HDL Lab Exercises Tutorial

TECHNISCHE UNIVERSITÄT DARMSTADT

Summer Term 2018
Prof. Dr.-Ing. Klaus Hofmann
M.Sc. Dominic Korner

Outline

1 Introduction

In the early 1980s technological advances enabled the production of ever more complex integrated circuits. However, both digital and analog designs were still done by schematic entry, slowing down designers to the point where they could not keep up with the available levels of integration. This "design crisis" led to the development of high level hardware description languages like Verilog and VHDL. In this lab the design flow from written Verilog code to synthesized hardware in the umc65ll technology is done.
The idea is to digitally filter an analog waveform. Therefore, the signal has to be converted into digital domain. After filtering, the signal can be converted back to analog signals. One example where this design is used is music recordings.

2 Exercise

1. Do the Cadence tutorial given in section 3.1 which includes implementing and simulating the given OpAmp.
2. Go through the ModelSim tutorial in section 3.2 and implement the clock generator.
3. Implement an ideal DAC in Verilog-A. At least the bit width and reference voltage have to be adjustable through parameters. If you have the time think of non idealities which could be implemented(Offset voltage, rise time....).
4. Implement a 16bit ADC by using the previously implemented DAC and the OpAmp which you have designed in the tutorial. The ADC have to consist of different building blocks in the Cadence schematic editor e.g. counter, logic and so on. The ADC features an input for the clock and analog signal, as well as a 16bit parallel output bus. For the first design you make, a minimum sampling frequency of 100kHz should be supported. First decide for a suitable architecture of your ADC, then start implementing your ADC. Later you can also change the architecture to improve the performance of your ADC and refer to it in the report.
5. Create a cell called `WAV-Reader` and a cell called `WAV-Writer`. The verilog code is given in: `home/Your_Group/HDL_Lab_UMC65/implementation/umc_65_ll/WAVE-Verilog`
   Also generate a symbol for each cell.
6. Imagine you want to save space on your hard drive. Therefore, you want to change the sampling frequency of your `.wav` audio files from 44.1kHz to 22.05kHz. Develop a digital filter for filtering the incoming WAVE file and a downsampling circuit in verilog.
7. Set up a Cadence cellview which includes the WAV-Reader, WAV-Writer and your filter. You can filter some example sounds given in `home/Your_Group/HDL_Lab_UMC65/implementation/umc_65_ll/WAVE-Verilog` or record your own sounds. Save the examples with a sampling rate of 22.05kHz and compare them, is there a difference between the original file and the filtered/downsampled? Additional, integrate your DAC and ADC in the schematic after the WAV-Reader to do a digital to analog and then analog to digital conversion. Is the performance of your ADC effecting the audio signal?
8. Synthesize your design of the digital filter with Synopsis and optimize your design based on the umc65ll process for speed and area. Therefore, you can also select different filter architectures. A short introduction of the synthesis with Synopsis Design vision is given in section 3.3.
9. Set up a voltage source of two added sinus waves with a dc offset of 2.5V: One with 500Hz and an amplitude of 1V and the other with 20kHz 1.5V. To convert the analog signal you need an ideal ADC, write the necessary Verilog-A module. Sample this voltage source with 22.05kHz and store the resulting data as `.wav` file. Compare the results of the sampled signal with and without your filter. Do you see/hear any differences between them? What is causing this difference?

3 Manual

3.1 Cadence

3.1.1 Schematic Entry

To start cadence open a Terminal and navigate to your working directory by typing

Terminal
```
cd HDL_Lab_UMC65/implementation/umc_65_ll
```

Then execute the start script with

Terminal
```
./virtuoso.sh
```

The main window should open as shown in figure 1:

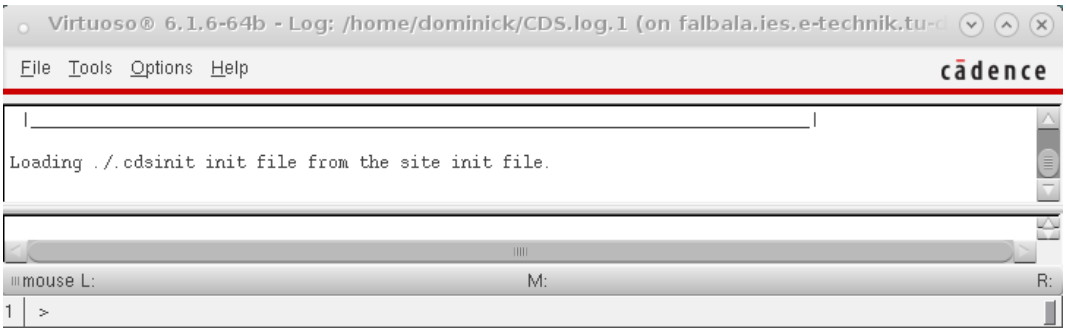

Figure 1: Main window of Cadence.

Click Tools » Library Manager to open the Library Manager. In this window you can see all the installed cell libraries and you can create your own library. Since the UMC65ll technology is used in this lab, all technology specific elements are placed in this folder. Check `Show Categories` to view the categories column. To create your own library click File » New » Library, in this example the name HDL_Lab is chosen, and click OK. In the opening window select `Attach to an existing technology library` and click OK, in the next window select the technology UMC65ll. Then the Library Manager should look like in figure 2.

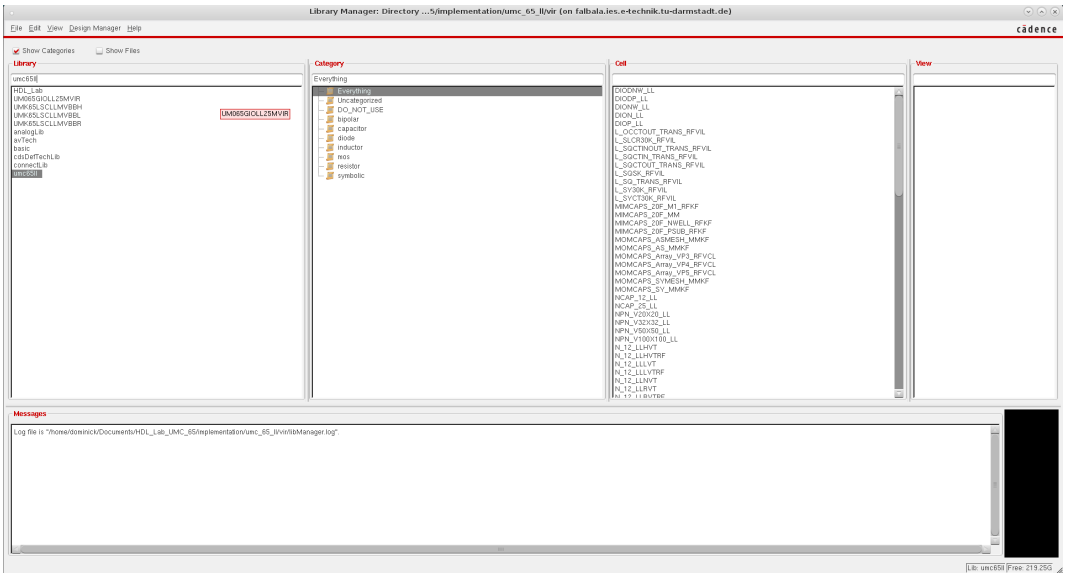

Figure 2: Library Manager of Cadence.

In this tutorial we will build an operational amplifier, therefore click on File > New > Cell View. In this window you have to define the name of your cellview in the row Cell and in the row Type, the type of the file which you want to open. Since we want to build an operational amplifier on the transistor level we open the type schematic. The resulting window should like in figure 3.

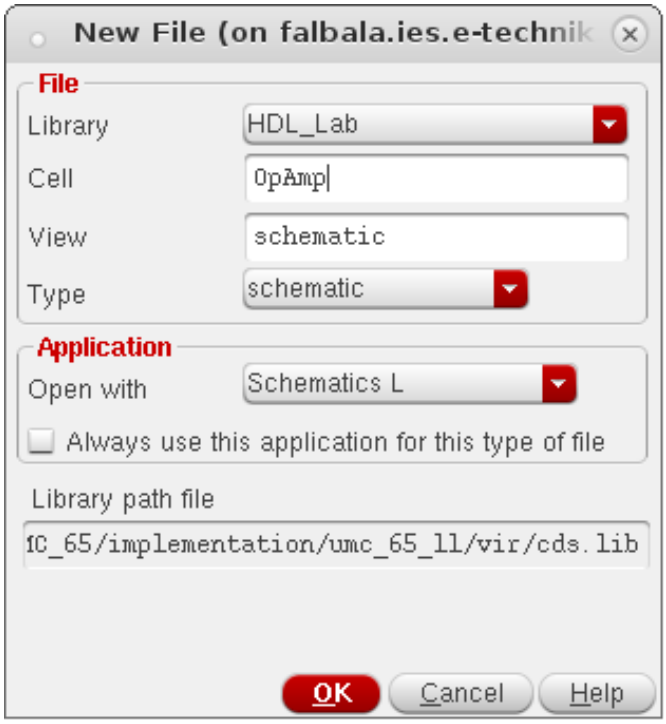

Figure 3: Cerate a new file in Cadence.

By clicking OK the schematic editor opens. With Create > Instance you can insert transistors and all the other elements provided by the technology UMC65ll. In the window choose Browse and in the opened Library Browser select the library umc65ll, there you will find the transistor N_BPW_12_LLRVT. Select the view Symbol, then you can place the NMOS transistor in your schematic. Figure 4 shows the schematic of the differential stage of the operational amplifier and figure 5 the circuit for generating the required bias voltages. After you placed the transistors you can wire them with Create > Wire (narrow). The names written on some on the nets are wire names. Nets with the same wire name are electrically connected to each other. For placing a wire name click Create > Wire Name. The red contacts at an end of a net are pins. Pins are later the electrical contacts to other cellviews. To place a pin select Create > Pin. Be careful to select the right direction of the pin(input, output or inputoutput). The last part which is missing in our schematic is the power supply. To add them, open the Library Browser go in the analogLib and search the cell vdd for the positive voltage supply, and gnd for the negative voltage supply. When you have added these parts to the schematic, your schematic should look the same as in figure 4 and 5.

