Lab Assignment #2

University of Pittsburgh
Department of Electrical and Computer Engineering
September 8, 2006 (Friday, Week 2)

1 Circuit Simulation

Circuit simulation is one of the fundamental steps in the design methodology of VLSI circuits. The operation characteristics of the designed product are predicted in a cost and time efficient manner using simulation. In the current assignment, circuit simulation is used to illustrate basic concepts of digital circuits and to check for hand-written SPICE models.

1.1 Setting up your account

In this lab, you will be using software provided by a company called Synopsys. HSPICE will be used to simulate the circuit designs, and CosmosScope will be used to view the output from the HSPICE simulations.

Before you get started on this assignment, you need to ensure that your account is properly set up to run the software needed to complete this lab. To verify that HSPICE and CosmosScope are properly set up, issue the following commands:

```
>which hspice
/CAD/synopsys/V-2004.03-SP1/bin/hspice

>which cscope
/CAD/synopsys/cosmos-scope_vV-2004.06/ai_bin/cscope
```

If you have any problems and you are sure that you properly have your account setup, inform your TA immediately.

2 Introduction to SPICE

SPICE is the most commonly used analog circuit simulator today and is enormously important for the electronics industry. SPICE is a general purpose analog simulator which contains models for most circuit elements and can handle complex nonlinear circuits.

2.1 Circuit elements

In SPICE syntax, all elements within a circuit are labeled as components. The wires, for instance, are labeled as `nets`, and the connectivity of the circuit is defined via these nets. The standard SPICE syntax requires the definition of an element (labeled by a letter such as `R` for a resistor, `C` for a capacitor etc.), the nets its terminals are connected to and the value for the element. The SPICE syntax for component definitions looks as follows:

```
C0 gnd net1 10f
```

```
R0 net2 net3 1k
V3 net4 net5 DC 3.3V
V0 net6 net7 pulse 0 vddp 0n 200p 200p 3n 6n
```

The first line is a capacitor, named C0, between gnd and net1. Its value is specified to be 10fF. The second line is a 1K resistor, called R0, connected between net2 and net3. The third line is a 3.3V DC voltage source named V3, between net4 and net5. The fourth line is another voltage source, connected between net6 and net7. The voltage source produces pulses of 0 V and vddp (e.g. 5V). The time values listed are respectively `delay`, `rise_time`, `fall_time`, `pulse_width` and `period`.

The format for a MOSFET is:

1. Name (must begin with M)
2. Node numbers in the following order : Drain Gate Source Base
3. MOSFET model name
4. W and L are the width and length of the gate (in meters) [μ is micron]

For example,

```
M_0 output input vdd vdd p W=2u L=1u
M_1 output input gnd gnd n W=1u L=1u
```

Note that the first letter of each line in the above examples specifies the element type. Most popular elements, used in this course are:

```
V Voltage
C Capacitance
R resistance
M MOSFET
Q BJT
```

2.2 Voltage input waveforms

There are some commands to insert input waveforms in the circuit, as well. 'PWL' and 'PULSE' are two input waveforms frequently used in transient analysis. The format for a piece-wise linear (PWL) waveform is:

```
PWL time0 value0 time1 value1 ...
```

where value0 is the value at time0, value1 is the value at time1 and so on.

An example could be:

```
Vin 3 0 PWL 0n 0V 0.4n 5V 14.6n 5V 15n 0V
```

The format for the PULSE command is:

```
PULSE (initial_value pulse_value delay rise_time fall_time pulse_width
period)
```

An example for this input waveform command is:

```
Vin 3 0 PULSE (0 5 4.9n 0.2n 0.2n 4.8n 10n)
```

2.3 Common commands

The syntax to define circuit components are listed in the previous sections. Here are some basic HSPICE commands:

- `.INCLUDE` is used to include text from one file at run time.
- `.PARAM` is used to define a parameter that is used as a variable in other statements in the netlist file.
- `.DC` performs DC analysis. For example `.DC Vin 0 5 0.1` performs a DC analysis using a sweep of Vin from 0V to 5V in 0.1V increments.
- `.TRAN` performs a transient analysis. For example, `.TRAN 0.1n 50n`.
- `.IC` is useful for setting initial conditions for transient solutions. This is essential whenever the circuit stores information, such as latches, flip-flops or on dynamic storage nodes (capacitances etc.). For example, `.IC V(1)=5` initializes node 1 to 5V
- `.END` is required at the end of an HSPICE netlist file.

