Xilinx Vivado 2018.2 Design Tool Guide (Simulation)

Logic and Digital System Design – CS 303

Sabanci University

Fall 2021

Software & Hardware

- For Lab assignments and Term Project, you will use **Xilinx Vivado 2018.2 Design Tool** to make design.
- During a design process, you will

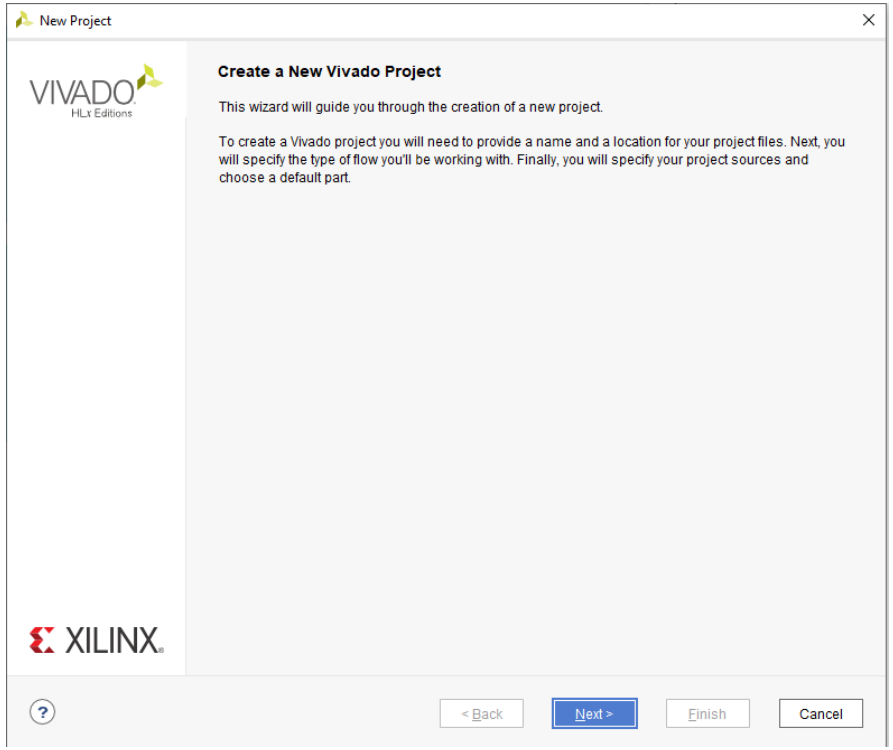
1. make your design on Xilinx Vivado 2018.2 Design Tool.
2. simulate your design on Xilinx Vivado 2018.2 Design Tool.
3. export your design to FPGA.

Creating a New Project

- After you start Xilinx Vivado 2018.2, you may use *Create Project* button to create a new project.

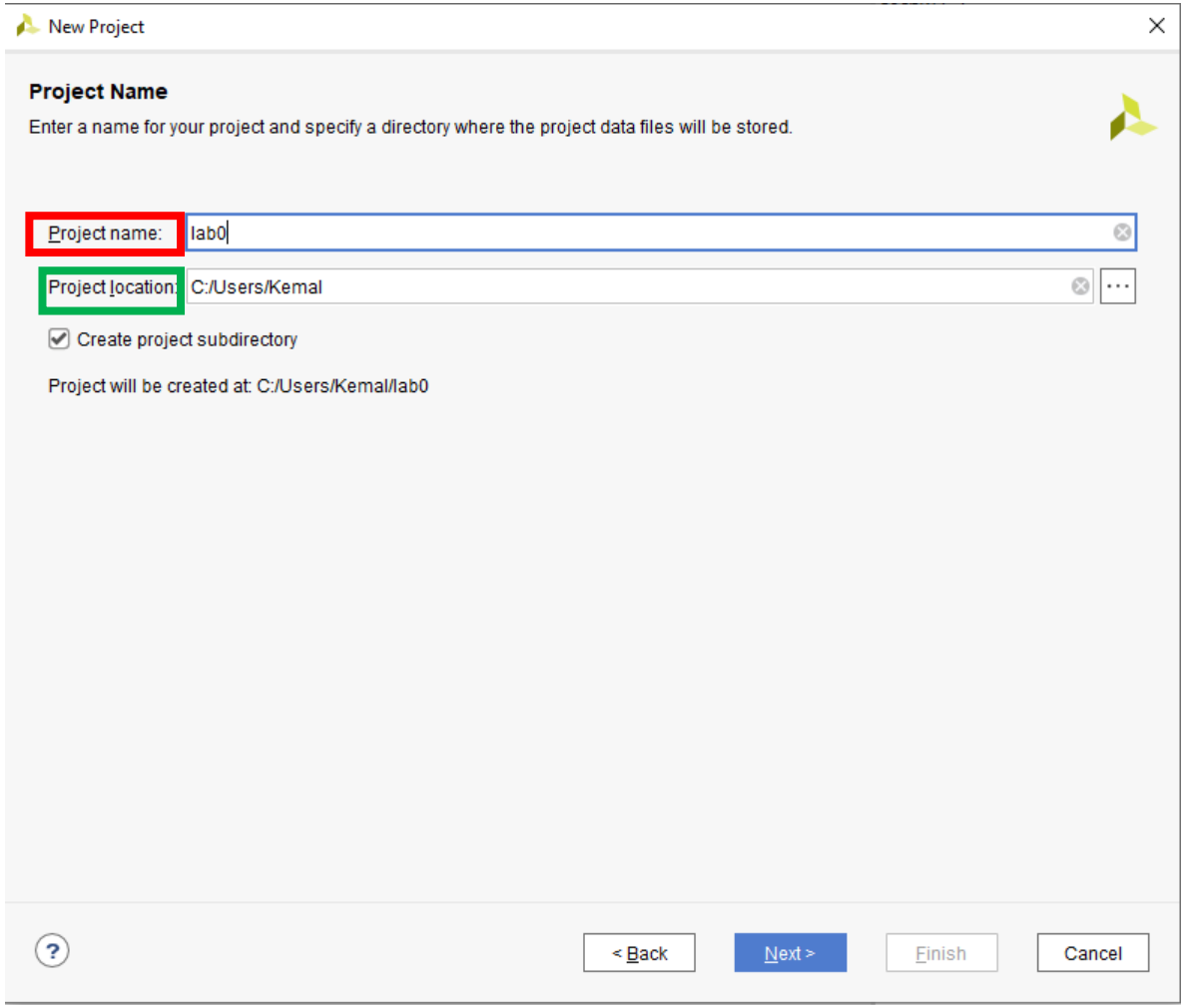
Creating a New Project

- Click on *Next* button.



Creating a New Project

- Specify project name
- Specify project location
- Click on *Next* button



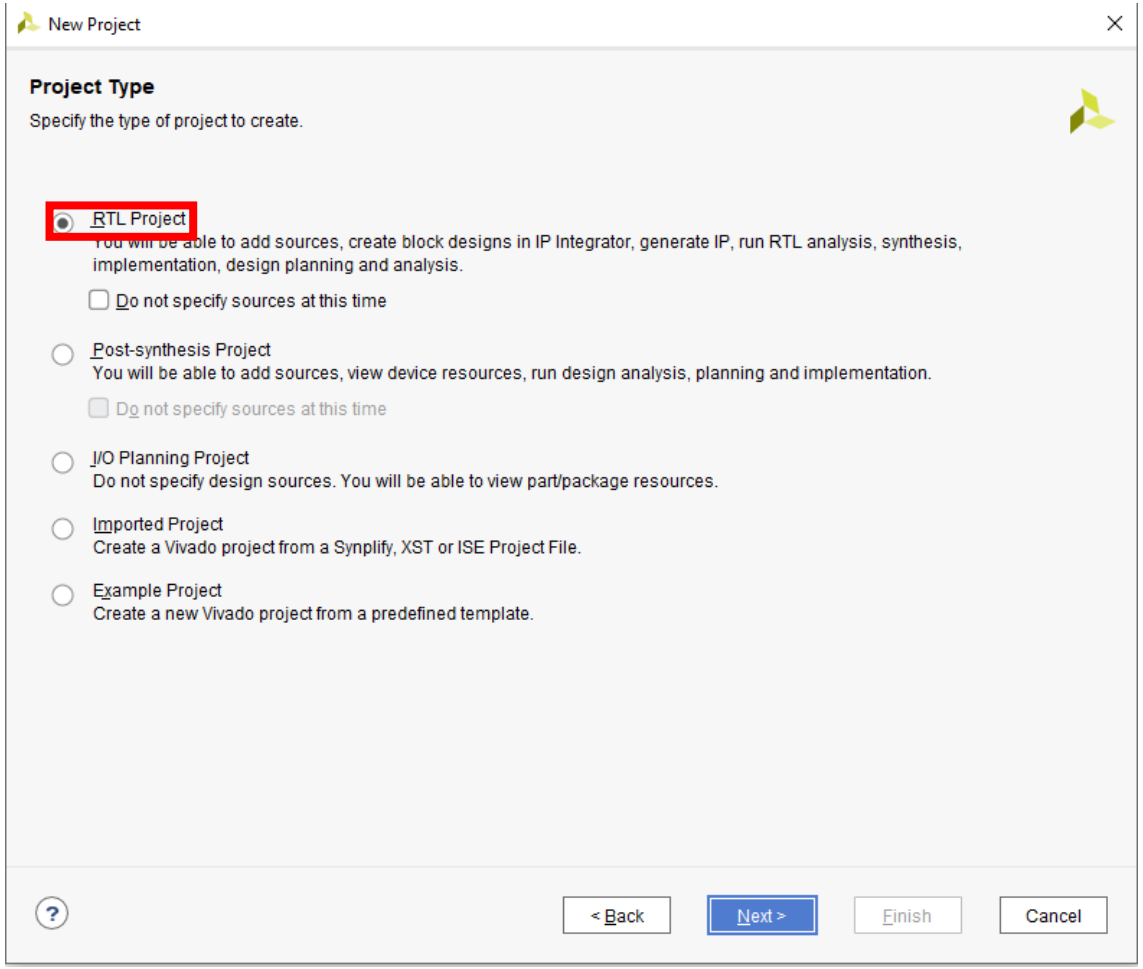
Creating a New Project

- Your project name should not include
- Turkish character
- Space

- Directory names in your project path should not include
- Turkish character
- Space
- e.g. D:\Xilinx\Yeni Klasör => Invalid project path
- e.g. D:\Xilinx\lab1 => Valid project path