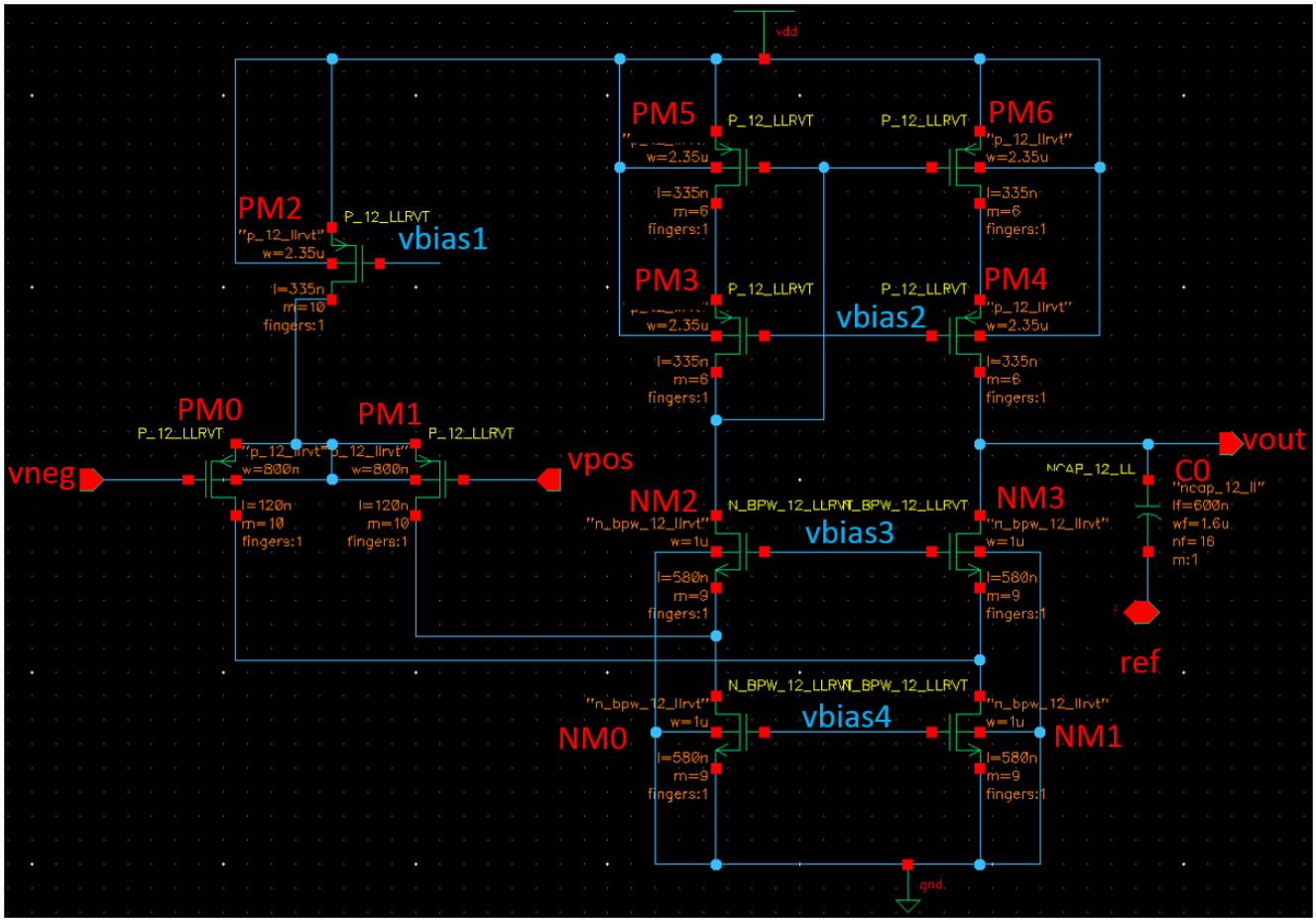

Figure 4: Differential stage of the OpAmp.

Table 1: Properties which have to be applied to the transistors.

Device	Identifier	Parameters			
		Length	Width	Finger	Multiplier
P_12_LLRVT	PM0,PM1	120n	800n	1	10
P_12_LLRVT	PM2	335n	2.35u	1	10
P_12_LLRVT	PM3, PM4, PM5, PM6, PM15, PM16	335n	2.35u	1	6
P_12_LLRVT	PM7, PM8, PM9, PM10, PM11, PM12	335n	2.35u	1	11
P_12_LLRVT	PM13	335n	2.35u	1	5
P_12_LLRVT	PM14	335n	2.35u	1	1
P_12_LLRVT	PM15	335n	2.35u	1	6
N_12_LLRVT	NM0, NM1, NM2, NM3,NM4, NM5, NM6, NM7, NM11	580n	1u	1	9
N_12_LLRVT	NM8, NM9	580n	1u	1	5
N_12_LLRVT	NM10	580n	1u	1	1
NCAP_12_LL	C0(210fF)	600n	1.6u	16	1
RNHR_LL	R0(19.99KOhm)	9.74u	2u		4 Segments

