

INTRODUCTION HANDOUT

A hardware description language (HDL) is a specialized computer language used to describe **the structure and behavior** of electronic circuits. These languages are

- VHDL (VHSIC Hardware Description Language) (VHSIC -> Very High-Speed Integrated Circuit)
- Verilog
- SystemVerilog
- CHISEL

What is Verilog?

Google Definition from Science Direct:

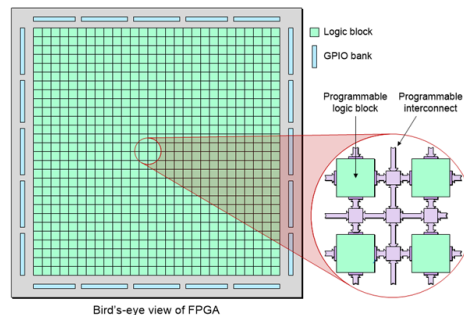
Verilog is a hardware description language used for simulation and synthesis in the field of computer science. It defines the syntax and semantics for creating digital systems, with a focus on creating a synthesizable subset of language constructs for logic synthesis.

My definition:

Verilog is the language where you can create digital circuits in your code. It is like using paint, very basic and straightforward, low-level. When complexity increases It might be hard to manage.

Where do we use Verilog?

The most important usage of Verilog is FPGA programming, but what is FPGA? FPGA is field programmable gate array:



There are ready to use blocks in the IC, we are programming functions and connections of these blocks. Literally hardware programming! There should be tool to use this language. This tool is Vivado in our lab.

Vivado:

There are two big companies in the digital circuit and microprocessor industry:

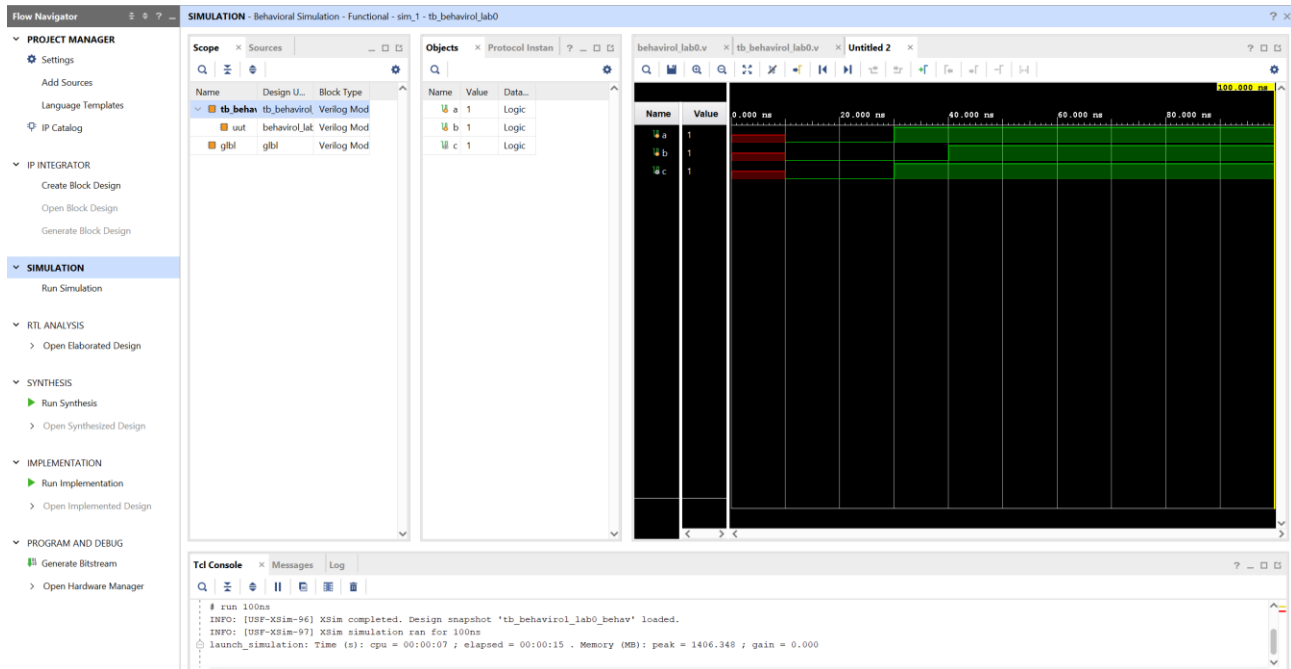


Vivado is the tool to program Xilinx (AMD) FPGAs. *Quartus* is the tool for Altera (Intel) devices.

