Microelectronic Systems

Lab 5

Deep in a CMOS structure: using Spice for characterizing cells

May 13, 2022

Introduction

During this laboratory experience you will learn how a **library gate** is described and how to carry on a **Spice simulation** for a detailed gate characterization.
As usual, copy the needed files from the following directory:

```
prompt> cp -r /home/repository/ms/cap5/* .
```

You will notice that two directories are present: *src* and *lib*. Please, work in the former directory, src.

5.1 Characterizing a library gate

In the following, you will analyze the performance of a **NAND2 gate**. In order to do that, consider the following instructions:

1. First of all, prepare your shell for ELDO simulations by setting the correct environment variables:

```
prompt> seteldo
```

2. Now read carefully the netlist *nandHS.sp*.

The netlist will allow you to simulate a NAND cell designed by a foundry and described as a sub-circuit, which takes into account its real layout. In particular, all the *parasitic parameters* are included in the description.

Note that the NAND sub-circuit is not included inside this file: a reference to it is given with the instruction *.include "$ST_HSPICE_LIB/CMOS013.spi"* and the NAND is directly instantiated in the same netlist. Could you tell in which point of the netlist?

3. Consider then the file **"../lib/CMOS013.spi"** and read it.

Notice that not only **W** and **L** are passed to the NAND model, but also the parameters **AD**, **AS**, **PD** and **PS**. These are respectively the *drain area*, *source area*, and *drain* and *source perimeters* used for computing the parasitic capacitances C_{DB}, C_{SB}, C_{GS} and C_{GD}.
More in detail, you can observe that inside the definition of the NAND sub-circuit there is a reference to the two MOS models, **ENLLGP_BS3JU** for the NMOS transistor and **EPLLGP_BS3JU** for the PMOS one. Only two parameters are passed to these models, the transistor width and its length.

Let's take a minute to analyze their descriptions:

4. Open the file:

```
prompt> emacs ../lib/mos_bsim3_HS.lib
```

   Skip all the definitions of the various parameters and look for the line containing the description of the NMOS model, "ENHSGP_BS3JU" *(row 700)*.

As you can see, there are many parameters that can be defined. Some of these can be provided while describing the netlist, others are computed through complex expressions that take into account the simulation conditions *(e.g. temperature)* and process variations. You are going to use the model in the simplest way, i.e. passing only the transistor values W and L. Be sure to have understood the details of the netlist description. If not, please ask.

Question 1
Sketch on paper the NAND sub-circuit, expressing the names of the internal nodes and neglecting the other parameters. Draw also the external structure which calls the sub-circuit with the node names. Note the definition of the input waveform and the use of the **.measure** instruction: what does it perform?

5. Now you can proceed with the simulation. Just type:

   `prompt>` **eldo nandHS.sp**

   and wait for the process to stop. You should read on the shell *"current simulation completed"*.

6. If you type a **ls** command, you can note in the working directory that the simulator has created a few files. The one with extension **.wdb** contains data that the *waveform viewer* can process. You can start this program by typing:

   `prompt>` **ezwave &**

7. Open the design: from the main menu select File→Open and inside the new window choose **nandHS.wdb**, then click *Open*. You have now a simulation manager on the left; in particular, focus on the transient menu *TRAN*. Click on the + icon: you can see the currents and voltages you are interested in. Double click on the voltages $v(ina)$, $v(inb)$ and $v(out)$, present on the nodes *ina*, *inb* and *out*, to display them on the right waveform viewer.

8. You are ready to measure the *50% delay* between input and output. Start by choosing the **input rising case**. First of all, you have to superpose the two waveforms: simply select the signal label $v(out)$ on the right, and drag it on the $v(inb)$ area. Zoom in by clicking the + lens, present on the top bar menu, and shift the window until you have a clear view of the input and output waves. Now, add a cursor using the *F5 key* and drag it in order to place it on the 50% of the input signal variation. Then, add another cursor at the position corresponding to 50% of the output transition. At the bottom of the new cursor you have now the *time difference dx*. You can perform the same operation to determine the *falling time*, and *both the delay and rising time of the second transition*, by properly using the zoom feature to move through the whole waveforms.

9. Finally, compare your measures with the ones that Eldo has automatically obtained considering the **.measure** instructions and which are provided inside the netlist: open the file **nandHS.chi**, search for the keyword *EXTRACT INFORMATION*, and read the measurements that follows. Do they correspond to yours? Please, keep in mind that inside the Spice netlist you are requested to add a measure command.

Suggestion: If you want to save the waveforms into a file, select the **File→Export** and choose the filename.

Summary of what is requested

Spice netlist, measures on output delay *(Tp_{HL} and Tp_{LH})* and rise/fall time performed by you *(write them inside a text file)* and by Eldo. Output waveforms.

