Tutorial 2: HSPICE Analysis

1 A MOS-Resistor-based Inverter

The inverter circuit in Figure 1 functions by pulling the node 'vout' low when 'vin' is high, and vice-versa. This tutorial uses this toy circuit to demonstrate DC and transient analysis in HSPICE.

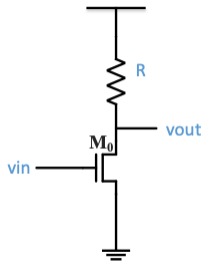

Figure 1: Basic inverter with NMOS transistor and pull-up resistor

2 Directory Hygiene

It will prove to be invaluable to keep your design files for each CAD organized. At the very least:

- In your ee476 directory, create a new directory for each tutorial/CAD
- Keep a single cadence directory to hold all of your libraries
- Within each CAD directory, create a link to the cadence work area, and a subdirectory for your spice analysis

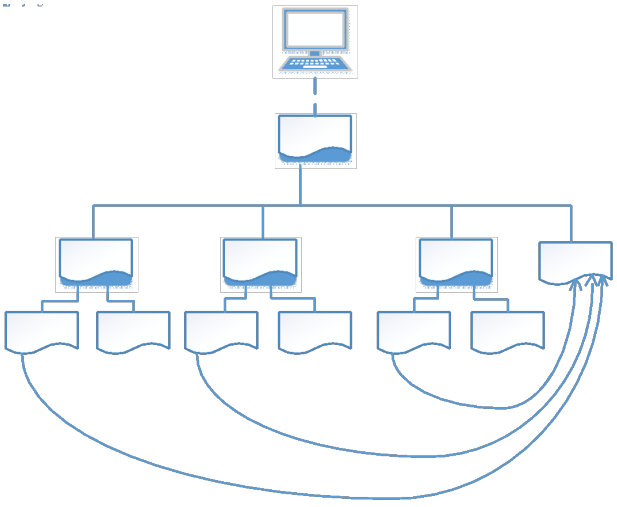

Figure 2: Directory structure for EE476 CAD assignments

3 Schematic and Netlist Creation

1. First, create a new directory called `tut_2` under your ee476 directory. (as in Figure 2.) Run virtuoso, and create a new design library, also called `tut_2`. In the new library, create a new schematic cell-view called 'nmos_inverter'.
2. Use hotkey "i" (or click the "Instance" button) to insert the `NMOS_VTL`, `res`, `vdd` and `gnd` components. Except for `NMOS_VTL`, which is in the library `NCSU_Devices_FreePDK45`, the rest of the components are in `analogLib`. If you do not know how to insert those components into your schematic view, see Tutorial 1 for details.

Use the hotkey `q` or right click the NMOS transistor and choose `Properties...` to change its characteristics. For this tutorial, we will use the 50 nm and 1 um as length (l) and width (w) of NMOS transistor, respectively.

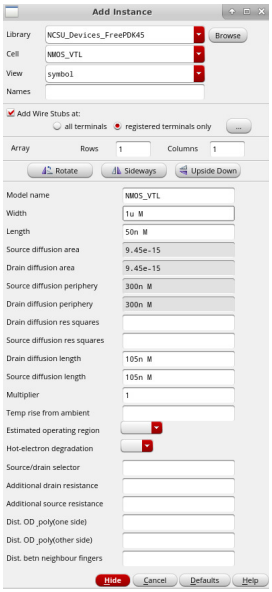

Figure 3: NMOS Transistor, NMOS_VTL

3. Connect the components as in Figure 4 below. (Do not forget to connect bulk of NMOS transistor with ground.)
4. Launch Analog Design Environment and generate the netlist file.

4 HSPICE DC Analysis

After generating your `input.ckt` file, you can make a new directory called `spice` in your Tutorial_2 directory. Create a soft link of your netlist file (input.ckt) to the spice directory under the name `nmos_inverter.ckt`. This ensures that the next time you modify your circuit and re-generate the netlist, you will not have to re-copy the new file to the spice directory.

