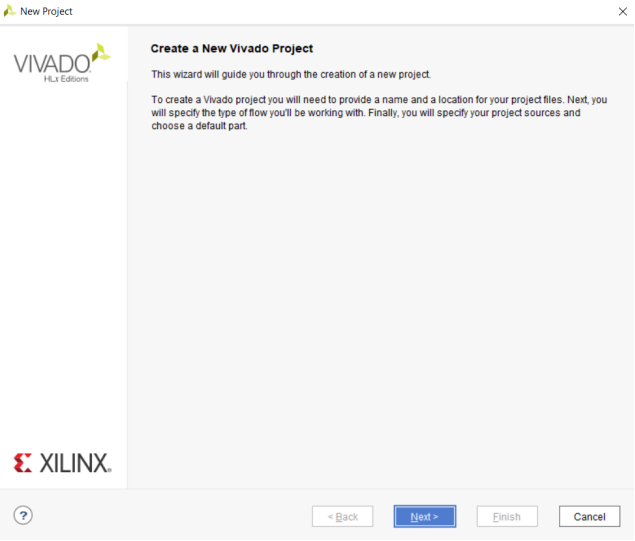
Apparatus Required: Vivado Design Suite.

Procedure:

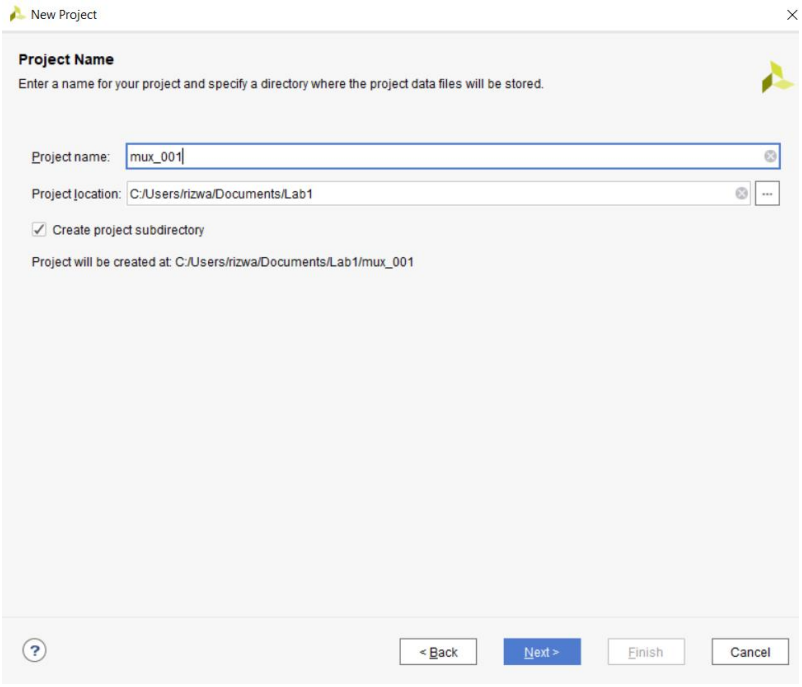
1) **To create a 4x1 MUX:**

- Open *Vivado Design Suite*
- click on *Create Project* option.

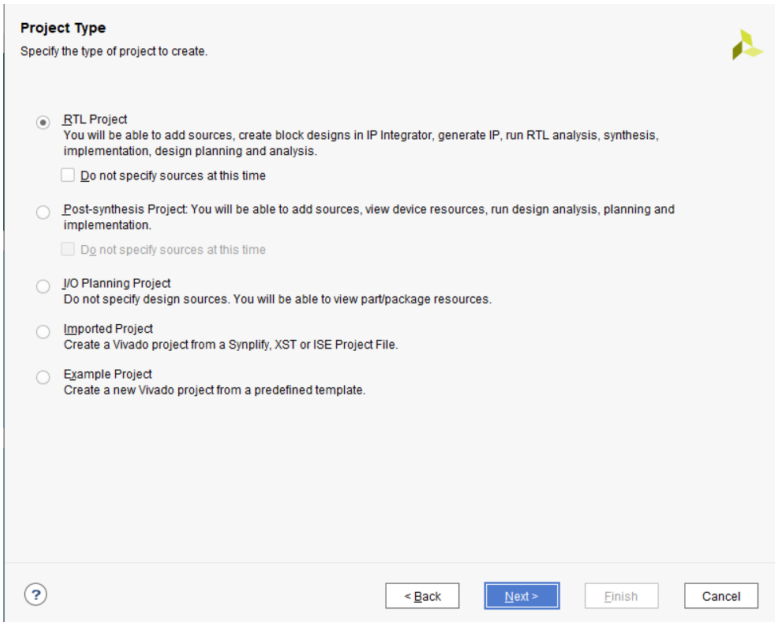
- Click next on Create Project window



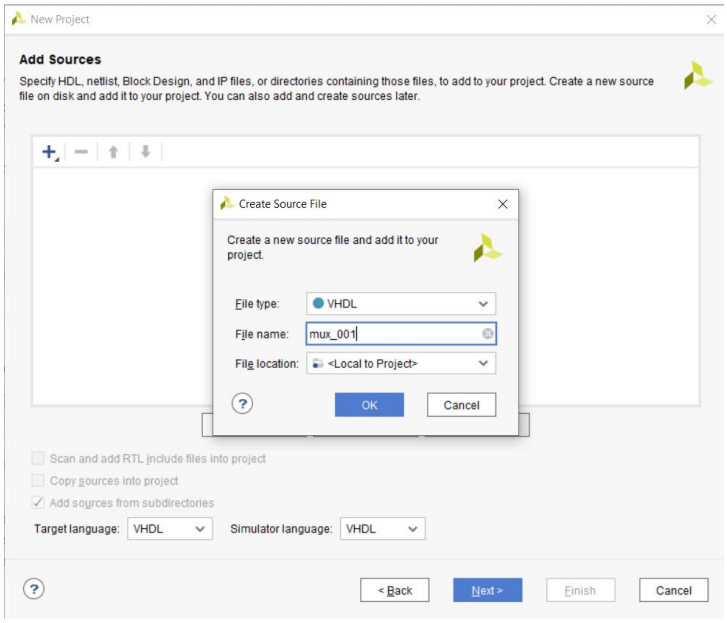
- Name the project "mux_003" and select the location to save the project, then click next



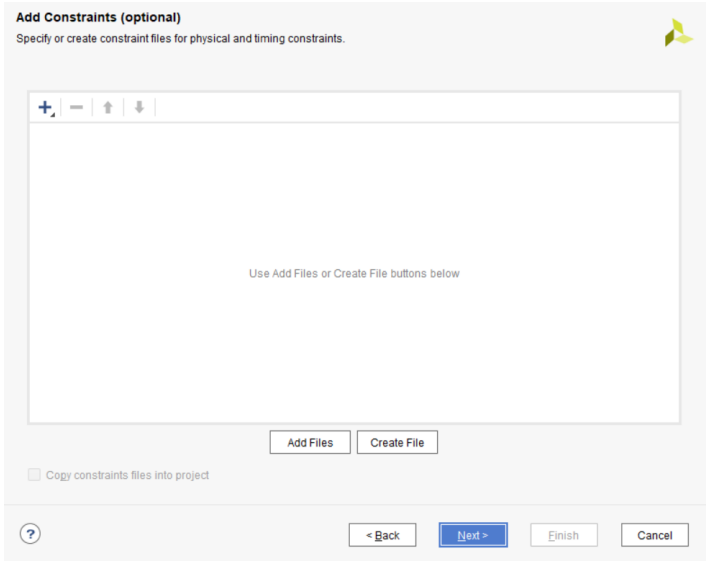
- In Project Type window select *RTL Project* and click next.



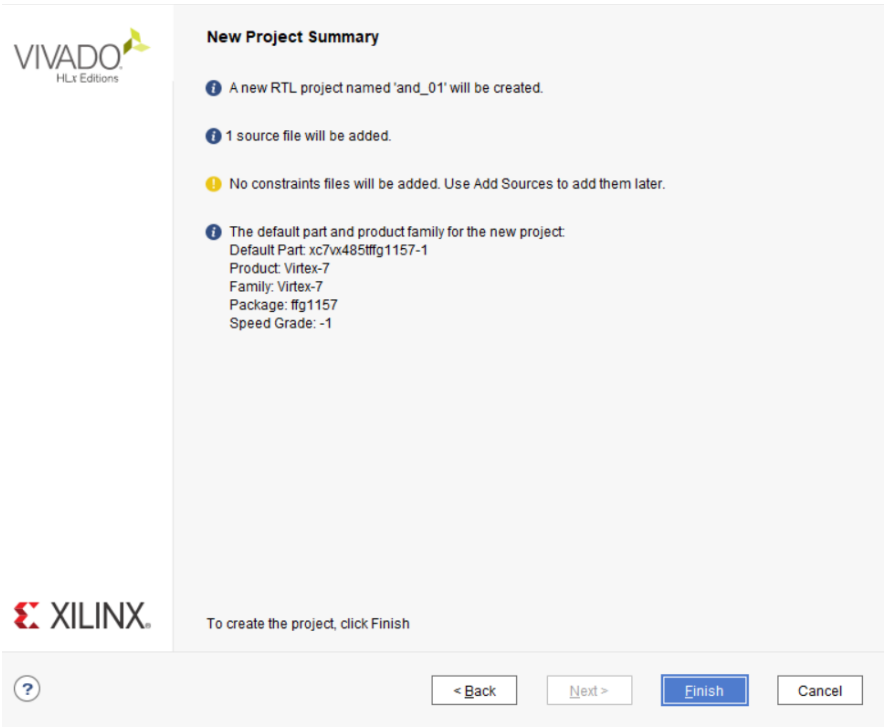
- In Add Sources window click *Create File*, then select file type as *VHDL* and name the file “mux_003”, click OK. Make sure *VHDL* is selected as Target language and Simulator Language and click Next



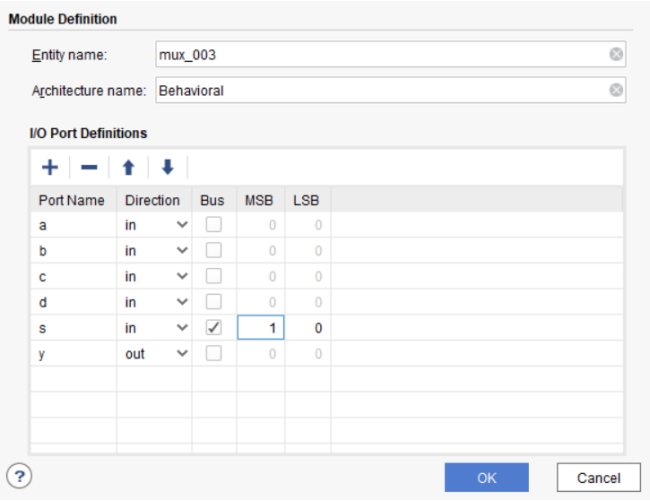
- Click next on the Add constraints window and in the Default Part window select the board available.



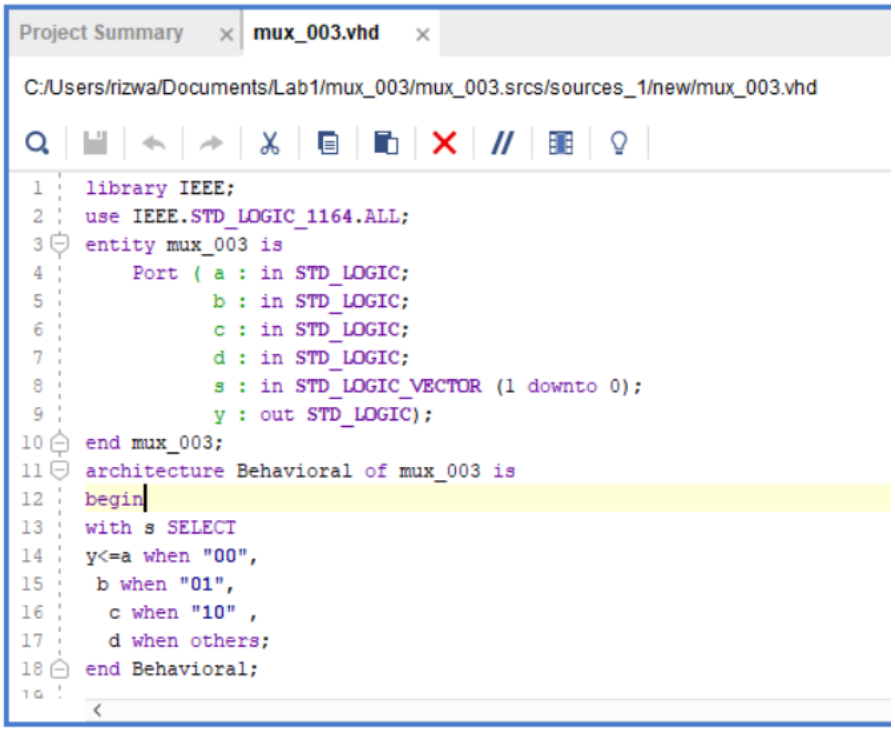
- Click Finish to create project.



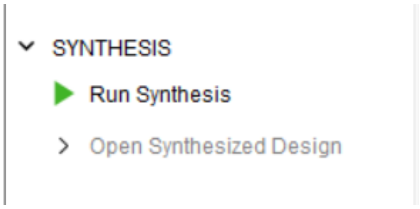
- Define module window appears, in the I/O Definitions tab of the window define the input and output ports for 4x1 MUX and click OK.



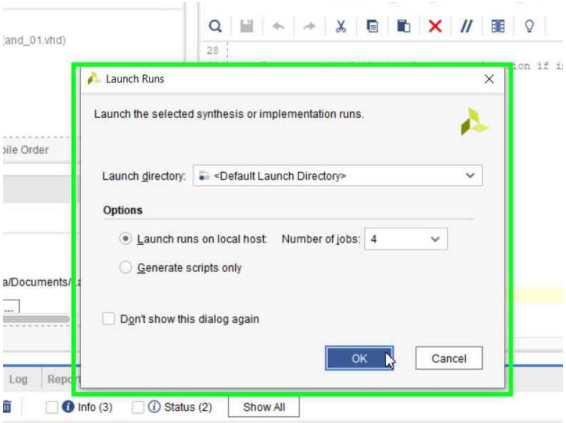
- In the mux_003.vhd file add the following code to define (Behavioral) the gate and save it by pressing ctrl + s:



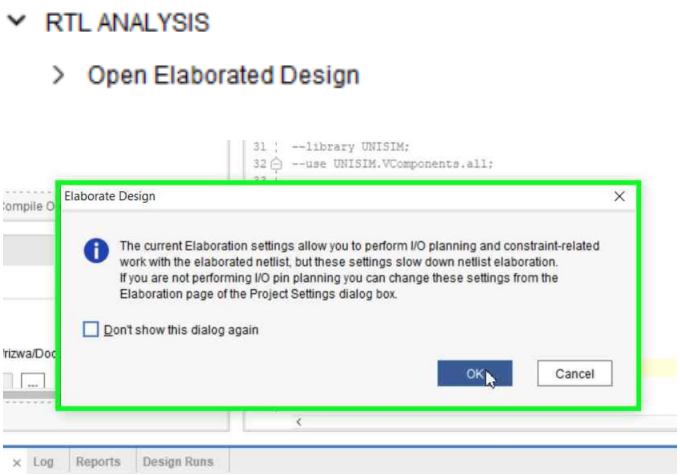
- To synthesize the design, click on Run Synthesis:



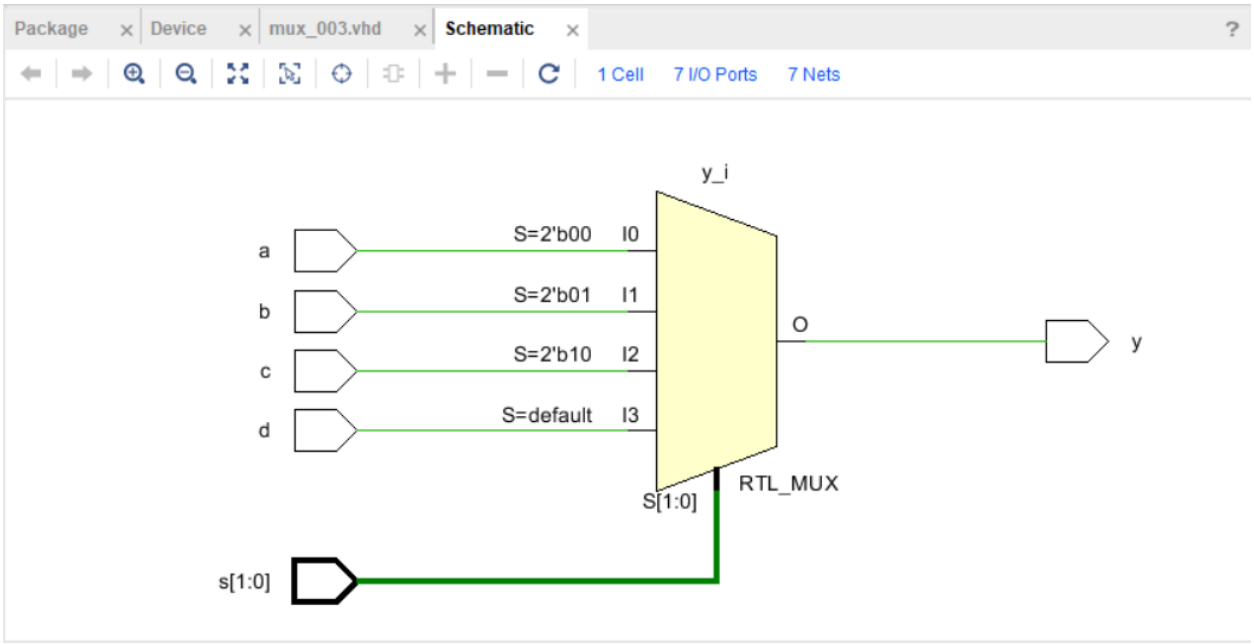
- Then click on OK in Launch Run window



- Close the synthesis completed dialogue, and then Click on Open Elaborated Design and Click OK

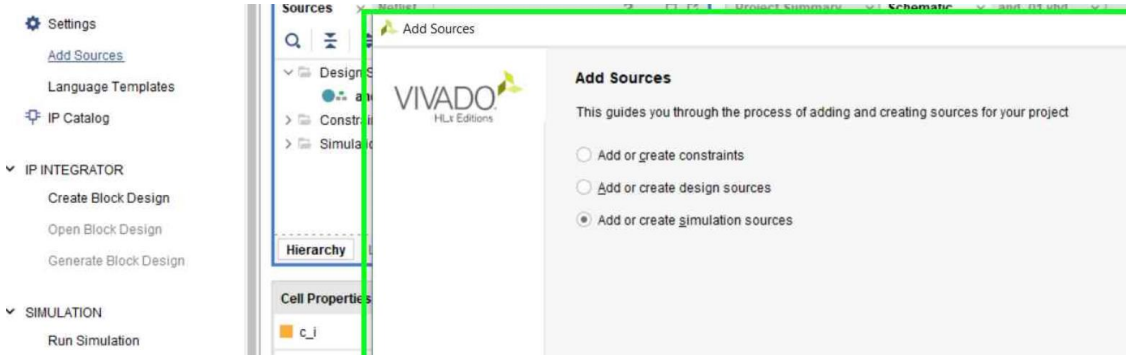


- This shows the schematic of the design

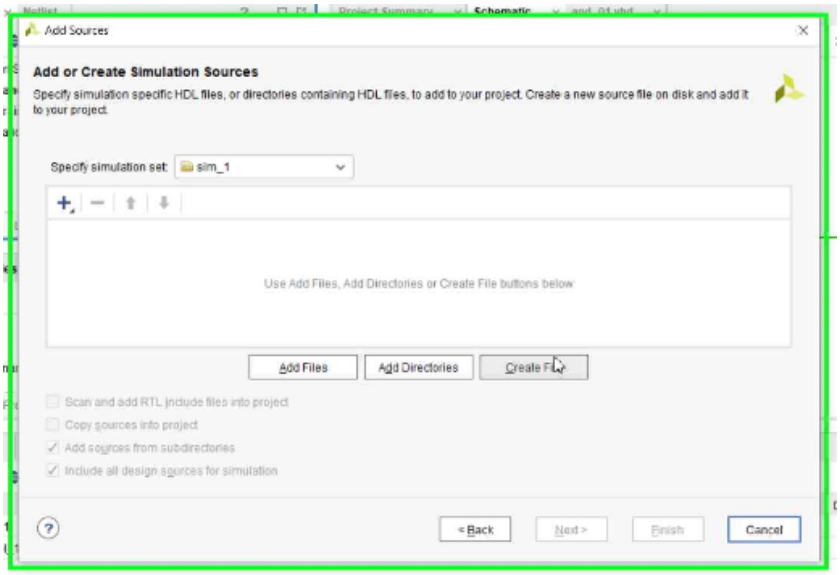


2) To Create a Test Bench:

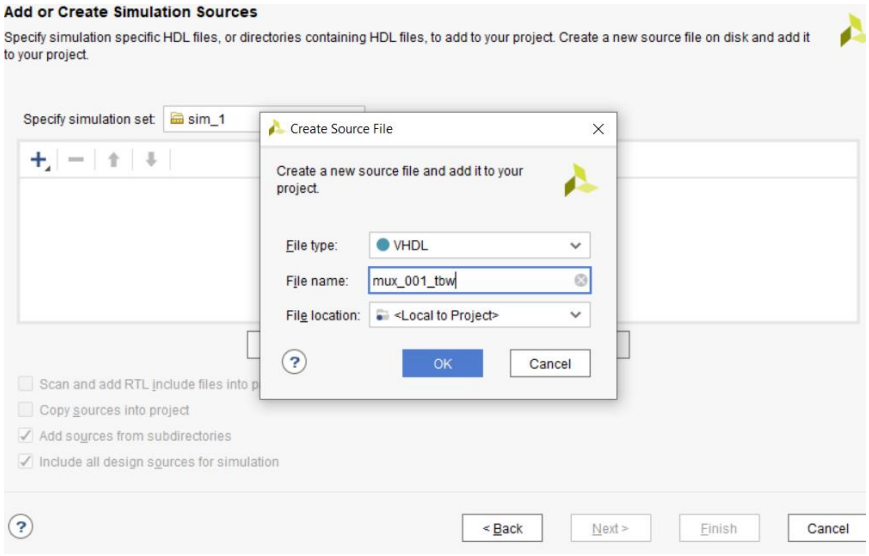
- Click on Add Source and choose Add or Create Simulation Source, then click next.