5.2 Characterizing a gate for output load

In this section you will characterize the previous gate, considering the variation of two parameters: the **output load** and the **input transition time**. You will measure not only the *delays* but the *power consumption* as well, by measuring the current peaks.

To carry out the task, consider the following steps:

1. Read the file **nandHScharLoad.sp** and analyze where the differences are with respect to the previous file.

Suggestion: If you don't understand a command, you can refer to the *hspice manual* by typing:

`prompt>` **helpeldo &**

Inside the netlist you should note the use of the *sweep* option with the *tran* command. It allows to repeat the simulation using different values of the load parameter, defined inside a table. Moreover, also *dummy generators* are present: they are DC voltage generators set on a value of 0V, which allow simply to measure the current flowing in the node.

2. Now, you can simulate it:

   `prompt>` **eldo nandHScharLoad.sp**

   Measure the *output delays*, *transition time* and *peak currents*. In order to obtain all the data you need, you have to complete the delay measure command. Take a look at the current measure instructions and try to understand if they are suited to your case *(minimum and maximum should be coherent with the current signs)*. Analyze also the resulting data inside the **.chi file**. Please notice that you are repeating the simulation many times and so you must search iteratively for many groups of measures in the same .chi file.

3. Then, open the new plot file using the **File Open** menu and selecting the correct file. Plot the following quantities: the voltages **v(inb)** and **v(outbis)** *(v(ina) is fixed)*, and the **currents flowing** through the **vdummy_vdd**, **vdummy_gnd** and **vdummy_c** generators.

To facilitate reading, drag the curves of the voltages on a unique plot and do the same for the currents, but inside another plot. In this way, the different quantities are comparable. Note that the simulation results are superposed: surfing on the waves you can see the index corresponding to the value of the input parameter.

Question 2

Try to explain the different behavior of the currents when changing the load, and especially the different contribution among *Vdd current*, *Gnd current* and *Cload current* in the various transition cases.

Summary of what is requested

Netlist, waveforms *(voltages and currents)*, measured values *(write them inside a text file)*. Explanation of different currents on different nodes in each transition *(again, inside the text file)*.

5.3 Characterizing a gate for transition time

At this moment, you are ready for characterizing the *performance* of the considered gate while changing the *input transition time*. Copy the content of the file **nandHScharLoad.sp** into a new file **nandHScharTran.sp**. Inside it, change both the **.data** values and labels using the values present at the first index of the **table_5**, defined by the foundry for characterizing the gate.

This table is normally given in a *Look-Up-Table* format:

```
lu_table_template( table_5 )
{ variable_1 : input_net_transition ;
  variable_2 : total_output_net_capacitance ;
  index_1 (" 0.0048, 0.1088, 0.2608, 0.5248, 1 ");
  index_2 (" 0.004, 0.012, 0.028, 0.08, 0.16 ");}
```

Notice that, inside the **.data vector** you have to change the parameter name from *load* to *t_tran*, while in the **values vector** you have to put the *input_net_transition* values.

Simulate and measure the output performance. Also in this case, you must complete the *delay measure command* in order to obtain all the data you need.

Summary of what is requested

Netlist, waveforms, measured values.

5.4 Comparing different gate sizing

In the previous cases you have considered a single NAND gate, optimized for driving up to a maximum load of 0.16fF *(this is the last value present inside the table, even if we forced higher loads for exercise)*: the other load values are meant for lowering *net* or *gate capacitance* that could be connected to the output node.
In the case a higher load is to be driven, a different gate should be used: each library has many versions of the same gate, each optimized for driving a specific value of maximum capacitance.

You will now analyze the differences between the smaller NAND *(with a driving capability of a X1 load)* and the bigger NAND *(X8 load)*, the latter optimized to optimally drive a maximum capacitance of 1.28f. You will simulate both the gates using as output capacitance **0.05f** and **50.0f** *(a higher capacitance is considered for enhancing the results)*. So, follow the instructions below:

1. Read the two files: **nandHScharMaxLoad.sp** and **nandHSX8MaxLoad.sp**. Read the two netlist and understand the differences among them and also the commands given. You can take a look at the different sizing parameters inside the file **"../lib/CMOS013.spi"**.

Before you proceed, complete the **delay** and **current measure commands** in order to obtain all the data you need.

2. Simulate both the gates and superpose the obtained waveforms *(open both the wdb files, double click on homologous signals and drag them in the same graph for easily comparing them)*.

Question 3
What happens to the values of delay and power dissipation? Check the output measure and analyze the advantages and disadvantages in using the two gates.

Summary of what is requested

Netlist, waveforms *(both current and voltages)*, measured values *(both current and voltages)*. Explanation of the different behavior *(write your considerations inside a text file)*.