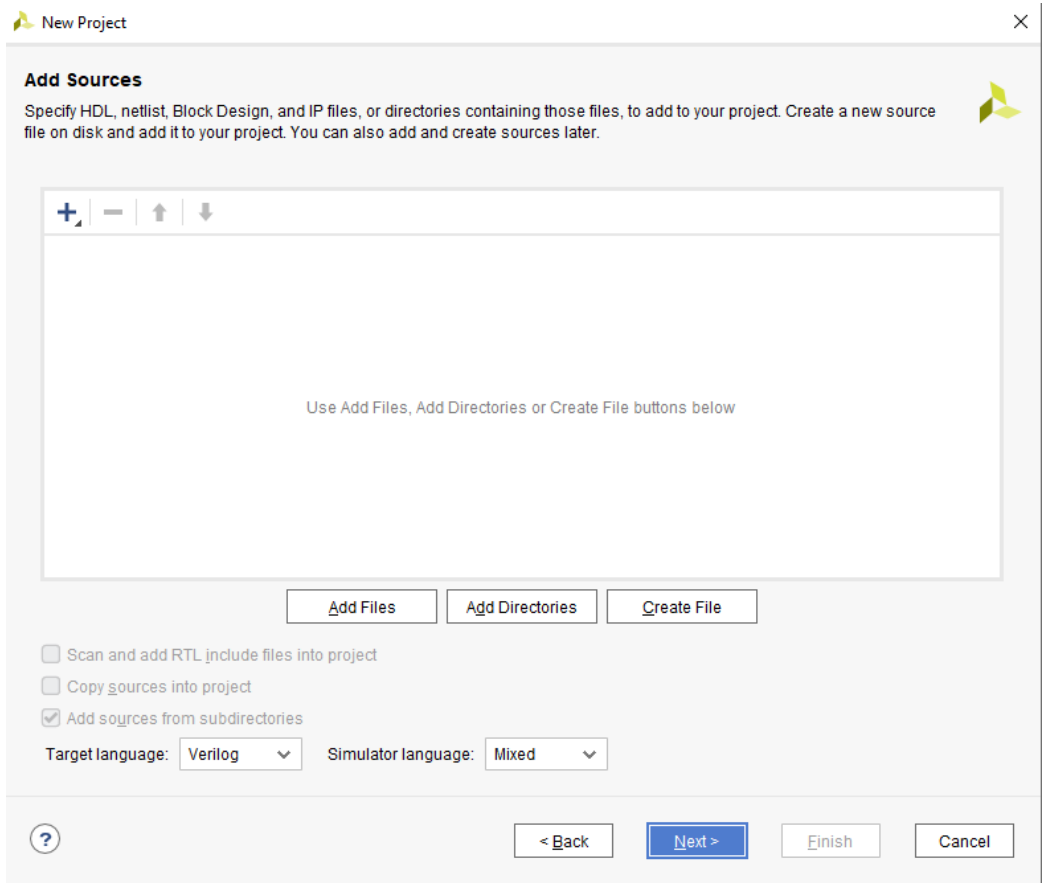
Creating a New Project

- Click on RTL project
- Click on *Next* button



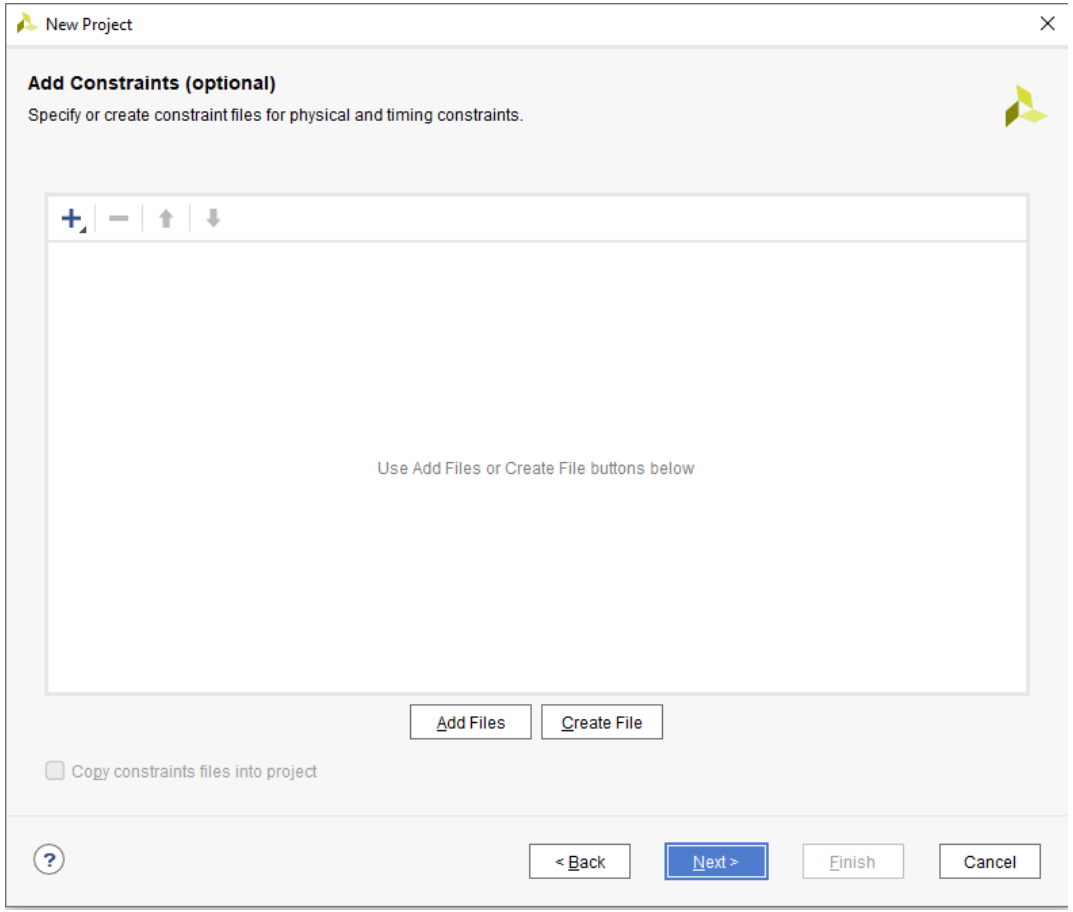
Creating a New Project

- Click on *Next* button



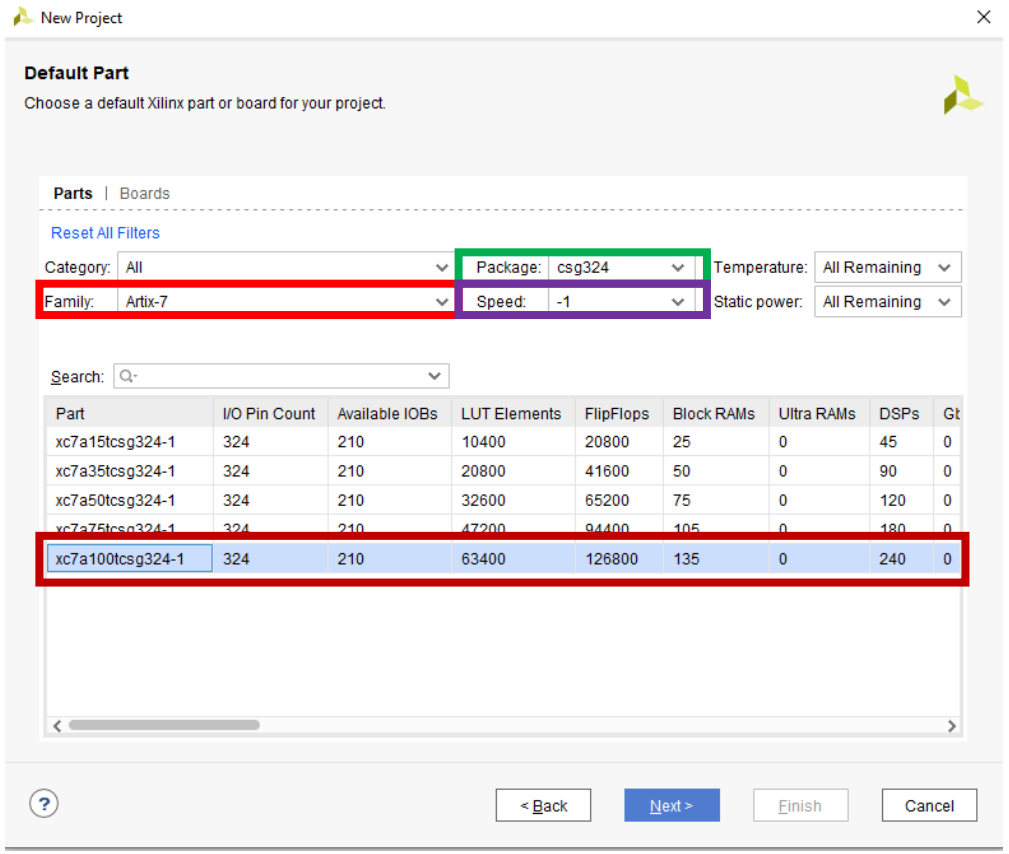
Creating a New Project

- Click on *Next* button



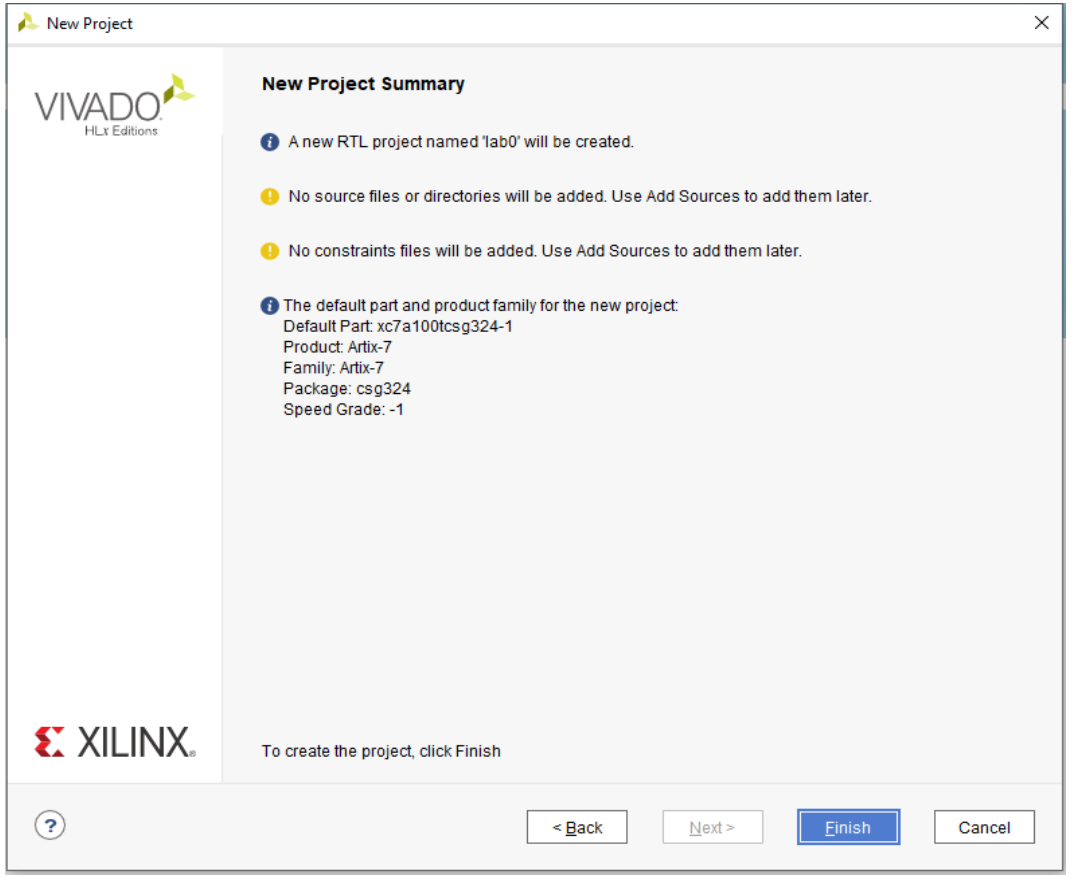
Creating a New Project

- You need to specify device properties.
- Family is Artix-7
- Package is csg324
- Speed is -1
- You need to select xc7a100tcsg324-1
- Click on *Next* button



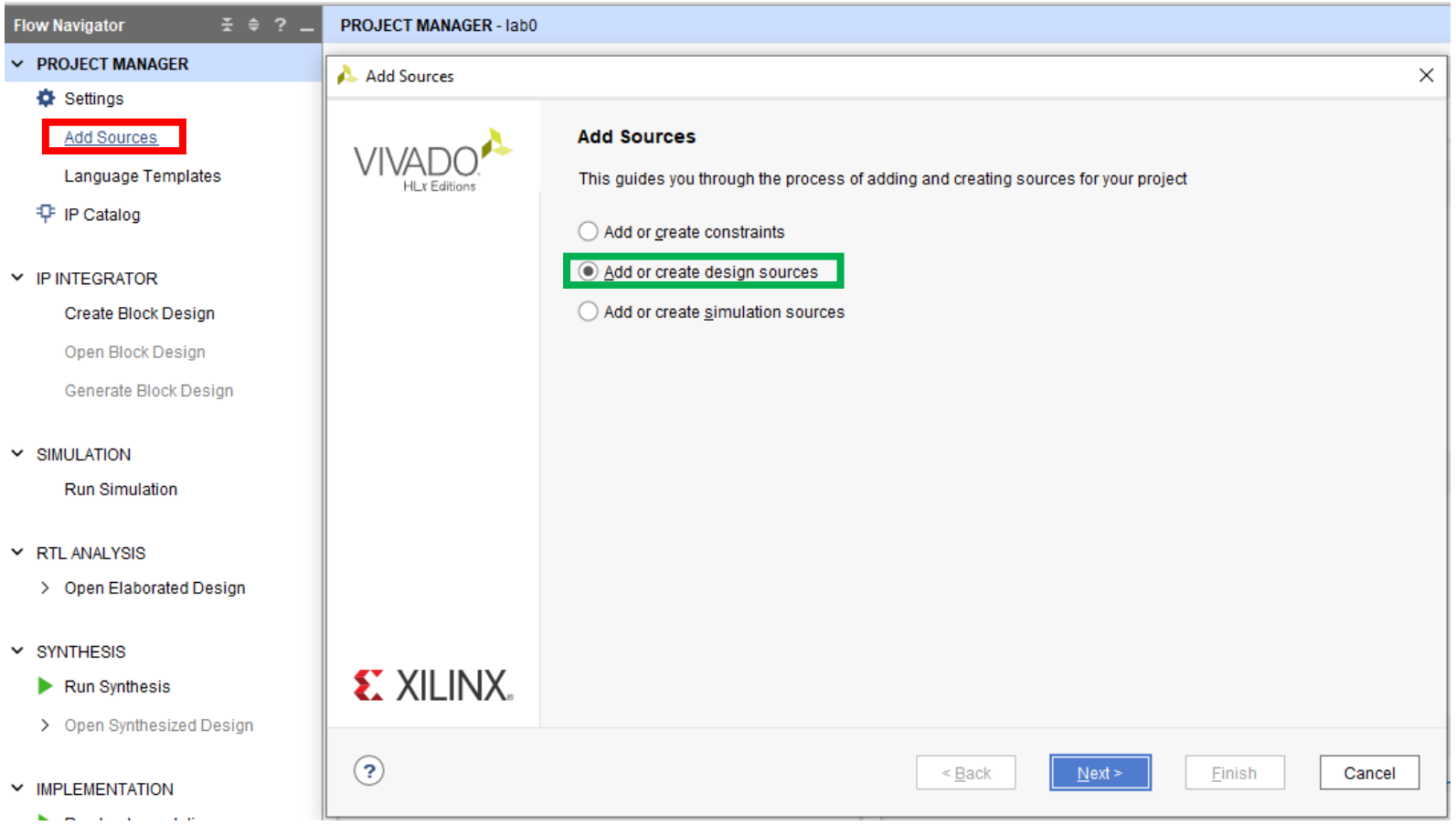
Creating a New Project

- Click on *Finish* button



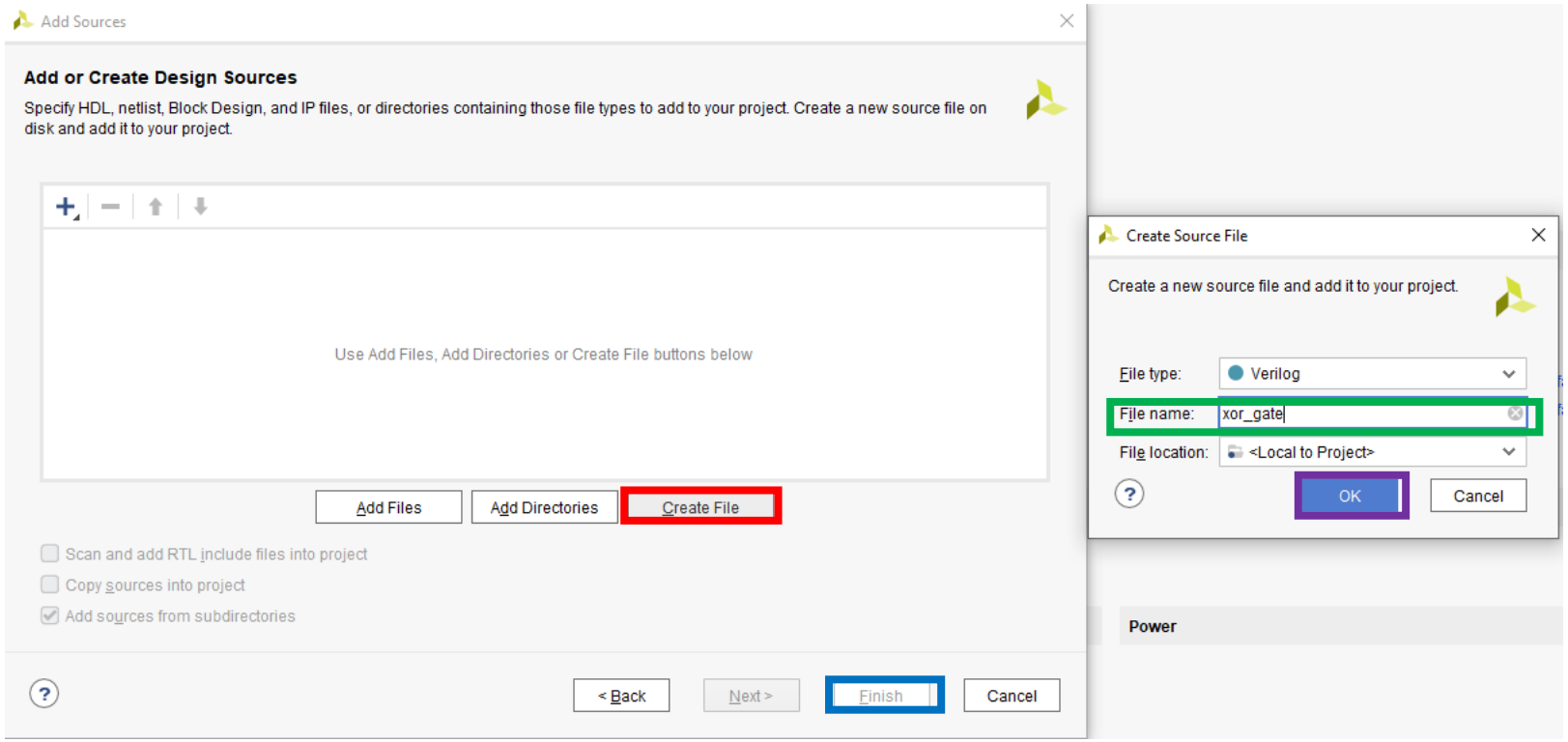
Creating a Source File

- Click on Add Source button
- Choose Add or create design sources
- Click on *Next* button



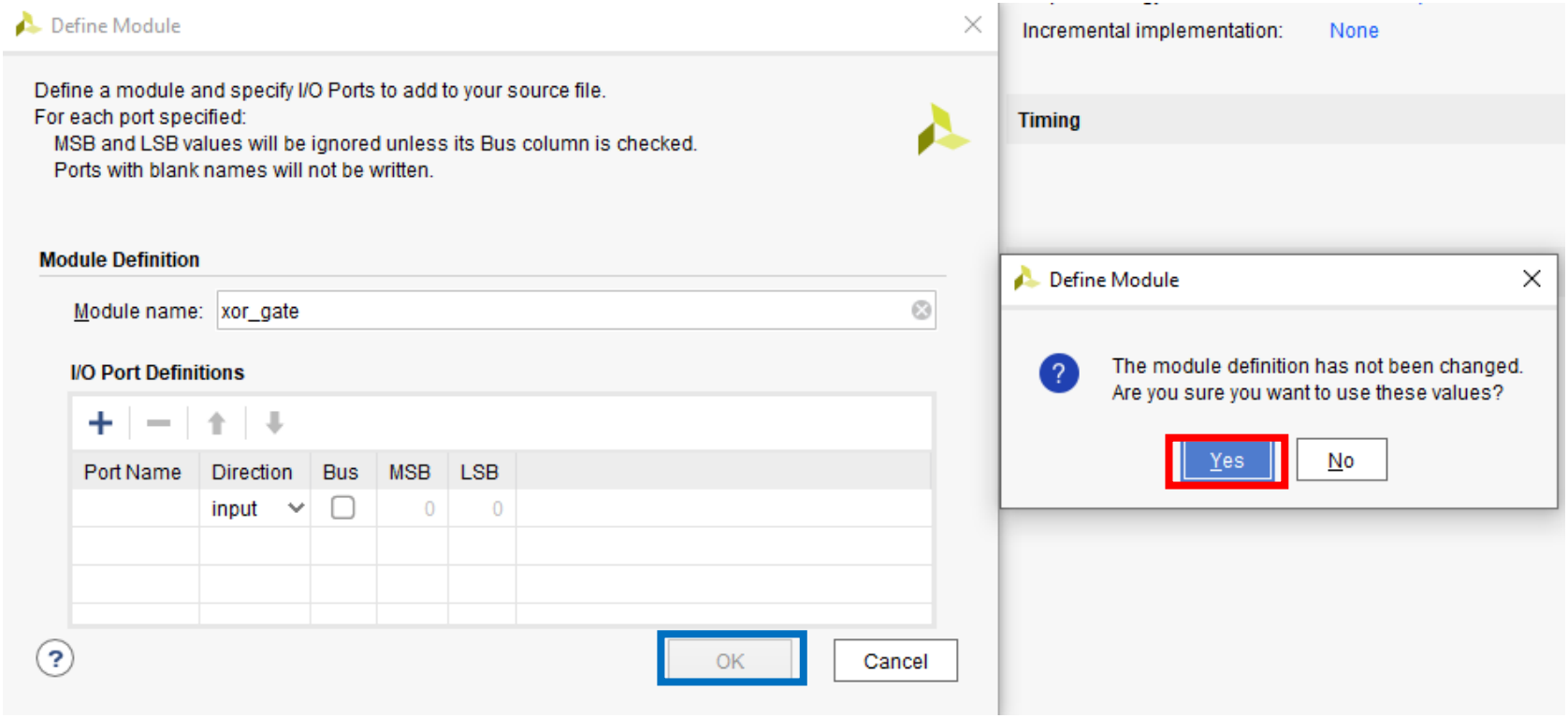
Creating a Source File

- Click on Create File
- Give a name to source file
- Click on OK
- Click on *Finish* button



