

# Lab 02: Simulating with Vivado

---

In this lab, we use a simple 4-bit adder design, to have a first look at simulation in Vivado.

## Lab Goals

- Run the vivado simulation
- Run the Vivado Waveform viewer
- Adding breakpoints

## The design

Port	Direction	Width
A	IN	4
B	IN	4
RESULT	OUT	5
ENABLE	IN	1

The design takes as input two signals **A** and **B** and sums them in the **RESULT** signal, if the **ENABLE** port is set high. Otherwise, it returns zero.

## Test bench

The provided test bench instantiates the adder and stimulates its inputs.

## 1. Starting the Exercise

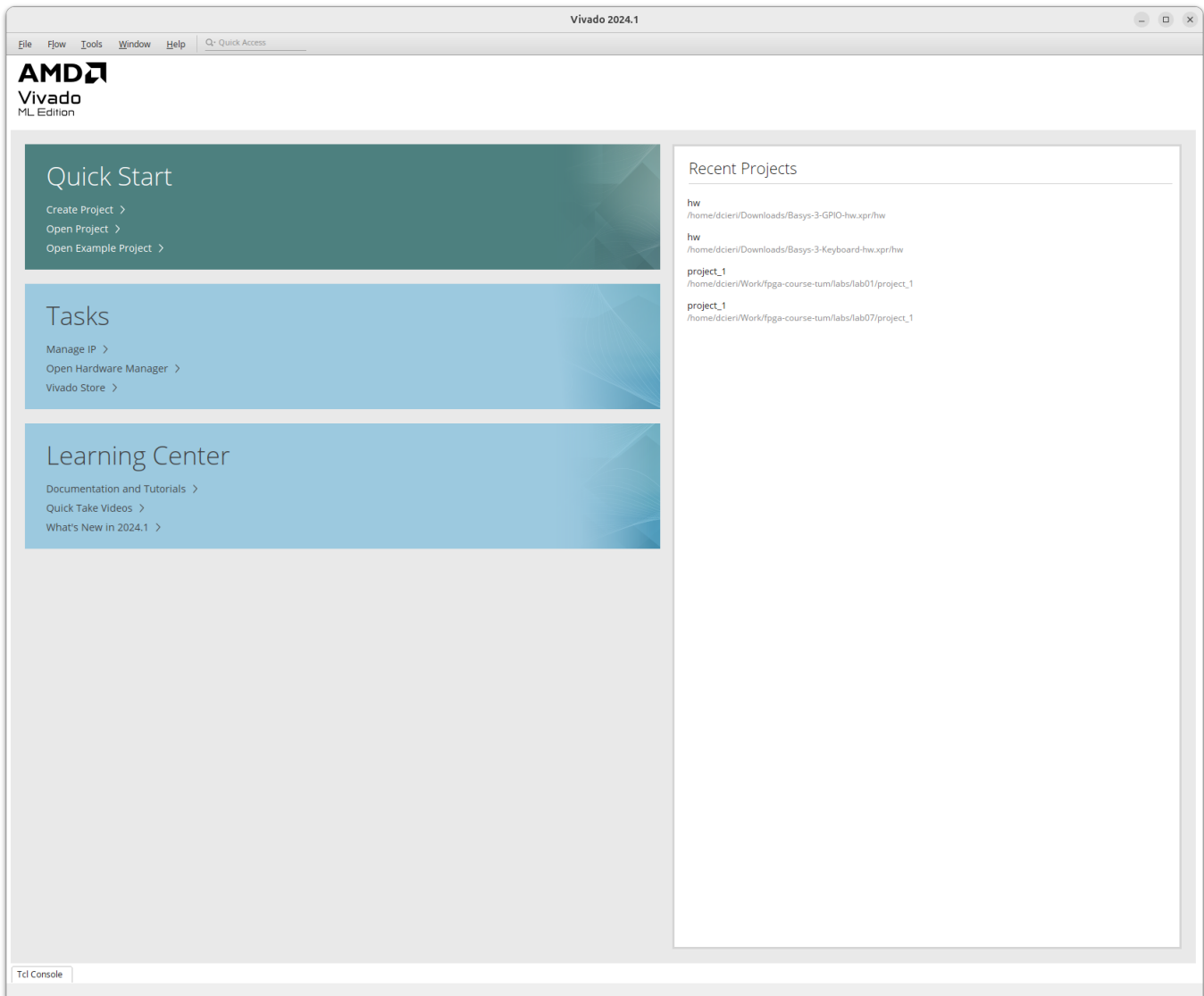
### a. Create the Vivado project

Open a new terminal and go to the **lab02** folder.

```
cd ~/labs/lab02/
```

Open Vivado and click on Create Project

```
vivado &
```

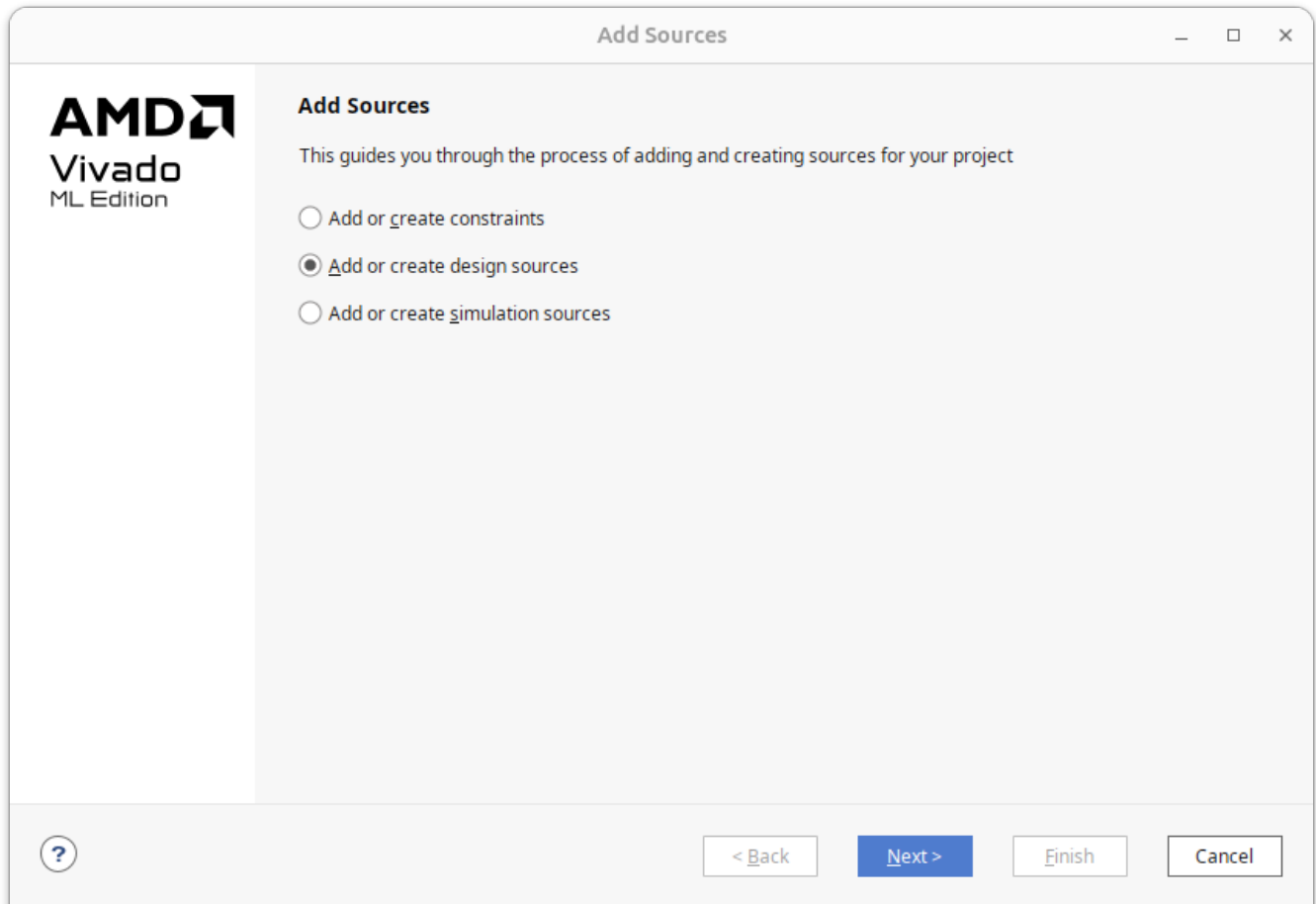


Follow the Wizard to create the project using the following information.