What is in Vivado?

We are going to focus on the dashboard on the left. There are multiple features:

- Project Manager
- ~~IP Integrator~~
- Simulation
- RTL Analysis
- ~~Synthesis~~
- ~~Implementation~~
- ~~Program and Debug~~



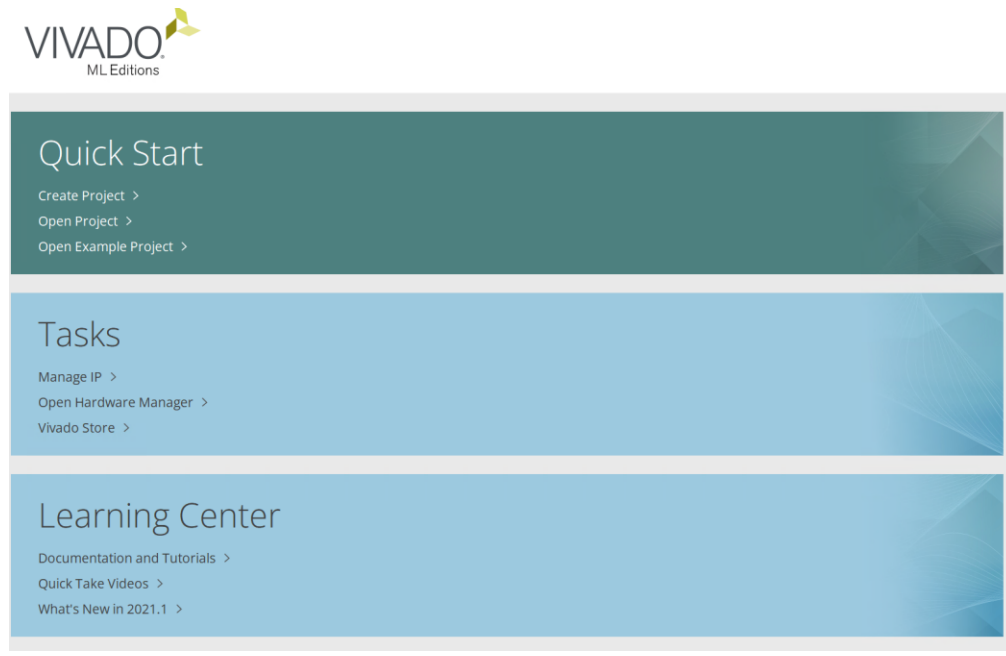
We will use Project Manager to open, add and view file structures. Simulation tab will be used to see wave form output like in the figure above. RTL analysis will give the logic gate result of the design. We can compare handwritten and programmed ones to understand possible mistakes.

Additional:

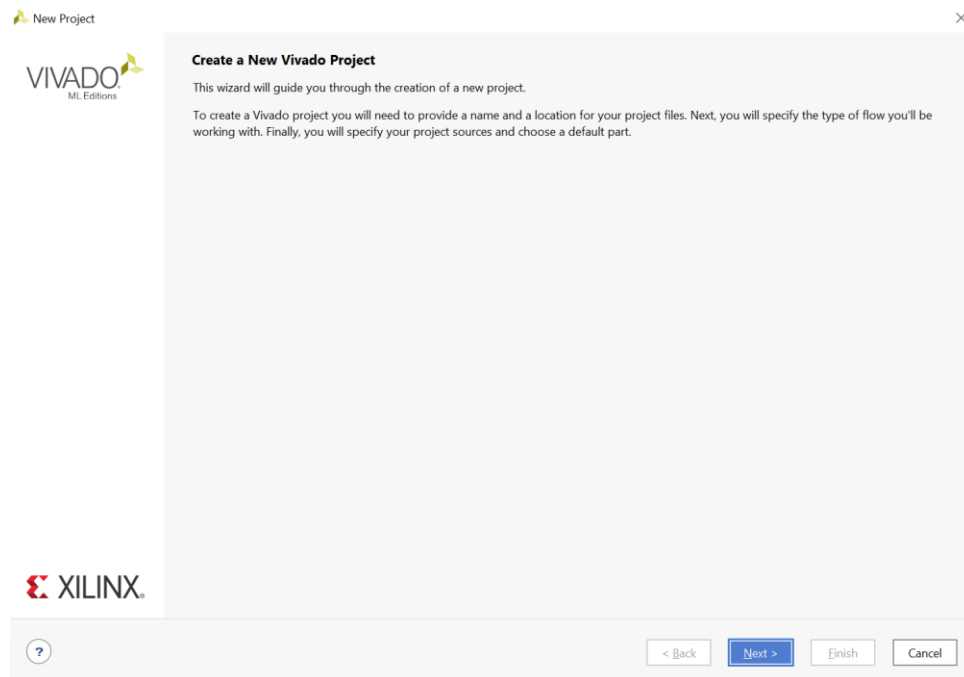
There is an syntax and code examples under Project Manager-> Language Templates. Instead of looking at sources from web browsers, this feature might be helpful to correcting your code and improve your speed.

STEP by STEP Vivado and Verilog

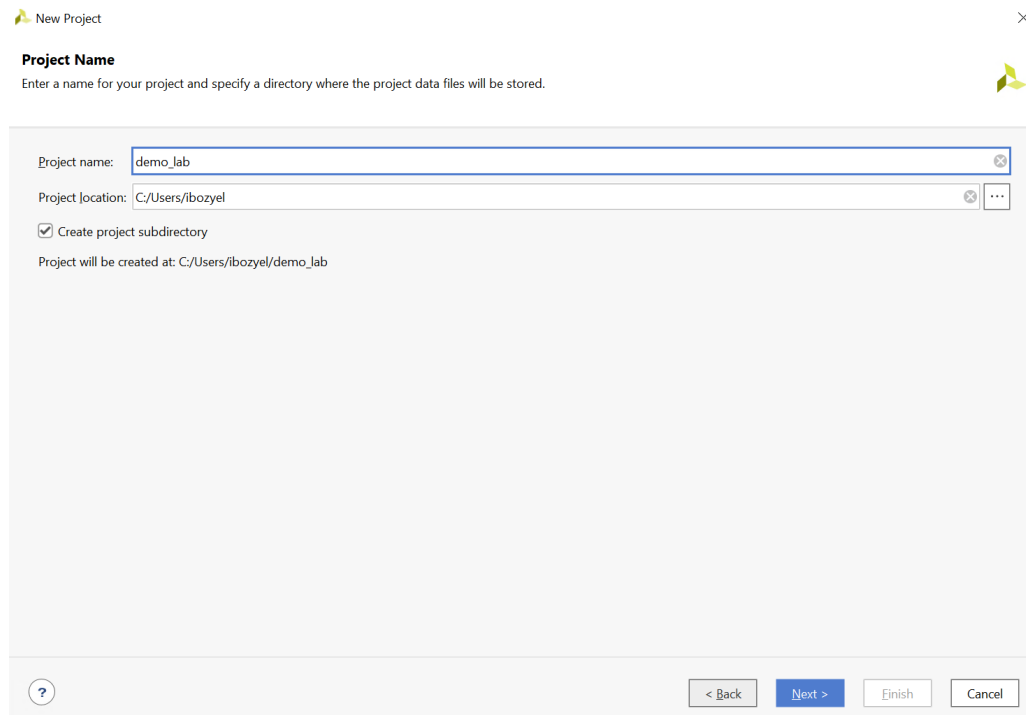
We'll go overstep by step how to open, run very simple logic on the Vivado. We are going to get familiar with syntax.



Choose to create project:



Give a proper name to your project:



New Project

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

Project name:

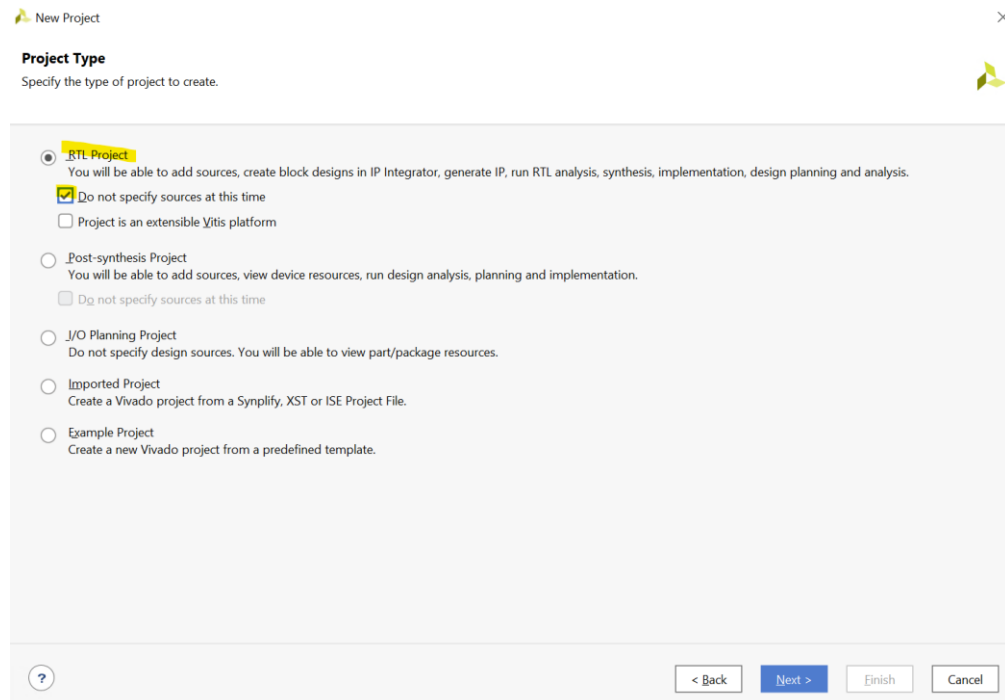
Project location: ...

☒ Create project subdirectory

Project will be created at: C:/Users/ibozyel/demo_lab

? < Back Next > Finish Cancel

Choose RTL Project option with do not specify sources. You can uncheck this option if you want to specify the source files like Verilog and test bench files. For our case we can skip it.



New Project

Project Type
Specify the type of project to create.

☒ **RTL Project**
You will be able to add sources, create block designs in IP Integrator, generate IP, run RTL analysis, synthesis, implementation, design planning and analysis.
☒ Do not specify sources at this time
☐ Project is an extensible Vitis platform

☐ Post-synthesis Project
You will be able to add sources, view device resources, run design analysis, planning and implementation.
☐ Do not specify sources at this time

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

☐ Imported Project
Create a Vivado project from a Synplify, XST or ISE Project File.

☐ Example Project
Create a new Vivado project from a predefined template.

? < Back Next > Finish Cancel

Now we need to choose our targeted device, in this lab we will not update our code to hardware. But simulation tool still requires a target device to create correct gate level synthesis. Therefore, please go “Boards”, and “Refresh” the list. Then you will able to see vendor named “digilentinc.com”. Please choose “Basys3”. This board will be used following classes.

New Project

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

Parts | **Boards**

[Reset All Filters](#)

Vendor: **digilentinc.com** Name: **Basys3** Board Rev: Latest

Search: Q:

Display Name	Preview	Status	Vendor	File Version	Part	I/O Pin Count	Board Rev	Available IOBs	LUT Elements	FlipFlops
Basys3		⊖	digilentinc.com	1.1	xc7a35tcpg236-1	236	C.0	106	20800	41600

Refresh Catalog was last updated on 10/16/2024 12:21:36 PM

< Back Next > Finish Cancel

We are good to go!

New Project

VIVADO
ML Editions

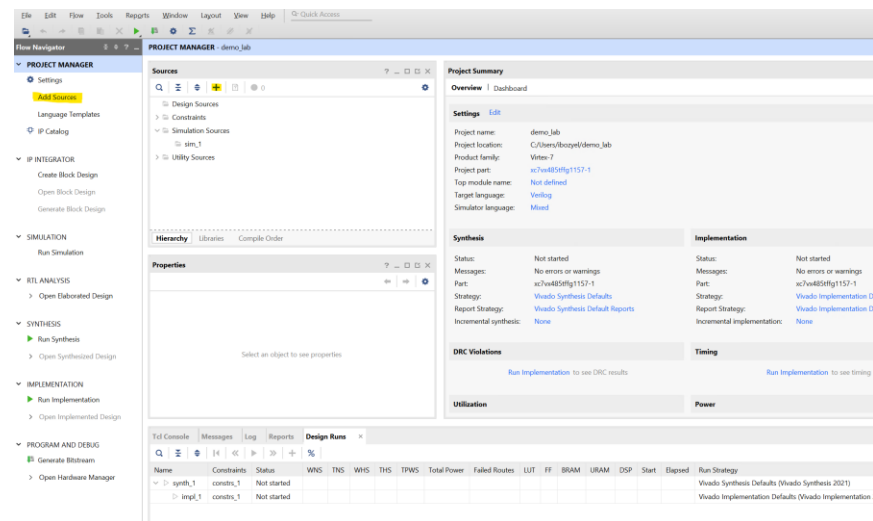
New Project Summary

- 1 A new RTL project named 'demo_lab' will be created.
- 1 The default part and product family for the new project:
Default Part: xc7vx485tffg1157-1
Product: Virtex-7
Family: Virtex-7
Package: ffg1157
Speed Grade: -1

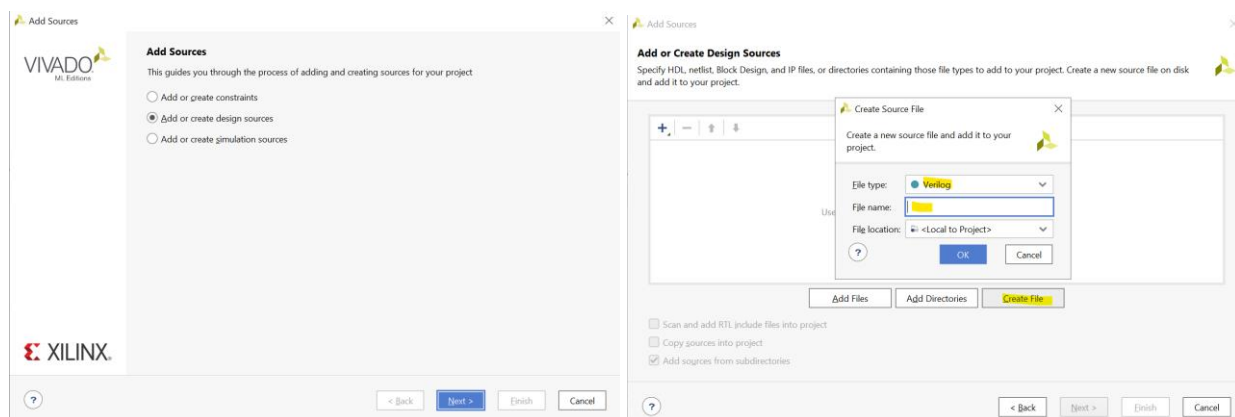
XILINX
To create the project, click Finish

? < Back Next > Finish Cancel

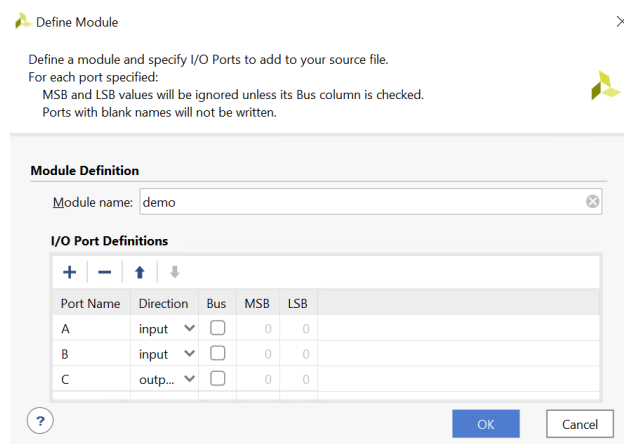
We create a project on Vivado, now we need to create our source code to generate Verilog blocks. You can click both to add source file.