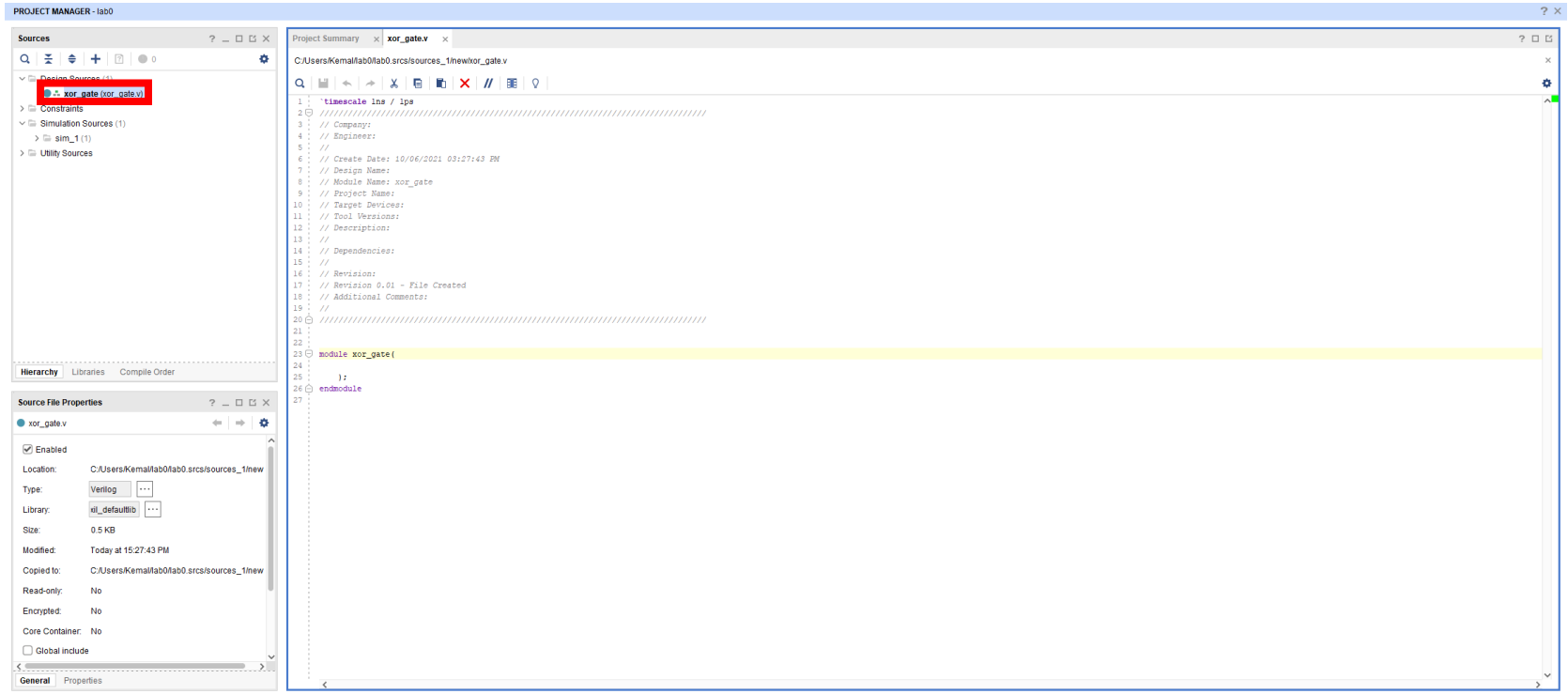
Creating a Source File

- Click on *OK* button
- Click on Yes button



Making a Sample Design

- Double click on your source file
- The editor will open it

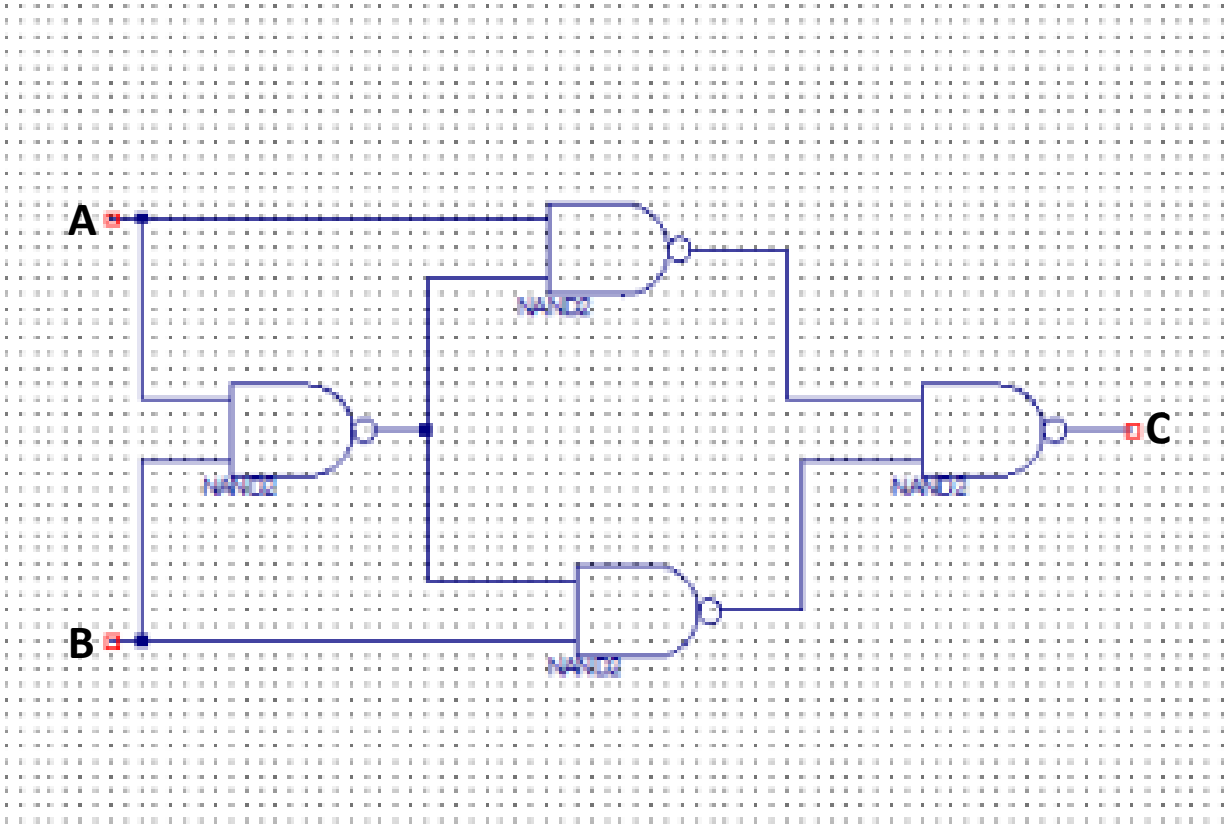


Making a Sample Design

- In this tutorial, we will implement a simple XOR gate using four NAND gates (do not worry about the details.)
- We are going to use *Verilog*
- Verilog is a Hardware Description Language (HDL).
- It used for describing digital circuits.
- You are describing connections between circuit elements.

A Sample Design – XOR gate

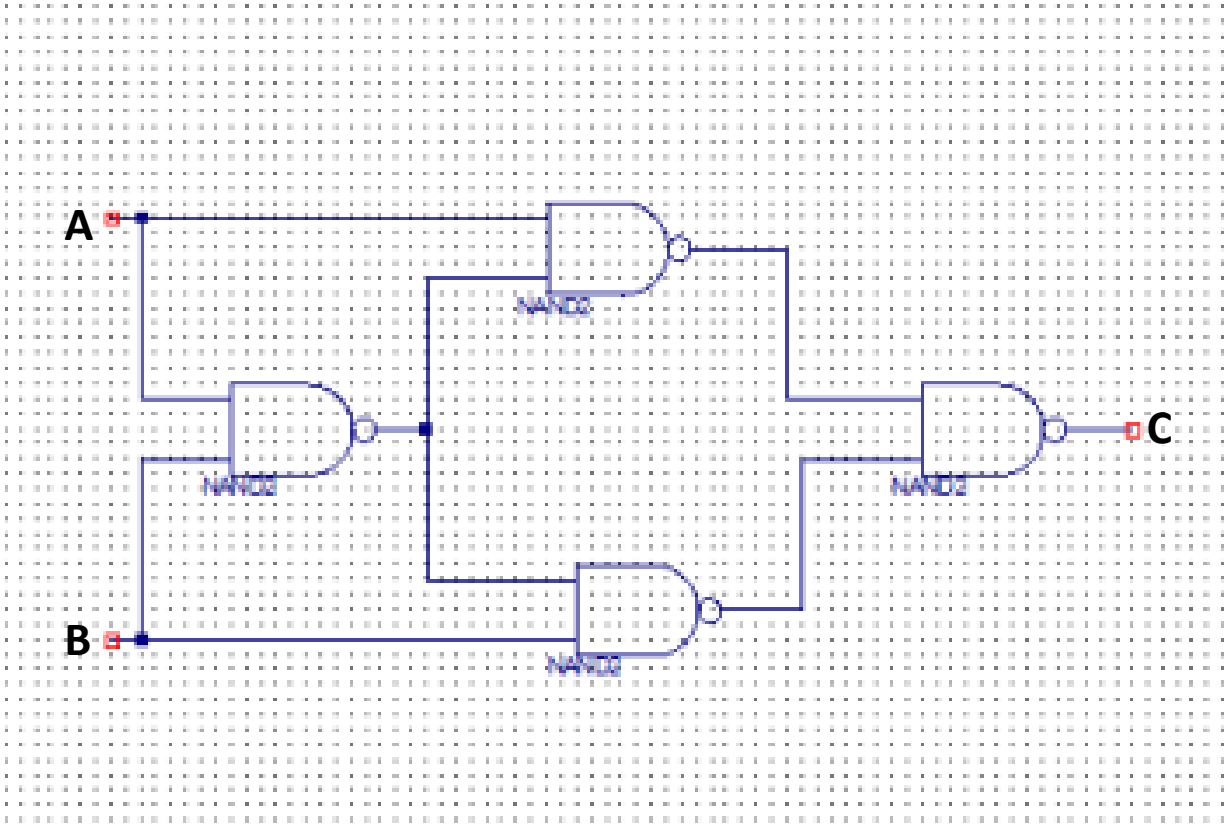
- A XOR gate
- 2 inputs
  - A
  - B
- 1 output
  - C



A Sample Design – XOR gate

- module/endmodule is used to define the design
- A unique name must be given to each design in a project. It has the same name as the source file.

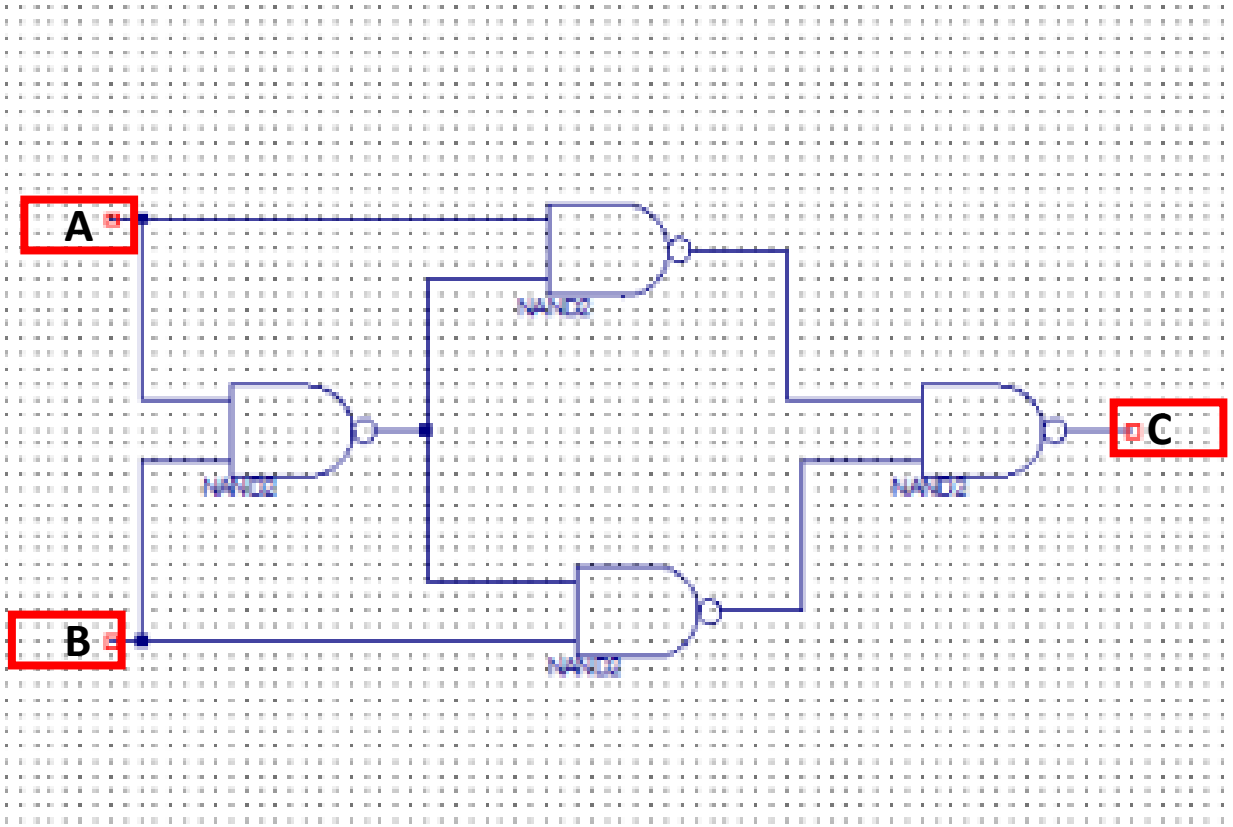
```
22
23  module xor_gate(
24
25      );
26  endmodule
27
```



A Sample Design – XOR gate

- All I/Os must be defined in argument list
- Order of the list is not important.

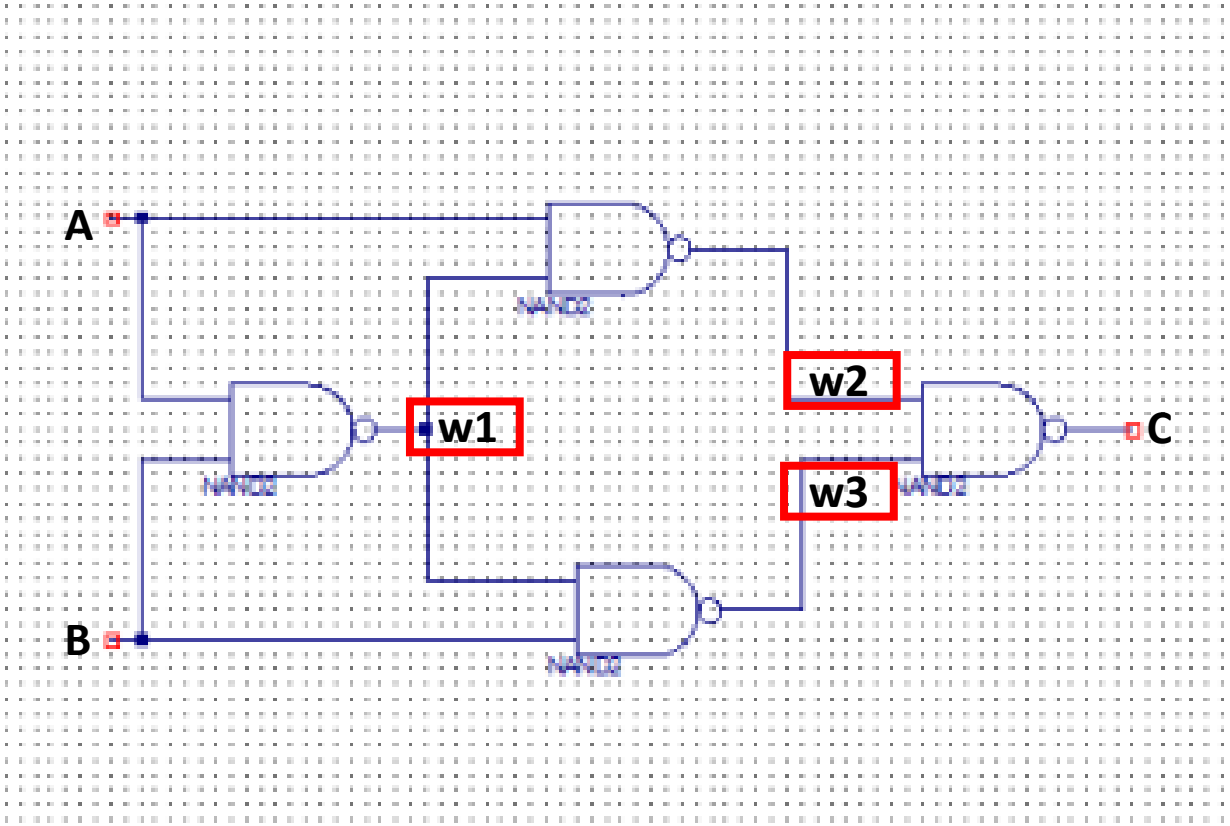
```
22
23  module xor_gate(input  A,
24                  input  B,
25                  output C
26
27      );
28  endmodule
29
```



A Sample Design – XOR gate

- There may be some interconnections between gates
- Gates are connected with nets.
- Nets are defined as wire.

