ECE/CS 6770 – Lab Assignment 1

Due 20 Jan 2017 via Canvas

1 Introduction

A good understanding of modeling and performance of transistors, gates, and systems is key to any VLSI "expert". An equally important skill set is to become a good *designer*, which is really a form of art. For this lab you will exercise both aspects: use your skill and creativity to create a fast and efficient circuit.

One of the most challenging aspects of circuit design is due to the cost in terms of both time and money to fabricate a design. If a chip design has a single failure, it may become uncompetitive or even completely non-functional. Thus we rely heavily upon simulation engines and design methodologies to ensure that our designs will be correct. Lab 2 will introduce a commercial design flow that will address correctness and productivity. For this lab, we will employ the most accurate simulation models we have to validate the quality of the design that we will create. Thus we will use a "spice" simulator to model the performance and power of our design.

The purpose of this lab is to:

1. Introduce you to an accurate simulation engine and models that will be used to evaluate the performance and correctness of your design.
2. Exercise creativity in employing your *designer's art* to create the best design possible.
3. Introduce various technology nodes and begin to give you some intuition for the differences in performance and power of modern VLSI processes.
4. Create a circuit optimization methodology.
5. Test a simple design structure.

2 Design and Art

The verb *design* in the Webster dictionary is defined as: *to create, fashion, execute, or construct according to plan.* As an engineer, we are constantly performing design to build products. This points out one key aspect of the design process: having a plan. *Productivity* in this class (and in the workplace) is very important. We will be employing numerous CAD tools to develop various projects. To be productive, you need to have a good plan for how you will learn the CAD tools, how to measure results of the projects, and how to perform design and optimization steps, and when you have converged to a satisfactory result. You also need to know when to quit. All of the steps in developing a product take time and effort. Putting in extended effort for a very small gain is a waste of your valuable time. So know when to quit, and where the most effective use of your talents can be employed. Note that this lab should not take very much time to complete if you have a good plan.

The noun form of the word "design" further elucidates a critical part of design that is really the key to success – but rarely addressed in engineering classes. I particularly like definitions 6 and 8 in Webster that read: *the arrangement of elements or details in a product or work of art*, and *the creative art of executing esthetic or functional designs*. Notice the word **art** that appears in both of these definitions. To me design has two main components. First is the skill or knowledge necessary to achieve the specification. This is what we spend most of our time on in school - learning the rules that allow us to have the knowledge and skill to create designs. The second part is what sets apart the outstanding designers: the ability to think creatively or artistically that results in an arrangement of elements that is far superior to other approaches.

I love VLSI because it provides a marvelous canvas and set of primitives that fosters the *art* of design. For the same reason I also love asynchronous circuit design as it cultivates design as an art form. In the end, the quality of the art or creativity of the designer gets measured and compared in mathematical terms such as the performance or power of a design. In general, I claim that employing the freedom of asynchronous design and doing it creatively and properly results in a 3× improvement over traditional design methodologies.

So in this first lab, you are asked to take a relatively simple function, and use some creativity in how you connect and size the gates in order to create a fast and efficient circuit.

3 Specification

In this lab we will create a circuit design for the following 5-input function:

$$f = abcd + ae \tag{1}$$

For this lab you will create a circuit for Eqn. 1 and optimize the design to create the smallest energy performance product $e\tau$ you can reach. You will evaluate the design running **HSPICE**. For this design you will come up with a test vector sequence that exercises the best and worst case rising and falling delay through your circuit. You will need to justify why these vectors are the correct vectors for testing the performance margins of your design. The results will be observed through both waveforms observed through **CosmosScope** and reports generated by HSPICE. You will have two designs that you simulate. One at the 32 nm process node, and one at the 130 nm process node. The minimum transistor width in your circuit will be 100 nm in the 32 nm process node and 300 nm in the 130 nm process node. The circuit node will drive a load that is 10× the signal with the maximum input load of your design You will create this load by tying the output to the gate of an n-type transistor with the appropriate width where the source and drain are tied to ground. (A 3,000 nm wide transistor for the 32 nm node.) You will report on the energy and performance of the design at these two process nodes.

This lab is to be performed independently. You are <u>not</u> to discuss circuit structure or optimizations with any of the other students in the lab. You may, however, discuss the usage of the CAD tools and help other students if they have questions with the tools.

4 Circuit Design

The goal of this lab is to create a circuit **design** that performs this function with the smallest *energy delay product*.