5.5 Comparing high speed and low leakage optimization

Now, you are required to analyze the same differences as before, but using gates that have been optimized for a **low subthreshold power dissipation**. In the library that you are considering there are two NAND gates, named **ND2LL** and **ND2LLX8**, each supporting a specific fan-out. Proceed as follows:

1. Copy the netlist you have just simulated into two new files, **nandLLcharMaxLoad.sp** and **nandLLX8MaxLoad.sp** respectively, and change the reference to the *Low Leakage gates*.
2. Simulate these two netlists and compare the new results you have obtained.

Question 4

Compare these results with those obtained from the two previous gates, as well by superposing the four waveforms. What's happening to the delay? What about the current? Which is the percentage of advantage and/or disadvantage in using the **LL gates** with respect to the **HS** ones?

Summary of what is requested

Waveforms for the four simulations superposed and measured values. Percentage variation between HS and LL for the X1 and the X8 cases *(write them inside a text file)*. Explanation of advantages and disadvantages *(again, inside the text file)*.

5.6 Characterizing a Flip-Flop

In this last section you will simulate the D-FF *FD1QLL*, whose schematic is depicted in Figure 5.1, and force a **violation of set-up and hold times**. Keeping this in mind, read the two files, **ffdH.sp** and **ffdSU.sp**. The former is used to characterize the hold time, while the latter is considered for the set-up time.

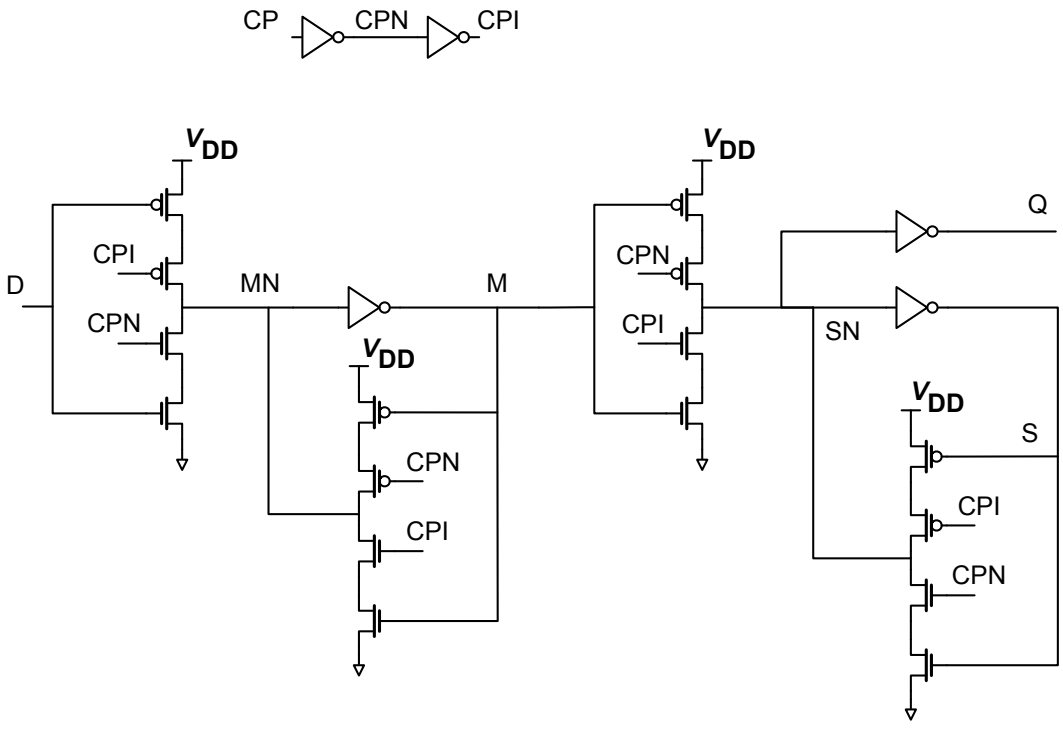

Figure 5.1: Schematic of a D Flip Flip

In both the netlists, a **.alter** instruction is used for reproducing both the HL and the LH output situation. Each time a **.alter** instruction is used, the whole circuit and the related stimuli are reproduced. The variations included after the **.alter** instruction affect the netlists and a new simulation is carried on. This means that you will have many simulation outputs, proper indexed with the associated .alter phase.

In order to perform the analysis, proceed following the instructions reported below:

1. Analyze the netlists you are given and the instructions used to force the violations.
2. Simulate both the netlists and try to understand what happens. Note that for each netlist you will have two simulations *(HL and LH)*. Sweep among the signals, half of them are for the HL transition, the remaining for the LH one.

Now, you are able to characterize both the set-up and the hold times for the two netlists.

Summary of what is requested

Netlists for the two cases, output waveforms of the critical points *(MN, M, CPI, CPN, CK, SN, D, Q)*, where the set-up and hold times are violated for the two transitions HL and LH. Comment inside a .txt file the waveforms where these times are violated.