First VHDL simulations and syntheses

In these sections you will learn or recall how to start the VHDL simulator, how to write and simulate the basic VHDL structures and how to synthesize them for an ASIC target. This first chapter, given your pre-requisites, shouldn't carry anything difficult to you, but please pay attention to the details, to the VHDL writing styles and constructs, on how to use test benches for a meaningful, portable and easy simulation phase. Probably you are a new-by for what concerns ASIC synthesis, so please pay attention to the first hints on this very important step.
As a rule of thumb you are expected to finish this setup part (vhdl simulating) within the first hour. The last part should be enought for the synthesis session.

If you are not a UNIX expert, you first need some suggestions on how to create directories, handling files, et cetera. So go through the next section as a first step.

0.5 Let's start!

At first you need to login on your working area. In the NX login window you already configured type the login and the password that have been assigned to you. In general it is

ms18.X for groups from 2 to 9

ms18.XX for all the other groups

that is the prefix **microsystem14** followed by the XX group number (assigned on the preadsheet you are supposed to have accessed already, GR_XX is the group where XX is the group number, 1 digit from 1 to 9). The password is set (default) to

mic!tem18sys

Once you are logged in YOU MUST CHANGE THE PASSWORD. You are not allowed to go on untill you don't change it. To do so:

- first open a "shell" terminal (the action is: right click over a blank zone of the screen, then click "Konsole")
- now choose a password of 6 characters, one or more than one of which should be a special character like **!** _ #
- then digit at prompt the command

  `prompt>` **passwd**

  The system asks for the OLD password first (your login in this case), then it asks twice for the NEW one
- After you completed this operation, COMMUNICATE the new password ONLY to the teachers during the first lab.

The following commands are ment to create directories and copy files. Clearly, you can use the directory browser. As this is pretty straightforward you will find herein instructions only for other approach: using terminal and manual commands: even though you may think this is more difficult, it is, once you are expert, much faster and powerful. In any case, then, this is a good opportunity to learn.

You can start now with creating the first directory. At the prompt digit

```
prompt> mkdir setup
```

which creates the directory **setup** under /home/ms16.XX if you are the group "XX". If you want to see it, type

```
prompt> ls
```

which is the command that lists the files in the current directory. Then enter the directory by typing

```
prompt> cd setup
```

Now you are in **setup**. In this directory you will organize all your results (not vhdl only). So just create a new subdirectory in which you will work with your vhdl simulations:

```
prompt> mkdir vhdlsim
```

and another directory in which you will run your synthesis:

```
prompt> mkdir syn
```

Enter the first directory:

```
prompt> cd vhdlsim
```

Suppose you want to copy a file located in the directory

/home/repository/ms/setup/

into your own new directory. Type the following command

```
prompt> cp /home/repository/ms/setup/iv.vhd .
```

(PLEASE please note the final dot!!! do not forget it, and there is a space between vhd and the dot) This action copies the file named **iv.vhd** in the repository's directory on your current (**.**) directory (the dot means the current directory). Thus the **cp** command has two arguments: source and destination.

Type again

```
prompt> ls
```

to see your new file.

Now copy all the files in the repository:

```
prompt> cp /home/repository/ms/setup/* .
```

(Please not the star *** Same files will be also available from the course website).

The directory now contains the VHDL description files of a few small gates, a test bench, and a very simple script for compiling the VHDL netlist in the modelsim environment.

You can edit your inverter file using a few possible editors as EMACS or KEDIT or VI or JOE. Most of the editors recognize the VHDL keywords. EMACS is a little tricky but more powerful as it enables a completion of VHDL constructs while you are writing. To try EMACS run the following command at the prompt (you can also find it in the applications menu)

```
prompt> emacs iv.vhd &
```

A new window opens. The symbol & helps in putting in background the process you started: this means that you will still be able to use the terminal in which you where working. This is true for

every command you are going to launch. Should you forget to type the &, you can anyway put the process in background by typing:

```
prompt> CNTR z
```

this action suspends the process, and after typing:

```
prompt> bg
```

Just to try EMACS' power: add and entity (you will not save this file.. it is just to try) by typing 'ent' and press tab .... entity has been completed; now type space: you can type the name as it appears in the bottom line of the window; now type enter: the 'end entity' appears and the 'generic' structure if you have one. Suppose you have not: type enter. Now the 'port' definition appears... now you can PLAY yourself...

Do not save this file and go through the following instructions.

Now you are ready to analyze and simulate a VHDL block.