Engineering is a competitive field, where we build products to improve the quality of life and enjoyment of humanity. We as engineers are rewarded or penalized based on the quality of the results that we produce. As a customer we want to purchase the best product; so as an engineer we should strive to build the best product. This first lab will be an emulation of a competitive market place where each of us owns our own company and will be bringing the highly demanded function of Eqn. 1 to market. The goal of this lab is to create the *best* design. But the market has such a high demand that there is room for all of us to sell our product, with a minor reward and penalty for the outliers. Part of the grade for this lab will be based on the quality of the results. The student with the smallest $e\tau$ result will get the most points for that part of the assignment.

4.1 Transistor Design Hints

One of the freedoms we have with VLSI design is to create custom transistor structures that will best implement your design. A good transistor structure will be key to achieving good power and performance. You will therefore want to implement the circuit with *custom gates*. The best design will most likely have the fewest transistors. You will most likely want to implement complex gates that integrate both and and or structures into a single netlist. If you are unfamiliar with and-or-invert (AOI) gates, then please spend some time with the instructor or TA, as this will be an important aspect of the design.

The circuit may contain more than one stage to implement. The best number of stages can be approximated using *logical effort*. Please review this if you are rusty with this technology.

You will also need to size the design properly. Normally the best ratio of p-type to n-type transistors is 2:1, but you are free in this lab to use whatever ratio between the pull-up and pull-down trees as you deem best.

You will probably want to first design your circuit on paper before implementing and testing it.

Achieving the proper transistor structure and sizing is one of the primary deliverables of the lab and will directly determine the speed and energy efficiency of your circuit.

5 Circuit Netlist

You will then need to generate a circuit model that will be simulated in hspice. You can either generate the netlist by hand, or export it from a Cadence schematic. **Please have your two spice netlist files end in .sp when you turn in your results.** Note that a '*' character in the first column of a line is the comment command in spice.

5.1 Hand Generated Spice Netlist

The format for spice files was defined over 40 years ago. Each device must start at the first of a line, and the first letter indicates the type of model being defined. For this lab we will only use MOS transistor models. These models all start with the letter M followed by an alphanumeric identifier which uniquely identifies the MOS gate. The terminals are net names in the following order: drain, gate, source, and base. The bsim model name follows (either nmos or pmos in our models), which will dictate whether this is a p- or n-FET. These models refer to the parameters specific to the process node that you are using for the design. The length and width of the transistor are then specified. Modeling the parasitics is essential to an accurate simulation. The transistor parasitics are calculated from the area of the drain (AD), area of the source (AS), periphery of the drain (PD), and periphery of the source (PS). The ground net is always the net numbered '0' and can be specifically employed. Hence, the model for an inverter consisting of one n-FET and one p-FET follow:

```
*    D  G   S   B
M1 out in   0   0  nmos  L=65n W=180n AD=25f AS=55f PD=180n PS=540n
M2 out in vdd vdd  pmos  L=65n W=180n AD=55f AS=55f PD=360n PS=540n
```

Note that this is an inverter in the 65 nm process node, as the transistor length is 65 nm.

5.2 Cadence ICFB Netlists

You can draw your circuit schematic in Cadence, and then export a spice netlist. However, for this lab I request you draw the circuit by hand. This is somewhat tedious, for which I apologize, but will allow you to become more familiar with spice netlists and their formats. In the future you will probably want to draw schematics and then export them to a spice netlist.

5.3 SPICE Details

You will need to read the hspice manual to generate the netlist, test vectors, and results. The Cadence manual for spice netlists and simulation control can be found in:
`/uusoc/facility/cad_common/Synopsys/hspice_G-2012.06-SP1/hspice/docs_help/`.
The file `hspice_sa.pdf` is the simulation manual.

You need to make sure that the values of your transistor parameters are correct, such as the length, width and areas. Spice will automatically assign ohms to resistors, farads to capacitors, and henries to inductors. You only need to provide the numeric values. You can do so with scientific notation or use the following units (not case sensitive):

f	1e-15	k	1e3
p	1e-12	meg	1e6
n	1e-9	x	1e6
u	1e-6	g	1e9
m	1e-3	t	1e12

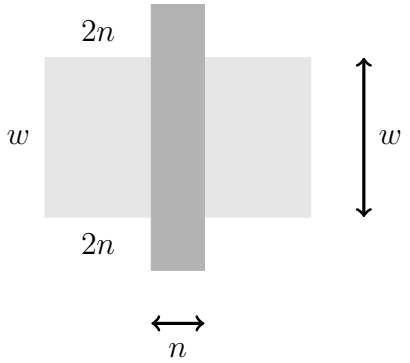

Figure 1: Calculating area and periphery of a MOS transistor, where w is the width, and n is the process node. For the periphery, w is only counted once since the gate side is part of the transistor channel.

