Budapest University of Technology and Economics
Department of Electron Devices

BSc Course in Microelectronics

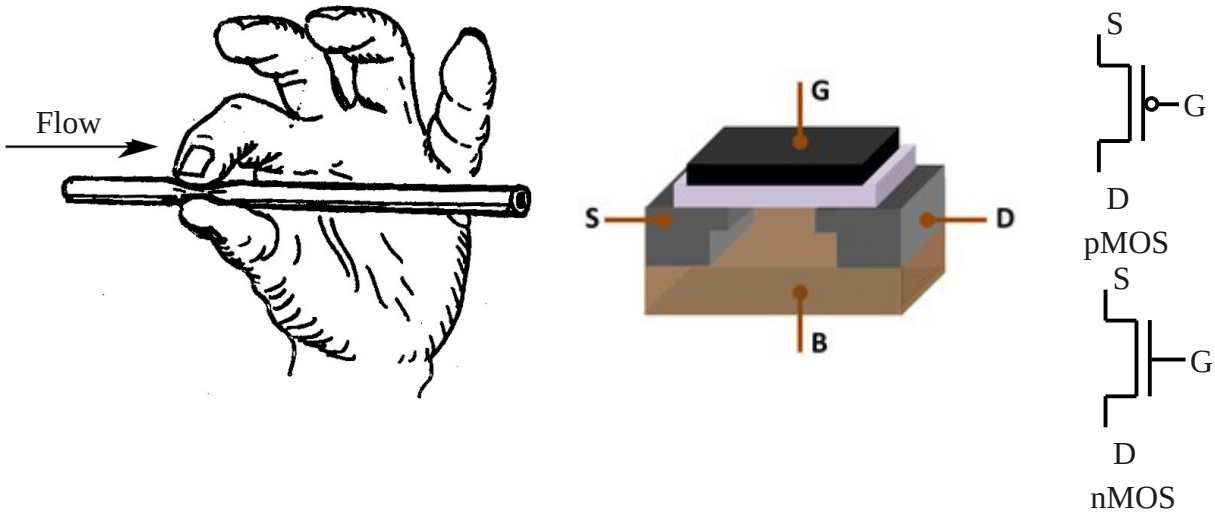
Laboratory Practice: CMOS circuit design and simulation

- Read through this summary carefully and answer the questions listed on the last page (similar questions are expected at the entry exam)
- Learn about how to use the simulator environment. Use this manual in the lab practice to help yourself setting up the simulator.

Introduction to field effect transistors

The metal–oxide–semiconductor field-effect transistor (MOSFET, MOS-FET, or MOS FET) is a transistor used for amplifying or switching electronic signals. Unlike the bipolar junction transistor, the MOSFET is a unipolar device.

The MOSFET is a four-terminal device with source (S), gate (G), drain (D), bulk (B). In practical applications the bulk is shortened to the source, therefore the bulk is not shown on schematic symbols. The MOSFET is by far the most common transistor in both digital and analog circuits, though the bipolar junction transistor was at one time much more common.



a) b) c)

Fig 1. a) understanding the field-effect b) cross section of a MOSFET transistor, c) schematic symbols of pMOS and nMOS transistors

In enhancement mode MOSFETs, a voltage drop across the oxide induces a conducting channel between the source and drain contacts via the field effect. The term "enhancement mode" refers to the increase of conductivity with increase in oxide electric field that adds carriers to the channel, also referred to as the inversion layer. The channel can contain electrons (called an nMOSFET or nMOS), or holes (called a pMOSFET or pMOS), opposite in type to the substrate, so nMOS is made with a p-type substrate, and pMOS with an n-type substrate.
The operation can be imagined by a straw tightened by two fingers. The flow-through rate can be affected by the force of tightening. In field effect transistors the channel can be closed or opened by applying external forces as well, though the external force is the voltage applied to the gate electrode. Therefore the electric current flowing through the channel (source to drain) is affected by the gate voltage applied (drain current is controlled by gate voltage). Note, that in bipolar junction transistors the current flowing through the device (from emitter to collector) is affected by the base current (collector current is controlled by base current).

MOSFETs in digital circuits

MOSFETs are commonly used in digital circuits. For investigating the digital operation, the following rules may applied:

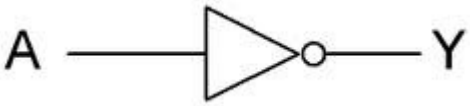
- In switching mode (in digital circuits) only two state of the transistor is utilized: the channel conduct, when we say 'the transistor is opened', or the channel does not conduct, when we say 'the transistor is closed'.
- Binary values are corresponding voltage values.
  E.g. binary 1 means 5 V, binary 0 means 0 V.
- From this aspect the operation of the nMOS and pMOS are the opposite.
- The nMOS transistor is normally closed, and opens when a positive voltage (e.g. 5 V) is applied to the gate electrode. When the gate voltage is 0 V, the nMOS transistor is closed.
- However the pMOS transistor closes when a positive voltage (e.g. 5 V) is applied to the gate electrode, but the channel is opened otherwise (e.g. the gate voltage is 0 V).
- Schematically, when the transistor is opened, it can be substituted by a short. When the transistor is closed, it can be substituted by an open.

Digital circuits are commonly built of using both pMOS and nMOS transistors. This type of digital circuits is called cMOS (means complementary MOS). A cMOS circuitry is consists a pMOS circuit block connected to the power supply (Vdd) and an nMOS circuit block connected to the ground (Vss). Two important characteristics of CMOS devices are high noise immunity and low static power consumption. Since one transistor of the pair is always off, the series combination draws significant power only momentarily during switching between on and off states. Consequently, CMOS devices do not produce as much waste heat as other forms of logic, for example transistor-transistor logic (TTL) or NMOS logic, which normally have some standing current even when not changing state. CMOS also allows a high density of logic functions on a chip. It was primarily for this reason that CMOS became the most used technology to be implemented in VLSI chips.

