Exercise 1:

a) Combinational Logic

In the first exercise, you will learn to design, simulate and verify a VHDL module with standard text editors and the widely used tool ModelSim from Mentor Graphics. As a first example, you should develop a simple module which realises the logic function $a \wedge \bar{b}$.

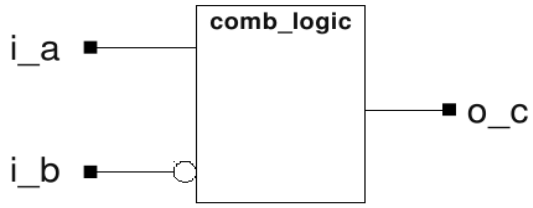

Figure 1: comb_logic

Port declaration:

- *i_a* : in std_logic
- *i_b* : in std_logic
- *o_c* : out std_logic

To write the VHDL-Code, you should use GNU emacs as your default editor. Emacs is a powerful editor with a steep learning curve, but it is a powerful tool for the development of VHDL code because of the many possibilities it offers.

Once you have written your code, you must simulate it to verify its functional behaviour. An important point you must consider with hardware description languages is the difference between simulatable and synthesizable code. Synthesizable code is the code you can implement in hardware. This code is only a small subset of VHDL. You cannot e.g. synthesize timing-conditions like “wait for 10ns”, since hardware such as a simple CMOS transistor does not understand any timings, but can only switch on and off. Nevertheless, VHDL offers such statements, along with other constructs, to simplify the simulation your modules.

Now you can start ModelSim. In order to do this properly, you must setup the ModelSim program as follows. Because of the files ModelSim creates, please be sure to start the program from the correct directory.

```
$ setup-lm modelsim 10.1b
```

The following message appears:

```
Setup for ModelTech ModelSim SE 10.1b ok.
Info: Doc: <mold>
     Start: <vsim>
```

Now you can start ModelSim with:

```
$ vsim &
```

There are several ways to work with ModelSim, two of them will be explained in this lab. The obvious way to work with the tool is with the help of the GUI. Another more common approach in EDA is the use of scripts written in TCL (Tool Command Language).

We will start using ModelSim by setting up a project with the help of the Project Navigator inside the GUI.

Working with the Project Navigator

First of all, you must create a new project as shown in figure 2.

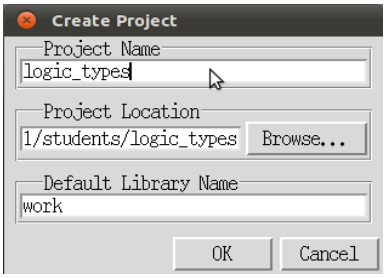

Figure 2: Creating a project

In the next window, you have the possibility to add existing files to your project. Here you add your *.vhd file combinational_logic.vhd, which now exists in your workspace, as you can see in the figure below. In the next step, you will compile your module.

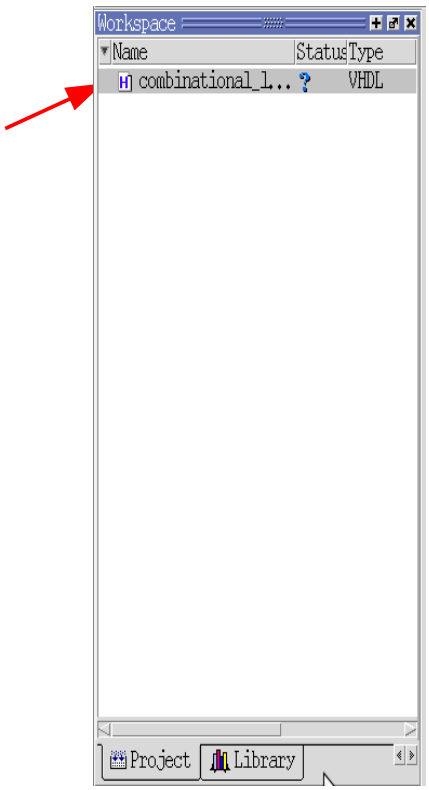

Figure 3: Modules in workspace

Figure 4: Libraries in workspace

You can compile your module by right-clicking on it in your project workspace. If the syntax of your module is correct, ModelSim will compile the module to the default library **work**. Otherwise, ModelSim shows you syntactic mistakes and the lines where they occur. If necessay, you should correct these mistakes step by step.
After a successful compilation, you know that your module has a correct syntax, but you don't know if it works correctly. To simulate the function of your module, you must add the **testbench**, a

special type of file that feeds your module with stimuli, to your project.

After compiling your testbench to the work library, you must add a simulation configuration to the project. Here you choose the testbench as design unit.