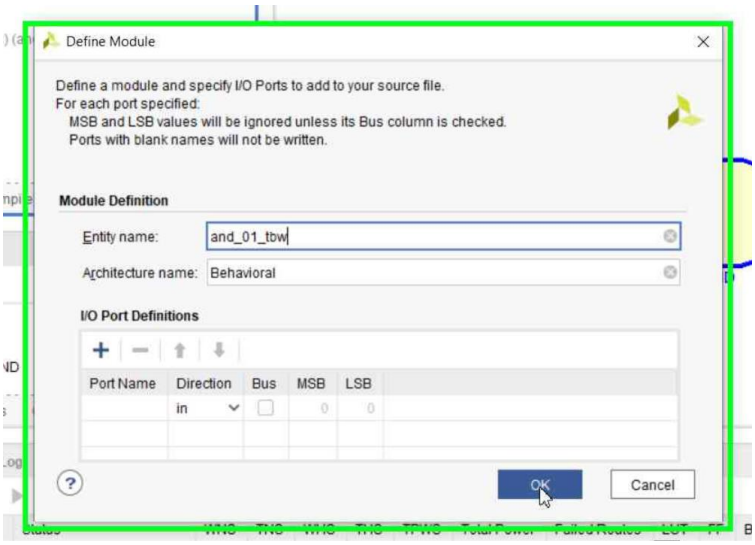
- In Add or Create Simulation Source Window click on Create File



- Select File type as VHDL and name the file “mux_003_tbw”. Then click OK and then Finish.



- In Define module window click OK and if the dialogue pops up click on YES.



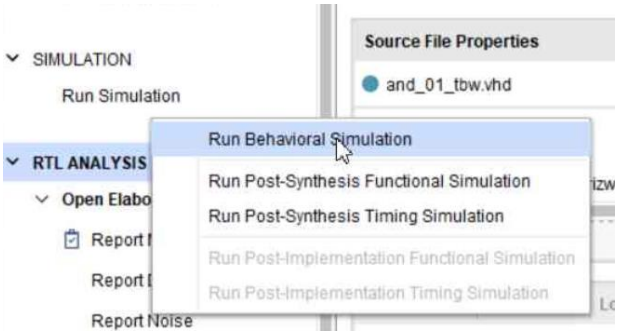
- Under sources window, double click on “mux_003_tbw” file.

- Then in mux_003_tbw.vhd file add the following code:

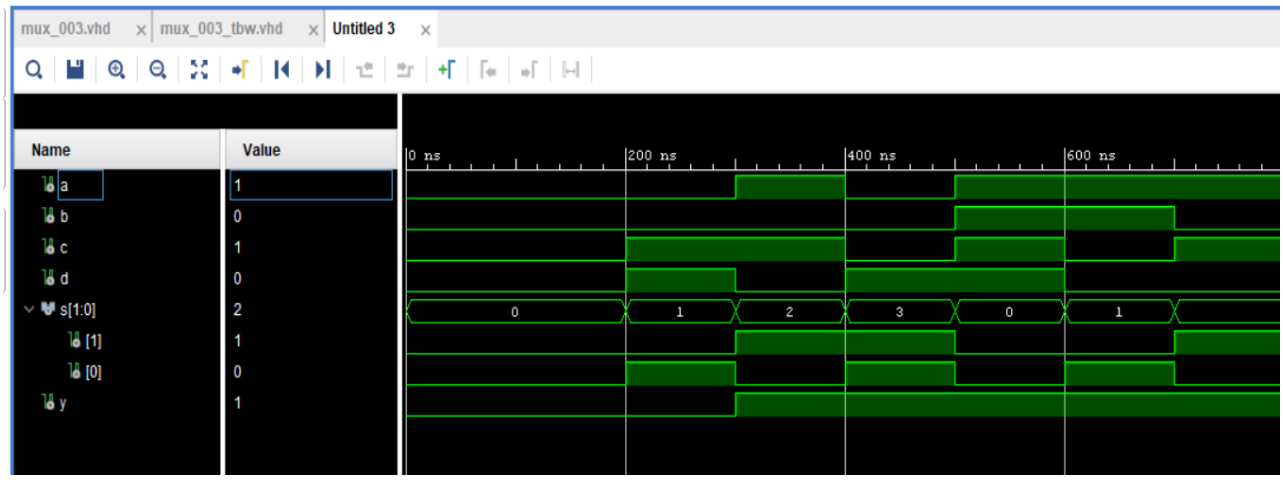
```
library IEEE;
use IEEE.std_logic_1164.ALL;
ENTITY mux_003_tbw IS
END mux_003_tbw;
ARCHITECTURE behavior OF mux_003_tbw IS
    -- Component Declaration for the Unit Under Test (UUT)
    component mux_003 IS
    PORT(
         a : IN  std_logic;
         b : IN  std_logic;
         c : IN  std_logic;
         d : IN  std_logic;
         s : IN  std_logic_vector (1 downto 0);
         y : OUT  std_logic
        );
    END COMPONENT;
   --Inputs
   signal a : std_logic := '0';
   signal b : std_logic := '0';
   signal c : std_logic := '0';
   signal d : std_logic := '0';
   signal s : std_logic_vector (1 downto 0) := "00";
     --Outputs
   signal y : std_logic;
BEGIN
      -- Instantiate the Unit Under Test (UUT)
   uut: mux_003 PORT MAP (
          a => a,
          b => b,
          c => c,
          d => d,
          s =>s,
          y => y
        );
   -- Stimulus process
   stim_proc: process
   begin
      -- hold reset state for 100 ns.
      wait for 100 ns;
      a<='0';
      b<='0';
      c<='0';
      d<='0';
      s<="00";
      wait for 100 ns;
```

```
    a<='0';
    b<='0';
    c<='1';
    d<='1';
    s<="01";
    wait for 100 ns;
    a<='1';
    b<='0';
    c<='1';
    d<='0';
    s<="10";
    wait for 100 ns;
    a<='0';
    b<='0';
    c<='0';
    d<='1';
    s<="11";
    wait for 100 ns;
    a<='1';
    b<='1';
    c<='1';
    d<='1';
    s<="00";
    wait for 100 ns;
    a<='1';
    b<='1';
    c<='0';
    d<='0';
    s<="01";
    wait for 100 ns;
    a<='1';
    b<='0';
    c<='1';
    d<='0';
    s<="10";
    wait;
  end process;

END;
```

- Save the file and under Simulation option double click on Run Simulation and choose Run Behavioral Simulation



- Behavioral Simulation window appears showing the, waveform of 4x1 MUX for different values of input.



EXPERIMENT 16

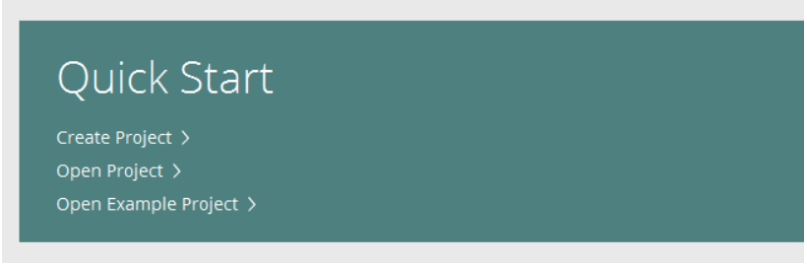
AIM: To design ALU to perform 8 operations

Apparatus Required: Vivado Design Suite.

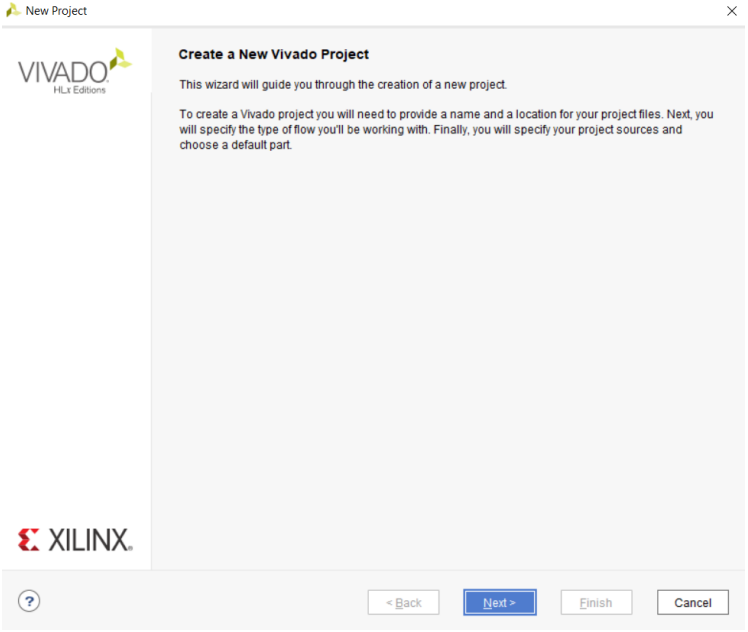
Procedure:

1) **To create an ALU:**

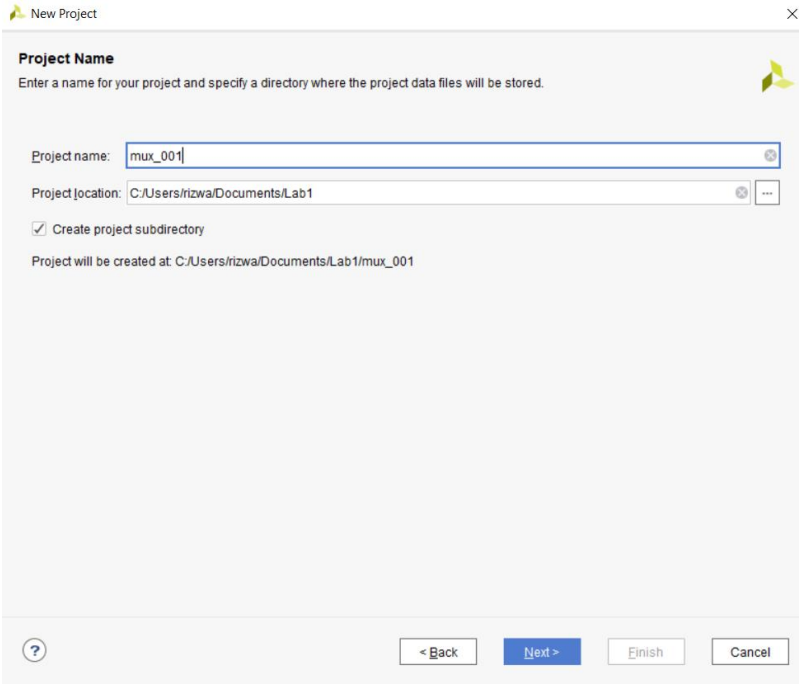
- Open *Vivado Design Suite*
- click on *Create Project* option.



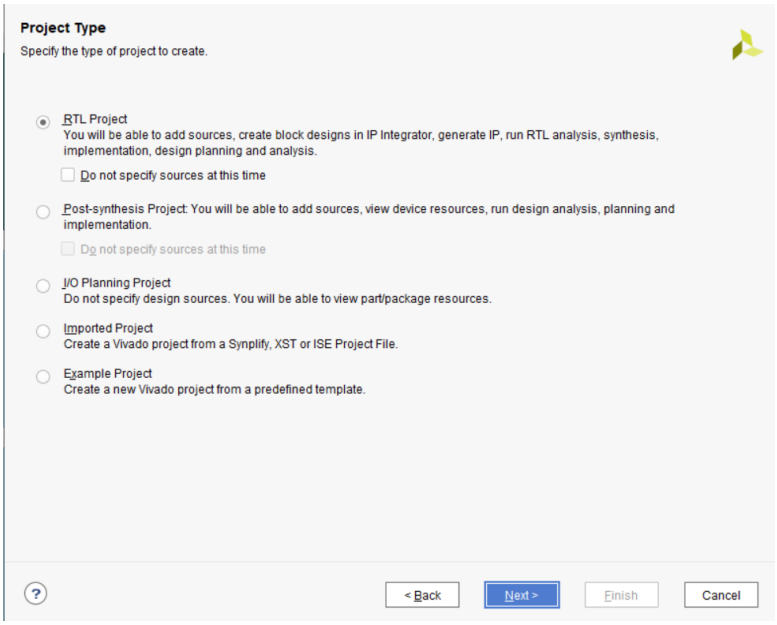
- Click next on Create Project window



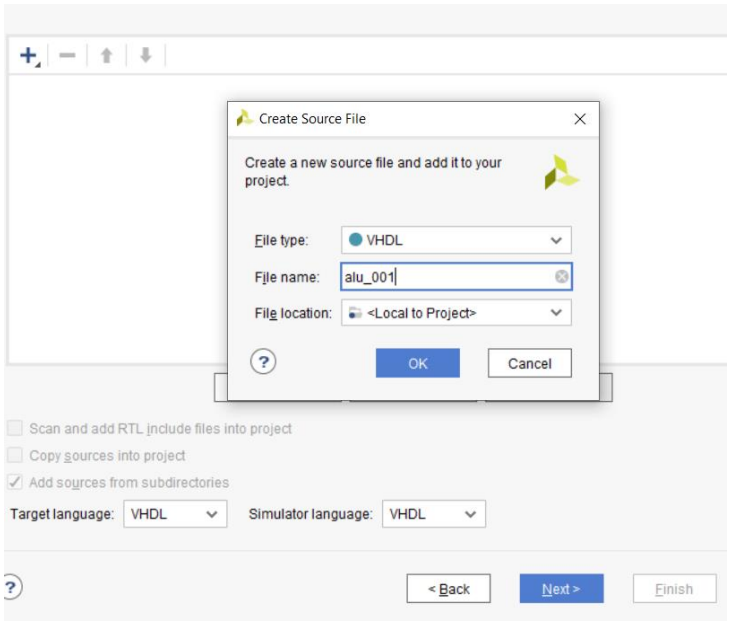
- Name the project “alu_01” and select the location to save the project, then click next



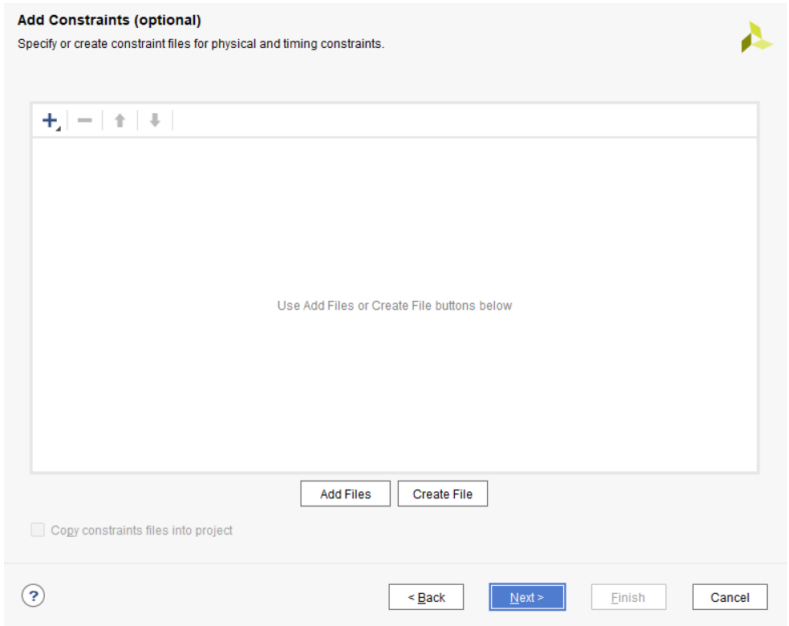
- In Project Type window select *RTL Project* and click next.



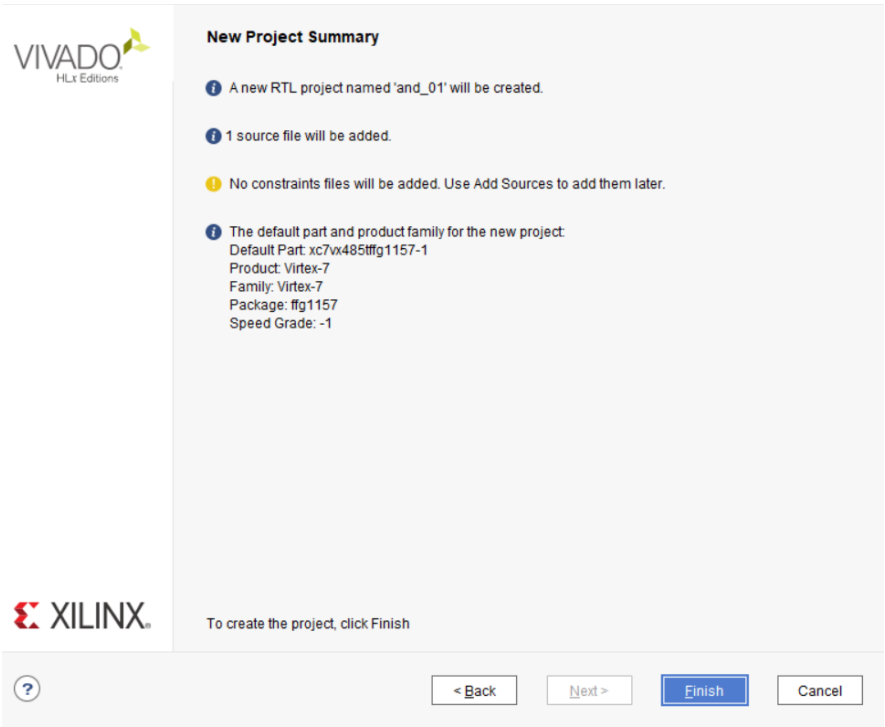
- In Add Sources window click *Create File*, then select file type as *VHDL* and name the file “alu_001”, click OK. Make sure *VHDL* is selected as Target language and Simulator Language and click Next



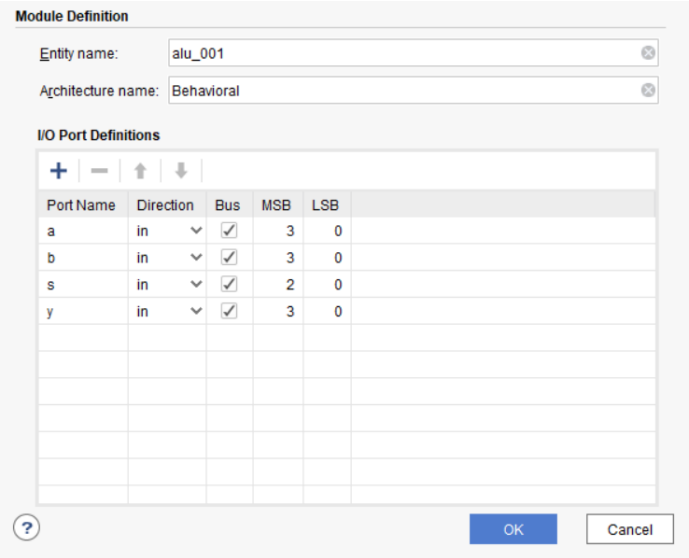
- Click next on the Add constraints window and in the Default Part window select the board available.



- Click Finish to create project.



- Define module window appears, in the I/O Definitions tab of the window define the input and output ports for ALU and click OK.