CMOS inverter

The simplest digital circuit is the inverter. An inverter has an input and an output. The output is always the opposite value of the input. The figure above describes the way of operation. When the input is '1' (gate voltages are e.g. 5 V) the pMOS closes and the nMOS opens, therefore the output is shorted to the ground. The output voltage equals to the ground potential, the digital value is '0'. If the input is '0' (gate voltages are 0 V) the pMOS opens and the nMOS closes. The output is shorted to the power supply, therefore the output voltage refers to '1' (e.g. 5 V).

IN	OUT
A	NOT A
1	0
0	1



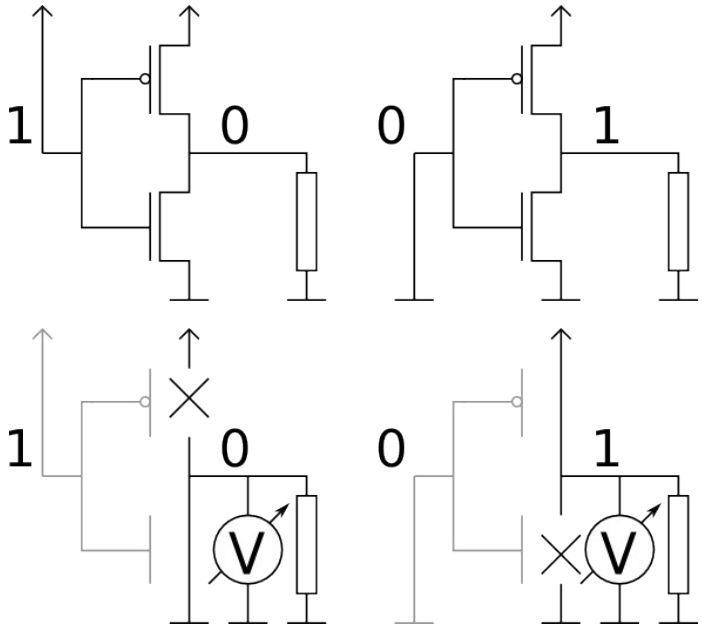


Fig 2. The inverter operation

Questions

1. What MOSFET stands for? What is the difference between nMOS and pMOS? How the terminals are called?
2. Describe the main differences between a bipolar juncion transisor and a field effect transisor.
3. What is cMOS? What are the main advantages of cMOS circuits?
4. *Collect infromation from the internet:* What is the minimal feature size (MFS) for a modern CPU today?
5. Describe the operation of a cMOS inverter, if the input is 1 (/ 0).
6. How an electric circuit is being verified by simulation? Describe the steps shortly!
7. What are the requirements to run a simulation?

Simulation of a cMOS inverter using CAD software [1]

Conventions Used

There will be several conventions used in this manual. The mouse has only two buttons but the scrolling wheel between them can be depressed, too, this will be referred to as the middle button. In the following there is some terminology explained which will be used in relation to mouse operations.

click left	quickly press and release the left mouse button
click middle	quickly press and release the middle mouse button (scrolling wheel)
click right	quickly press and release the right mouse button
drag left	press and hold the left mouse button while moving the mouse
drag middle	press and hold the middle mouse button while moving the mouse
drag right	press and hold the right mouse button while moving the mouse

If more than one OPUS window is open then the relevant window will be specified by adding *WWW:* for the window WWW.

If a double target *xxx->yyy* is specified with clicking, that may happen to be two separate clicks at *xxx* and *yyy* or a drag from *xxx* to *yyy*, depending upon how the popup menu for yyy comes up.

<...>	depress the key on the keyboard that corresponds to what is inside the brackets (either a character or a special key like CR (carriage return or enter), ESC (escape), SHIFT, CTRL, ALT.
type **something**	you should type (verbatim) whatever is printed boldfaced.

Starting OPUS

The very first start only initializes the design environment. Left click at the icon **IC Design Framework** on the desktop. A new LINUX shell comes up and asks for the design-directory of OPUS. Type **ams37 <CR>**, the name of the recently established subdirectory.

In return OPUS offers the available technologies. *AMS 0.35 um CMOS (c35b3)* has to be used, so type **1<CR>**. Then OPUS reports that several setup files have been created. This happens only at the first start of OPUS. Afterwards send this window to the panel with a left click at the upper right corner.

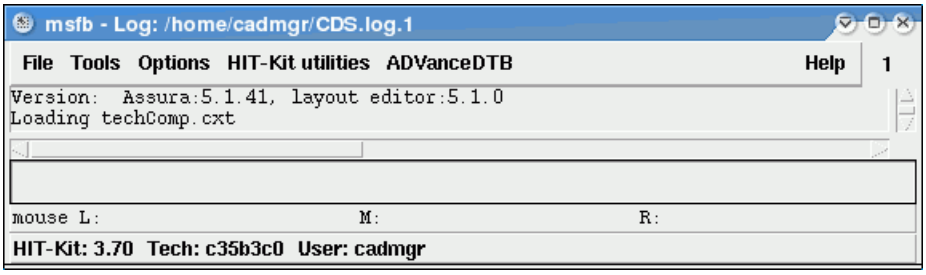


Fig. 1 Command Interpreter window

OPUS goes on. The *Log* window appears with some logging messages, then it changes to the *msfb-Log* window which is called the *Command Interpreter Window (CIW)* because it can accept commands which you type in (Fig. 1.). The library manager window, too, starts automatically, but

1 Based on P. Gaertner: AMS Hit Kit 3.7 manual, BME DED 2006

here you have to exit OPUS by clicking at **msfb:File->Exit**. So the initialization is done. When you start again, clicking at the icon, then OPUS won't ask any more question and you can go on using the library manager window.

The library manager window can be used for opening existing libraries or cells or creating new ones. The left column of the library manager window is a list of the current (accessible) libraries. Among these *PRIMLIB* contains the transistors you will need for the inverter.

Left click at **PRIMLIB**. The middle column shows the elements of *PRIMLIB*. Left click at **nmos4**. This is the basic n-MOS transistor. In the third, rightmost column you can see several views of *nmos4*. Of these you will need the *symbol* view for the schematic and the *layout* view for building the layout.