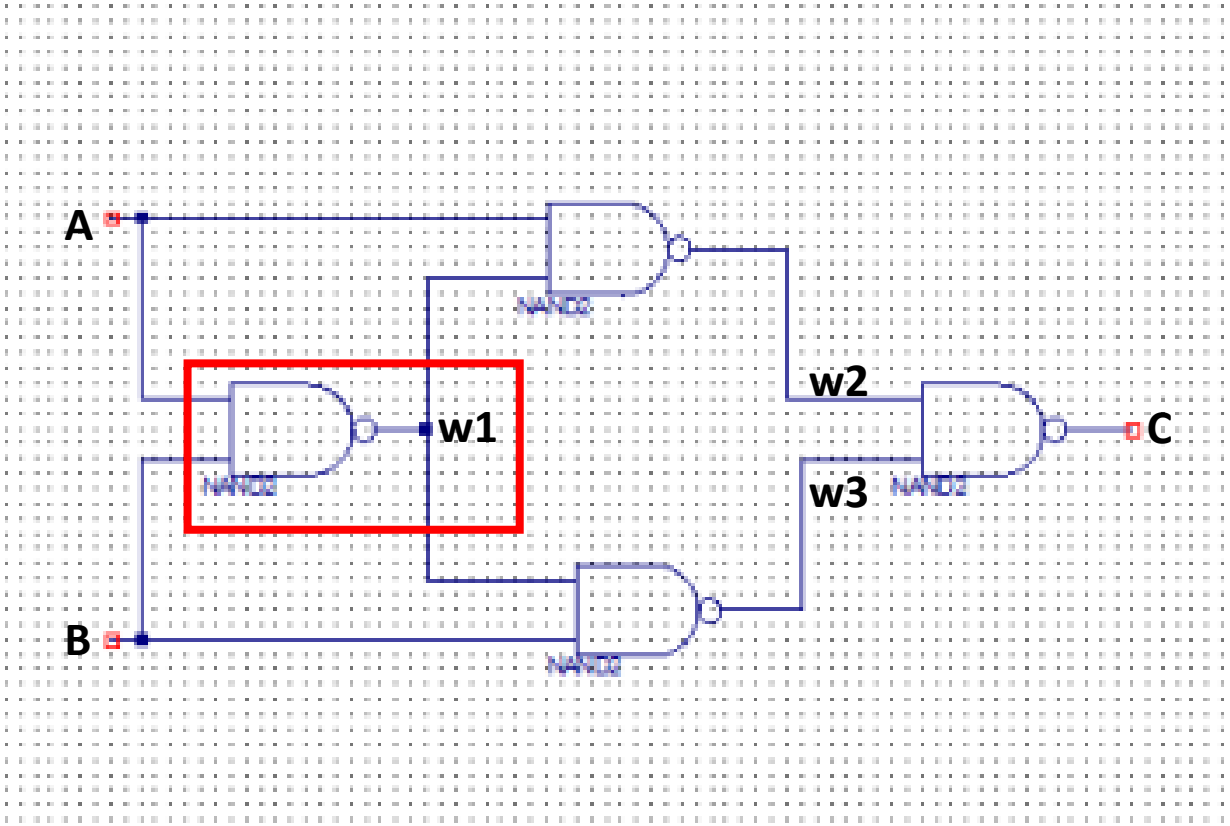
```
21
22
23  module xor_gate(input  A,
24                  input  B,
25                  output C
26
27      );
28
29      wire w1, w2, w3;
30
31  endmodule
32
```



A Sample Design – XOR gate

- assign is used for defining wires.
- & is AND operation
- ~ is NOT operation
- Not + AND = NAND

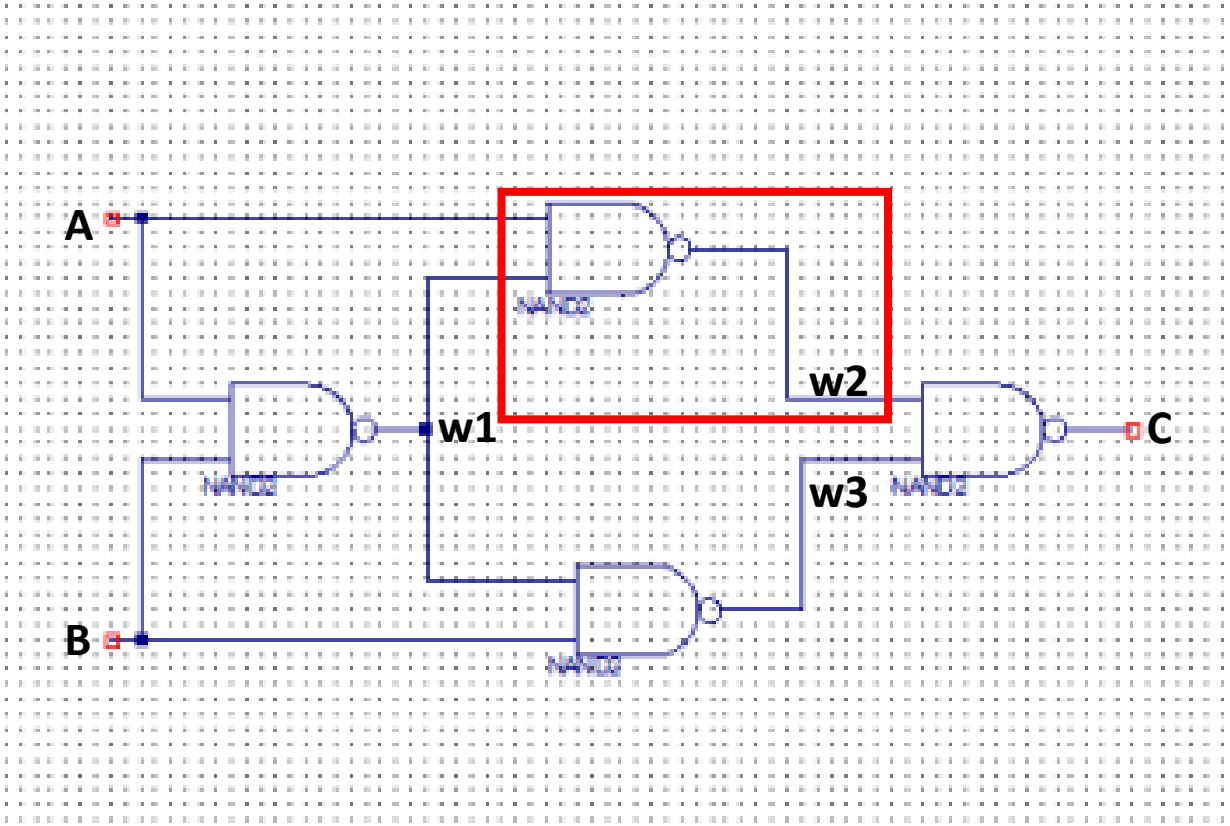
```
22
23  module xor_gate(input  A,
24                  input  B,
25                  output C
26
27      );
28
29      wire w1, w2, w3;
30
31      assign w1 = ~(A & B);
32
33  endmodule
34
```



A Sample Design – XOR gate

- assign is used for defining wires.
- & is AND operation
- ~ is NOT operation
- Not + AND = NAND

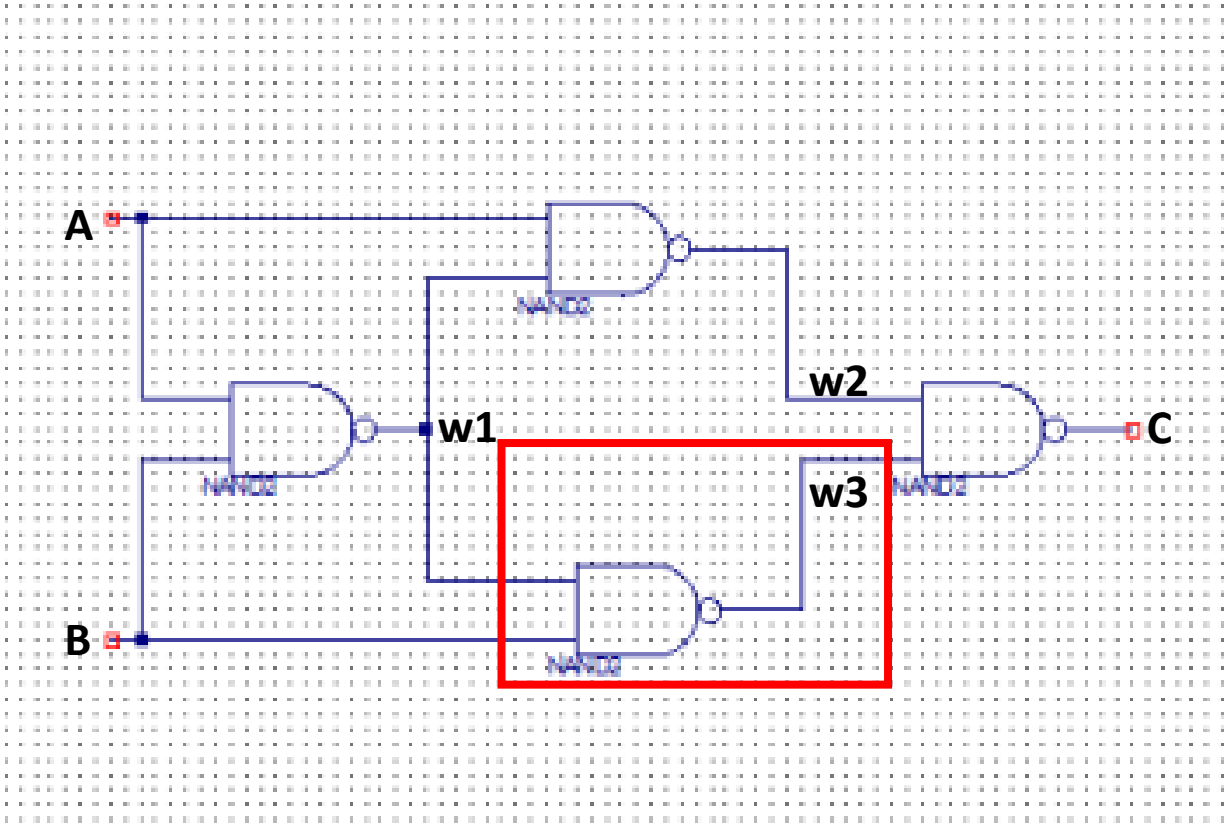
```
22
23  module xor_gate(input  A,
24                  input  B,
25                  output C
26
27      );
28
29      wire w1, w2, w3;
30
31      assign w1 = ~(A & B);
32      assign w2 = ~(A & w1);
33
34  endmodule
35
```



A Sample Design – XOR gate

- assign is used for defining wires.
- & is AND operation
- ~ is NOT operation
- Not + AND = NAND

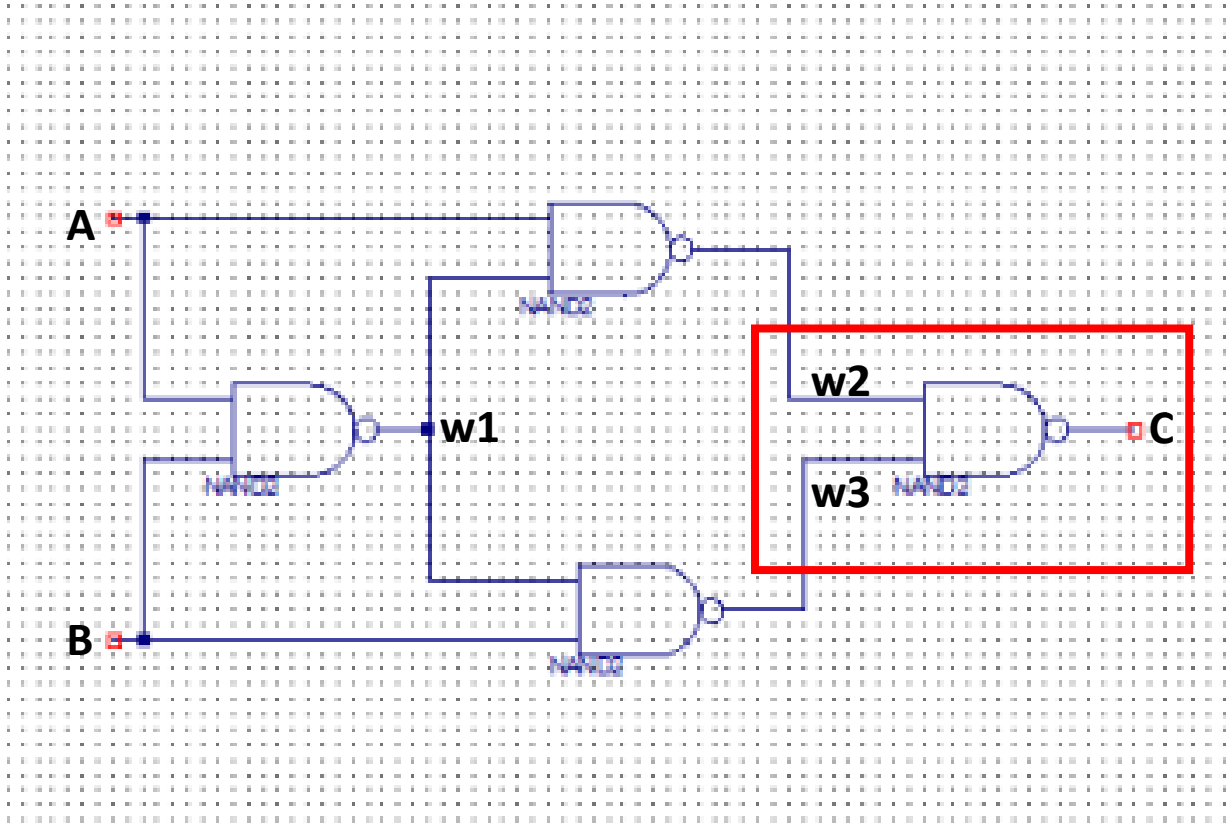
```
22
23  module xor_gate(input  A,
24                  input  B,
25                  output C
26
27      );
28
29      wire w1, w2, w3;
30
31      assign w1 = ~(A & B);
32      assign w2 = ~(A & w1);
33      assign w3 = ~(B & w1);
34
35  endmodule
36
```



A Sample Design – XOR gate

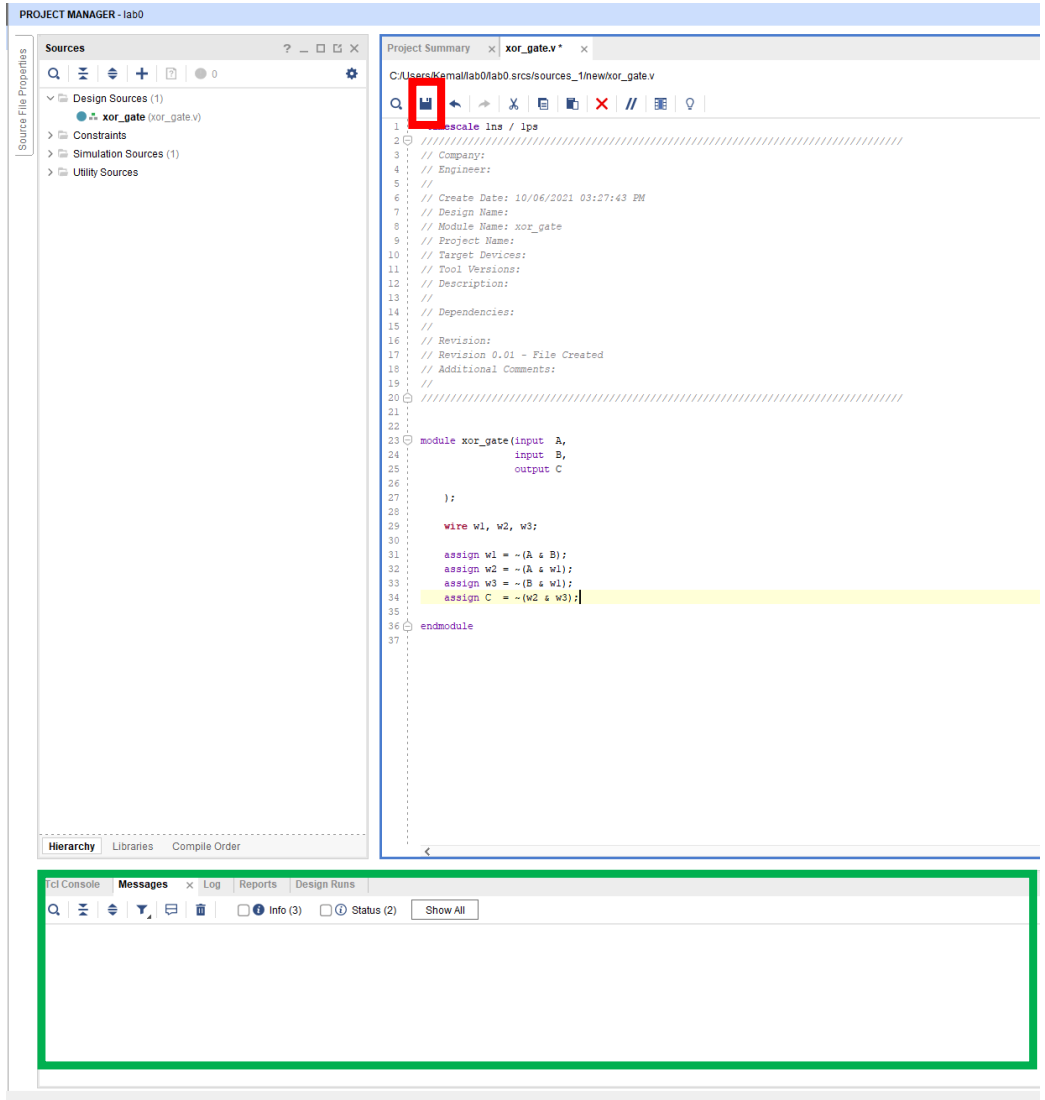
- assign is used for defining wires.
- & is AND operation
- ~ is NOT operation
- Not + AND = NAND

```
22
23  module xor_gate(input  A,
24                  input  B,
25                  output C
26
27      );
28
29      wire w1, w2, w3;
30
31      assign w1 = ~(A & B);
32      assign w2 = ~(A & w1);
33      assign w3 = ~(B & w1);
34      assign C  = ~(w2 & w3);
35
36  endmodule
37
```



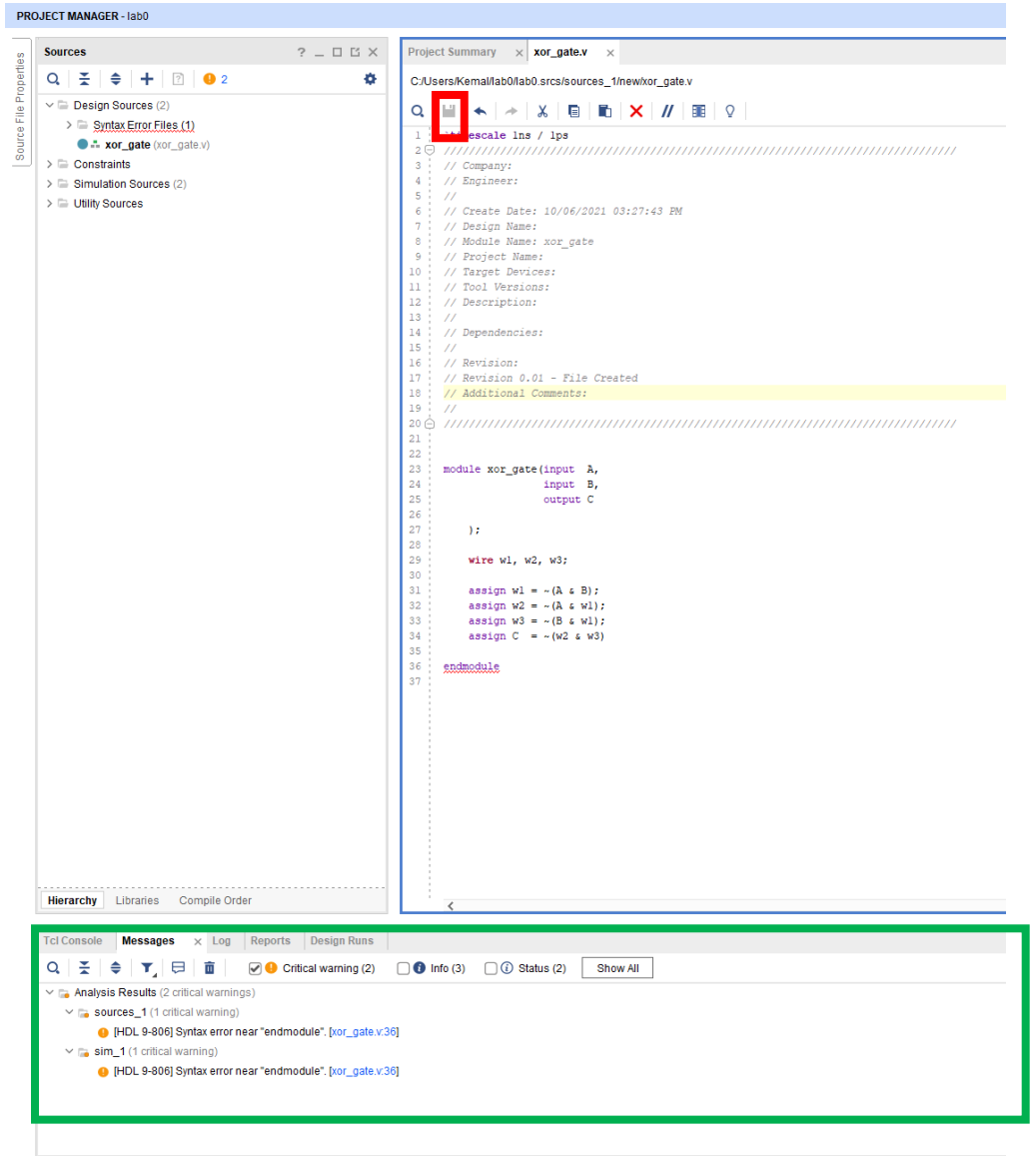
A Sample Design – XOR gate

- When you finish your design, click on Save file icon.
- If there is no error in your design, you will not see any warnings in the message section.



