

# 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.



## Quick Start

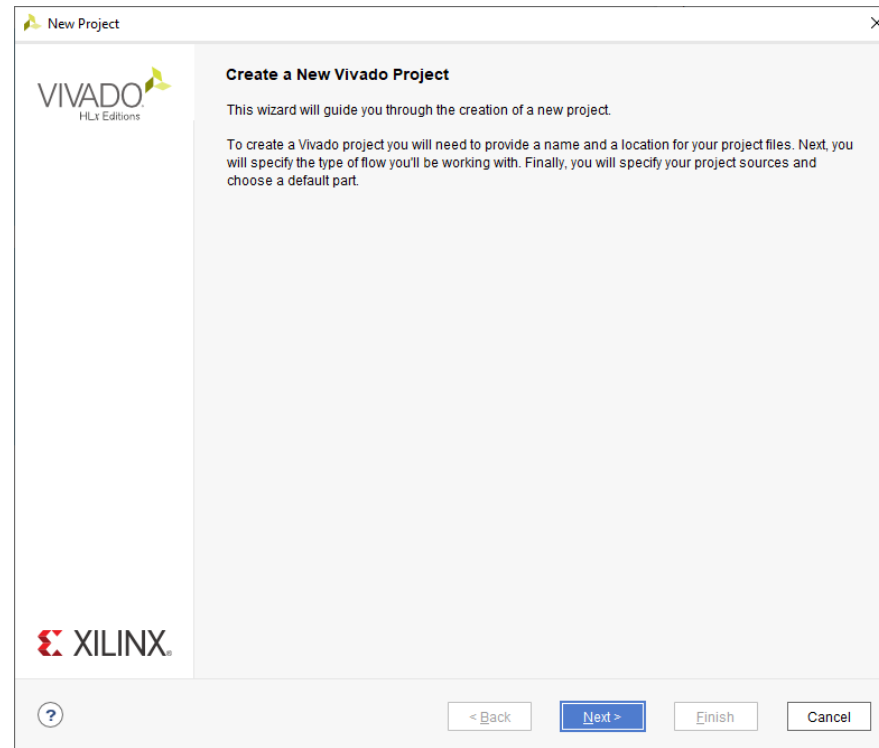
Create Project >

Open Project >

Open Example Project >

# Creating a New Project

- Click on *Next* button.



# Creating a New Project

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

New Project

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

Project name: lab0

Project location: C:/Users/Kemal

☒ Create project subdirectory

Project will be created at: C:/Users/Kemal/lab0

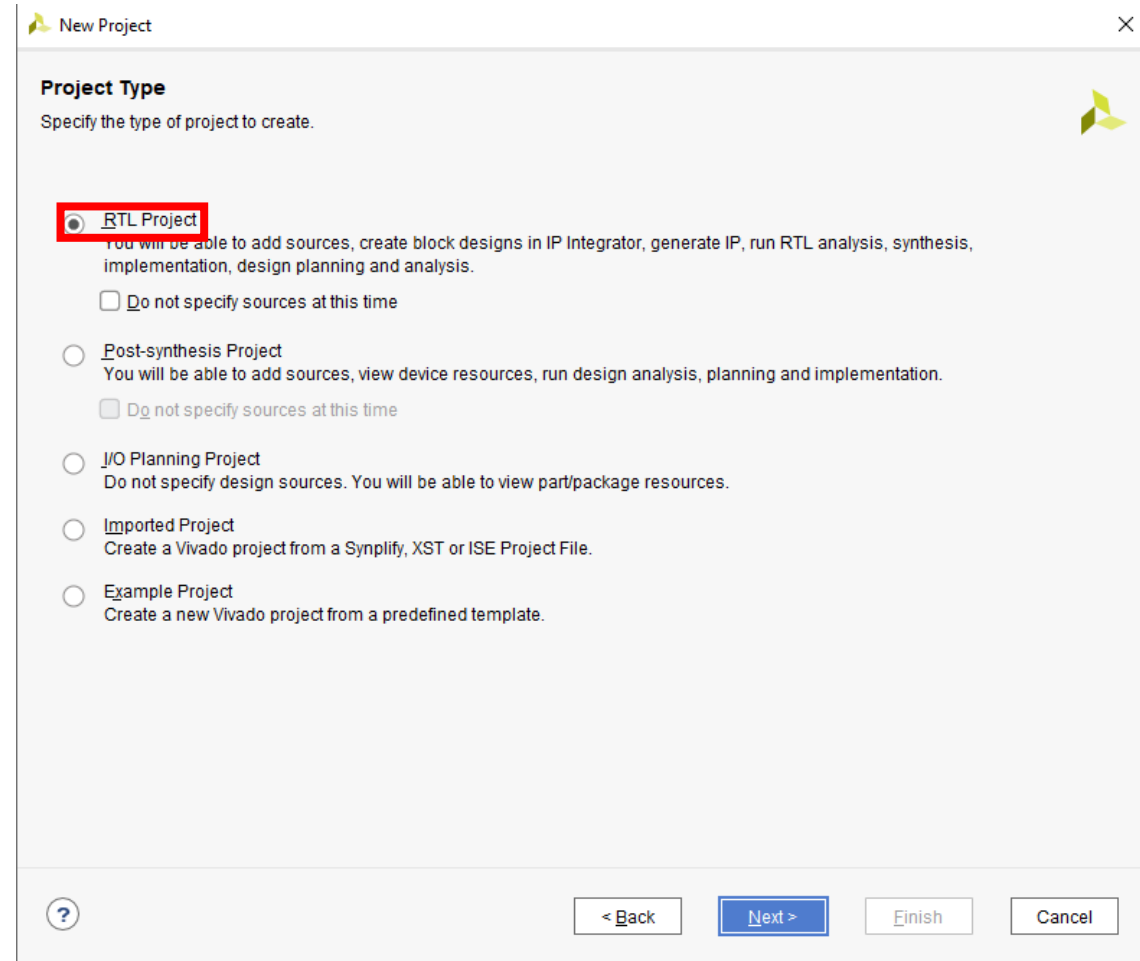
? < Back Next > Finish Cancel

# 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

New Project

**Add Sources**

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

Use Add Files, Add Directories or Create File buttons below

Add Files Add Directories Create File

☐ Scan and add RTL include files into project

☐ Copy sources into project

☒ Add sources from subdirectories

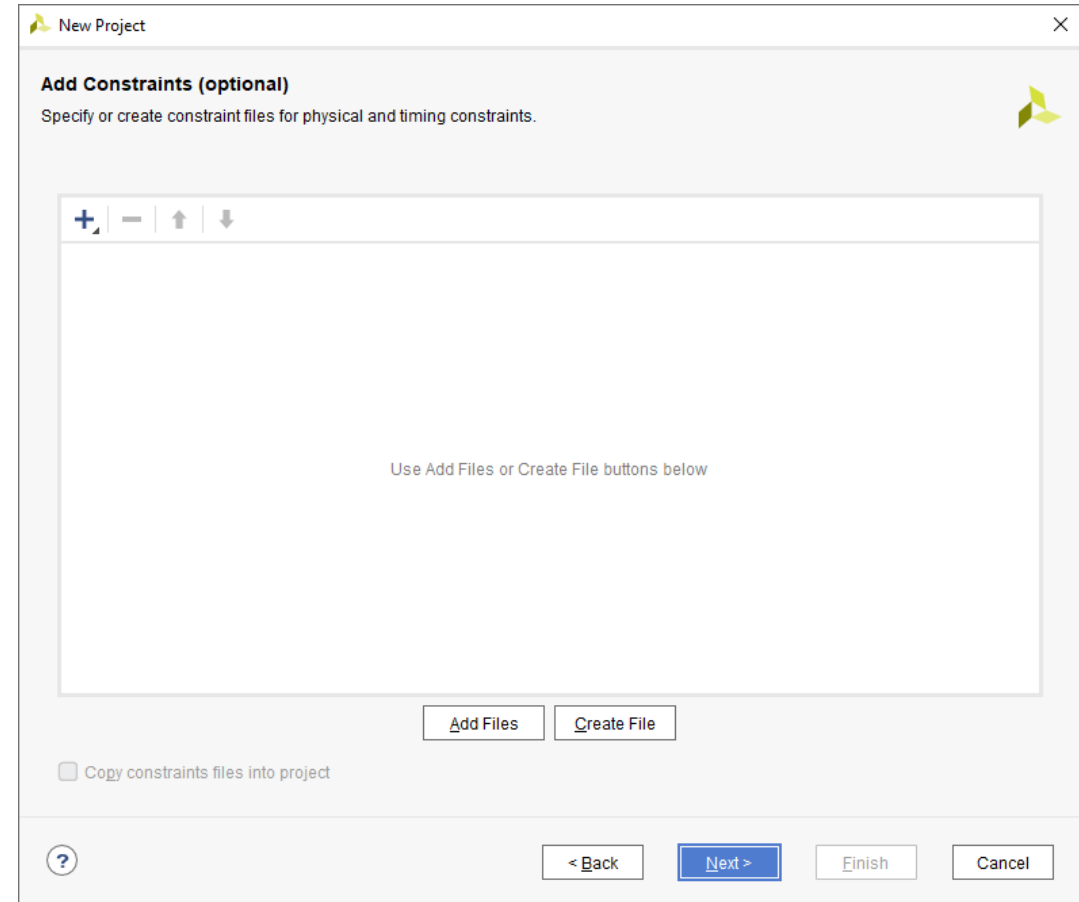
Target language: Verilog Simulator language: Mixed

< Back Next > Finish Cancel



# 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

New Project

**Default Part**  
Choose a default Xilinx part or board for your project.

Parts | Boards

[Reset All Filters](#)

Category: All Package: csg324 Temperature: All Remaining  
Family: Artix-7 Speed: -1 Static power: All Remaining

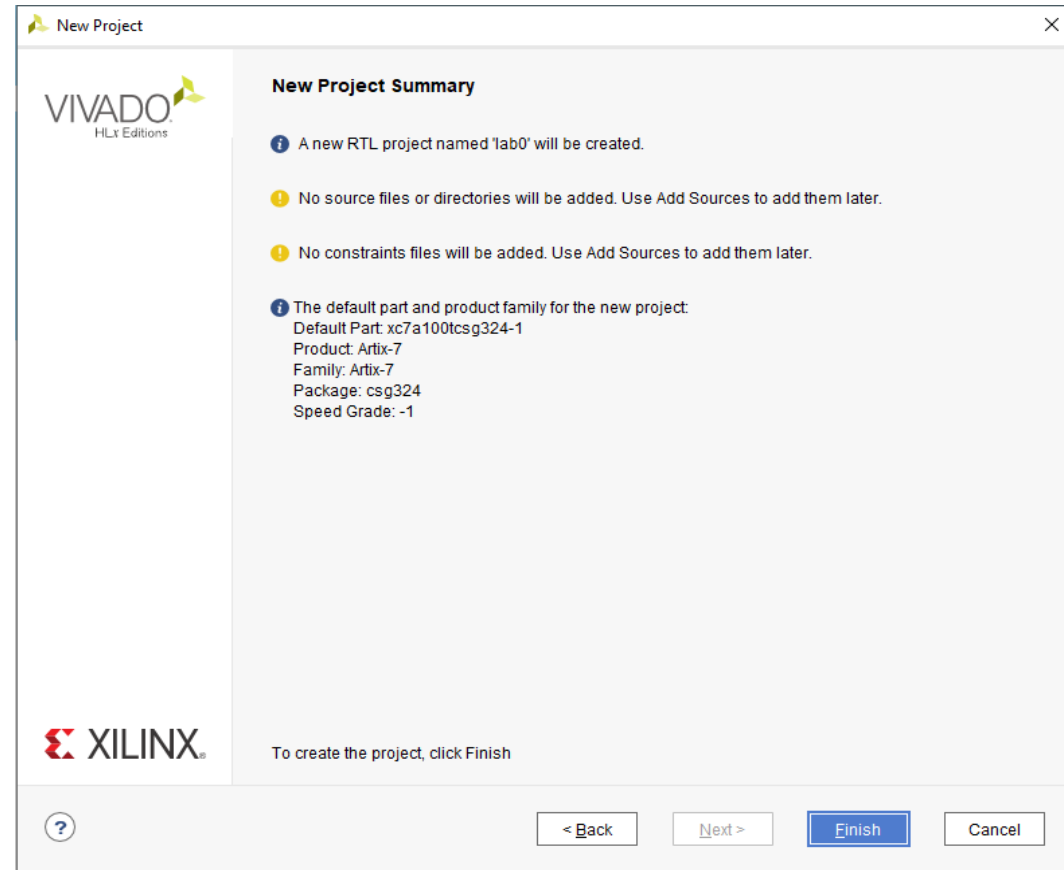
Search: Q-

Part	I/O Pin Count	Available IOBs	LUT Elements	FlipFlops	Block RAMs	Ultra RAMs	DSPs	Gt
xc7a15tcsg324-1	324	210	10400	20800	25	0	45	0
xc7a35tcsg324-1	324	210	20800	41600	50	0	90	0
xc7a50tcsg324-1	324	210	32600	65200	75	0	120	0
xc7a75tcsg324-1	324	210	47200	94400	105	0	180	0
xc7a100tcsg324-1	324	210	63400	126800	135	0	240	0