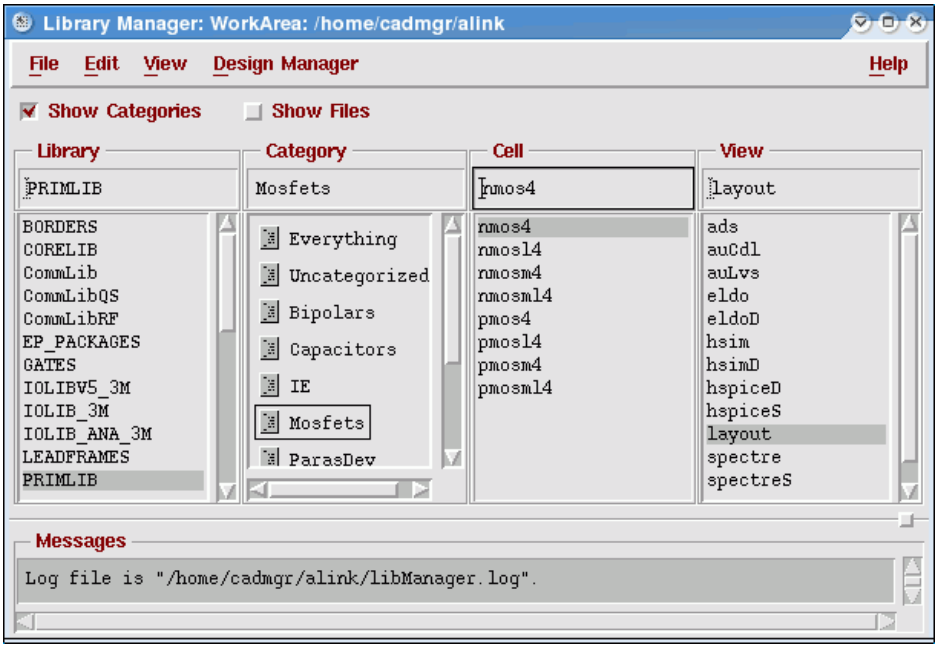


Fig. 2 Library Manager window

Create a new working library

Before building the schematic you have to create a working library. In the *Library Manager* left click at **File->New->Library**. A dialog box appears, asking for the place and name of the new library (Fig. 3.).

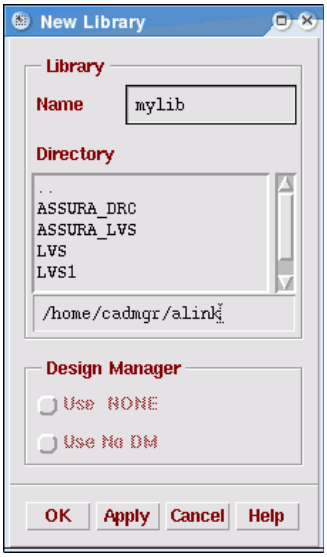


Fig. 3 New Library dialog box

Leave the directory at the default, and enter a name for your working library where you are going to design the inverter, such as, for instance, *mylib*. Left click the **OK** button. Cadence now creates a new subdirectory named *mylib* in its home directory (*ams37*). A new window will appear asking information about the technology file. The second option, **Attach to an existing techfile**, will be used, click at it. Then click on the **OK** button. A small dialog box appears asking for the existing techfile (Fig.4.). Left click at the Technology Library button. A list of possible choices pops up. Click at **TECH_C35B3** and then **OK**. Your library is created now and you should be able to locate the new library *mylib* in your Library Manager.

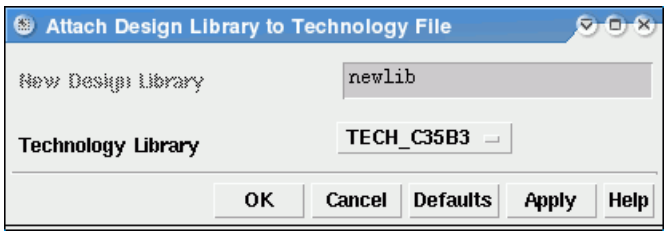


Fig. 4 Choosing the technology for the project

Create the schematic of the inverter

In the *Library Manager* left click on **File->New->Cell view**. The *Create New File* form appears (Fig. 5.).

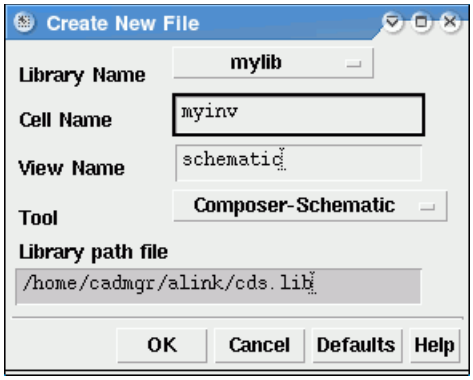


Fig. 5 Specifying the name and view of a new cell

Type a meaningful name in the *Cell Name* block, such as e.g. *myinv*. In the View Name block type **schematic** or from the *Tool* menu choose *Composer-Schematic* and the View Name block will be automatically filled. Set the library for the would-be cell *mylib*. Left click the **OK** button. The *Virtuoso Schematic Editing window* should show up (Fig. 6:).

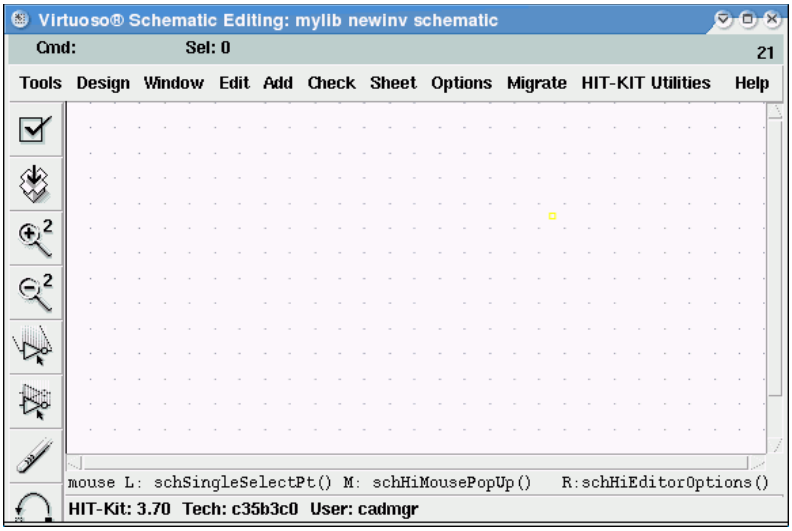


Fig. 6 Schematic Editing window

Left click **Virtuoso Schematic Editing:Add->Instance**. The *Add Instance* dialog box appears (Fig. 7.). Type **PRIMLIB** in the Library field. To choose a four-terminal NMOS transistor type **nmos4** in the *Cell* field and **symbol** in the *View* field. Note that you can use the *Browse* button in order to browse through the libraries and find the cell you want. Generally, typing in known names is faster.