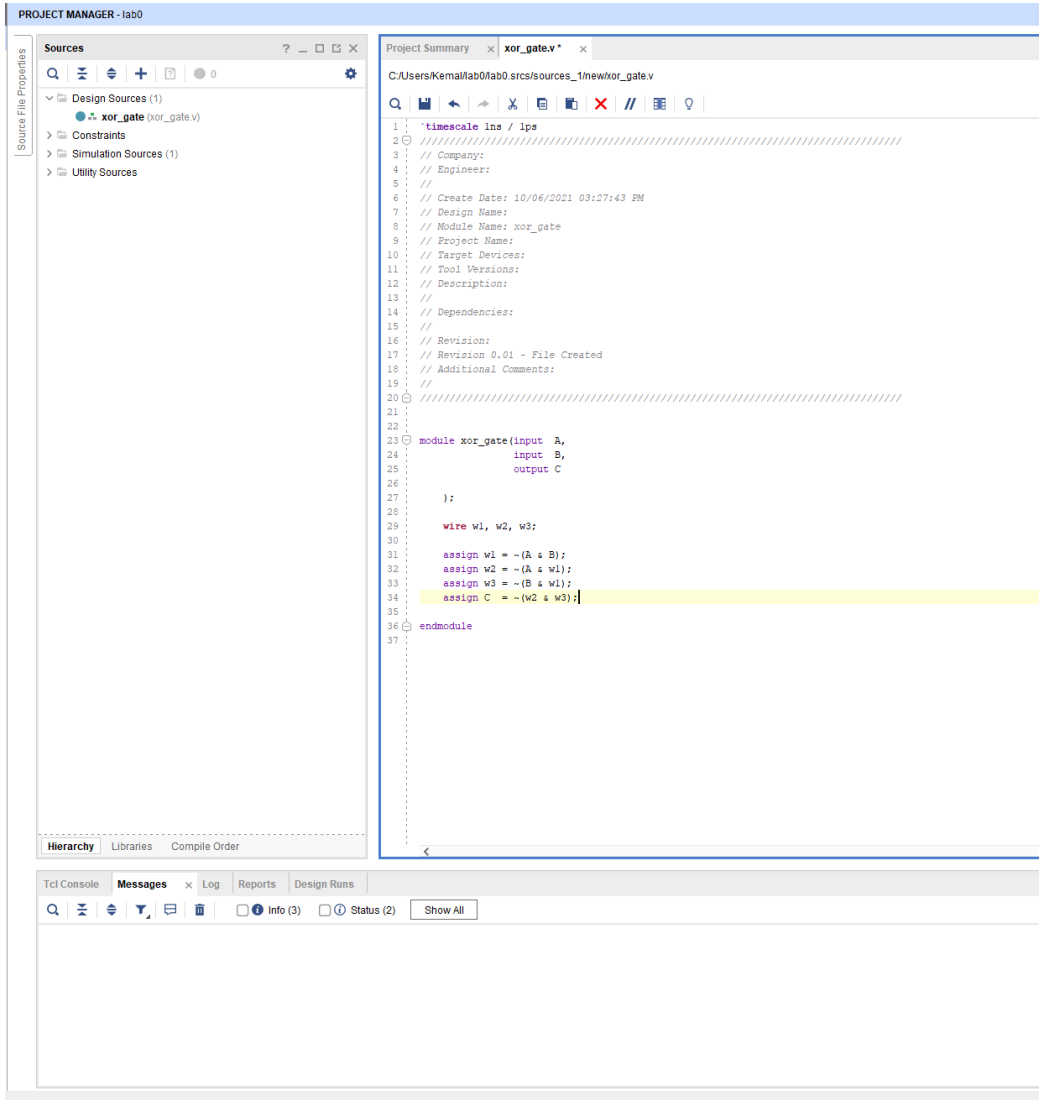
A Sample Design – XOR gate

- When you finish your design, click on Save file icon.
- If there is an error in your design, you will see warnings in the message section.



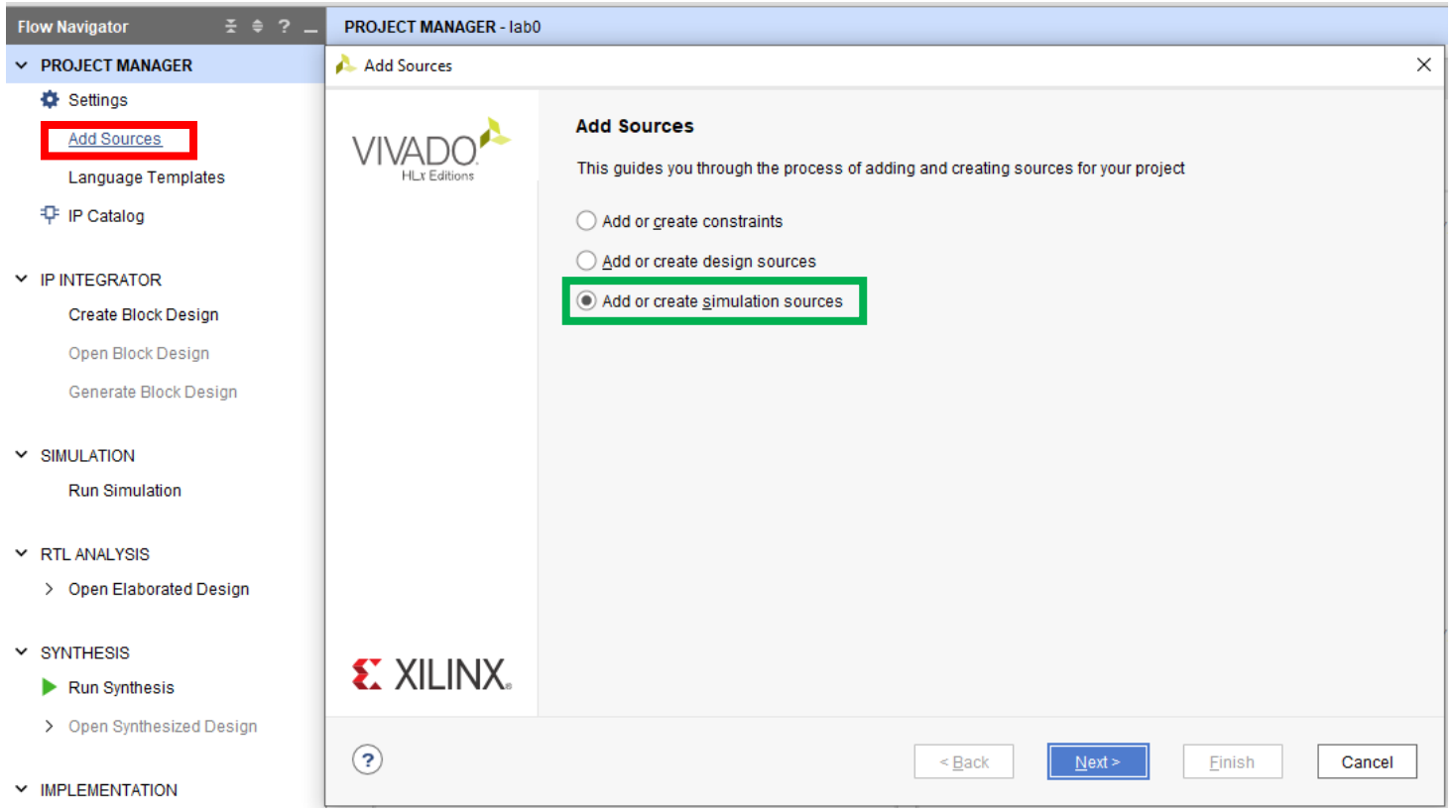
A Sample Design – XOR gate

- Design process is finished.
- You need to simulate your design.
- Do not forget to save it.



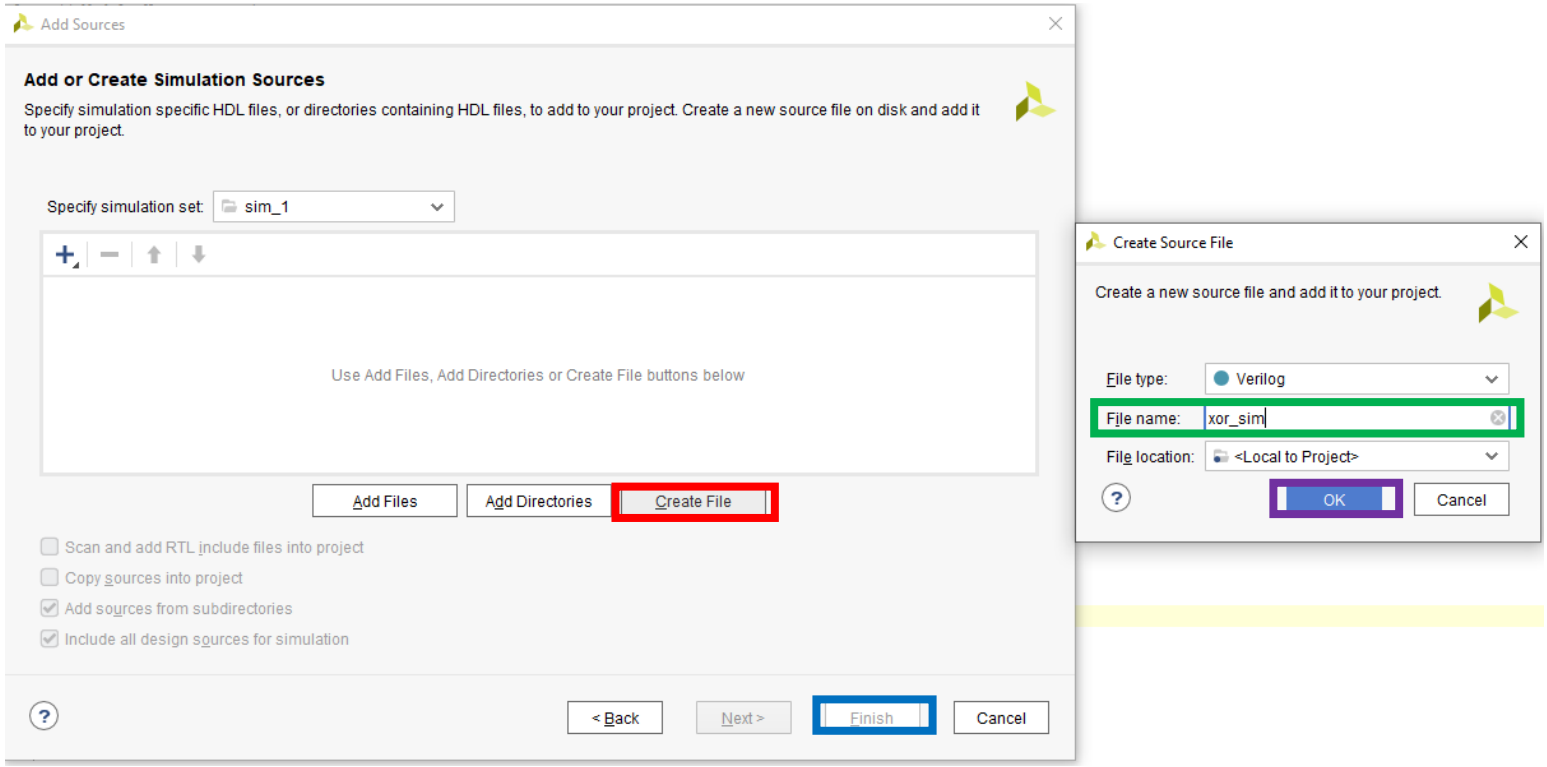
Simulation

- Click on Add Source button
- Choose Add or create simulation sources
- Click on *Next* button



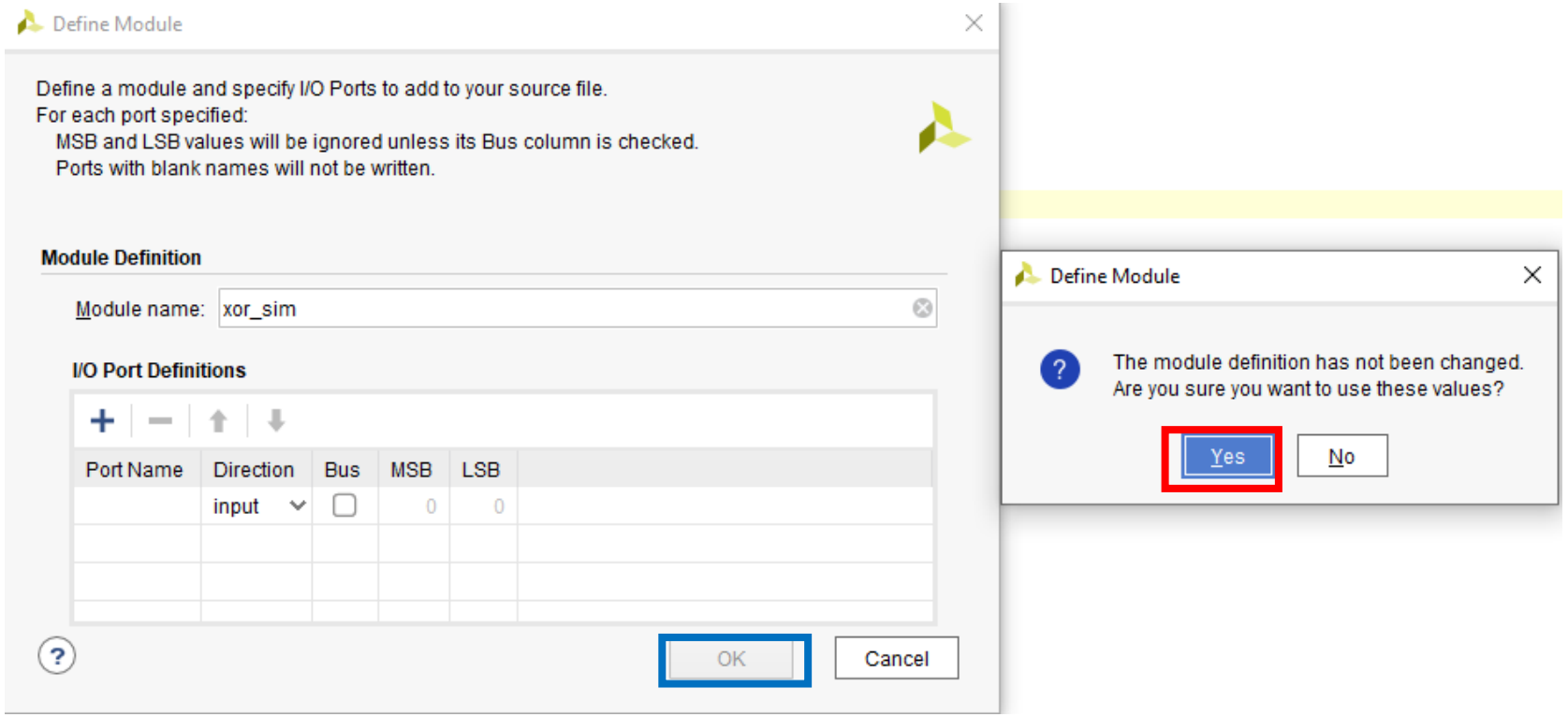
Simulation

- Click on Create File
- Give a name to source file
- Click on OK
- Click on *Finish* button