To observe the DC transfer function of the NMOS-inverter we built, we can supply commands to HSPICE with a control file. Control files have the extension `.ctl`, compared to netlists which have the `.ckt` extension. A control file specifies the type of analysis, defines voltage sources for the analysis, and carries out any automatic measurements. Follow the steps below for a simple DC analysis, or skip below to see the completed code after.

- Create a new file called **nmos_inverter.ctl**
- Put a comment `*` on the first line spice first line. It should be the description of your anaylis.
- Include the Technology Library, replace `<TECH_PATH>` with the apporpiate value:

Warning: Linux Labs and Hyak has different folder organization

```
Linux Lab Servers: "/home/projects/ee476.2022aut/common/cadence_setup/freepdk45.l" tt_lib

Hyak: "/gscratch/ece/courses/476_aut2022/common/cadence_setup/freepdk45.l" tt_lib
```

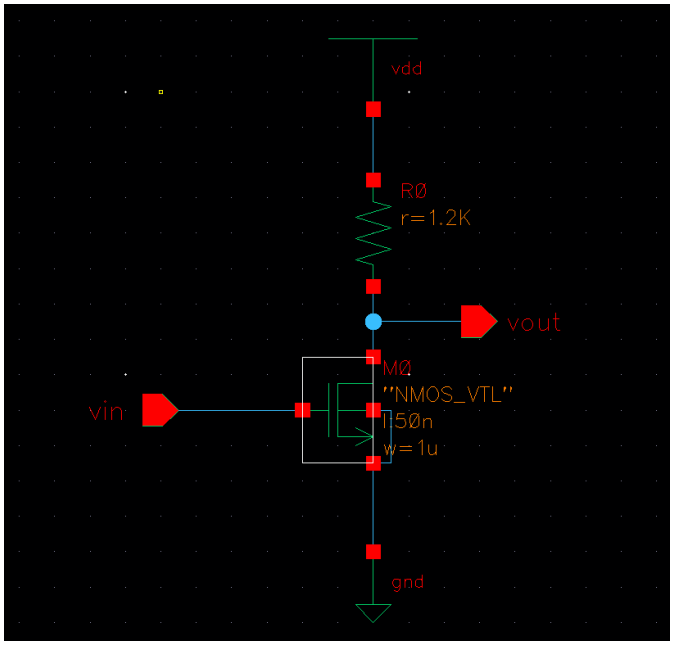

Figure 4: Virtuoso - NMOS Inverter Schematic

Add the following line to your `ctl` file:

```
.LIB <TECH_PATH>
```

Warning: replace `<TECH_PATH>` with the apporpiate value

- Include the circuit netlist with:

```
.INCLUDE "mos_inverter.ckt"
```

- Add independent voltage source into your control file as follows:

```
*V<name> <+ terminal> <- terminal> <voltage_level>
Vvgs vin  0 1.2
Vvdd vdd! 0 1.2
```

The two command lines above add two voltage sources between **vin** and **vdd!** to ground. You could also replace the constant `1.2` with a parameter using the `.PARAM` statement. For instance:

```
.PARAM vdd = 1.2
Vvgs vin  0 vdd
Vvdd vdd! 0 vdd
```

(Note: The name of node and the name of parameter can be same, and HSPICE allows this, but it is up to you to not be confused between them.)

- Add a DC analysis statement into your control file.

```
* .DC <source> <start_val> <end_val> <step_size>
.DC Vvgs 0 1.2 0.05
```

In the example above, the voltage-source **Vvgs** is being swept from 0V to 1.2V in increments of `0.05V`.