We need a design source first! After hitting next, we will see empty list of designs sources, so we can create a file which is Verilog with a proper name. This name will be our module name, and click Finish.



A new window will pop-up. This screen shows I/O configuration of the module. You can choose port name in here like A,B as input and C as output.



Now we can write our Verilog code in this screen (You can see inputs and output are already defined):

The screenshot shows the Vivado IDE interface. On the left, the Project Manager pane displays the design hierarchy: Sources (demo), Constraints, Simulation Sources (sim_1), and Utility Sources. The Source File Properties pane shows the file 'demo.v' is enabled, located at 'C:/Users/bozyl/demo_lab/demo_lab.srcs/sources_1/new', with a Verilog type and xilinx_defaultlib library. The main editor displays the Verilog code for the 'demo' module, which has inputs A and B, and output C. The logic is defined as 'assign C = A | B;'. The Design Runs table at the bottom shows the synthesis and implementation status for the design.

Name	Constraints	Status	WNS	TNS	WHS	THS	TPWS	Total Power	Failed Routes	LUT	FF	BRAM	URAM	DSP	Start	Elapsed	Run Strategy
synth_1	constrs_1	Not started															Vivado Synthesis Defaults (Vivado Synthesis 2021)
impl_1	constrs_1	Not started															Vivado Implementation Defaults (Vivado Implementation 2021)

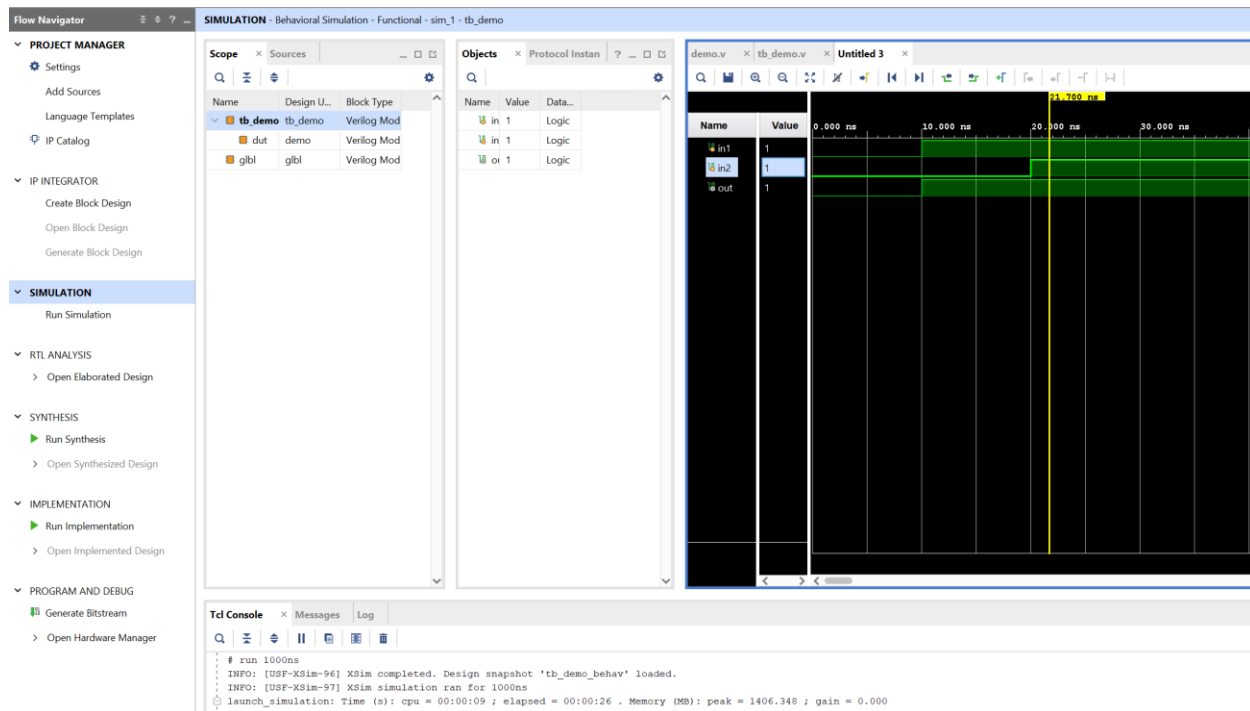
We assign output C “OR” combination of A and B. We need to create a simulation source (test bench). To see the signal output of this block. We will do the same steps add source file but we will choose simulation source. Now do not assign and input and output to this module.

