**Xilinx Vivado Simulation Tutorial**

This tutorial guides you step-by-step on how to use Vivado design suite to create and simulate a Verilog description of a 2-input AND gate.

Before we begin, a typical design flow consists of

1. Designing a digital circuit (translate a problem statement into a digital model – logic

equation, logic circuit, state diagram, …

b) Creating a Xilinx Vivado project

c) Writing Verilog code (in this class) to describe the digital circuit/system that you want to simulate/implement

d) Performing a simulation (to check that your Verilog code represents a digital circuit/system that you want to design)

e) Synthesizing and implementing the design (see Xilinx Vivado Synthesis Tutorial document).

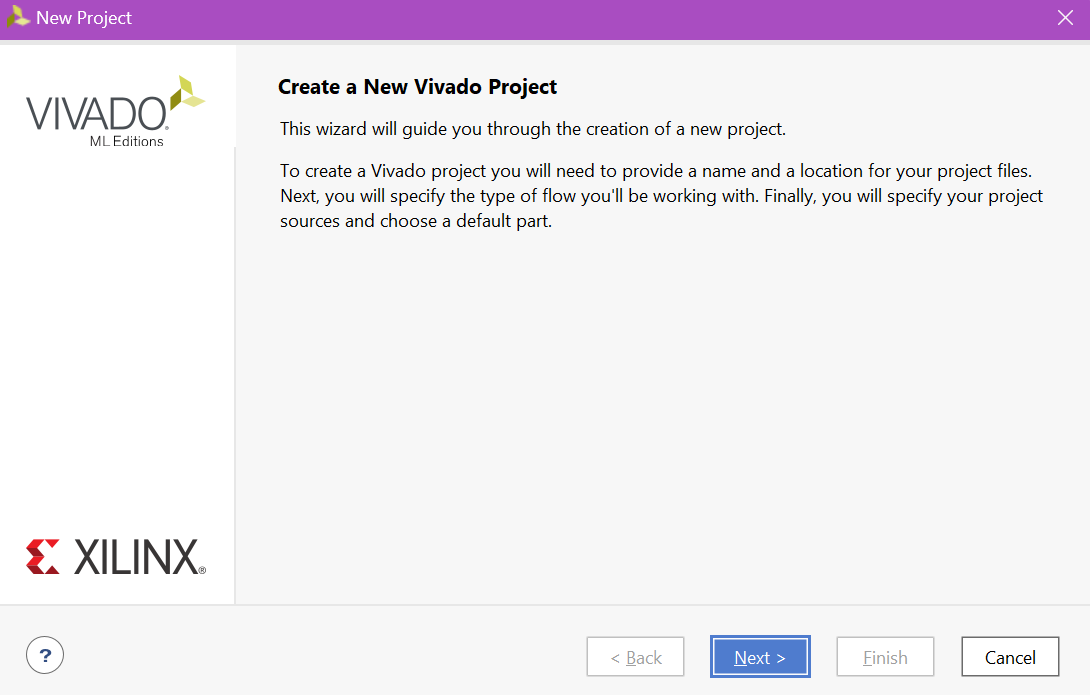
**Procedure**:

**Step 1**: Start Xilinx Vivado  . You will then see the first screen of Vivado.

**Step 2**: Create a new project:

* 1. Click on ***Create Project*** or Click File -> Project (dropdown menu) -> New.

The New Project window will be opened. Click ***Next***.

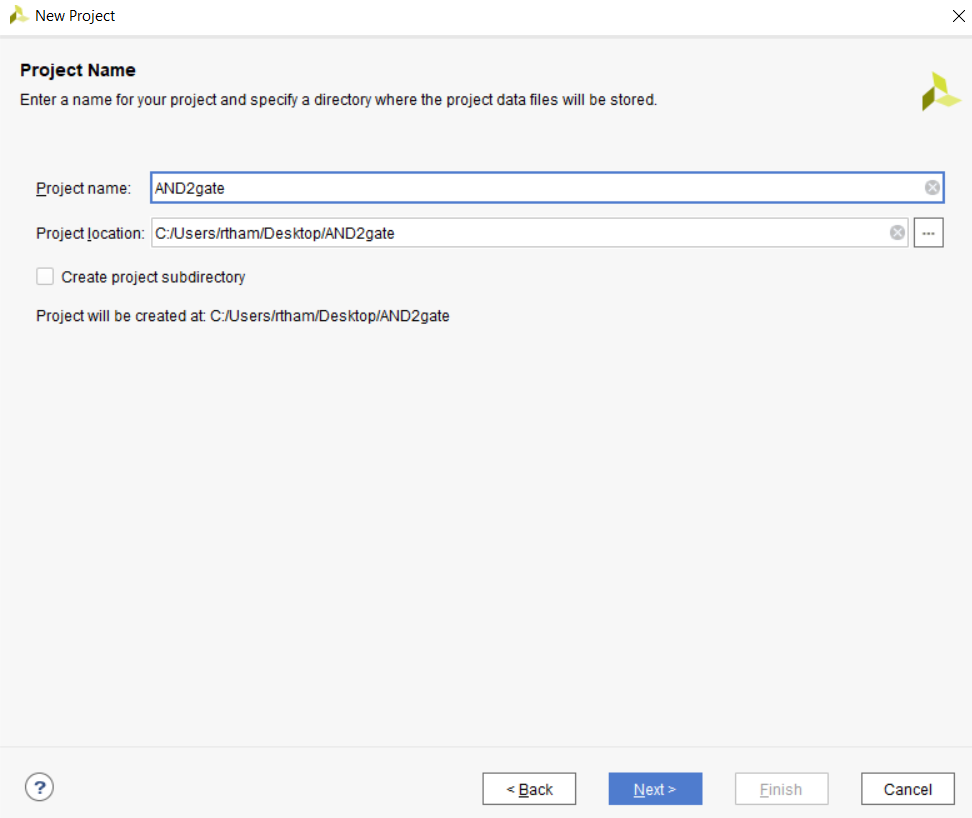


* 1. Enter ***Project name***, the project, in this tutorial, is called “**AND2gate**”

Choose your ***Project location***, this project folder is stored at “**C:/Users/rtham/Desktop/AND2gate**”. Click this button to change to the location/directory/folder that you want to keep the project. After choosing the location, i.e. “C:/Users/rtham/Desktop”, you have to type /AND2gate in the project location.