0.6 VHDL simulation

0.6.1 Simulating a MUX

Reading the INVERTER netlist you can recall the use of generics. Notice that a library called **WORK.constants** is used: the constant IVDELAY is not defined within the file. It is in **constants.vhd**: have a look of it. [1].

Now you are ready to analyze the ND2.vhd gate and the various descriptions of the MUX21. Please, take your time and be sure you understand the meaning of all the code you are reading.

Now we want to use the mux file within a so called "test bench": **tb_mux21.vhd**.

The test bench is a wrapper which uses the circuit you designed; it helps the simulation, but is not part of the architecture. It is extremely useful and avoids forcing signal by hands.

It contains an *instance* named **U1** of the **COMPONENT MUX21** which is first declared (keyword **component**) and then instanced (keyword **Port Map**). This test bench also instances the other three types of MUX, by declaring U2, U3, and U4. Notice that the entity is always the same. The different architectures are differentiated by the configuration declaration at the bottom.

Remember that the assignment

```
inputA1 <= '0' after 1 ns, '1' after 2 ns, '0' after 10 ns;
```

creates a signal waveform for the signal named **inputA1**.

Now let's start with the simulator. We use Modelsim by Mentor Graphics in these labwork. As a first step we must set up the environment variables, so type the command:

```
prompt> setmentor
```

Now type at the prompt the command

```
prompt> vlib work
```

Type again **ls**. You should see that a new directory called "work" has been created. This is a temporary work directory in which the simulator save its results. Now call the simulator:

```
prompt> vsim &
```

A window named *MODELsim SE PLUS 6.2* opens. The operations needed for the simulations are:

[1] Remember that if you simply want to read the content of a file widthout necessarily modify it you can type on a shell the command: **more filename**, and press the spacebar to go through the file or CNTR c to interrupt the reading.

- Go to **Compile→Compile**. Double click on file **constants.vhd**. You'll see that in the command window of the main window a command **vcom** has been executed. This command COMPILES your VHDL file *constants.vhd*. The result of the compilation is written in the directory **work**.
- Now compile the other files in a hierarchical sequence from bottom to top. You can use either the menu, as in the step before, or type the equivalent command manually in the command window: for example `vcom iv.vhd`. Or better, you can use the shell script (outside the simulator, in the terminal):

  ```
  prompt> ./compile
  ```

  What happens?
- When it is finished open the simulation window (Simulate→Start Simulate). Choose a simulator *resolution* of 10 ps and *add* the design **work/MUX21TEST** (which is the configuration name in the test bench: be careful, if you have different configurations and you want to simulate them all that you need is to select the configuration and not just the entity in this point). Then click **load**. Disable the optimization option. The design is now ready for simulation. Note that the equivalent command that has been executed in the main command window is: `vsim -t 10ps work.MUX21TEST`.

  REMEMBER in the future to set in all the simulations the RESOLUTION. The smaller is the number you choose the longer is the simulation. However, the resolution must be lower that the smallest timing value you happened to use in your VHDL description. REMEMBER!
- If you want to see the waveforms type the following command

  ```
  add waves *
  ```

  in the command window. A new waveform window opens. You can also play with the menus to decide the waveforms you would like to display. Obviously there's nothing to plot until you don't run the simulation!
- To do so type

  ```
  run 3 ns
  ```

  which means that you start the simulation from 0 to 3 ns. The step for the simulation is 10 ps as you chose before. The time unit defaults is nanoseconds. But you can choose another value. Try for instance to type again

  ```
  run 8000 ps
  ```

  Look at the waveform window. Does the muxes mux?.Are there differences for the various different architectures used?

You can plot the waveform window to a file: the action is click the waveform window, then select **File→Print postcript**, a print window opens: select the **File name** option and add a file name with extension ".ps". In the directory in which you launched the simulator a new postcript file is now present. If you want to view it type at the prompt:

```
prompt> kpdf filename.ps &
```

In case you want to transform it into a pdf file, just use the shell command `prompt> ps2pdf filename.ps` and the equivalent pdf will be generated.

To better visualize the waveform in the sheet, before printing you can select the signals you want to print, and select signal height in pixel by the menu: **Wave→Format→Height** (select for example 250).

Here is now a string recommending the most important point you should have at the end of the exercise (this will be given for each exercise.

Summary of what is requested

We suppose you already knew how a MUX works. So at this point you are supposed to know how to write and simulate a vhdl circuit. Nothing else is required.

0.7 From VHDL to synthesis

You are going to synthesize the blocks you have simulated in previous section. You will use a standard cell flow (subject of this course, described later) based on one of the most powerful synthesis tool diffused in industry.

