Politecnico di Torino

Collegio ETF - ICM

Low-Power Electronic Systems
Lab 5

Mariagrazia Graziano, Fabrizio Riente, Marco Vacca

March 30, 2023

Leakage: using spice for characterizing cells and pen&paper for memory organization

In this lab you will analyze power contributions, with a look to leakage power, from two points of view.

First you will simulate at spice level a NAND gate based on transistors optimized for low leakage and compare it to a similar one where transistors are optimized for high speed.

Second you will use a memory generator and compare the values of memory blocks of different types and organization.

Copy all the files:

`prompt>` **cp -r /home/repository/lowpower/ese5 .**

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

5.1 Characterizing a library gate TL & TP)

We want to analyze the performance of the NAND2 gate optimized for high speed in order to have a reference point for the low leakage implementation. 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: *ENHSGP_BS3JU* is for the NMOS transistor, while *EPHSGP_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. This is the high speed (HS) 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). Search the occurrences of the term "vth", related to the threshold voltage: do you appreciate how complex is this model?

Now go back to the cell. 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 (PWL) and the use of the **.measure** instruction: what does it perform? Notice that there is an incomplete command, i.e. one timing value is missing. Can you insert the correct fourth statement?
Now, if you have understood the netlist description 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*. In the same shell, if you go up, you can find the "TOTAL POWER DISSIPATION:" value. Annotate it as later you will compare with other cases. 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, double click on it and you will see the current and voltages we are interested in. Double ckick 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 $v(out)$ 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**, search the keyword: EXTRACT INFORMATION, and read the measures below . Do they correspond to you ones? Mind that in the spice netlist you were requested to add a measure command!

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

Remark point

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

5.1.1 Measuring the threshold voltage

As we are interested in leakage power we want to keep track of the threshold voltage for the NAND transistors in all the simulations we are going to do.
The simplest way is to use the command

.print VT(nameofthetransistor)

that is already added at the end of your netlist: uncomment this line

.print VT(XNAND.XMN0.M1) VT(XNAND.XMN1.M1) VT(XNAND.XMP0.M1) VT(XNAND.XMP1.M1)

As we don't need the values in all the simulation steps, we just re-run a simple dc simulation. To do this, comment the .tran command and uncomment the .dc command. Save the file, and run again the

simulation. Finally open the file with a .chi extension, and look (starting from the bottom) for rows in which you have:
1: VT(XNAND.XMN0.M1)
2: VT(XNAND.XMN1.M1)
3: VT(XNAND.XMP0.M1)
4: VT(XNAND.XMP1.M1)

Just above you have the values for the VT of the four transistors. Notice the different values for P and N: are they as you expected? Notice the values of the two NMOS-VT and of the two PMOS-VT: are they identical two by two?
Remember to repeat this operation in all the circuits we are going to analyze.

Remark point

VT at DC for the four transistors.

5.2 Characterizing a gate for output load (TL & TP)

We want to perform a characterization of the gate considering the variation of output load. We will measure not only delays but power as well, by picking the current peaks. Read file **nandHSchar-Load.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 &**

Once you have understood the netlist (note the use of the *sweep* option in the *tran* command and the new added dummy generators) 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 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.

Remark point

Netlist, waveforms (voltages and currents), measured values in the .chi file. Explanation of different currents on different nodes in each transition (text file), total power dissipation in all the cases.

5.2.1 Threshold voltages

Exactly as before, comment the .tran line, uncomment the .dc and the .print line and rerun the simulation: which are the resulting voltages? What did you expect with load variation?

Remark point

VT and comments.

5.3 Comparing different gate sizing (TL & TP)

In the previous cases we used a single nand gate optimized for driving up to a maximum load of 0.16fF: 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 0.06f and 60.0f (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. Notice the total power dissipation of the two gates: is the ratio between the values of the two gates near to what you expected?

Remark point

Netlist, waveforms (current and voltages), measured values (current and voltages). Explanation of the different behavior (text file). Total power dissipation.

5.3.1 VT

Again, comment the .tran line and uncomment the .dc and .print lines; rerun the simulation and find the threshold voltages. Are they changing?

Remark point

VT for the two cases. Comparison with previous gate.

5.4 Comparing high speed and low leakage optimization (TL & TP)

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? Annotate the total power dissipation and compare it with the former results.

Remark point

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

5.4.1 VT

Again, comment the .tran line and uncomment the .dc and .print lines; rerun the simulation and find the threshold voltages for the two cases. Are they changing with respect to the high speed gates?

Remark point

VT for the two cases. Comparison with HS cases.

5.5 Temperature dependency (TL & TP)

Use the netlist nandLLcharMaxLoad.sp, comment in the .data command the small load value (0.06f) just to simplify. Add this line after the .tran command:

.temp -50 0 50 100 130 160 190

Run the simulation again: a sweep with all the previous temperature values will be performed. When finished, scroll the shell and get all the values of the total power. Insert them in a file named TEMP with this format:

-50 value
0 value
50 value
....
....

Save it and then from a shell in the same directory type:

gnuplot plottemp

a plot appears with the power behavior with temperature; if you type enter on the shell two files will be generated: TEMP.eps and TEMP.png. You can open the .png directly using display TEMP.png. What do you notice? Is it what you expected?

Now let's analyze the same thing but for the VT detail. Open the .chi file and look for the VT data. You will find a matrix with the four VT values as a function of temperature. Copy and paste these values using the same format in a new file TEMPVT and save it. In a shell type:

gnuplot plottempvt

and again a plot of the VT variation with temperature appears and a file is generated. I this result what you expected?

Remark point

Total power and VT data depending of T, final plots, comments.

5.6 Power consumption variation with power supply voltage (TL & TP)

Finally, we want to perform an analysis of how the power consumption change with the power supply voltage. Copy and paste the file nandHScharLoad.sp, rename the new file **nandHScharVoltage.sp**. Now instead of defining a variation of load, we will define a variation of voltage. Change the parameter used in the vectorload variable, from **load** to **alim**, so that the sweep is performed on the supply

voltage instead of the load. Then insert the voltage values inside the vector, from 0.7V to 1.4V with a step of 0.1V.

Run the simulation, open the *nandHScharVoltage.chi* files and search for the keyword **POWER**. You will find the average power consumption calculated for each value of supply voltage. Now to plot the results you can exploit the simple gnuplot scripts used in the previous exercises. Create a file named *VOLT*, and insert the voltage values with the usual format.

0.7 value
0.8 value
0.9 value
....
....

Write the values of power consumption that you find in the .chi file. To plot the results you can modify one of the given gnuplot scripts, for example *plottemp*, substituting the new name of the file (*VOLT* instead of *TEMP*) and Voltage (V) instead of Temperature (C) as the x axis label, and then renaming the file for example as *plotvoltage*. Then you can execute the script from the command line to get the graph of the power consumption.

gnuplot plotvoltage

How is the variation of power consumption with the power supply voltage? How much is the increment of power consumption from 0.7V to 1,4V (2x increase in voltage)?

Remark point

Netlist, plot of the power consumption variation, comments.