< Back Next > Finish Cancel

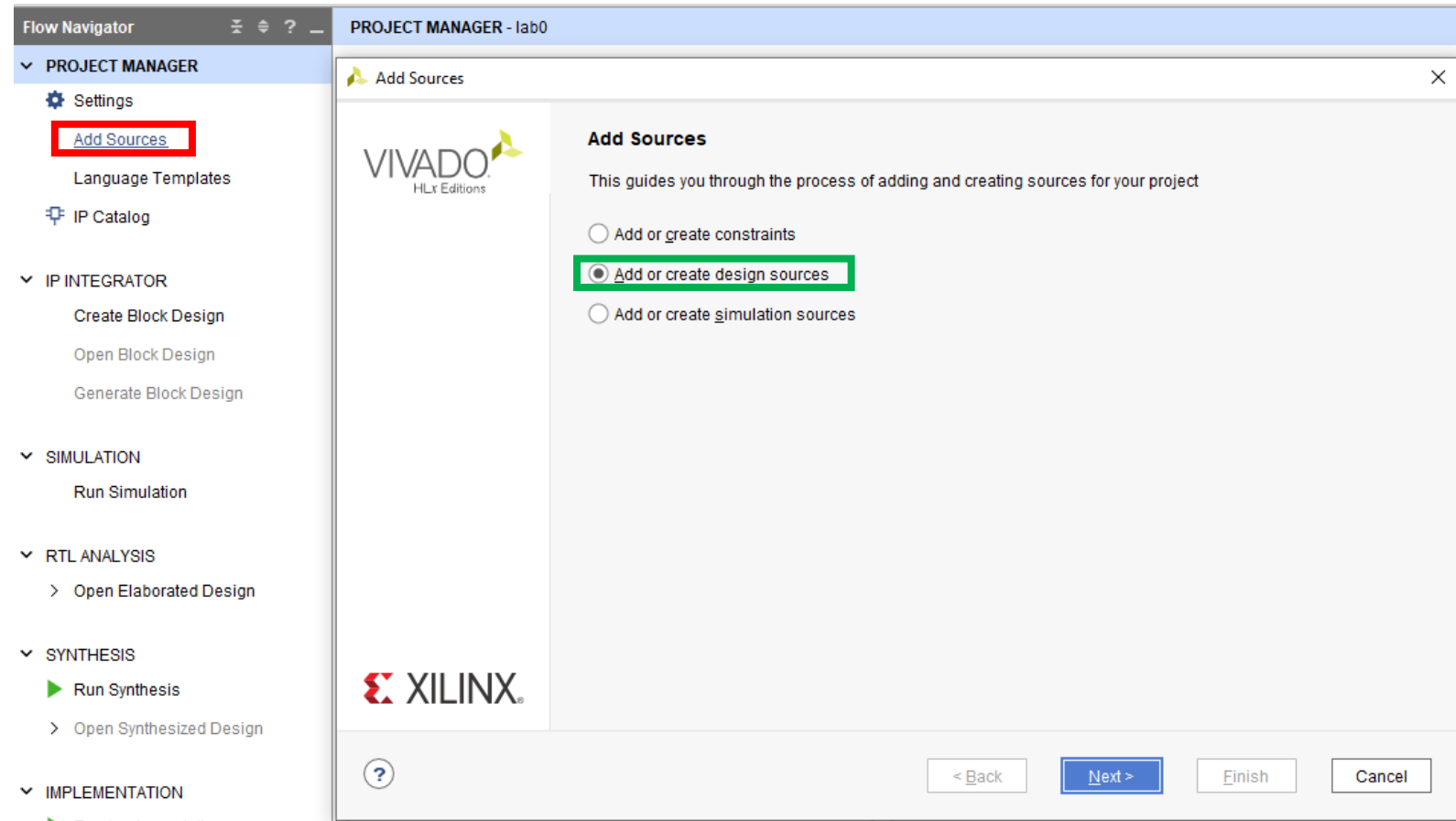
# Creating a New Project

- Click on *Finish* button



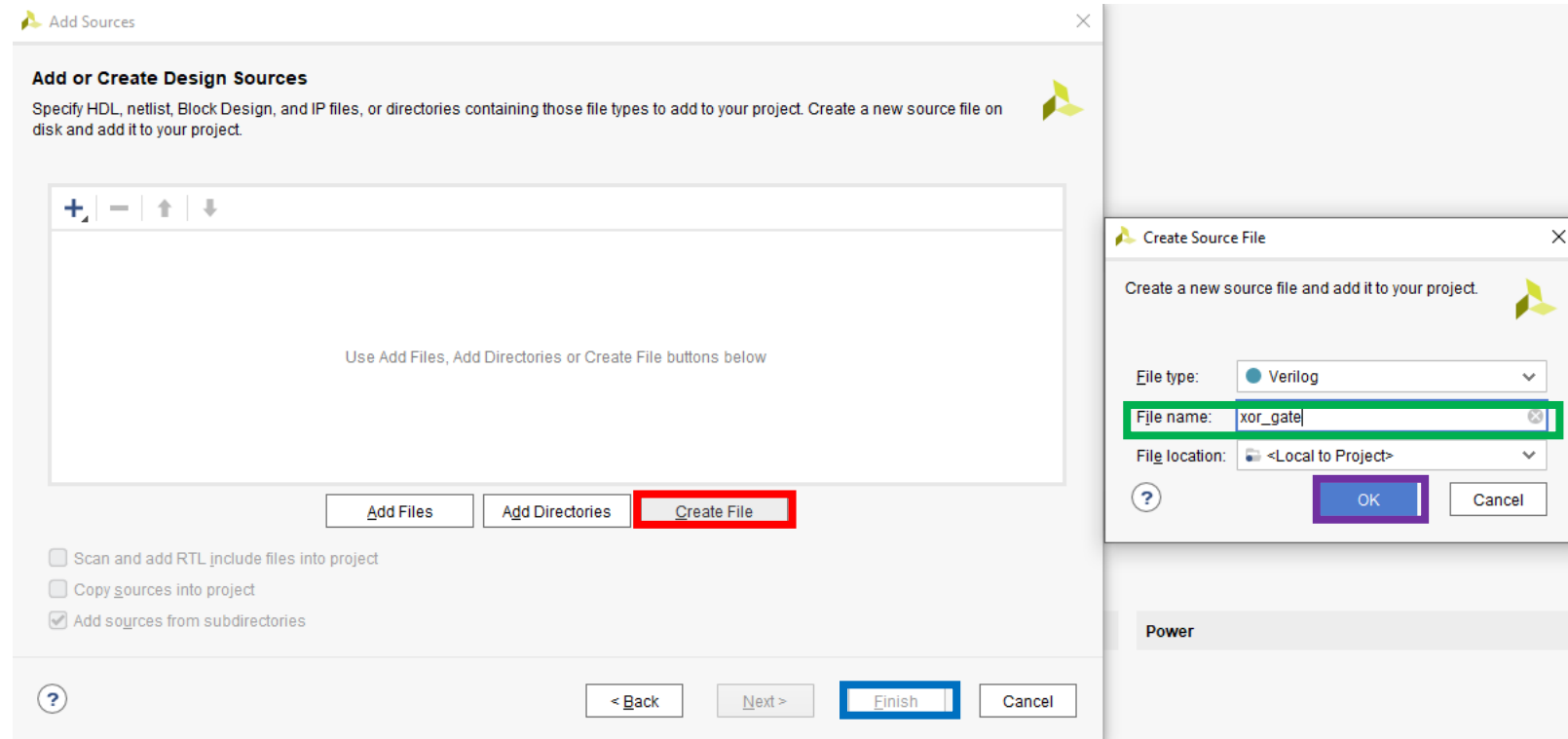
# Creating a Source File

- 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

Define a module and specify I/O Ports to add to your source file.  
For each port specified:  
MSB and LSB values will be ignored unless its Bus column is checked.  
Ports with blank names will not be written.

**Module Definition**

Module name: xor\_gate

**I/O Port Definitions**

Port Name	Direction	Bus	MSB	LSB
	input	<input type="checkbox"/>	0	0

OK Cancel

Incremental implementation: None

**Timing**

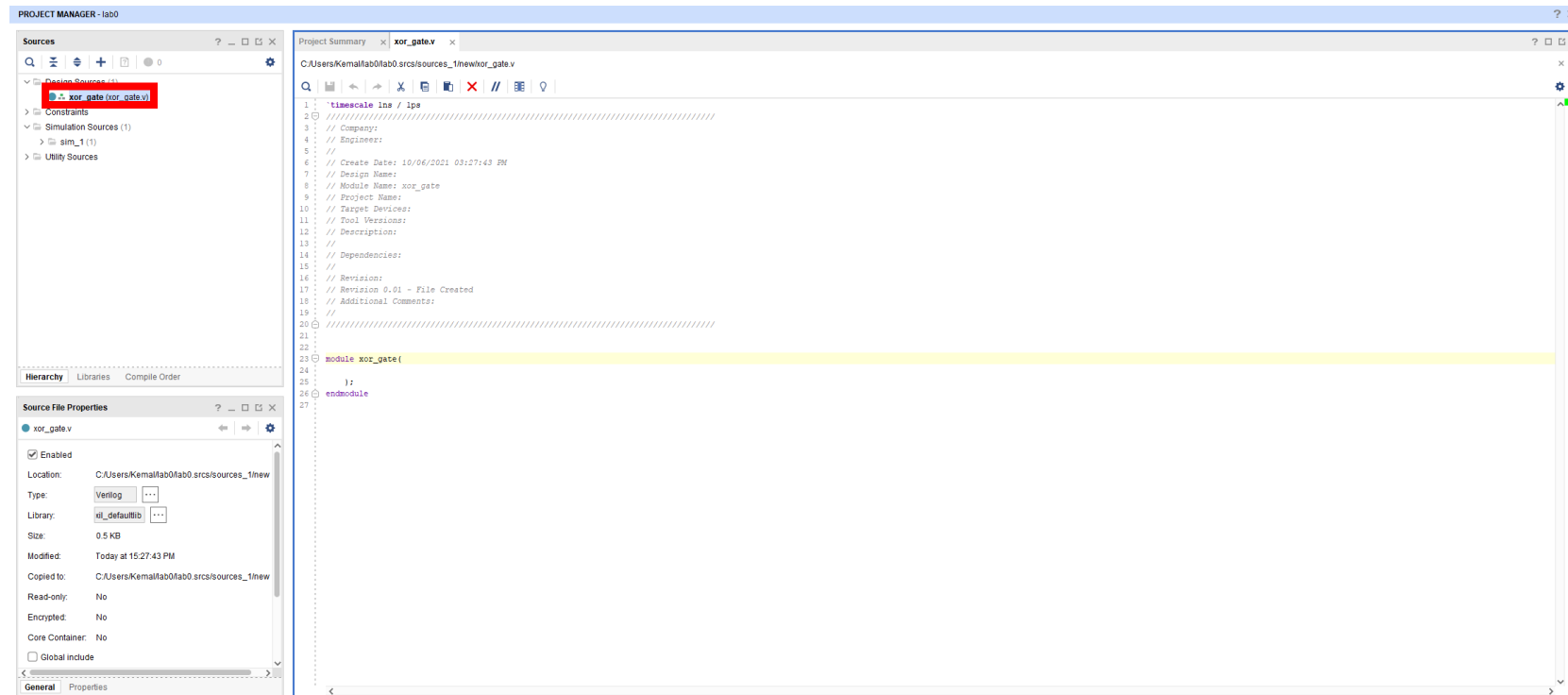
Define Module

? The module definition has not been changed.  
Are you sure you want to use these values?

Yes No

# Making a Sample Design

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



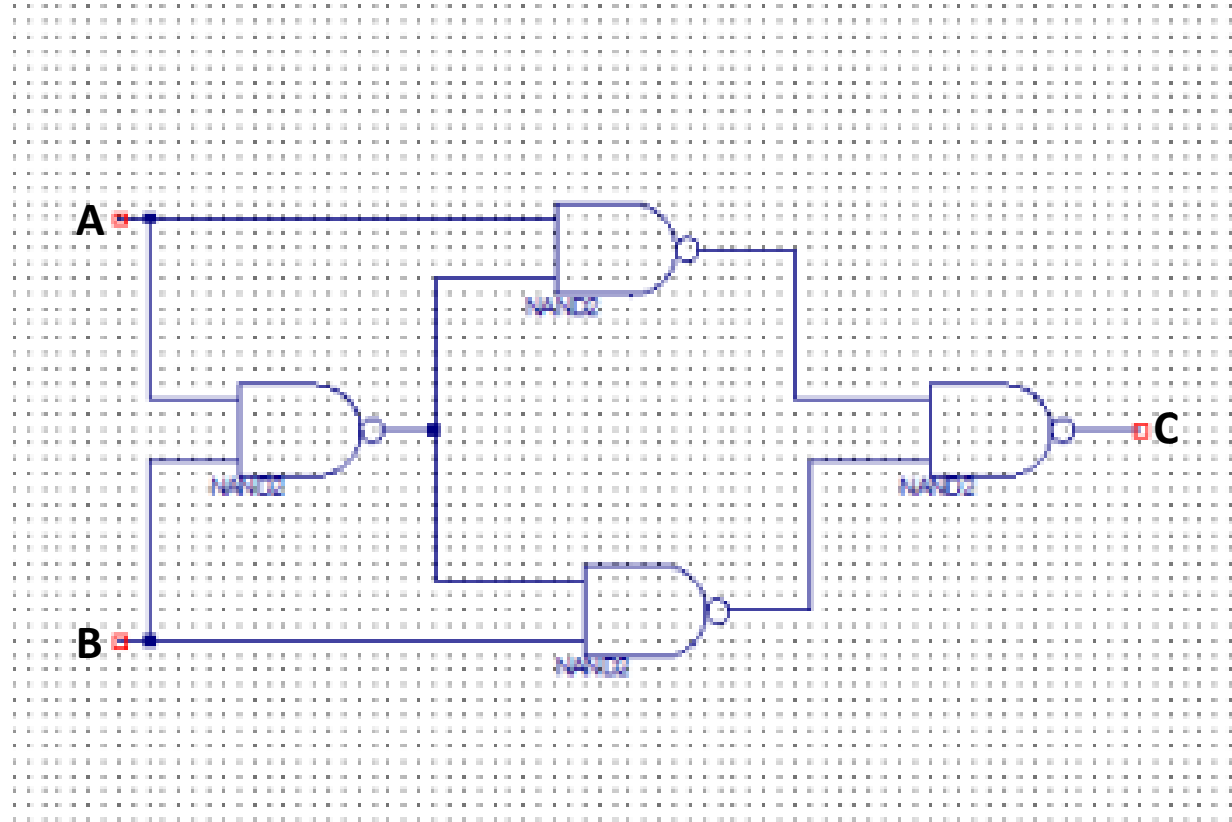
# Making a Sample Design

- 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

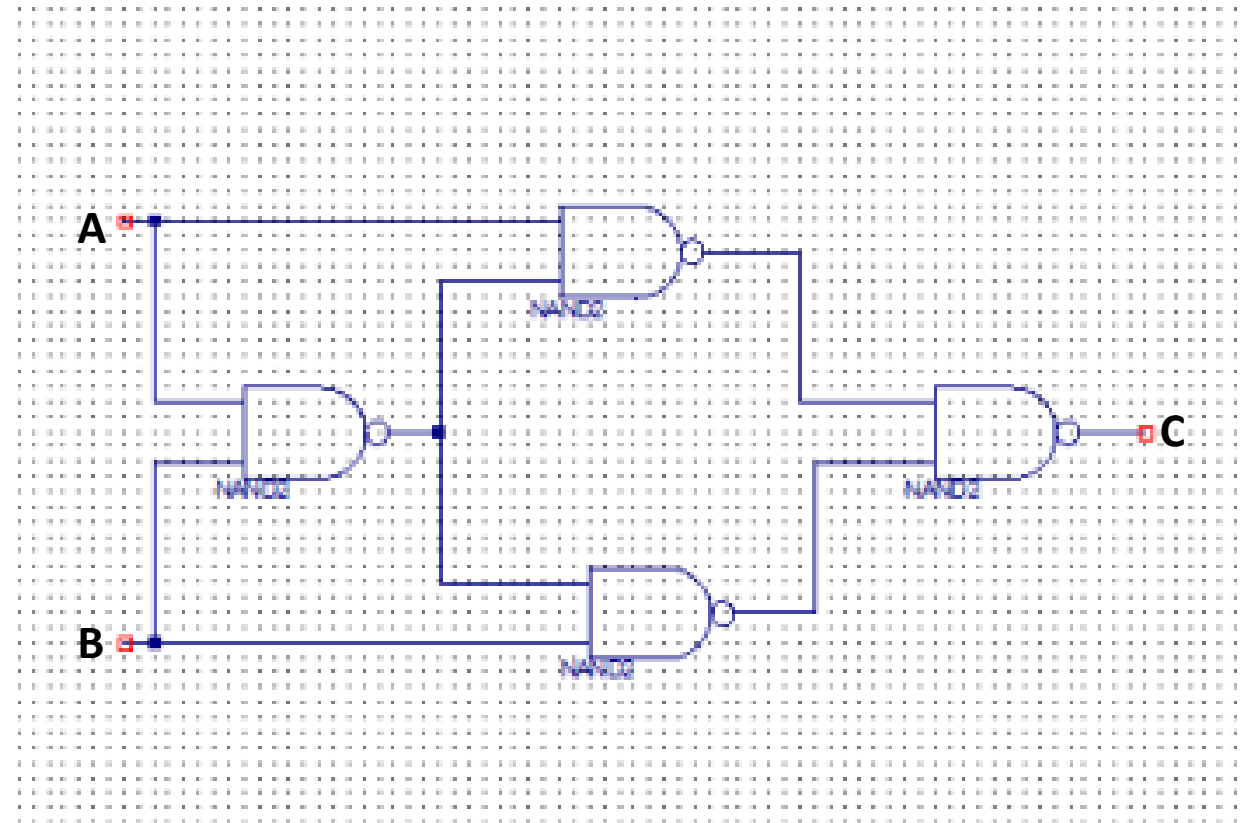
- A XOR gate
  - 2 inputs
    - A
    - B
  - 1 output
    - C