- Project Name:
  - Project name: `adder_prj`
  - Project Location: `~/labs/lab02/`
  - Create project subdirectory: `Yes\`
- Project Type
  - RTL Project
- Add Sources
  - Click on `Add Files` and select `~/labs/lab02/src/adder.vhd`
- Add Constraints (skip)
- Default Part
  - Click on `Boards`, and select `Basys3` (use the search function)

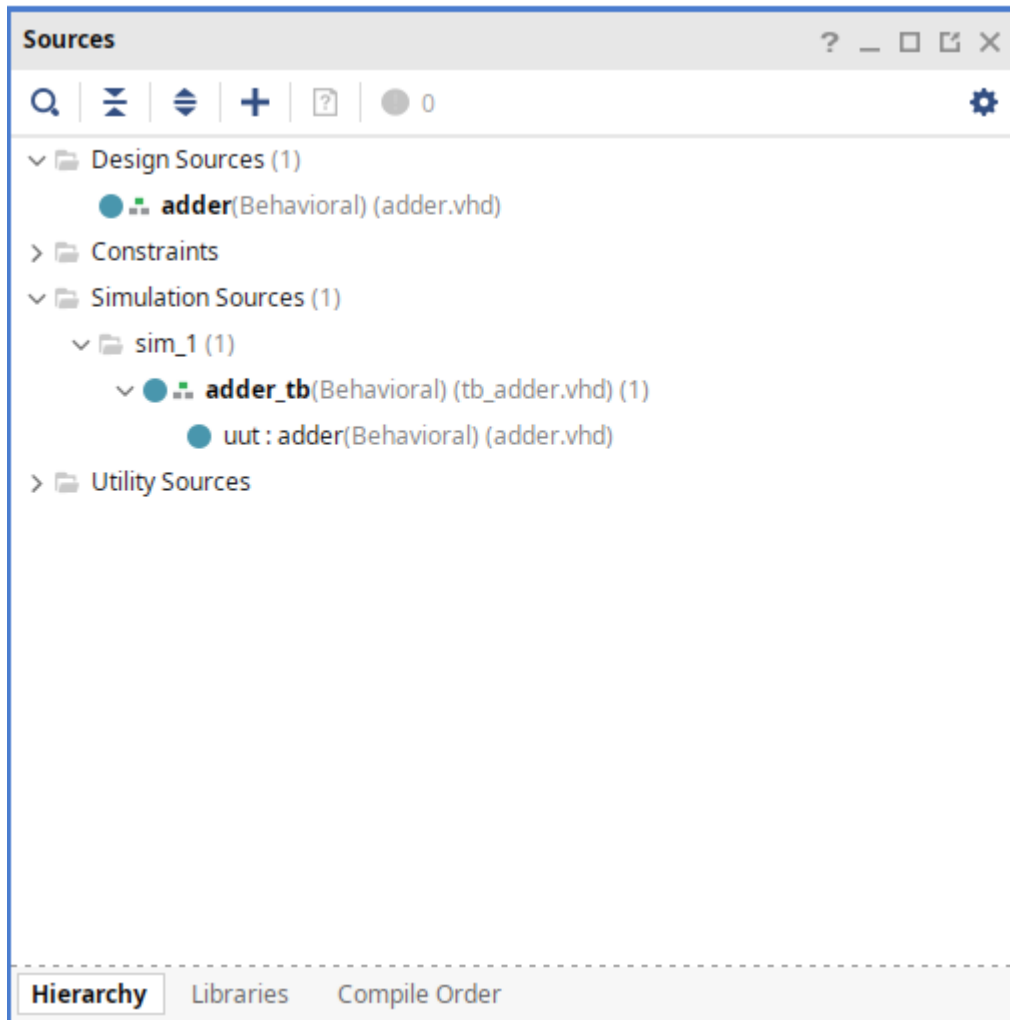
## b. Add Simulation Sources

Click on `Add Sources` on the left sidebar, and select `Add or create simulation sources` in the pop-up window.