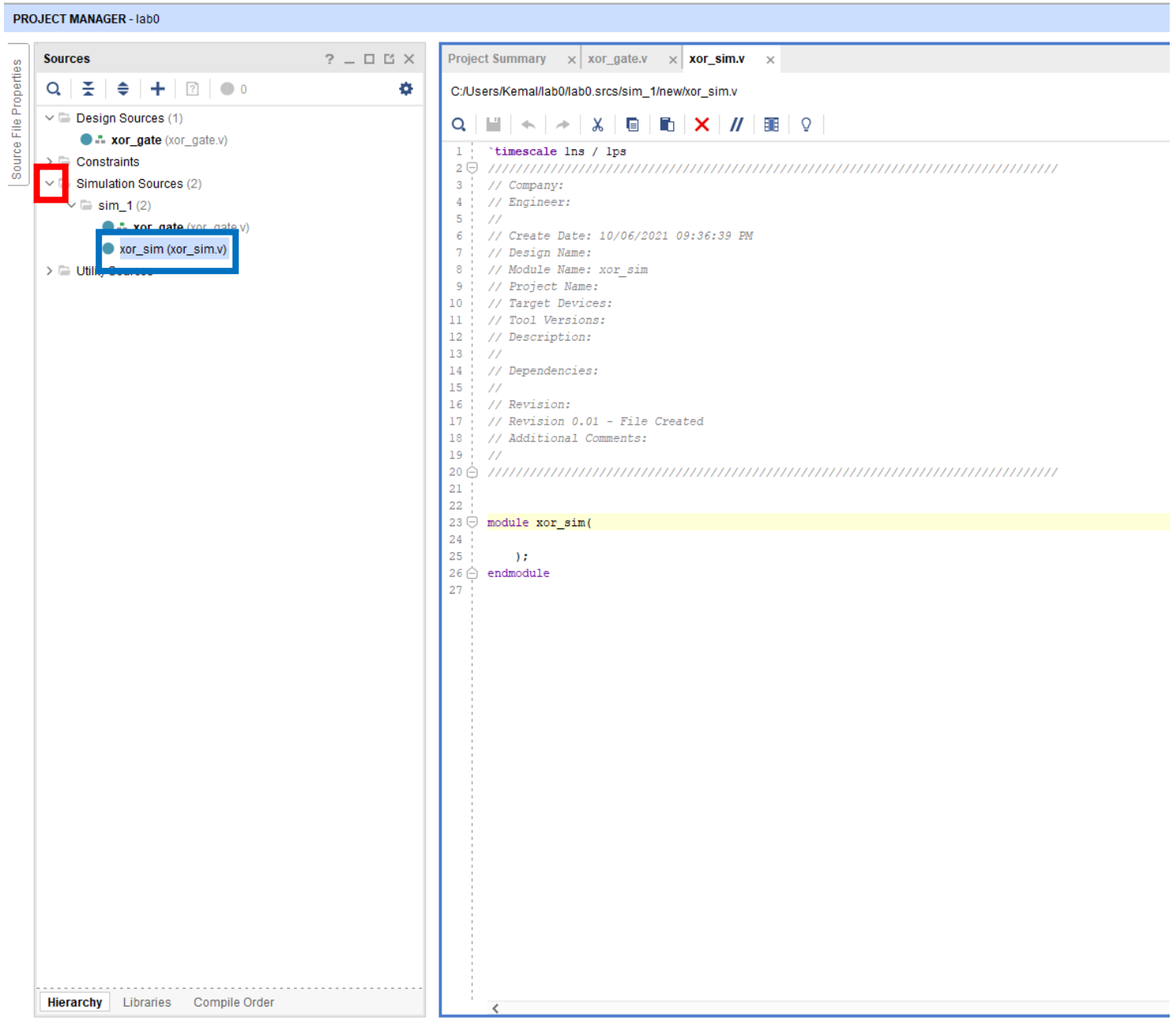
Simulation

- Click on *OK* button
- Click on Yes button



Simulation

- Click on Simulation Sources button
- Double click on *sim file*
- The editor will open it



Simulation

- Test file needs to be created manually.
- Inputs
- Outputs
- Instantiation
- Scenario

```
20  //////////////////////////////////
21
22
23  module xor_sim(
24
25      );
26
27      //Inputs
28      reg A;
29      reg B;
30
31      //Outputs
32      wire C;
33
34      //Instantiate the UUT
35      xor_gate UUT(
36      .A(A),
37      .B(B),
38      .C(C)
39      );
40
41      //Initialize Inputs
42      initial begin
43          A = 0;
44          B = 0;
45      end
46  endmodule
47
```

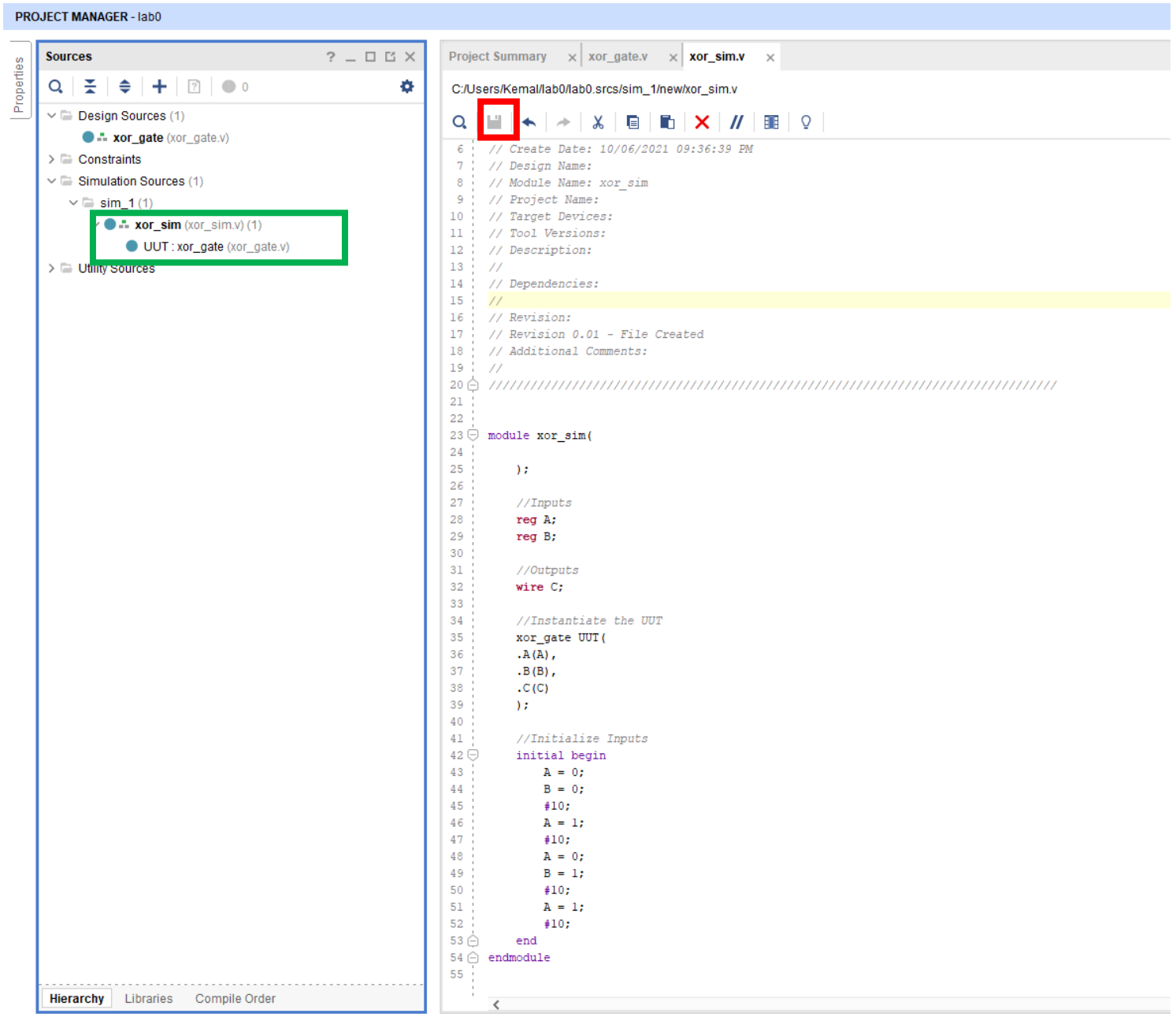
Simulation

- You will write your test scenario between initial begin and end
- A = 1; is used to apply Logic-1 to input A.
- #10; puts 10ns delay

```
21
22
23  module xor_sim(
24
25      );
26
27      //Inputs
28      reg A;
29      reg B;
30
31      //Outputs
32      wire C;
33
34      //Instantiate the UUT
35      xor_gate UUT(
36      .A(A),
37      .B(B),
38      .C(C)
39      );
40
41      //Initialize Inputs
42      initial begin
43          A = 0;
44          B = 0;
45          #10;
46          A = 1;
47          #10;
48          A = 0;
49          B = 1;
50          #10;
51          A = 1;
52          #10;
53      end
54  endmodule
55
```

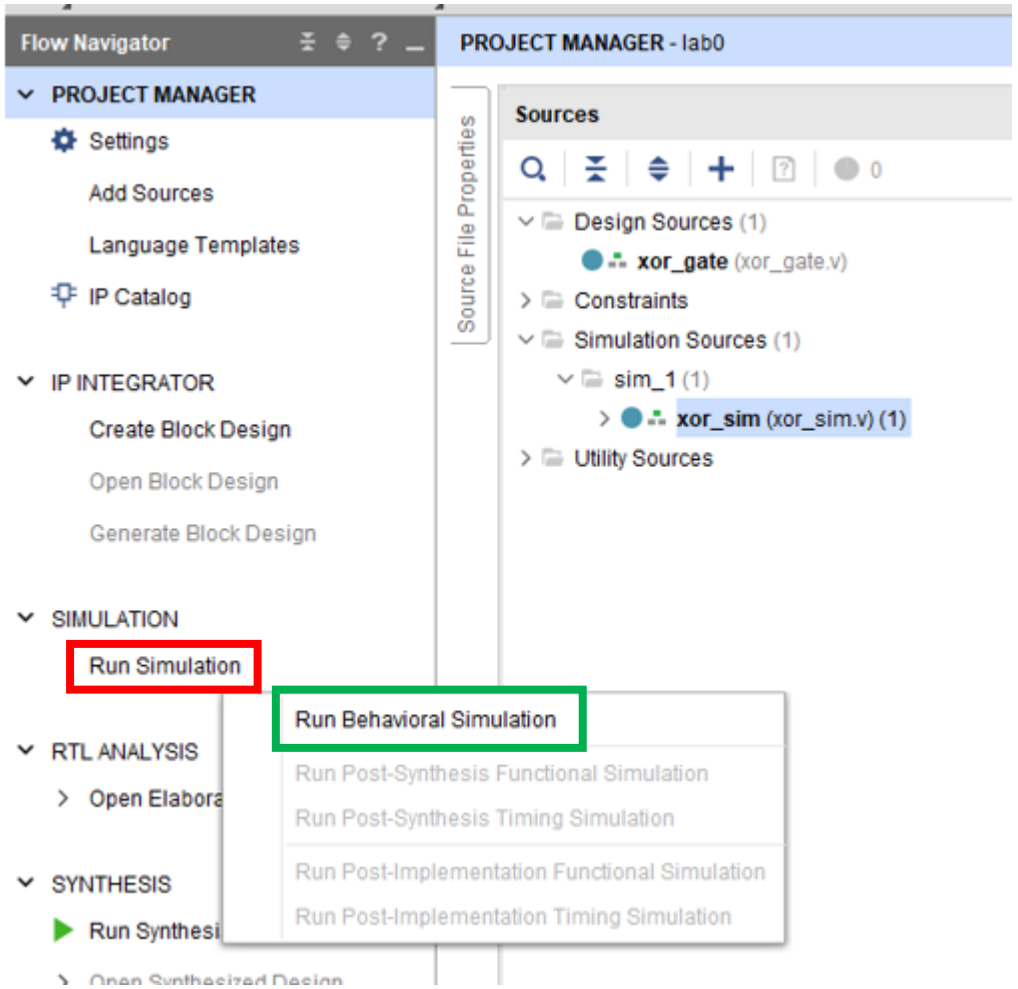
Simulation

- When you finish your design, click on Save file icon.
- When you save it, you will see that sim file is set as Top module in the simulation sources.



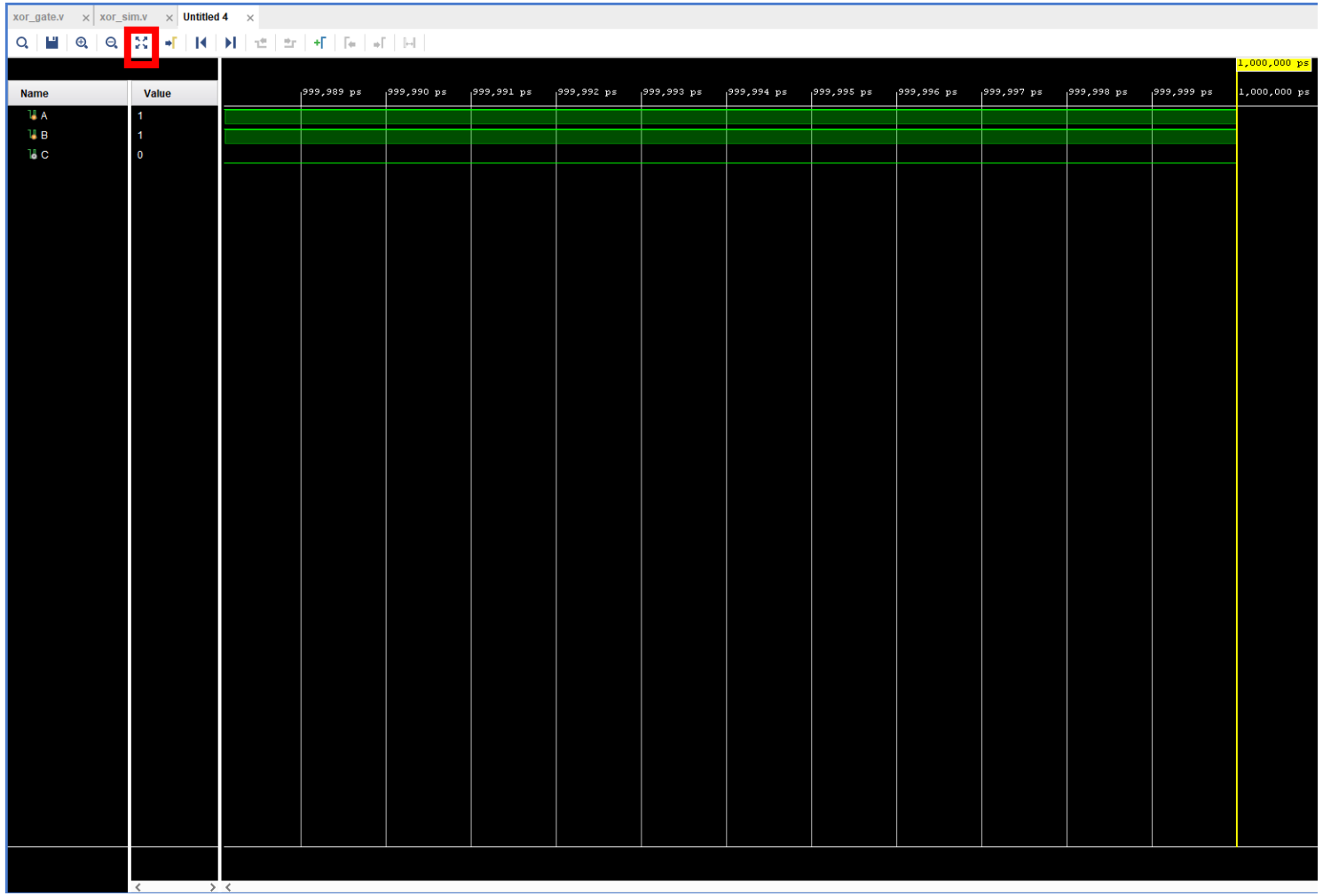
Simulation

- Click on Run Simulation
- Click on Run Behavioral Simulation



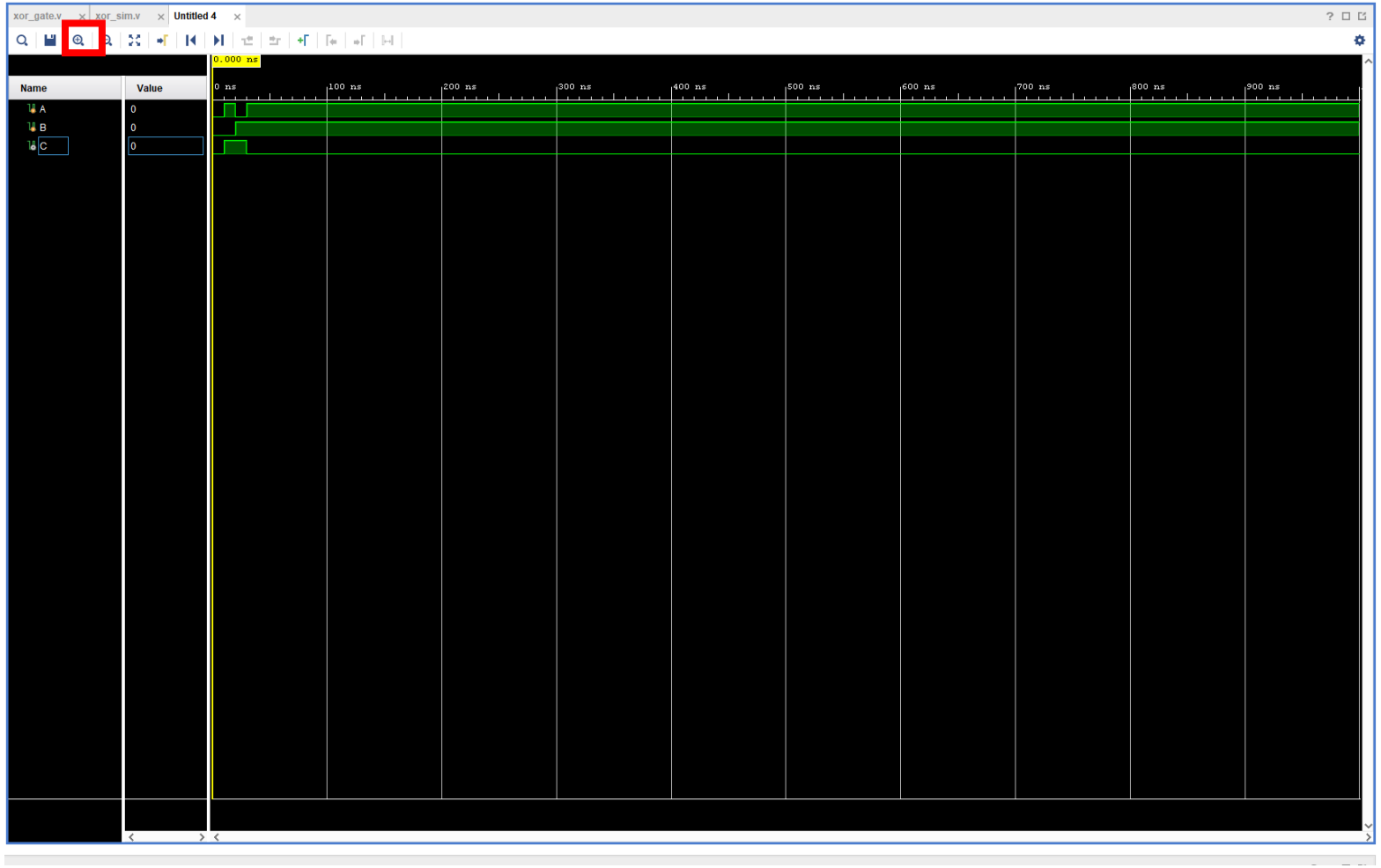
Simulation

- Simulation shows the end of the simulation time by default.
- You can see all run time by Zoom Fit