# 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.

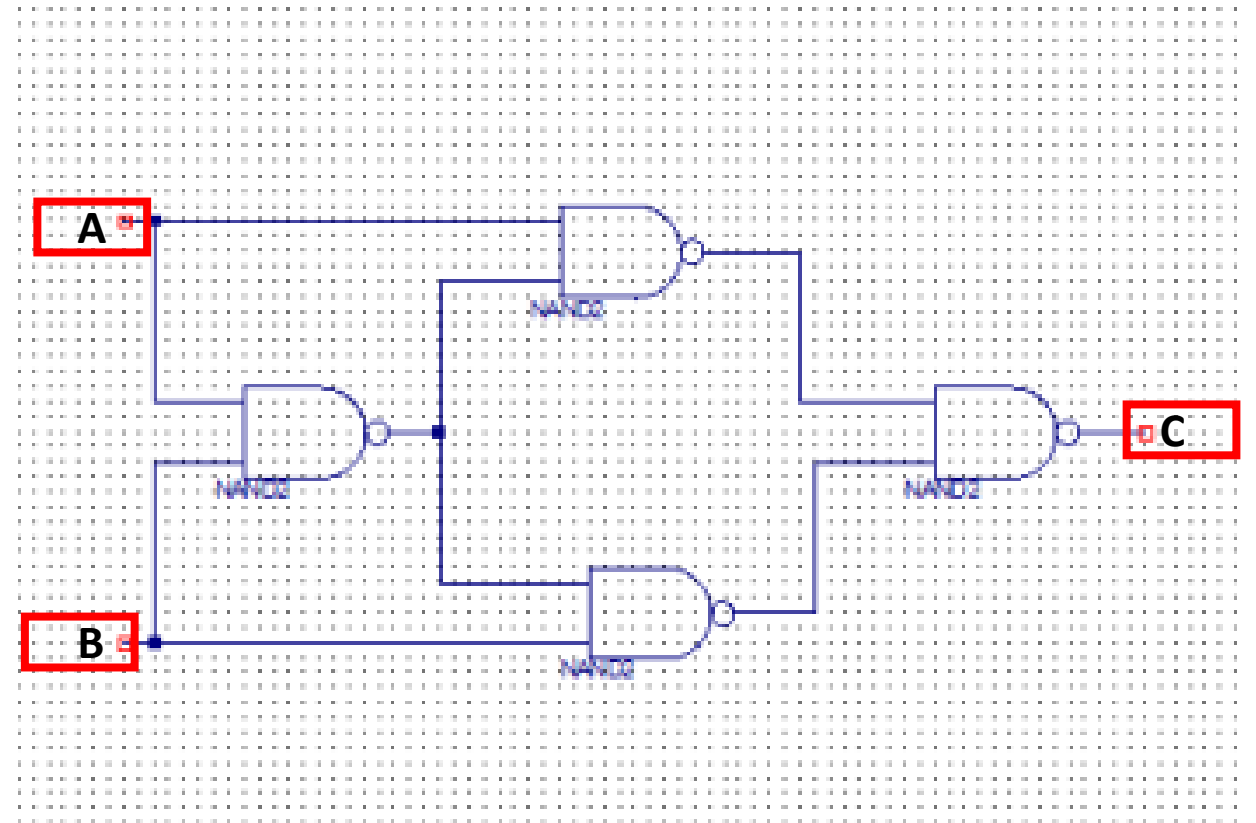
```
21  
22  
23 module xor_gate(  
24  
25 );  
26 endmodule  
27
```



# A Sample Design – XOR gate

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

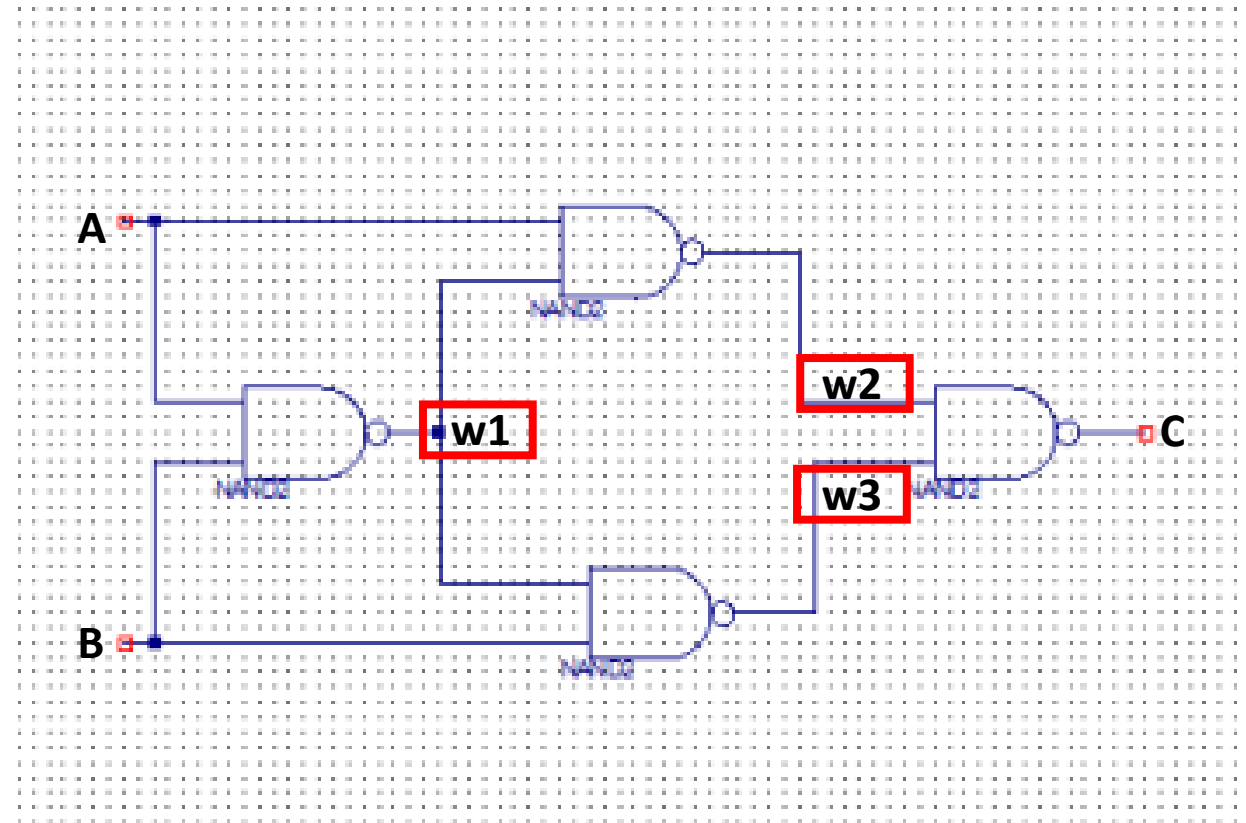
```
22 |
23 | module xor_gate(input A,
24 |               input B,
25 |               output C
26 |             );
27 |
28 | endmodule
29 |
```



# A Sample Design – XOR gate

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

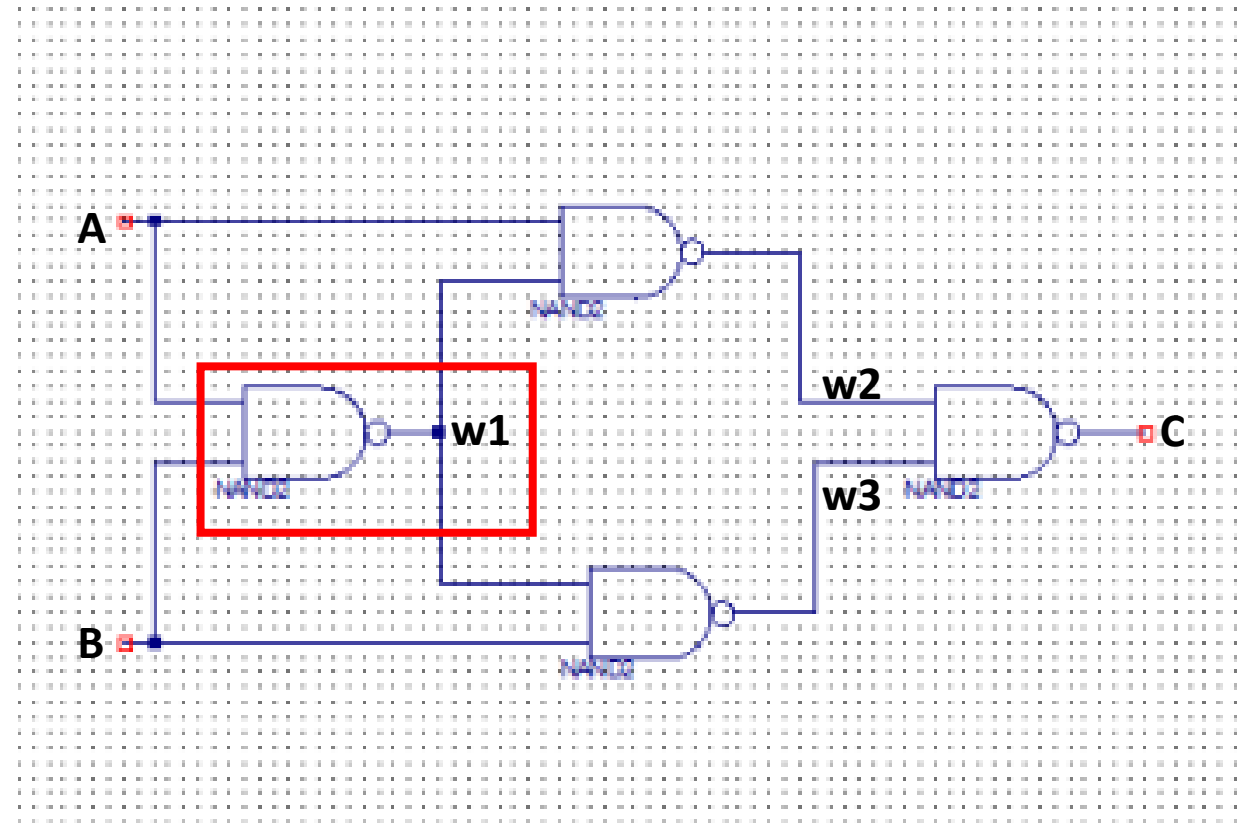
```
21  
22  
23 module xor_gate(input A,  
24                 input B,  
25                 output C  
26  
27                 );  
28  
29     wire w1, w2, w3;  
30  
31 endmodule  
32
```



# A Sample Design – XOR gate

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

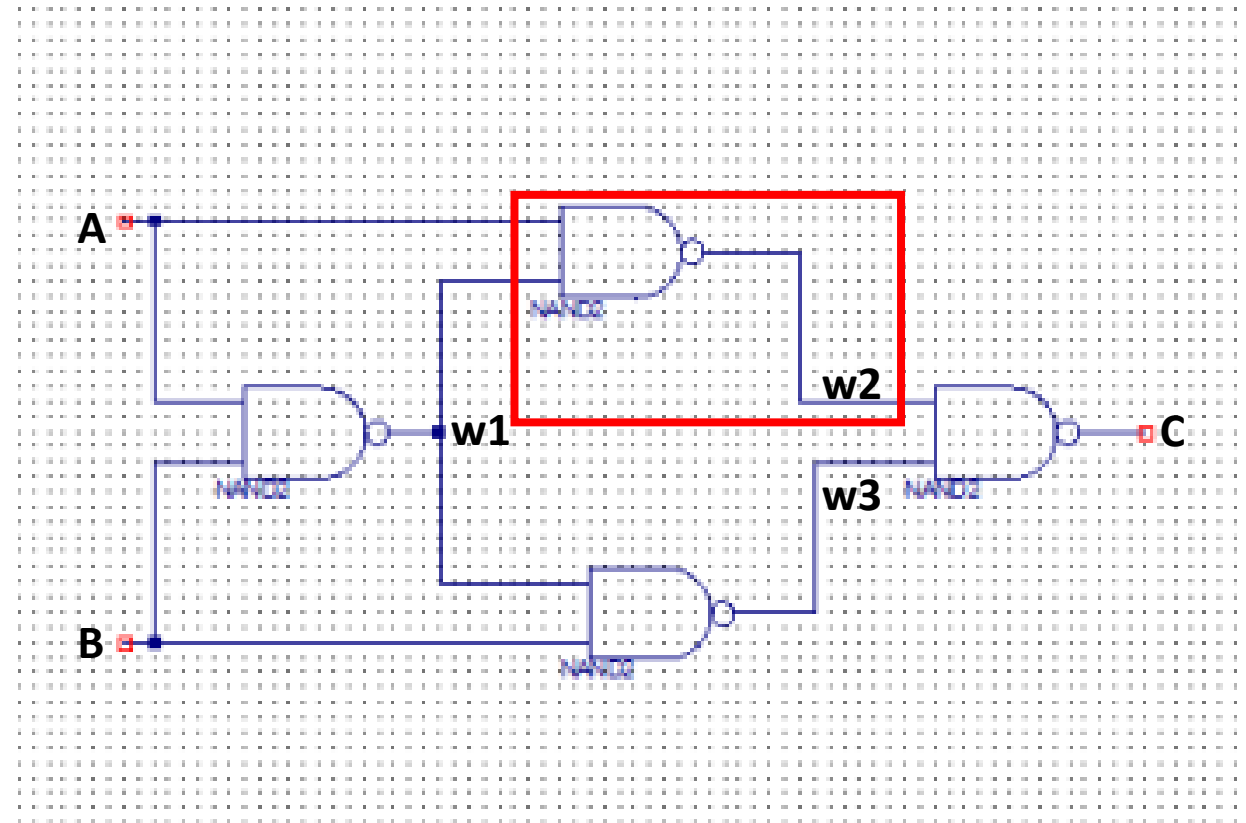
```
22
23 module xor_gate(input A,
24                 input B,
25                 output C
26
27                 );
28
29     wire w1, w2, w3;
30
31     assign w1 = ~(A & B);
32
33 endmodule
34
```



