CHAPTER 5

Deep in a CMOS structure: using spice for characterizing cells

In this chapter you will learn how a library gate is described and how to carry on a spice simulation for a detailed gate characterization.

Copy all the files:

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

You will have two directories: src and lib. Work in directory src.

5.1 Characterizing a library gate

We want to analyze the performance of the NAND2 gate. First of all prepare your shell for ELDO simulations by setting the correct environment variables:

```
prompt> seteldo
```

Now read netlist nandHS.sp carefully.

The netlist will allow you to simulate a nand designed by a foundry and described as a subcircuit which takes into account its real layout. This means that all the parasitics are included in the description. Note that the nand subcircuit is not included in this file: a reference to it is given with the instruction *.include '$ST_HSPICE_LIB/CMOS013.spi'* and the nand is directly instanciated in this very netlist (in which point of the netlist?).
You can see the file containing the small library we are going to use hereinafter: **../lib/CMOS013.spi**. Note that not only W and L are passed to the NAND model, but AD, AS, PD, PS parameters as well. These are the drain area (AD), source area (AS), and drain and source perimeters (PS, PD) used for computing the CDB, CSB, CGS and CGD parasitics capacitances.

In the definition of the nand subcircuit there is a reference to the MOS models: *ENLLGP_BS3JU* is for the NMOS transistor, while *EPLLGP_BS3JU* is for the PMOS one. We are passing to this model only two parameters, transistor width and lenght. Let's take a minute to see such transistor model. Open directly the file containing its description:

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

skip all the parameters definitions ad search the line containing the NMOS model ENHSGP_BS3JU (row 700). As you can see there are many others parameters that could be defined; some of these could be given while describing the netlist, others are computed within the model on the basis of complex expressions that take into account simulation conditions (e.g. temperature) and process variations. We are then using the model in the simplest way (passing transistor W and L).

Sketch in a figure the nand subcircuit with the internal node names (neglect parameters) and the external structure calling the subcircuit with the node names. Note the input waveform definition and the use of the **.measure** instruction: what does it perform?

Now, if you have understood the netlist description (if not, please ask ...!) you are ready for simulation. Just type

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

and wait for the simulation stopping. You should read on your shell *current simulation completed*. If you type a **ls** command, you can note in the directory that the simulator has created a few files. One is the **.wdb** one: it contains the data that your waveform viewer can process. You can start it by typing:

`prompt>` **ezwave &**

now open the design: File→Open select nandHS.wdb in the new window and click Open. You have now a simulation manager on the left; in particular you have the transient menu TRAN. Click on the +: you have the curent and voltages we are interested in. Double click on $v(ina)$, $v(inb)$ and $v(out)$: the voltages at the ina, inb and out nodes are being displayed on the right waveform viewer. You are ready now to measure the 50% delay between input and output. Chose the input rising case first. You must first of all superpose the two waveforms: simply select the signal label $v(out)$ on the right and drag it on the $v(inb)$ area. Zoom by clicking the + lens on the top bar menu and shift the window until you have a clear display of input and output waves. Now add a cursor using the F5 key and drag it so that it reaches 50% of input variation. Now add another cursor and drag it until it reaches 50% of the output transition. At the bottom of the new cursor you have now the time difference dx. Now you can perform the same thing for the falling time, and for the delay and rising time of the second transition detail, using the zoom features to go through the whole waveforms.

Now compare your measures with the ones eldo automatically performed following the **.measure** instructions given in the netlist: open the **nandHS.chi**, searchi the keyword: EXTRACT INFORMATION, and read the measures below . Do they correspond to you ones? Mind that in the spice netlist you are requested to add a measure command!

If you want to save the waveform file select the **File→Export** and choose the name.

Summary of what is requested

Spice netlist, measures on output delay (Tp_{HL} and Tp_{LH}) and rise/fall time performed by you (text file) and by eldo. Output waveforms.

5.2 Characterizing a gate for output load

We want to perform a characterization of the gate considering the variation of a few parameters: output load and input transition time. We will measure not only delays but power as well, by picking the current peaks. Read file **nandHScharLoad.sp**, analyze where are the differences with respect to previous file. If you don't understand a command you can refer to the hspice manual by typing:

`prompt>` **helpeldo &**

In the netlist you should note the use of the *sweep* option in the *tran* command which allows to repeat the simulation using different values of the load parameter defined in a table. Moreover we also added dummy generators: they are DC voltage generator of value 0V, and allow simply to measure the current in the node. Now you can simulate it (eldo nandHScharLoad.sp) and measure the output delays, transition time and peak currents (you must complete the delay measure command in order to obtain all the data you need). Note 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). Note the resulting data in file .chi (as in this case we are repeating the simulation many times in the .chi file you must search iteratively for many groups of measures). Now open the new plot file using the **File Open** menu and selecting the correct file. Plot: v(inb) and v(outbis) as voltages (ina is fixed), and the currents flowing through the *vdummy_vdd*, *vdummy_gnd* and *vdummy_c* generators.

For an easy reading drag the voltages on the same plot and the current on a single plot as well, so that they 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.

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

Summary of what is requested

Netlist, waveforms (voltages and currents), measured values (text file). Explanation of different currents on different nodes in each transition (text file).