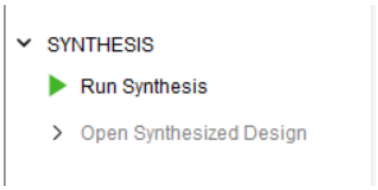
- In the alu_001.vhd file add the following code to define (Behavioral) the gate and save it by pressing ctrl + s:

Project Summary | alu_001.vhd

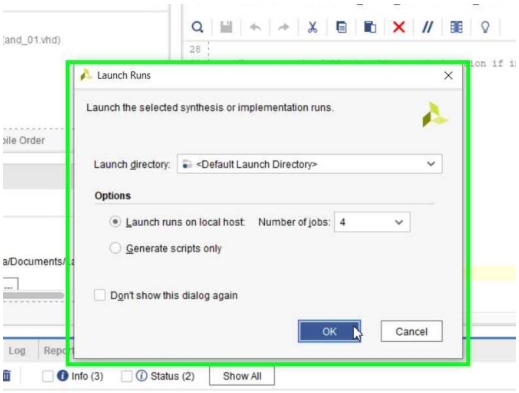
C:/Users/rizwa/Documents/Lab1/alu_001/alu_001.srcs/sources_1/new/alu_001.vhd

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3  entity alu_001 is
4      Port ( a : in STD_LOGIC_VECTOR (3 downto 0);
5             b : in STD_LOGIC_VECTOR (3 downto 0);
6             s : in STD_LOGIC_VECTOR (2 downto 0);
7             y : out STD_LOGIC_VECTOR (3 downto 0));
8  end alu_001;
9  architecture Behavioral of alu_001 is
10 begin
11 with s SELECT
12 y<= a and b when "000",
13  a or b when "001",
14  not a when "010",
15  a xor b when "011",
16  a nand b when "100",
17  a nor b when "101",
18  not b when others;
19 end Behavioral;
```

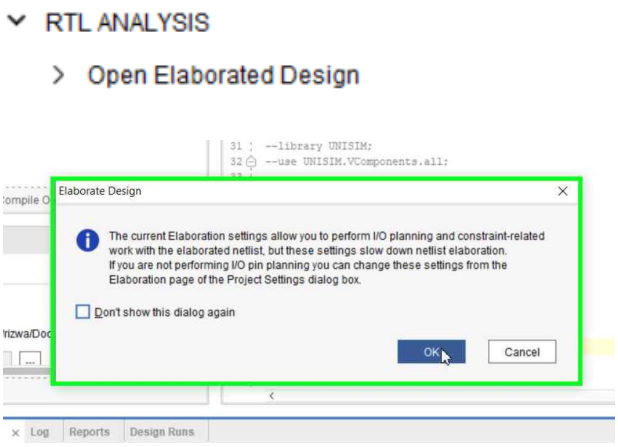
- To synthesize the design, click on Run Synthesis:



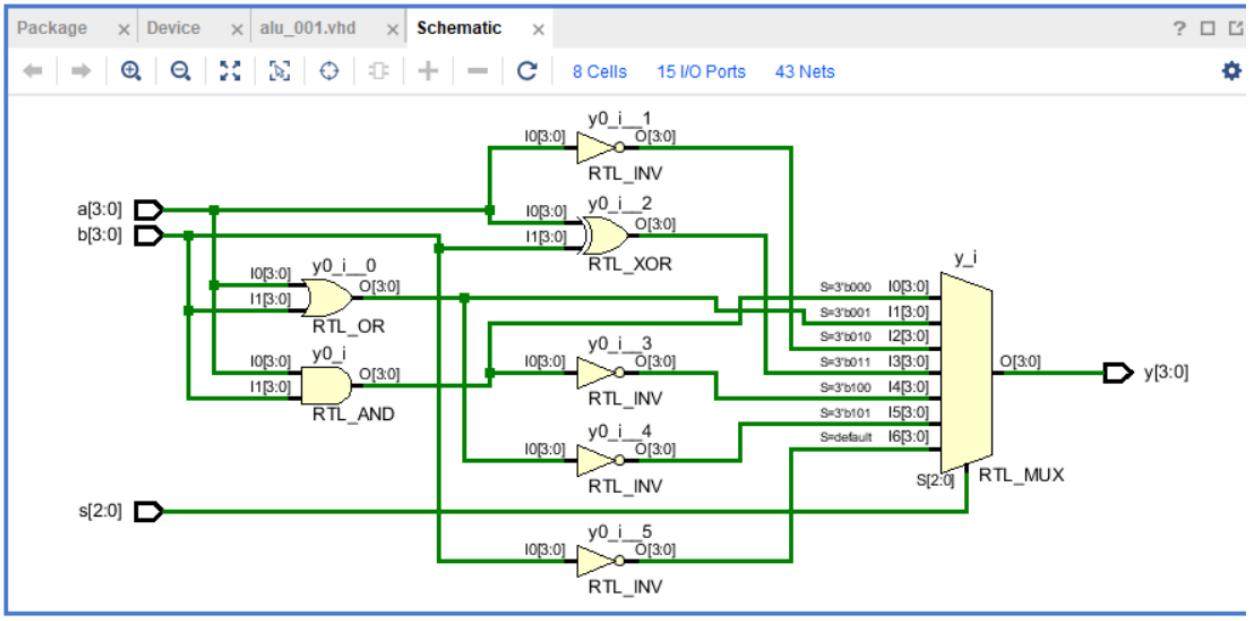
- Then click on OK in Launch Run window



- Close the synthesis completed dialogue, and then Click on Open Elaborated Design and Click OK

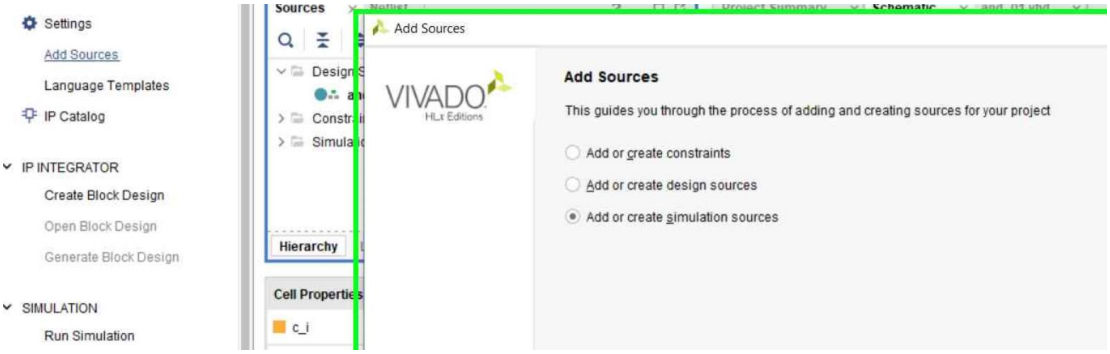


- This shows the schematic of the design

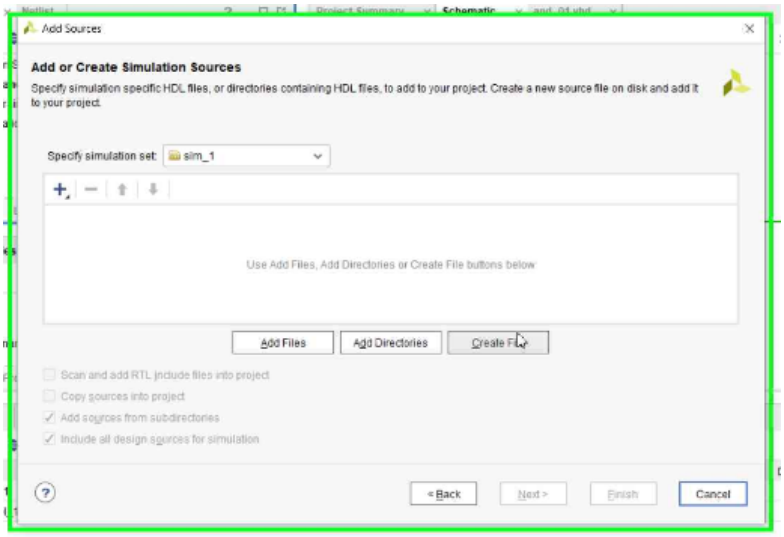


2) To Create a Test Bench:

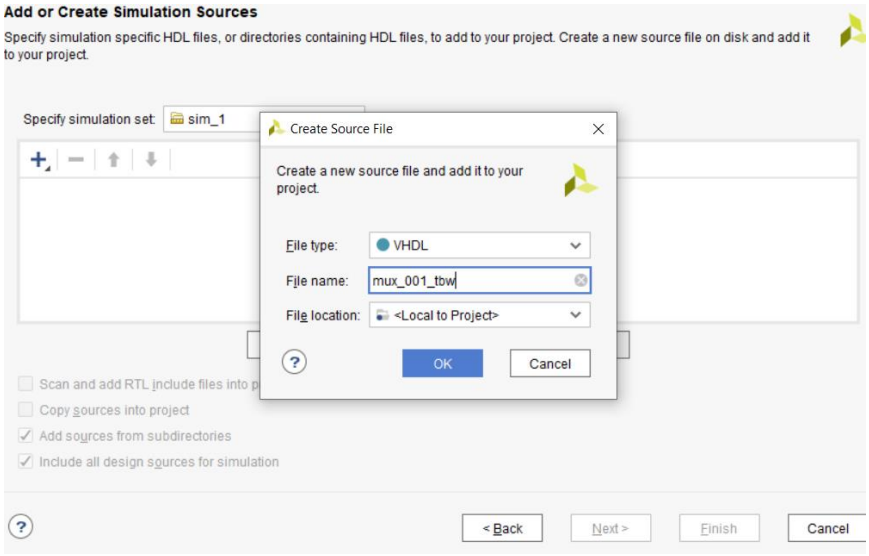
- Click on Add Source and choose Add or Create Simulation Source, then click next.



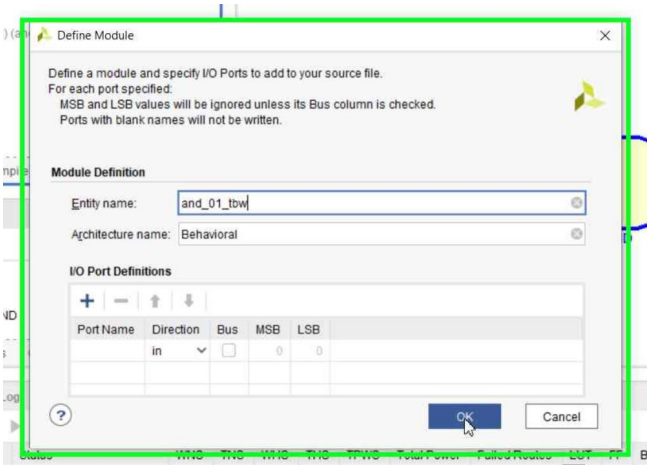
- In Add or Create Simulation Source Window click on Create File



- Select File type as VHDL and name the file “alu_001_tbw”. Then click OK and then Finish.



- In Define module window click OK and if the dialogue pops up click on YES.



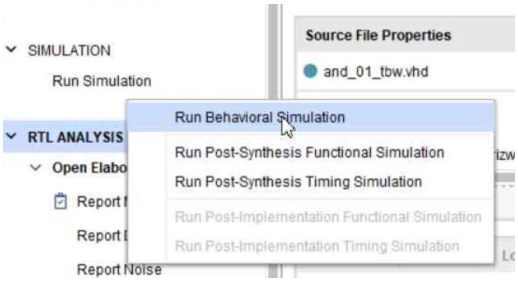
- Under sources window, double click on “alu_001_tbw” file.

- Then in alu_001_tbw.vhd file add the following code:

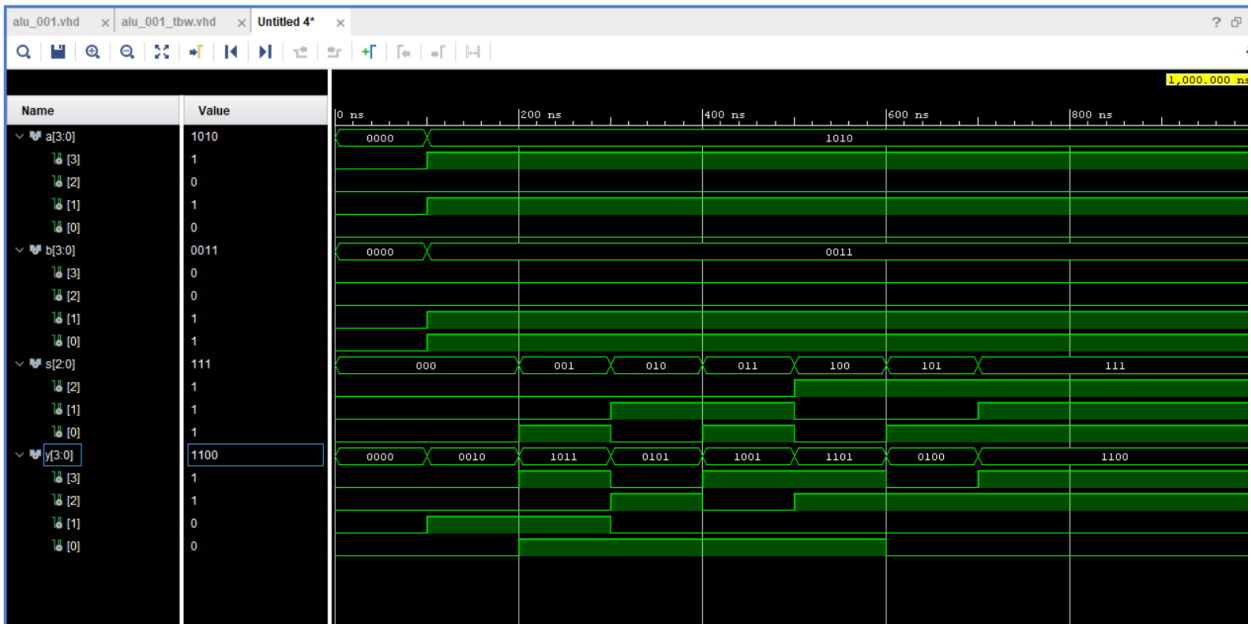
```
library IEEE;
use IEEE.std_logic_1164.ALL;
ENTITY alu_001_tbw IS
END alu_001_tbw;
ARCHITECTURE behavior OF alu_001_tbw IS
    component alu_001 IS
    PORT(
         a : IN  std_logic_vector(3 downto 0);
         b : IN  std_logic_vector (3 downto 0);
         s : IN std_logic_vector (2 downto 0);
         y : OUT  std_logic_vector (3 downto 0)
        );
    END COMPONENT;
   signal a : std_logic_vector(3 downto 0) := "0000";
   signal b : std_logic_vector(3 downto 0) := "0000";
   signal s : std_logic_vector(2 downto 0) := "000";
     --Outputs
   signal y : std_logic_vector(4 downto 0);
BEGIN
   uut: alu_001 PORT MAP (
          a => a,
          b => b,
          s =>s,
          y=> y);
   stim_proc: process
   begin
      wait for 100 ns;
      a<="1010";
      b<="0011";
      s<="000";
      wait for 100 ns;
      a<="1010";
      b<="0011";
      s<="001";
      wait for 100 ns;
      a<="1010";
      b<="0011";
      s<="010";
      wait for 100 ns;
      a<="1010";
      b<="0011";
      s<="011";
      wait for 100 ns;
      a<="1010";
      b<="0011";
```

```
    s<="100";
    wait for 100 ns;
    a<="1010";
    b<="0011";
    s<="101";
    wait for 100 ns;
    a<="1010";
    b<="0011";
    s<="111";
    wait;
  end process;
END;
```

- Save the file and under Simulation option double click on Run Simulation and choose Run Behavioral Simulation



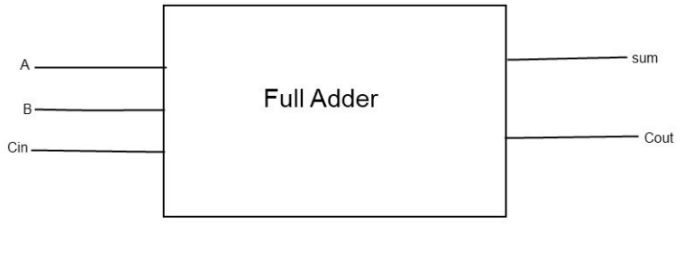
- Behavioral Simulation window appears showing the, waveform of ALU for different values of input.



EXPERIMENT 17

AIM: To design Fulladder cell.

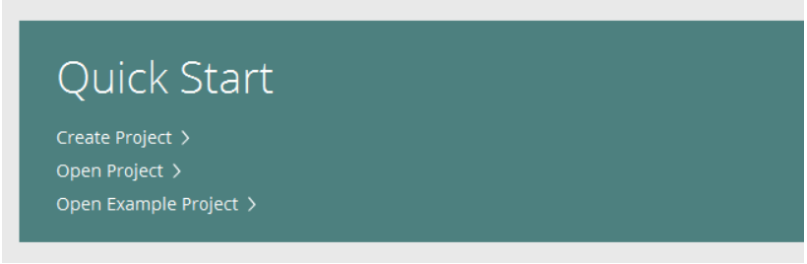
Apparatus Required: Vivado Design Suite.



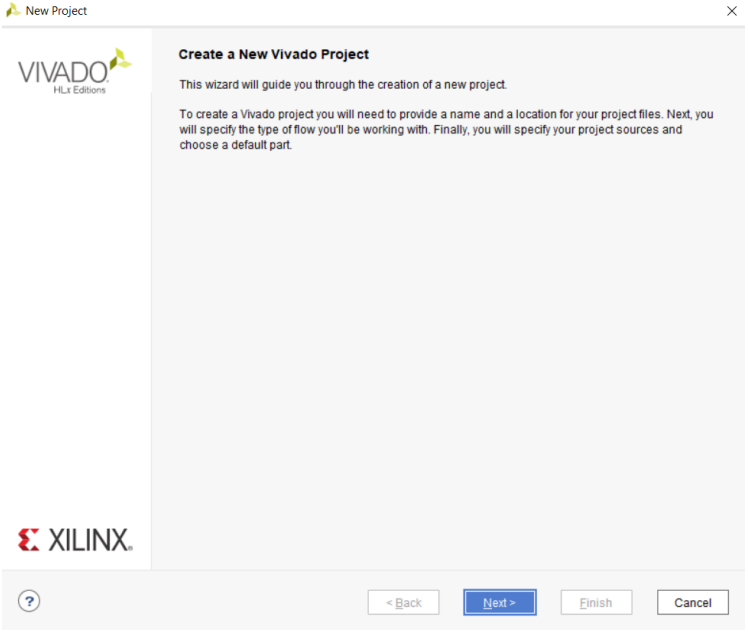
Procedure:

3) To create an FULLADDER:

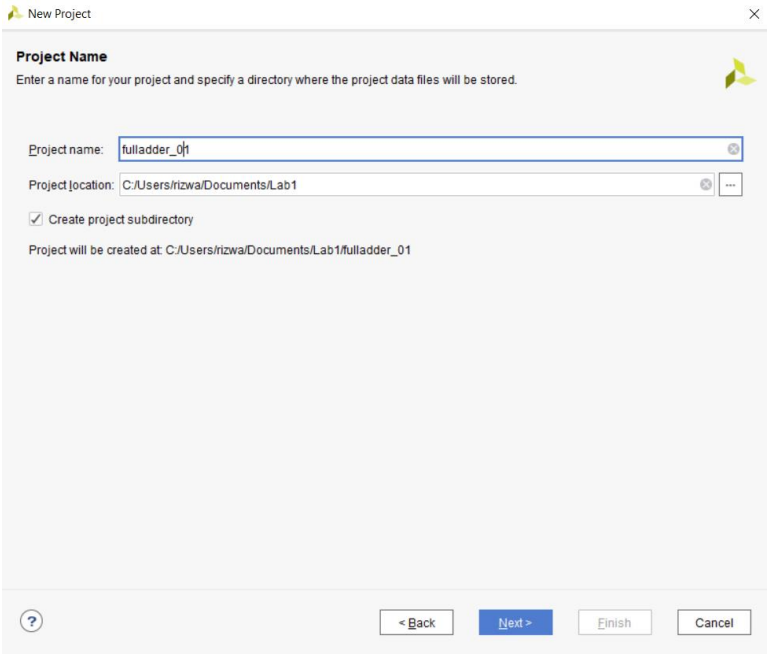
- Open *Vivado Design Suite*
- click on *Create Project* option.



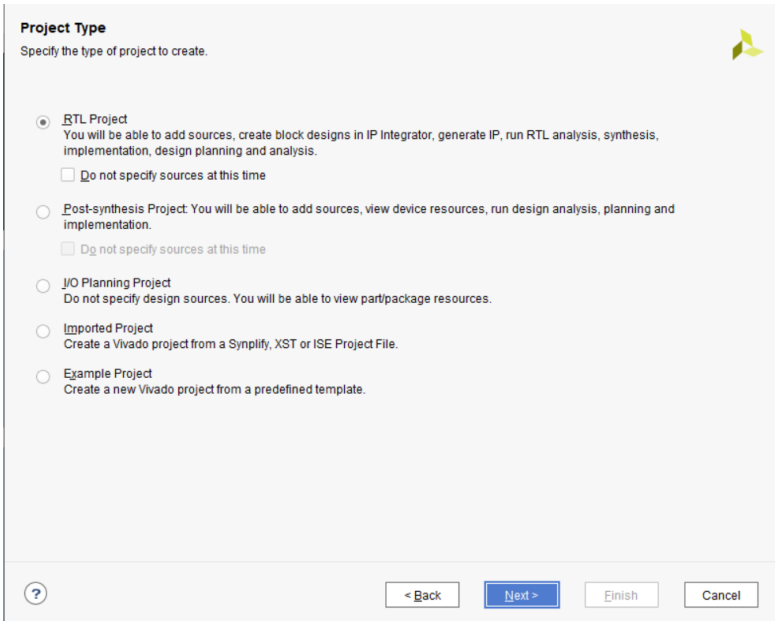
- Click next on Create Project window



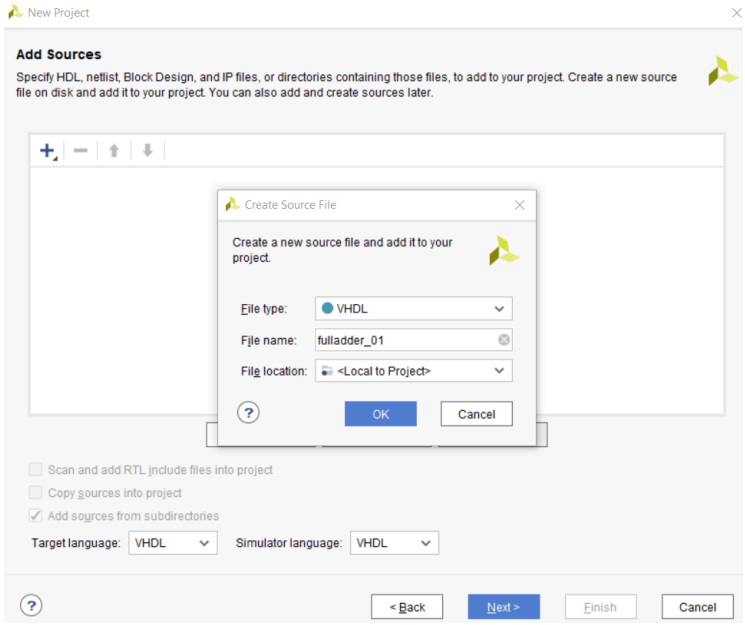
- Name the project “fulladder_01” and select the location to save the project, then click next



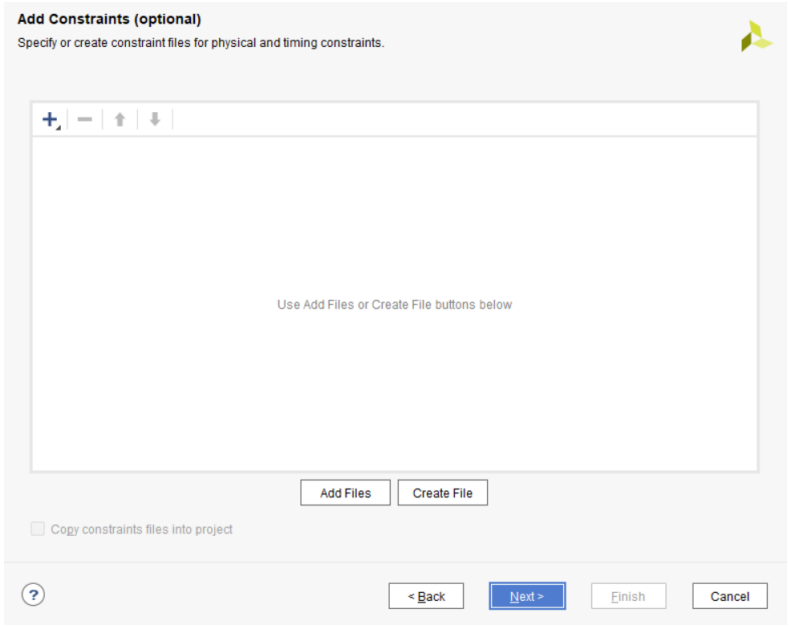
- In Project Type window select *RTL Project* and click next.



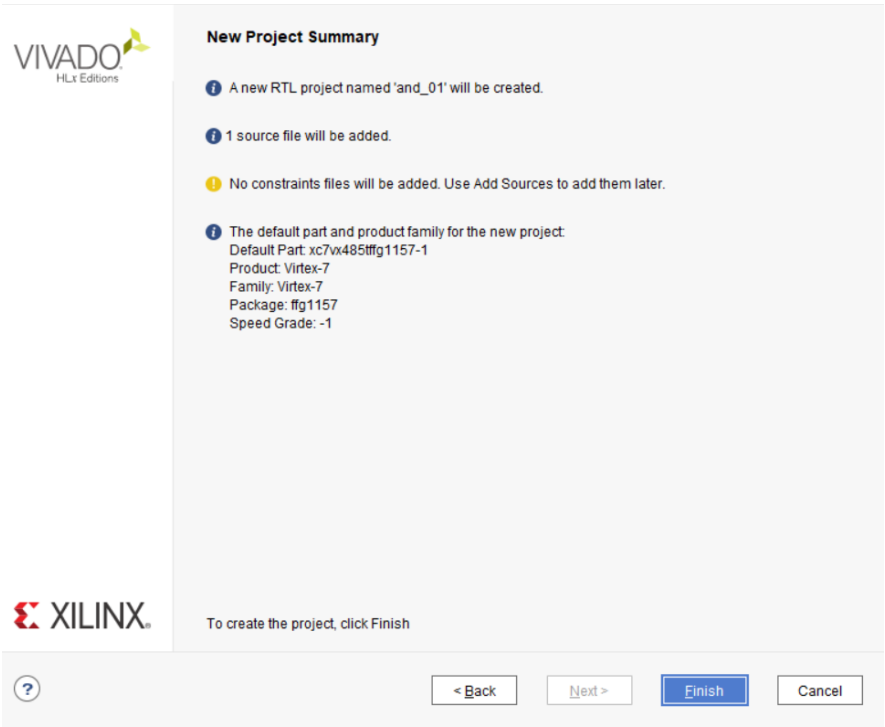
- In Add Sources window click *Create File*, then select file type as *VHDL* and name the file “fulladder_01”, click OK. Make sure *VHDL* is selected as Target language and Simulator Language and click Next



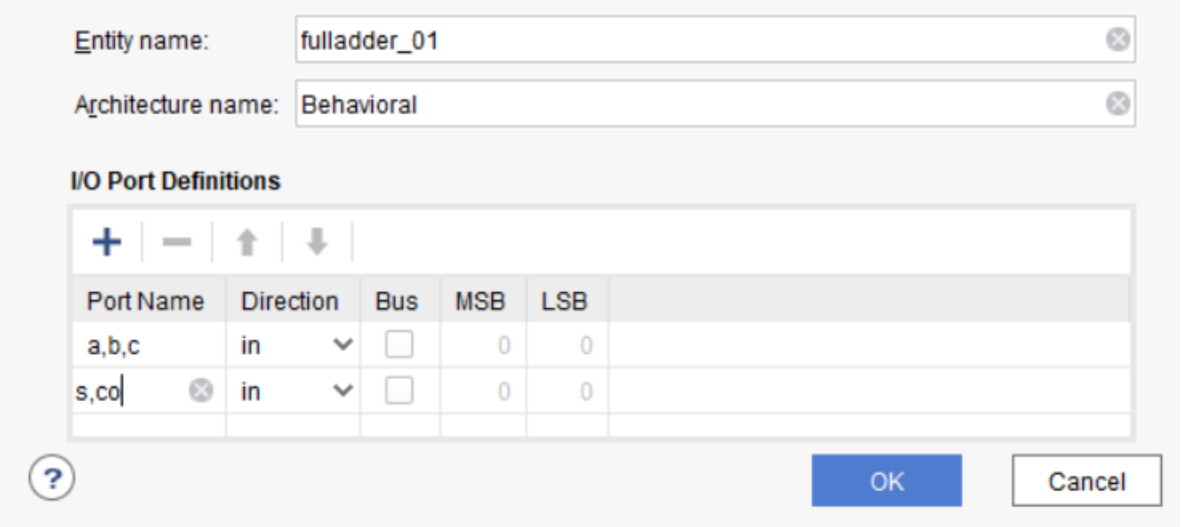
- Click next on the Add constraints window and in the Default Part window select the board available.



- Click Finish to create project.



- Define module window appears, in the I/O Definitions tab of the window define the input and output ports for FULLADDER and click OK.



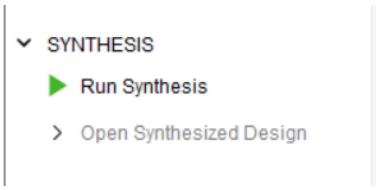
- In the fulladder_01.vhd file add the following code to define (Behavioral) the gate and save it by pressing ctrl + s:

Package | Device | fulladder_01.vhd | Schematic

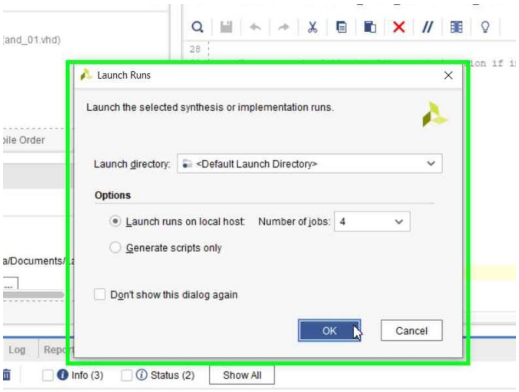
C:/Users/rizwa/Documents/Lab1/fulladder_01/fulladder_01.srcs/sources_1/new/fulladder_01.vhd

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3  entity fulladder_01 is
4      Port ( a, b, c : in STD_LOGIC;
5             s, co : out STD_LOGIC);
6  end fulladder_01;
7
8  architecture Behavioral of fulladder_01 is
9  signal a1 : std_logic;
10 signal a2: std_logic;
11 signal a3: std_logic;
12 begin
13 a1<= a xor b;
14 a2<= a1 and c;
15 a3<= a and b;
16 co <= a2 or a3;
17 s <= a1 xor c;
18 end Behavioral;
19
```