# A Sample Design – XOR gate

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

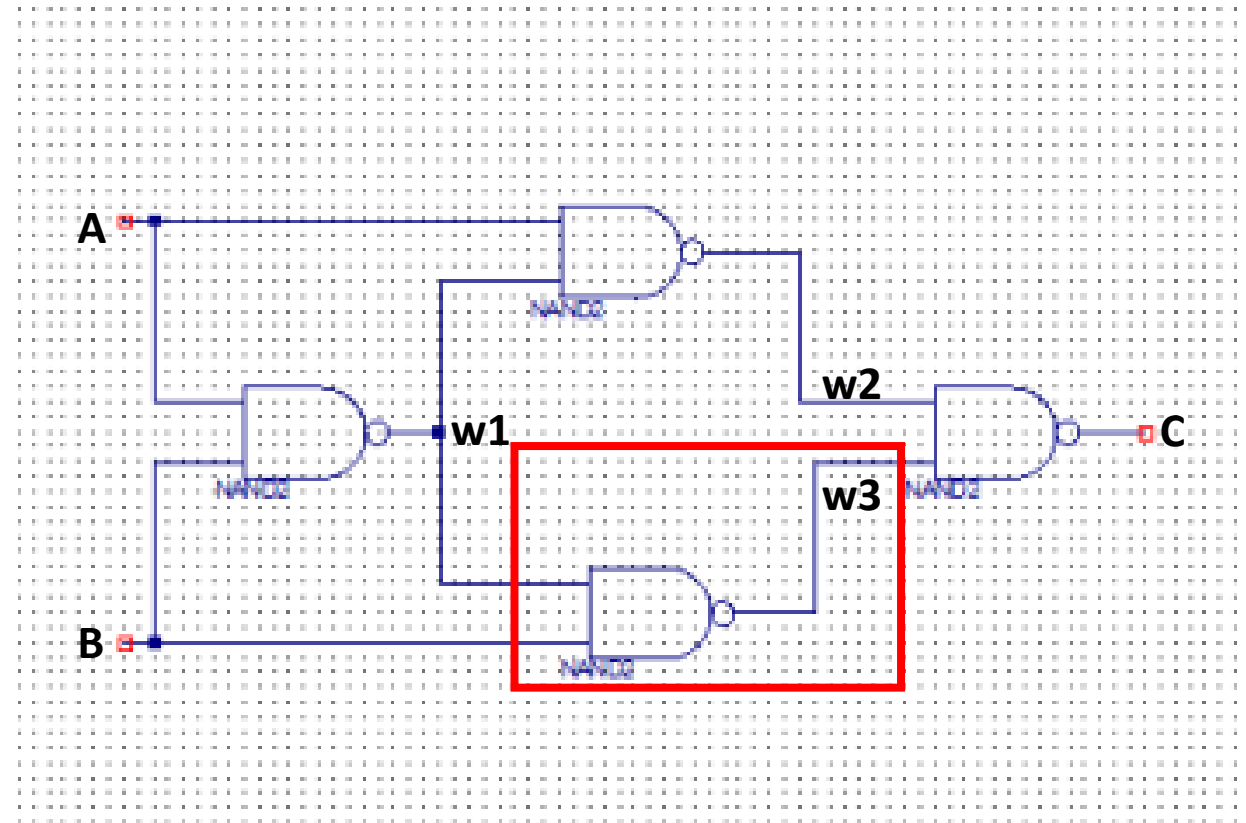
```
22 |
23 | module xor_gate(input A,
24 |                 input B,
25 |                 output C
26 |
27 |                 );
28 |
29 |     wire w1, w2, w3;
30 |
31 |     assign w1 = ~(A & B);
32 |     assign w2 = ~(A & w1);
33 |
34 | endmodule
35 |
```



# A Sample Design – XOR gate

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

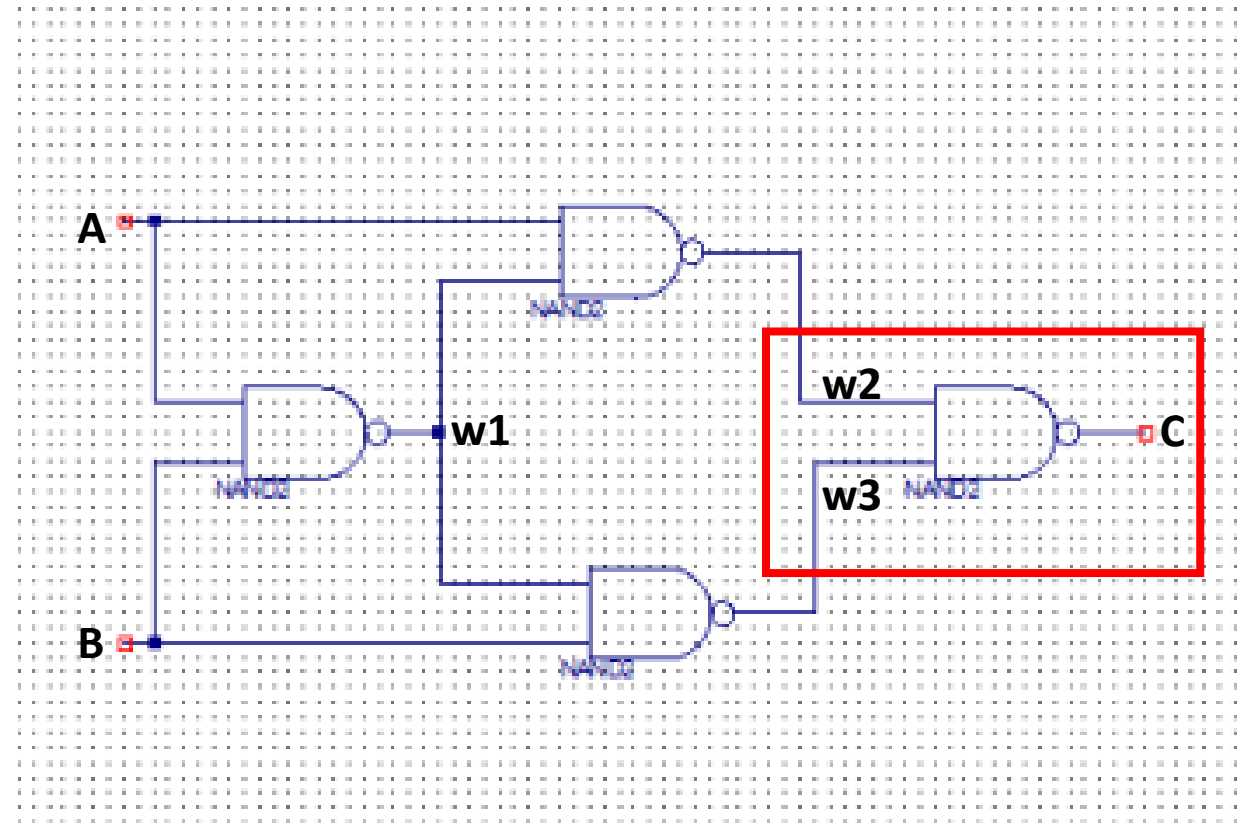
```
22 |
23 | module xor_gate(input A,
24 |                 input B,
25 |                 output C
26 |
27 |                 );
28 |
29 |     wire w1, w2, w3;
30 |
31 |     assign w1 = ~(A & B);
32 |     assign w2 = ~(A & w1);
33 |     assign w3 = ~(B & w1);
34 |
35 | endmodule
36 |
```



# A Sample Design – XOR gate

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

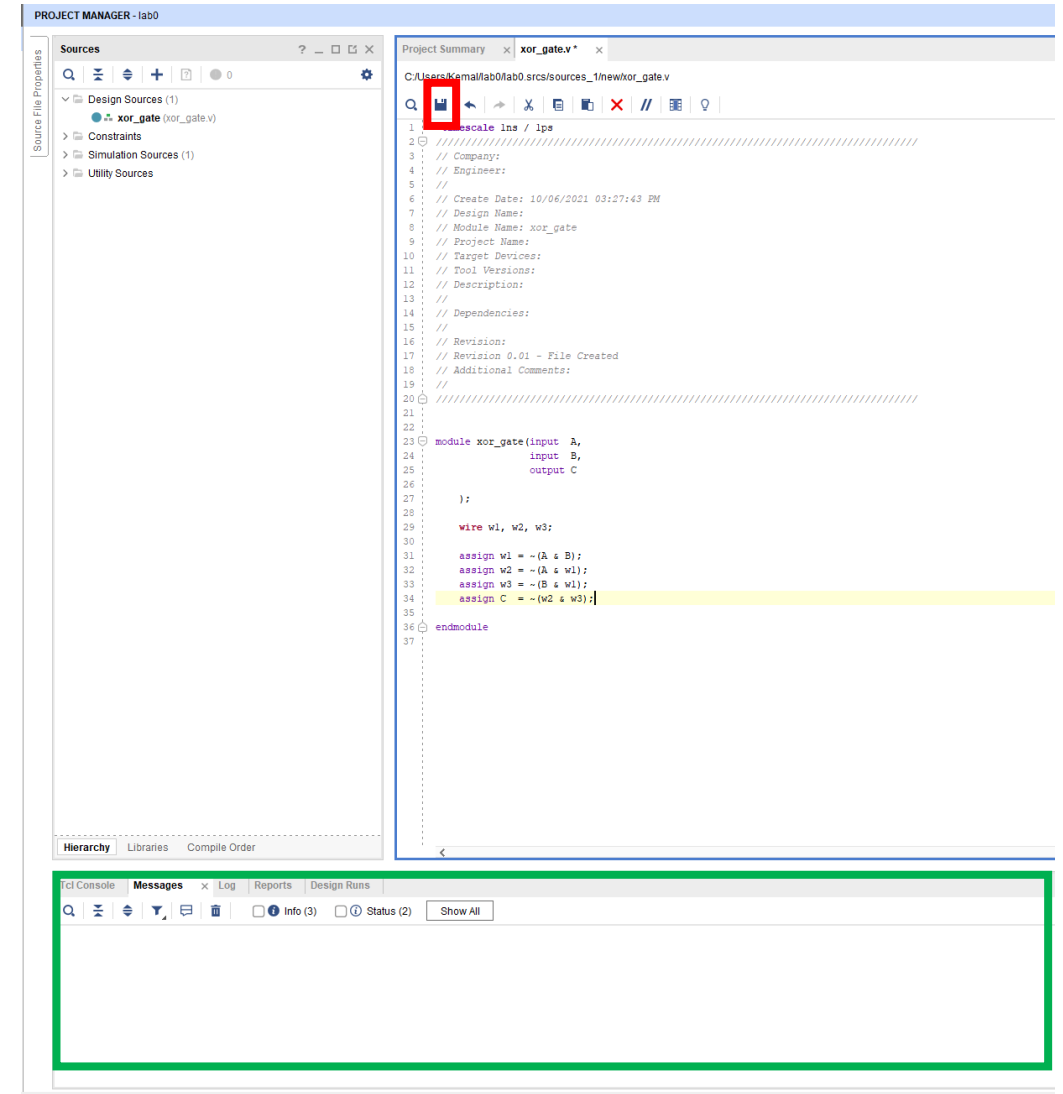
```
22
23 module xor_gate(input A,
24                 input B,
25                 output C
26
27                 );
28
29     wire w1, w2, w3;
30
31     assign w1 = ~(A & B);
32     assign w2 = ~(A & w1);
33     assign w3 = ~(B & w1);
34     assign C = ~(w2 & w3);
35
36 endmodule
37
```





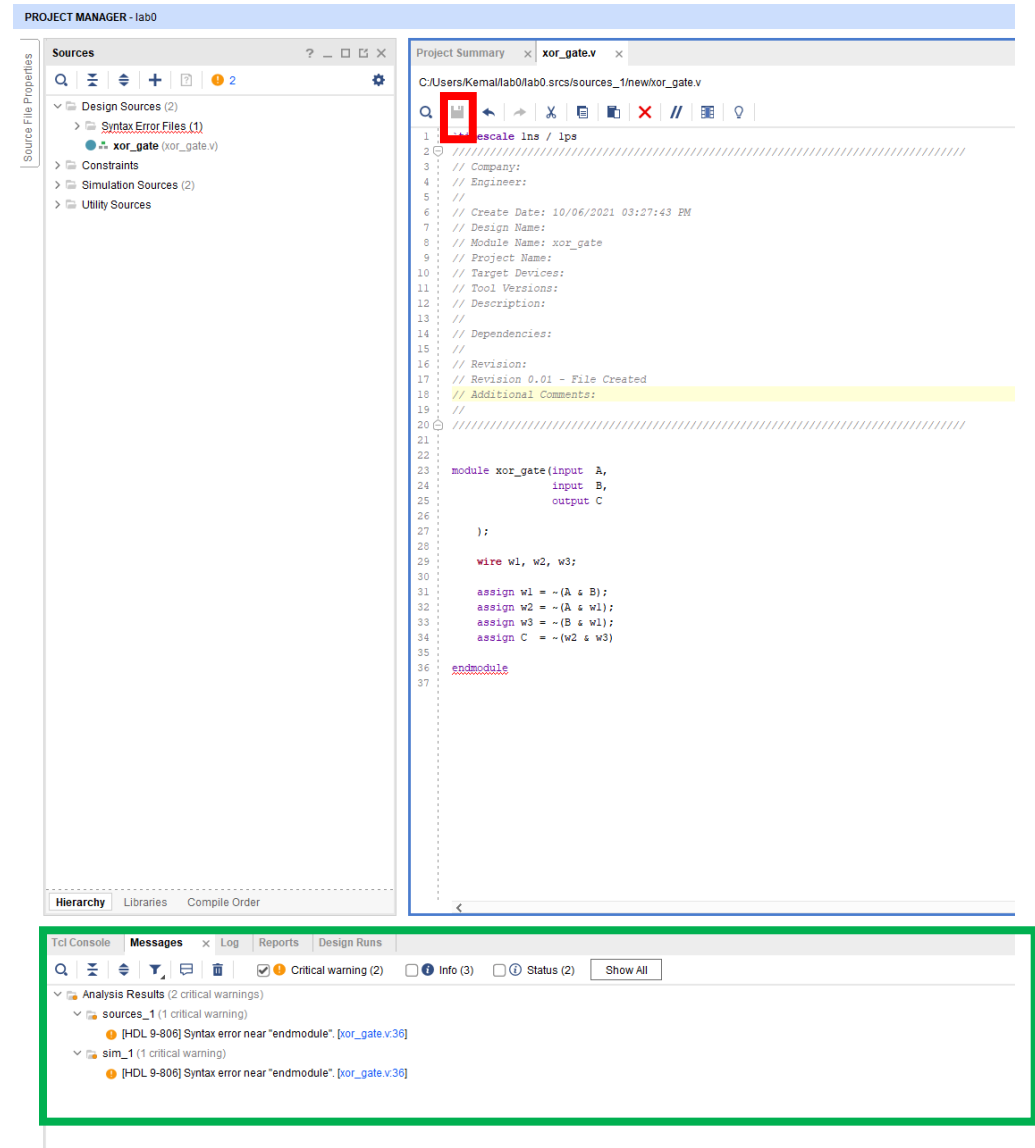
# A Sample Design – XOR gate

- 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**.



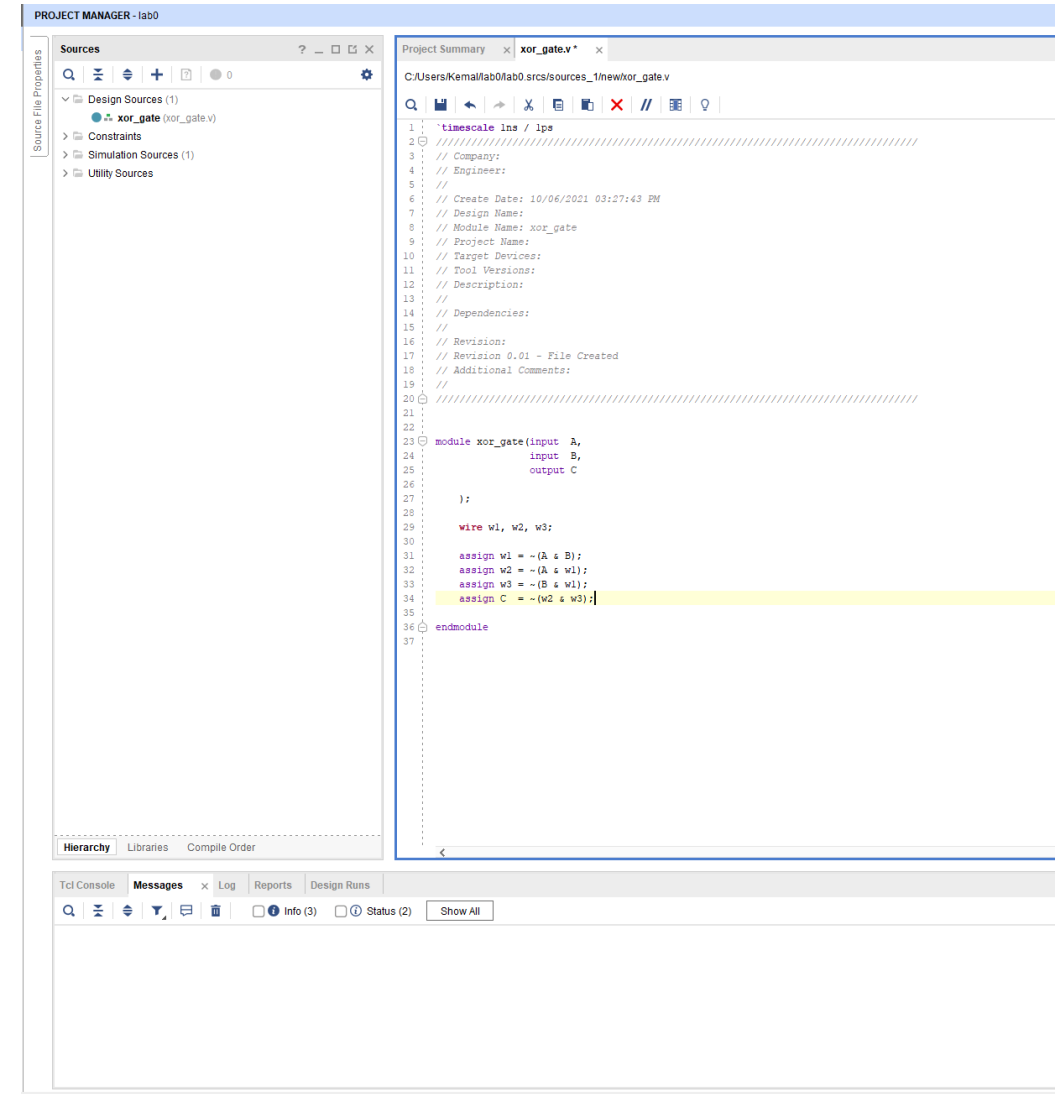
# A Sample Design – XOR gate

- 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.



