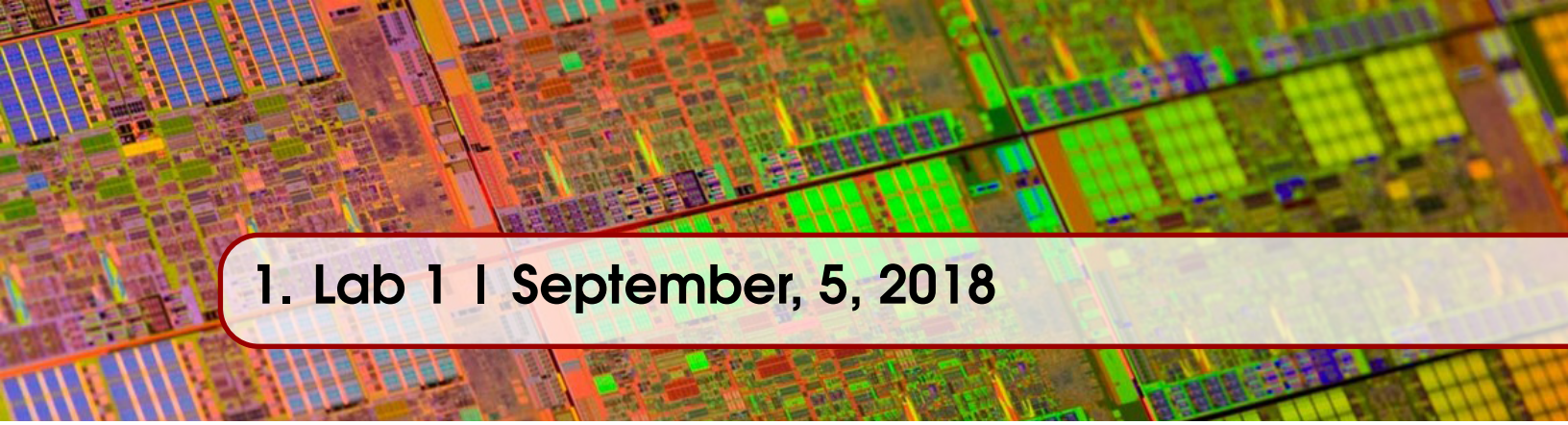

1. Lab 1 | September, 5, 2018

CMOS Inverter - Schematic and Circuit Simulation

Pre-lab Assignment: Due on September, 3, 12pm.
Lab Report: Due on September, 9, 11pm.

1.1 Objectives

The goal of this first lab is to teach you a basic full-custom VLSI set of skills by drawing the schematic and performing electrical simulations of a CMOS inverter using the TSMC 180*nm* design kit.

1.2 Pre-lab Assignment

Answer the following questions and submit a **.pdf** file through Canvas:

1. What is CMOS technology? Briefly explains how it works.
2. For a transistor, what I_{on} and I_{off} refer to? How are they usually measured?
3. What is the function of a CMOS inverter? Draw its schematic.
4. Briefly explain the *Voltage Transfer Characteristic* (VTC) and switching point of a CMOS inverter.
5. What does DRC, LVS and PEX stand for? What are they used for?
6. When considering both input and output curves from a logic gate, how are the propagation delay, rising and falling time usually measured?

1.3 Introduction the the Tools

In this lab, you will use a very well known industrial software: Virtuoso® from Cadence. Virtuoso is the platform for creating and simulating your designs. It consists of the Schematic Editor, the Layout Editor, the Analog Design Environment (ADE) which is the graphical front-end for the circuit simulator (called Cadence Spectre), and many other tools.

1.4 Lab Assignment

1.4.1 Running the Software and Creating a New Library

1. Launch Cadence Virtuoso® using the 180*nm* design kit. To do so, go to the working directory you created for the labs and into the virtuoso subdirectory and run the following commands:

```
cd virtuoso
virtuoso-tsmc
```

At this point, the software should start and the *Command Interpreter Window* should pop-up (Fig. 1.1). The CIW is the main window and allow you to launch all the Virtuoso® tools through the menus or through direct commands with the Cadence script language (SKILL). In this window, you can see all the information, errors and warning so always remember to take a look at it when something is not working.

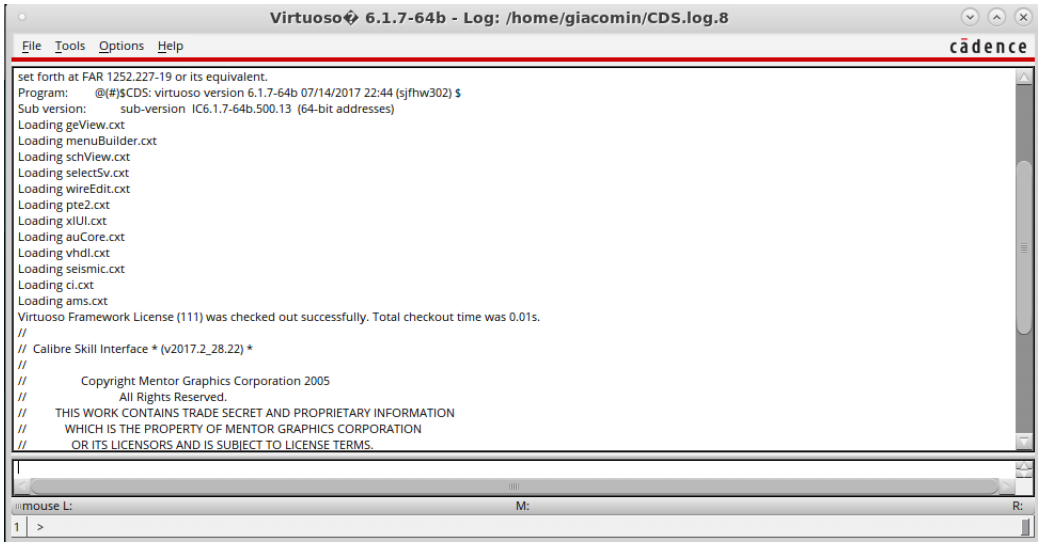

Figure 1.1: The Command Interpreter Window.

Another window should also pop-up (Fig. 1.2): the library manager window. In virtuoso, all the designs are stored in different libraries. Each library contains several cells and each cell is defined through different views (referred as cellviews). A single cell can have multiple cellview that can be different way of representing the cell (schematic, layout, symbol, *etc.*). By default, some libraries are already provided to you:

- **basic:** contains graphical elements for drawing schematics.
- **analogLib:** contains all the useful elements for electrical simulation (voltage and current sources, ideal resistors and capacitors, switches, ground, *etc.*). This library cells not being physical, this library is only used for simulation purposes.
- **tsmc18:** contains all the core devices (transistors, resistors, capacitors) from the TSMC foundry for the 180*nm* technology node. You will use those devices to create your own designs.
- **sealring:** contains the sealring layout views, which isolates the integrated circuit from the external environment. This library will only be used in case you will manufacture your chips.