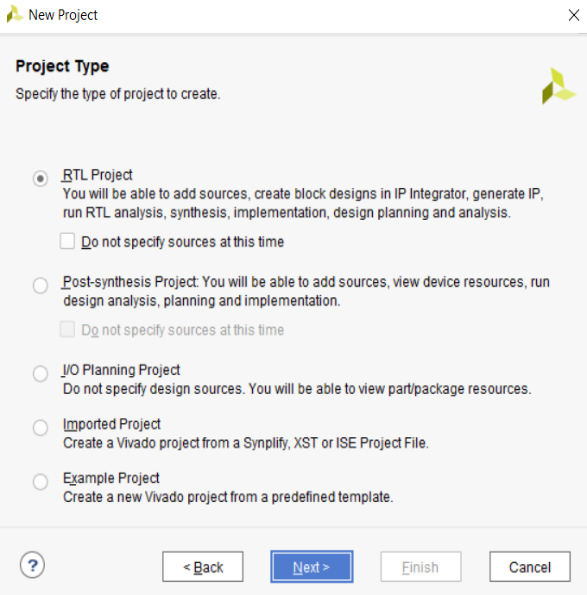
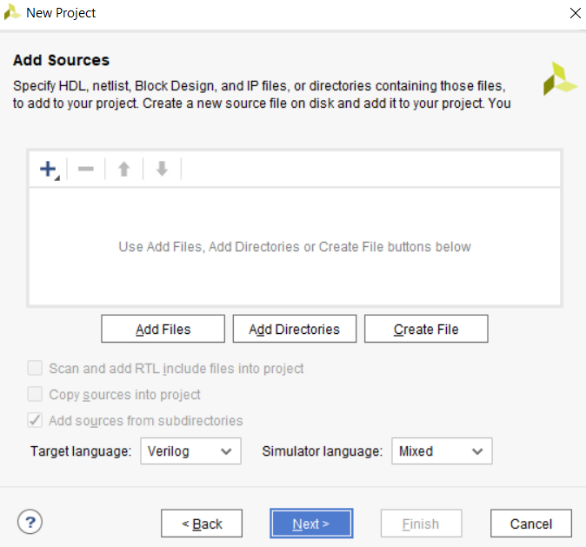
Note: This location makes it easily accessible especially when using computers in ECE 301 lab. When using your own computer, you can choose any location (just make sure that you know where your project is located).

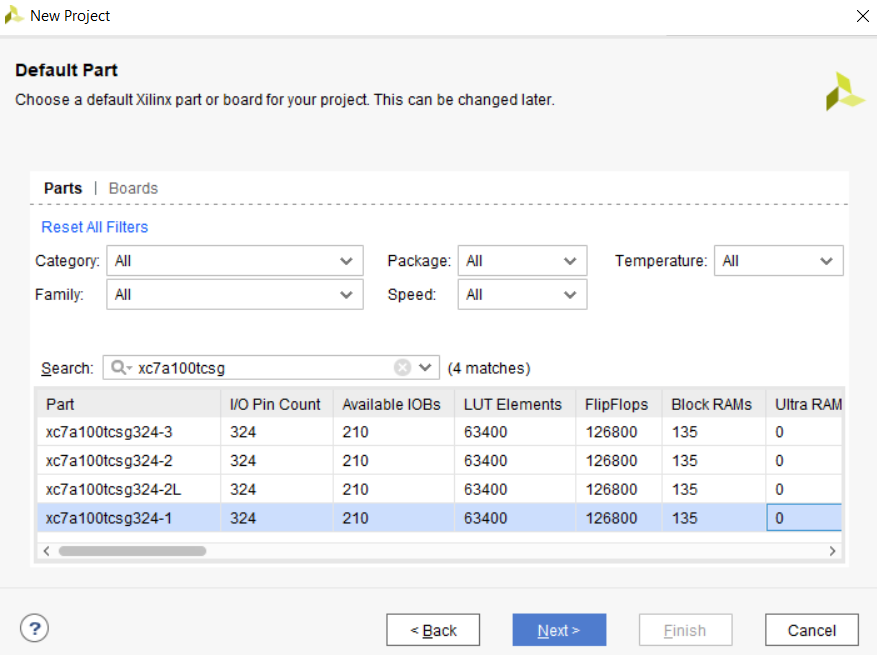
Click ***Next***.



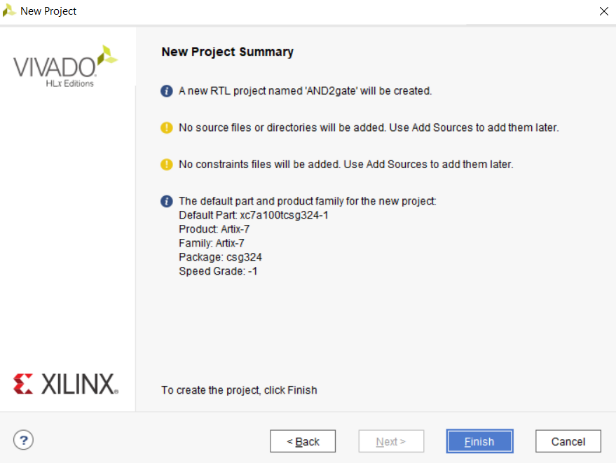
* 1. Choose ***RTL Project*** then click ***Next***. 2.4 On “**Add Sources**” window, make sure

that Target language is **Verilog**. We will add sources (Verilog (.v) files) later. Then click ***Next***.