2.4 Subcircuits

A subcircuit allows you to define a collection of elements as a subcircuit (e.g. an RC circuit) and to insert this description into the overall circuit. A subcircuit defintition is begun by a .SUBCKT, followed by the circuit description as follows:

```
.SUBCKT SUBNAME N1 N2 N3 ……
Element statements
.
.
.
.ENDS SUBNAME
```

where SUBNAME is the subcircuit name and N1, N2, N3 are the external nodes of the subcircuit. The external nodes cannot be 0. The node numbers used inside the subcircuit are strictly local, with the exception of 0 (ground) which is always global. `.ENDS` must be used at the end of the subcircuit definition. The SUBNAME (subcircuit name) used with the `.ENDS` statement indicates which subcircuit definition is being terminated. If SUBNAME is omitted, all subcircuits being defined are terminated. The name is needed only when nested subcircuit definitions are being made.

The following example shows the use of a subcircuit:

```
.SUBCKT RC_Circuit net1 net2
R1 net1 net2 1k
C1 net2 gnd 10f
.ENDS RC_Circuit


X1 vdd netx RC_Circuit
```

When using a subcircuit, its name must start with ‘X’. In the above example, the subcircuit is called X1.

2.5 Simulation

In order to run simulations on the netlist file, specify the durations of the simulation and/or the initial and incrementing values for the input voltages. Basically two types of simulations will be used:

1. DC Sweep: The `.DC` statement allows you to increment (sweep) an independent source over a certain range with a specified step value. The format of using this statement is as follows:

   ```
   .DC source_name Vstart Vstop Vincrement
   ```

   The DC sweep command can be nested and is often used to plot transistor characteristics. The format for the nested DC Sweep is:

   ```
   .DC source_name1 Vstart1 Vstop1 Vincrement1 source_name2 Vstart2
   Vstop2 Vincrement2
   ```

2. Transient Analysis: The .TRAN statement specifies the time interval over which the transient analysis takes place and the time increment. The format of this command is:

   ```
   .TRAN tstep tstop <tstart <tmax>>
   ```

   where tstep is the time increment, tstop is the final time, tstart is the starting time (if omitted, tstart is assumed to be zero), tmax is the maximum step size. tstart and tmax are optional.
   For example :

   ```
   .TRAN 0.01N 20N
   ```

2.6 SPICE code

The SPICE file should contain the technology parameters, in order to produce a good prototype of the circuit and produce realistic simulation results. This is done by including a technology file stored in a path, or by directly trying to write the parameters. Most of the manufacturers release their process technology parameters for use in simulation and circuit realization.

Now look at the following source:

```
* FILE: invert.sp
.include '/classes/ece1192/fall06/CLASS/ami_C5N.mod'

VDD vdd 0 DC 5

* My code starts here
.tran 0.01n 100n
M0 out in gnd gnd n W=2u L=1u
M1 out in vdd vdd p W=4u L=1u
C1 gnd out 15f
R1 gnd out 1K
VA in gnd PWL 0n 0V 0.2n 5V 14.6n 5V 15n 0V
.END
```

The lines starting with '*' are just comments and not a part of the code.

The line starting with `.include` command includes the path for the technology file. This technology, which is also being used in layout drawing, is called `AMI_C5N' technology. '5' indicates that twice the minimum length for a gate of a transistor is 0.5um ($2*l_{min} = 0.5um$).

To write a full-custom SPICE code, create a regular text file with '.sp' extension, and copy the lines specifying the technology and settings to a new text file (from the beginning to the comment line: 'My code starts here'). Then, in this new text file, append your SPICE code, where you implement your circuit and include simulation commands.

Note: The presented file includes unnecessary comment lines. You do not need to include this in your `.sp` files. However, **The first line of every SPICE file MUST be a comment**. You might want to create one such file as a template file and make copies of it every time you need to write different SPICE files.

2.7 Additional information

To generate the netlist file, Cadence or MMI tools (VLSI design packages) can be used. The circuit is designed in these schematics/layout editing tools by using the built-in commands of the tools to create a netlist. The simulations are run and resulting waveforms are observed for circuit functionality. For this assignment, the SPICE files will be written from scratch (using any text editor), as one of the objectives of this assignment is to practice writing full-custom SPICE codes to represent circuits. **NOTE:** SPICE code requires all nodes of a circuit to be connected (to a net). No floating nodes are allowed.