- Add `.OPTION POST` in your control file. This tells HSPICE to generate waveform files. For a DC analysis, the output file is a `.sw<0-9>` file.
- Finish the control file with `.END`. At this point, the control file might look like the following:

```
* DESCRIPTION ctl file
```

```
* ===== Tech Lib ======
   .LIB <TECH_PATH>

* ==== Netlist ====
.INCLUDE "nmos_inverter.ckt"

* ==== Voltage sources ====
.PARAM vdd = 1.2
Vvgs vin  0 vdd
Vvdd vdd! 0 vdd

* ==== DC analysis definition ====
.DC Vvgs 0 1.2 0.05

.OPTION POST

.END
```

Invoke HSPICE on your control file by typing `hspice nmos_inverter.ctl -o nmos_inverter.lis` at the command line. After running HSPICE, you should be able to view the `IDS vs VDS` curve by opening the `.sw0` file in Silicon Explorer.

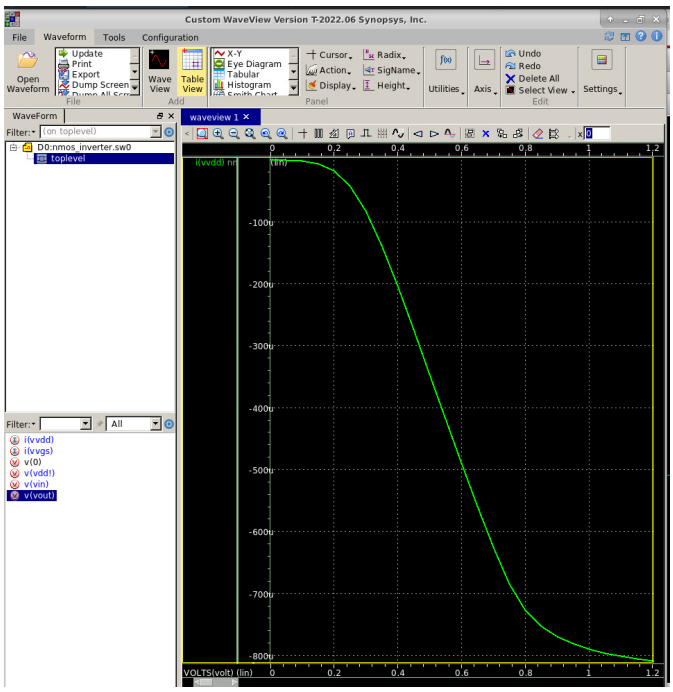

Figure 5: SX - IDS vs VDS

- We can add automatic measurements to the control file with `.MEAS` statements. The following creates a measurement to find the inverter switching-threshold voltage.

```
.MEAS DC switch_thresh FIND V(vin) WHEN V(vout) = V(vin)
```

In this case, the measurement statement `FIND V(vin) WHEN` the output voltage (vout) is equal with the input voltage. The result is stored under the name **switch_thresh**.

`V(<node_name>)` and `I(<instance name>)` are references to the voltage and current of nodes and elements, respectively.

- Re-run HSPICE and open the `.ms0` file to read the measurement result. If the measurement failed, see the HSPICE output in the `.lis` file.

You are now finished with the DC analysis portion of this tutorial. The next section covers transient analysis.

5 HSPICE Transient Analysis

Transient (or time-based) analysis is the most common type of simulation in the analysis of digital VLSI circuits. In this subsection, we will look at how to perform a simple transient simulation for our toy inverter. A helper spice control file is available for your reference in the common directory (`<476_FOLDER>/spice/tutorial_2/inx_mos_res_tran.ctl`).

- Create a new file called `nmos_inverter_tran.ctl`.
- Include the netlist, define static voltages, and recreate the HSPICE file from the DC analysis portion except for the lines with the `.DC` statement, and the `Vvgs` statement. (We will instead use a `.TRAN` statement instead of `.DC`, and define a periodic input for `Vvgs`)
- Define the transient analysis with the following:

```
*.TRAN <t_step> <t_duration>
.TRAN 1p 10n
```

The above example defines a transient analysis that will run for 10ns, with a resolution of 1ps.