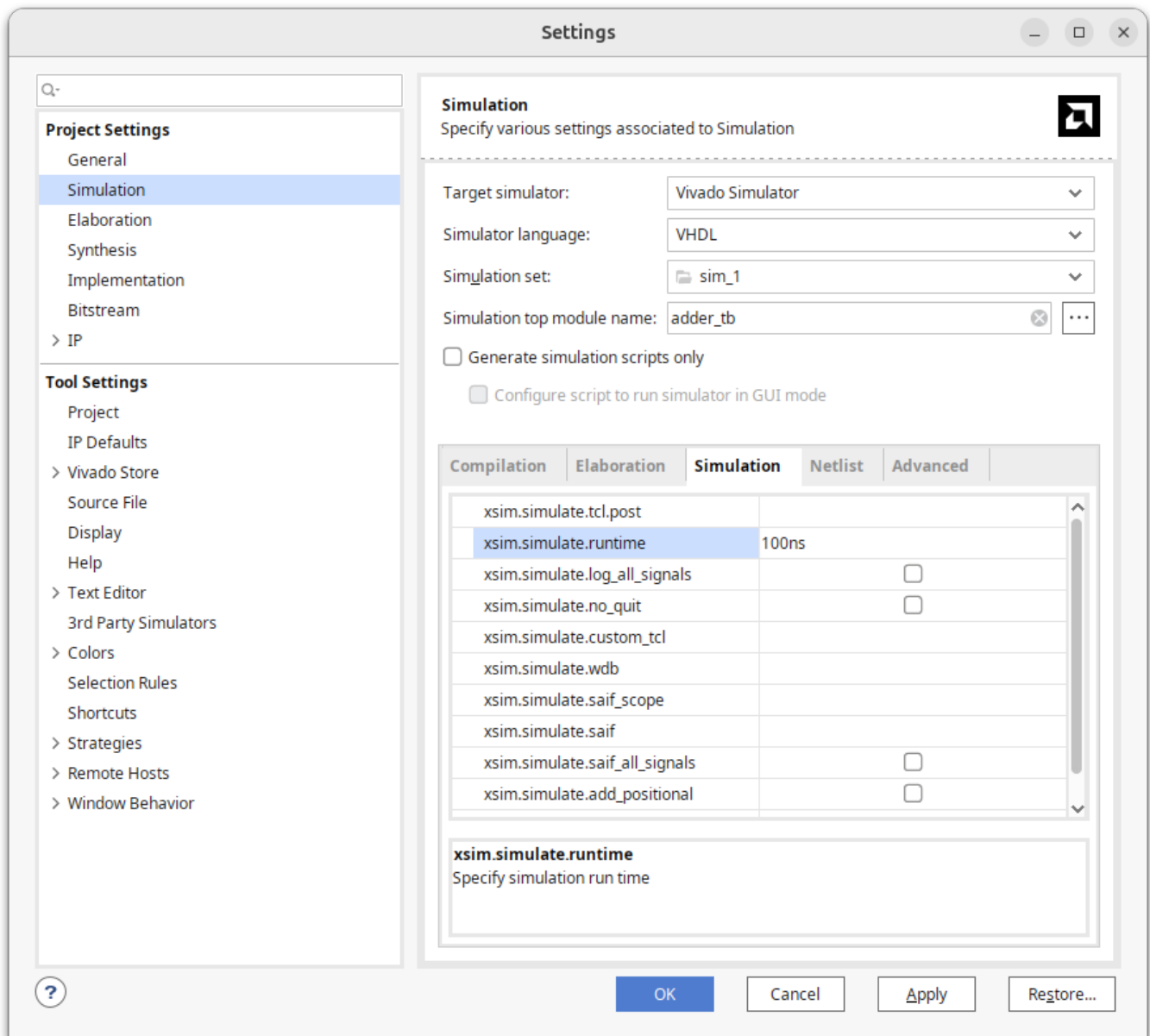


Click on **Add Files** and select `~/labs/lab02/src/tb_adder.vhd`. Click on **Finish**.