Before stepping over, please go in the other sub-directory in which you will synthesize:

```
prompt> cd ../syn
```

Copy in there all the VHDL files except tb_mux21.vhd (is used only for simulation and would be wrong to synthesize it).

Now open a new shell and go inside such new directory.

0.7.1 VHDL Synthesis of your simple circuits

In this section you will learn how to run a synthesis, given a synthesizable VHDL code and how to analyze a few results of your synthesis. You will be given further informations in the next chapters.

In general the phases you should follow using a synthesis tool are reported below:

1. **Define a library** of gates previously characterized (ex: AND2, input capacitance, delay function of fan-out, power consumption at each output switching, ... in the next chapters you will learn more in detail this point).
2. **Analyze and elaborate the file** you are going to synthesize (in a HDL format): during this step the tool checks if your design is correctly synthesizable. If not, it returns error messages and you can't go through the synthesis until you don't modify the code.
3. **Define constraints** for the synthesis. This is not strictly necessary, but if you don't perform this step you'll obviously get non optimal results. You can set one or more constraints in the same time: the tool builds a multi-objective function and try to figure out the result that assures the minimum distance to all the constraints.
4. **Synthesize the design**. The synthesizer maps the VHDL code on the gates in the library, using their characteristics to choose the gates optimal from the constraints point of view.
5. **Report results**. You can analyze at this point if the constraints are met or not, you can list the critical paths and the power informations for nets and cells, even if the design is a hierarchical one.

In our case we will use as a synthesizer the **Synopsys Design Compiler**, which is the real engine, and its GUI: **Synopsys Design Vision**. Before starting the exercise it is necessary you go through a few steps for making the tool working correctly.

Be sure you have in this directory the file **.synopsys_dc.setup** (if you don't have, you should copy it from **/home/repository/ms/setup/**) :

cp ../.synopsys_dc.setup .

This is a hidden file that you can list using the command:

prompt> **ls -a**

and that you can normally display using **more** or **emacs**. It gives some information to the tool, such as which is the library you are going to use and where it can be found. Anyway, you don't need to modify it right now, just remeber to have it in the directory you are using for the sysnthesis whenever you are synthesizing.

Now the environment variables needed to launch the Synopsys synthesizer must be defined: this is done in a script file that you can execute by typing the command:

prompt> **setsynopsys**

As a last preliminary operation create a directory in which the synthesizer saves temporary files during compilation:

prompt> **mkdir work**

A final IMPORTANT comment. With the informations you will be given hereinafter, you should be able to synthesize all the VHDL designs you have already completed. Two are the main ways to perform a synthesis: using the graphical user interface and going to and fro around the monitor clicking with you mouse here and there, or using a command window in which you are allowed to execute singular commands or a script grouping a command sequence that you execute only once. The use of the command window and of scripting is obviously the clever way, but it is not for a newbie. So you'll learn here how to click in the right sequence and understand what's going on; sometimes informations and suggestions will be given so that, hopefully, you will be automagically transformed from a newbie to an experienced user.

0.7.2 Synthesis of the MUX

The aim of this section is twofold. The first is to let you learn how to use the synthesizer. The second is to let you learn how the different VHDL descriptions are synthesized.

Now launch the synthesizer by typing:

prompt> **design_vision &**

A main window opens with title: "Design Vision". Immediately note the bottom command line, from which you can manually set commands. In the space above, the results of your commands are displayed. If you operate using the main window menu, the name of the corresponding command and its results are displayed in the command window as well. For the moment use this command line only when suggested.
Please note exactly above the command line the menu with the alternative "Log", which corresponds to the log window above, and "History": select it. Each time you will type a command in the command line, or give a command by the Design Vision buttons, you will have it in this history window. You will be able to use it for editing, executing or saving such commands, so that you will easily generate scripts for you next sessions.

Now let's go back to the Log window and let's use the interface: using its menu perform the following operations:

Analyze file: From menu select: **File→analyze**; a window opens from which you can type "Add", select the **constants.vhd** file and click OK to confirm your selection. In this phase the code is checked and errors are displayed if it is not synthesizable. Note the corresponding command in the Log window and in the history window.
Now analyze the **iv.vhd** file, the **nd2.vhd**, and the **mux21.vhd** files with the same steps as before. The top entity is the MUX21. Note that in file constants the delay values are defined, but should be commented in the top level: the timing parameters are ignored for the analysis, and must not be included in the further steps (open the files **iv.vhd** and **nd2.vhd** that have the delays included, comment the delays using "- -" , you can recognize the delays from keyword **after** and save files; then analyze again).