- Create a periodic input waveform:

```
* V<name> <+ terminal> <- terminal> PULSE +
* <low_voltage> <high_voltage> <delay> +
* <rise_time> <fall_time> +
* <pulse_width> <period>

Vvgs vin 0 PULSE +
0    1.2 0 +
20p  20p   +
1.0n 2.0n
```

The above creates a square-like pulse with a low-value of 0V, a high value of 1.2V, rise-time and fall-time of 20ps each, and a period of 2ns, where the high-time is 1ns. Of course, the constants could (and should) be parameterized.

The `+` is the HSPICE line continuation symbol; it is not necessary, but it may make the code more readable.

At this point, your control file may look like the following:

```
* First line comment
* ===== Tech Lib ======
.LIB <TECH_PATH>

* ==== Netlist ====
.INCLUDE "nmos_inverter.ckt"

* ==== Voltage sources ====
.PARAM vdd = 1.2
Vvdd vdd! 0 vdd

* ==== TRANsient analysis definition ====
.TRAN 1p 10n

* ==== Input waveform ====
.PARAM rise_time   = 20p
.PARAM fall_time   = 20p
.PARAM pulse_width = 1n
.PARAM period      = 2n
.PARAM time_delay  = 0n
Vvgs vin 0 PULSE
   + 0 vdd time_delay
   + rise_time fall_time
   + pulse_width period
```

```
* (Un-parameterized code)
* Vvgs vin 0 PULSE
* + 0    1.2 0
* + 20p  20p
* + 1.0n 2.0n

.OPTION POST

.END
```

The following picture shows the Transient Simulation of the circuit:

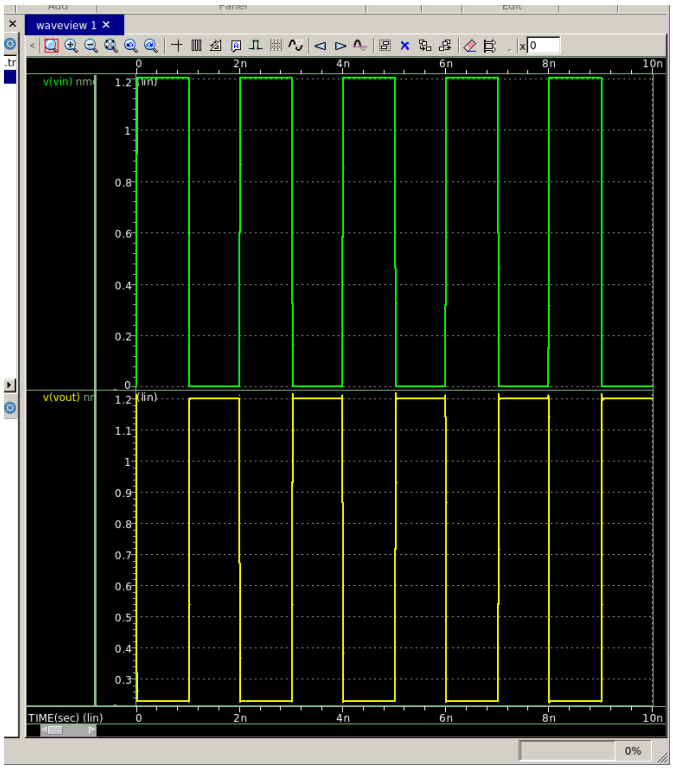

Figure 6: SX - VIN vs Time and VOUT vs Time

6 Additional Measurements and Tricks

There are additional examples of measurements, and ways to make your life easier, in the example control files for Tutorials 1 and 2.

The Tutorial 1 control file is named 'rc_series.ctl' and can be found in:

- `/476_FOLDER>/common/spice/tutorial_1.`

The Tutorial 2 control file is named 'inx_mos_res_tran.ctl', and can be found in:

- `/<476_FOLDER>/common/spice/tutorial_2.`

Please look carefully over these example files, since the tutorials are not exhaustive.