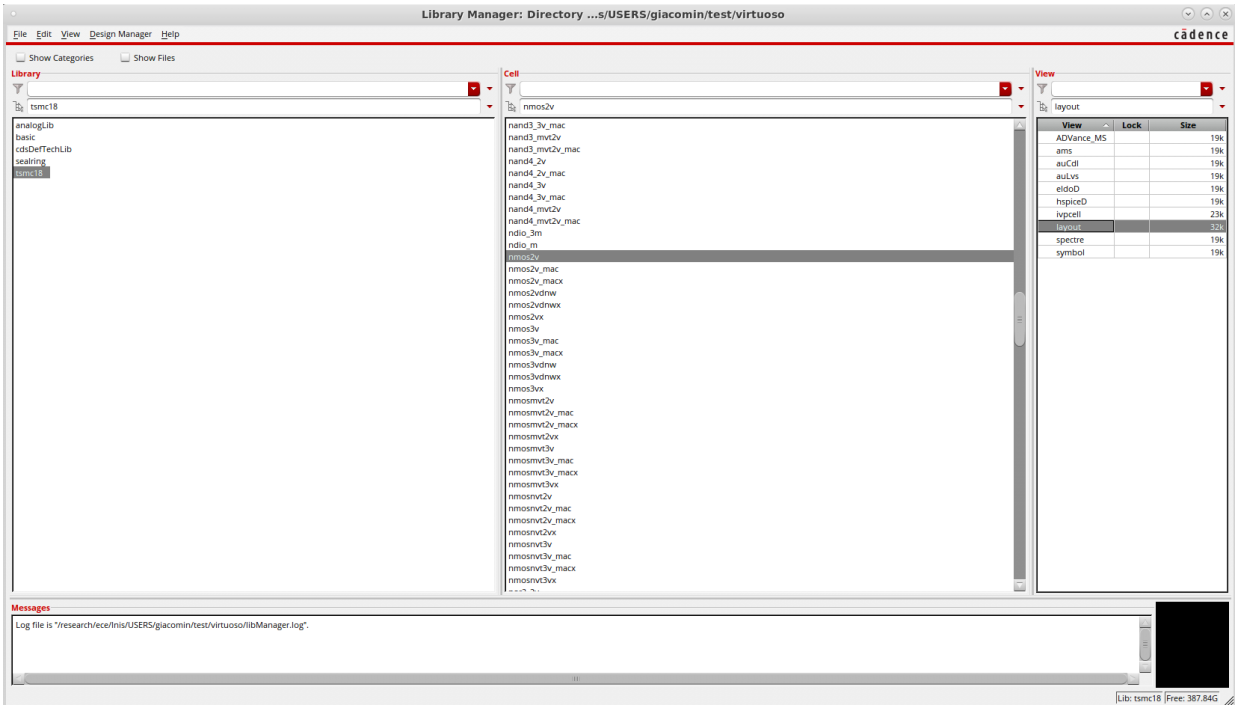

Figure 1.2: The Library Manager window.

2. Now, you first need to create a new library for your designs (referred as your design library). Create a new library in Virtuoso® *(Library Manager -> File -> New -> Library...)*. After choosing the location of your library (it is recommended to put it in the **/libs** directory inside your current working directory to ensure a better organization), a pop-up window appears, as shown in Fig 1.3.

Figure 1.3: Technology file choice for new library.

3. Select *Attach to an existing technology library* and select *tsmc*18. By doing this, your design library will use the same technology process (physical layers, layout rules, *etc.*) than the provided TSMC 180*nm* library by the foundry.

4. For the rest of this lab and the following labs, when creating new cells, place them in the design library you just created.

1.4.2 Technology Characterization

1.4.2.1 Creating the Schematic

In this section, you will first characterize the technology you are using. It is always a good practice to first draw an I-V curve of the transistors you work with to study their characteristics such as I_{on}, I_{off}, *etc*.

In this part, you will create a single schematic cellview for your *nmos* transistor and the voltage sources you will use to perform the I-V characterization. However, in the rest of the lab, it is suggested to always create a schematic cellview for your logic gate (i.e. inverter, OR) and another schematic cellview for your testbench (where you instantiate the voltage sources as well as the gate previously defined). In that way, it allows some flexibility since your gate cellview can be reused in multiple testbenches.

1. Create a new cell for the *nmos* characterization (*File -> New -> Cellview...*). Verify that your design library is selected in the *Library* field. Specify the name of your cell in the *name* field (such as *nmos_carac*). For now, you will perform an electrical simulation through a schematic so choose the schematic type, as depicted in Fig. 1.4, and click OK.

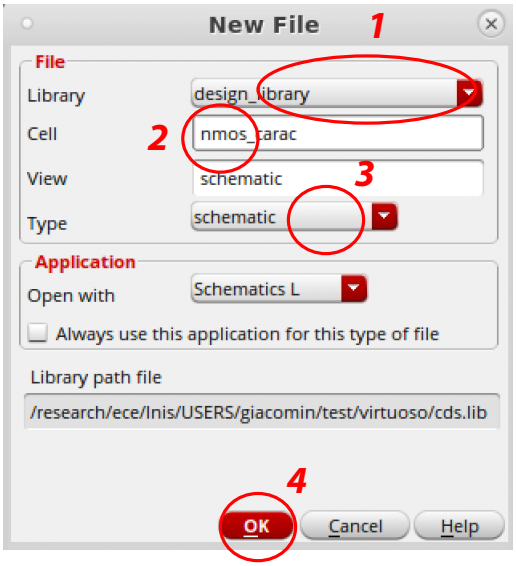

Figure 1.4: Creating a new cell.

A cell named nmos_carac is now created, with one view called schematic and the Virtuoso® Schematic Editor L window should pop-up, allowing you to edit your new cellview (Fig. 1.5). The **navigator** part allows you to quickly browse through all the devices, pins, nets, *etc.* of your design. The **property editor** allows you to quickly change the property of the currently selected device. The **schematic display area** is where you will instantiate all the components, voltage sources, *etc.*