The screenshot shows the Vivado IDE interface. On the left, the Project Manager pane displays the design hierarchy: Sources (demo, tb_demo), Constraints, Simulation Sources (sim_1), and Utility Sources. The Source File Properties pane shows the file 'tb_demo.v' is enabled, located at 'C:/Users/bozyl/demo_lab/demo_lab.srcs/sim_1/new', with a Verilog type and xilinx_defaultlib library. The main editor displays the Verilog code for the 'tb_demo' module, which is a test bench for the 'demo' module. It initializes inputs in1 and in2, and outputs out1 and out2, and calls the 'demo' module. The Design Runs table at the bottom shows the synthesis and implementation status for the design.

Name	Constraints	Status	WNS	TNS	WHS	THS	TPWS	Total Power	Failed Routes	LUT	FF	BRAM	URAM	DSP	Start	Elapsed	Run Strategy
synth_1	constrs_1	Not started															Vivado Synthesis Defaults (Vivado Synthesis 2021)
impl_1	constrs_1	Not started															Vivado Implementation Defaults (Vivado Implementation 2021)

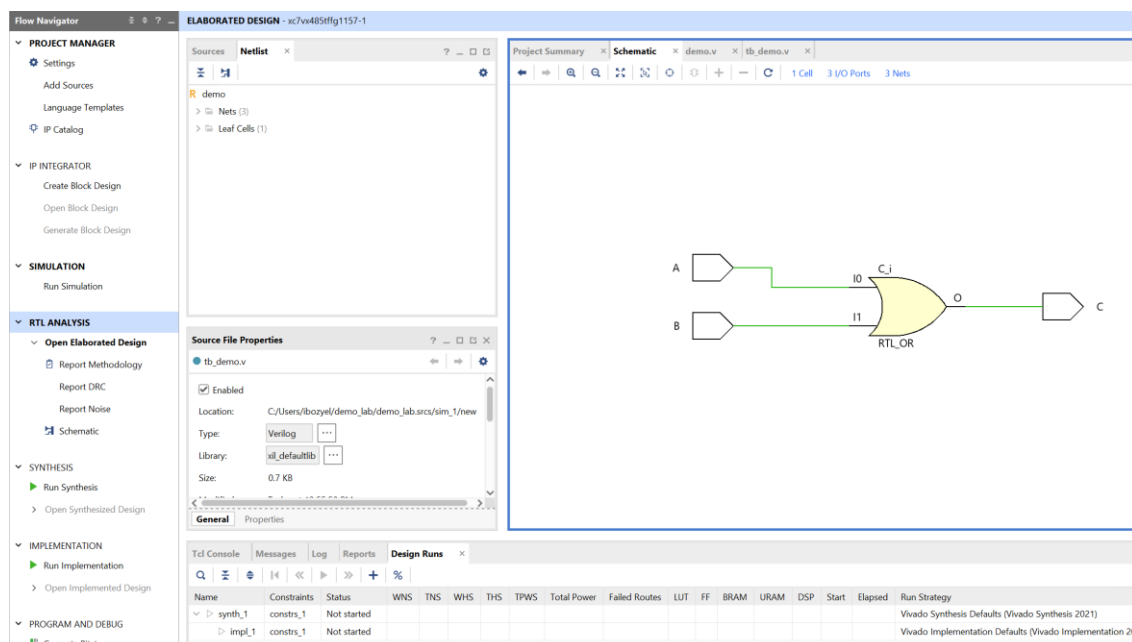
We create a test bench that includes demo module as submodule, dut (device under test) is given name for called submodule you can write anything there apple, tree, dut, subdemo etc. Then we need to initialize our

signal connections then we can begin to assign different values to inputs. Each step has 10ns waiting time before next case. Now we need to run simulation to see the output waveforms.



We can see that if one input is 1, output is 1. The output is 0 when both inputs are 0, which is characteristic of an OR gate. Then we can run RTL result to see our design on gate level.

Here is the result:



We programmed an OR gate which has A and B as input, C as output.