You should now see in the **Sources** panel, in the **Simulation Sources** the `adder_tb` testbench.



Now right click on `Run Simulation` on the sidebar and select `Simulation Settings`.

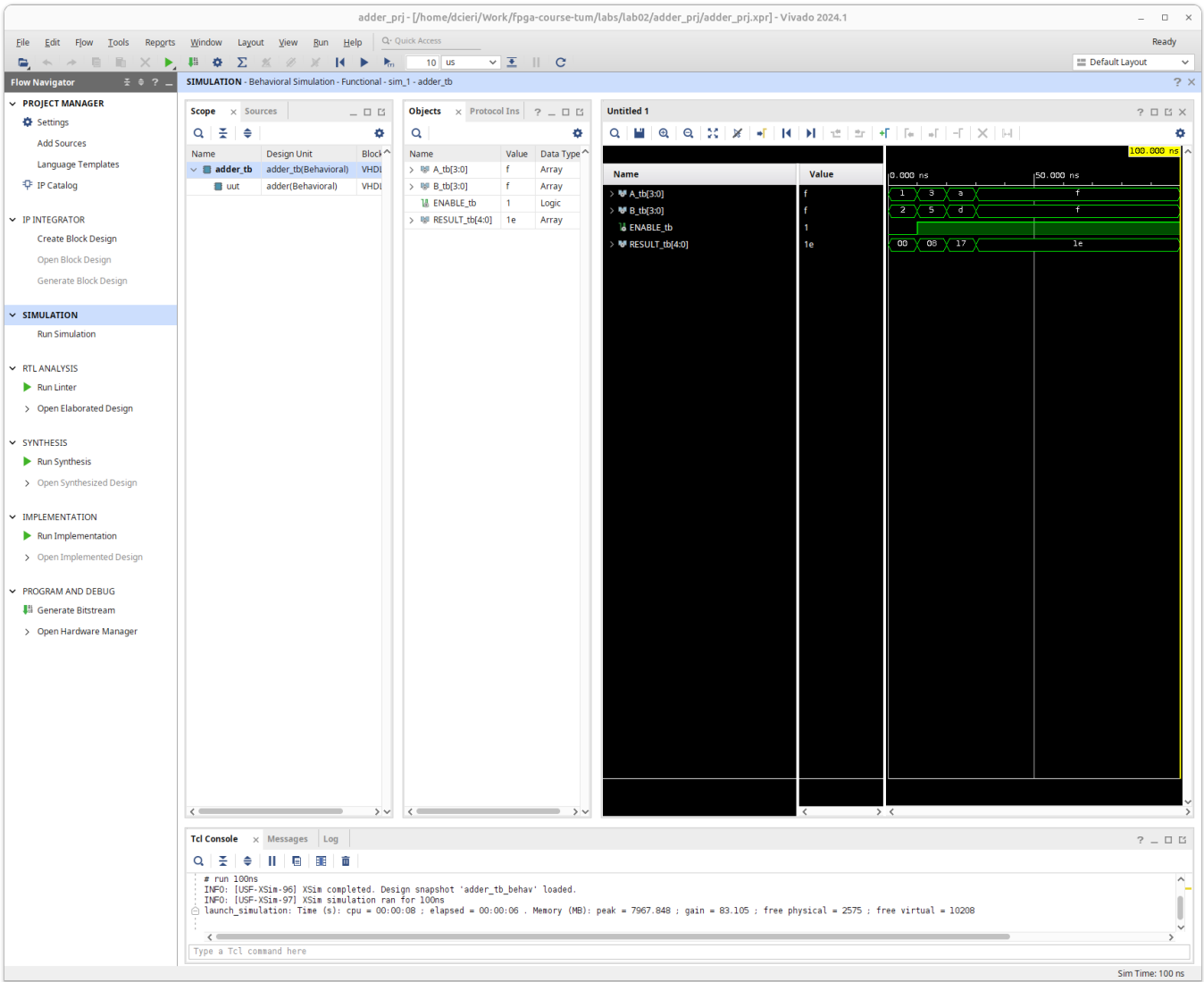


In the settings window, set the simulator language to `VHDL` and go to the `Simulation` tab in the bottom panel. Set the simulation runtime `xsim.simulate.runtime` to `100ns`. Click `OK` to save the settings.