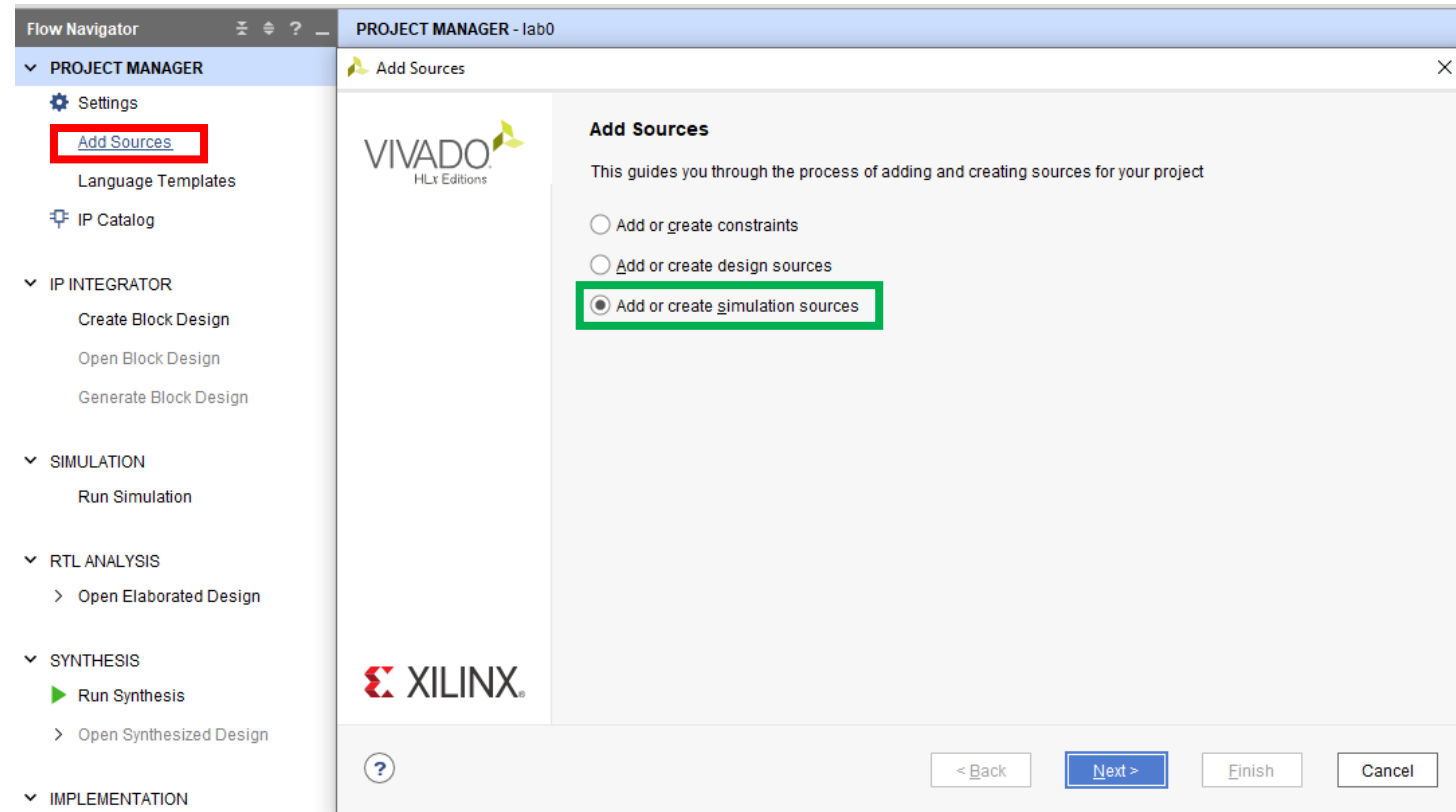
# A Sample Design – XOR gate

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



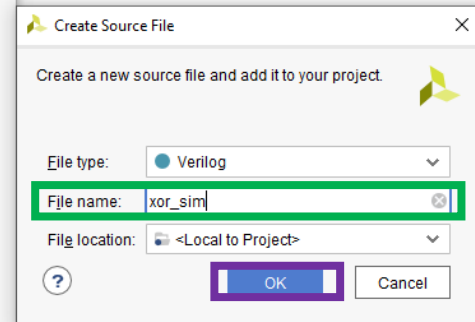
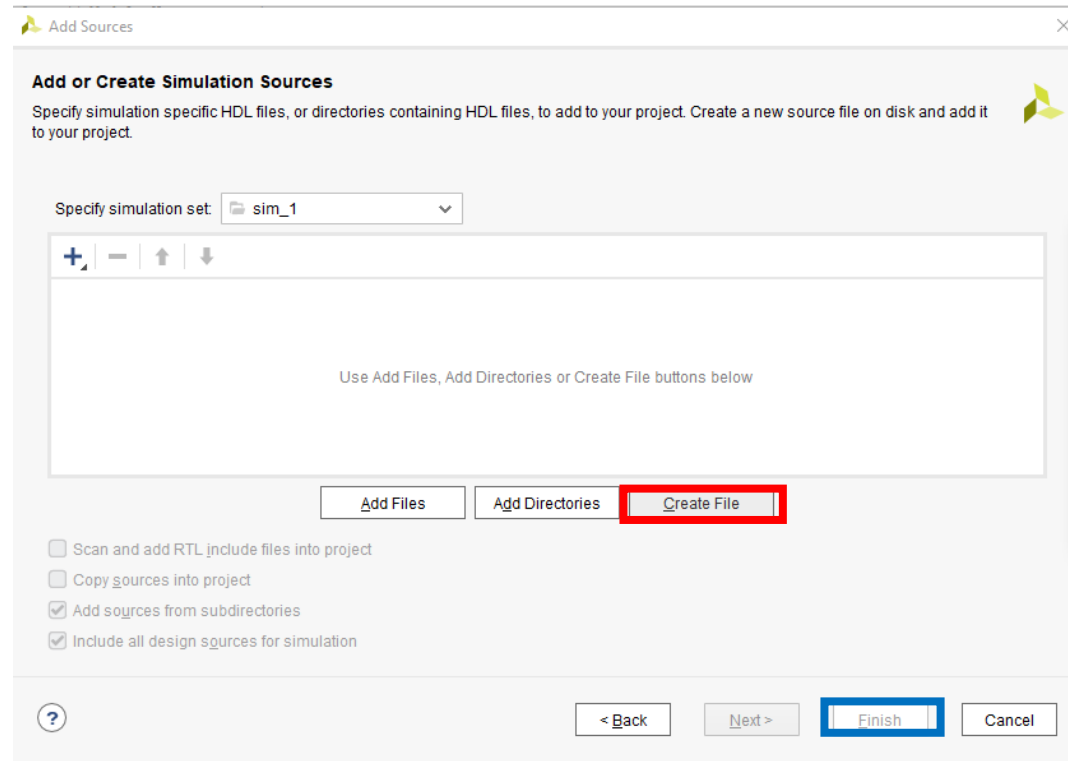
# Simulation

- 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

Define Module

Define a module and specify I/O Ports to add to your source file.  
For each port specified:  
MSB and LSB values will be ignored unless its Bus column is checked.  
Ports with blank names will not be written.

**Module Definition**

Module name:

**I/O Port Definitions**

Port Name	Direction	Bus	MSB	LSB
	input	<input type="checkbox"/>	0	0

?

**OK** **Cancel**

Define Module

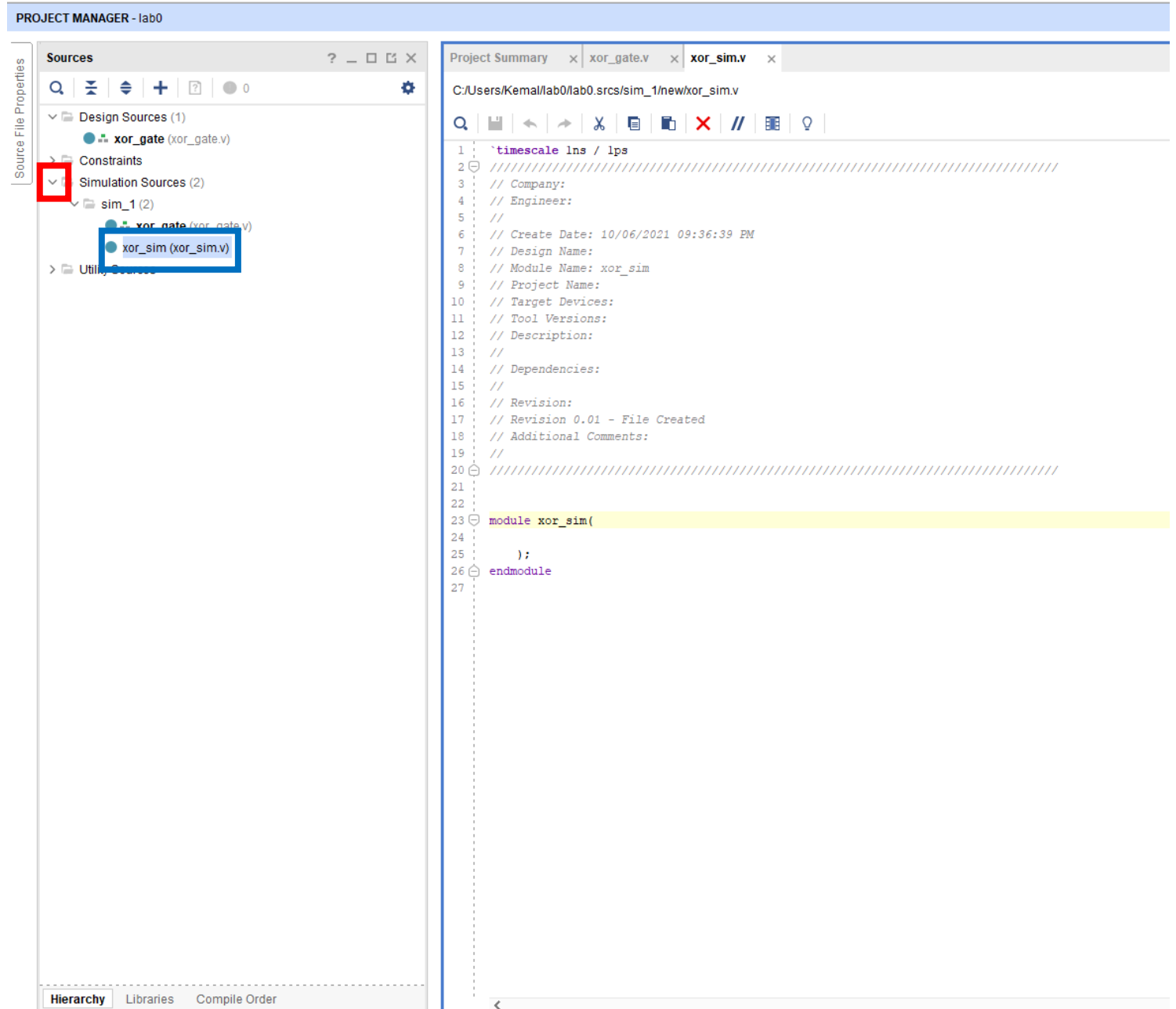
?

The module definition has not been changed.  
Are you sure you want to use these values?

**Yes** **No**

# 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.

The screenshot displays the 'PROJECT MANAGER - lab0' interface. On the left, the 'Sources' panel shows a tree structure with 'Design Sources (1)' containing 'xor\_gate (xor\_gate.v)', 'Constraints', 'Simulation Sources (1)' containing 'sim\_1 (1)' which includes 'xor\_sim (xor\_sim.v) (1)' and 'UUT: xor\_gate (xor\_gate.v)', and 'Utility Sources'. The 'xor\_sim' entry is highlighted with a green box. On the right, the 'Project Summary' editor for 'xor\_sim.v' is open, showing a Verilog code snippet. A red box highlights the 'Save' icon (a floppy disk) in the toolbar above the code editor.

```
// Create Date: 10/06/2021 09:36:39 PM
// Design Name:
// Module Name: xor_sim
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
////////////////////////////////////

module xor_sim(
);
//Inputs
reg A;
reg B;

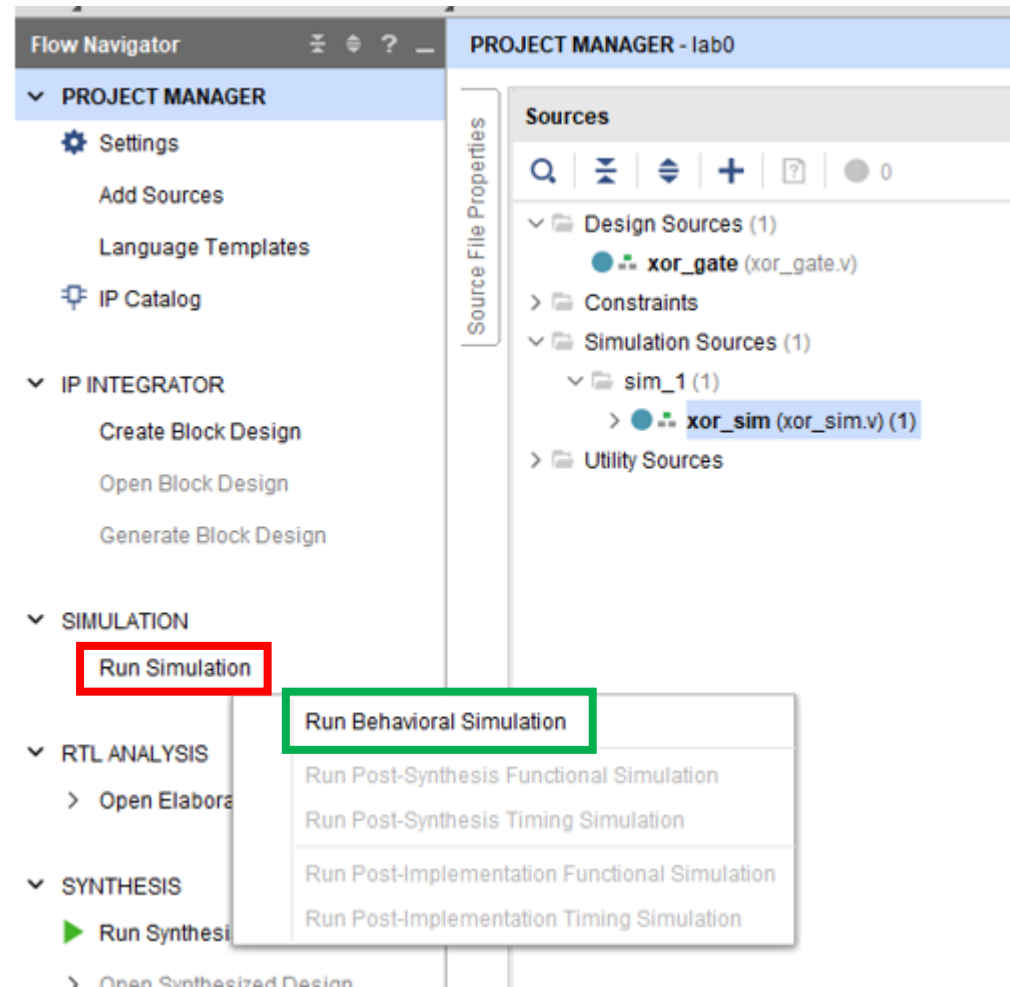
//Outputs
wire C;

//Instantiate the UUT
xor_gate UUT(
.A(A),
.B(B),
.C(C)
);

//Initialize Inputs
initial begin
    A = 0;
    B = 0;
    #10;
    A = 1;
    #10;
    A = 0;
    B = 1;
    #10;
    A = 1;
    #10;
end
endmodule
```