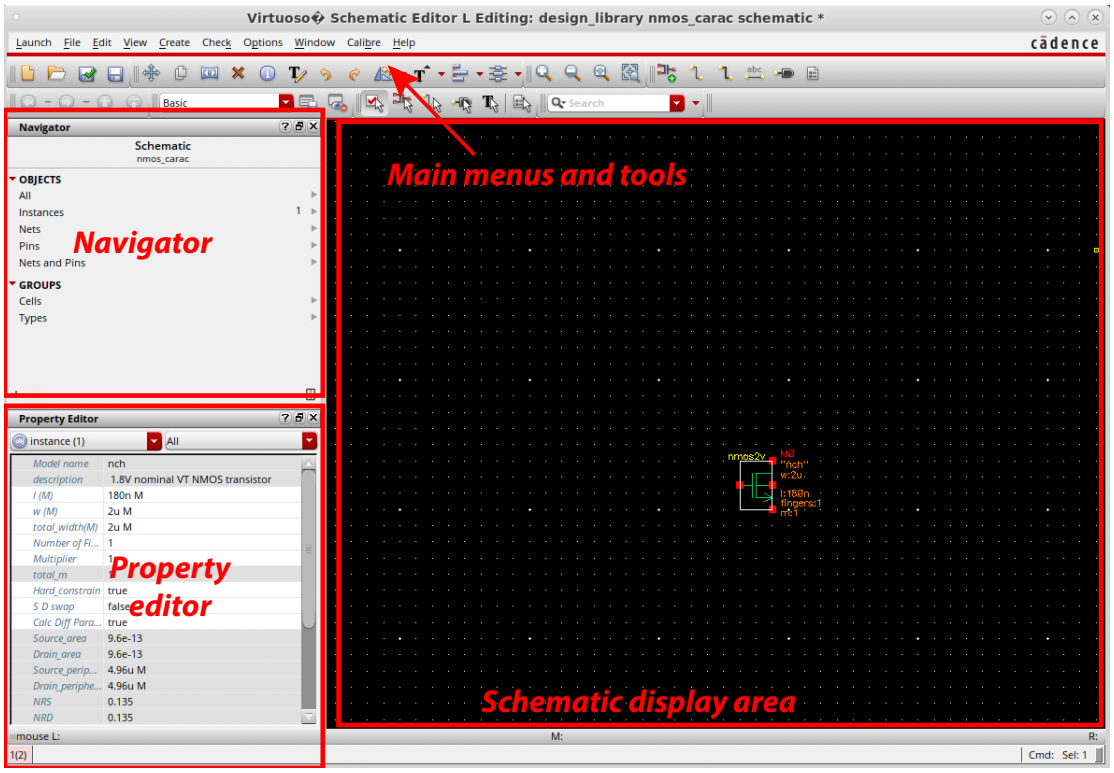

Figure 1.5: Schematic editor L window.

2. First, instantiate (*Create -> Instance...* or press **i**) an *nmos* transistor (*nmos2v* from the *tsmc18* library), as shown in 1.6. *nmos2v* and *pmos2v* are the regular 1.8V *nmos* and *pmos* transistors you will use for all your designs in the remaining of the labs.
3. Try to move your transistor on the display area to get more familiar with the tool. To do so, press **m**, click one the transistor and move it across the display area. Click again to place it where you want.
4. Check the transistor width. To do so, click on the transistor and press **q** (or click on the transistor and modify its width directly through the property editor as explained earlier). From the window, you can modify it width, length, number of fingers, *etc.*. Check that its width is 220*nm*, which is the smallest width for the TSMC 180*nm* technology. Here, we say that this transistor is "minimum sized". For the rest of the labs, you will generally use minimum sized transistors (but also take into account the fact that *pmos* transistors have to be larger than *nmos* and you also need to size your transistors accordingly when transistors are in series).

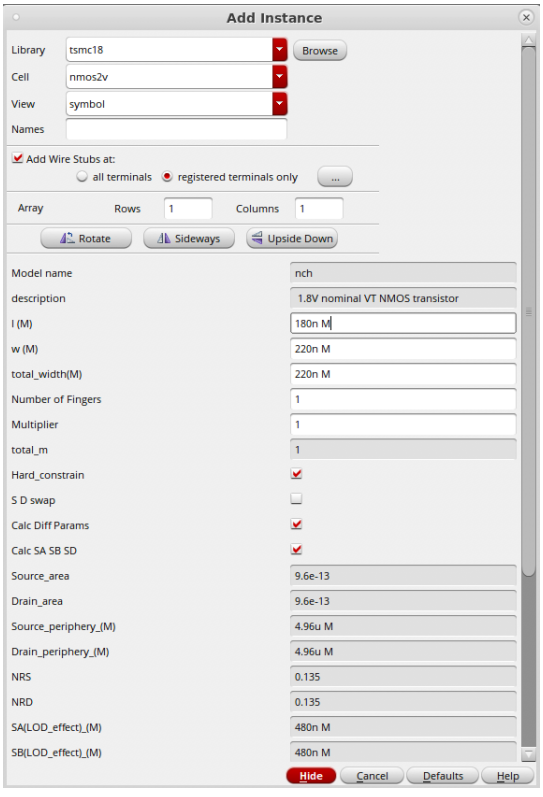

Figure 1.6: Component instantiation window.

5. Instantiate two voltage DC sources (*vdc* from the *analogLib* library) and place them in your design. One will be used to provide the gate voltage and the other one will be used for the drain voltage so place them near those terminals.

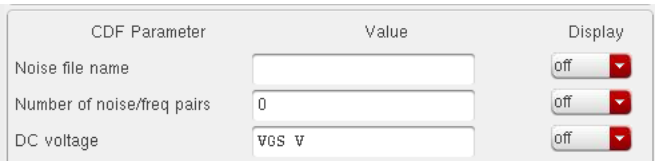

Figure 1.7: DC source parameter setup.

6. Connect the voltage sources correctly to the drain and gate of your transistor. To do so, you need to create some wires to connect the different parts of your circuit ((*Create -> Wire (Narrow)* or press **w**) and join the two points you want to connect.

> After pressing **w** to create a wire, you can press **s** (snap) so the wire will automatically connect to the closest pin from your mouse.

7. Connect the bottom pin of your voltage sources to the ground. You need to use the *gnd* instance from the *analogLib* library.
8. In the same manner, connect the source of your *nmos* to the ground. Do not forget to connect the bulk to the ground as well since you are using an *nmos*.
9. Edit each of your voltage sources (click on it and press **q**) to specify the DC voltage as a variable as shown in Fig. 1.7. In that way, the voltage of your sources are defined as global variables and you will be able to directly modify them later in your simulation.
10. Create some labels for the important wires (in this case, the drain D and gate G of your transistor). All the connected wires are electrically at the same potential and together define a **net**. It will also help you to know which curve is what when doing the simulation (otherwise the names are assigned randomly). To do so, *Create -> Wire Name...* or press **l**. Then specify the name(s) of the wire(s) you want to label and click Ok, as depicted in Fig. 1.8. Then, click on the respective wire(s) you want to label from your schematic.

Figure 1.8: Label creation.

11. Check and Save your schematic (*File -> Check and Save* or **Shift + X**). The schematic will be checked for potential errors. If there are any errors or warnings, a dialog box will inform you about them and the CIW will display a detailed message for each problem found.
12. Your schematic should look like Fig. 1.9.

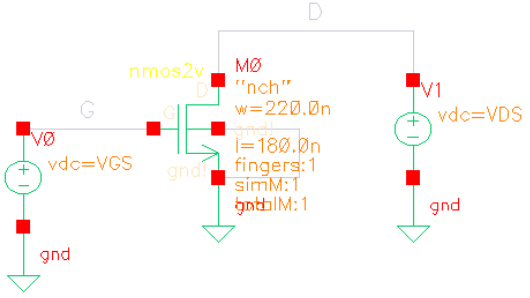

Figure 1.9: *nmos* transistor testbench schematic.

1.4.2.2 Performing the Simulation

1. Launch the simulation environment (*Launch -> ADE L*). The ADE L window should appear (Fig. 1.10). The **global variables** is where all your design variables are defined (transistor width or length, source voltages, *etc.*). Note that only the variables you define as a parameter in your schematic can be used as global variable. The **analyses** panel defines the different analyses you will run on your schematic (transient, DC, AC, *etc.*). The **display outputs** is where you choose which output (voltage, current, parameter) will be displayed after the simulation.