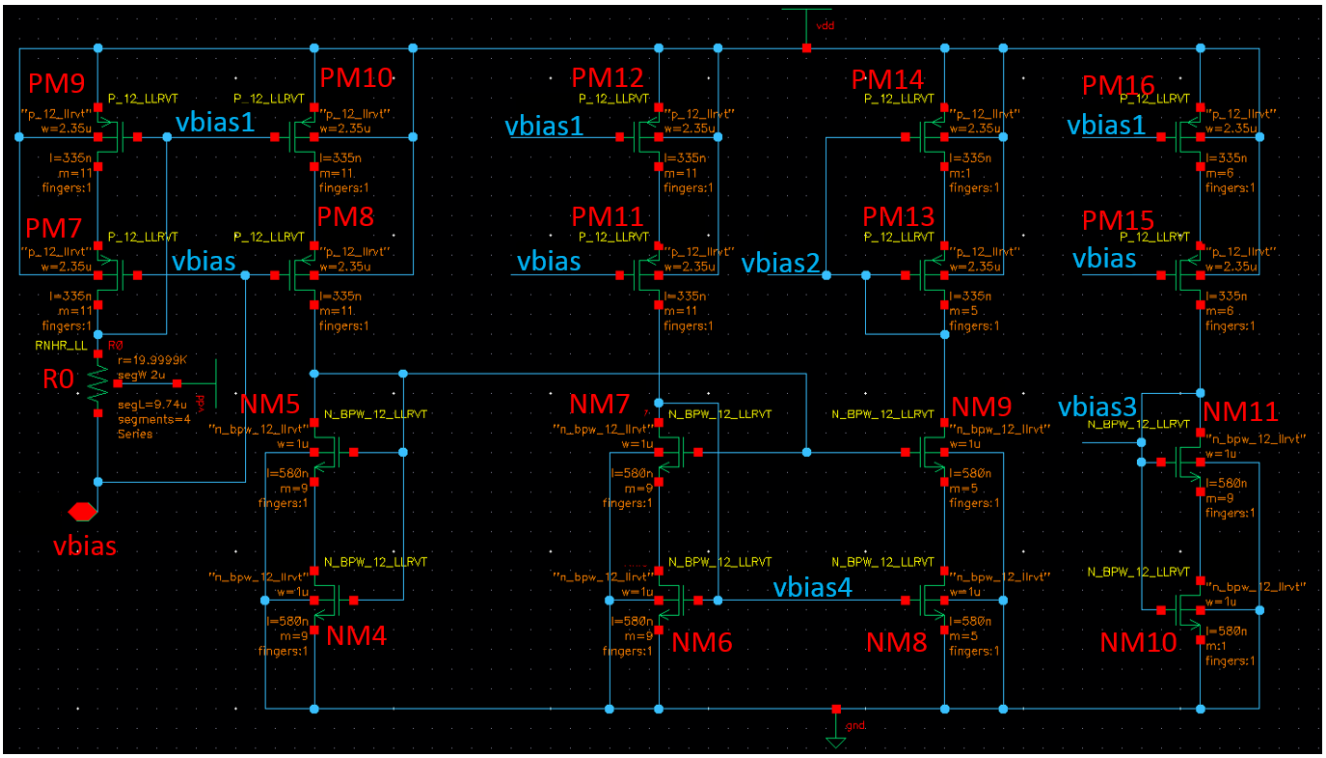

Figure 5: Bias circuit of the OpAmp.

The last step in the schematic entry is now to set up the right properties for each component in your design. Therefore click on an instance in your design and press q. The `Object Properties` windows opens and you can change the necessary properties. The required properties are given in table 1. This you have to do for each component in the schematic.

After you have the right configuration of all the components, you can generate a symbol for your operational amplifier. Therefore, go to Create > Cellview > From Cellview. In the opening window click OK. The next window should look like the one in fig 6. After clicking OK, the `Symbol Editor` opens. Here you can redraw the symbol in such a way that the cell view is easily recognizable as an OpAmp when instantiated on a higher level cellview.

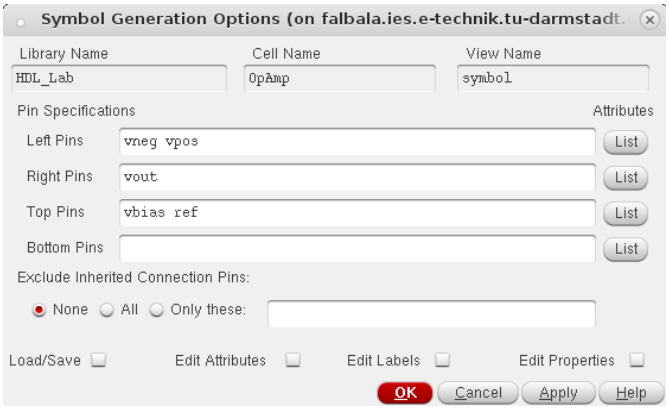

Figure 6: Symbol generator window.