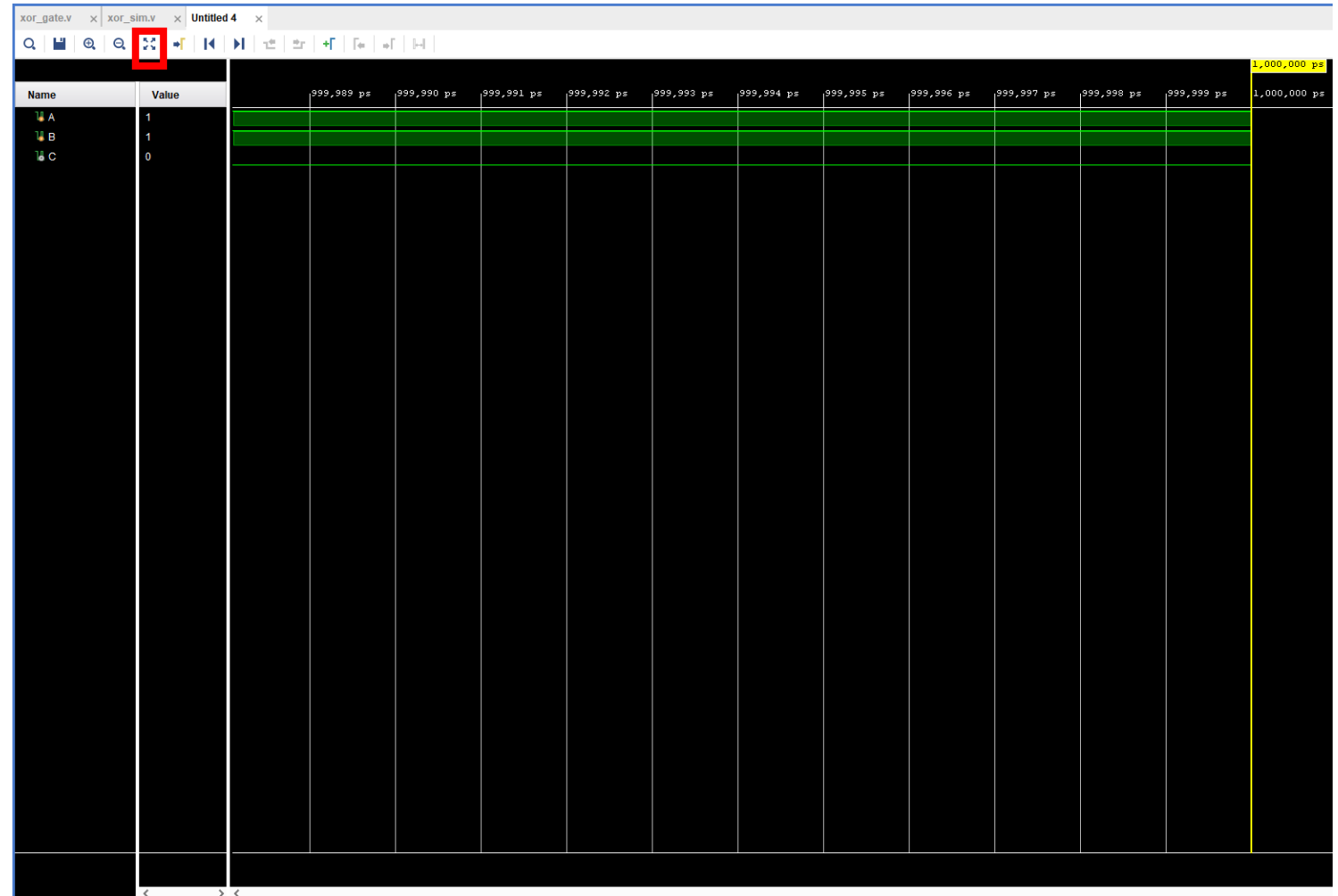
# Simulation

- Click on **Run Simulation**
- Click on **Run Behavioral Simulation**



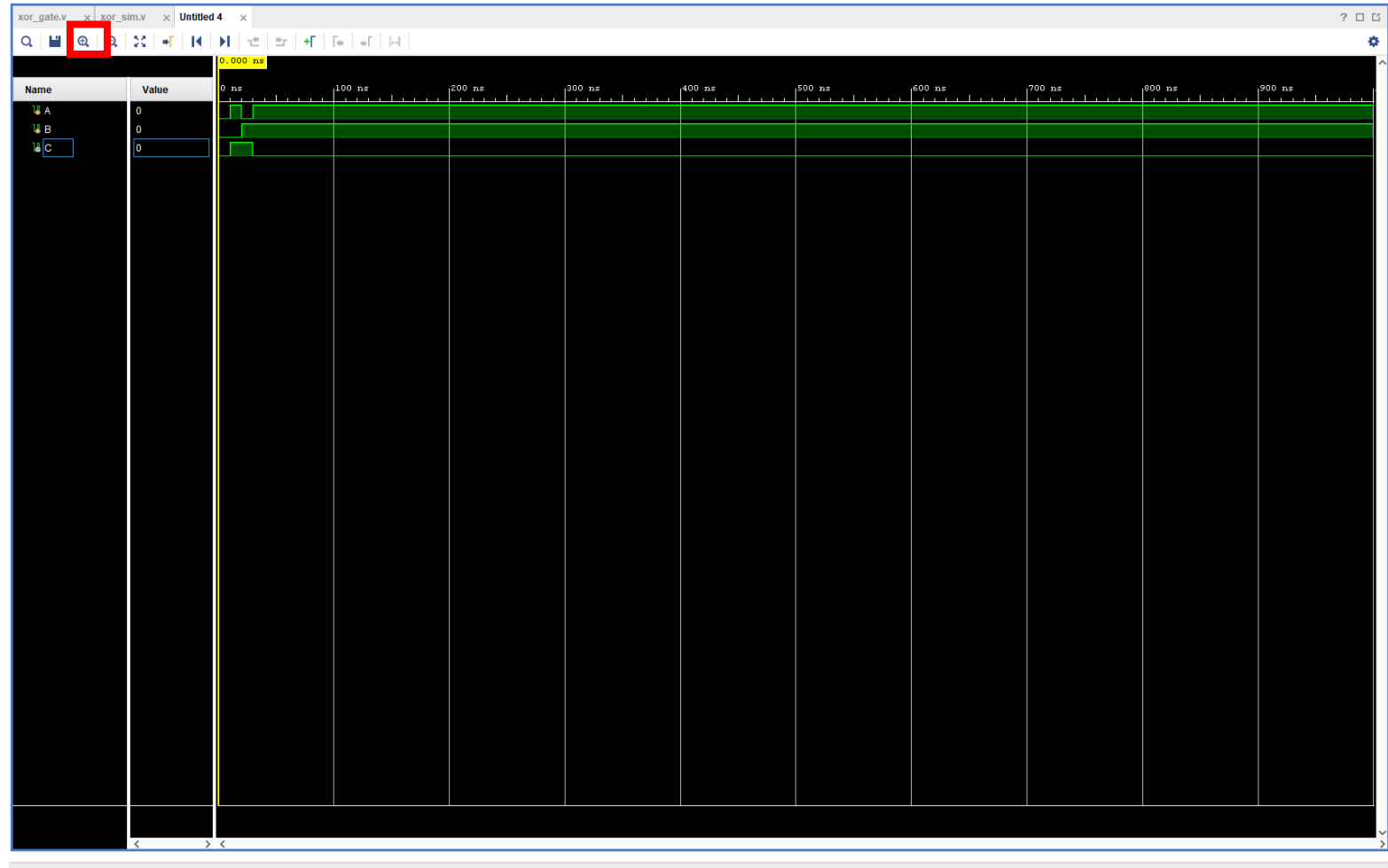
# Simulation

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



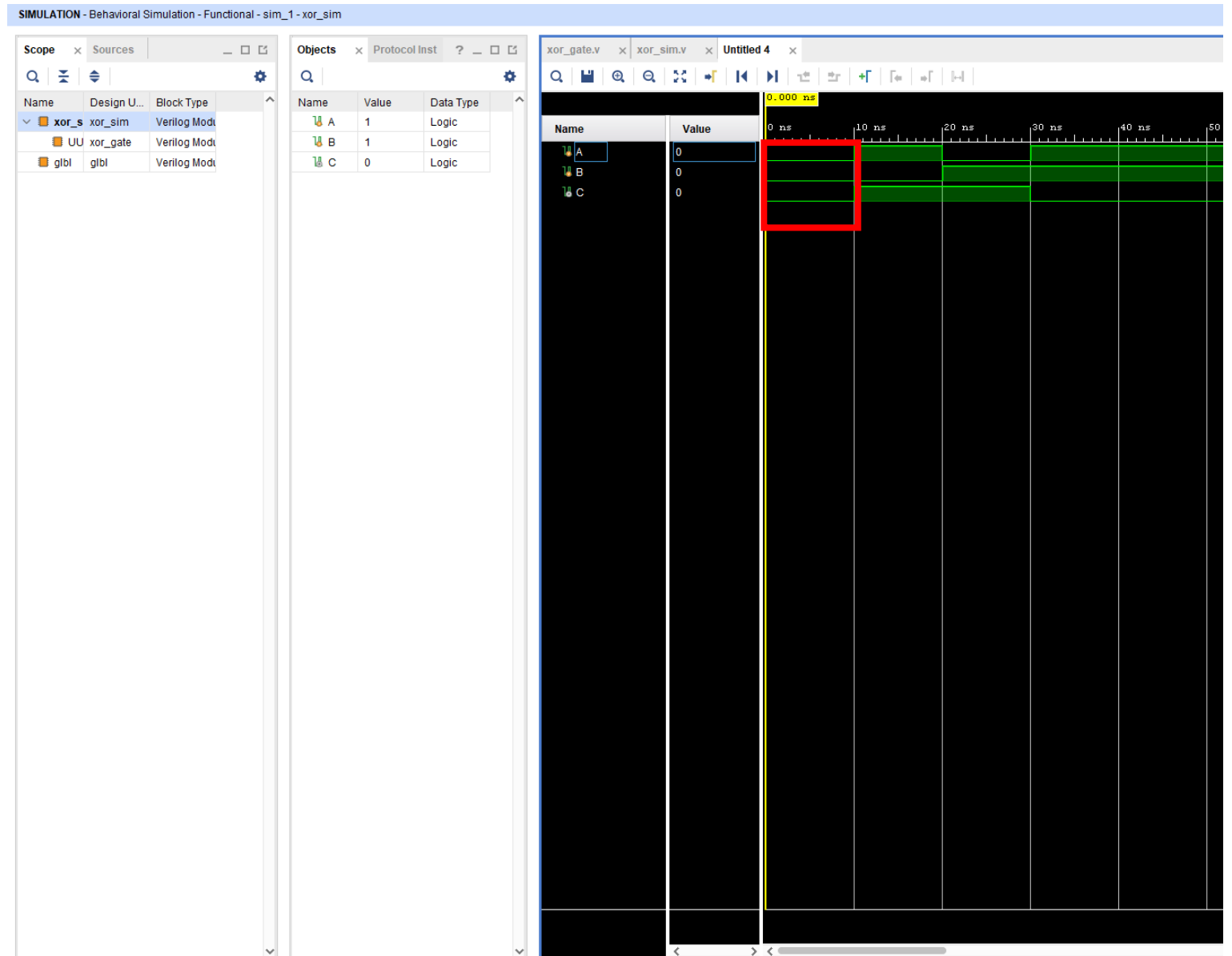
# Simulation

- 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.