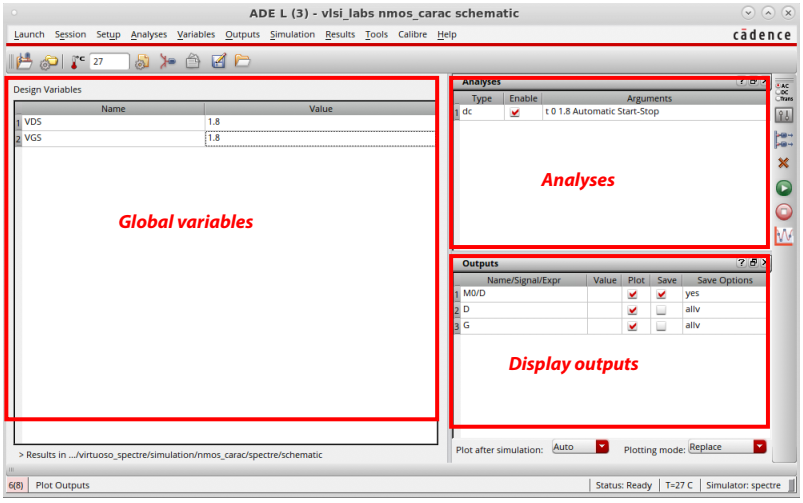

Figure 1.10: ADE L window.

2. First, you need to import your design variables (in this case the gate and drain source voltage you previously defined as parameters) to be able to define their value. To do so, go on the ADE window and do: *Variables -> Copy from Cellview*. Don't forget to specify an initial value for your parameter in the left field, otherwise, the simulation will not run. In our case, we can set both voltages to the nominal supply voltage of this technology node: 1.8V.
3. Specify which kind of simulation you want to run. In this case, we are doing a DC simulation (*Analyses -> Choose... -> dc*) and we want to draw an I-V curve so we need to sweep the drain voltage from 0 to $1.8V$. To do so, you need to select the *designvariable* option, choose the variable name as VDS and sweep if from 0 to $1.8V$, as illustrated in Fig. 1.11.
4. Select which outputs you want to plot (*Outputs -> To Be Plotted -> Select On Design*). The schematic window should appear. Select the gate and drain voltage as well as the drain current. Your ADE window should look like the one on Fig. 1.10.

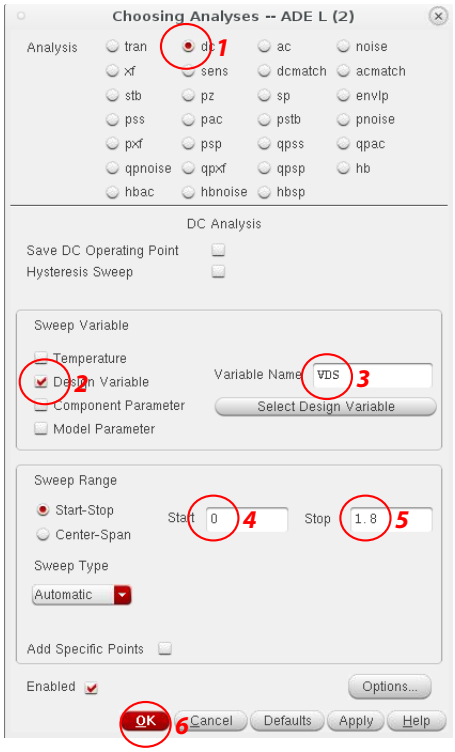

Figure 1.11: DC sweep analysis.

R When selecting which output to display for the ADE, you can select a voltage by clicking on the associated wire. To select a current, click on the associated terminal (red square on the schematic).

5. Run your simulation by clicking on the green triangle button and observe the I-V curve (Fig. 1.12 (a)).

(a) ***(b)***

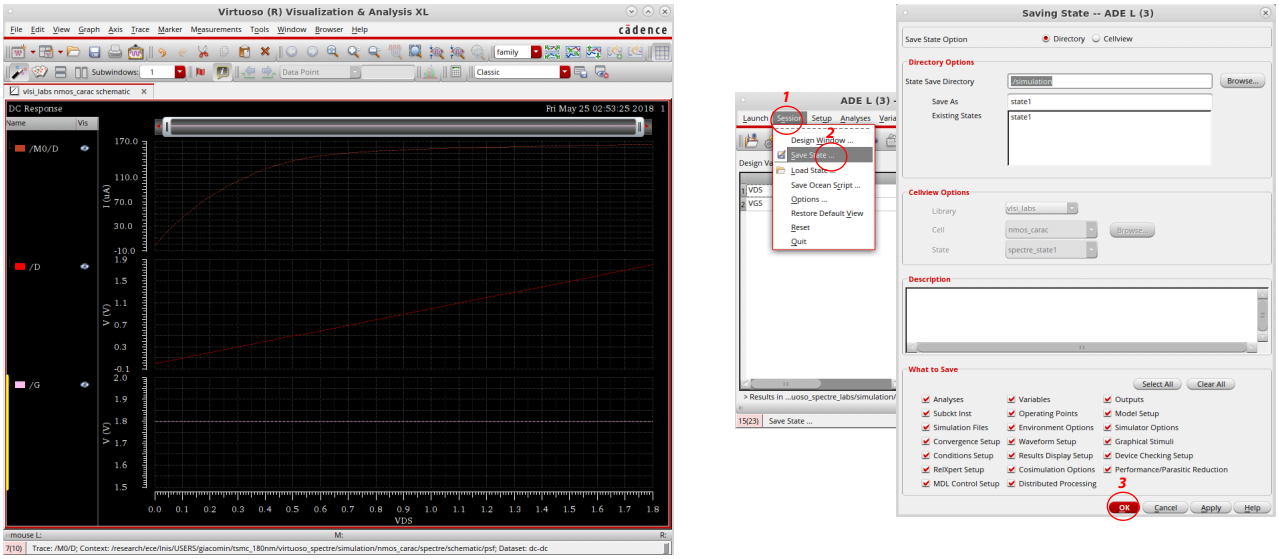

Figure 1.12: (a) *nmos* I-V curve; (b) Saving the ADE state.

6. Save your ADE L configuration so you can reopen it later without having to re-specify the displayed output signals, the analysis setup, the parameters, *etc*. To do so: *Session* -> *Save State...* -> *OK*, as shown in Fig. 1.12 (b).

> For the rest of the lab, don't forget to save your ADE configuration for each testbench you do.

7. Now, plot the I-V curve for different VGS. To do so, you need to run a parametric DC analysis. A parametric analysis allows you to perform several simulations (transient, DC, *etc*.) by sweeping a parameter. In case of a DC simulation, it allows you to sweep another parameter (you will sweep VDS for different VGS). To run the parametric analysis, do as follows:

   (a) From the ADE L window, go to: *Tools* -> *Parametric Analysis*. You can then choose which parameter to sweep as shown in Fig 1.13. You can also choose the range and the number of steps. Here, you want to sweep VGD from 0 to 1.8V with 10 steps.

Variable	Value	Sweep?	Range Type	From	To	Step Mode	Total Steps
VGS		☑	From/To	0	1.8	Auto	10

Figure 1.13: Parametric analysis setup for the DC sweep.

   (b) Run your simulation by clicking on the green triangle button from the parametric analysis window and observe the different I-V curves, as depicted in Fig. 1.14.