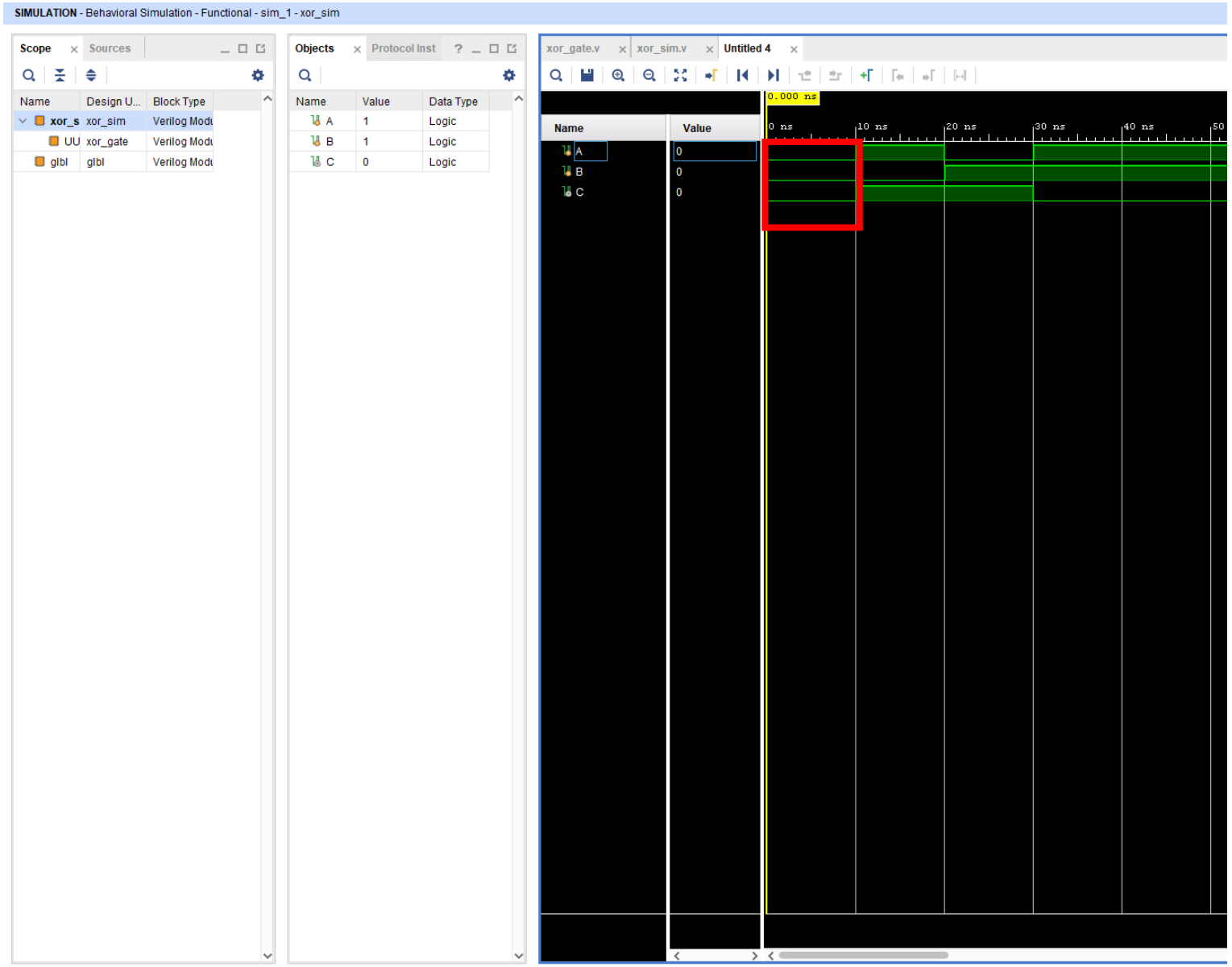
Simulation

- You can bring yellow cursor to the start of the simulation by clicking on the waveform.
- You can use Zoom In button to enlarge the area around the yellow cursor.



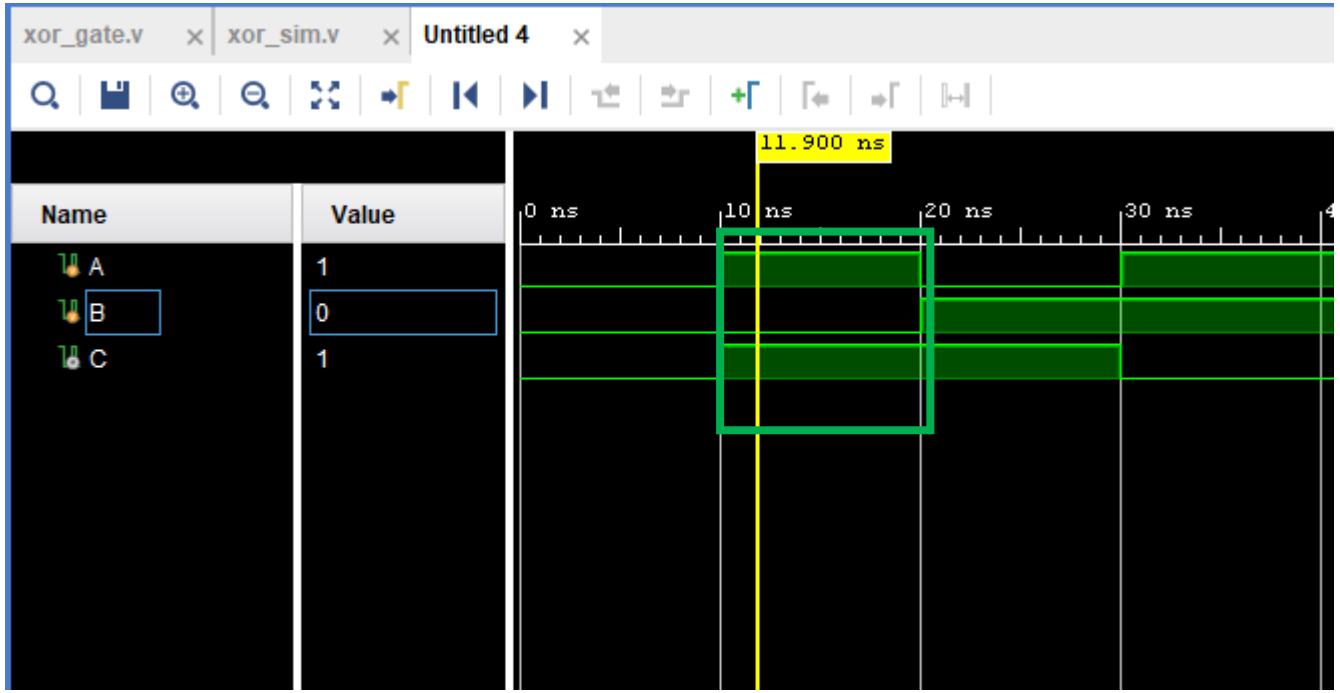
Simulation

- ⊕ is XOR operation.
- A ⊕ B = C
- 0 ⊕ 0 = 0



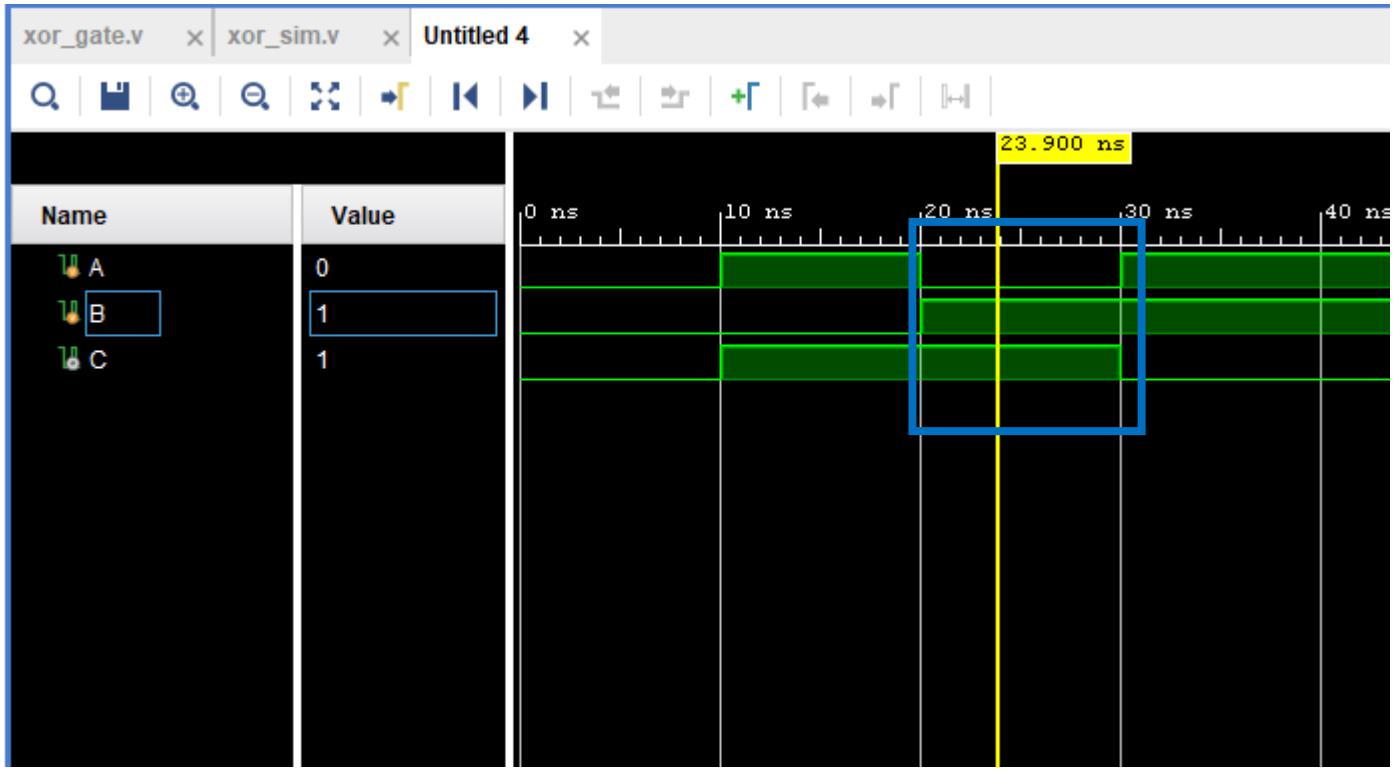
Simulation

- ⊕ is XOR operation.
- A ⊕ B = C
- 0 ⊕ 0 = 0
- 1 ⊕ 0 = 1



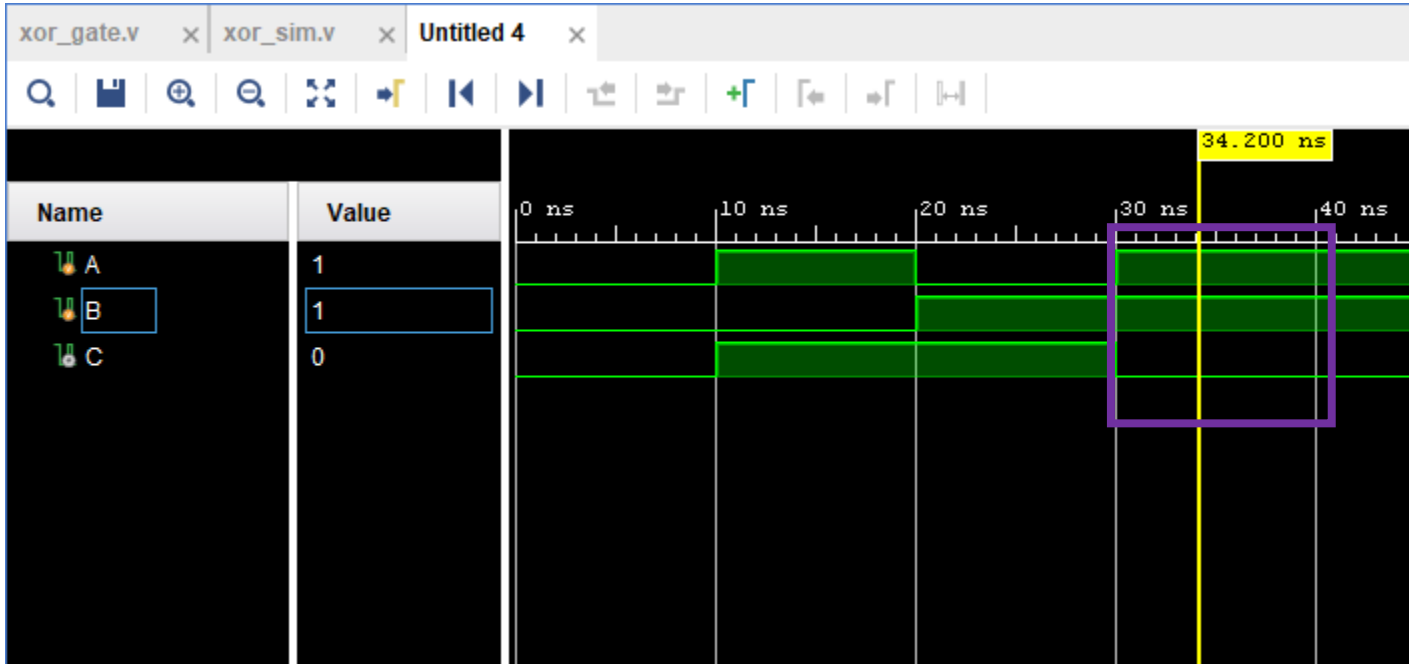
Simulation

- ⊕ is XOR operation.
- A ⊕ B = C
- 0 ⊕ 0 = 0
- 1 ⊕ 0 = 1
- 0 ⊕ 1 = 1



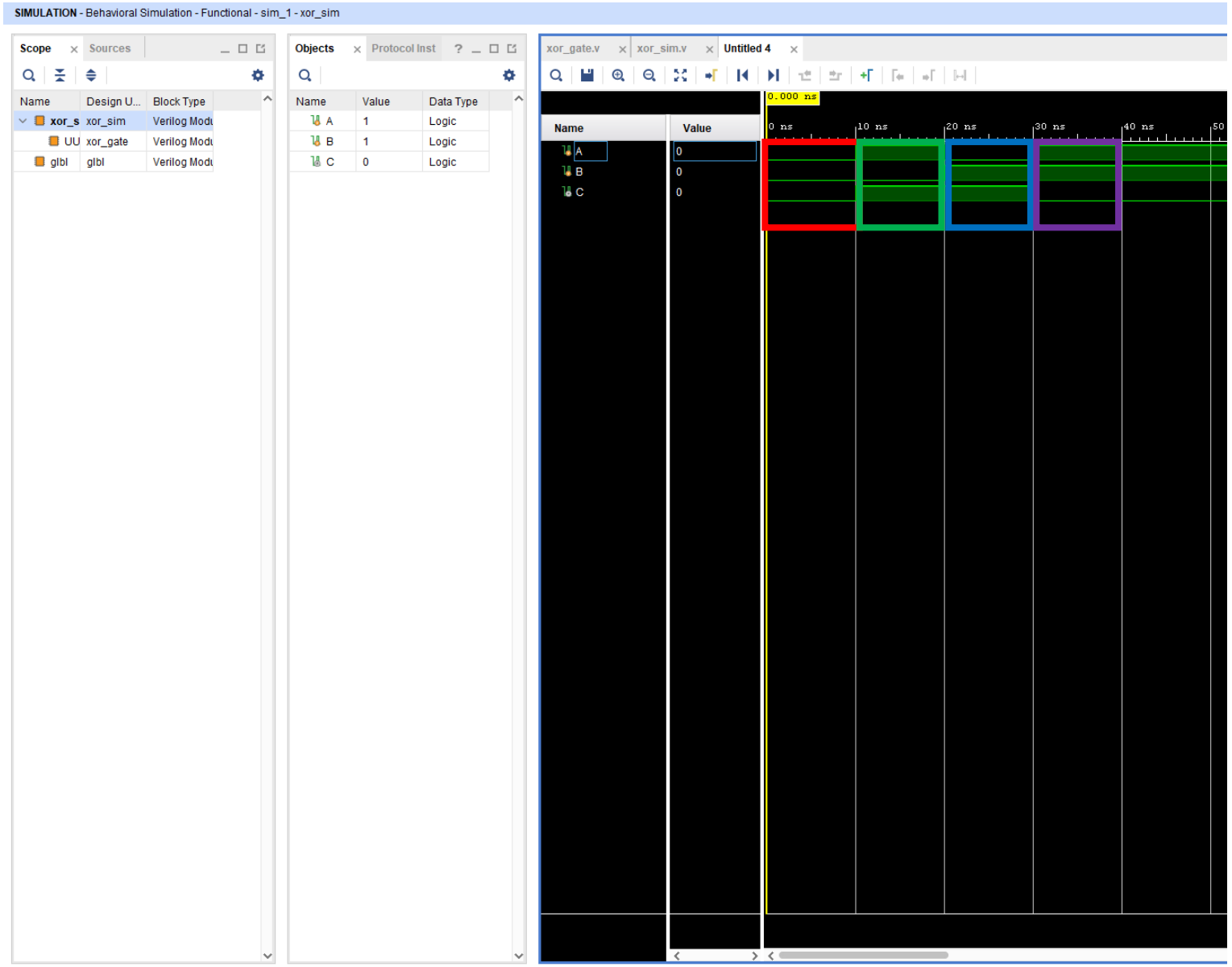
Simulation

- ⊕ is XOR operation.
- A ⊕ B = C
- 0 ⊕ 0 = 0
- 1 ⊕ 0 = 1
- 0 ⊕ 1 = 1
- 1 ⊕ 1 = 0