Click now on `Run Simulation -> Run Behavioural Simulation`.

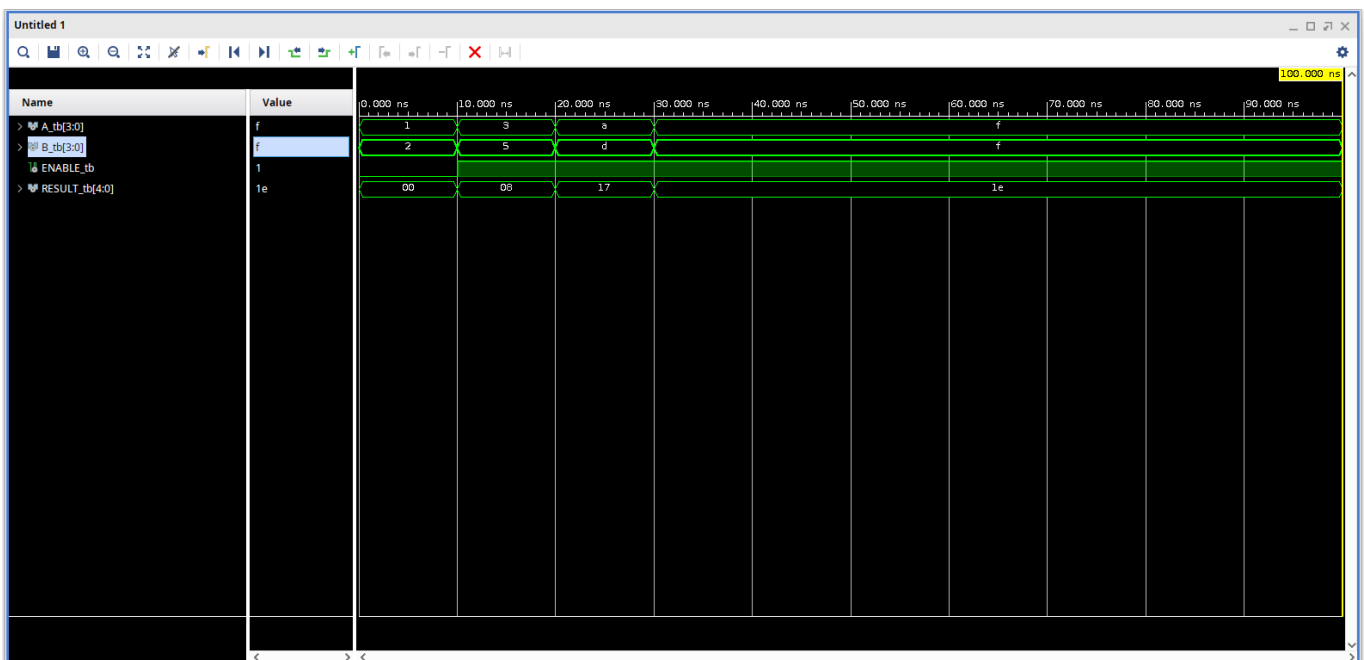
This compiles the code using the vivado simulation compiler (`xelab`) and launches the simulation using the Vivado simulator (`xsim`).

This opens a new Vivado context (`Simulation`), with new panels (Scope, Objects and Waveform). On the bottom panel, you have the Tcl console, where you can read the commands actually executed by Vivado (in blue) and their output (in black).



c. Exploring the waveform

The waveform window shows the evolution of the signals as a function of the simulation time.



Click on the three zoom buttons (zoom in, out and fit) and see how the view changes.