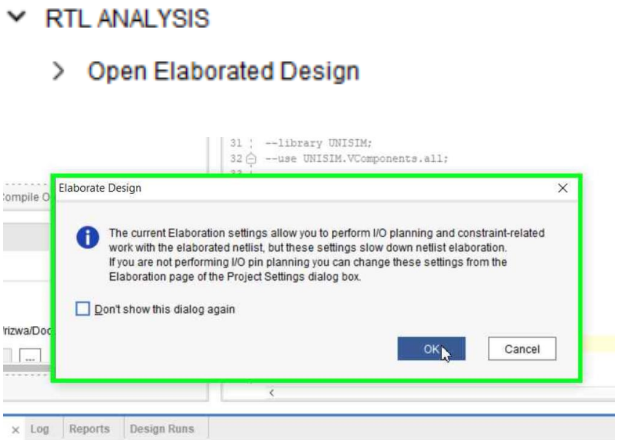
- To synthesize the design, click on Run Synthesis:



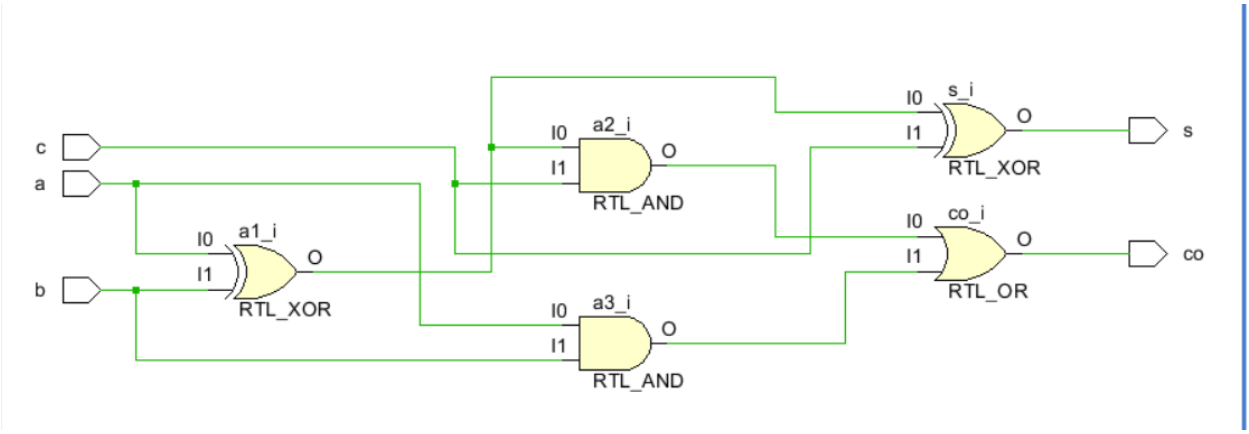
- Then click on OK in Launch Run window



- Close the synthesis completed dialogue, and then Click on Open Elaborated Design and Click OK

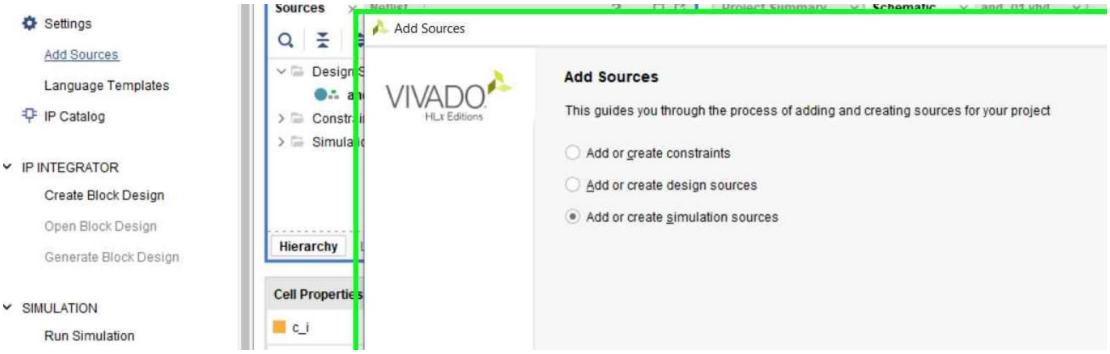


- This shows the schematic of the design

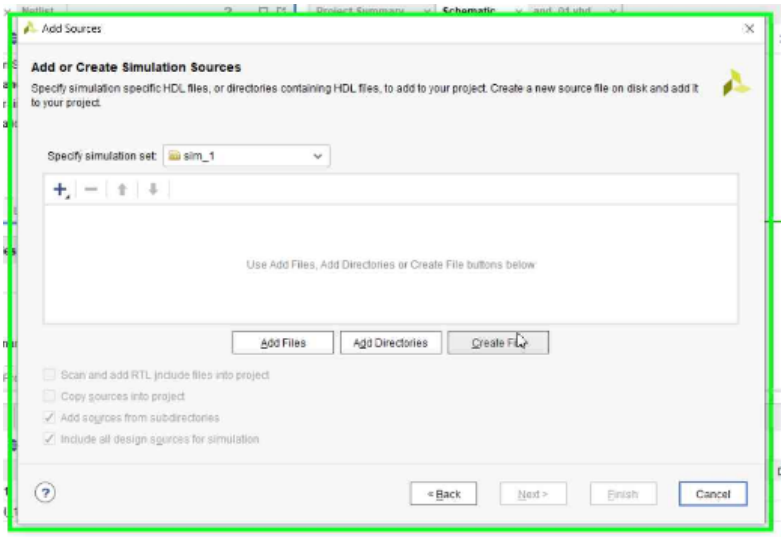


4) To Create a Test Bench:

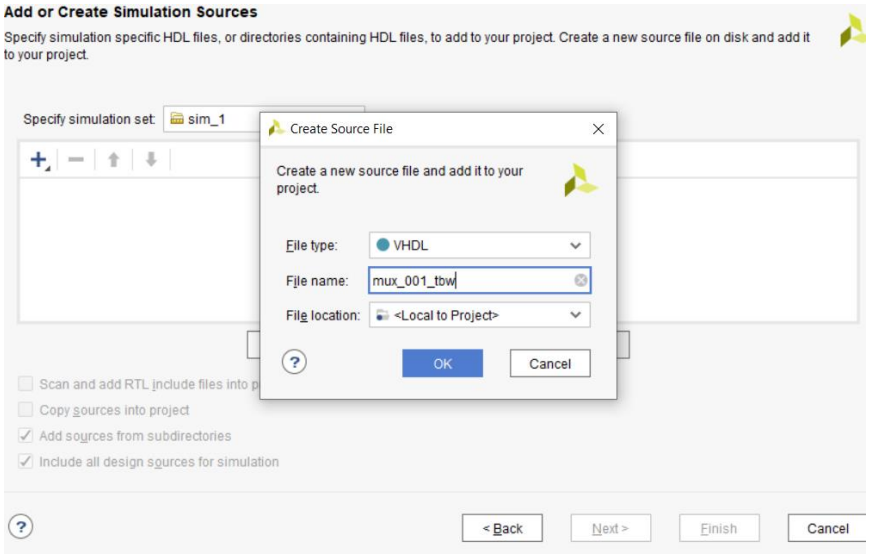
- Click on Add Source and choose Add or Create Simulation Source, then click next.



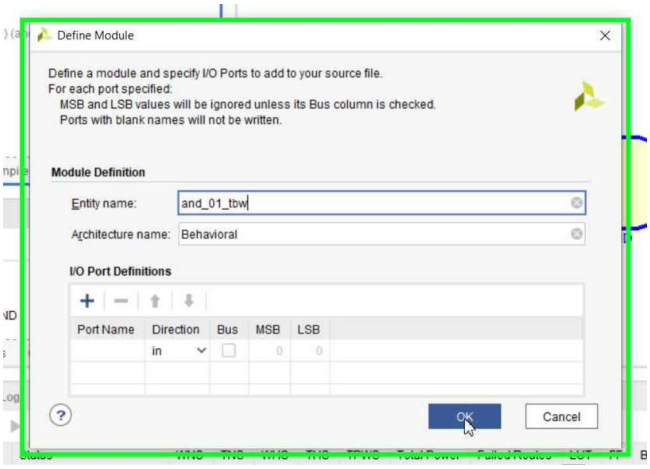
- In Add or Create Simulation Source Window click on Create File



- Select File type as VHDL and name the file “fulladder_01_tbw”. Then click OK and then Finish.



- In Define module window click OK and if the dialogue pops up click on YES.



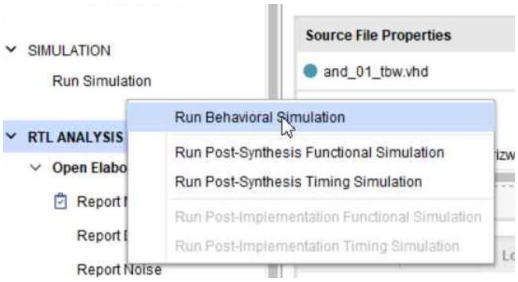
- Under sources window, double click on “fulladder_01_tbw” file.

- Then in fulladder_01_tbw.vhd file add the following code:

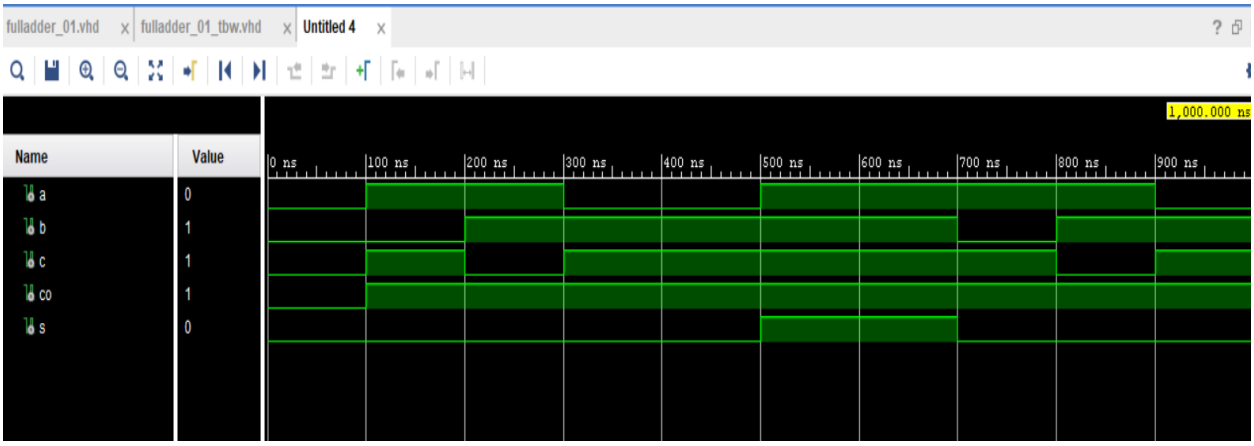
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity fulladder_01_tbw is
end fulladder_01_tbw;
architecture Behavioral of fulladder_01_tbw is
component fulladder_01 is
PORT( a, b, c : in std_logic;
co, s: out std_logic);
end component;
signal a: std_logic:='0';
signal b: std_logic:='0';
signal c: std_logic:='0';
signal co:std_logic;
signal s:std_logic;
begin
uut: fulladder_01 Port Map( a=>a, b=>b, c=>c, s=>s, co=>co);
stim_proc: process
begin
wait for 100ns;
a<='1';
b<='0';
c<='1';
wait for 100ns;
a<='1';
b<='1';
c<='0';
wait for 100ns;
a<='0';
b<='1';
c<='1';
wait for 100ns;
wait for 100ns;
a<='1';
```