When OPUS learns that you want to place an instance of a transistor then it adds fields to the box for the parameters of the transistor, already containing default values. Just change the *Width* to **2u** (two microns).

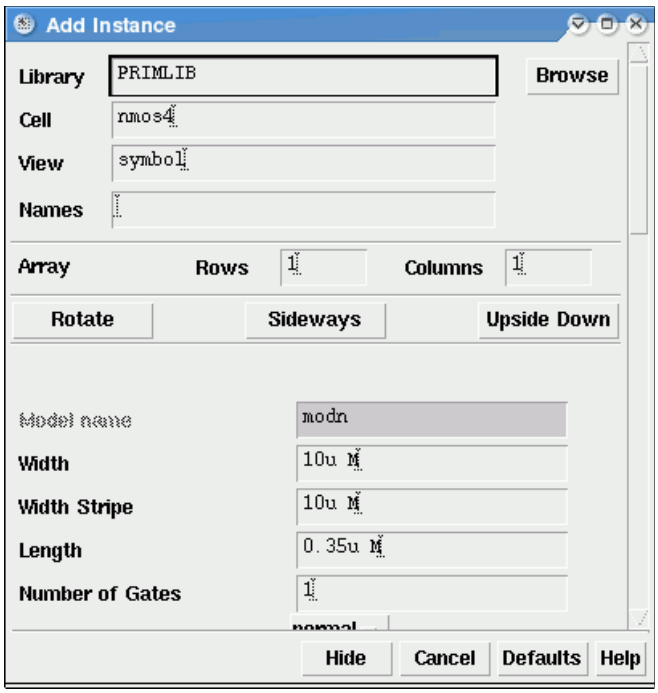


Fig. 7 Add Instance – specifying a transistor

Move the cursor into the editing window. Notice that there is an nmos transistor there instead of the normal cursor. Position it where you want to put the transistor, and place it by a left click. Having placed the first component, change the name of the transistor in the dialog box to **pmos4** and the width to **5u**. Now you have prepared the second half of the inverter and you should place the pmos transistor somewhere over the nmos device so that they can be connected by a straight line. If you type <ESC> then the dialog box disappears and OPUS is waiting for your next command. This will be adding external pins for the inverter.

Left click **Add->Pin**. The *Add Pin* dialog box appears (Fig. 8. next page). Type **in out** in the *Pin Names* field for the pins of the inverter. Set the *Direction* to *input*. (Note that the order of the pins is not important. You may even place one pin at a time and repeat the procedure.)

Move the cursor into the editing window. A pin symbol appears with a small square at the right edge. Place it left of the transistors in the middle with a left click.

In the dialog box *in* disappears from the pin-list, only *out* remains. Change the *Direction* to *output*. In the editing window an output pin symbol appears, having a small square at the left edge. Place it right of the transistors in the middle with a left click.

In a simple case power supply pins (vss, vdd) should be added, too. However, in an IC power supply is provided in a centralized way. In the schematics it is done by the global nodes *gnd!* and *vdd!* while in the layout there are the power rails which do it. Global node names end with an exclamation point (!). Contact to the global nodes *gnd!* and *vdd!* is established by placing instances (small symbols) of the cells *ground* and *vdd*. They are stored in the library *analogLib*. The procedure is the same as for the transistors. Invoke the *Add Instance* window with a left click at **Add->Instance**. Select with the browser the library *analogLib* and then the cells *ground* and *vdd*. Place them underneath and over the transistors, respectively.

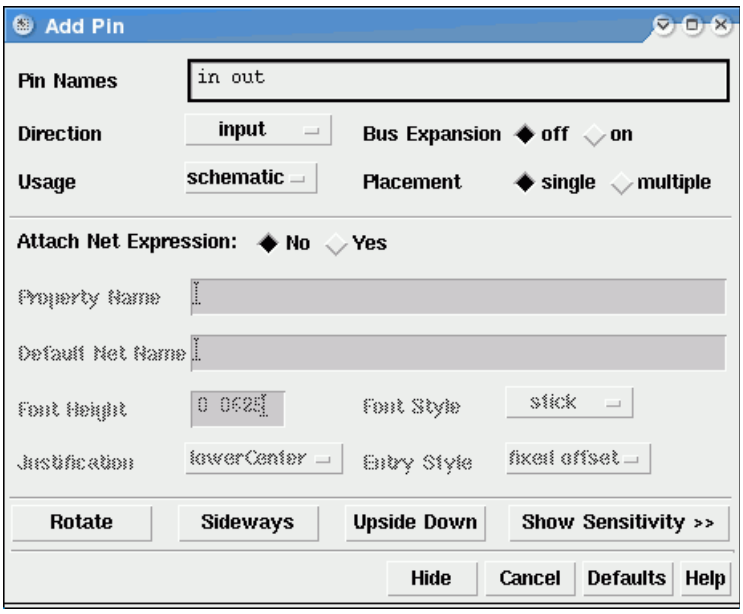


Fig. 8 Add Pin dialog box with pin specification

Now we'll add the wires to make things work. Click **Editing:Add->Wire**. Notice that as you get closer to one pin than to another (including those on devices), a small diamond will show up inside of or around that pin. That is where you can click to connect a wire.

To begin with, left click the diamond at the pin of the symbol *vdd*, then left click on the source terminal of the PMOS transistor. The first connection is finished. Now left click on the drain terminal of the PMOS transistor and then on that of the NMOS transistor. Follow that with a wire from the source of the NMOS transistor to the pin of *gnd*. Make one more vertical connection between the gates of both transistors. Now, left click on the diamond in the *in* pin. Move the cursor horizontal to the wire you connected the two gates together with. A diamond will form around the cursor, as long as it is on the wire. Left click. You have just connected the input to the gates of both transistors. Repeat the procedure from the output pin to the wire connecting the drains of both transistors.