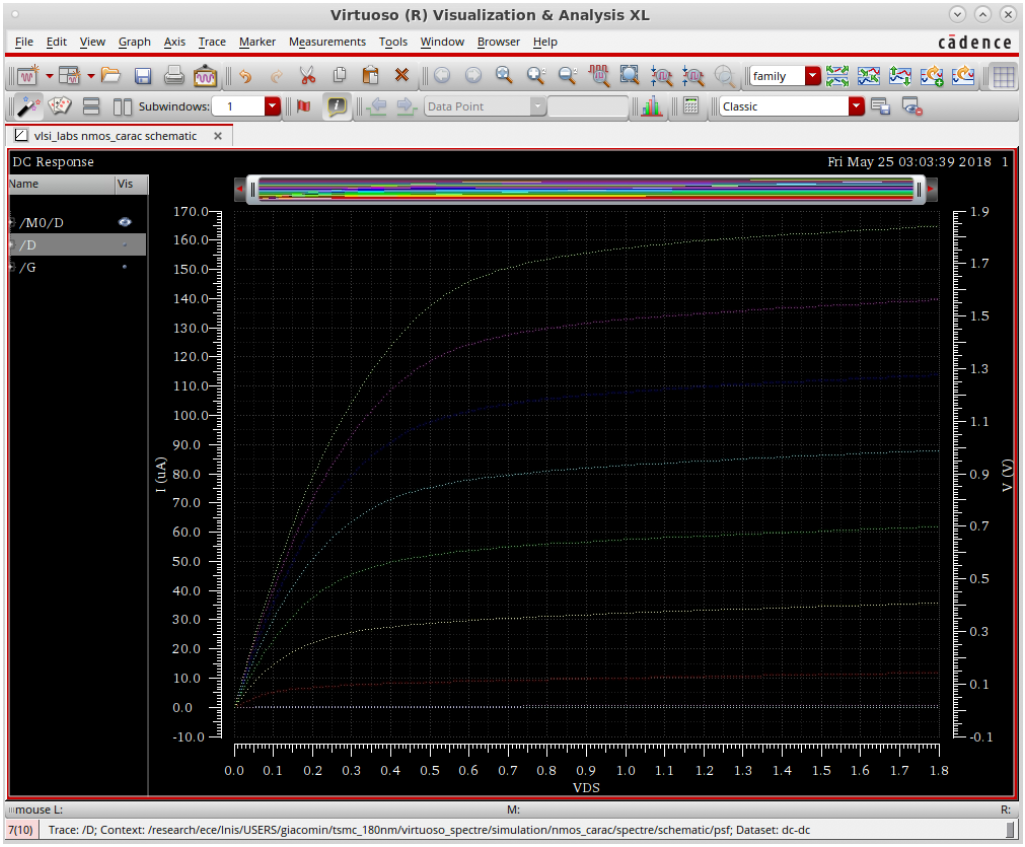

Figure 1.14: I-V curves for different *VGS*.

Assignment

(a) Report the I-V curves under different VGS voltages for your *nmos* transistor.
(b) What are the I_{on}, I_{off} of your *nmos* transistor? Remember that I_{on} is the maximum achievable current and I_{off} if the current when VDS is set to the supply voltage but when the gate is off ($V_{GS} = 0V$).

8. Do the same exercise (create a new cell for it) to characterize a *pmos* (*pmos*2*v* from the *tsmc*18 library) transistor. Do not forget that this time, the source of the *pmos* has to be connected to V_{DD} and that the *pmos* is *on* when $V_{GS} = 0V$.

Assignment

Report the I-V curves under different VGS voltages for your *pmos* transistor as well as its I_{on}, I_{off}.

R Don't forget that the power supply of this technology is $1.8V$. Also, don't forget to perform your I-V characterization for minimum sized *nmos* and *pmos* transistors.

1.4.3 CMOS Inverter Simulation

Now, you will create the schematic and symbol of a CMOS inverter. You will also perform some electrical simulations to study its transfer curve as well as its different delays.

1.4.3.1 CMOS Inverter Schematic and Symbol Views

1. Create a new schematic view in your design library for the CMOS inverter (*File -> New -> Cellview...* from the Library Manager) and name it *inv*.
2. Instantiate two transistors (an *nmos* and a *pmos*. Use regular transistors (*nmos2v and pmos2v* from the *tsmc18* library). Transistors should be sized to minimal length. Adapt the width of the *pmos* arbitrarily to start (2 the width of the *nmos*).
3. Create some wires as previously to connect the two transistors together, following the inverter schematic you saw in class. Don't forget to connect the source and the bulk of the *nmos* together, as well as for the *pmos*.
4. Create pins for your cell. Pins will define the different connections (interface) between a cell and its environment. If you instantiate a cell in other designs, the only accessible nets will be the previously defined pins. Pins are defined by the name and the direction (input, output or input-output). The purpose of the direction is to check for possible wrong connections (e.g. two outputs shorted together, or floating inputs). Typically, input-output pins are used for power supplies and bidirectional interfaces.

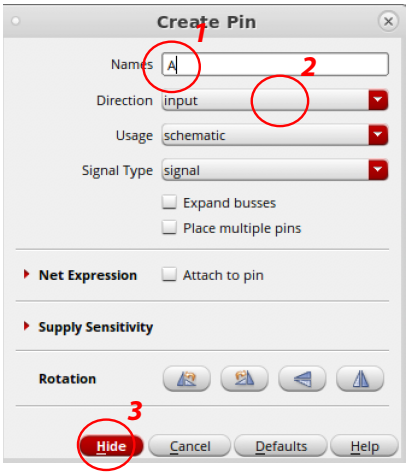

Figure 1.15: Schematic pin creation.

   - Create the input pin. To do so: *Create -> Pin* or press **p**. A window appears (Fig. 1.15, prompting you to enter the name (*A*) and the direction (*input*) of the pin. Click on *Hide* and place the pin on the schematic. You can either place them directly on the net they are supposed to connect to, or on the left of your schematic (and connect the nets later through some labels).
   - Repeat the same process for the output (*generallydenotedZ*) and power supply pins (V_{DD} and G_{ND}). Don't forget to specify the appropriate direction for each pin.
   - Your schematic should look like Fig. 1.16. Don't forget to Check and Save your schematic and ensure there is no errors or warnings.

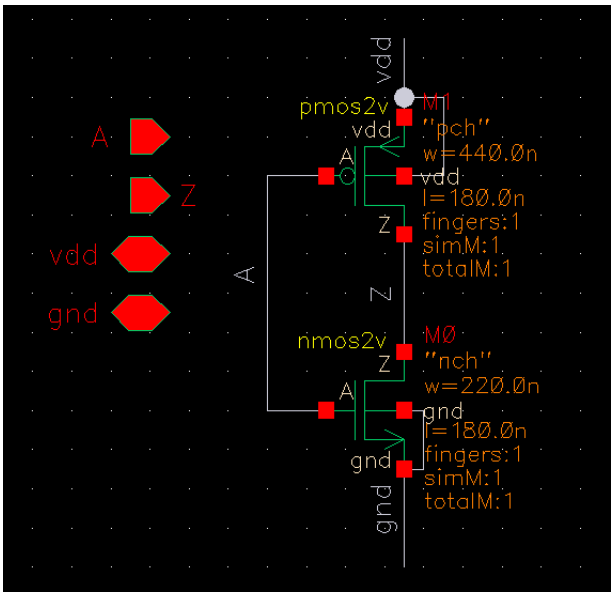

Figure 1.16: CMOS inverter schematic.