```
b<='1';
c<='1';
wait for 100ns;
end process;
end Behavioral;
```

- Save the file and under Simulation option double click on Run Simulation and choose Run Behavioral Simulation

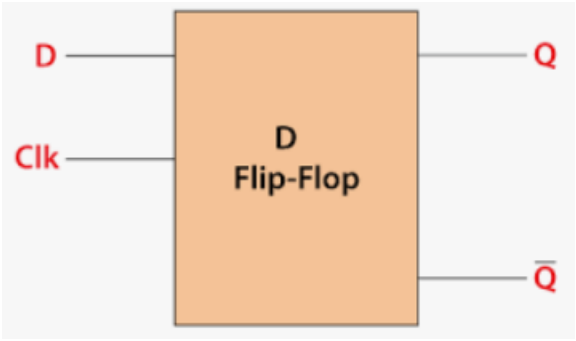


- Behavioral Simulation window appears showing the, waveform of FULLADDER for different values of input.



EXPERIMENT 18

AIM: To design and simulate D filpflop

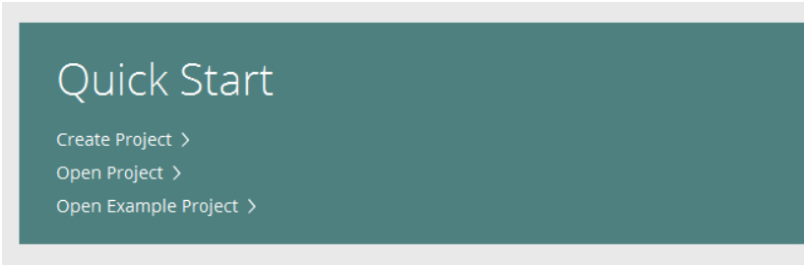


Apparatus Required: Vivado Design Suite.

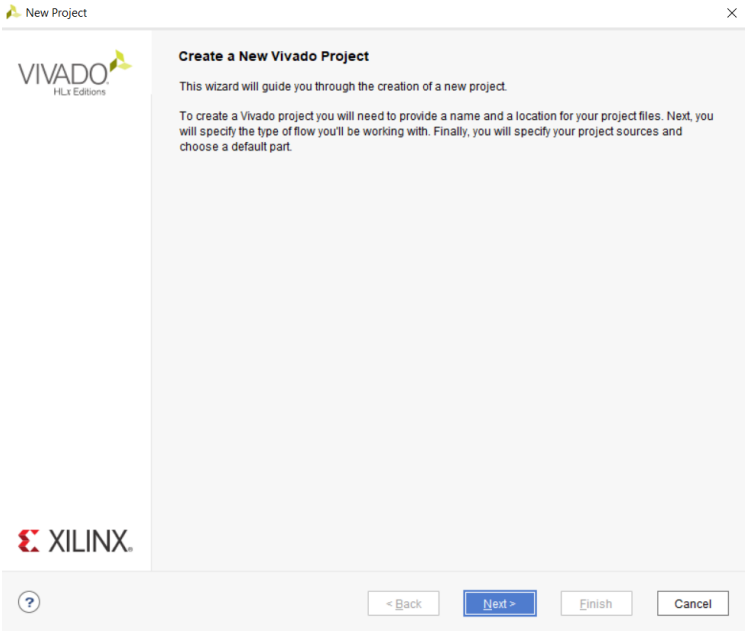
Procedure:

5) **To create a D Flipflop:**

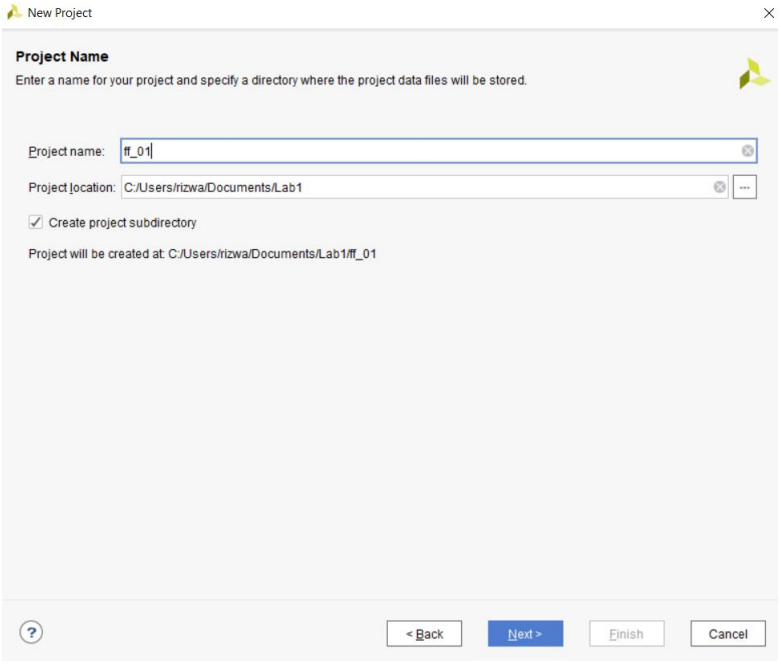
- Open *Vivado Design Suite*
- click on *Create Project* option.



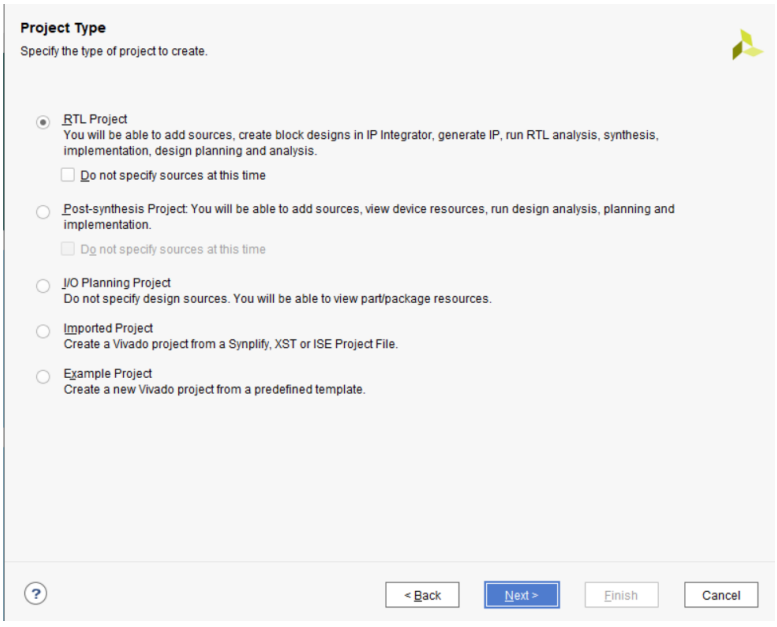
- Click next on Create Project window



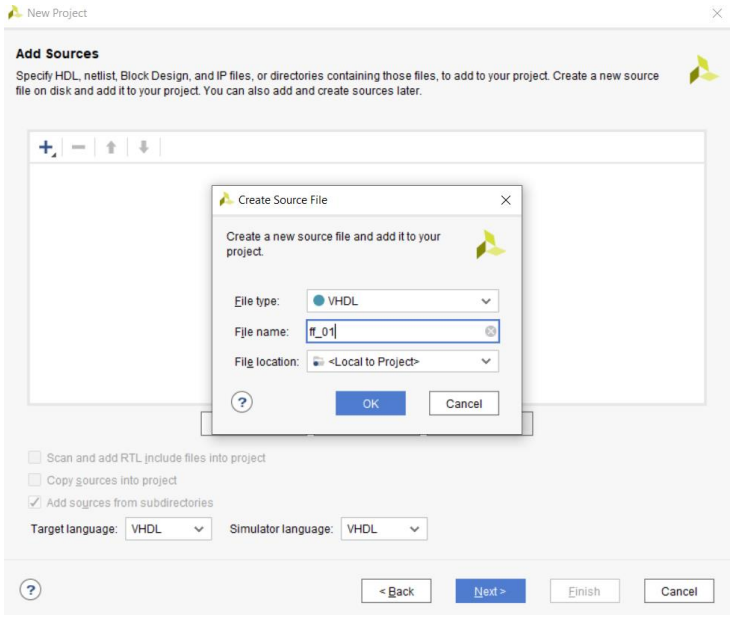
- Name the project “ff_01” and select the location to save the project, then click next



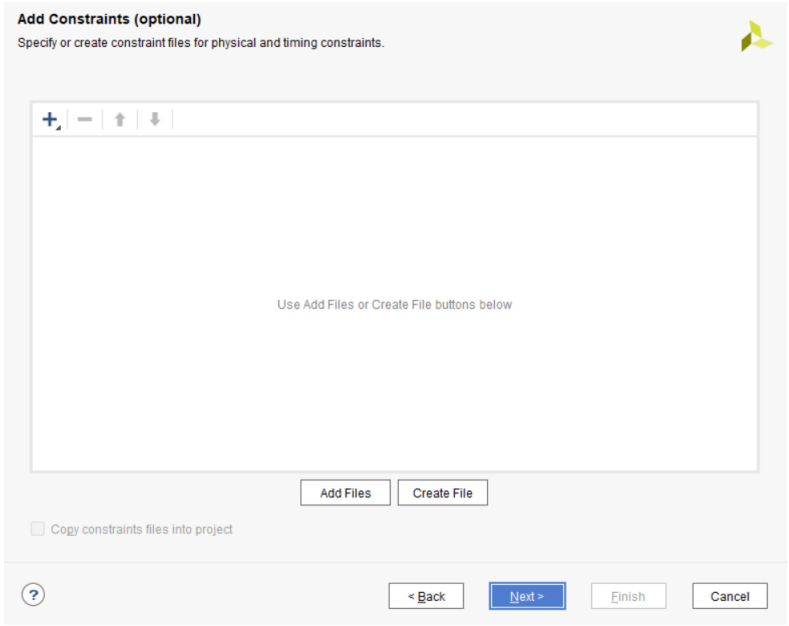
- In Project Type window select *RTL Project* and click next.



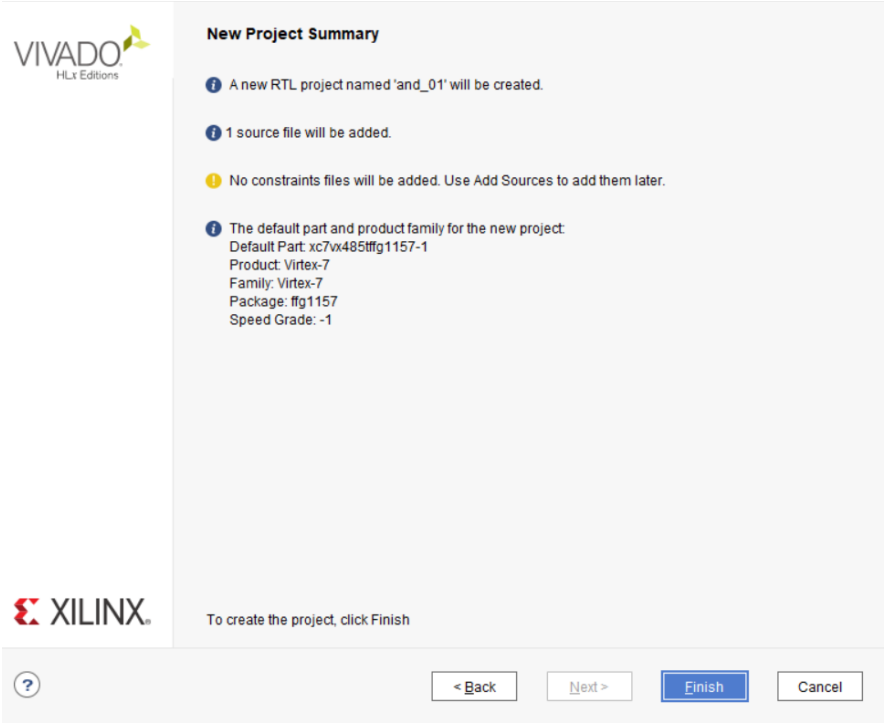
- In Add Sources window click *Create File*, then select file type as *VHDL* and name the file “ff_01”, click OK. Make sure *VHDL* is selected as Target language and Simulator Language and click Next



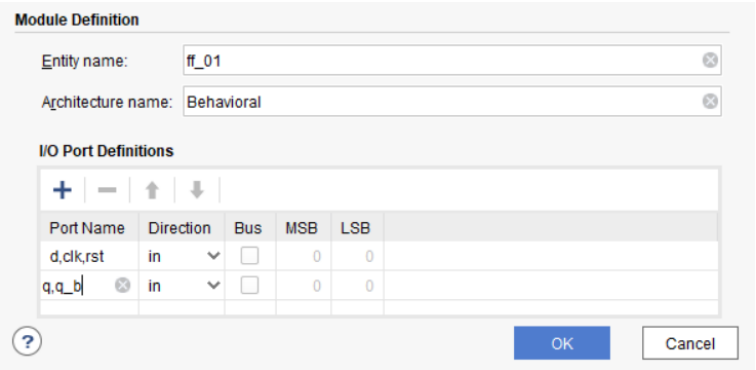
- Click next on the Add constraints window and in the Default Part window select the board available.



- Click Finish to create project.



- Define module window appears, in the I/O Definitions tab of the window define the input and output ports for D flipflop and click OK.



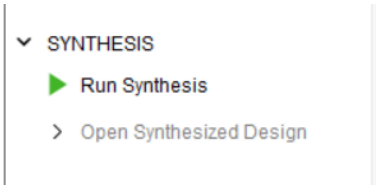
- In the ff_01.vhd file add the following code to define (Behavioral) the gate and save it by pressing ctrl + s:

Project Summary | ff_01.vhd

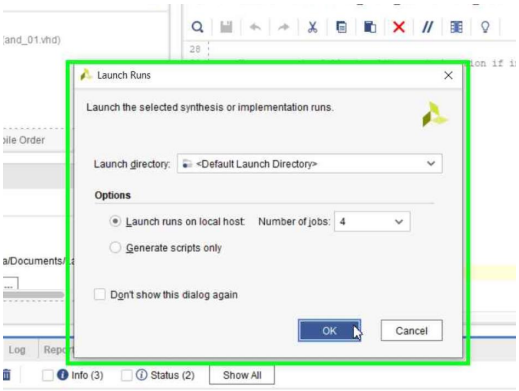
C:/Users/rizwa/Documents/Lab1/ff_01/ff_01.srcs/sources_1/new/ff_01.vhd

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3  entity ff_01 is
4      Port ( d,clk,rst : in STD_LOGIC;
5             q,q_b : out STD_LOGIC);
6  end ff_01;
7  architecture Behavioral of ff_01 is
8  begin
9  process (clk, rst)
10 begin
11 if (rst='1') then
12 q <='0';
13 q_b <='1';
14 elsif (clk 'EVENT AND clk='1') then
15 q <= d;
16 q_b <= not d;
17 end if;
18 end process;
19 end Behavioral;
```

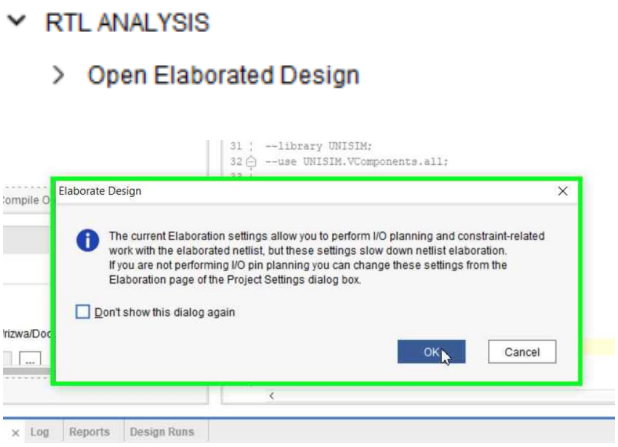
- To synthesize the design, click on Run Synthesis:



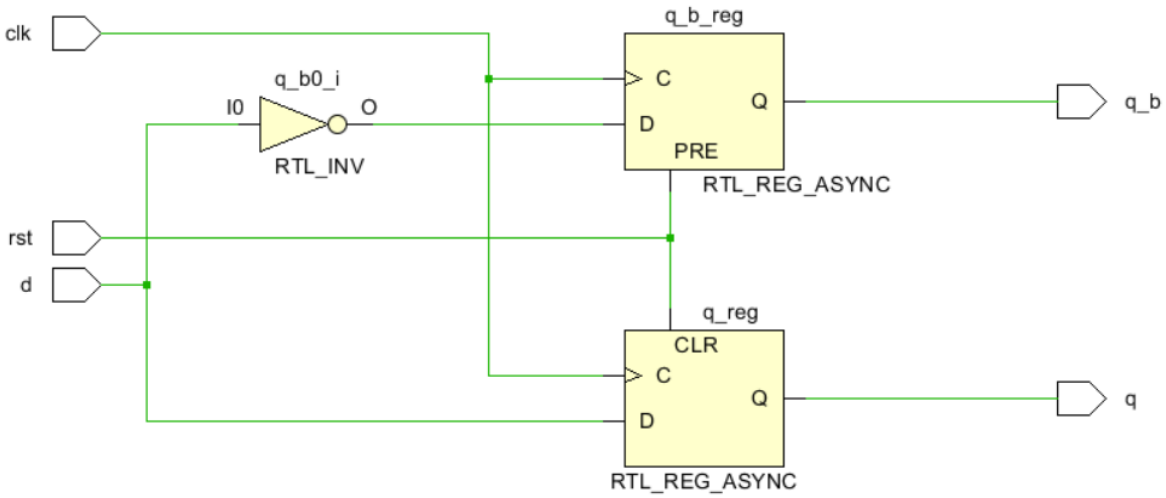
- Then click on OK in Launch Run window



- Close the synthesis completed dialogue, and then Click on Open Elaborated Design and Click OK

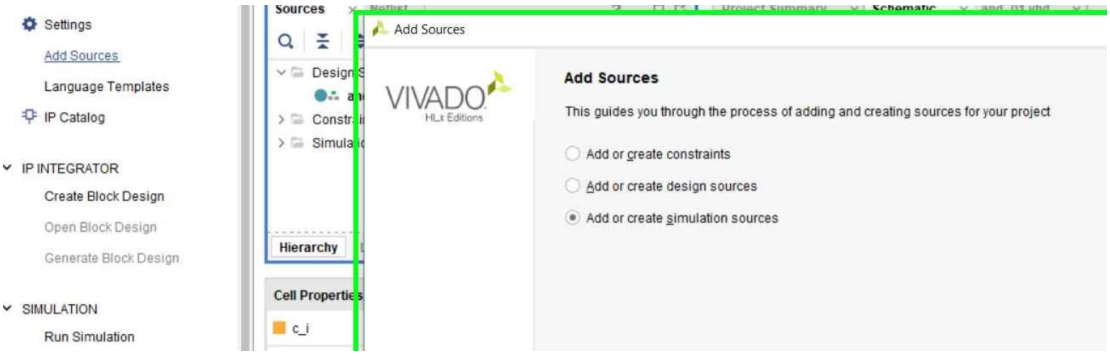


- This shows the schematic of the design

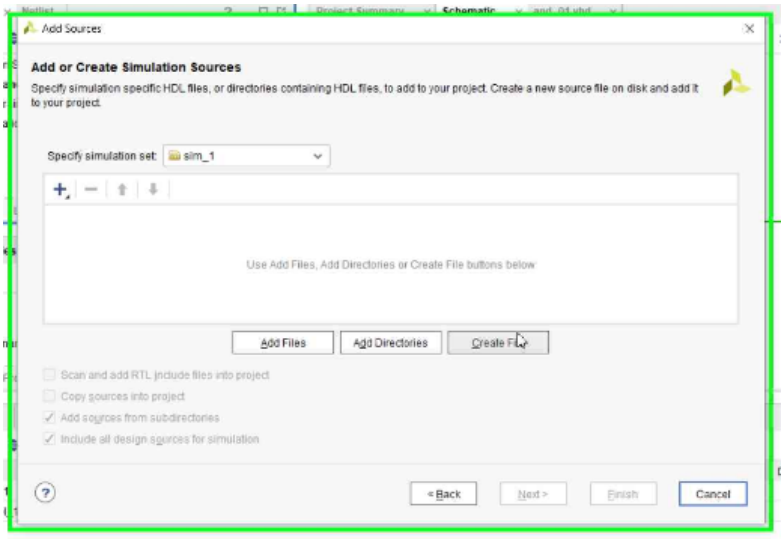


6) To Create a Test Bench:

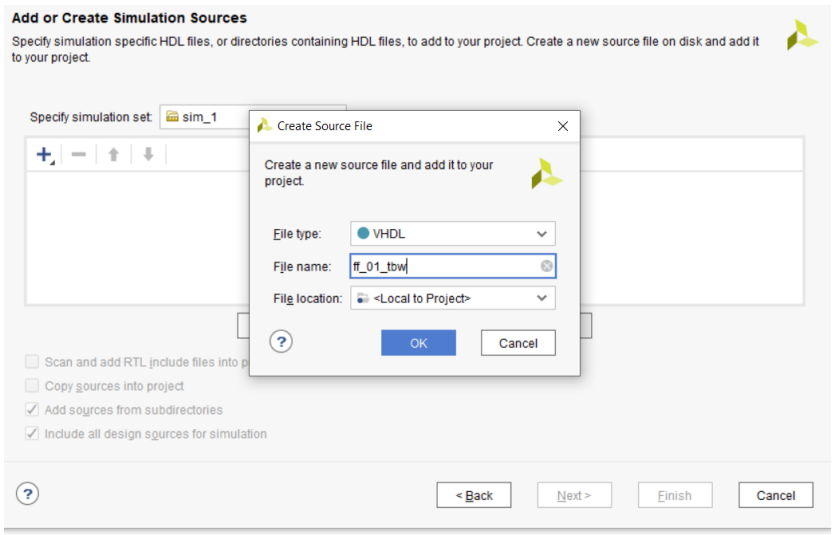
- Click on Add Source and choose Add or Create Simulation Source, then click next.



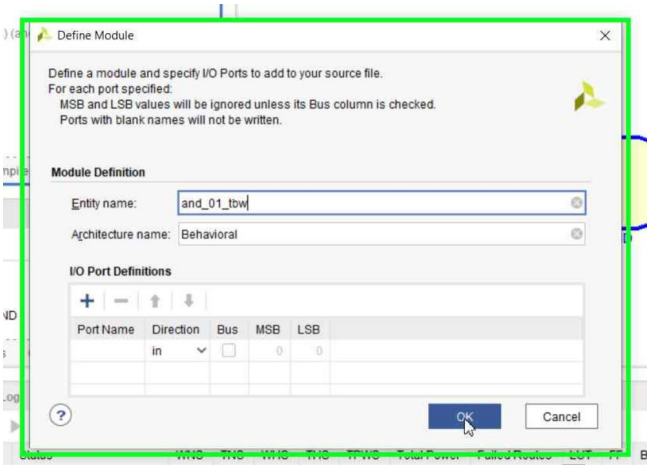
- In Add or Create Simulation Source Window click on Create File



- Select File type as VHDL and name the file “ff_01_tbw”. Then click OK and then Finish.



- In Define module window click OK and if the dialogue pops up click on YES.

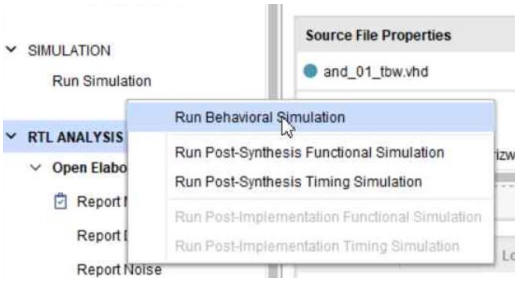


- Under sources window, double click on “ff_01_tbw” file.

- Then in ff_01_tbw.vhd file add the following code:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity ff_01_tbw is
--  Port ( );
end ff_01_tbw;

