Digital Design Lab (EDA322)

Lab 0 (Introductory Lab)

Examiner: Prof. Ioannis Sourdis
TAs: Ahsen Ejaz, Ghaith Abou Dan, Fredrik Jansson, Konstantinos Sotiropoulos, Magnus Östgren, Neethu Bal Mallya, Panagiotis Strikos

Last Update: 2024-01-15

1 Introduction

1.1 Preparation

- Complete the coding tutorial: *Coding_ 1_ VHDL_ basics*.
- If you are going to use your personal computer for the lab sessions, download and install ModelSim and Xilinx Vivado (only for Lab 5). Refer to the respective installation manuals on Canvas
- Download `tutorial_lab.vhd` and `tutorial_lab.do` files from Canvas (*Files* > *Labs* > *Lab 0* > *lab0_files*) to a local folder so that you can access them from within ModelSim

1.2 Learning Outcome

1. Learn to recognize and understand the use of several commonly occurring digital logic components
2. Get introduced to designing digital logic with VHDL language and modern CAD software that will be used in the lab sessions of the course
3. Become familiar with ModelSim

1.3 Required Tools

In this lab, you will use the following tool:

1. **ModelSim/ QuestaSim**: ModelSim and QuestaSim are tools created by Mentor Graphics for simulating descriptions of digital circuits in hardware description languages such as VHDL. Although QuestaSim is the commercial flagship product, for this project ModelSim is sufficiently capable and comes with the added benefit of a student license. Therefore, for simulating your VHDL designs on your personal computer, you can use ModelSim, while on the lab computers, you will be using QuestaSim.

2 Tasks

This lab will require you to complete the following task:

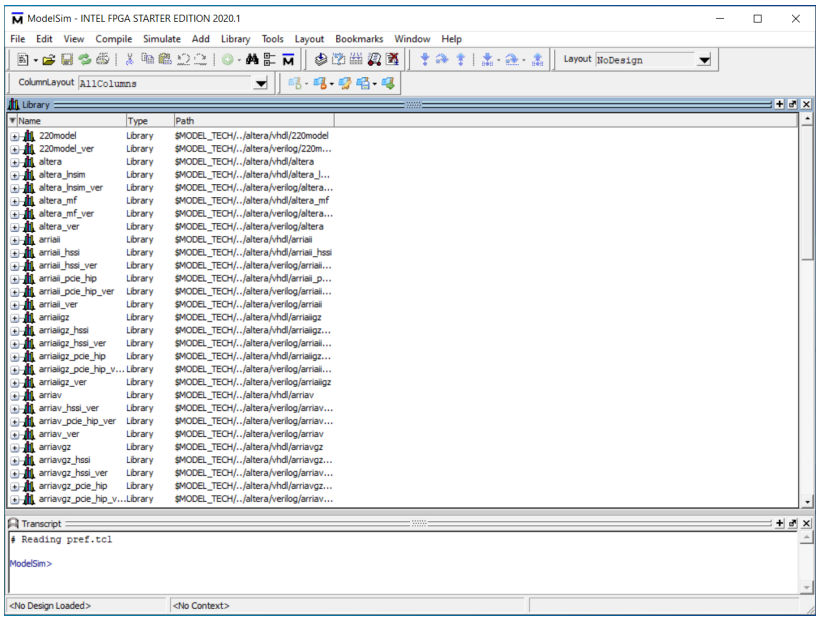
1. Simulate a sample VHDL design in ModelSim/ QuestaSim and verify the correctness of the operation using a *do file*.

To complete the above task, follow the steps given in the following subsections.

2.1 Task 1: Simulating VHDL designs using ModelSim/ QuestaSim

***Note**: We have used ModelSim (Intel FPGA Edition 2020.1) for the screenshots in this section. However, the steps to follow are the same for QuestaSim and other versions of ModelSim.*

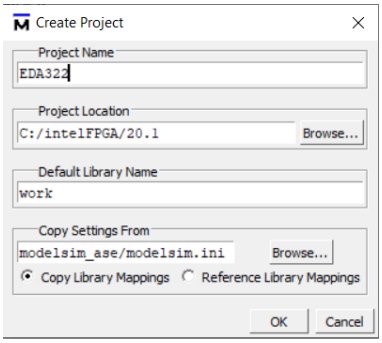
(Step 1) Start ModelSim. You should see a window similar to the one shown below.