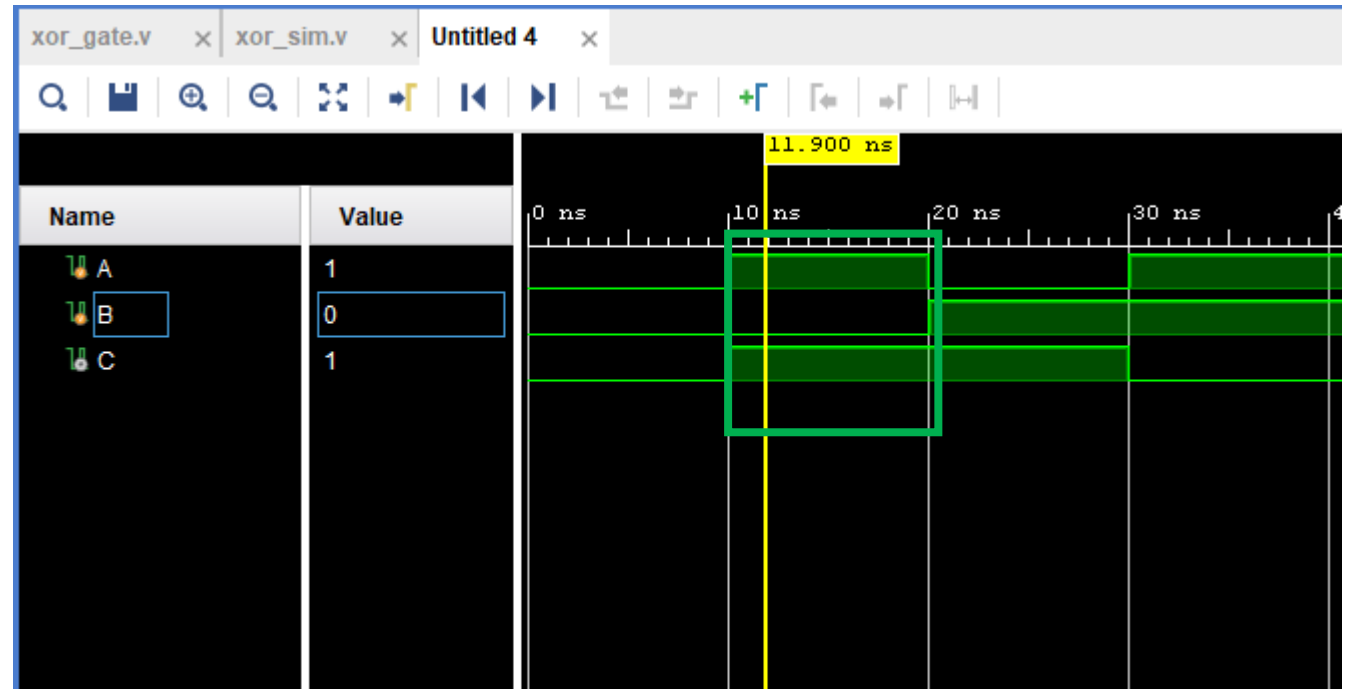
# Simulation

- $\oplus$  is XOR operation.
- $A \oplus B = C$
- $0 \oplus 0 = 0$



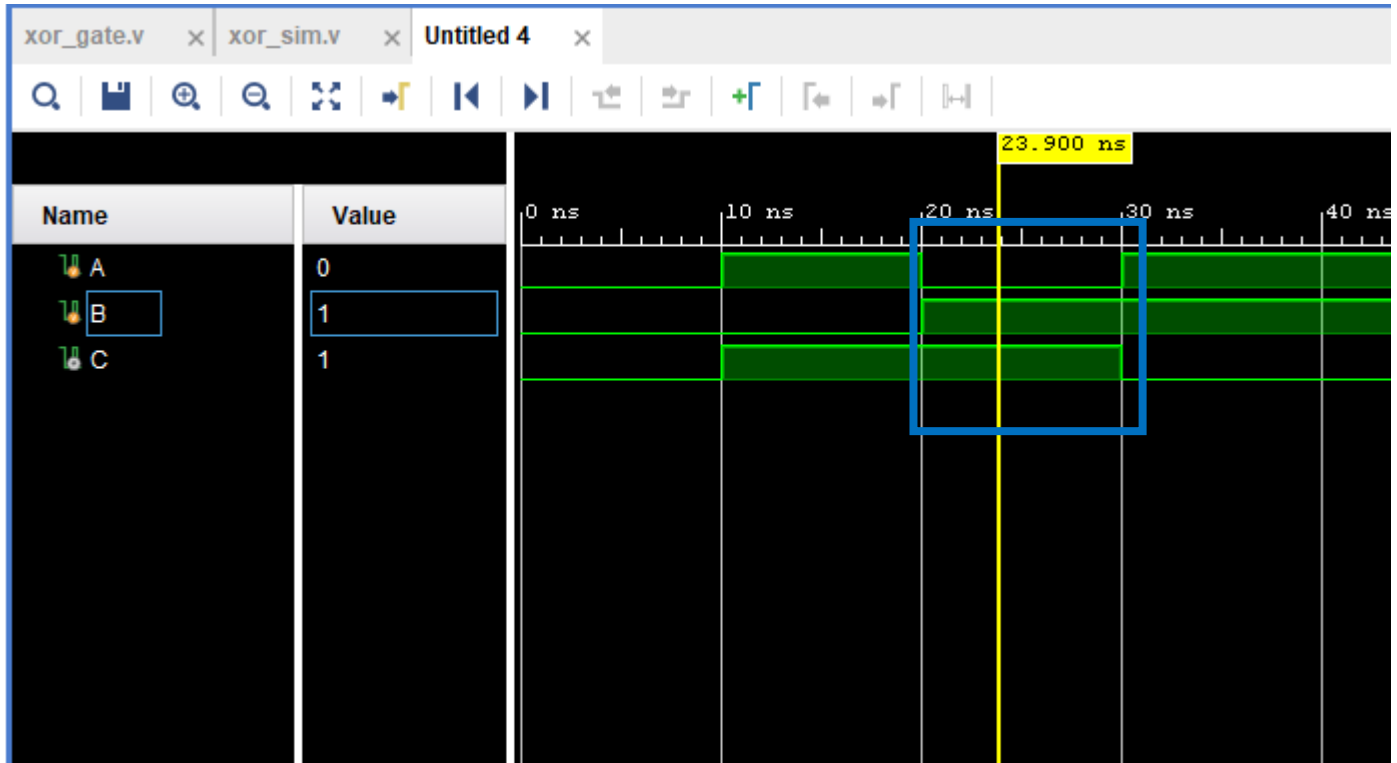
# Simulation

- $\oplus$  is XOR operation.
- $A \oplus B = C$
- $0 \oplus 0 = 0$
- $1 \oplus 0 = 1$



# Simulation

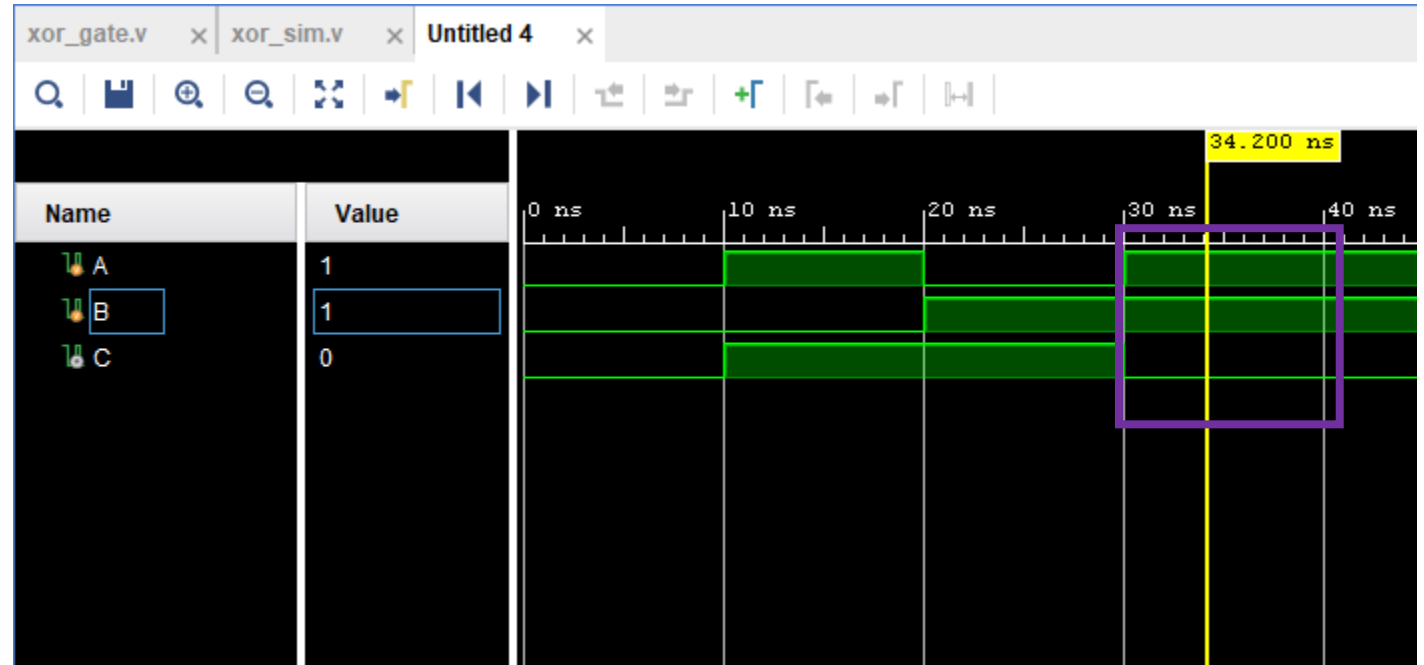
- $\oplus$  is XOR operation.
- $A \oplus B = C$
- $0 \oplus 0 = 0$
- $1 \oplus 0 = 1$
- $0 \oplus 1 = 1$





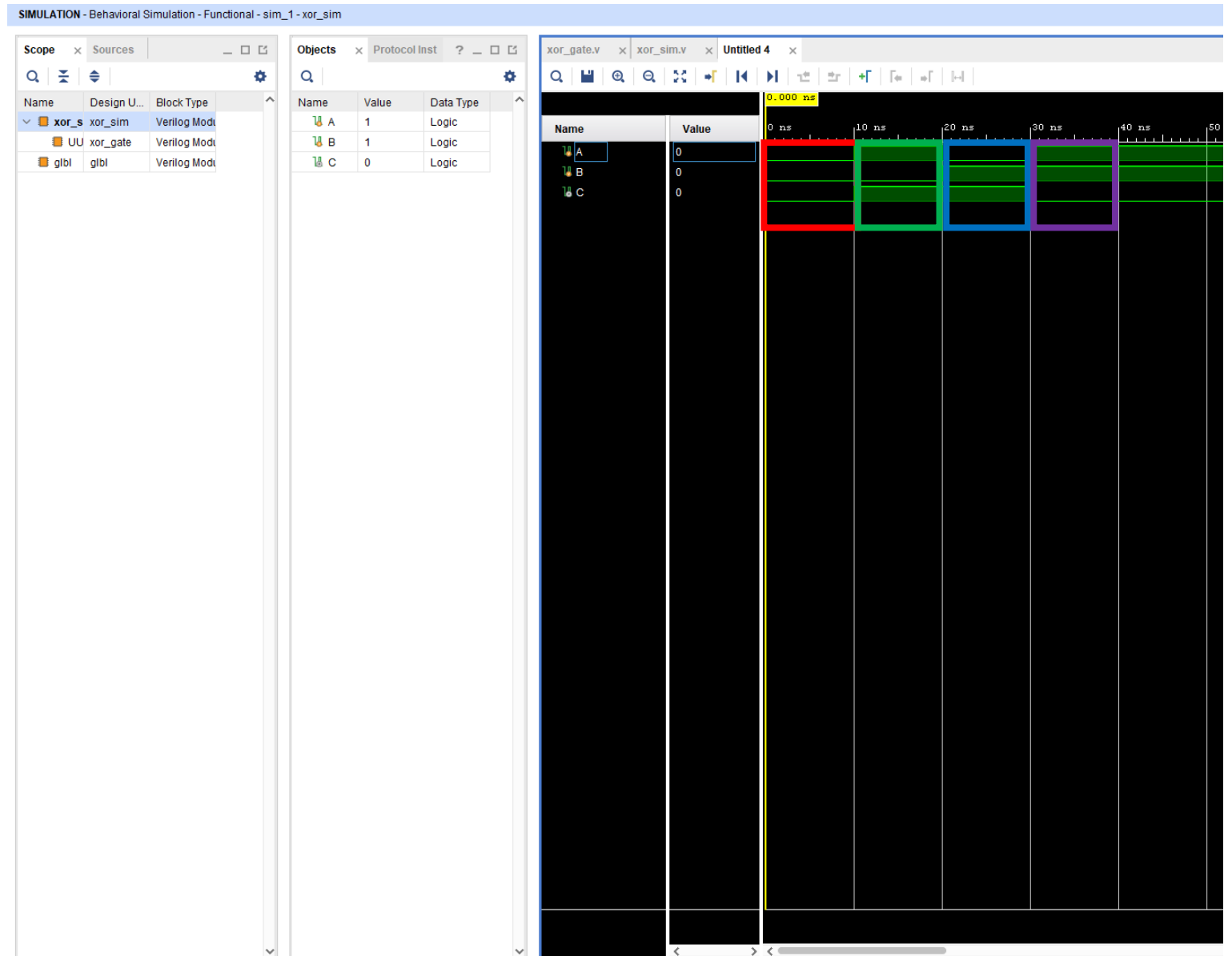
# Simulation

- $\oplus$  is XOR operation.
- $A \oplus B = C$
- $0 \oplus 0 = 0$
- $1 \oplus 0 = 1$
- $0 \oplus 1 = 1$
- $1 \oplus 1 = 0$



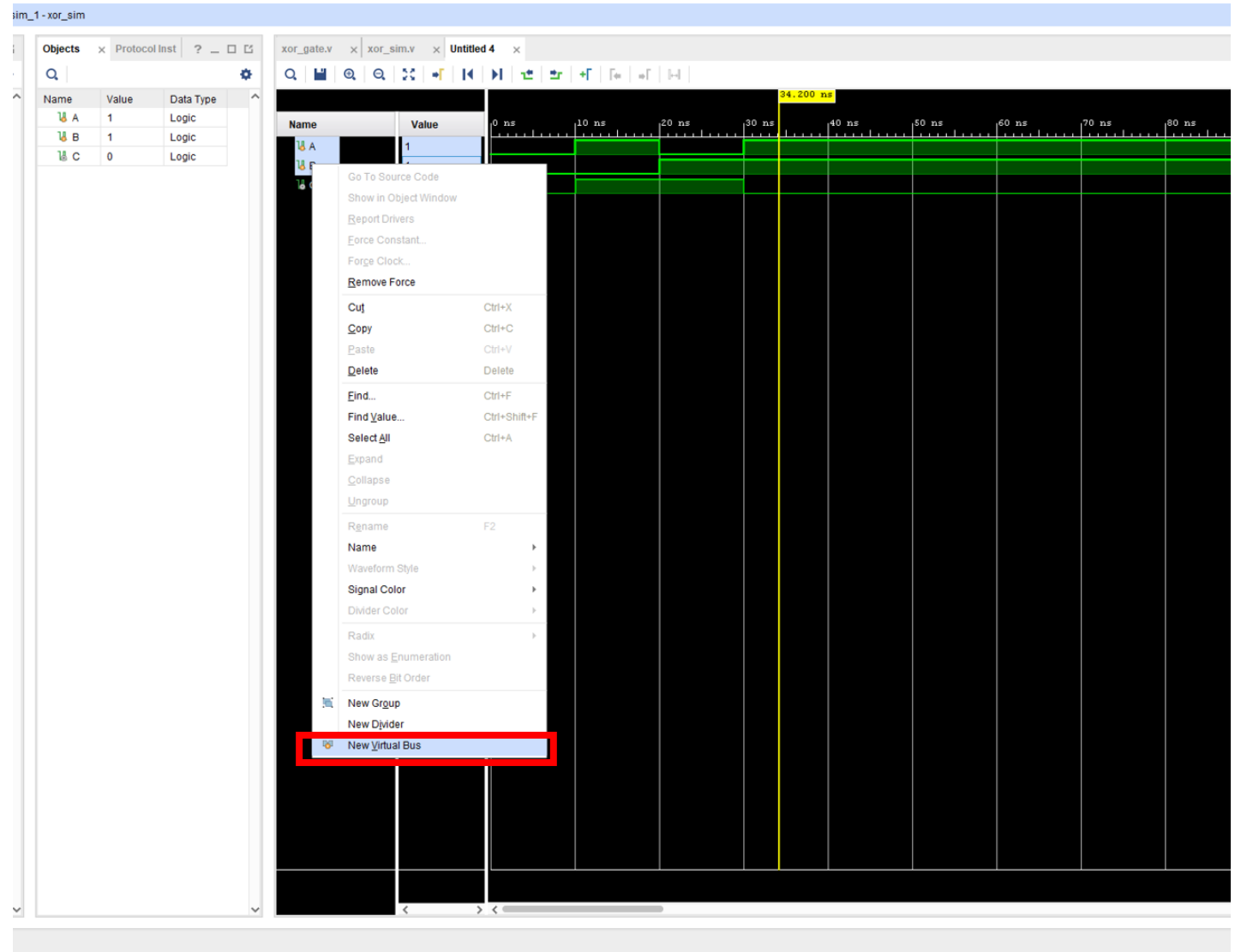
# Simulation

- $\oplus$  is XOR operation.
- $A \oplus B = C$
- $0 \oplus 0 = 0$
- $1 \oplus 0 = 1$
- $0 \oplus 1 = 1$
- $1 \oplus 1 = 0$