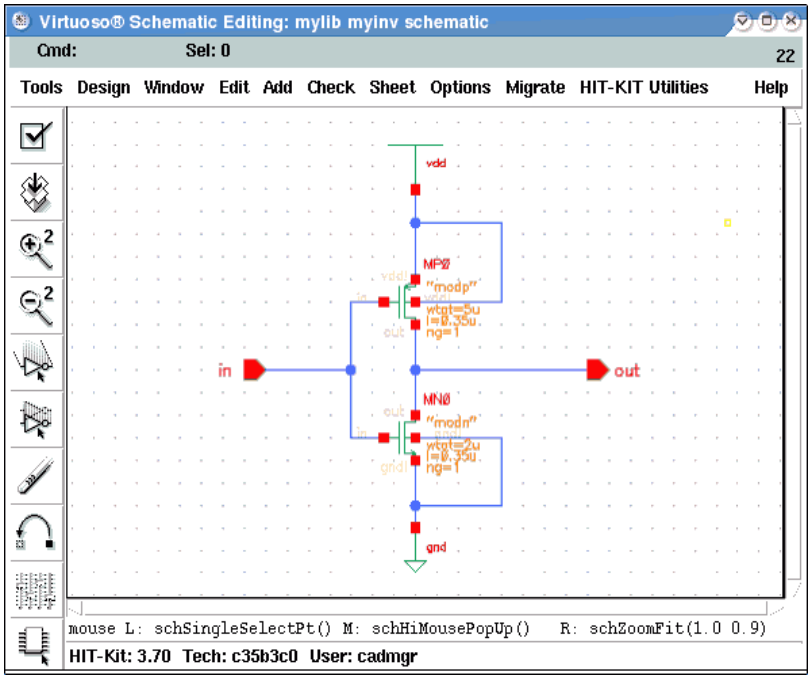


Fig. 9 Complete circuit diagram of the inverter

What remains is connecting the bulk (body) terminals of the transistors. Left click on the bulk terminal of the PMOS transistor. Move the cursor a little right, and left click. The wire will turn here. Now move upwards halfway to *vdd*. Left click again and move to the wire connecting the drain and *vdd*. Connect the bulk of the NMOS transistor to *vss* in a similar manner.

If you happen to put a wire where you don't want it to go, you can delete it by left clicking **Editing:Edit->Delete** and then left click on the object you want to delete (wire, pin, component, etc.).

Once you have done editing, left click the **check mark** (√) icon on the left side of the screen. This will check your work for connection errors and will save your cell (more exactly: its schematic view!) in the library. You can accomplish the same by left clicking **Editing: Design->Check and save**. Fig. 9 shows what the complete schematic should look like.

This very simple schematic of an inverter will likely be flawless but in more complex designs OPUS may find errors which will be highlighted after the check (blinking). In such a case you may click **Editing:Check->Find Marker**. Then the *Find Marker* window opens (Fig. 10.) and you will find there the list of the highlighted errors and warnings with the reasons stated.

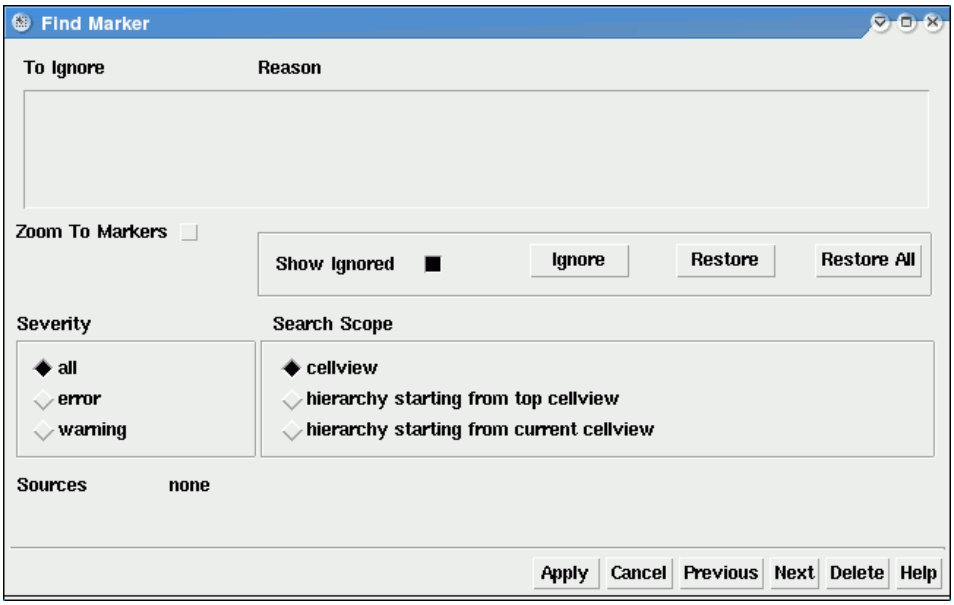


Fig. 10 The window for the list of errors and warnings

Plotting the Schematic of the Inverter

Now that the schematic is complete, you will want to print it out. To do this left click **Editing:Design->Plot->Submit**. The *submit Plot* window should appear (Fig. 11, next page). The default settings usually comprise your schematic and the plotter nearby, so a click on the **OK** button will start plotting. Ensure that the *Header* button is *NOT* selected. This option would produce an extra page with general information on your plot like name and size etc..

If the paper box of the plotter is empty and you happen to want to plot on a sheet of paper which only has one free (empty) side, then make sure that the empty side of the paper looks downwards.

Create a symbol for the inverter

The symbol editor lets you create a "black box" description of a cell using labels, pins, shapes, notes and a selection box. Symbols enable you to introduce hierarchy into your designs. In the *Library Manager* left click on **File->New->Cell view**. The *Create New File* form appears. Ensure that the library name is *mylib*. Fill in the cell name *myinv* and the view name *symbol*. Left click the **OK** button. The *Virtuoso Symbol Editing* window should show up. (Fig. 12. on the next page shows it with the would-be result.).