architecture Behavioral of ff_01_tbw is
component ff_01 is
port( d, clk, rst : in std_logic;
q, q_b: out std_logic);
end component;
signal d : std_logic:='0';
signal rst : std_logic:='0';
signal q : std_logic;
signal q_b: std_logic;
signal clk: std_logic;
begin
uut : ff_01 port map (d=>d, rst=> rst, q=>q, q_b=>q_b, clk=>clk);
   clk_process : process
   begin
     clk <= '0';
     wait for 100 ns;
     clk <= '1';
     wait for 100 ns;
 end process;
stim_proc: process
begin
d<='1';
rst<='1';
wait for 150ns;
d<='1';
rst<='0';
wait for 70ns;
d<='0';
rst<='1';
wait for 80ns;
d<='1';
rst<='0';
wait for 100ns;
d<='0';
rst<='0';
wait for 100ns;
end process;
end Behavioral;
```

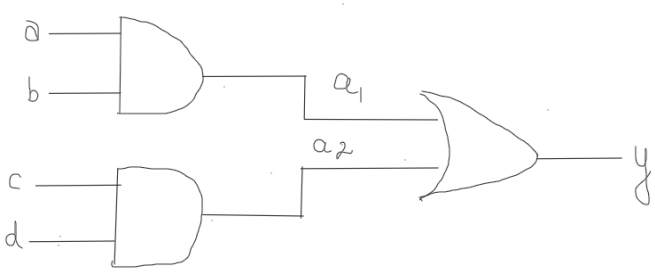
- Save the file and under Simulation option double click on Run Simulation and choose Run Behavioral Simulation



- Behavioral Simulation window appears showing the, waveform of D Flipflop for different values of input.

EXPERIMENT 19

AIM: To create the given deigns using instantiation.

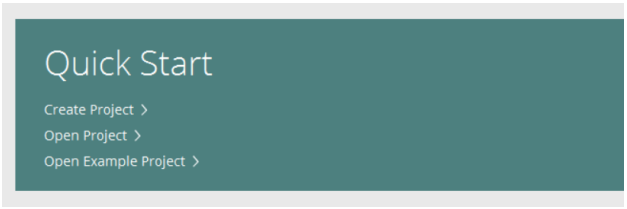


Apparatus Required: Vivado Design Suite.

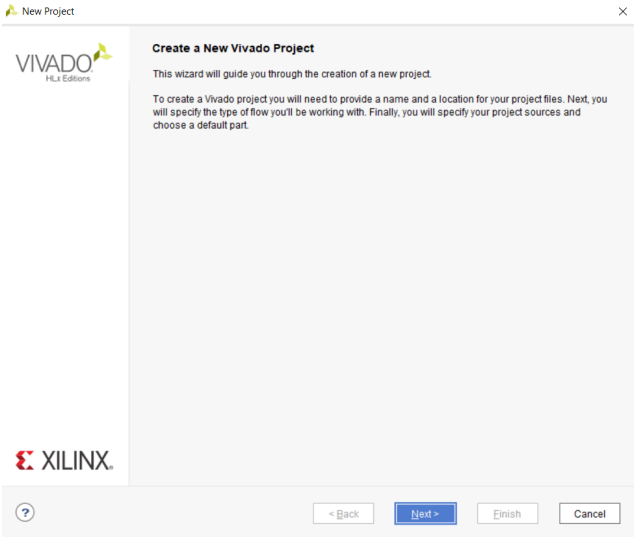
Procedure:

7) **To create a Full adder:**

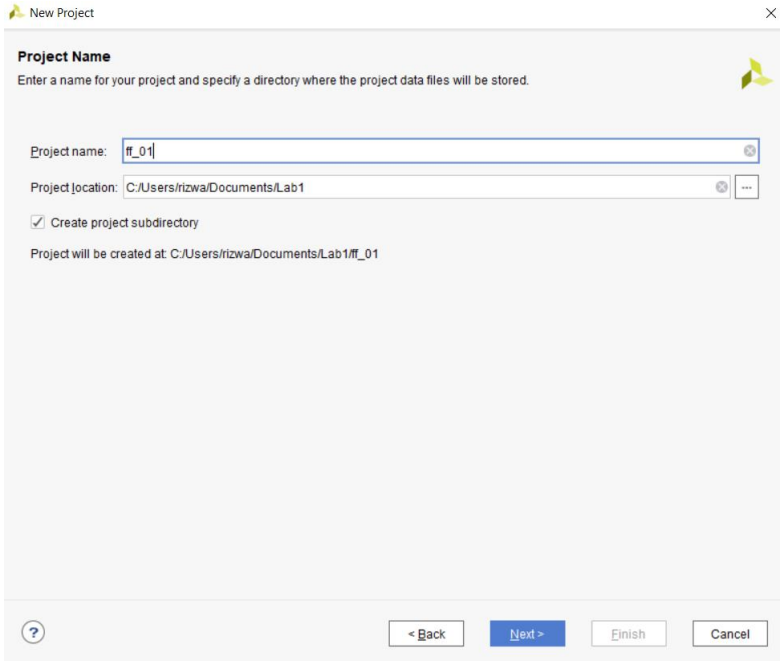
- Open *Vivado Design Suite*
- click on *Create Project* option.



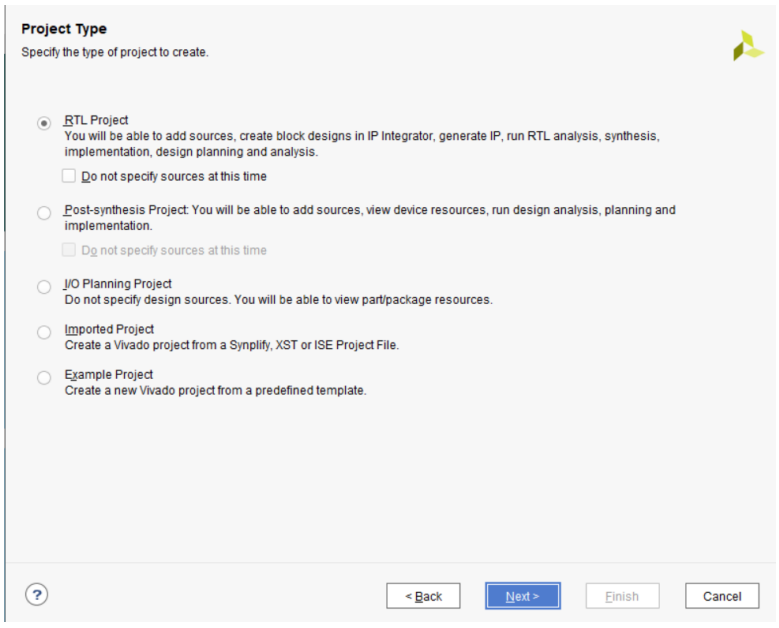
- Click next on Create Project window



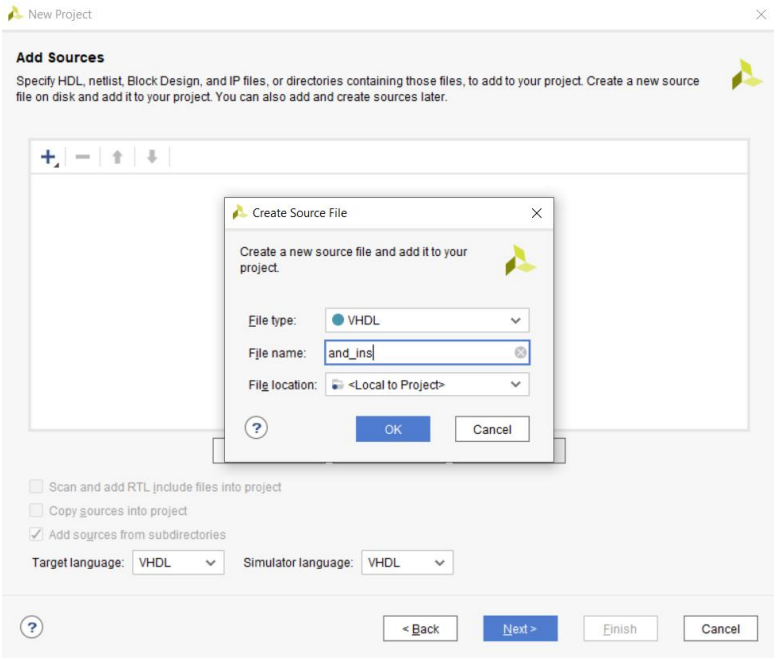
- Name the project “design” and select the location to save the project, then click next



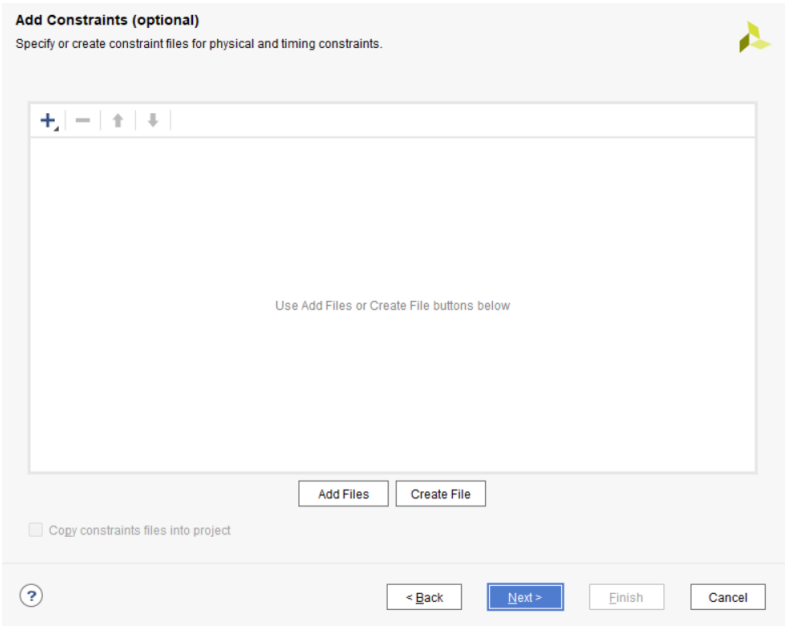
- In Project Type window select *RTL Project* and click next.



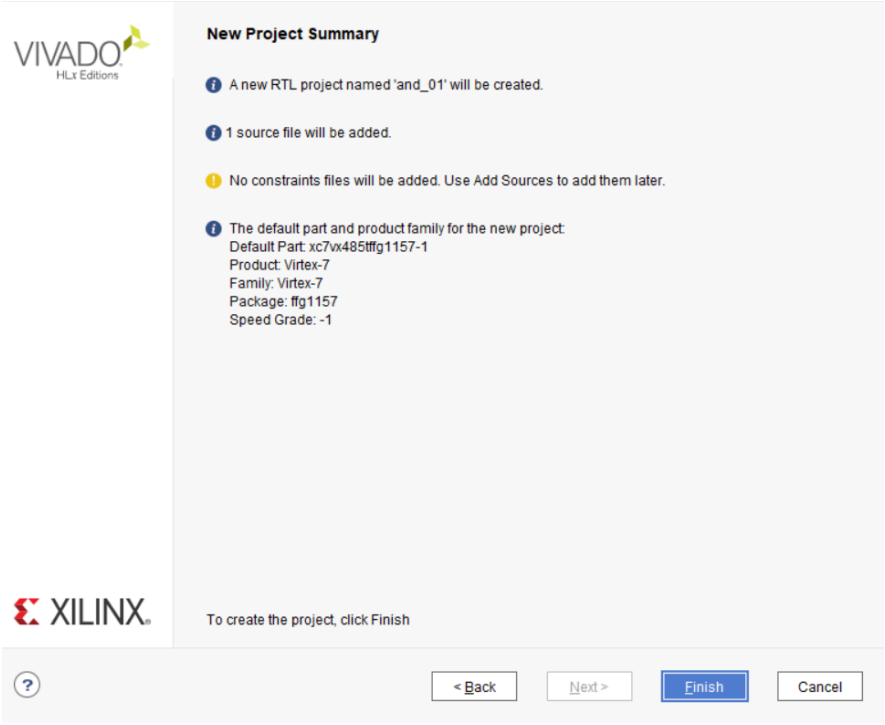
- In Add Sources window click *Create File*, then select file type as *VHDL* and name the file “and_inst”, click OK. Make sure *VHDL* is selected as Target language and Simulator Language and click Next



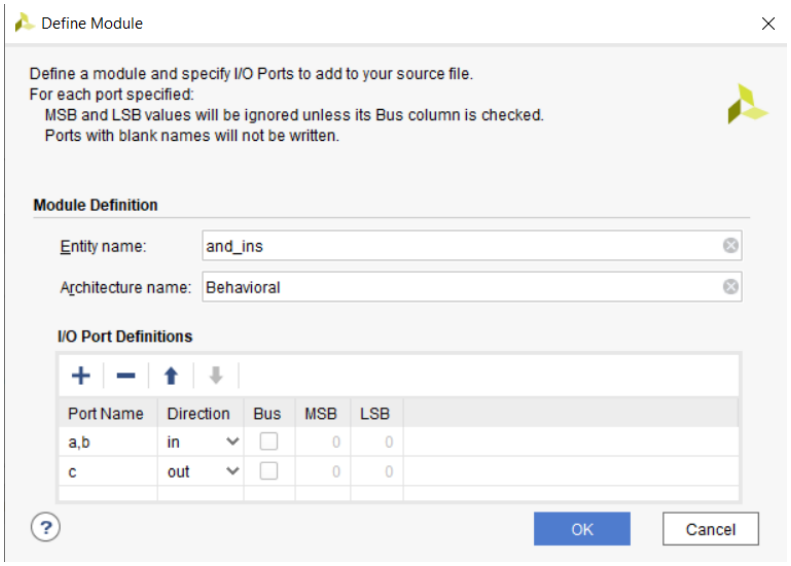
- Click next on the Add constraints window and in the Default Part window select the board available.



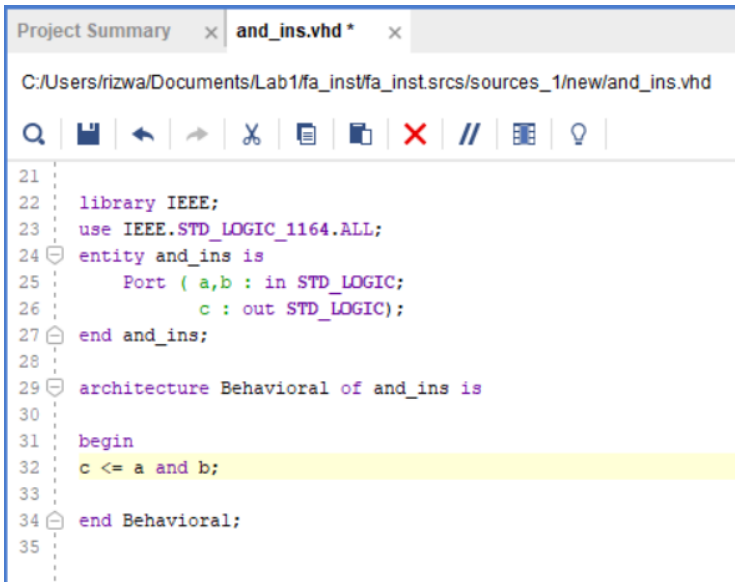
- Click Finish to create project.



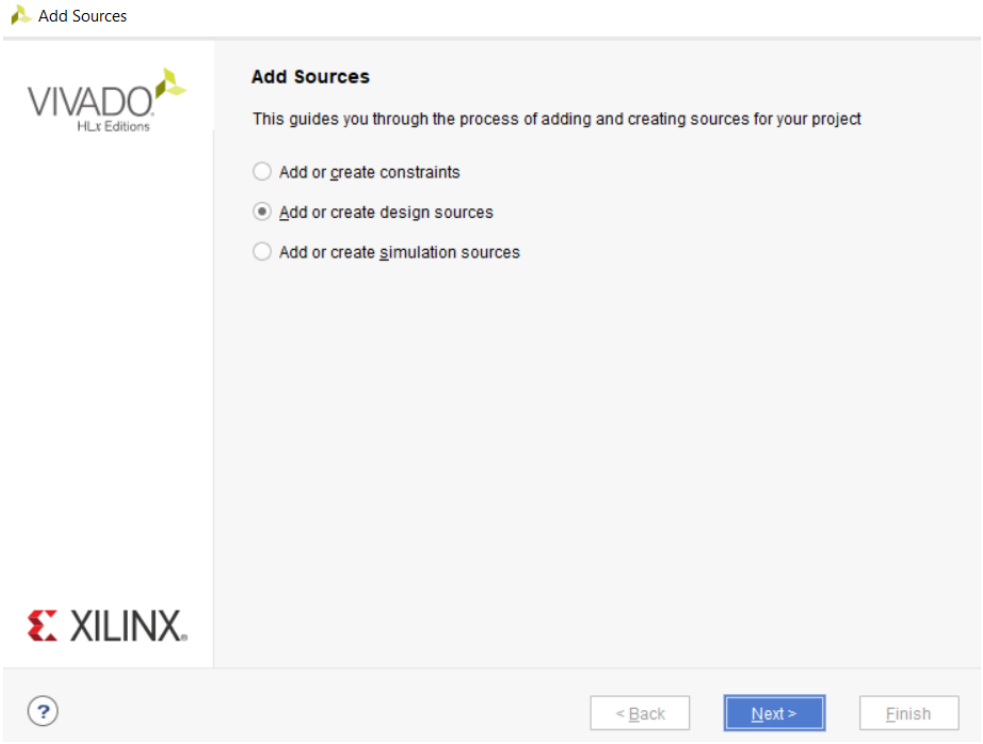
- Define module window appears, in the I/O Definitions tab of the window define the input and output ports for and_ins and click OK.



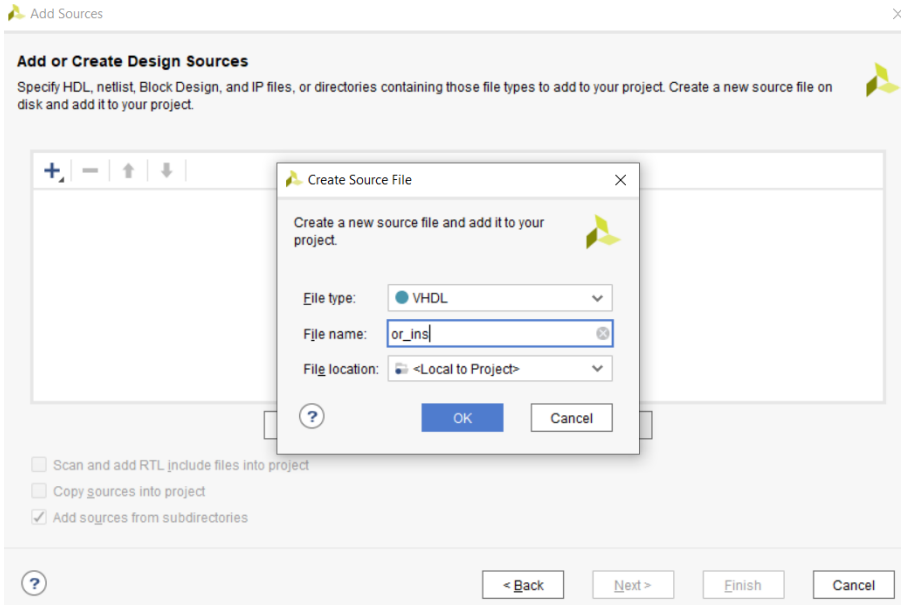
- In the and_ins.vhd file add the following code to define (Behavioral) the gate and save it by pressing ctrl + s:



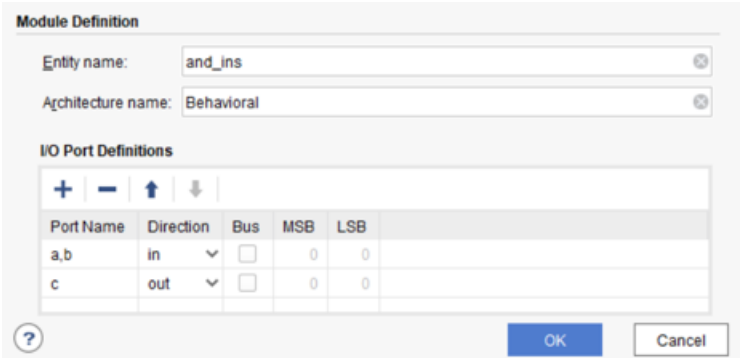
- Now, click on add source, then select add or create design source, then next.



- In Add Sources window click *Create File*, then select file type as *VHDL* and name the file “or_inst”, click OK. Make sure *VHDL* is selected as Target language and Simulator Language and click Finsih.



- Define module window appears, in the I/O Definitions tab of the window define the input and output ports for or_ins and click OK



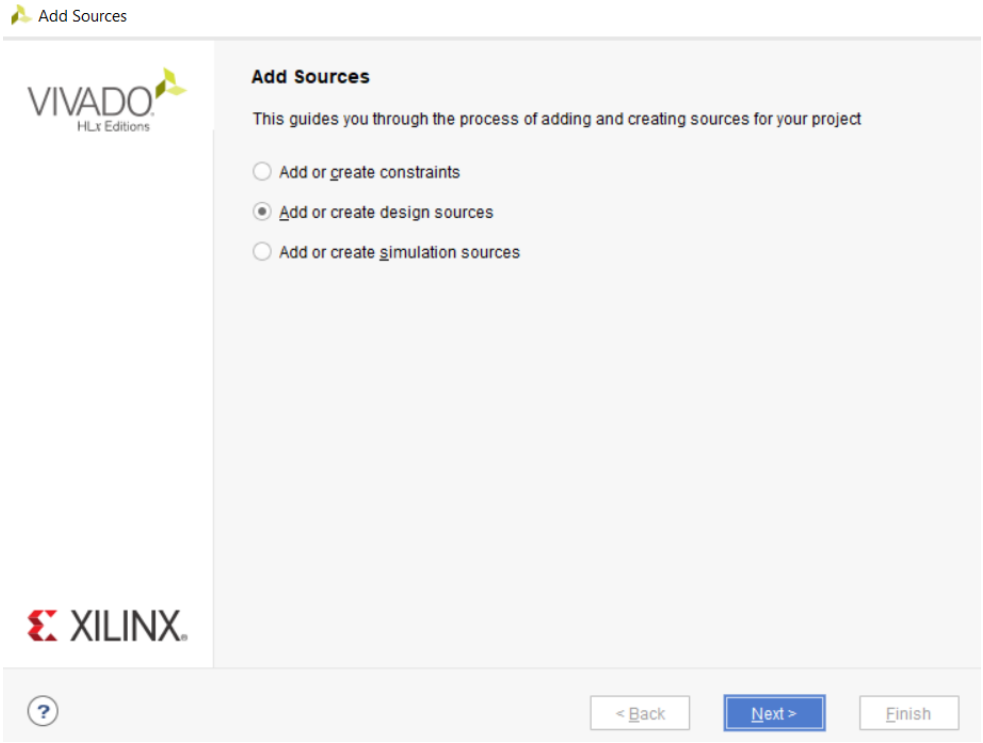
- In the or_ins.vhd file add the following code to define (Behavioral) the gate and save it by pressing ctrl + s:

Project Summary | and_ins.vhd | or_ins.vhd

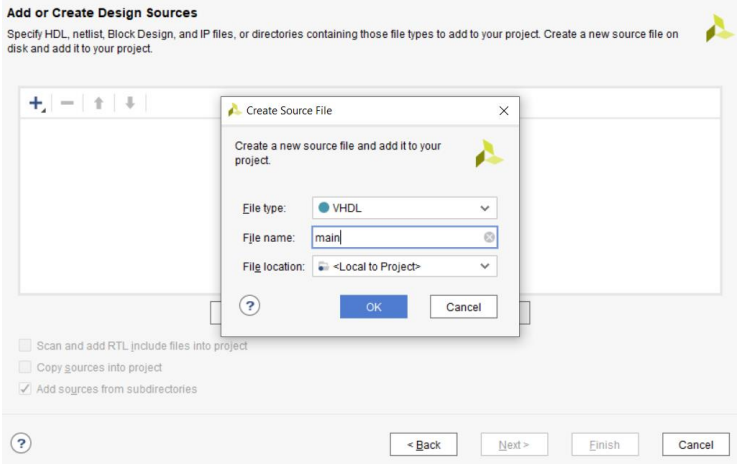
C:/Users/rizwa/Documents/Lab1/fa_inst/fa_inst.srcs/sources_1/new/or_ins.vhd

```
1
2   library IEEE;
3   use IEEE.STD_LOGIC_1164.ALL;
4   entity or_ins is
5       Port ( a,b : in STD_LOGIC;
6              c : out STD_LOGIC);
7   end or_ins;
8
9   architecture Behavioral of or_ins is
10
11  begin
12  c <= a or b;
13
14  end Behavioral;
15
```

- Now, again click on add source, then select add or create design source, then next.



- In Add Sources window click *Create File*, then select file type as *VHDL* and name the file “main”, click OK. Make sure *VHDL* is selected as Target language and Simulator Language and click Finsih.



- Define module window appears, in the I/O Definitions tab of the window define the input and output ports for ‘main’ and click OK

a, b, c, d	in
y	out

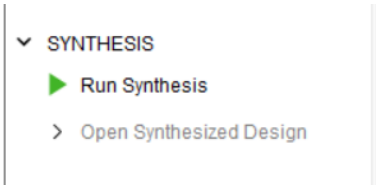
- In the or_ins.vhd file add the following code to define (Behavioral) the gate and save it by pressing ctrl + s:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity main is
   Port ( a : in STD_LOGIC;
          b : in STD_LOGIC;
          c : in STD_LOGIC;
          d : in STD_LOGIC;
          y : out STD_LOGIC);
end main;
architecture Behavioral of main is
signal a1, a2 : std_logic:='0';
component and_ins is
   Port ( a : in STD_LOGIC;
          b : in STD_LOGIC;
          c : out STD_LOGIC);
end component;
```