In particular, note the difference between m, meg, and x. Spice is not case sensitive, so m and M mean the same thing – 1e-3!

5.4 Transistor Model Hints

Make sure the following all hold for your design:

1. The channel length parameter is correct for your design (130n or 32n)
2. The width of the transistors are correct for the model you are using.
3. The area and peripheral values for the source and drains can be approximated but should be on the correct order of magnitude for the size of the transistor that you are using. Cheating here by giving very small parasitics will result in a faster design ..., but you will not win the design contest unless your area and peripheral values are of the correct relative size. For this lab just use the following equations for to calculate the source and drains (see Fig. 1):

$$AS, AD = 2n \times w \quad (2)$$
$$PS, PD = 4n + w \quad (3)$$

   where w is the transistor width and n is the technology node (e.g. 32 nm for the 32 nm node).
4. Spice is not case sensitive.
5. Make sure you attach the correct load transistor to your design!

6 Control File

To simulate a spice circuit you need the spice netlist generated above that describes the circuit, and a control file that defines the process node parameters, simulation waveforms, and result measurements. *Please name your control files with the same file body as the .sp netlist, but with a .control extension.*

An example circuit netlist and control file for an inverter are found on the class web page. You may first want to run this circuit through hspice and CosmosScope to get a feel for the tools.

The control file consists of the following sections:

1. An .include command to read in your circuit netlist (.s).
2. An .include command to read in the technology node parameters.
3. A command .option acct=1 post that tells hspice to write a binary waveform file that can be viewed in CosmosScope.
4. Your supply voltage. (Use 1.8V for the 130 nm node and 1V for the 32 nm node for your supply voltage.)
5. A description of your waveforms (use piecewise linear for this lab).
6. A transient simulation command that defines the type and duration of your run.
7. Measurement commands for evaluating delays and slopes of your signals.
8. End the file with a .end command.

6.1 Test Vector Set

Testing the correctness of a design is a fundamental aspect of the design flow. However, *for this lab only*, we will employ a weak test strategy. For this lab, you will need to define test vector sets for each design (the 130 nm and 32 nm designs). One vector set will be designed that results in the fastest input-to-output delay of your function, and a second that produces the slowest input-to-output delay. This will be up to you to determine these vectors. Points will be deducted if your vectors are poorly chosen and do not come close to the best and worst delays.

The best and worst vectors will depend on the number of transistors in series, the size of the transistors, and the number of stages that your signal must pass through from input to output. Note that *typically* the transistor closest to the output will result in the fastest transition, and the one farthest will result in the slowest transition for a gate. Based on the transistor structure of your design, evaluate the top two or three candidates for each case, and run them through your circuit.

For the lab results you will report four delays: the best and worst falling and rising times for each process node. The input signal rise and fall times must be implemented as specified in the following paragraph. The input-to-output delay for each simulation vector will be measured from the 50% to 50% point on the transition from the latest switching input to

the output. If you have multiple test vectors in a single simulation, each vector must be separated in time sufficiently to allow the output signal to rise or fall to at least 95% of the voltage range.

The rise and fall times for all your signals must be **saturated ramps**. These are voltage changes on an input signal that switch from ground to power (or vice versa) with linear voltage changes over time. Rise and fall saturated ramps will be identical at 40 ps for the 130 nm node, and 20 ps for the 32 nm node. These will be specified as **piecewise linear** waveforms in your .control file, similar to the inverter example from the web page.

You may create a single simulation or have up to four independent simulations to test your vectors. (Note that from an energy perspective, changing the fewest number of signals will give the lowest energy expense.) If you can put multiple vectors into a single simulation it will be easier to report the results, but feel free to run several simulations.

The winner of the contest will be the student with the smallest sum of the slowest rise and fall delays for both circuits.