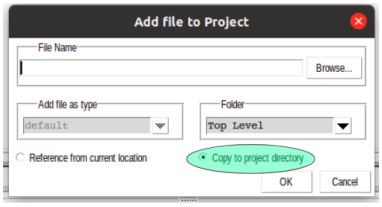
(Step 2) **Creating a new project**

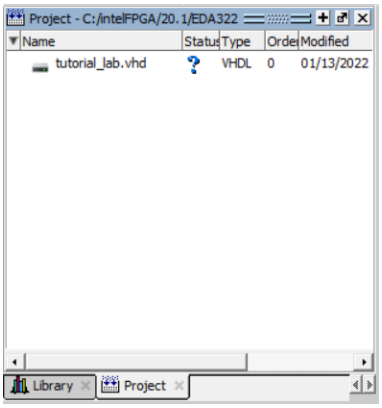
Select *File* → *New* → *Project*. This opens the *Create Project* dialog where you can specify a project name, location and default library name.

- Enter *Project Name*, say ***EDA322***
- Choose a suitable *Project Location* so that your code is maintained in a secure place. If you are using one of the lab's machines, please make sure that the project's folder and files are stored in your user's **private space**, and not in a public folder that other students might have access to.
- You can generally leave the *Default Library Name* set to ***work***.



(Step 3) **Adding files to the project:** After selecting *"OK"* in *Create Project* dialog, the software will open a dialog, *Add items to the project*. You can select *"Add existing file"* and add the VHDL file (with the extension `.vhd` or `.vhdl`) to the project. Make sure you select *"Copy to project directory"*. You can add files to the project at a later point in time as well. When you select *"OK"*, the file(s) is added to the *Project* tab.



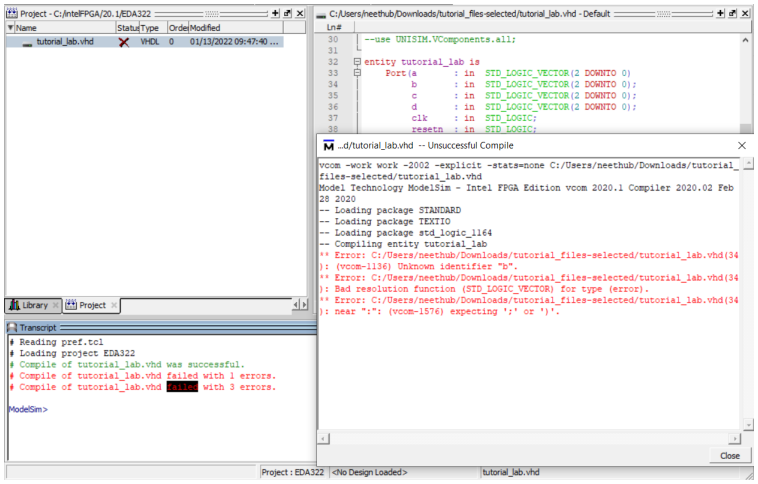


The question mark in the *Status* column in the *Project* tab denote either the files haven't been compiled into the project or the source has changed since the last compile.

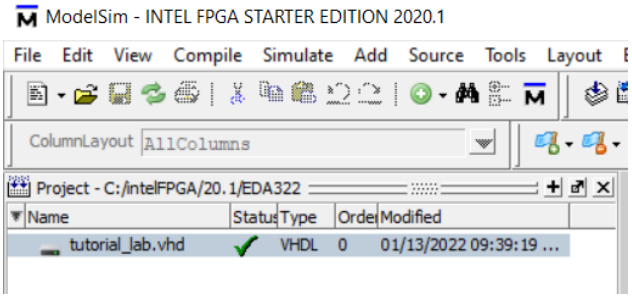
(Step 4) **Editing the files:** Double-clicking the file name will open the file for editing in an editing window. You can now look at how the description of a simple hardware component looks like in VHDL.

(Step 5) **Compiling the files:** To compile the files, select *Compile* → *Compile All* from the menu or right click in the *Project* tab and select *Compile* → *Compile All*.

If there are errors or warnings, you can see them by double-clicking on the error message (appeared in red) in the *Transcript* window. A new window appears where the errors/warnings are listed, like in the figure below. Using this information, you can fix them and recompile them.



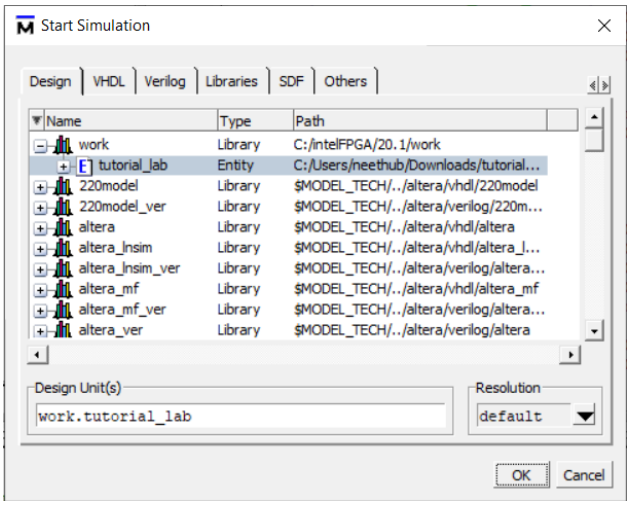
When compilation is successful, you should be able to see a green tick mark (✓) in the status field beside the file name.