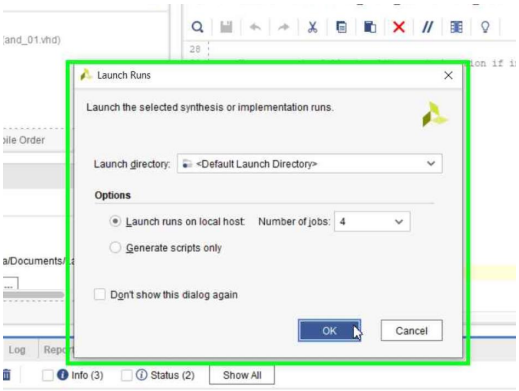
```
component or_ins is
   Port ( a : in STD_LOGIC;
          b : in STD_LOGIC;
          c : out STD_LOGIC);
end component;
begin
INS1: and_ins port map(a=>a,b=>b,c=>a1);
INS2: and_ins port map(a=>c,b=>d,c=>a2);
INS3: or_ins port map(a=>a1,b=>a2,c=>y);
end Behavioral;
```

- Right click on the main file under the source tab, and the click set as top

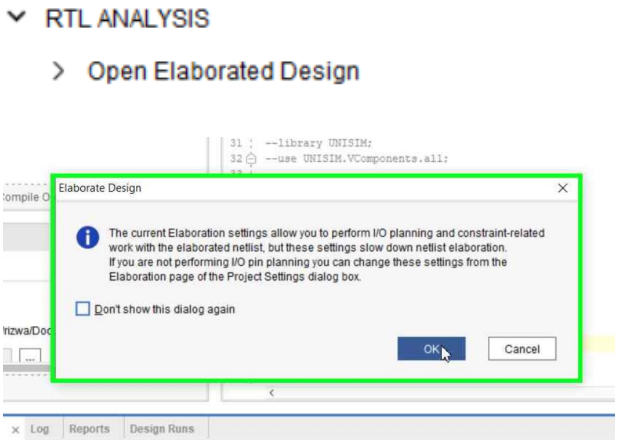
- To synthesize the design, click on Run Synthesis:



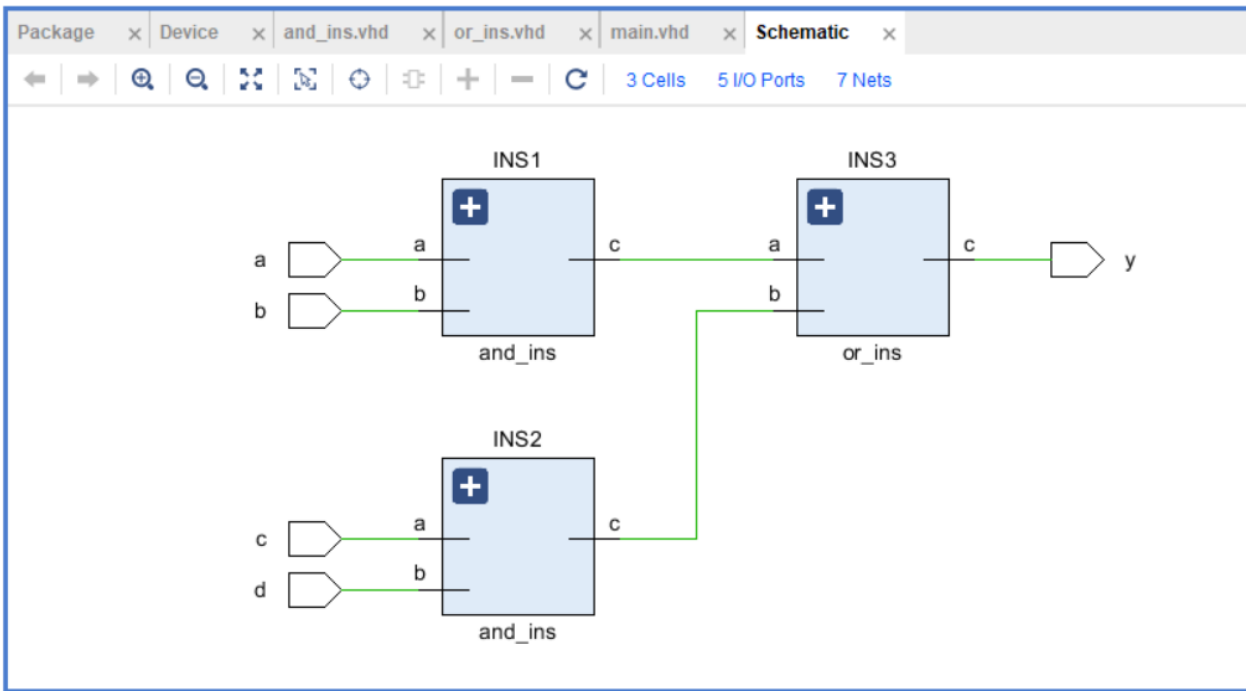
- Then click on OK in Launch Run window



- Close the synthesis completed dialogue, and then Click on Open Elaborated Design and Click OK

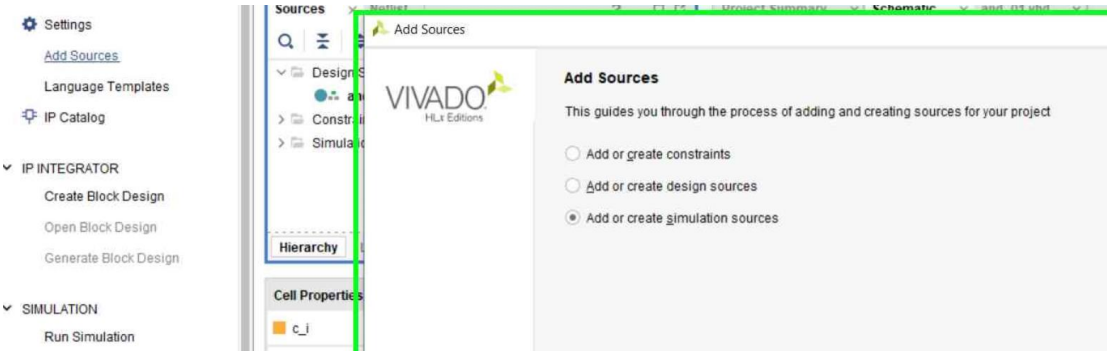


- This shows the schematic of the design

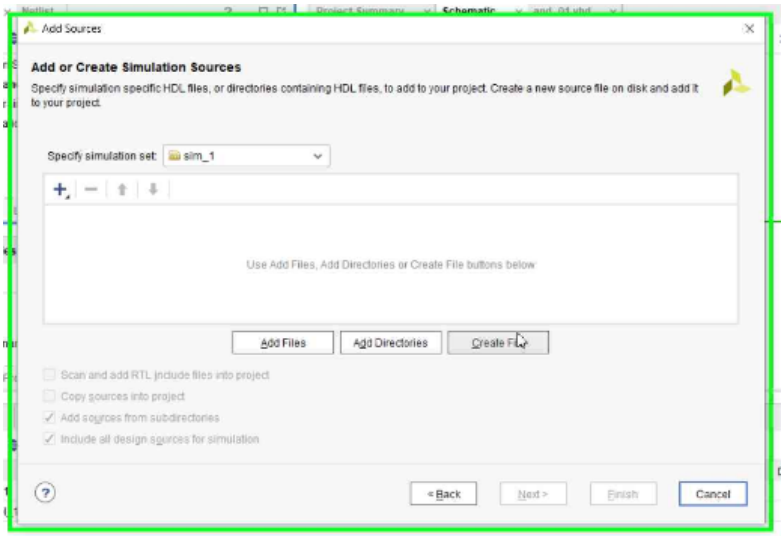


8) To Create a Test Bench:

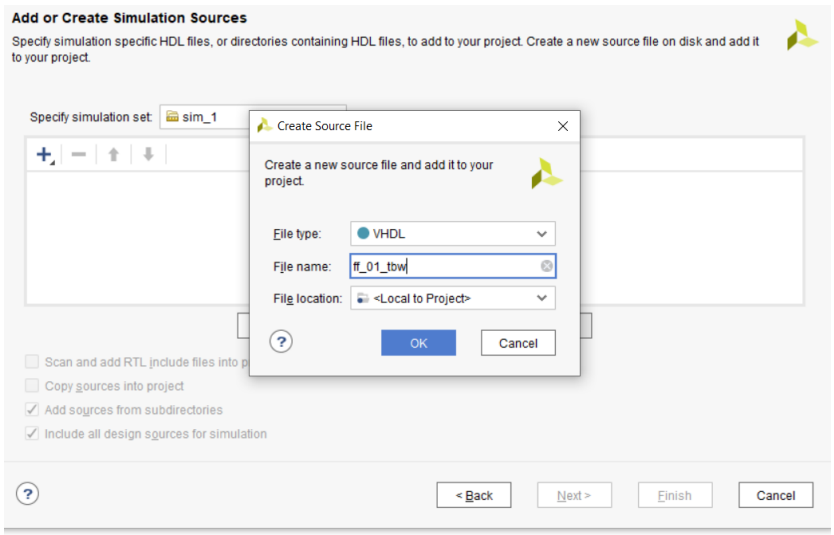
- Click on Add Source and choose Add or Create Simulation Source, then click next.



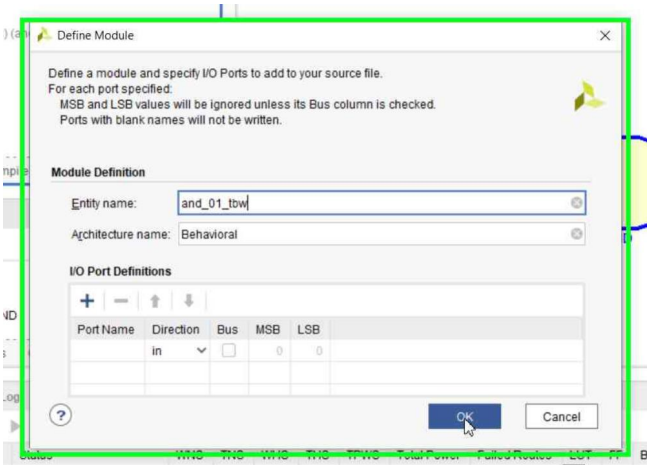
- In Add or Create Simulation Source Window click on Create File



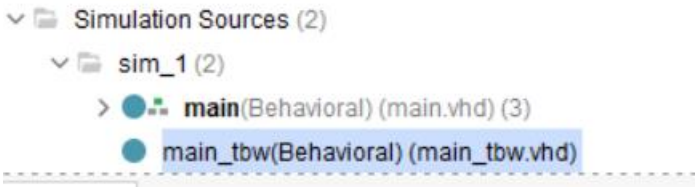
- Select File type as VHDL and name the file “main_tbw”. Then click OK and then Finish.



- In Define module window click OK and if the dialogue pops up click on YES.



- Under sources window, double click on “main_tbw” file.

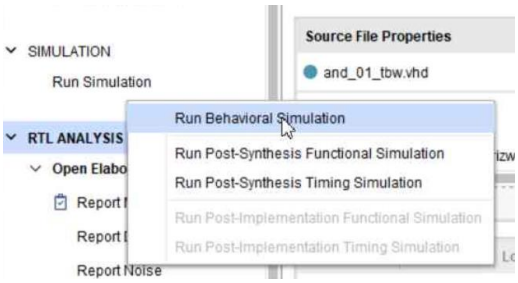


- Then in main_tbw.vhd file add the following code:

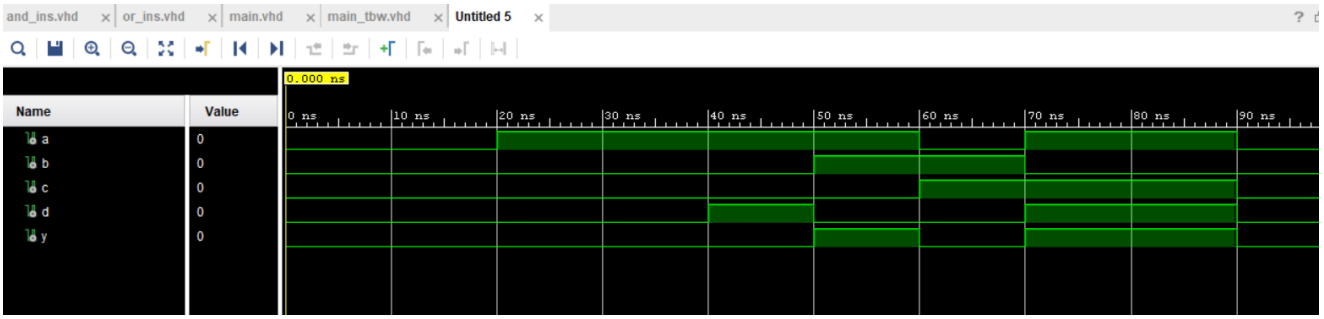
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity main_tbw is
end main_tbw;

architecture Behavioral of main_tbw is
component main is port(a,b,c,d: in std_logic;
 y: out std_logic);
 end component;
 signal a: std_logic:='0';
 signal b: std_logic:='0';
 signal c: std_logic:='0';
 signal d: std_logic:='0';
 signal y: std_logic;

begin
uut: main port map (a=>a,b=>b,c=>c,d=>d, y=>y);
stim_proc: process
begin
wait for 10 ns;
a<= '0';
b <= '0';
c<='0';
d<='0';
wait for 10 ns;
a<= '1';
b <= '0';
c<='0';
d<='0';
wait for 10 ns;
wait for 10 ns;
a<= '1';
b <= '0';
c<='0';
d<='1';
wait for 10 ns;
a<= '1';
b <= '1';
c<='0';
d<='0';
wait for 10 ns;
a<= '0';
```

```
b <= '1';
c<='1';
d<='0';
wait for 10 ns;
a<= '1';
b <= '0';
c<='1';
d<='1';
wait for 10 ns;
end process;