3.1.2 Verilog AMS Simulation

For simulating the OpAmp, you first have to open a new schematic cell view, you can name it OpAmp_Testbench for example. In this cellview you place your designed OpAmp, as well as the following instances:

- A DC voltage source to specify the supply voltage vdd = 1.2V
- A ground node `gnd`
- Three DC voltage sources to supply the voltage `vneg`, `vbias` and `ref` with 0.6V
- The input stimuli for the OpAmp, for a short test you can use a sine wave `vsin` with an amplitude of 0.5V, a dc offset voltage of 0.5V and a frequency of 1MHz.

All the before mentioned elements are located in the `analog lib`. If the schematic looks similar to the one in figure 7 you are ready to configure the simulator.

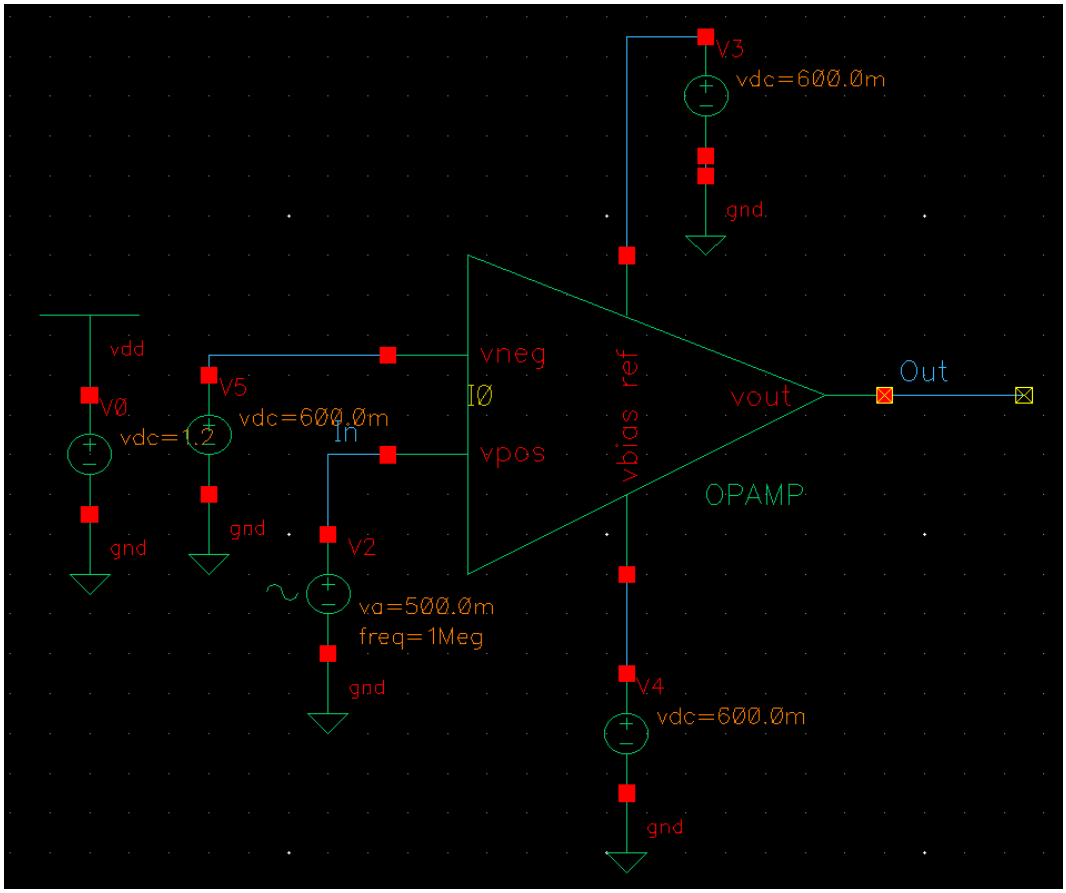

Figure 7: Testbench schematic for the OpAmp.

Later we want to simulate a mixed signal environment, so we have to use the AMS simulator. To configure the simulator you have to create a new cell view in the same directory as your testbench, in our example the OpAmp_Testbench cell, of the type config. Then click OK and the `New Configuration` window opens. Select `schematic` as view and then click `Use Template` and select the AMS template. After you have finished the configuration it should look like in fig 8. Click OK and the windows closes. Now you are in the `Hierachy Editor`. Here you can define which simulation model you use for every cell view in your schematic. Leave everything as it is, press the save button and close it.

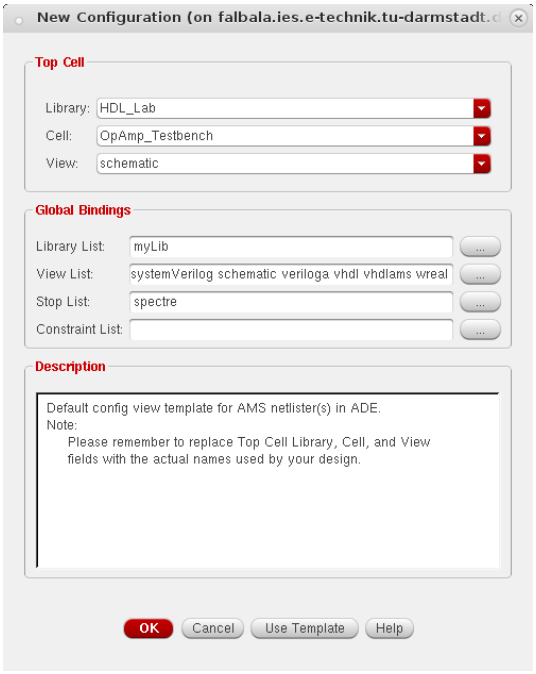

Figure 8: New Configuration windows.