5. Check and Save your schematic (*File -> Check and Save* or **Shift + x**). Verify that there is no errors or warnings.
6. Now, you will create a symbol for your inverter. A symbol is useful when you work with larger designs and you need to instantiate previously designed gates (e.g. designing an adder with some XOR and AND gates, *etc.*), instead of redrawing the transistor schematic every time. The symbol will provide information on how to connect the cell from the outside. Note that the pins of your symbol should be the same ones defined in your schematic. To do so:

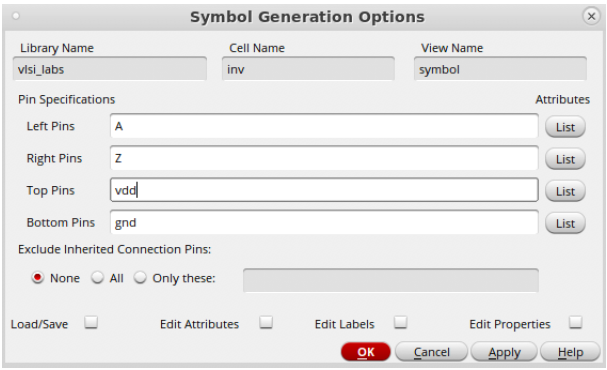

Figure 1.17: Cellview pin location.

   - On the schematic window, (*Create -> Cellview -> From Cellview..*).
   - A window appears. All the options should be set correctly so just press OK.
   - Another window appears (Fig. 1.17) where you can specify the location of the pins on your symbol. It is a good practice to set the inputs on the left, the output on the right and the power supply pins on top or bottom, depending on the direction you chose when you defined your schematic pins, the pins should be correctly placed here.
   - Click OK.
   - Shape your symbol how you want it, as depicted in Fig. 1.18 and Check and Save it. There should be no warnings or errors.

R You don't need to change the [@instanceName] and [@partName] labels in the generated symbol. When you instantiate the cell, these labels will display the instance name and the cell name respectively. By default, the instances you place in a schematic will be named I0, I1, I2, *etc*. To change the name of an instance, select the instance and press *Q*.

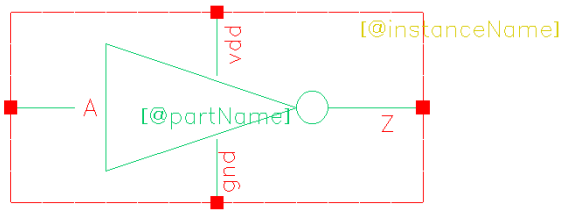

Figure 1.18: Inverter symbol.

1.4.3.2 CMOS Inverter Testbench View

You previously created a schematic view of your inverter. However, this view only contains transistors. To perform electrical simulations on it, you need to create a view where you will instantiate some voltage sources, ground points, *etc*. To do so, you need to create a new schematic cellview for your testbench in which you will instantiate your inverter.

1. Go to the Library Manager and create a new schematic cellview (*inv_testbench_dc* for instance).
2. Instantiate your inverter (you need to select the appropriate library and the inverter symbol cellview you just created) in your schematic.
3. We will study the *Voltage Transfer Characteristic* of the inverter so you first need to connect the input of your inverter to a DC source voltage (*vdc* from the *analogLib*). Edit its property to specify the DC voltage as a parameter VIN.
4. Instantiate another DC voltage source for the power supply. Set its DC voltage as a global parameter (VDD).
5. For the V_{DD} and G_{ND} symbols, use the *vdd* and *gnd* cells from the *analogLib* library.
6. For the load capacitor, use the *cap* cell from the *analogLib* library and sets its capacitance value to $20f$.
7. Don't forget to label your nets (input and output).
8. Your testbench should look like Fig. 1.19.

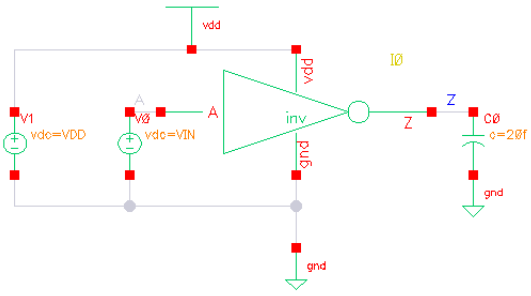

Figure 1.19: Inverter testbench schematic.

R At this point, you should have 2 cells: one for the inverter and one for the inverter testbench (in which the inverter is instantiated). For the rest of the labs, always create a different cell for your testbench and instantiate the standard cell you want to simulate in it.

1.4.3.3 CMOS Inverter DC Simulation

In this section, you will perform a DC simulation of an inverter in order to study the effect of the *pmos* width on the VTC of your inverter.

1. Launch ADE L.
2. Import your designs variables (*VIN* and *VDD*) and set them to an initial value of 1.8V.
3. Select the kind of analysis you want to run. In this case, you want to perform a DC sweep on *VIN* going from 0V to 1.8V (the setup is really close to what you do previously for the *nmos* characteristic).
4. Select the signals to be displayed (*in* and *out* here).
5. Run the simulation. Your curve should look like Fig. 1.20.

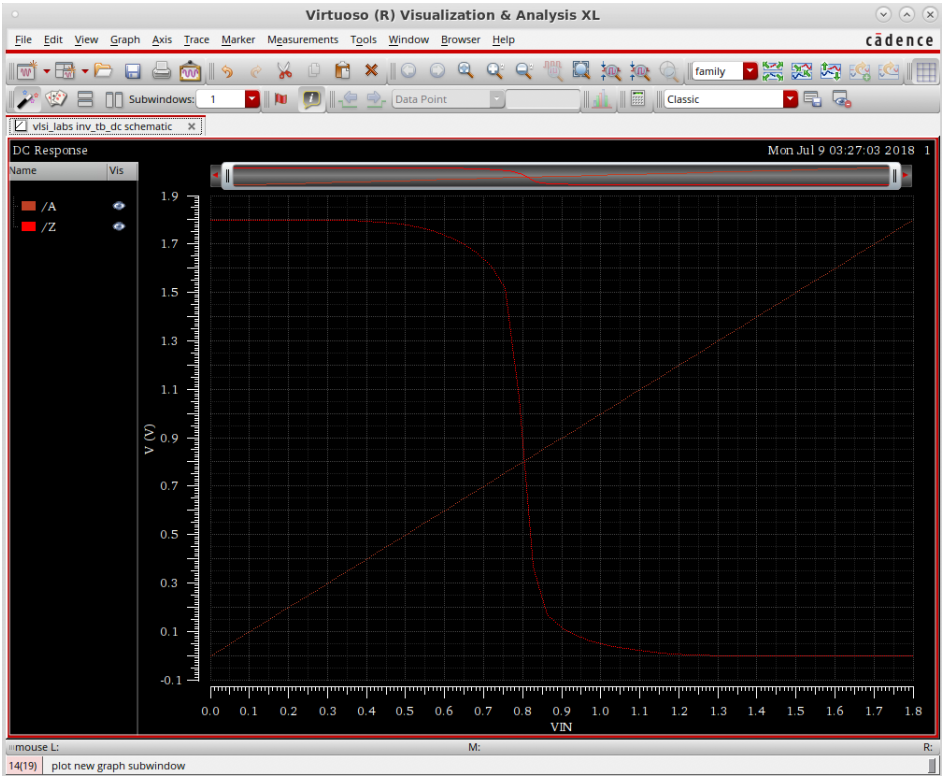

Figure 1.20: Inverter VTC curve.

