1 Linux environment

For this course all laboratories are performed in the Linux environment, and often we will use the command line interface (CLI). Some useful commands and Linux information can be found on our internal wiki:
`http://www.microlab.ti.bfh.ch/wiki/huce:microlab:linux:start`
You will have to login first and then you'll get access to the Linux part of the wiki. Read the different sections and train yourself by little assignments to improve your skills.

2 Server: xena.bfh.ch

For the simulation, synthesis and place & route we are going to use the HuCE-microLab server *xena.bfh.ch*. This server is a 16-core machine based on Intel Xeon I7 processors with 48 GB of DDR3 memory; therefore, the server is a lot more powerful than the local machine.
To use the server's power, you have to establish a remote connection. To do so, execute the following command in a terminal: **ssh -Y xena.bfh.ch**

IMPORTANT: Only the commands in this terminal are going to be executed on the server! Because only this CLI has a remote session established.

Assignment 1

Open a new CLI ([CTRL] + [ALT] + t) and login to the server by executing the command

```
ssh -Y xena.bfh.ch
```

Start ModelSim by executing the command:

```
vsim
```

You will see no difference, as the GUI will be displayed on your local screen; however, the tool executes on the server and, therefore, executes faster than on your local machine.

3 Git

All sources for the laboratories are in the corresponding HuCE-microLab's Git tree. Git is a distributed version control and source code management system (DVCS). The students need to clone the git repository which contains all the laboratories of the module. Here is the procedure to do this:

1. Open a new CLI ([CTRL] + [ALT] + t) and use **cd** ~ to change to the *home directory*.
2. Create a new folder named e.g. praktika and change to that folder. In this document we use the name "praktika" to reference that folder. Use the following commands to proceed:

   ```
   mkdir praktika
   cd praktika
   pwd
   ```

3. Now, clone the corresponding repository by using Git. The repository on the server is named in consideration to the module identifier hence *btf4220-digital* and located on the HuCE-microLab Git server.

   ```
   git clone git://pm.ti.bfh.ch/btf4220-digital.git
   ```

4. The Git repository btf4220-digital contains sub-modules such as *bsc-course-templates*. To finish you'll have to initialize them too. During the exercises use Git features to get used to that work-flow. A nice feature is the branching method Git provides.

```
cd btf4220-digital
git submodule init
git submodule update
```

In the directory ∼**/praktika/btf4220-digital/laboratory1** is a directory structure as listed in table 1. Change to that directory by executing the command:

```
cd ~/praktika/btf4220-digital/laboratory1
```

Directory	Information
doc/	Documentation.
sandbox/	Directory for all temporary files.
vhdl/	Directory containing the VHDL files.
config/	Directory containing configuration files.

Table 1: Sub-directories of each laboratory.

4 Simulation

We are going to simulate the VHDL code of the traffic light exercise that you know from previous semester. For this purpose login to the server and change to that directory:

```
cd ~/praktika/btf4220-digital/laboratory1
```

Run the command **make clean** to set up the environment correctly.
Change to the sandbox directory:

```
cd sandbox
```

Before continuing make sure you are in the correct directory. (Up to this point you were working with Linux, Git and Make theoretically, only. For improvement of your skills analyze the Makefile and get used to the Linux command interface (CLI). Make sure you know what you did. Do not just copy-paste. Tipping the commands help learning them for free, so do not waste time take the opportunity you got.)
Execute following commands:

```
rm -rf work  # This command will remove the simulation library.
vlib work    # This command will create a new simulation library.
                                         #
vcom ../vhdl/trafficLight-entity.vhdl    # This command will compile the entity.
vcom ../vhdl/trafficLight-behavior.vhdl  # This command will compile your VHDL
                                         # description of the traffic light state
                                         # machine.
vsim -t ps trafficLight  # This command will start the ModelSim GUI and will start
                         # the simulation (-t ps sets the simulation mode to ps
                         # accuracy).
```

Note: Remember the slide with the simulation work-flow (writing VHDL models → building simulation modules → simulation/analyze the component's behavior.) Did you see the separation of the entity and architecture on file level? All characters after the "-" are ignored hence the system interpret the two files as if all where in one single VHDL file. This method is handy for increasing structure and readability.

Assignment 2

We are now going to simulate all possible combinations of the inputs to verify the functionality of the block - called Device Under Test (DUT). Start up the "**wave**" window by executing the command shown below in the ModelSim command window.

```
add wave -hex *
```

A window should pop-up with all inputs and outputs of the circuit. Next we have to provide the simulator with stimuli. For all inputs of the component. Some examples are given for *ClkxCI* and *RstxRI* by the following "**force**" commands:

- `force ClkxCI 0 0ns,1 1ns -repeat 2ns`
- `force RstxRI 1 0ns , 0 10ns`

Formulate also the command for the input *BtnxDI*. How long do we need to simulate x in ns? Draw the signal diagram of the input stimuli on a paper and think about the test-cases. What are your expectations, the DUT behaves? Perform the simulation by the command:

- `run x`ns

5 Automated simulation

For this simple example it is "easy" to type in all commands or to use the GUI; however, in complex designs with a lot of error prone components we require (1) reproducibility and (2) exclusion of errors such as typing and/or GUI clicking mistakes. For this purpose most industrial tools provide us with command line interface (CLI) that can be used in an automated manner that guarantees the above two items. In this assignment we are going to use such an automated environment to redo the previous simulation.
The automated simulation method is already partially in place and uses the GNU make and command line utilities Linux systems provide. The primary goal of this assignment is not to fully understand the automated simulation flow, but to use it.

Assignment 3

On the server change to the directory ∼**/praktika/btf4220-digital/laboratory1/**.
Run in the terminal the command **make**. You will see a small explanation of the make targets. Execute in the terminal the command **make clean** to remove all temporary files from the **sandbox** directory, first.

IMPORTANT: Before executing this command make sure that you haven't copied any file in the **sandbox** directory that you want to keep!

For the automated simulation there are two important files, namely (1) **config/project.files** and (2) **config/project.force**. The first file contains all the names of the VHDL files that are required for the simulation whilst the latter contains the commands required for the simulation. The file **config/project.files** is already complete; however, you need to add the command used in Assignment 2 to the file **config/project.force**. Don't forget to put the command **vsim -t ps trafficLight** before the "add wave" command to setup the timescale properly.

▶ `add wave -hex *`

After having added these commands to the file, go back to the laboratory1 folder and execute the make target for the simulation.

```
cd ..
make sim
```

As you see you get automatically the same simulation as performed manually in Assignment 2. That time it is reproducible and repeatable the same way weeks later.