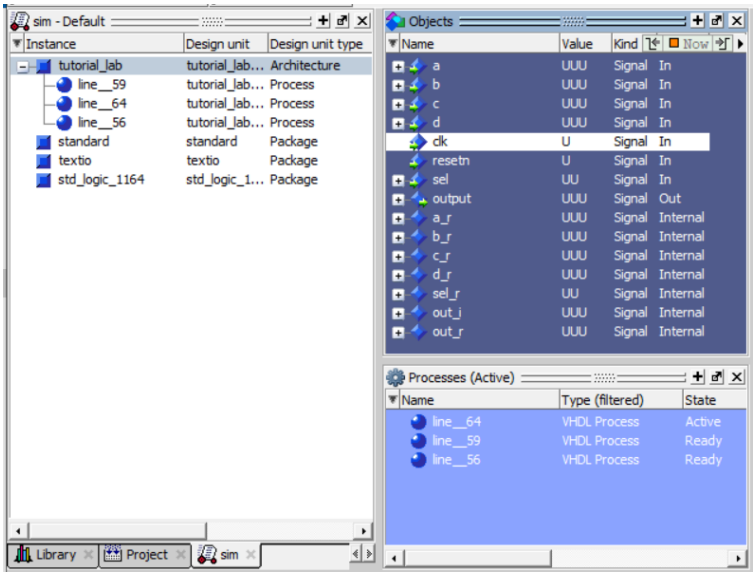
(Step 6) **Simulating a design**

Select *Simulate* → *Start Simulation* to open the *Start Simulation* dialog.

Select a *Design Unit (s)* that you want to simulate in the *Design* tab. In the ***work*** library, choose the name of the entity (or the top level entity if you have several entity descriptions comprising your design). In this tutorial, the entity name `tutorial_lab` is the top level entity in your default ***work*** library.



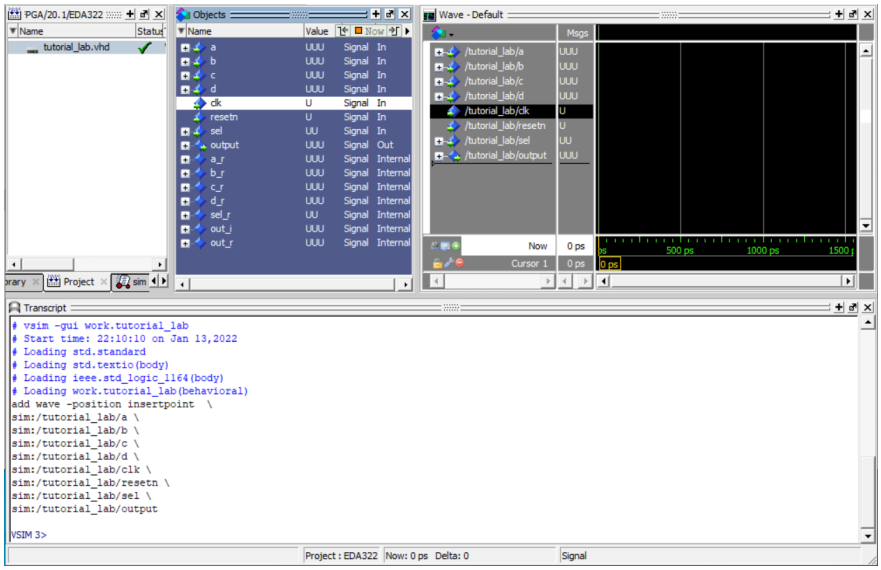
After you click *"OK"*, new windows, named *sim* and *Objects*, appear that show the structure/hierarchy of the design and signals in the simulated entity, respectively.



At this point, you are ready to run the simulation and analyze your results. You often do this by adding signals to the *Wave* window and running the simulation for a given period.

(Step 7) **Adding objects to the *Wave* window**

The *Wave* window allows you to view your simulation results as waveforms and their values. To add the signal(s) to the *Wave* window, you can right-click on signal(s) in the *Objects* window and select *"Add Wave"*. Alternatively, you can drag and drop the signal(s) in the *Objects* window to the waveform area.

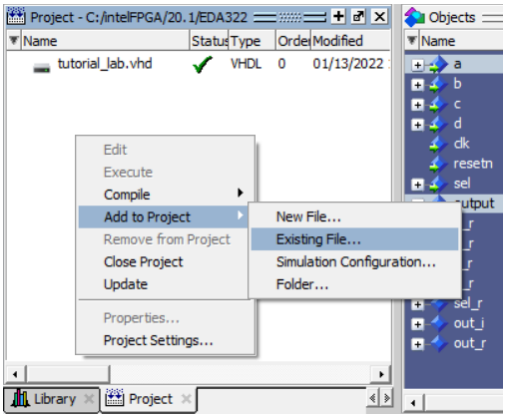


Also, notice the *Transcript* window at the bottom of the ModelSim window, which also displays a command line. The `add wave` command is reflected in the *Transcript* window when you add objects into the *Wave* window.