Assignment

Plot the Voltage Transfer Curve (VTC), report the switching point of the inverter and specify the dimensions of your transistors.

6. Run a parametric DC analysis for different width of the *pmos* by sweeping the voltage from 0V to V_{DD}. To do so:

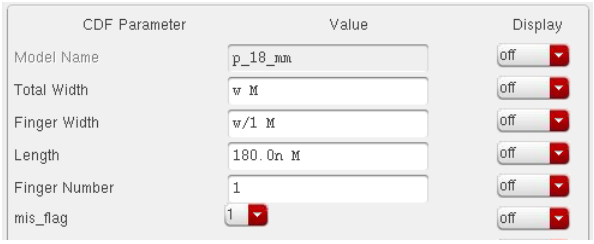

Figure 1.21: Setting the transistor width as a parameter.

   - Go to your inverter cell. To do so, you can either go to the library manager and open the inverter cell directly, or you can directly double click on the inverter symbol from your testbench schematic and click OK (in this case, you went down into the hierarchy. If you want to go back, press ctrl + e).
   - Change the width of the *pmos* as a parameter and not a fixed number as shown in Fig 1.21.
   - On the ADE window, do: *Variables -> Copy from Cellview* to import the new width variable on ADE. Don't forget to specify an initial value for your parameter in the left field (220nm for instance).
   - Go to: *Tools -> Parametric Analysis*. Specify the parameter to sweep as shown in Fig 1.22 (here *w* is the width of the *pmos*).
   - Run the simulation.

Variable	Value	Sweep?	Range Type	From	To	Step Mode	Total Steps
w	240n	☑	From/To	240n	1.5u	Auto	10

Figure 1.22: Parametric analysis setup.

Assignment

Plot the VTC curve for different width of the *pmos*. What is this effect called? Report for which value of the *pmos* width the switching point is equal to $V_{DD}/2$.

7. As you can see, the *pmos* needs to be very big to have a switching point equals to $V_{DD}/2$. This leads to an unacceptable area and induces big parasitic capacitances. **For the rest of the labs, we will use the approximation** $W_{pmos} = 2 * W_{nmos}$.

R In the next labs, don't forget to use the same approximation to size your transistors.

! Don't forget to change the width value of the *pmos* back to a fixed value (440nm). Otherwise, it will not be able to pass the LVS since the width will not be fixed.

1.4.3.4 CMOS Inverter Transient Simulation

In this part, we will perform another kind of simulation: transient simulation. It allows to study the behavior of your circuit overt time, when applying a specific input voltage sequence. Since the testbench will be slightly different, a good practice is to create a new testbench view for this particular simulation.

1. First, create a new testbench for the transient simulation. Since the testbench is really similar to the DC simulation (only the input voltage source will change), an easy way to do is to copy the DC testbench and modify it. To do so:
   - Go to the Library Manager, select your design library and your inverter testbench cell and right click and on and select *Copy*....
   - On the window, select the name you want (*inv_testbench_tran* for instance) for this new testbench in the appropriate field and click OK (Fig. 1.23).

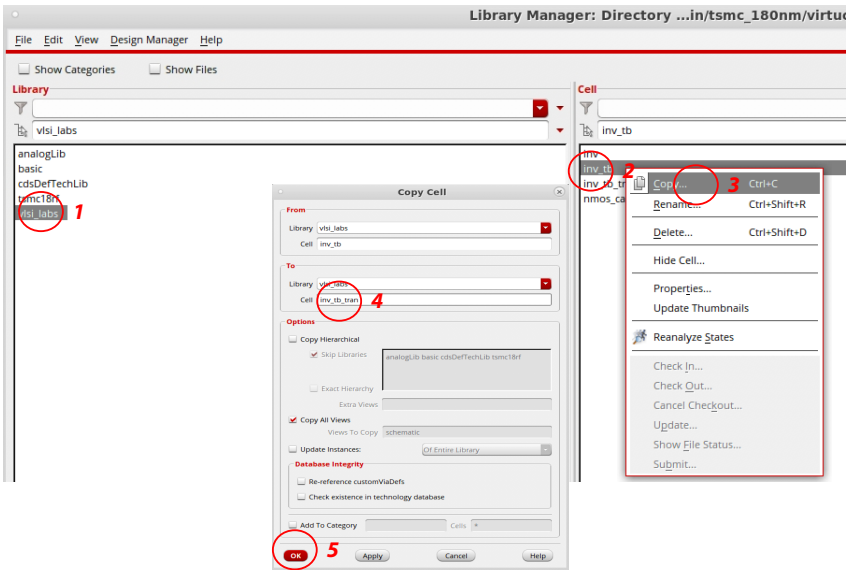

Figure 1.23: Copying a cell.

- Open the schematic cellview of your newly created testbench (*inv_testbench_dc*) and replace the input (not the supply one) source DC voltage by a pulse source *vpulse* from the *analogLib* library.
- Specify the parameters of your pulse source as shown in Fig. 1.24 (a). Note that this time, the DC voltage is specify as *Voltage2* and not as DC Voltage. Here, the period of your source is specified as a parameter *period*. You will be able to change this parameter in the ADE window later on. The rising and falling times depends on your period.

(a)

CDF Parameter	Value	Display
Frequency name for 1/period		off
Noise file name		off
Number of noise/freq pairs	0	off
DC voltage		off
AC magnitude		off
AC phase		off
XF magnitude		off
PAC magnitude		off
PAC phase		off
Voltage 1	0 V	off
Voltage 2	VDD V	off
Period	period s	off
Delay time		off
Rise time	0.01*period s	off
Fall time	0.01*period s	off
Pulse width		off

(b)

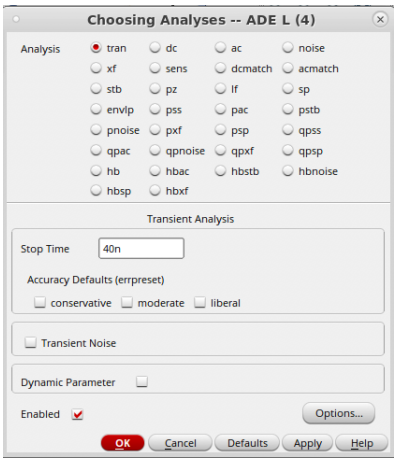

Figure 1.24: (a) Transient simulation parameters; (b) Voltage pulse source parameters.