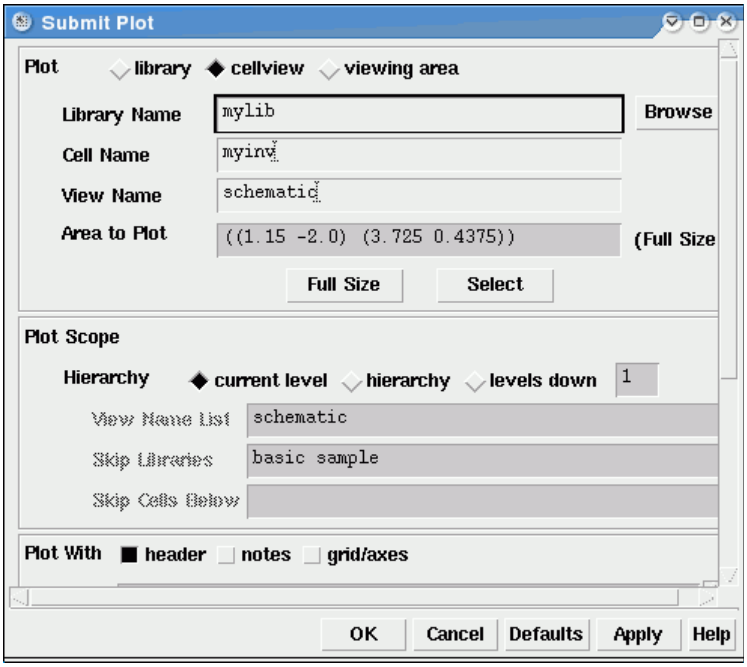


Fig. 11 Sending a circuit diagram to the plotter

Start drawing with a triangle to represent the inverter body. Left click **Editing:Add->Shape->Polygon**. To draw a polygon, left click at a start point and then click at the corners of the shape you want to create. To finish the polygon, click again on the start point. Since we have an inverter, we need an "inverter-like" triangle.

As to the size of the symbol: Note that there are small white dots in the black background of the editing window. If you carefully move the cursor then you will find that its movement is quantized, between two dots it can make 16 small jumps. The triangle should occupy about a "4 by 4 jump" area.

The inverter needs a negation circle at the sideways corner of the triangle, so left click **Editing:Add->Shape->Circle**. Left click at the would-be center of the circle and then at the corner of the triangle. A radius of "one jump" is recommended.

Next you have to create pins for the symbol. It is similar to creating pins in the schematic but the pins look different. They consist of a little red dot and of a piece of line. The dot is the pin itself. The line binds it to the body of the symbol, its length can be adjusted.

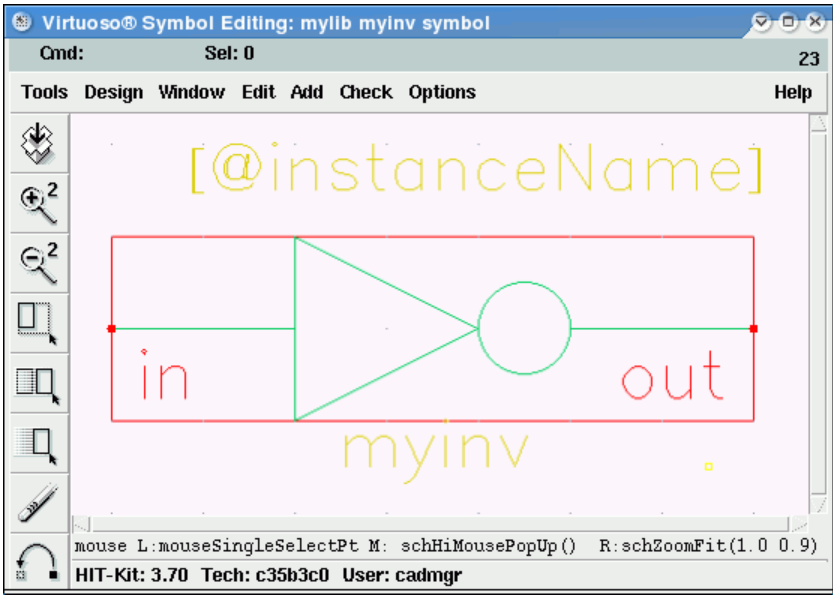


Fig. 12 Symbol Editing window with the symbol

Left click **Editing:Add->Pin**. The *Add Pin* box shows up. Type the pin names, they *must exactly match* those of the schematic: **in out**. Do not forget to set the correct direction for the pins before placing them. Moving the cursor to the editing window the pin appears. With left clicks on **Add Pin:Rotate** you can change the direction of the connecting line. Place the pins so that the red dots are at the far end and the connecting lines join the body of the symbol. At last the position of the pin names have to be adjusted so that the symbol looks nice. Moving the cursor to a name a yellow box appears around it. Now you can left drag the name to its final position.

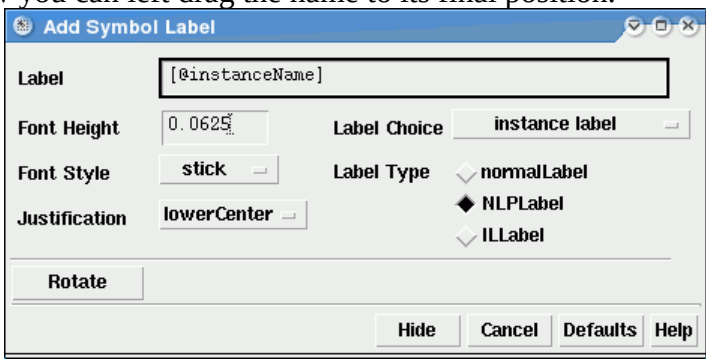


Fig. 13 Adding a label to the symbol

Next we want to add two labels to the symbol. Left click **Editing:Add->Label**. The *Add Symbol Label* dialog box should appear (Fig. 13.). The usual default setting is *[@instanceName]*, Label Choice: *instance label*, Label Type: *NLPLabel*. With this setting you only have to move the cursor to the editing window. The label *[@instanceName]* at once appears and you can place it with a left click. The next label is the name of the cell. Fill into the label field **myinv** and choose *Label Type* **normalLabel**. Place it again with a left click.