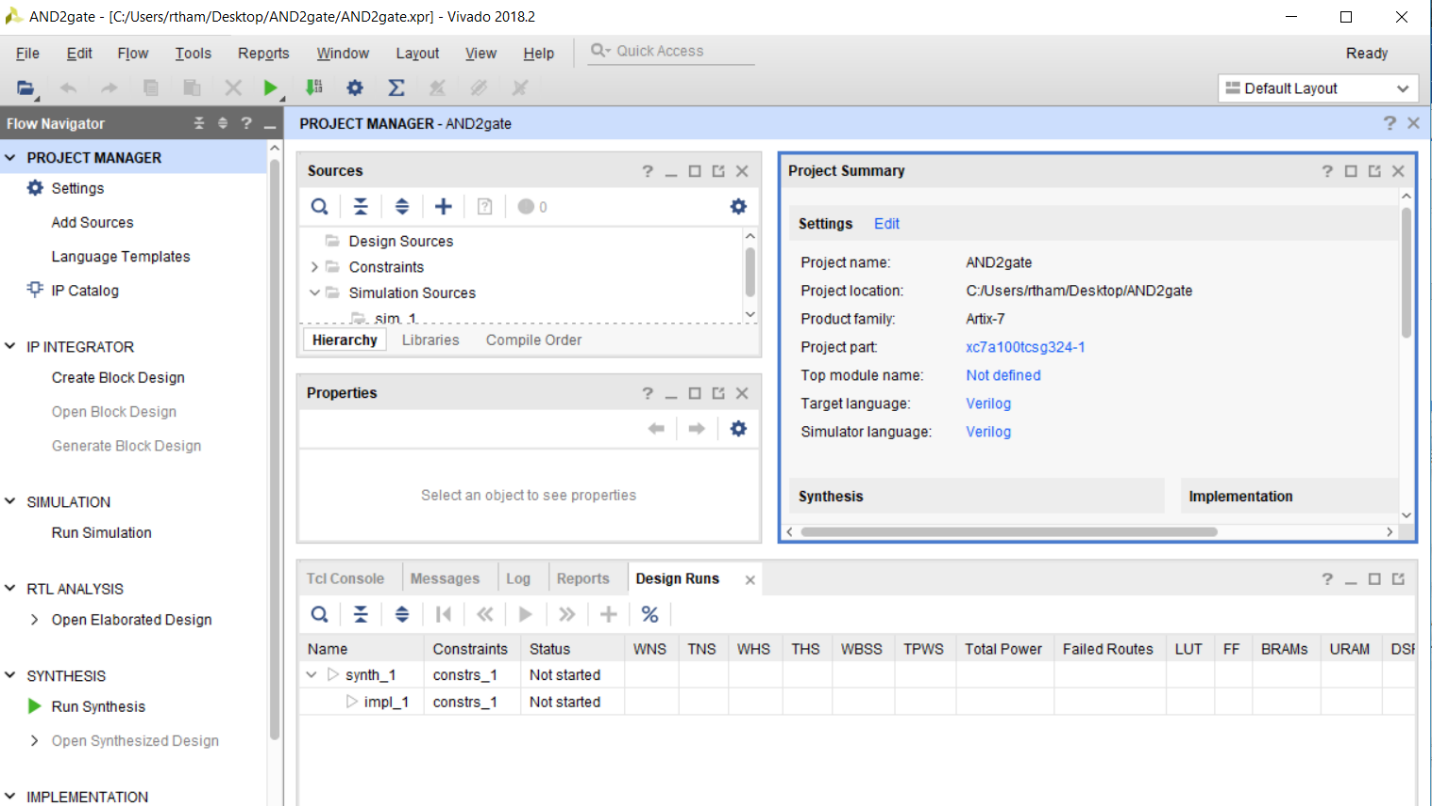
 

* 1. On “**Add Constraints** (optional)” window, click ***Next*** since we will create/add a constraint file later.
  2. On “**Default Part**” window, **type xc7a100tcsg in the Search box**, then choose **xc7a100tcsg324-1**. Click ***Next***. 

The New Project Summary window appears. Click ***Finish***. At this step, you should see the project/folder called “AND2gate” on Desktop (or at the location that you chose in step 2.2).

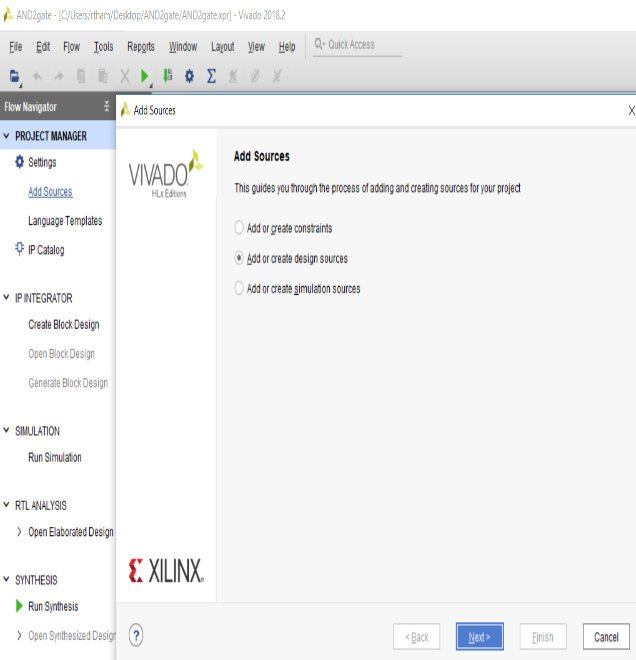
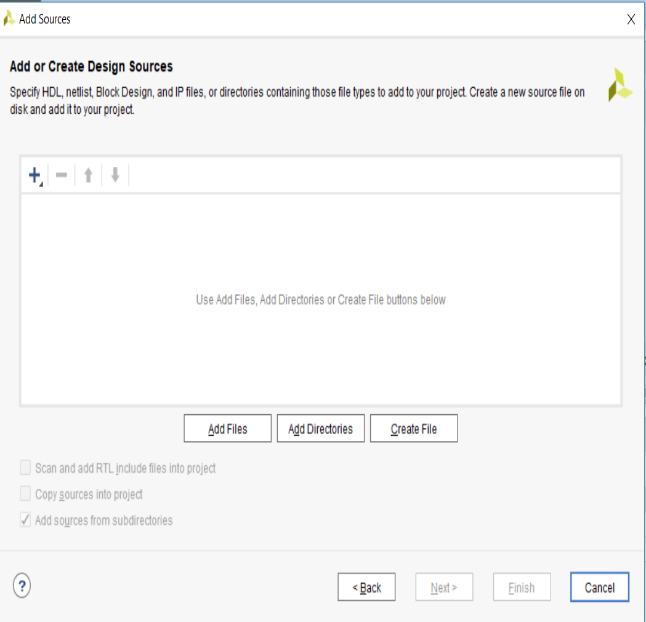