Then select the `A_tb` signal and click on the previous and next transitions buttons, to see how the time cursor moves.

If you select multiple signals, the cursor will move to any transitions of them.

You can also add markers to the waveform, to mark certain points in the simulation.

Add two Markers, one at 10 ns and one at 50 ns.

Go now to time 0, by clicking on the corresponding icon. You can click on the Next Marker icon, to jump from one Marker to the other. Try also to jump back to the first marker from the second.

Delete all Markers click, either by selecting individually and using the Delete key, or clicking on Delete All marker icon.

### Add new signals to waveworm viewer

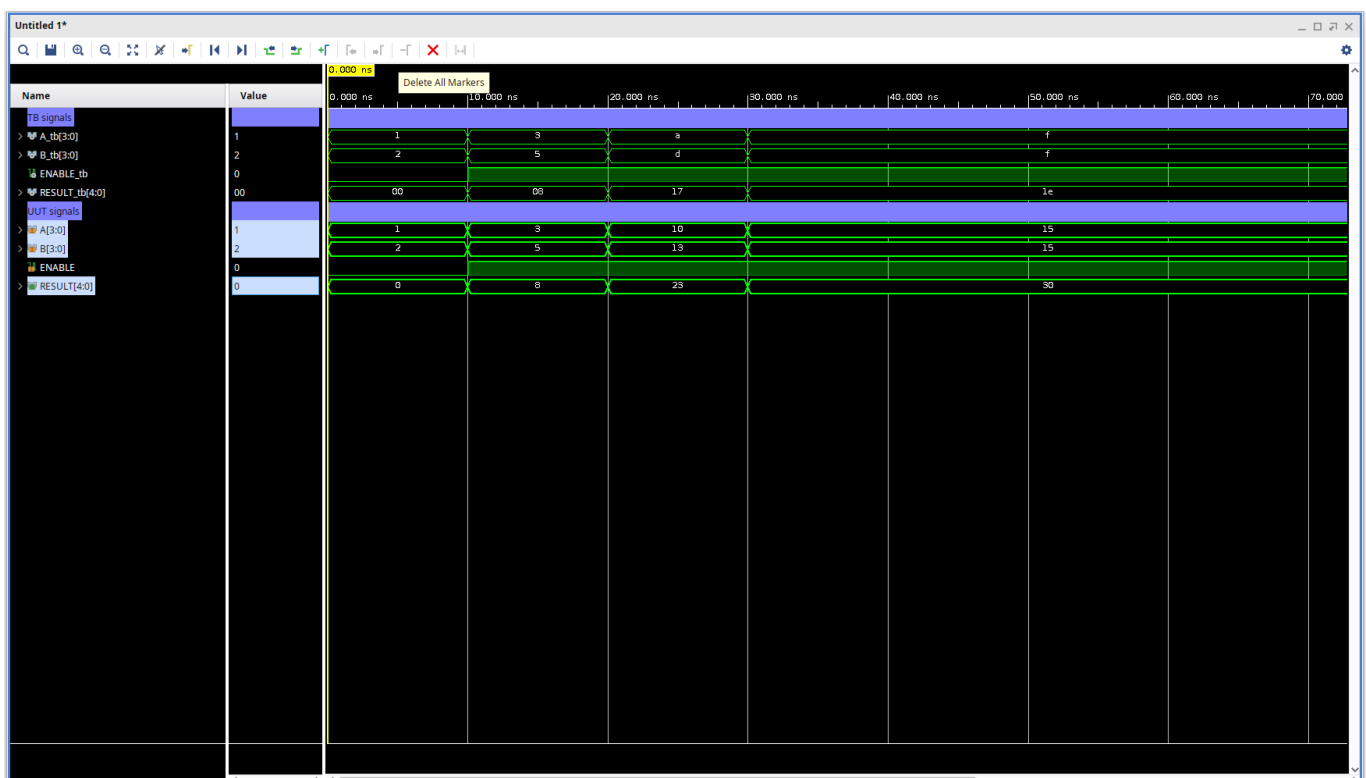
By default, Vivado shows only the signals of the top level testbench `tb_adder`. Let's add some other signals.