Elaborate From menu select: **File→elaborate**. During this step the compiler creates an internal graph of your circuit and maps it virtually to general gates. You need to elaborate only the top level entity. In this first step we will work on the fist behavioral architecture, so, in the "Elaborate design" window select the "WORK" library, and the "MUX21(CFG MUX21 BEHAVIORAL_1)" option.
You can now explore this "logical" structure of the MUX. In the left "Logical Hierarchy" window select the MUX structure. Now, in the top menu the "Create Symbol View" and the "Create Design Schematic"appear. Click the "Create Symbol View" button: a MUX symbol appears. Now double click inside of it: the schematic view appears. You can surf in it by using the right mouse button (Zoom selections... ESC to leave the Zoom option). If you click in the internal boxes you will only get the logic representation created using generic basic ports. These are not the cells in the library we are going to use for the synthesis.
Before stepping over, click on the UP ARROW (on the top menu) till you arrive to the top view in case the arrow is black.
Check that in the log window appears: **current_design "MUX21"**

Look one moment in the history window the correspondence between your selection and the given command.

Synthesize: for the moment we do not use constraints for the synthesis, so go to the main window and select **Design→Compile Design**. A small window opens, in which, for the moment, you don't need to select anything different from the default: just click OK. This corresponds to giving in the command window the instruction

compile -exact_map

Now the tool is mapping your design on the library we are using ($0.045\mu m$). When it has finished you can explore the results from the main window exactly as before. Note that the MUX structure did not change, but the internal structure, which is behavioral, has been replaced by exact cells present in the given library. For example, select the internal leaf. Click with the right mouse button selecting "Properties": you will read the real name of the used gate in the library.

You can plot to a file the schematic displays in the main window by selecting **File→Print schematic**, checking the "Print to file" option and then typing the file name.
If you want to go up in hierarchy you must click on the up arrow displayed on the top bar of the main window. Remember that whatever command you are giving, it will be applied to the hierarchical level you are in.

Save the design: We want to save this compiled design so that, in case we optimize it or change something, we can start again from here without reproducing the previous steps. From the main menu select: **File→Save as**, and in the new window put a meaningful name, eg. "mux1.ddc". Hereinafter you will be able to get your design at this point by simply reading it from the main menu (**File→Read** and select it.)
Please look at the command in the history page and remember it.

Generate to VHDL file: We want to see the VHDL netlist of the whole design so that we would be able to perform a backannotaded simulation.

Now, exactly as before, from the main menu select: **File→Save as** and choose vhdl as extension (vhdl, not vhd). Now look at the file end analyze the mapped structure: clearly it supposes to have the entity description of each of the component declared.
Please look at the command in the history page and remember it.

List reports: Now we want to analyze the circuit performance. From the main menu select **Design→Report Area**. A new window opens. You can decide (do decide it now..) to write the report on an external file by checking the "to File" box and inserting the file name, eg. *mux-area.txt*. The same command can be obtained by typing in the command line

report_area > mux_area.txt

Hereinafter instruction for saving the reports will not be given: it's up to you to decide when you need to save the results on a file or to write down the results you will find from these reports so that you can compare them with future results obtained by constraining the synthesis (next chapters).

Now report the timing analyses: from the main menu select **Timing→Report Timing paths** and leave the default settings. Save the report on a file. You will see the timing report for the worst critical paths on the report window: try to understand the given informations.
The same results could have been obtained simply writing in the command line (try it) the command:

report_timing

In this way the worst critical path is displayed. If you want to see a graphical correspondence, return to the top level schematic view and select from the main menu: **Highlights→Critical Path**. Now explore the path of the critical signal going inside the blocks. Follow in detail the correspondence between the displayed path and the sequence given in the timing report. You can chance the highlight path color (play as you like...)

Finally you are for sure interested on how to analyze power. The command to be used is:

report_power

Look at the different power contributions and analyze them.

Mind: this command must be carefully used: during the next labs you will be warned about this point.

It's your turn: Now you know how to run a synthesis. Repeat the same steps as before for the other MUX architectures and analyze carefully the different results.

Remember close or suspend the NX session when you are done. To save your work you can send it by e-mail using the available browser using your student account, or use your disk space in your Portale della didattica disk, or use the ssh command.

For this lab, nothing is to be submitted! But please, in all the next labs you will be using this systems and commands, so you are expected to know them very well.