end Behavioral;
```

- Save the file and under Simulation option double click on Run Simulation and choose Run Behavioral Simulation

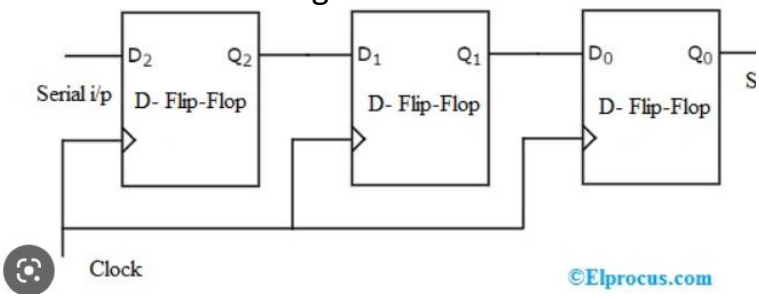


- Behavioral Simulation window appears showing the, waveform of the design for different values of input.



EXPERIMENT 20

AIM: To design and simulate shift register

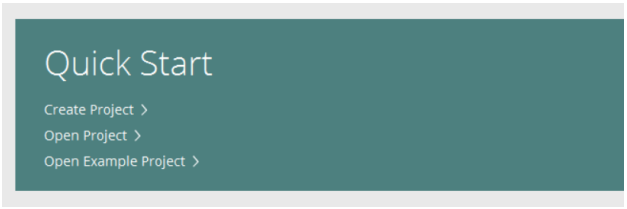


Apparatus Required: Vivado Design Suite.

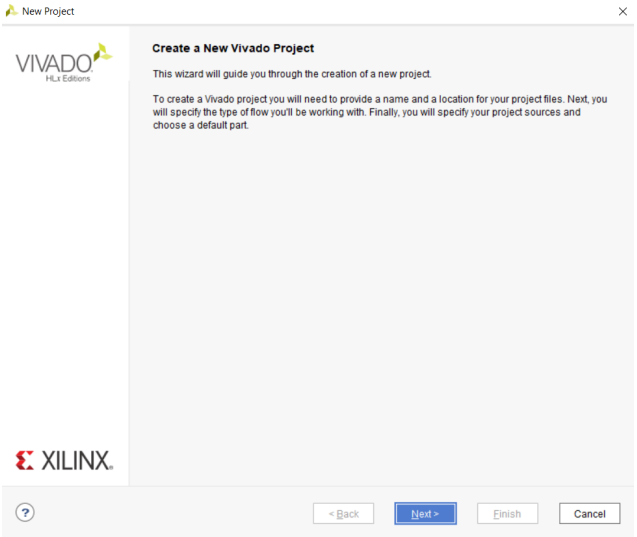
Procedure:

9) To create a Shift Register:

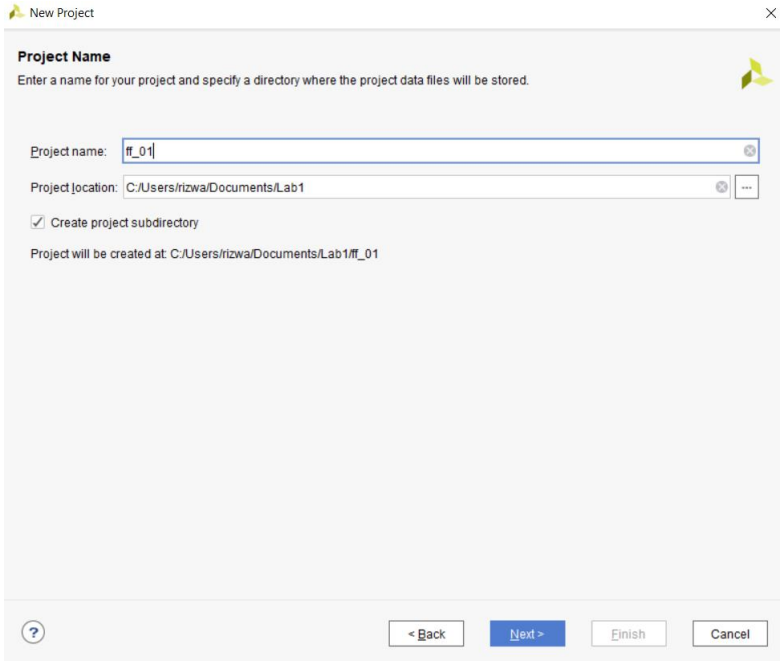
- Open *Vivado Design Suite*
- click on *Create Project* option.



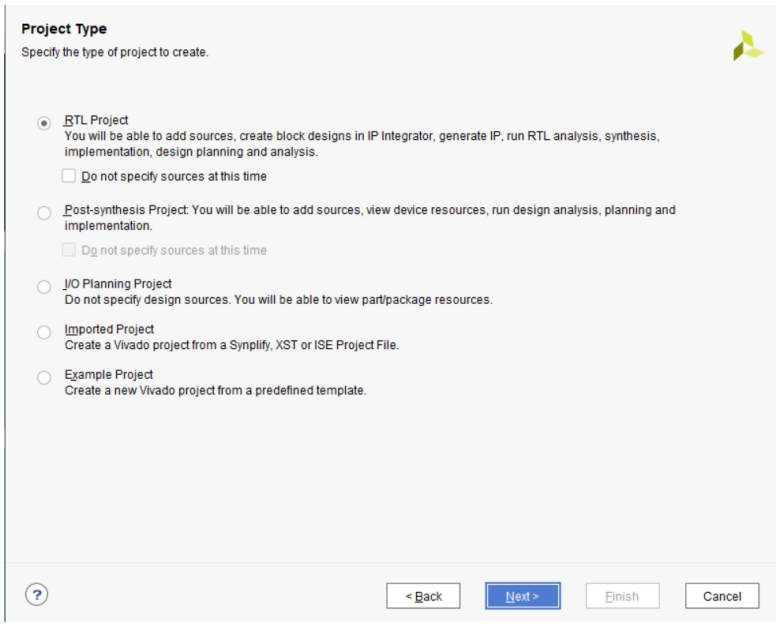
- Click next on Create Project window



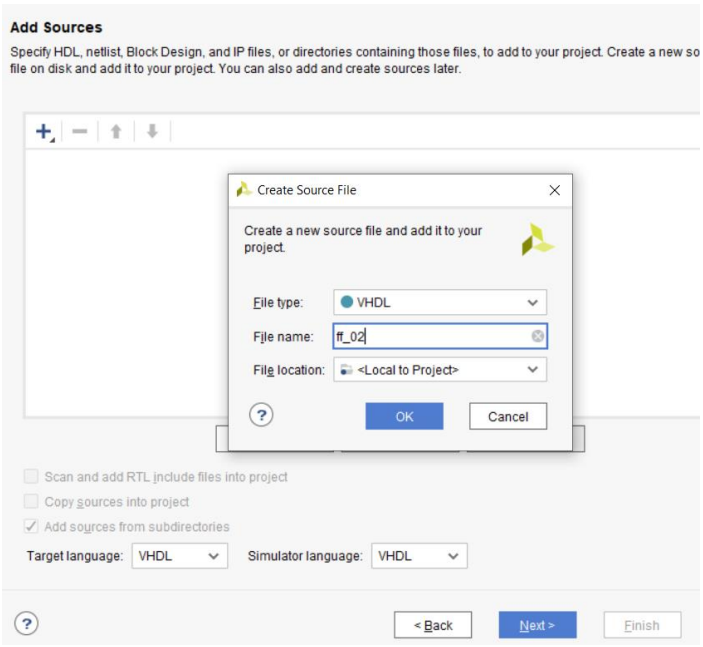
- Name the project “Register” and select the location to save the project, then click next



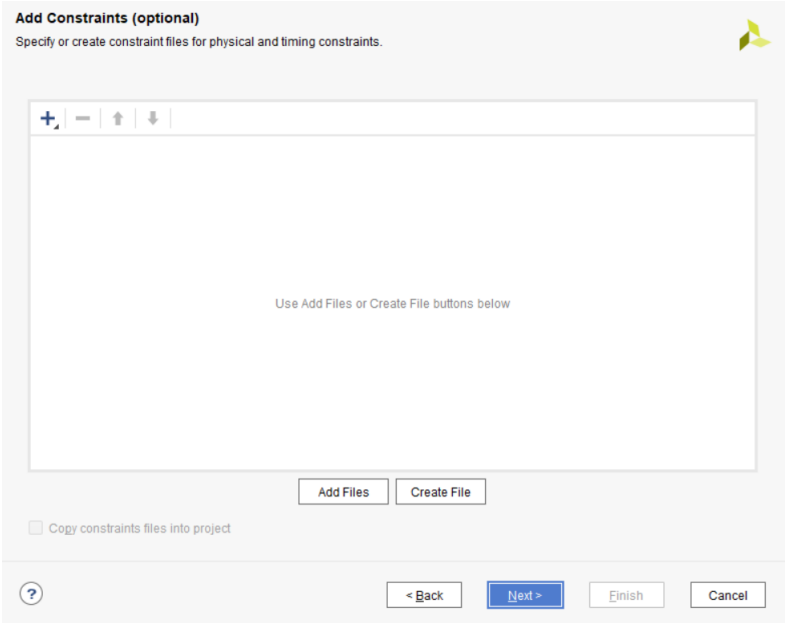
- In Project Type window select *RTL Project* and click next.



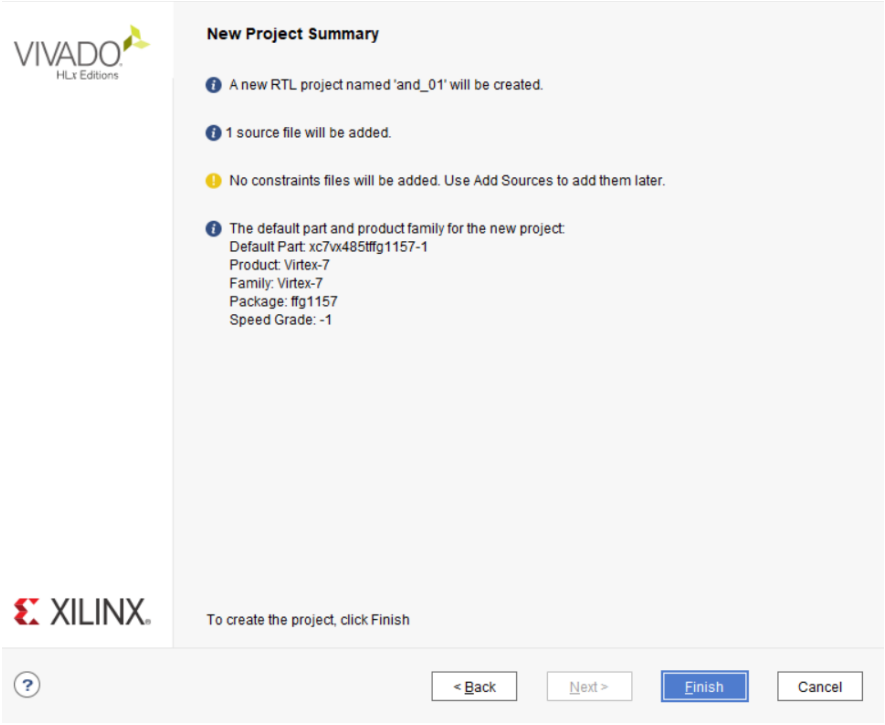
- In Add Sources window click *Create File*, then select file type as *VHDL* and name the file “ff_02”, click OK. Make sure *VHDL* is selected as Target language and Simulator Language and click Next



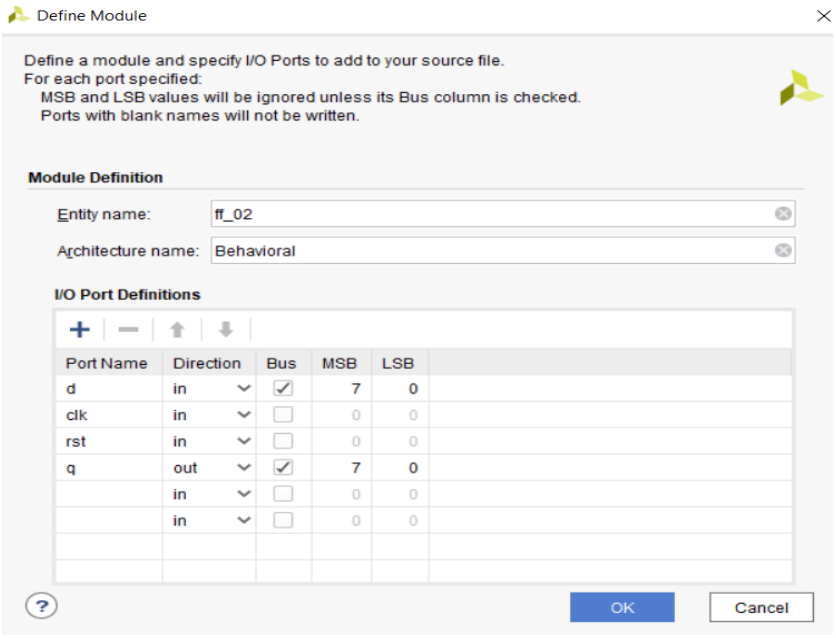
- Click next on the Add constraints window and in the Default Part window select the board available.



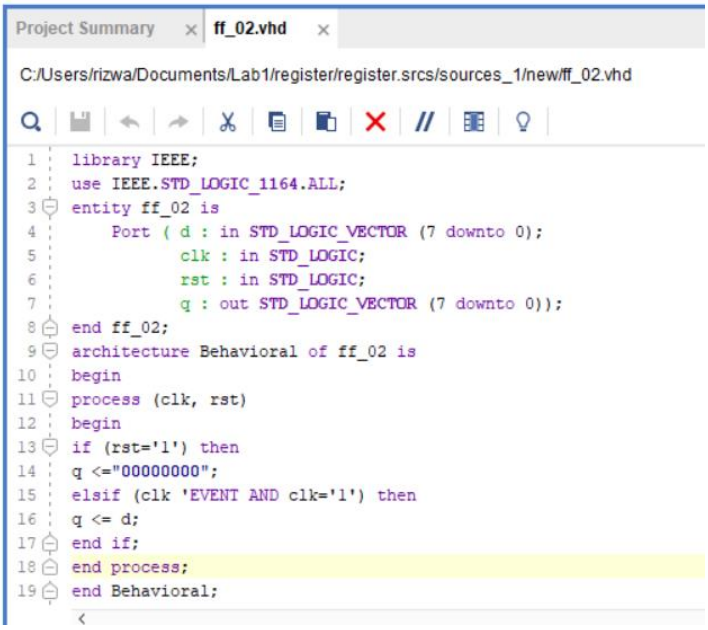
- Click Finish to create project.



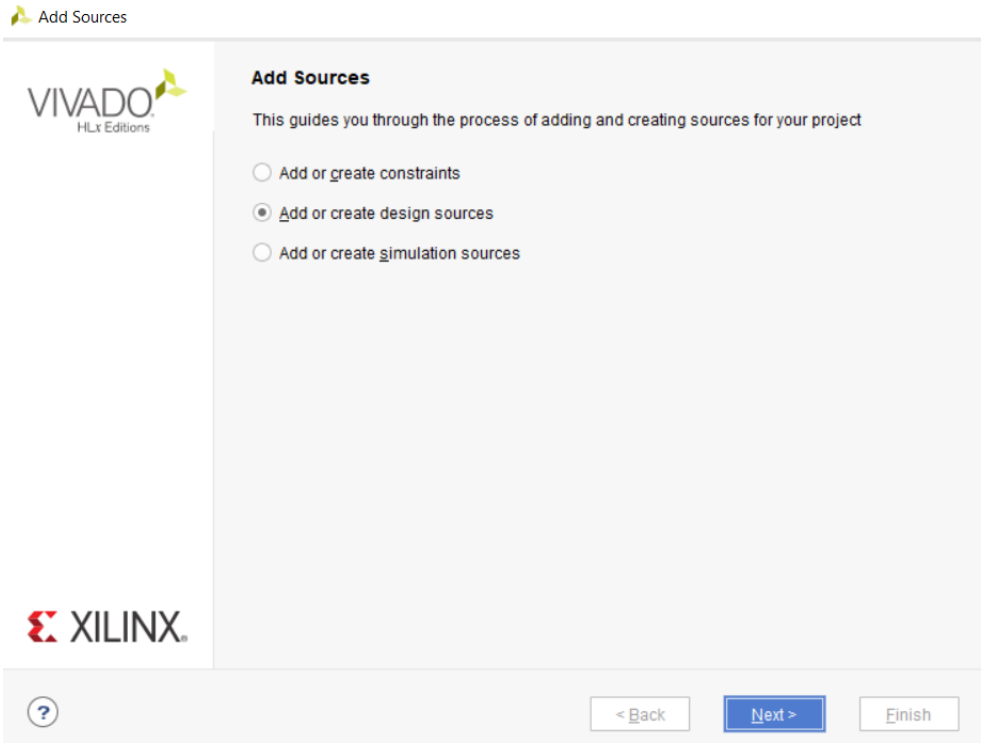
- Define module window appears, in the I/O Definitions tab of the window define the input and output ports for ff_02 and click OK.



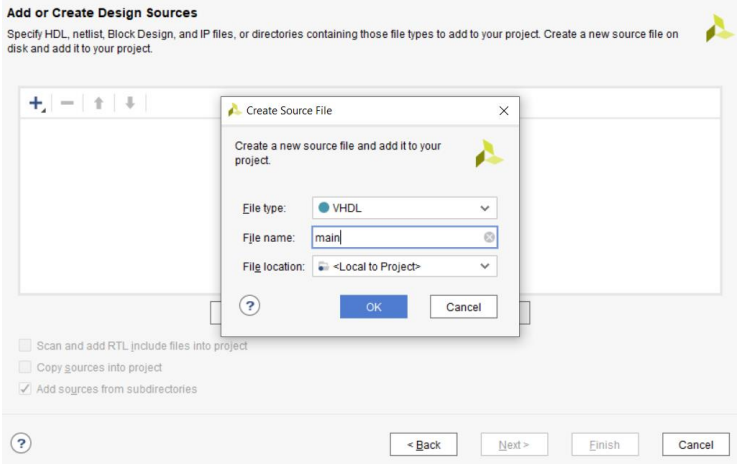
- In the ff_02.vhd file add the following code to define (Behavioral) the gate and save it by pressing ctrl + s:



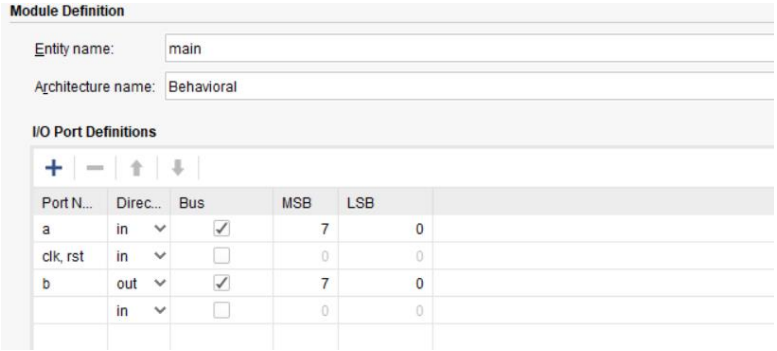
- Now, click on add source, then select add or create design source, then next.



- In Add Sources window click *Create File*, then select file type as *VHDL* and name the file “main”, click OK. Make sure *VHDL* is selected as Target language and Simulator Language and click Finish.



- Define module window appears, in the I/O Definitions tab of the window define the input and output ports for ‘main’ and click OK



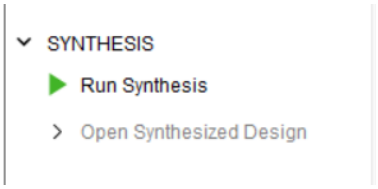
- In the main.vhd file add the following code to define (Behavioral) the gate and save it by pressing ctrl + s:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity main is
    Port ( a : in STD_LOGIC_VECTOR (7 downto 0);
           clk, rst : in STD_LOGIC;
           b : out STD_LOGIC_VECTOR (7 downto 0));
end main;
architecture Behavioral of main is
signal a1, a2,a3 : std_logic_vector(7 downto 0):="00000000";
component ff_02 is
    Port ( d : in STD_LOGIC_VECTOR (7 downto 0);
           clk : in STD_LOGIC;
```

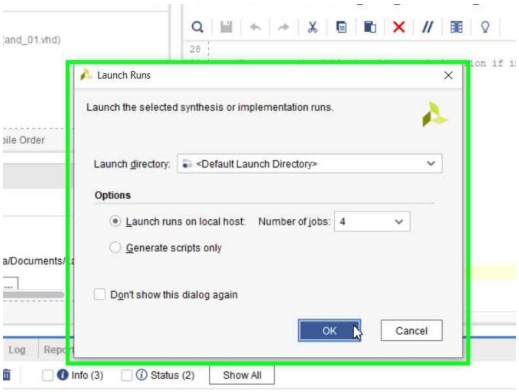
```
        rst : in STD_LOGIC;
        q : out STD_LOGIC_VECTOR (7 downto 0));
end component;
begin
INS1: ff_02 port map(d=>a,q=>a1, clk=>clk, rst=>rst);
INS2: ff_02 port map(d=>a1,q=>a2, clk=>clk, rst=>rst);
INS3: ff_02 port map(d=>a2,q=>a3, clk=>clk, rst=>rst);
INS4: ff_02 port map(d=>a3,q=>b, clk=>clk, rst=>rst);
end Behavioral;
```

- Right click on the main file under the source tab, and the click set as top

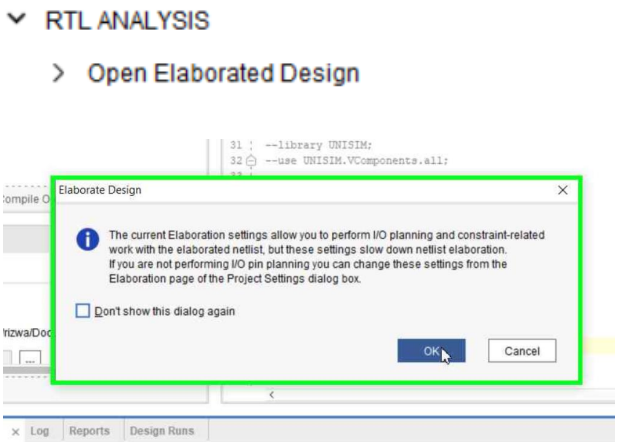
- To synthesize the design, click on Run Synthesis:



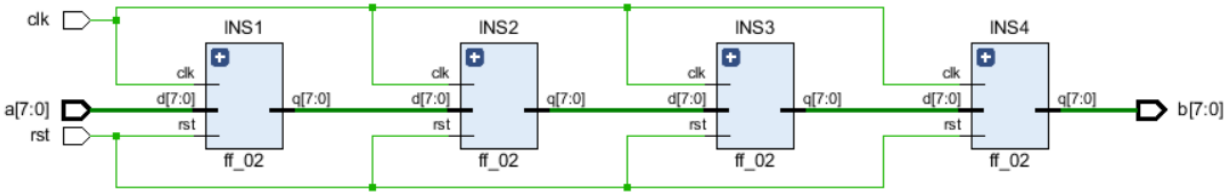
- Then click on OK in Launch Run window



- Close the synthesis completed dialogue, and then Click on Open Elaborated Design and Click OK

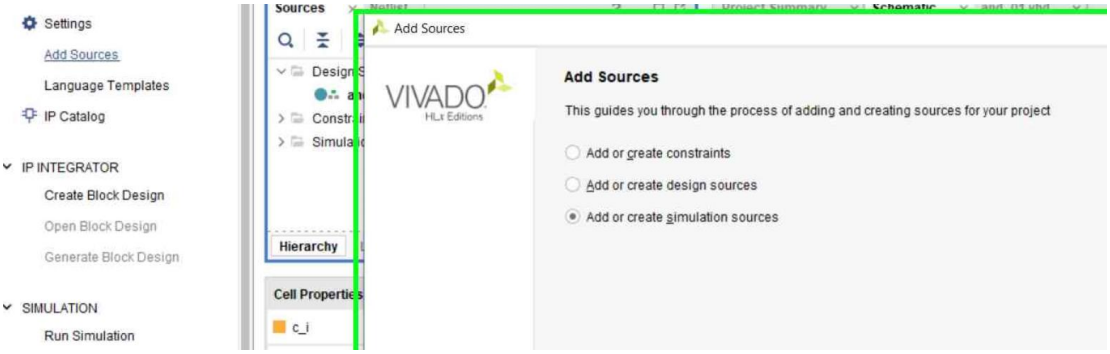


- This shows the schematic of the design

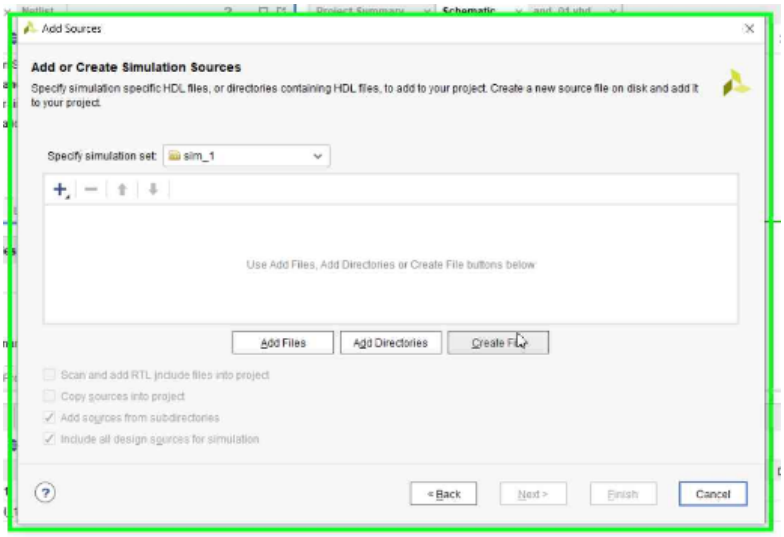


10) To Create a Test Bench:

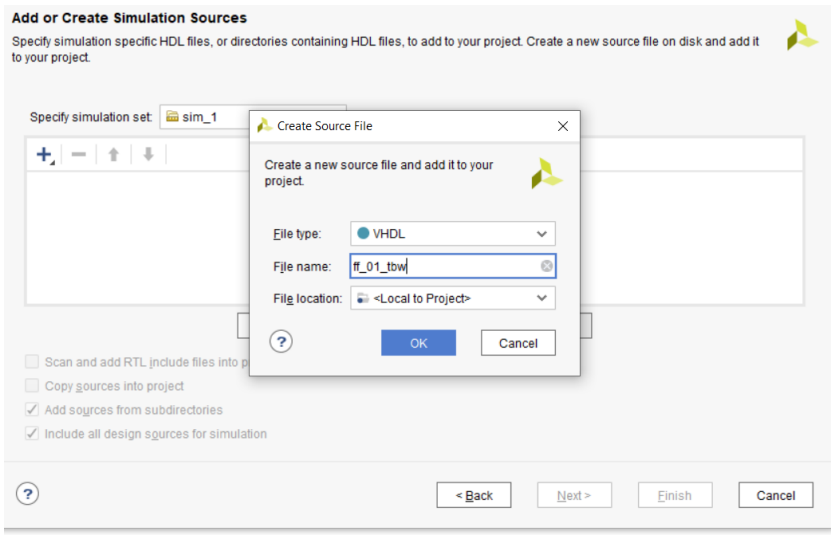
- Click on Add Source and choose Add or Create Simulation Source, then click next.



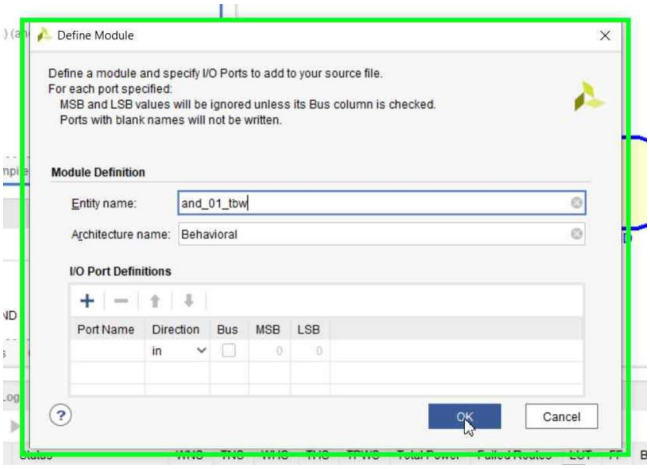
- In Add or Create Simulation Source Window click on Create File



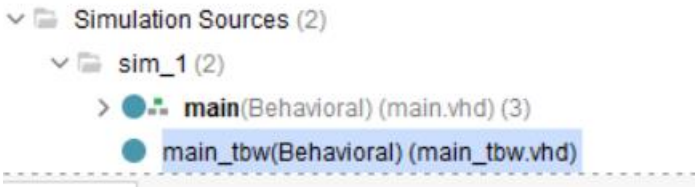
- Select File type as VHDL and name the file “main_tbw”. Then click OK and then Finish.



- In Define module window click OK and if the dialogue pops up click on YES.



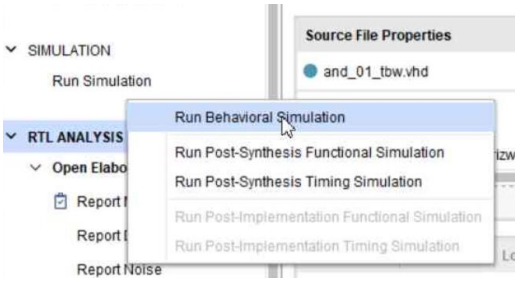
- Under sources window, double click on “main_tbw” file.



- Then in main_tbw.vhd file add the following code:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity main_tbw is
end main_tbw;

architecture Behavioral of main_tbw is
component main is port(a : in std_logic_vector(7 downto 0);
 clk,rst: in std_logic;
 b: out std_logic_vector(7 downto 0));
 end component;
signal a : std_logic_vector(7 downto 0):="00000000";
signal clk, rst: std_logic:='0';
signal b : std_logic_vector(7 downto 0);
begin
uut : main port map (a=>a, rst=> rst, b=>b, clk=>clk);
   clk_process : process
   begin
     clk <= '0';
     wait for 10 ns;
     clk <= '1';
     wait for 10 ns;
 end process;
stim_proc: process
 begin
 a<="00011100";
 rst<='0';
 wait for 10ns;
 a<="00000000";
 rst<='0';
 wait for 10ns;
 a<="10101010";
 rst<='0';
 wait for 10ns;
 a<="11111111";
 rst<='0';
 wait for 10ns;
 a<="11100111";
 rst<='0';
 wait for 10ns;
 end process;
end Behavioral;
```

- Save the file and under Simulation option double click on Run Simulation and choose Run Behavioral Simulation



- Behavioral Simulation window appears showing the, waveform of the shift register for different values of input.