The last thing to add is a selection box. This will tell the software how much of the symbol is actually used. Left click **Editing:Add->Selection Box**. Left click the **Automatic** button. The selection box will be automatically drawn.
The symbol is now finished and you can save it by left clicking **Editing:Design->Save**. If the pin names and attributes do not match those of the schematic then warnings show up and you have to correct the mismatch.

Simulate the schematic

The functionality of an integrated circuit is verified by simulation. For a simulation the following things are needed:

- **netlist**
  - a text file consisting how the components are connected together and what are the component values (e.g. the resistance of a resistor)
- **power supply**
  - the power supply applied to operate the circuit
- **input signal source**
  - the signal source which drives the input(s) of the circuit
- **stimuli**
  - the time function of the signal(s) of the signal source(s) allows to simulate all the functionality of the circuit

The netlist can be extracted from the schematic, it is usually done automatically. Power and input signals are provided by generators. They might be directly added to the schematic but this method is not recommended. Instead, a test bench should be built which takes the cell to be tested as an instance and provides the necessary simulation environment. This has several advantages. The cell remains unchanged and independent of the simulation. It is quite easy to simulate and compare different versions and views of the cell, you only have to specify a cell and a view in the test bench.

Create a test bench

To keep things apart, test benches are usually built in a separate library. For this reason open a new directory *testlib* just the same way you created *mylib* for the schematic of the inverter (page 4). Then open a new cell with schematic view, e.g. *test_inv*. The first element of the testbench is the cell to be tested. Left click **Editing:Add->Instance**. A dialog box comes up and you can either type *Library* **mylib**, *Cell* **myinv**, or you can browse through the libraries and specify it there. Place the symbol of the inverter in the middle of the screen.

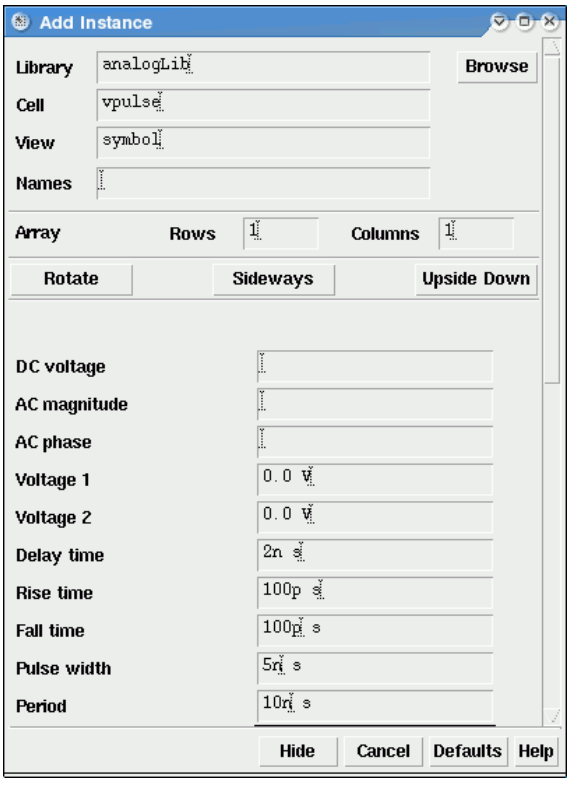


Fig. 14 Specifying a pulse generator

Left click **Add->Instance->Browse**. Select the generator *vdc* from the library *analogLib*. Set the field *DC Voltage* to **5 V**. This will be the power supply, place it far left. Now you might go on browsing for other components but in this case it can be done easier, too. Change the name of the cell to *vpulse* and close it with a <TAB>. The dialog box changes and parameter fields of the pulse generator appear (Fig. 14.). *Voltage 1* is the low level of the pulse, fill in **0**. *Voltage 2* is the high level of the pulse, fill in **5** for 5 Volts. Specify the timing as follows: *Delay time* = **2n**, *Rise time* = . **1n**, *Fall time* = .**1n**, *Pulse width* = **5n**, *Period* = **10n**. This generator will drive the inverter, you may change the Instance Name to *drv*, and place it near the input of the inverter. Change the name of the cell to *cap* and close it with a <TAB>. The dialog box changes and now the value of the capacitive load can be set to **.3p** (0.3 pF). Place the capacitor near the output of the inverter.

Now left click **Add->Wire** or type simply <**w**>. Make the common ground net connecting the lower terminals of the three components. Next connect *vpulse* to the input and the capacitor to the output of the inverter. Place one more piece of wire to the output and add an output pin named *out* (left click **Add->Pin**, etc.). Left click **Editing:Add->Wire Name**. Fill in *Names*: **uin**. Move the cursor to the wire connecting *vpulse* and the input of the inverter and place the name onto it. In the next step you have to provide for the global *vdd!*. It is done by placing a *vdd* symbol from the library *analogLib* and connecting it to the positive terminal of the *vdc* generator.

It is important for the simulator that the ground net has the name *gnd!* and the internal node number zero. This can be achieved by placing one more component. Left click again **Editing:Add->Instance** and fill in: *analogLib* for the Library and *gnd* for Cell name. Place the ground symbol underneath and connect it to the ground net by a piece of wire. If you happen to get very unusual and unlikely voltage values resulting from the simulation then check if this condition is fulfilled! The test bench is complete, left click **Design->Check and Save** (Fig. 15.).