(Step 8) **Verification of the design**

Now, we have to ensure that the design meets the timing and functionality requirements. Typically, testbenches are used to simulate and analyze the designs. However in this tutorial, we will use a *do file* to verify your design. We will use testbenches for design verification in the upcoming labs.

Before we proceed, make sure that the provided *do file* is placed in the working directory for the project. You can add the given `tutorial_lab.do` to your project as shown below.

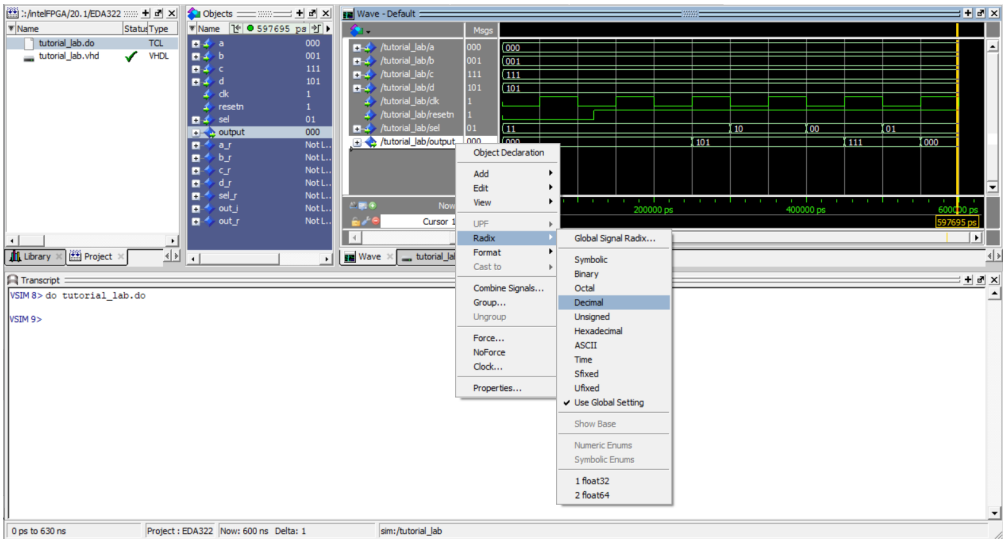


In the *Transcript* window, execute the below command to run the *do file* you have downloaded.

```
do tutorial_lab.do
```

When the command is executed, you can see that values in the *Wave* window are updated.

The figure below shows various signal transitions when the VHDL module is simulated. You can zoom in/out using the *Zoom* toolbar buttons. You can also change the *radix* in which the signal(s) appear by right-clicking on a signal(s) and selecting radix. A signal can be added several times using different radixes (can be done through the graphic interface or inside the *do file*), which can be practical.



Notes: ***DO files*** are used to control simulation in ModelSim. The *do file* contains a sequence of commands that can add signals to the *Wave* window (`add wave`), provide stimulus to those signals (`force`), advance the simulation (`run`), restart the simulation time (`restart -f`), etc. You can also run these commands directly in the command line. However, a *do file* facilitates quick repeated simulation when debugging. So it is highly recommended to practice writing *do file* for your lab demonstrations.

Take some time at this point to familiarize yourselves with the syntax of the *.do* file and with the features available in the simulator – like adding/removing signals from the waveform window, zooming in and out, cursor control, changing the way values are represented in the window (radix), etc. A quick guide (*ModelSimQuickGuide.pdf*) and a command reference (*ModelSimCommandsReference.pdf*) have been

provided for your guidance on Canvas. Getting familiar with these features will help you test and debug code more effectively – a skill that will be valuable throughout the labs.

Try to answer the following questions:

1. What functionality is modeled in the code?
2. How many flip-flops have been used in the code?
3. What happens when the `resetn` port is set to 0?

This task showed you how to create your own project in ModelSim, add files to your project, compile your source files, start your simulation, and view your waveforms. You're ready to explore all of the examples from the coding tutorials and run the simulations for yourself.

3 Hints and Tips

- **Forcing values in the *.do* file**

  Depending on the version of the ModelSim you use, you can utilize different formats for assigning a value to a signal. But, it is strongly recommended that you use the generally accepted explicit format where # symbol should be used to define the radix of a value. For instance:

  - `force mySignal 2#101` assigns binary value **101** is assigned to `mySignal`
  - `force mySignal 10#240` assigns decimal value **240** is assigned to `mySignal`