5.3 Characterizing a gate for transition time

Now you are ready for characterizing the performance while changing the input transition time. Copy the file nandHScharLoad.sp in a new file **nandHScharTran.sp**, change the **.data** values and labels using the values in the first index used in the **table_5** used by the foundry for characterizing this gate. It 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 ");}
```

Note that in the the .data vector you must change the parameter name from load to t_tran and in the values vector put the input_net_transition values. Simulate and measure the output performance. 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 we used a single nand gate optimized for driving up to a maximum load of 0.16fF (the last value in the table even if we forced higher loads for exercise): the other load values are for lower net or gate capacitance that could be connected to its output node.
If an higher load is to be driven a different gate should be used: each library has many views of the same gate, optimized for driving other maximum capacitance values.
We now analyze the difference between the smaller nand (X1 load) and the bigger nand (X8 load) which is optimized for optimally driving a maximum capacitance of 1.28f. We will simulate both the gates using as capacitance 0.05f and 50.0f (we are using an higher capacitance for enhancing the results). Read the two files: **nandHScharMaxLoad.sp** and **nandHSX8MaxLoad.sp**. Read the two netlist and understand the differences and the commands given; you can look at the different sizing in the file **../lib/CMOS013.spi**. You must complete the delay and current measure command in order to obtain all the data you need. Simulate both and superpose the waveforms (open both the wdb files, double click on homologous signals and drag them in the same graph for easily comparing them). What does it happen to 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 (current and voltages), measured values (current and voltages). Explanation of the different behavior (text file).

5.5 Comparing high speed and low leakage optimization

Now we want to analyze the same difference as before but using gates that have been optimized for a low subthreshold power dissipation. In the library that we are using there are two nand gates. Their names are: **ND2LL** and **ND2LLX8** for the two fan out types. 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. Simulate these two netlists and compare these new results. Compare them with the two previous ones as well by superposing the four waveforms. What's happening to the delay? And to the current? Which is the advantage and/or disadvantage in percentage in using the LL gates with respect to the HS?

Summary of what is requested

Waveforms for the four simulations superposed; measured values; percentage variation between HH and LL for the X1 and the X8 cases (text files). Explanation of advantages and disadvantages (text file).

5.6 Characterizing a Flip-Flop

We want now simulate a FF and force a violation of set-up and hold times. Keeping this in mind read the two files: **ffdH.sp** and **ffdSU.sp**. The first is used for the hold time and the second for the set-up time characterization.
In both the netlist a **.alter** instruction is used for reproducing the HL and the LH output situation. Each time a **.alter** instruction is used, the whole circuit and stimuli are reproduced, the variations included after the **.alter** instruction affect the netlists and a new simulation is carried on. This means

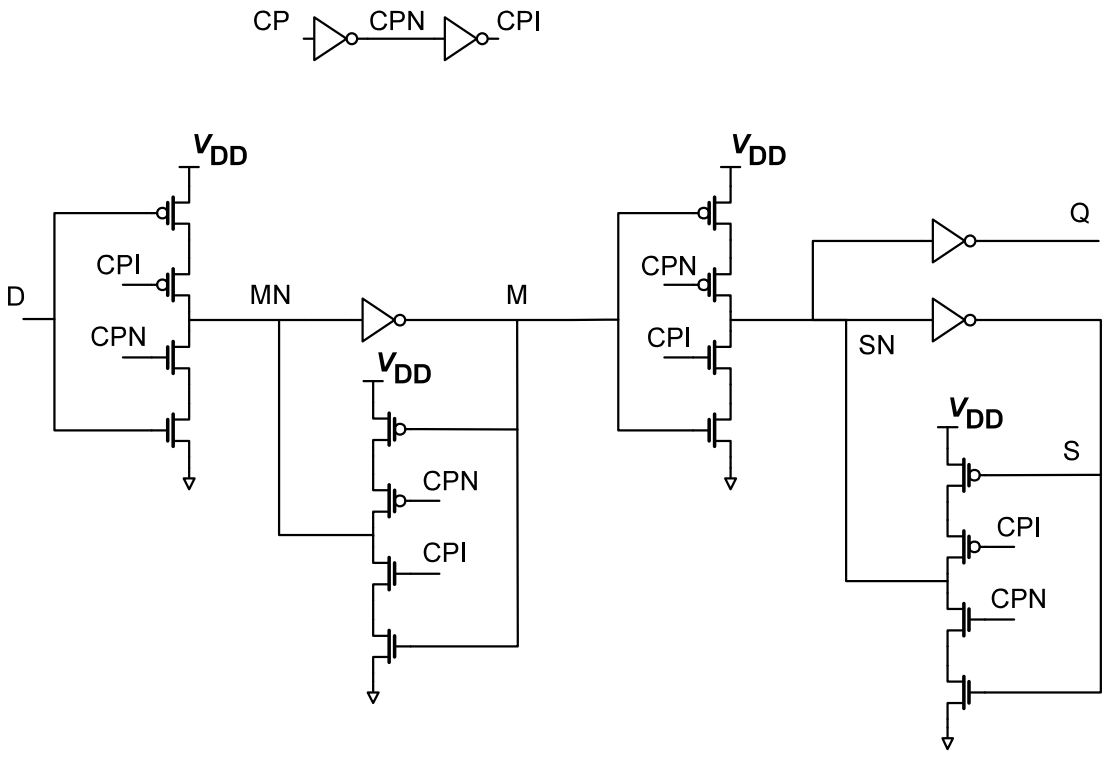

Figure 5.1: Schematic of the flip flip D

that you will have simulation outputs with indexes related to the current .alter phase. Analyze the netlist given and the instructions used to force the violations. Fig. 5.1 shows the schematic of the flip

flip D *FD1QLL*. 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. Now you are able to pick up in 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, CP, PN, CK, D, Q*), where the set-up and hold times are violated for the two transitions HL and LH. Comment in a .txt file the waveforms where these times are violated.