You should see the AND2gate project as shown on the next page. Observe the **Flow Navigator** column on the left-hand side. This is where it will guide you to the steps needed to design and implement your digital circuits/systems.

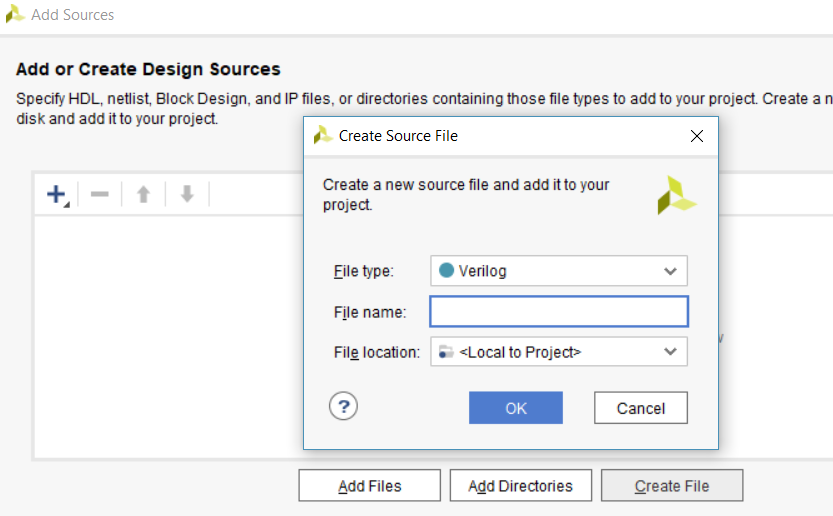


**Step 3**: Create a Verilog (.v) file

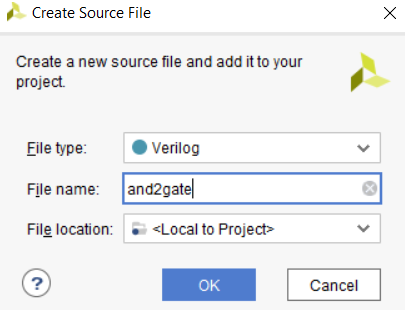
3.1 Click “**Add Sources**” and then choose “**Add or create design sources**”. Click ***Next***.