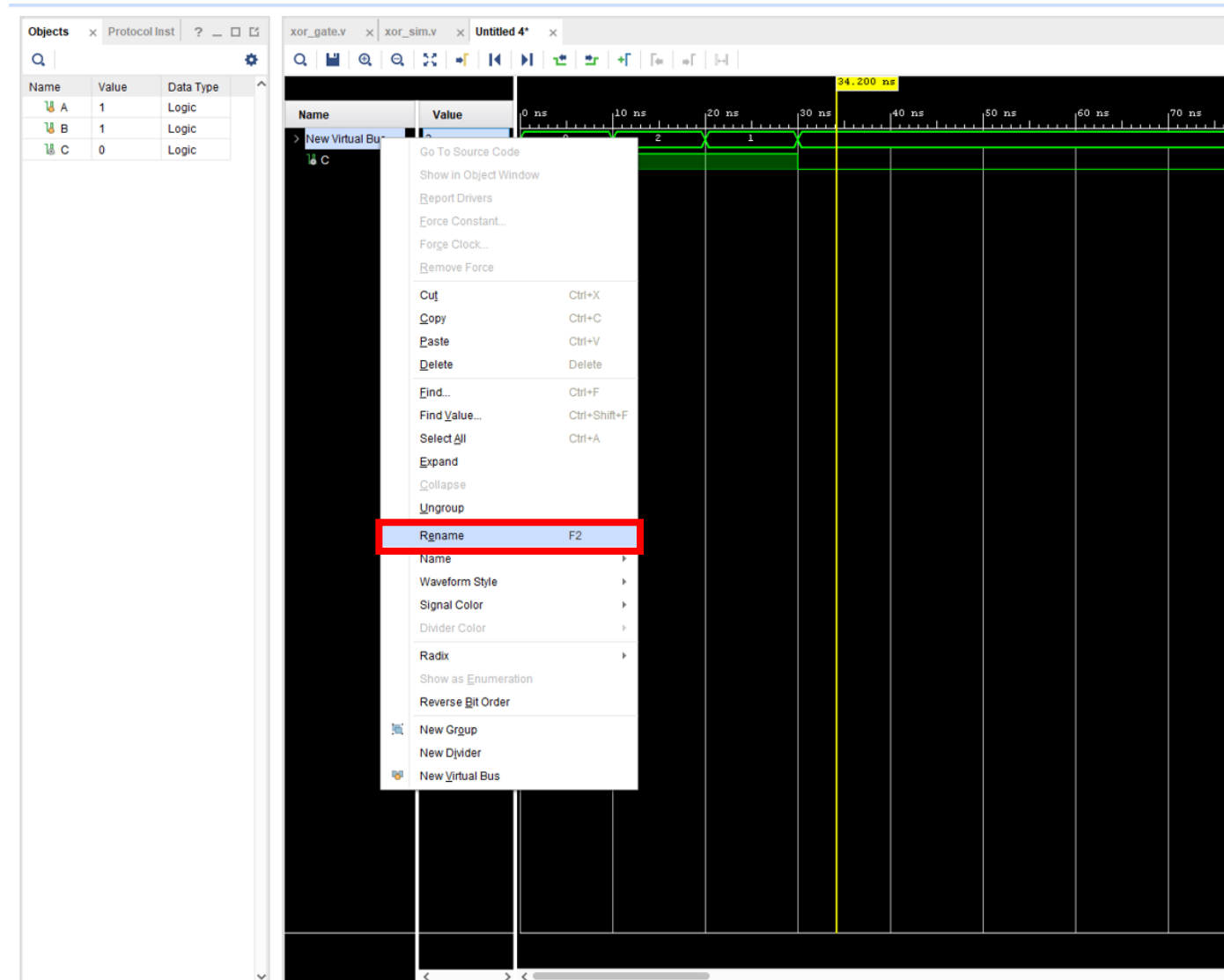
# Simulation

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



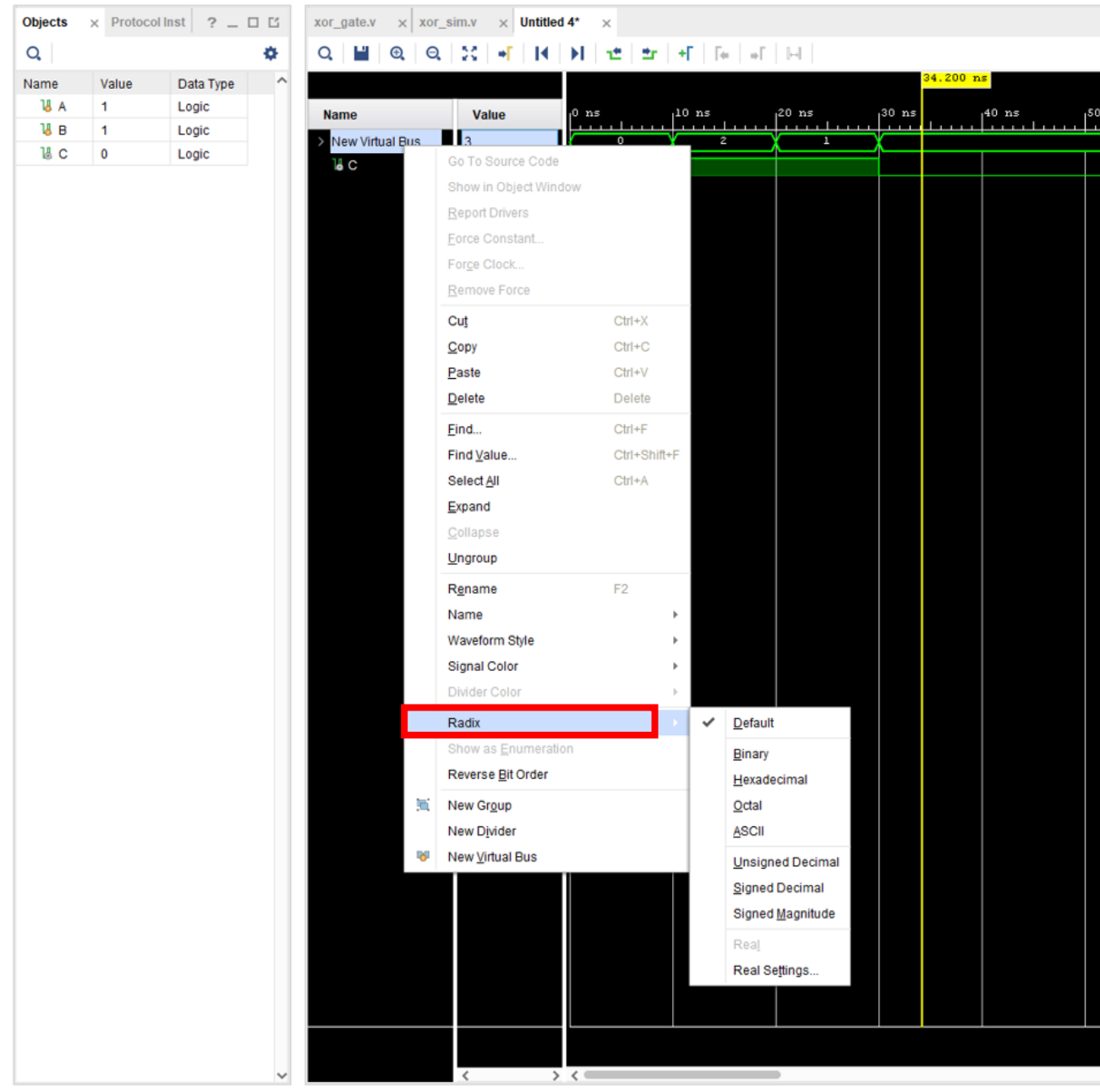
# Simulation

- You may **Rename** your virtual bus



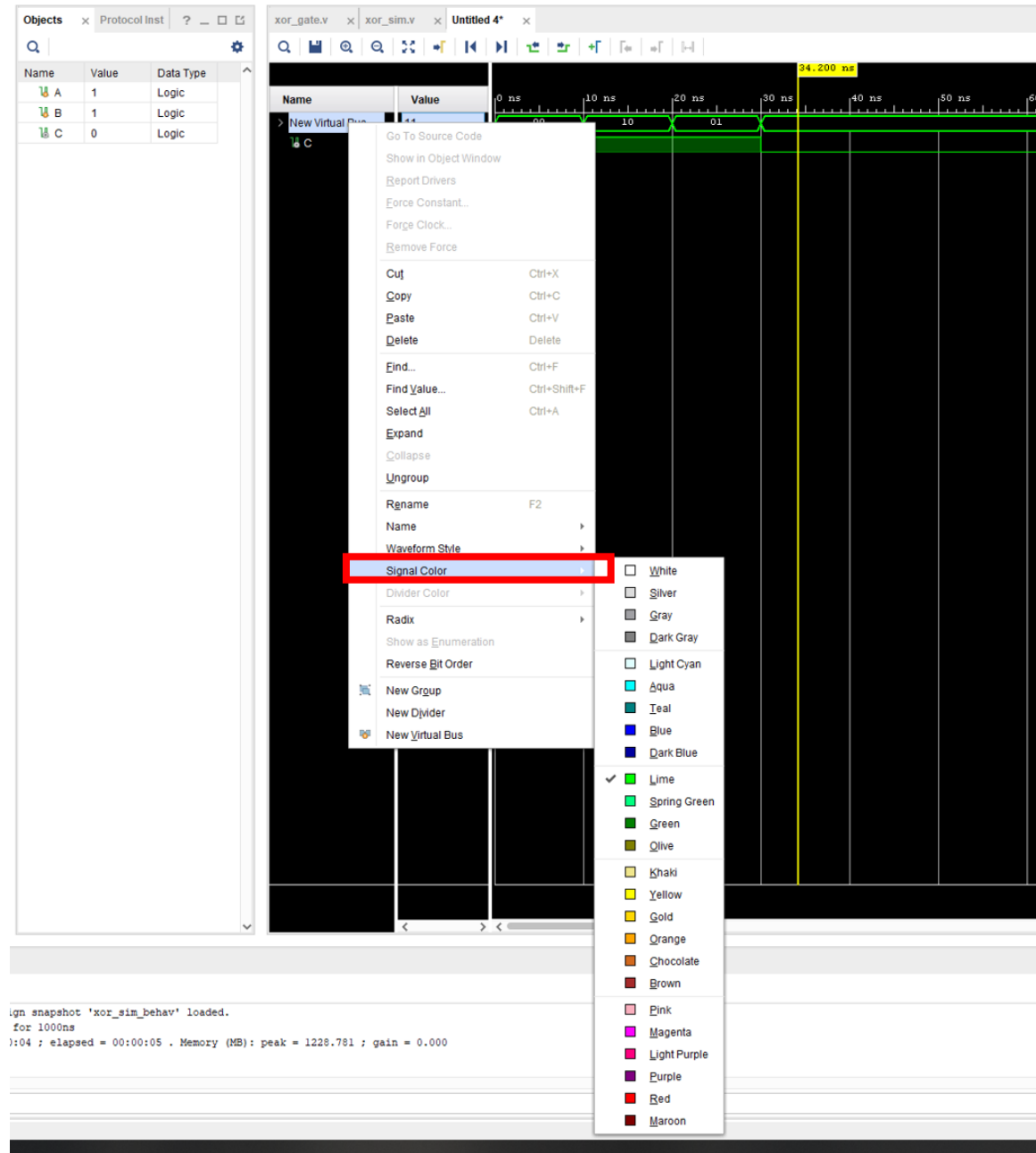
# Simulation

- You may change **Radix** of virtual bus



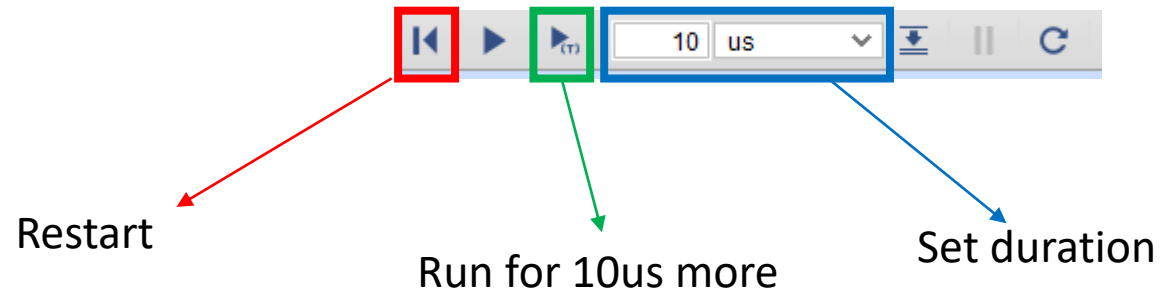
# Simulation

- You may change **Signal Color** for better visualization



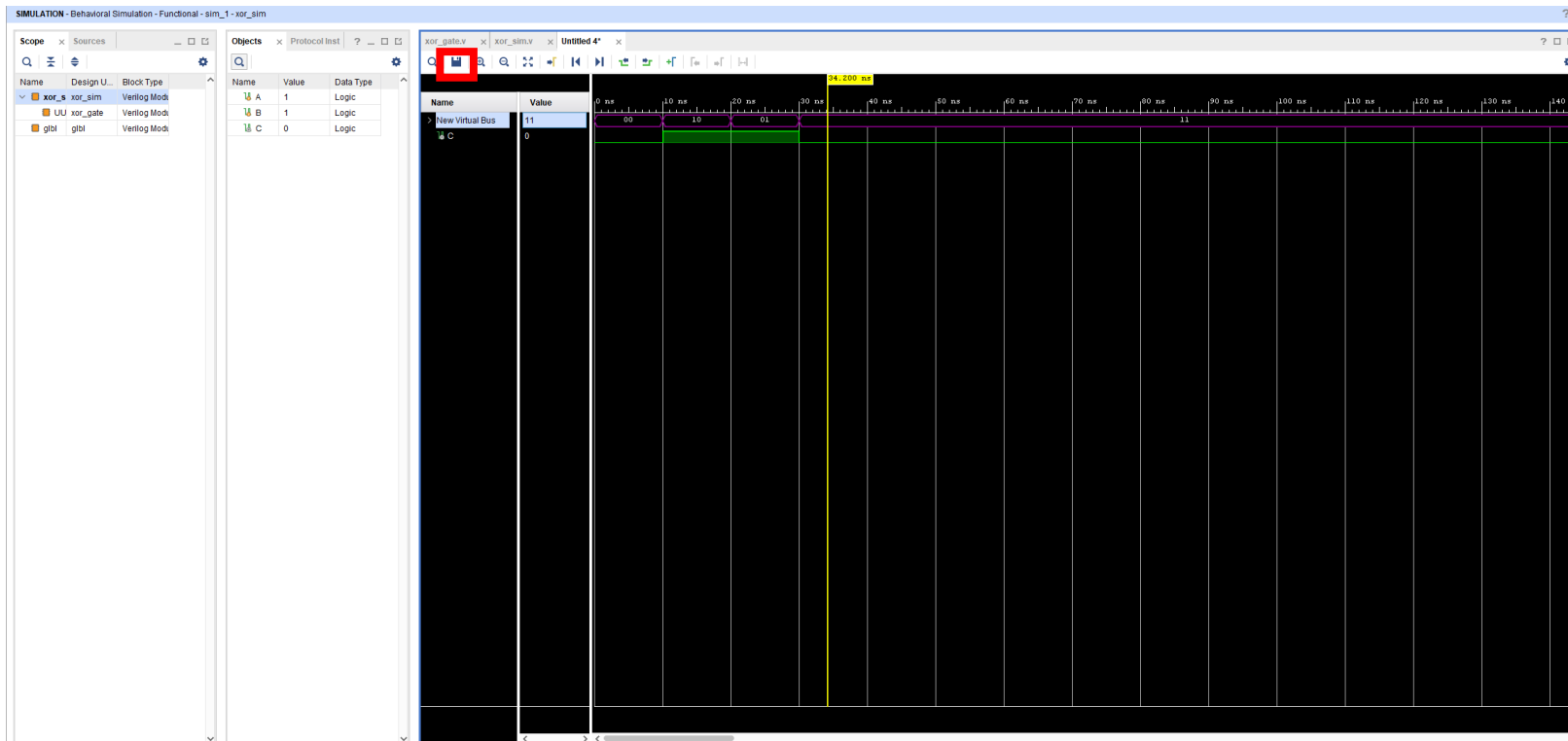
# Simulation

- 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**