2.8 Running HSPICE

To run HSpice on any SPICE input file, type "`hspice <filename>.sp`" at UNIX command prompt. Remember SPICE files must be named with ".sp" extension. Follow the output on your screen. Watch out for any warnings or errors that occured during simulation.

Several files are created by HSPICE:

<filename>.ic0: text file containing circuit initial conditions

<filename>.st0: text file containing a summary of the simulation

<filename>.tr0: Binary file containing transient analysis waveforms

<filename>.sw0: Binary file containing dc sweep analysis waveforms

To learn more about HSPICE refer to the online manuals and the related documentation on the world wide web.

2.9 Quick introduction to CosmosScope

After simulating the SPICE code, the output waveforms can be viewed using CosmosScope. To use CosmosScope, type "cscope &" at the UNIX command prompt.

- Click on `File > Open > Plotfiles` and choose your ".tr0" file that contains the transient analysis.
- A signal manager window and the plot file window will open up.
- Choose which variables you want to plot (voltage, current). For example, you want to plot v(in), double click v(in) in the plot window.
- The waveform editor has rulers and zoom menus to ease measurement.
- You can also view waveforms for the DC sweep analysis. For this, click on `File > Open > Plotfiles` and choose your ".sw0" file that contains the DC sweep analysis.
- CosmosScope has a help manual which can be invoked from within the program.

3 Assignment

1. To familiarize you with HSPICE and CosmosScope, you will enter the following circuit description in a .sp file, simulate it, and then view the waveforms.

```
.OPTIONS LIST NODE POST            <- required for analysis

.TRAN 1u 20m                       <- transient analysis from
                                      0-20ms,with a time step
                                      of 1 micron

.PRINT TRAN V(1) V(2) I(R1) I(C1)  <- print voltage on node 1
                                      and 2, and current
                                      through R1 and C1

V1 1 0 PULSE 0v 5v 0m 0m 0m 5m 10m <- pulse voltage source, 0-5
                                      volts no delay, rise_time,
                                      or fall_time period 10ms
                                      with 50\% duty cycle

R1 1 2 1k                          <- 1k resistor between nets
                                      1 and 2
C1 2 0 1u                          <- 1uF capacitor between
                                      nets 2 and 0

.END                               <- also required by SPICE
```

   - After simulating, **plot** the voltage at the capacitor (V(2)) and the current through the resistor (I(R1)) using CosmosScope.
   - Now you need to specify an initial condition for the voltage on the capacitor. To do this, enter the following line in the spice code before `.END`.

     ```
     .IC V(2)=2
     ```

   - Again, after simulating, **plot** the voltage on the capacitor.

2. Design a CMOS inverter, where both PMOS and NMOS transistors have a length of .5 microns. Set the width for the PMOS transistor to 2 microns and set the width of the NMOS transistor to 1 micron. Load the inverter with a 30fF capacitor. Assume Vdd to be 5 Volts. *NOTE:* the following lines must be added before `.END` for the transistors to function.

```
.MODEL p PMOS LEVEL=1
.MODEL n NMOS LEVEL=1
```

   - Assume an ideal pulse at the input with a magnitude of 5 volts. Delay, rise time, and fall time are negligable. Use a period of 20ns with a duty cycle of 50%.
   - **Plot** the voltage at the output of the inverter.
   - Modify the input signal to have 2ns rise and fall times.
   - **Plot** the voltage at the output.

3. Using the same inverter in the previous task, perform a DC sweep on the input voltage. You must remove the lines starting with `.MODEL` for this part and include the following line.

   ```
   .include '/classes/ece1192/fall06/CLASS/ami_C5N.mod'.
   ```

   This will set the proper technology parameters for your transistors.

   - Perform the DC sweep on the input from 0 to 5 volts using a 0.05 increment and **plot** the Voltage Transfer Curve (input and output voltages) for the inverter.

4. Now create a subcircuit of an inverter and use it to build a 7-stage ring oscillator. **Measure the delay** of the ring oscillator and then **compute** the delay of a single-inverter.

5. Print outs of **all source code (.sp files)** used for each part and **all waveforms generated (plots)**, **all measurements**, and **computations** are due by next Friday, September 15, 2006.

Hint for Problem 4: A ring oscillator is a device composed of an odd number of inverters whose output oscillates between two voltage levels, representing true and false. The inverters are attached in a chain and the output of the last inverter is fed back into the first inverter. The schematic for a 7-stage ring oscillator is shown below: