High Speed CMOS VLSI Design

Problem Set 2

1: Device Characterization

The objective of this problem is to develop a script to characterize CMOS processes. You will then use it to obtain data on the class 0.8 micron process. We have provided a script that measures some of the parameters (written in Perl). The script calls SPICE decks, substituting various parameters for different measurements. We have also provided a few decks for the first questions to get you started. Read them carefully and follow a similar style when you are writing additional decks. The idea is not to spend all night running simulations by hand, but rather to practice writing SPICE decks and postprocessing programs. If you can't get your script to work, just hand in your SPICE decks which handle a single process corner or portion of each question. DON'T waste time running zillions of simulations by hand.

Most good circuit designers spend an unreasonable fraction of their time building scripts to automate data collection and processing. Your boss won't care what language you use to solve the problem, only that you get accurate and timely results. We don't care what language you use either; find one that is easy for you to use and natural for the problem you are solving. PERL is a favorite among many designers because of powerful string processing capabilities, but AWK, C, C-SHELL, or even COBOL are fine if you prefer. We've provided a good start in PERL, so it is probably most convenient to start by extending the script we've provided.

Problem Set 2

When you are done with this problem, you should have a handy tool to characterize any process you happen to encounter in a dark alley in the future. You will find that as a circuit designer, you frequently have to switch processes. Your script should let you quickly get a feel for the capability of the new process.

To do the assignment, you will need to download the class process file and the script provided from the class web page. Place the HSPICE .lib library files in your ~/lib directory. Invoke the script by giving it lambda, VDD, and a path to the process file. Lambda is half the minimum drawn channel length, 0.5u for this process.

```
char.pl 0.5u 3.3 ~lib/opCondition.lib
```

You may assume the following things about the library:

- The library defines TT, SS, and FF process corners
- The parameter 'Sup' must be set to the value of VDD before issuing the .lib statement

When characterizing gates, assume:

- The β ratio of gates is 2.
- Inverters have a fanout of 4 unless otherwise noted.
- All transistors use minimum channel length. Note: λ is actually 0.5 μm in this process, contrary to what is stated in the Process Overview. All gates are drawn 1 μm long. During manufacturing, the gates are shrunk to 0.8 μm. The HSPICE models take gate lengths drawn as 1 μm and compensate for the shrink to give correct performance results.
- Operate in the TT corner unless otherwise noted.
- Include diffusion parasitics where appropriate, assuming a good layout according to MOSIS design rules (i.e. 2 λ of uncontacted diffusion or 6 λ of contacted diffusion between transistors, 5 λ of contacted diffusion at the top of a stack). Don't fold your transistors.
- Measure delays from input crossing VDD/2 to output crossing VDD/2
- The input slope corresponds to the gate being driven by a FO4 inverter.

Your script should extract the following data and print it out in an easy-to-read format. Run it on the class process with Sup=3.3 and hand in your results.

1. Effective gate capacitance per μm of total transistor width for an inverter (for delay purposes). A SPICE deck is provided in the problem set directory which does this for you. Read it carefully and use it as the basis of decks you write for other problems. The deck uses the optimizer to adjust a real capacitor until the delay of an inverter driving the capacitor equals the delay of an equivalent inverter driving a larger inverter. The larger inverter and real capacitor therefore have the same effective capacitance for delay purposes.

2. Effective gate capacitance per µm of total transistor width for an inverter (for power purposes). Do this by measuring the current into the load and integrating it over the transition. .MEASURE has an INTEGRAL mode which may be helpful.
3. Effective diffusion capacitance per µm of width for contacted NMOS and PMOS transistors (for delay). Do this measurement by loading the circuit with the drain of an off transistor, then optimizing a linear capacitor as in part 1.

   Break your results down by gate overlap/µm, diffusion area/μm^2, diffusion edge/µm and total diff cap/µm for a wide contacted transistor.