Click on **Create File**

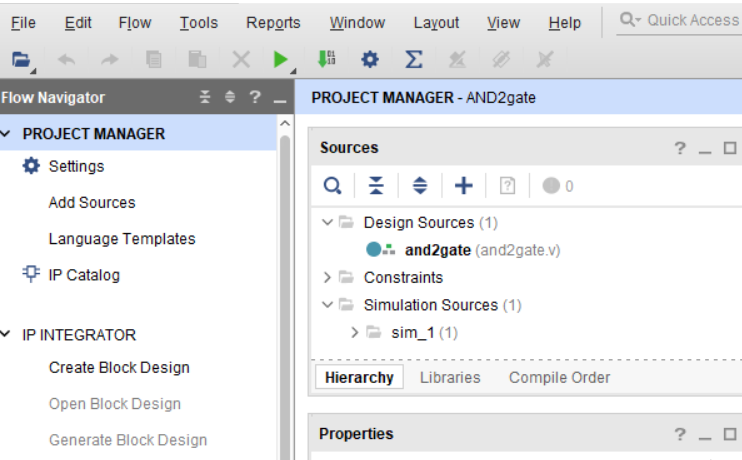


3.2 In **File name** box, enter the desired file name. For this tutorial, ***and2gate*** is used. Click ***OK***.

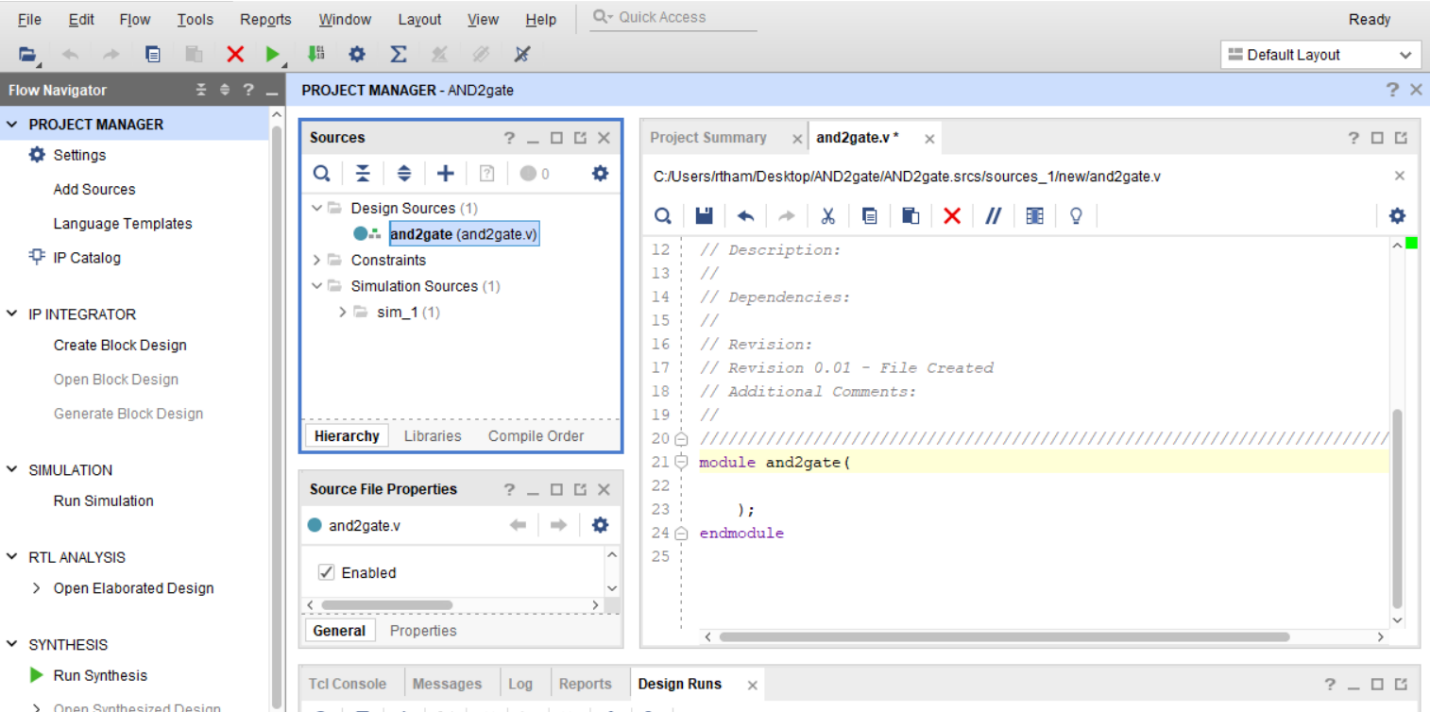


3.3 Then ***Finish*** button in the **Add Sources** window. Then, *Define Module* dialog window will pop up, you can specify input and output, but we leave it as is. Just press ***OK*** button. Then Click **Yes** in the next pop-up window to finish creating the file.

Now, you should see ***and2gate.v*** file is added to the project under Design sources.



***Double-click*** on **and2gate.v** to open the file (to see its content). You are now ready to write Verilog description of the and2gate module functionality.



3.4 Writing a Verilog description (.v files)

Observe that Vivado has already entered a comment section (lines begin with // and in gray-colored) along with a couple of lines of code for us.

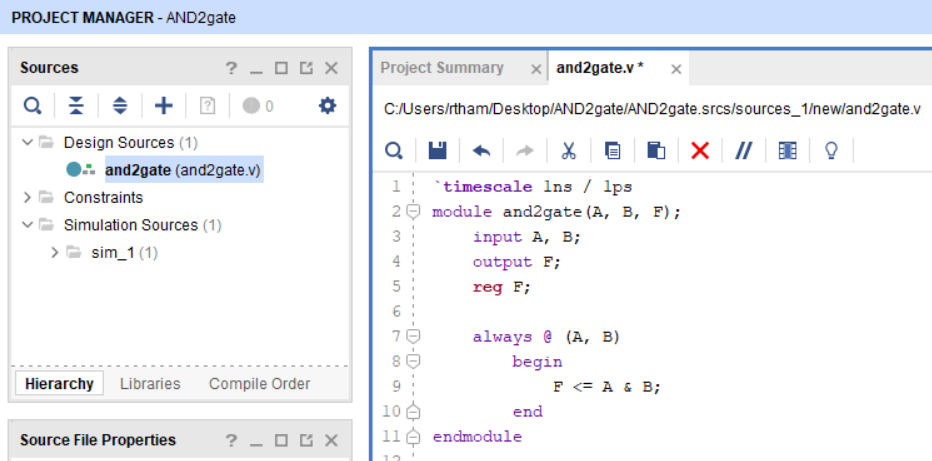
* The line **`timescale 1ns / 1ps** is located at the top of the file. The Verilog language uses dimensionless time units, and these time units are mapped to “real” time units within the simulator.

**`timescale** is used to map to the “real” time values using the statement `timescale <time1> / <time2>, where <time1> indicates the time units associated with the #delay values, and the <time2> indicates the minimum step time used by the simulator.

* The and2gate module is also declared using **module and2gate();** and **endmodule**, but the ports are left for us to define.

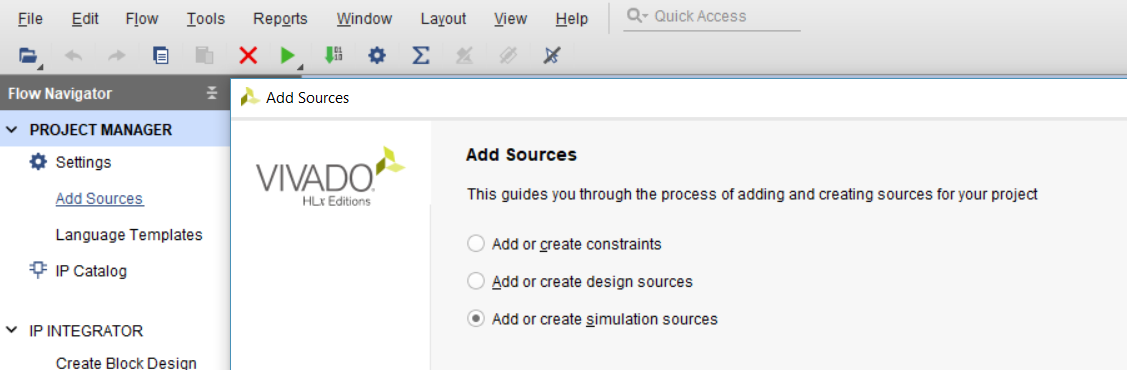
We finish specifying the functionality of the **and2gate** module as shown below. **Type** the code below into **the and2gate.v file**. Then click on File -> Save File (or Ctrl+S).