2. Launch ADE.
3. Import your design variables and set some initial values. Use a period of 2ns.
4. Specify a transient analysis (*Analyses* -> *Choose...* -> *tran*) as shown in Fig. 1.24 (b) and just specify the stop time (8ns in this case in order to observe 4 period).
5. Define the delay measurements. To measure some stuff from the curves (falling time, propagation delay, maximum voltage etc.), we will use the Cadence calculator. To do so, first open the Calculator (in the ADE window: *Tools* -> *Calculator...*).The calculator window should appear, as shown in Fig. 1.26. The Calculator is a very powerful tool which can help you define expressions and waveforms, plot circuit time or frequency responses, perform useful transforms, signal post-processing and/or analysis. It has many pre-defined mathematical and processing functions and also allows you to define your own functions.

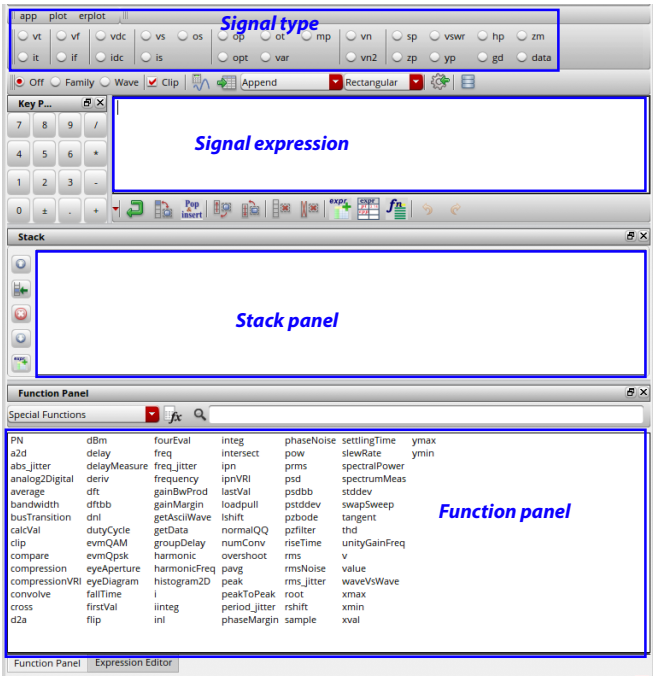

Figure 1.25: ADE calculator setup.

The *Signal type* panel allows you to choose which type of signal you want to consider (for instance, *vt* stands for transient voltage). The *Signal expression* panel which displays the expression of the signal you selected from your schematic through the *Signal type* panel. Finally, the function panel allows you to choose which type of function you want to use (delay calculation, frequency response ,*etc*).

6. To compute the propagation delay:
   (a) In the function panel, search for *delay*.
   (b) Select the appropriate signal you want to measure from (in this case, we want to measure the rising and falling time so select the output signal of your inverter): click on **vt** and select the signal on the schematic (step 1 in Fig. 1.26).
   (c) Copy/paste the name of the signal in the Signal 1 and signal 2 fields of the calculator (step 2 in Fig. 1.26). Here, signal 1 and signal 2 are the same since we are measuring the falling delay which is on a single signal.

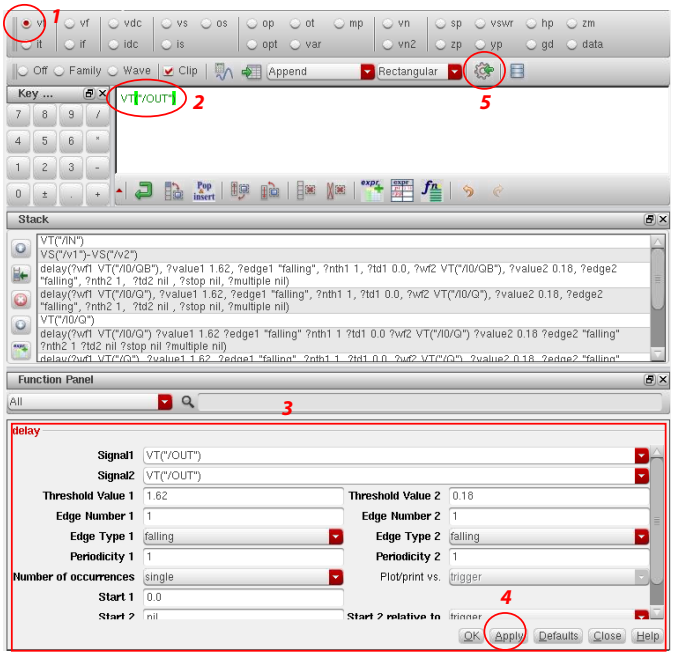

Figure 1.26: ADE calculator setup.

(d) Specify the other fields as in the picture (step 3 in Fig. 1.26). In this example, we show how to measure the falling delay (although the rising delay is obtained similarly). Threshold Value 1 is the value for which the time will be measured for the first point. If you want to measure the falling delay, it is 90% of V_{DD} so 1.62V. In the same manner, Threshold Value 2 is the value for which the time will be measured for the second point, so it is 10% of V_{DD} so 0.18V. Both Edge Types are falling since you are measuring a falling delay.

(e) Click on *Apply* (step 4).

(f) Finally, send the expression the ADE outputs (step 5). If you go back to the ADE window, the expression should be in the display output panel. Right click on it and select *Edit* and change the *Name (opt.)* field to an appropriate name (e.g. *falling delay*).

(g) Redo the previous steps for the rising delay and the propagation delays. For the rising delay, note that both edges should be rising and the Threshold Values should be changed appropriately. For the propagation delay, don't forget to change Signal1 and Signal2 (this time, you are measuring a delay between the input and the output and not a delay on a single signal). Also, don't forget to carefully choose the threshold values (50%) and the appropriate edge types.

Assignment

1. Report the curve of the input/output of your inverter and the rising and falling times. Which one is faster? Why?
2. Report the propagation delay of your inverter. How could you optimize it? At which cost?
3. We considered a period time of 2ns. Is it slow, fast or appropriate compared to the frequency time of chips based on the 180*nm* process? Justify your answer.

1.5 Assignment and Checkpoint Summary

Write a report and answer the assignments asked during the lab, which are summarized below. Do not forget to validate the checkpoints, summarized below as well, by an assistant before the end of the lab.

Assignments

1. Report the I-V curves under different VGS voltages for your *nmos* transistor.
2. What are the I_{on}, I_{off} of your *nmos* transistor? Remember that I_{on} is the maximum achievable current and I_{off} if the current when VDS is set to the supply voltage but when the gate is off ($V_{GS} = 0V$).
3. Report the I-V curves under different VGS voltages for your *pmos* transistor as well as its I_{on}, I_{off}.
4. Plot the Voltage Transfer Curve (VTC), report the switching point of the inverter and specify the dimensions of your transistors.
5. Plot the VTC curve for different width of the *pmos*. What is this effect called? Report for which value of the *pmos* width the switching point is equal to $V_{DD}/2$.
6. Report the curve of the input/output of your inverter and the rising and falling times. Which one is faster? Why?
7. Report the propagation delay of your inverter. How could you optimize it? At which cost?

8. We considered a period time of 2ns. Is it slow, fast or appropriate compared to the frequency time of chips based on the 180*nm* process? Justify your answer.