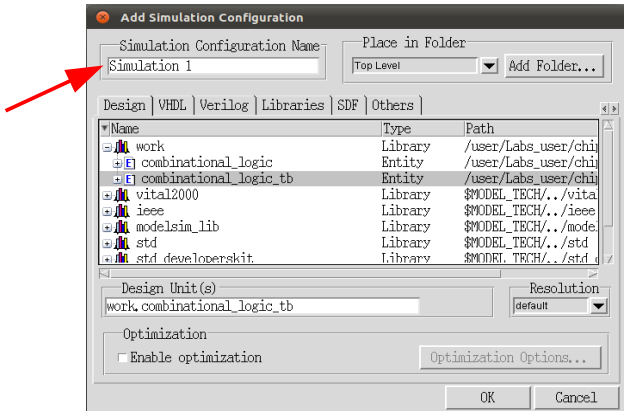

Figure 5: Adding a simulation configuration

By double-clicking on the simulation configuration, you start the simulation environment.

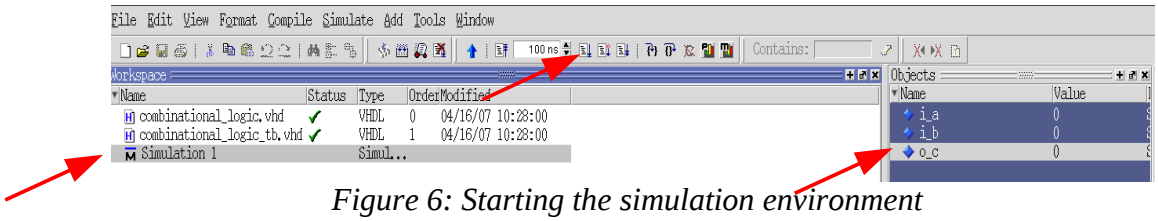

Figure 6: Starting the simulation environment

You can add waves to a W**aveform**, by right-clicking on the objects. If you now press ***run-all***, ModelSim generates a Waveform and you can see the signal sequence your module generates as shown in figure 7 .

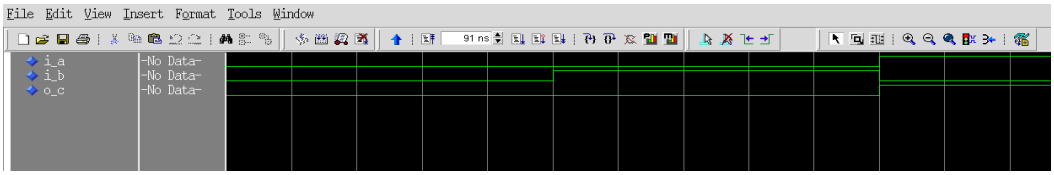

Figure 7: Example Waveform

By analysing this Waveform, you can see that the output o_c is only set when i_a is high and i_b is low, whis means that the function of this example is correct and satisfies the logic equation given.

b) Developing and Testing a Simple Register

In the next example, you must describe the function of a register and simulate it with the help of a provided script. The register you have to develop is a single-bit register with synchronous reset and should have the same function as a standard D Flip-Flop. This means that the value of the input *i_d* is assigned to the output *o_q* on each rising clock edge.

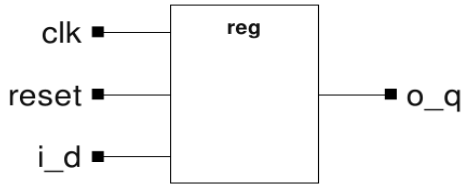

Figure 8: reg

Port declaration:

- *clk* : in std_logic
- *reset* : in std_logic
- *i_d* : in std_logic
- *o_q* : out std_logic

Working with scripts

As you have seen, working with the Project Navigator requires many manual steps inside of the GUI to simulate your code. This can quickly become tedious for larger projects with many design-iterations, so designers often try to automate this process through the use of scripts. ModelSim offers the possibility to use standard TCL or a simpler ModelSim-specific TCL-derivative.
The provided script you have to use is built with a few, simple commands. Of course, ModelSim offers much more possibilities to run a simulation with a script. For further information, please see the ModelSim Quick Guide on the lab's homepage.

```
# vlib creates library
vlib work

# vcom compiles your VHDL-modules (use vlog for Verilog-Code)
# regard hierarchy of modules, compile order is down to top
# if not declared in another way, modules are compiled to the
# default library work
vcom reg.vhd
vcom reg_tb.vhd

# start simulator
vsim -novopt reg_tb

# add wave adds wave to waveform
# hint: you can also add the waveforms by hand, save the
# waveform file and start the waveform file with simulation
# script (do waveform.do)
add wave sim:/reg_tb/clk
add wave sim:/reg_tb/reset

# insert divider
add wave -divider inputs
add wave sim:/reg_tb/i_d
add wave -divider output
add wave sim:/reg_tb/o_q

# run simulation
run -all
```

You can start the script in the ModelSim prompt with the ***do*** command.

```
VSIM 5>  do SCRIPT_NAME.do
```

Now the waveform window should open and you can check the function of your Flip-Flop.

To get used to work with ModelSim, you will have to use each of the described methods once in the next two exercises. For thefollowing exercises, you are free to choose the method you prefer.