Note: The \* next to the name and2gate.v disappears when the file is saved.

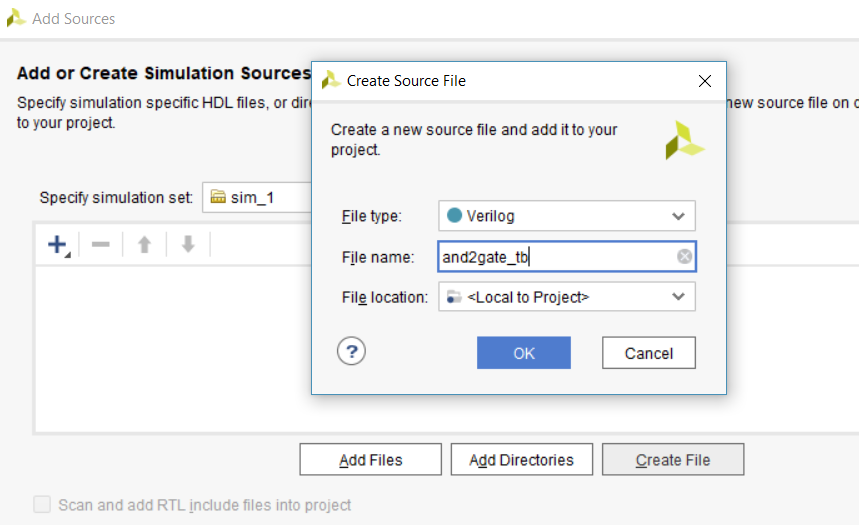


**Step 4: Simulation** (Creating a testbench)

4.1 Click “**Add Sources**” on the Flow Navigator column (on the left-hand side of the project). Then choose “**Add or create simulation sources**” and click ***Next***.



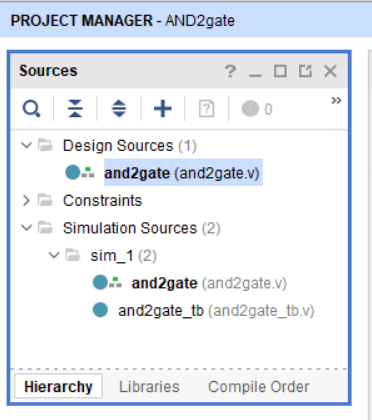
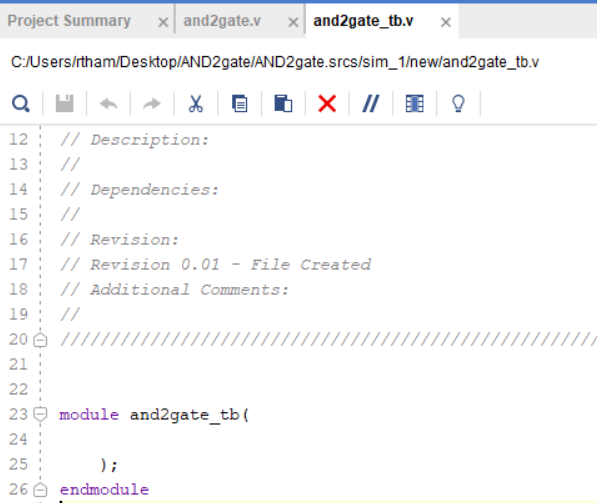
Click on “**Create File**”. In **File name** box, type ***and2gate\_tb.*** Click ***OK***

******

.

Then click ***Finish***, ***OK***, ***Yes*** for the next three windows that are popped-up (same as step 3.3).

Under Simulation Sources, click **>** at **** to expand sim\_1 and you should now see that ***and2gate\_tb.v*** file is added to the project. Double-click on the name to see its content.

Type the following testbench (Verilog) code into and2gate\_tb.v file and then save it (Ctrl+S).

`timescale 1ns / 1ps

module and2gate\_tb();

reg A\_t, B\_t;

wire F\_t;

and2gate and2gate\_1(A\_t, B\_t, F\_t);

initial

begin

// case 0

A\_t<=0; B\_t<=0;

#1 $display("F\_t = %b", F\_t);

// case 1

A\_t<=0; B\_t<=1;

#1 $display("F\_t = %b", F\_t);

// case 2

A\_t<=1; B\_t<=0;

#1 $display("F\_t = %b", F\_t);

// case 3

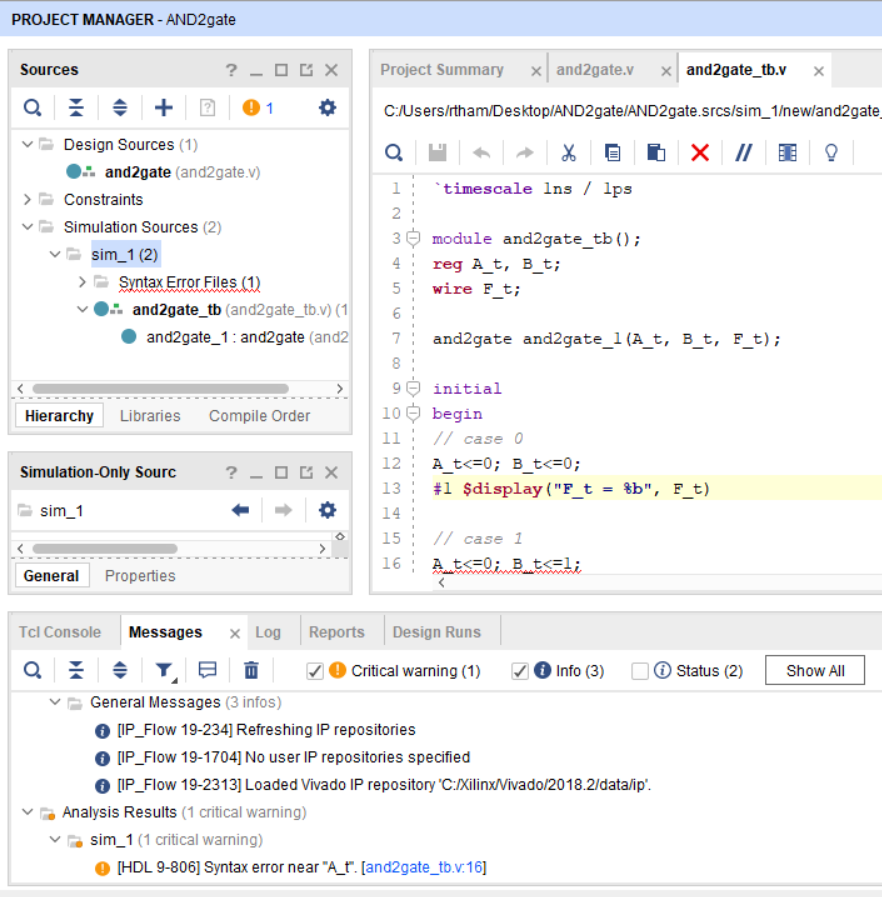
A\_t<=1; B\_t<=1;