Go back to the testbench schematic of your OpAmp and open the `ADE L` by using Launch » ADE L. In the `ADE L` go to Setup » Design and choose `config` as `View Name`. After clicking OK navigate to Setup » Simulator/Directory/Host ... and in the opening window select the ams simulator as shown in figure 9 and click OK.

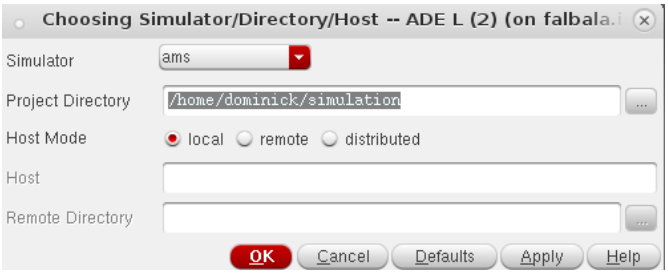

Figure 9: Chose the AMS simulator.

Since you use transistors of the umc65ll technology you have to tell the simulator where to find the simulation models. Go to Setup » Model Libraries in the `ADE L` window and add the libraries as shown in figure 10. Remember to also choose the right section of the library!

Figure 10: Libraries which have to be added for the AMS simulator.

Next go to Setup > Connect Rules/IE Setup ..., select Interface Element/IE-card Based Setup(OSS/UNL) and change the Vsup Value/Net to the supply voltage of your digital blocks, in this technology it is 1.2V.
To set up a transient simulation go to Analyses > Choose and select the tran simulation and a stop time of 10u. The last step is now to select the nets which you want to plot after the simulation, to do so, select Outputs > Setup and in the opened window click From Design. The schematic viewer opens and you can click on the nets which you want to plot. If you have given the nets names(for example In and Out by Create > Wire Name...) the resulting ADE L should look like in figure 11.

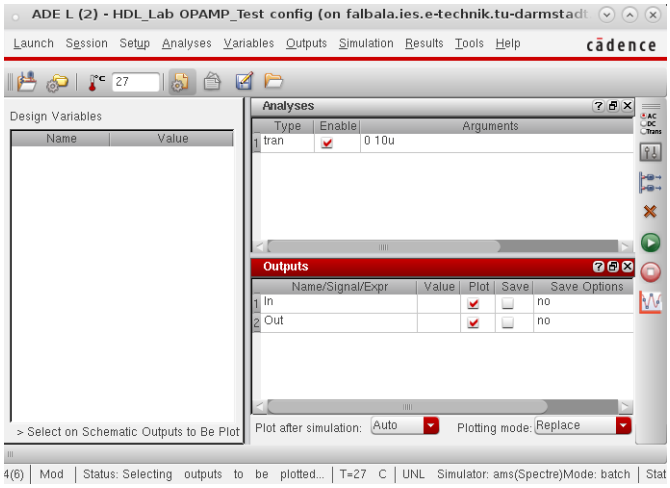

Figure 11: The ADE L configurator.

Finally to start the simulation click on Simulation > Netlist and Run in the ADE L. After a short time the Virtuoso waveform viewer opens and show the simulation results, this should look like in figure 12.