   Find the results as average of rising and falling edges (diff cap depends on voltage)
4. % change / degree C for the capacitance (for delay) of an inverter from part (1). We have provided the script to do this for you (using the same deck from part (1)).
5. % change / volt VDD for the capacitance (for delay) of an inverter from part (1). Leverage the existing deck and script as much as possible for this part.
6. Results of (1) in FF and SS process corners. Remember that voltage and temperature also change with process corner.
7. Effective pullup and pulldown resistance/μm^{-1} of an inverter driven by a FO4 inverter. Do this by measuring the delay, calculating the capacitance (from your results in part (1), and dividing. For best results, build inverters with zero diffusion capacitance (because diffusion capacitance is strongly voltage dependent).
8. Effective pullup and pulldown resistance/μm^{-1} of a tri-state inverter with both series transistors simultaneously driven by a FO4 inverter. As usual, the output should be loaded by an inverter with four times the input capacitance of the tri-state. The inverter should in turn drive another inverter four times larger.
9. Rise, Fall, and Average delay vs. fanout of inverter driving FO1, FO2, FO4, FO6, FO8 (TT corner). Be sure to include appropriate diffusion capacitance when measuring gate delays (think about the layout to determine areas and perimeters).
10. Average delay of FO4 inverter in FF and SS corners. Remember that voltage and temperature are different in these corners.
11. Best fit for a and b in Average Delay = a + b * fanout for inverter from previous data (TT corner)
12. %change / degree C for the average delay of a FO4 inverter around TT corner
13. % change / volt VDD for the average delay of a FO4 inverter around TT corner
14. Rise, Fall, and Average delay of FO4 NAND and NOR with outer input switching. The outer input is the transistor closest to the power rail. Size your gates as you did in Problem Set 1 (for equal nominal drive strength). Remember to use fanout (C_{out}/C_{in}) of 4, not gain of 4.
15. Average delay of FO4 NAND and NOR gate in FF and SS corners, as simulated in the previous part.

16. Best fit to delay = a + b * fanout for 2-input NAND gate delay (TT corner). Find the fit for the inner input switching, the outer input switching, and both inputs simultaneously switching.
17. Best fit to equation above for 2-input NOR gate delay (TT corner). Do this for the inner input switching, the outer input switching, and both inputs simultaneously switching.

You should refer to the notes about recommended process characterization techniques (in a separate handout) while writing the script. Use the M-factor to create larger gates.

2: Analysis of Characterization Results

a) We found the gate capacitance per lambda (for delay) from 1.1 to be 2.19 fF/µm when the output drives just an inverter and 1.83 fF/µm when the output drives a FO4 inverter which is in turn loaded with another FO4 inverter (the way the spice deck provided is configured).

i) Which capacitance would you want to use for hand estimation? Why?

ii) Why are the results different?

iii) Why is capacitance for delay different than capacitance for power?

iv) Does the capacitance for delay differ significantly across temperature or voltage? (Use your measurements from 1.4 and 1.5.)

b) We found the effective R per μm^{-1} of an NMOS transistor in an inverter to be smaller when driven by a step input than when driven by a FO4 inverter.

i) Which resistance would you want to use for hand calculation? Why?

ii) Why are the results different?

c) Look at the resistance / μm^{-1} which you measured for an inverter and for a tri-state (skip this part if you can’t extract the data properly).

i) Why don’t the resistances simply double when two transistors are placed in series?

ii) Why does the NMOS series resistance increase by a smaller factor than the PMOS series resistance?

d) We found the following data on average inverter delay in the TT corner:

Fanout	Delay (ps)
1	173
2	250
4	420

Fanout	Delay (ps)
6	577
8	743

The best curve fit is t = 90 + 82 * fanout. What is the worst case percentage disagreement between curve fit and actual data? Is the linear delay model (t = a + b*fanout) a good match with reality?

e) We found the following average delays for FO4 inverters, 2-input NAND gates, and 2-input NOR gates across process corners (from Problem 1, parts 10 and 15):

TABLE 1. Gate Delays

	Inverter	NAND2	NOR2
TT	420	500	689
FF	317	383	520
SS	807	998	1340

i) Express the NAND and NOR delays in terms of FO4 inverter delays in the same process corner.

ii) How well do the NAND and NOR delays track with inverter delays? Is FO4 delay a good process-independent metric for measuring delay of logic paths?

f) In the Logical Effort model, delay can also be written as a + b * fanout.

i) Show that the logical effort of a gate can be measured as b of the gate / b of an inverter.

ii) Using the a and b data you extracted on parts 11, 16, and 17 of problem 1, compute the logical effort of a 2-input NAND gate and 2-input NOR gate with the outer input switching. How do these logical efforts compare with what you calculated in Problem Set 1? Explain any discrepancies.