First add a divider panel in the window by right clicking in the wave window and selecting `new divider`. Call it `TB signals` and move it to the top of the window. Create now a new divider and call it `UUT signals`.

In the scope panel select `uut`. In the `Objects` panel, you should now see the internal signals of the `adder` module. Select all of them, right click and `Add to Wave Window`.

By default, Vivado prints the value of vector signals in hex format. You can however change this to Signed, Unsigned decimals or binary as well.

Select the `A`, `B` and `RESULT` signals in the wave form, right click and select `Radix->Unsigned Decimals`.



## Grouping Signals

Signals can also be grouped to improve clarity. Select the **A** and **B** signals, right click and select **New Group**, naming it **Input Data**.

### e. Relaunching the simulation

Every time you make some change to your design or your testbench, you have to recompile the code.

Open the **Sources** Tab in the left panel and double click on the `adder_tb`. Change the last value of `B_tb` to `1010`, and save.

Go back to the Waveform **Untitled 1** and relaunch the simulation, by clicking on the corresponding icon




. Check the changes in the waveform.

### f. Stepping through the code.

Sometimes is useful to run the simulation one step at the time to understand the behaviour of your design.

To do so, first reset the simulation clicking on the corresponding icon



Click now on the step button , and see how the simulation executes different part of the code. In the bottom panel, you can also check the log in the Tcl console.

Also check how the waves are updated in the waveform viewer.

### g. Adding break points

You can also add breakpoints to stop the simulation once reaching a certain line in the code. Let's add one when the **ENABLE** signals goes from 0 to 1.

Open the `tb_adder.vhd` and go to line 34. To add a breakpoint at this line, simply click on the red circle at the beginning of the line. The circle should become full.

Reset now the simulation and click on **Run All** . The simulation should now stop, once the breakpoint is reached.

## 2. Launching the simulation from batch

To simulate your design you don't need to create a Vivado project. You can directly use `xvhd1`, `xelab` and `xsim` commands.

Open a terminal and go the `~/labs/lab02` folder.

```
cd ~/labs/lab02
```

Now you need to compile the vhd files in the project using `xvhd1`



```
xvhdl src/adder.vhd
xvhdl src/tb_adder.vhd
```

Now you can elaborate the design

```
xelab adder_tb -s adder_tb_sim --debug typical
```

The `debug` flag enables specific debugging ability. When setted to `typical`, it enables line breakpoints, waveform generation and signal driver value probing.

Finally you can run the simulation, either with the GUI.

```
xsim adder_tb_sim -gui
```

or in batch mode

```
xsim adder_tb_sim -R
```

Explore all the functionalities of the Vivado simulator commands by having a look at the documentation page.

<https://docs.amd.com/r/en-US/ug900-vivado-logic-simulation/Running-the-Vivado-Simulator-in-Batch-Mode>

### 3. Simulating with GHDL

GHDL is an open-source VHDL simulator, which can be used alternatively to the Vivado Simulator. GHDL does not provide an integrated waveform viewer, but it can produce waveform files that can be then read using other tools, like `gtkwave`.

Let's try to simulate our `adder` design using GHDL. Again open a new terminal and go the the `lab02` folder.

```
cd ~/labs/lab02
```

First you need again to compile the VHDL files.

```
ghdl -a -fsynopsys src/adder.vhd
ghdl -a -fsynopsys src/tb_adder.vhd
```

The `-fsynopsys` flag is required, because the design is using some libraries GHDL needs to import.

Now we have to elaborate the design.

```
ghdl -e -fsynopsys adder_tb
```

Finally, we can run the simulation

```
ghdl -r -fsynopsys adder_tb --wave=addder.ghw
```

The waveform is saved in the `adder.ghw` file, that you can open now with `gtkwave`

```
gtkwave adder.ghw
```

Try now to explore the functionalities of `gtkwave` to analyse the results, by replicating some of the steps done in Section 1.

Useful links:

- GHDL documentation: <https://ghdl.github.io/ghdl/index.html>
- GtkWave documentation: <https://gtkwave.sourceforge.net/gtkwave.pdf>