#1 $display("F\_t = %b", F\_t);

end

endmodule

Note: How do you know whether there are syntax errors in your Verilog file? In the example below, I intentionally left out ; in one of the statement below.



Then, what you have to do is correct

that syntax error. If the file no longer

has error, the “Syntax Error Files” will

disappear which means you are ready

for simulation.

Click onti, it will show

First, “Syntax Error Files (1)” show up.

Second, under the “Messages” tab,

It will show where the syntax error is at.

Most of time, it will be 1-2 lines above

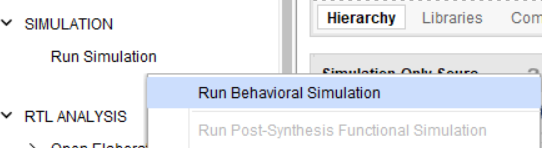
where it specifies.

Click onti, it will show

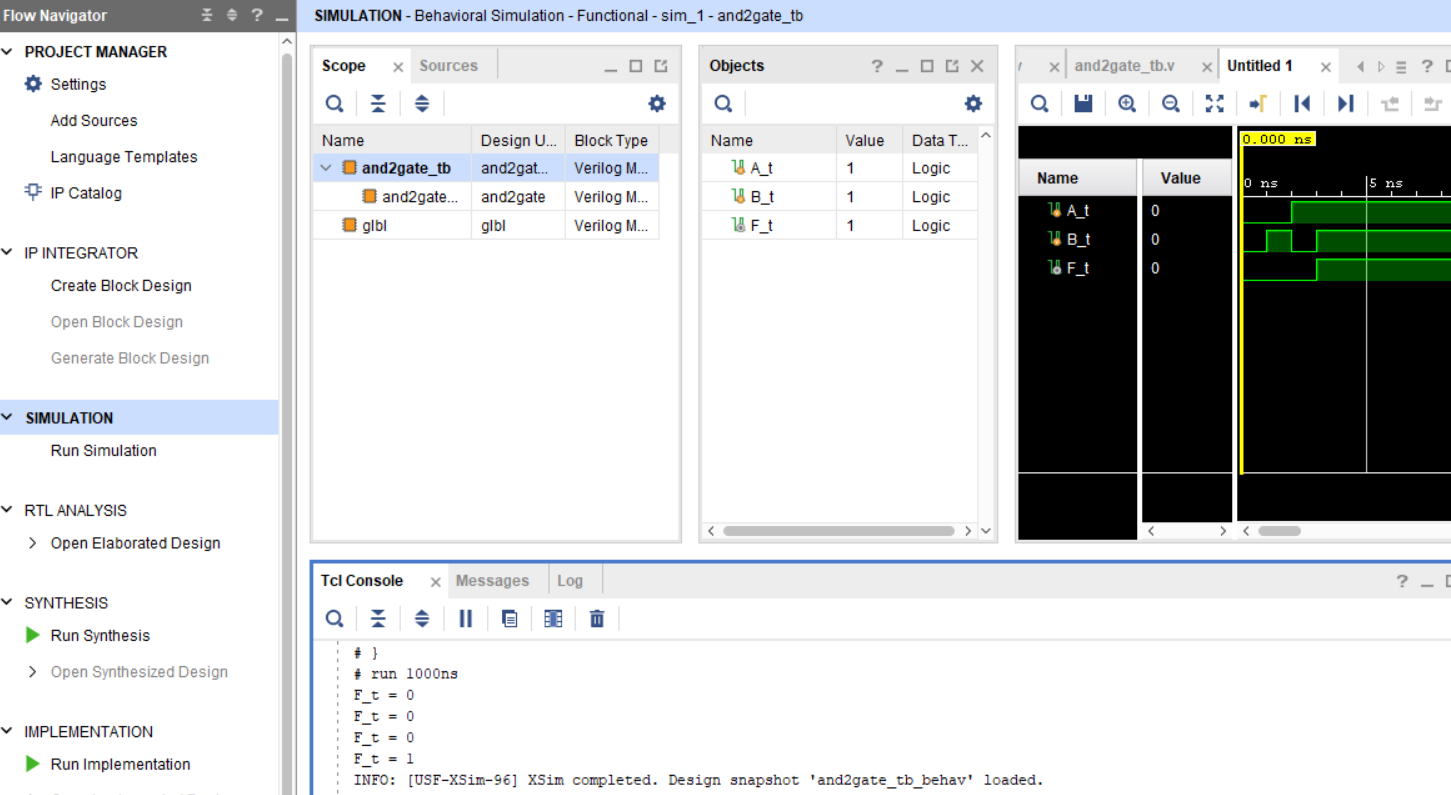
missing ;

**4.2 Simulation**

* Click on ***Run Simulation*** (on Flow Navigator column) and ***Run Behavioral Simulation***



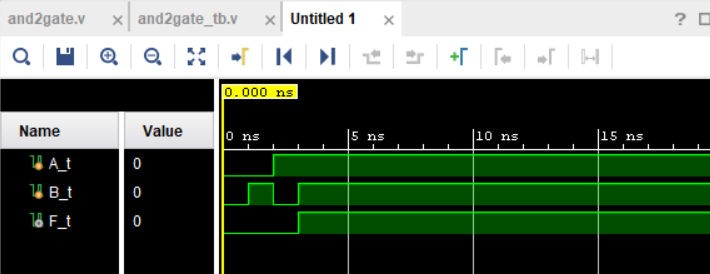
The testbench and source files will be compiled and the Xsim simulator will be run (assuming no error). You should see the simulator output similar to the one shown on the next page.