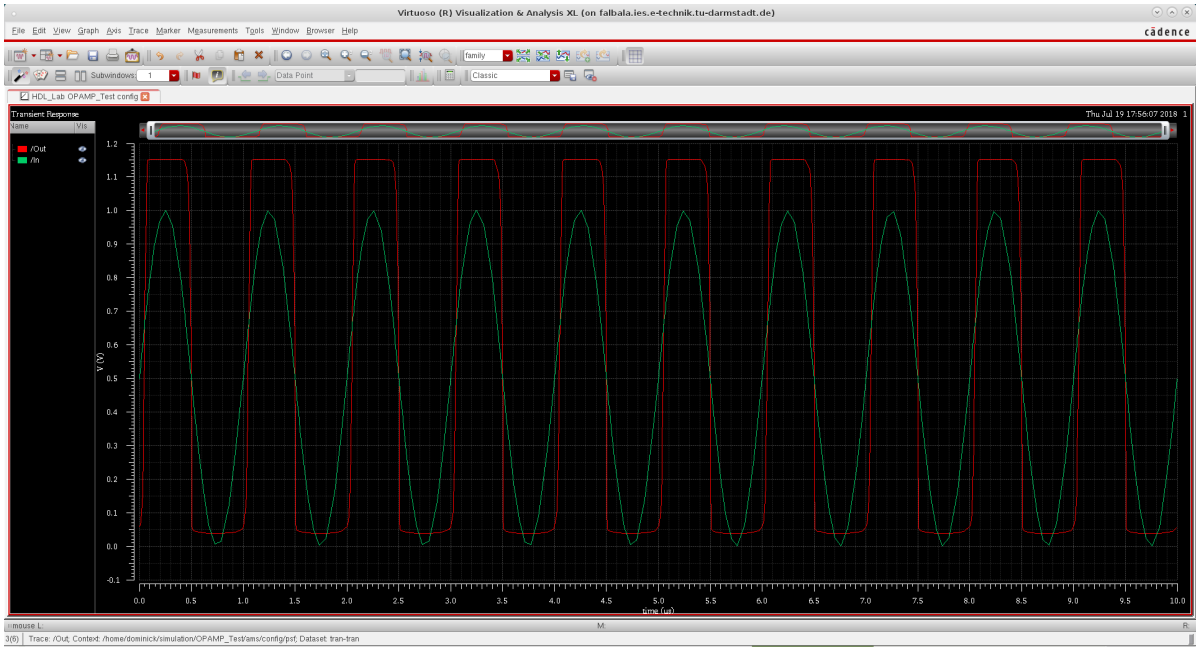

Figure 12: Waveform viewer Virtuoso.

All the previous mentioned procedure works in the same way for Verilog and Verilog-A simulation, except that you have to open a different type of editor for opening a new Cellview(type Verilog for Verilog and type VerilogA for Verilog-A). For running pure digital simulations I recommend to create the verilog code and the testbench in ModelSim, test the code there and after it is proven to work, copy the code to the Cadence editor.

3.2 Modelsim

Modelsim is a HDL simulation environment, to start it, change the working directory of a terminal by typing:

Terminal
```
cd HDL_Lab_UMC65/implementation/umc_65_ll
```

Then open the modelsim start script:

Terminal
```
./modelsim.sh
```

Now ModelSim should open. Create a Project by clicking on File > New > Project and enter a name, for example Clock_generator and click OK.
A new window opens and you can add items to your project. If you have already opened a new project and want to add later files to your project, right click in your project tab and select `Add to project` and `new File`. Give the file a name e.g. `CLK`. Be careful you select a verilog file(not vhdl) and click OK. Now the editor window opens where you can built a simple clock generator like shown in figure 13.

```
2   `timescale 100ns/100ps
3   module CLK (clk);
4
5           output clk;
6           reg clk;
7           initial
8           begin
9           clk = 1;
10          end
11
12          always
13          begin
14          #10 clk =~clk;
15          end
16  endmodule
```

Figure 13: Verilog code for the clock generator.

After you have entered your code to the editor click on the save button on the top of the ModelSim window. Then add a second file to our project, this time the testbench, CLK_Testbench. The required code is given in figure 14.

```
2   module CLK_Testbench;
3
4           wire clk;
5
6           CLK dut(
7                   .clk    (clk)
8                   );
9
10  endmodule
```

Figure 14: Testbench clock generator.

Also save this file. To compile the written code, go to Compile > Compile All. If compiling was successful the message `Compile of CLK..v was successful` appears on the bottom in ModelSim. If not, you can double click on the error message to get further information about where the error is located. It is best to compile often, to find error in the beginning of coding. After you have finished compiling, you can start the simulation with Simulate > Start Simulation. In the opening window select your testbench, like shown in figure 15 and click OK.

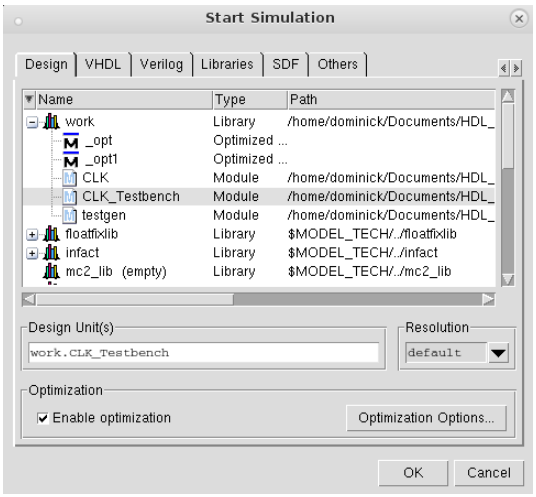

Figure 15: Start simulation window.

The `Sim` tab opens. Right click on your testbench in this tab and select Add to > Wave > All items in region. And now you can start the simulation by Simulate > Run > Run 100. Then the wave tab should show the waveform output of you clock generator like shown in figure 16.

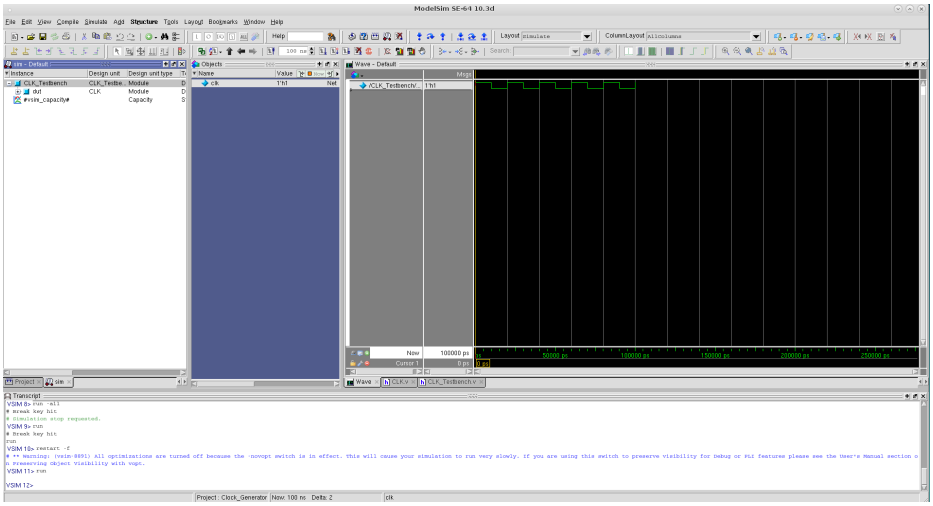

Figure 16: Wave tab in modelsim.