6.2 Waveform Generation

There are two tedious aspects of this lab. This is partially due to the low level simulations we are performing here. Keep positive! Most of the rest of the labs will use higher levels of abstraction. *Please see the instructor if you are struggling with any aspects of this assignment.*

The first tedious aspect is developing the spice transistor netlist and the specification of the `Mos` transistor models. Make sure you double and triple check these to make sure all the transistor parameters are correct, including connectivity. The second tedious aspect is generating correct waveforms for your circuit. I developed a software package that would automatically generate these waveforms when at Intel, but unfortunately we will need to do this by hand because Intel would shoot me if I took the software I developed with me... To simplify this process, make sure that you have some simple tests. Initially just run a single vector through the design and make it simple. Keep all signals stable but one that will switch the output. View everything in CosmosScope to make sure your results make sense. I also suggest that you test your best and worst vectors individually until you have selected the four best vector. I would only generate the composition of the four waveforms at that point.

7 Calculating Results

The quality of your design is measured as the $e\tau$ value of your circuit. This is calculated as follows. Use Spice to calculate the average power of your circuit across the transient simulation of your waveform. This is performed with the spice command .meas tran NAME avg power from=XXXps to YYYps. To calculate the energy expended in the simulation, you multiply the average power by the runtime: YYYps − XXXps. This gives you e for this simulation. You then calculate the rise and fall times of the signals.

Four test vectors need to be run at each process node. The time τ we use for our quality metric is the sum of the input to output delays for the fastest and slowest rising and falling

transitions.

Thus our final $e\tau$ result is the sum of the energy of all of our simulation runs multiplied by the sum of the four delays.

8 CAD Tool Setup

We run our suite of VLSI CAD tools from the CADE lab. To run the CAD tools, you will need to update your shell scripts, usually .cshrc. We normally try to run the latest version of the tools, that are enumerated by 'S' (spring) or 'F' (fall) and the year. Place the following into your .cshrc to allow access to the tools (but you may want to add additional years):

```
set uu_cad_path=( \
        /uusoc/facility/cad_common/local/bin/S17        \
        /uusoc/facility/cad_common/local/bin/F15        \
        /uusoc/facility/cad_common/local/bin/F13        \
        /uusoc/facility/cad_common/local/bin/S13        \
        /uusoc/facility/cad_common/local/bin/F12        \
        /uusoc/facility/cad_common/local/bin/S12        \
        /uusoc/facility/cad_common/local/bin/F11        \
        /uusoc/facility/cad_common/local/bin/S11        \
        /uusoc/facility/cad_common/local/bin/F10        \
        /uusoc/facility/cad_common/local/setups )

set path=($path $uu_cad_path)
```

You should now be able to directly run hspice and CosmosScope. Hspice is run with the command:

```
hspice -i inverter.control -o inverter.out
```

If the hspice run succeeds, it will echo "hspice job concluded" to the shell. If it tells you "hspice job aborted", look in the `.lis` file and find the error messages. Your measure commands will be written to a `.mt0` file.

CosmosScope is run with the command:

```
syn-cscope
```

To observe your waveforms in cscope, you need to open a waveform "plotfile" by selecting the inverter.out.tr0 file. Do this by clicking on the folder icon. This opens a popup that contains a list of circuit nodes you can graph. Double click on the waveforms that you want to observe in the graph display. I suggest expanding the waveforms to fill the whole cscope display. **Combine waveforms into a single graph** by dragging the waveform names in the cscope graph display on top of each other. *When you report results, you need to have the input and output waveforms in a single graph.*

If you are debugging your command file, do not close cscope. You can view the results of multiple hspice runs. There is a button in cscope that will refresh the display so you don't need to redefine the graphs and waveforms you wish to observe.

9 Deliverables

All of the following files should be on one directory and then submitted as a single tar file via **canvas**. All documents should be in .pdf or text formats.

1. **README.txt** : A text file containing a brief description of your design, the directory and file structure for your submission, and documentation of any problems you encountered.

2. Include the .control files, .sp files, and .mt files for your simulations.

3. Draw a schematic of your circuit. (You can do it on engineering pad and scan it for submission, or draw it in the Cadence tools, or other drawing package.)

4. Describe the simulation vectors that you generated, and why you think they are the best and worst case vectors.

5. Include images of *all* of your simulation runs that you used to calculate $e\tau$. Only plot the last input that changes, which causes the output to change. Plot this input and output in the same graph. If the signal names and behaviors are not obvious, then label the plots. Send in the plot as an image file, such as .png. (You can get an image of your plot with the `xv` tool.)

6. Report the overall $e\tau$ value for the four transitions of your design for the 32 nm and 130 nm designs. Create a table reporting the power, rise time and fall times for each simulation vector. Define how you calculated this value.