* In “Scopes” window, you can see that ***and2gate\_tb*** uses the module ***and2gate*** https://lh6.googleusercontent.com/pt9hPtKUEhUi8UpSfkGgKtESS8Z7aTKbZ046gEYcVLvxILWj1teQYnuC0UcCypm22i1MBr9drokwQ9prDisMoSegCEu9uLjAz5EnxoC3tu01sYfNlKlL0kMkjHt_ZKWAJufh2yc
* The “Objects” window show the top-level signal (inputs and output of the testbench module).
* The waveform window show the simulated waveform.
* The Tcl console displays the simulation activities. Since the testbench contains “$display …” statements, the output F\_t is displayed for each case here (you may have to scroll up or down in your Tcl Console to see the output F\_t)

On the **waveform window**,

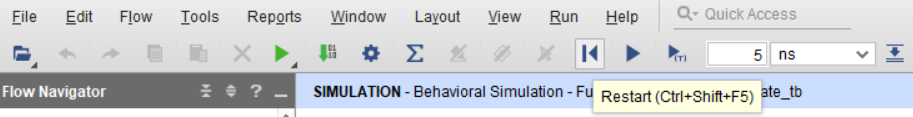
* To view the beginning of simulation, click on  (Go to Time 0) button.



* Click on  (Zoom in) button until you see the simulated waveform. Since this is simulation of the 2-input and gate, the output F\_t is correct (F\_t is 1 only when both A\_t and B\_t are 1)

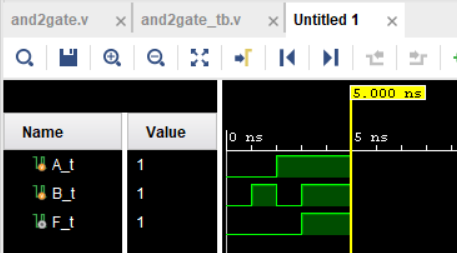
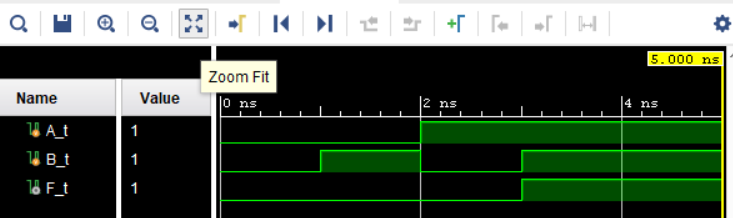
Note: We can simulate for a specific length of time (the above uses 1000 ns whereas our simulation needs about 4 ns as you can see from the waveform above).

Under the main menu bar, we will first restart the simulation by clicking on  (restart) button. We will now have an empty waveform.



Second, enter the desired simulation time. Type in 5 (choose ns from the dropdown arrow) and then click on  to run for 5ns

Now, you should see that the simulated waveform is stop at 5 ns. Click on  to make the entire waveform fit in the window.

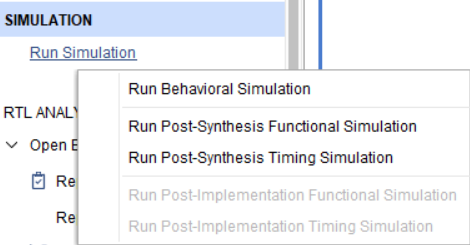
4.3 Close Simulation: If you want to close a simulation, click ***File*** (on menu bar) and then choose ***Close Simulation***.

At this point, the simulated waveform (output) should be correct (the output waveform should show the correct value for any given input sequence). If it is not, it is very likely that your Verilog code does not describe the designed circuit correctly – check the code.

Two more simulations that we can do

**1) Post-Synthesis Functional Simulation**

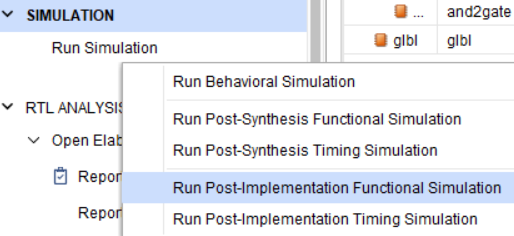
* On **Flow Navigator**, left-Click on **Synthesis** to expand (if needed), choose **Run Synthesis**. Click ***Ok*** on the pop-up window to get started.
* When it finishes, the window asking you to Run Implementation will pop up. Click ***Cancel***.
* Left-Click on ***Run Simulation*** (on Flow Navigator column) and ***Run Post-Synthesis Functional Simulation.***



* The window containing the simulated waveform should show up. For this simple circuit, the output waveform should look the same as the one generated from the Behavior simulation.

**2) Post-Implementation Functional Simulation**

* On **Flow Navigator**, left-Click on **Implementation** to expand, choose **Run Implementation**. Click ***Ok*** on the pop-up window to get started.
* When it finishes, the window asking you to Open Implemented Design will pop up. Click ***Cancel***.
* Left-Click on ***Run Simulation*** (on Flow Navigator column) and ***Run Post-Implementation Functional Simulation***



* The window containing the simulated waveform should show up. For this simple circuit, the output waveform should look the same as the one generated from the Behavior and Post-Synthesis function simulations.

Your next step is to implement the circuit on the FPGA board – See **XilinxVivado\_SynthesisTutorial\_Nexys-A7** document