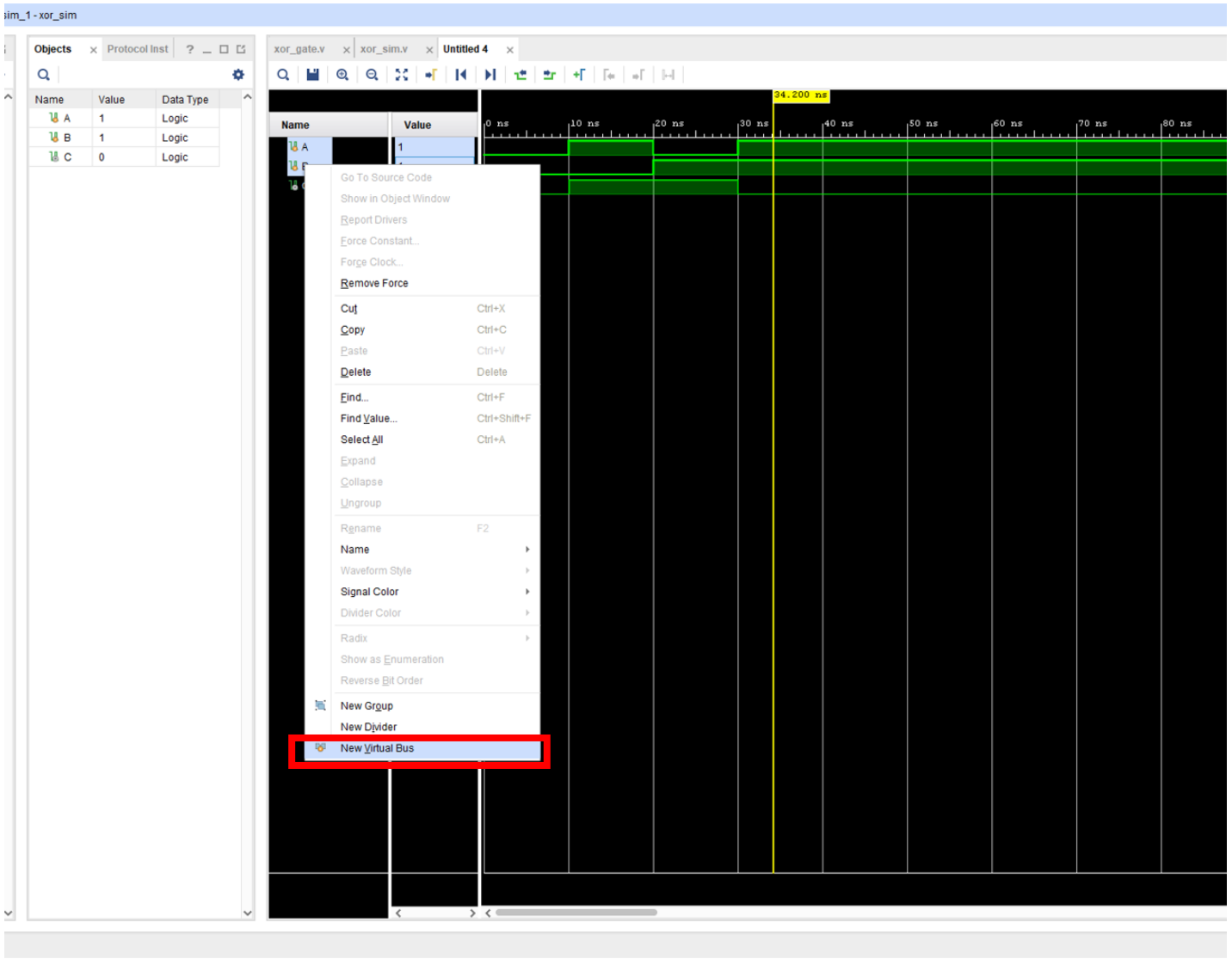
Simulation

- ⊕ is XOR operation.
- A ⊕ B = C
- 0 ⊕ 0 = 0
- 1 ⊕ 0 = 1
- 0 ⊕ 1 = 1
- 1 ⊕ 1 = 0



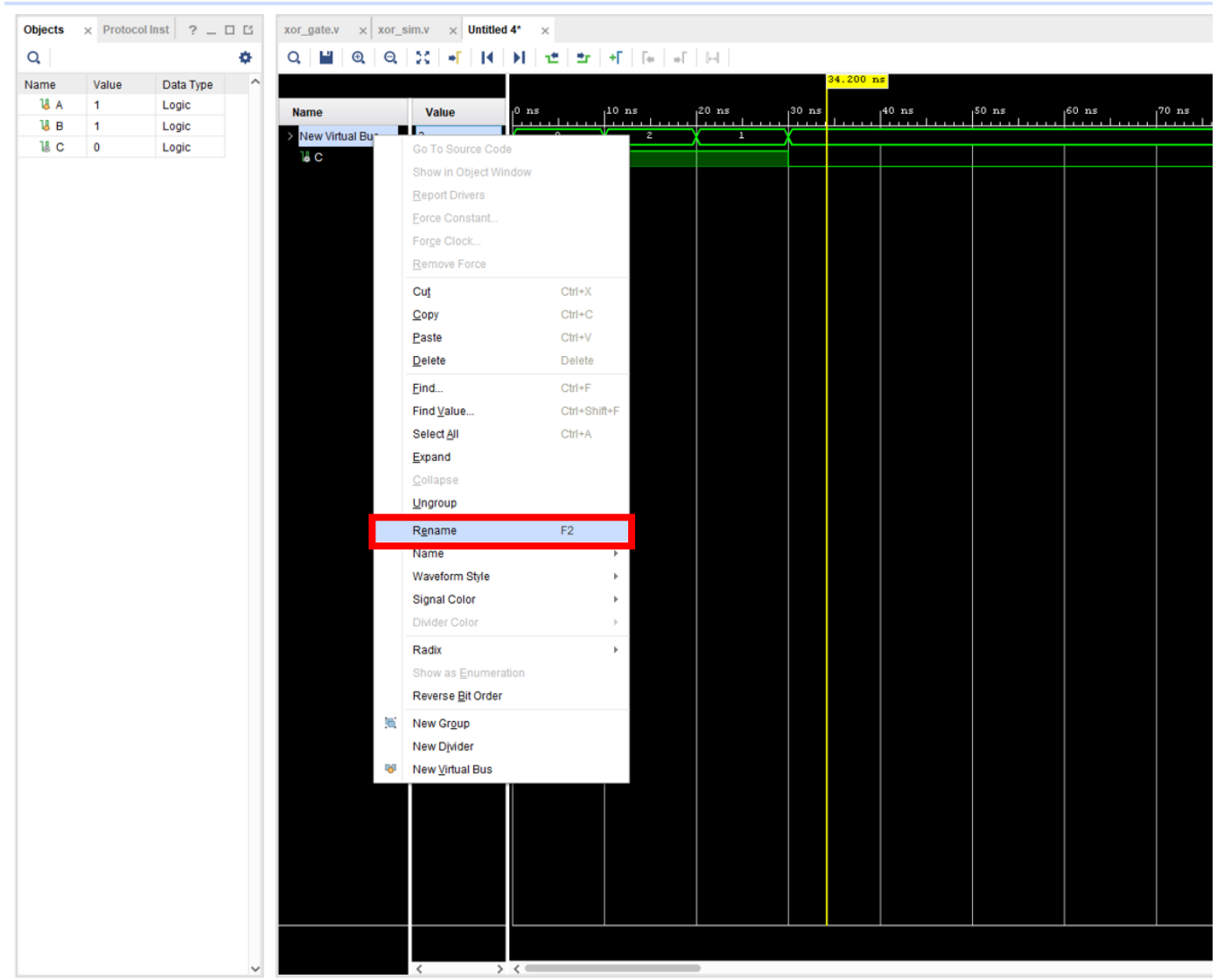
Simulation

- For better visualization, you may group inputs (and/or outputs) via New Virtual Bus option



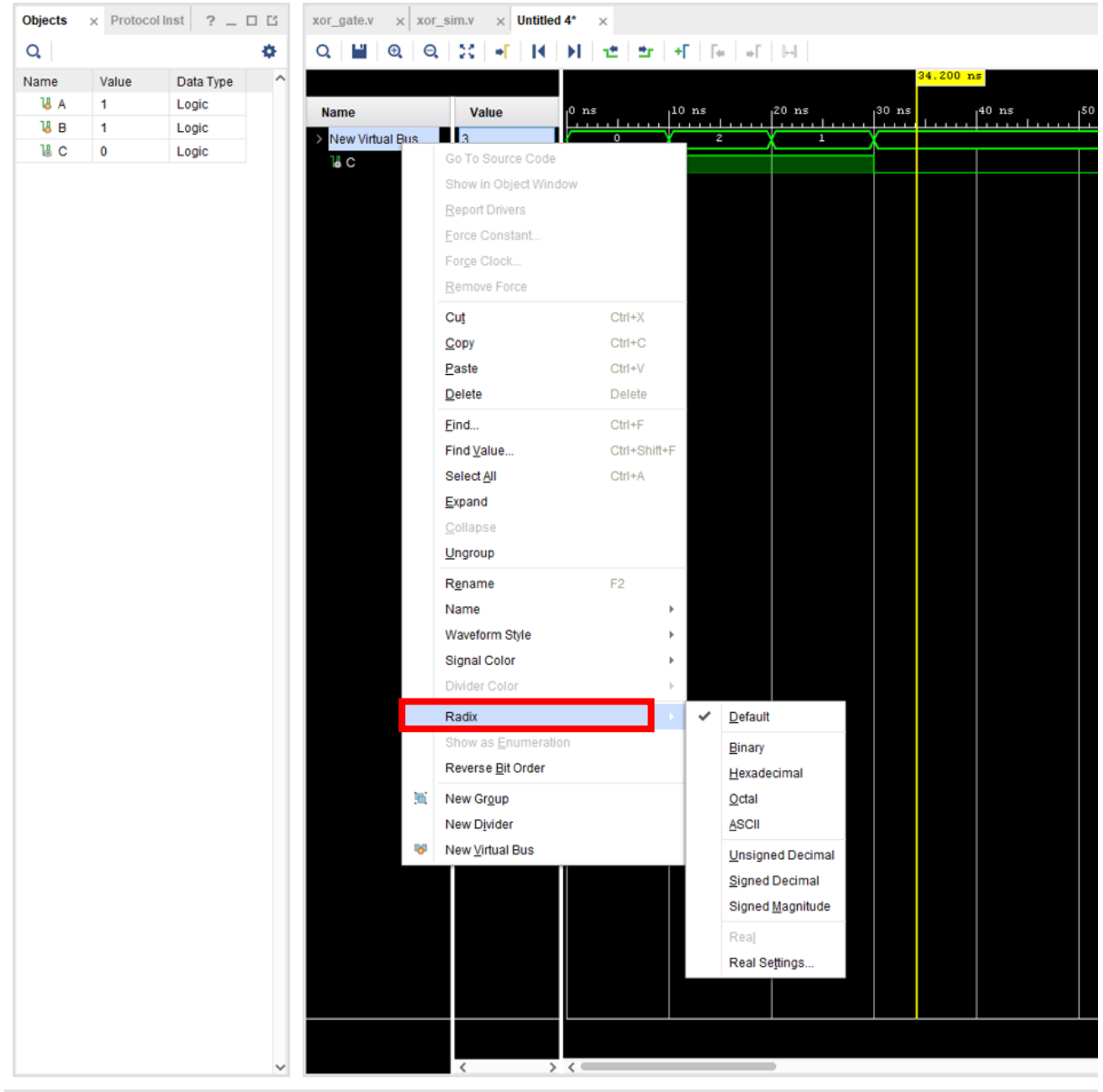
Simulation

- You may Rename your virtual bus



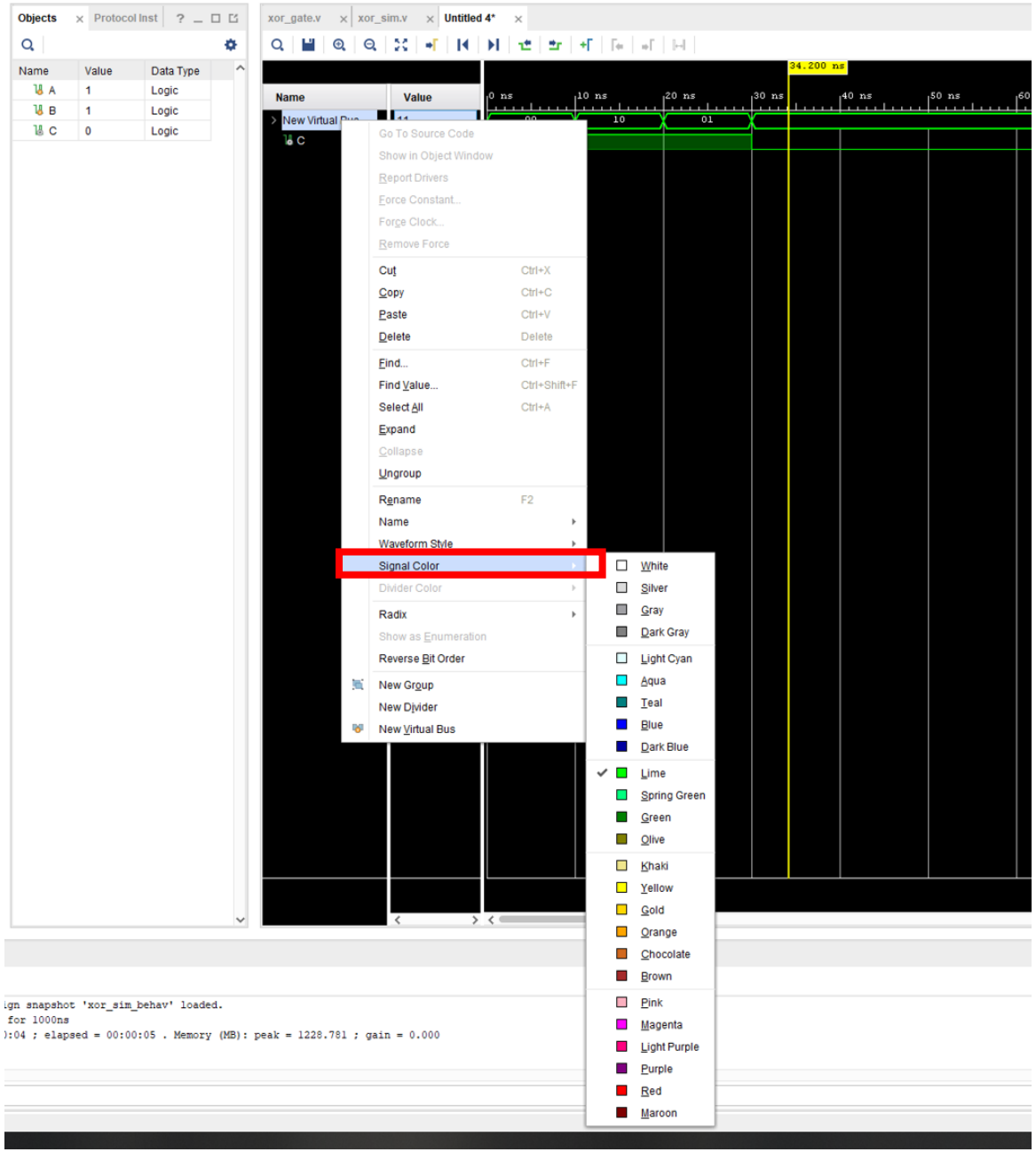
Simulation

- You may change Radix of virtual bus



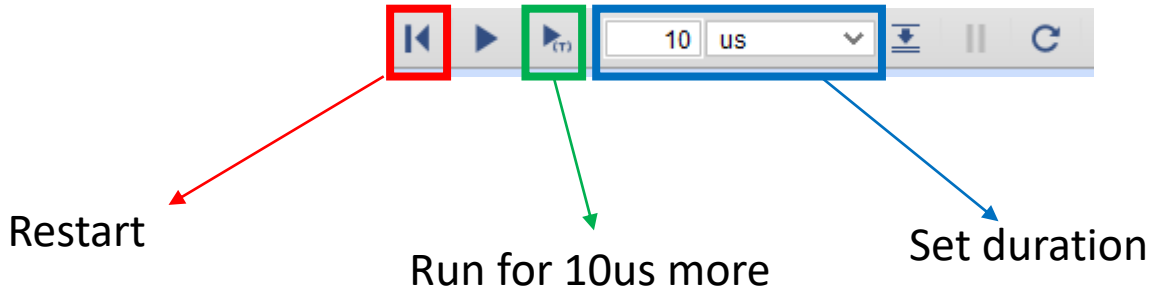
Simulation

- You may change Signal Color for better visualization



Simulation

- You can also change the duration of the corresponding simulation.



Simulation

- After you adjust the waves, you may save the waveform to use later.
- Save Waveform Configuration

Simulation

- Give a name to wave configuration
- Save it