3.3 Synopsis Design Vision

After you have written synthesizable code, you have to synthesize your filter design.
Therefore, you first have to tell Design Vision how your top module and your other verilog files are named. Open the script `02-load_design.tcl` in `home/Your_Group/HDL_Lab_UMC65/implementation/umc_65_ll/syn/script`. Change the names in the two lines showed below:

02-load_design.tcl
```
set TOP_LEVEL_MODULE    “YOUR_TOPMODULE_NAME“

set FILE_LIST    {Your_Verilog_Files.v}
```

Then you have to tell the program what is your clock signal and what is you reset signal. The standard reset signal defined in the script is the signal `rst_n` which stands for a low active reset. You have to use the same signal name in your design as in the script, so change the lines shown below, or update your verilog design.

05-constrain.tcl
```
# set as ideal net for synthesis
set_ideal_network [get_port rst_n]

# Async reset assertion timing is not important (only the de-assertion is important)
set_false_path -fall_from [get_ports rst_n]

# model latency:
set_ideal_latency -max 0.0 [get_port rst_n]

# model transition:
set_ideal_transition -max 0.03 [get_port rst_n]
```

Same applies for the clock signal in your design. The standard name is `clk`, so change the line below, or adapt you verilog files. Design Vision optimizes your design according to the given clock period. This can be also changed in the `05-constrain.tcl` script.

05-constrain.tcl
```
create_clock [get_ports clk] -name "CLOCK" -period 10.0
```

In this example Design Vision tries to optimize your design for a clock period of 10ns. If you use higher frequencies, Design Vision optimize your design towards higher speed, which will result in a higher area and energy consumption of the circuit. After you have adapted the scripts to your design you can start Synopsis Design Vision. To start Synopsis Design Vision open a terminal and change the working directory:

Terminal
```
cd HDL_Lab_UMC65/implementation/umc_65_ll
```

Then open the Synopsis start script enter:

Terminal
```
./design_vision.sh
```

In Design Vision click on File > Execute Script, navigate to the script folder and execute the `synthesis.tcl` script. The design process should take a few minutes. After finishing the design process, your design will be showing up in the `Logical Hierarchy`. By marking your design and clicking of Schematic > New Schematic View you can view the schematic of your module.
Further reports of your design are in the folder `reports`, there you can find the size of your module, the power consumption and so on.

4 Deliverables

4.1 Files

Make a folder named (xx is your group number)

hdllab2018_xx

and place your files in it. Then pack everything in a zip archive and email it to

Dominic.Korner@ies.tu-darmstadt.de

not later than 03.08.2018(Friday) 23:59:59. Use the following subject line

HDL Lab2018GroupXX Data

Make the report named

hdllab2018_report_xx.pdf

Use the following subject line

HDL Lab2018GroupXX Report

and email it not later than 09.08.2018(Thursday) 23:59:59.

4.2 Written Report

4.2.1 Report Template

Use either Latex or Word to prepare the report. Corporate Design templates for both platforms are available on the TU-Darmstadt homepage. However, you could use a suitable Word or latex template from another lab/seminar/thesis you have undertaken at our institute. This is a good occasion to learn LaTeX, because you may need it for your thesis. After the lab you are given sufficient days to write your report for exactly this reason.

4.2.2 General Remarks

Basically, your written report for this lab should discuss in detail how you did the design (schematics and written code!) as well as the respective results you achieved, e.g. what design decisions you made and why, the problems you had and the solutions you found, what specific techniques you applied, how you value your results, your conclusions etc. Include the schematic and code of the individual sub-circuits of each component. Also name all relevant results of your ADC as speed, offset, SNR and include plots of the output. Which architecture have you chosen and why?
Provide all data of the filter you have designed, like filter topology, frequency response, SNR, synthesized area, maximum speed, power consumption etc. Provide plots of the spectrum of a signal before and after it is filtered. Please make sure that the report does not exceed 20 pages!
On the last day of the lab take all the necessary waveforms with you to write your report.
Sad but true: Plagiarism is embarrassing! Point out your own as well as others' original work. Cite when you use other people's results (even when paraphrasing). Ask if in doubt. All reports and source code at IES are checked with automated tools.

5 Examination - HDL Lab

The examination is mandatory, oral and takes place in groups of two. It lasts 30 minutes and the date will be defined in the first week of the lab. Be sure both team members know everything of the designs and are able to explain it. Bring a printed copy of your written report to the examination!