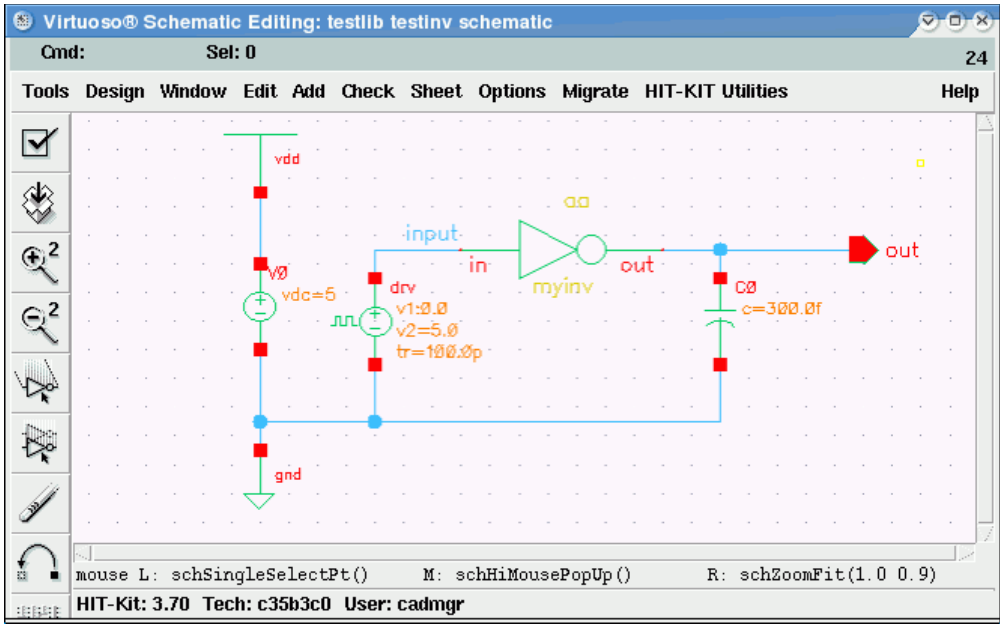


Fig. 15 Testbench – test environment for the inverter

Spice simulation with *eldoD* of Mentor Graphics

We are going to analyze the DC transfer characteristics and the transient behavior of the inverter. Open the schematic *test_inv* in the library *testlib*. Left click **Tools->Analog Environment**. The *Virtuoso Analog Design Environment* window opens (Fig. 16). In the status bar (second from top) the default simulator Spectre is displayed. In order to change the simulator left click **Setup->Simulator/Directory/Host**. The *Choosing Simulator* window opens. The simulator box displays *Spectre*. Left click on the box and select *eldoD*, then click *OK* (Fig. 17.). Check the change in the status bar of the *Design Environment* window.

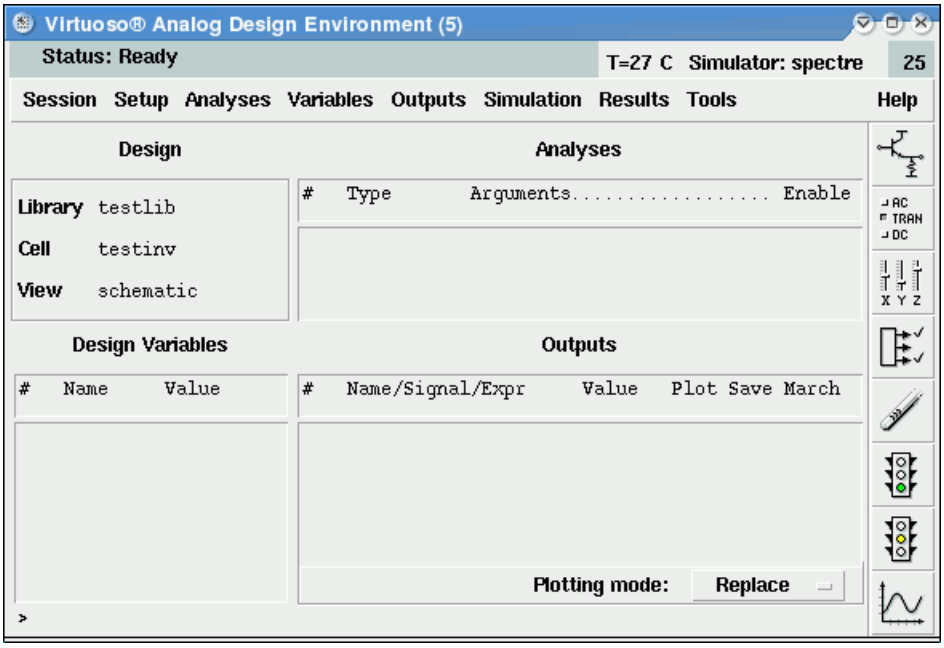


Fig. 16 Analog Environment -- main control panel

Left click **Analyses->Choose**. The *Choosing Analyses* dialog box comes up. Now specify the simulation. Select *Analysis*: **dc** (Fig. 18, next page), switch off *Print Operating Point*, select *Sweep Variable*: **Source** (1). Next you specify *Sweep Range*. Fill in *Start* **0** and *Stop* **5** and *By* **0.01**. Left click at the box **Select S1**. Switch over to the schematic and select the the sweeping source by clicking at the pulse generator *drv* at the input of the inverter. Then switch back to the *Choosing Analyses* dialog box. In the *1stSource* field */drv* should appear. See if the *Enabled* checkbox is switched on and click at **Apply**. In the same window you can specify the transient simulation as well. Select **tran** and set *From Time* **0**, *To Time* **25n**. Switch on the *Enabled* checkbox and click **OK**. In the *Analyses* field of the *Analog Environment* window the specified data appear.

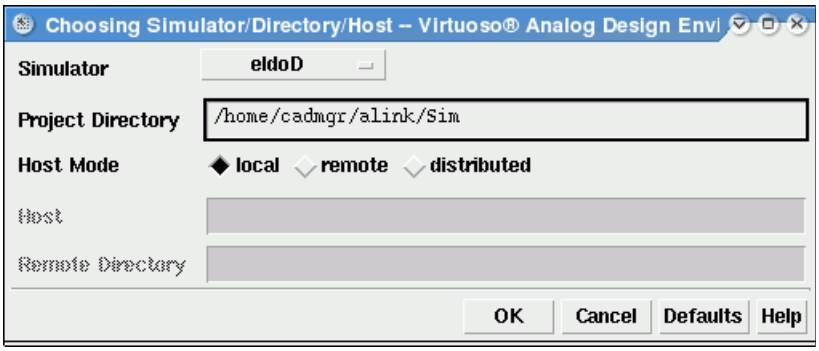


Fig. 17 Choosing the eldoD simulator

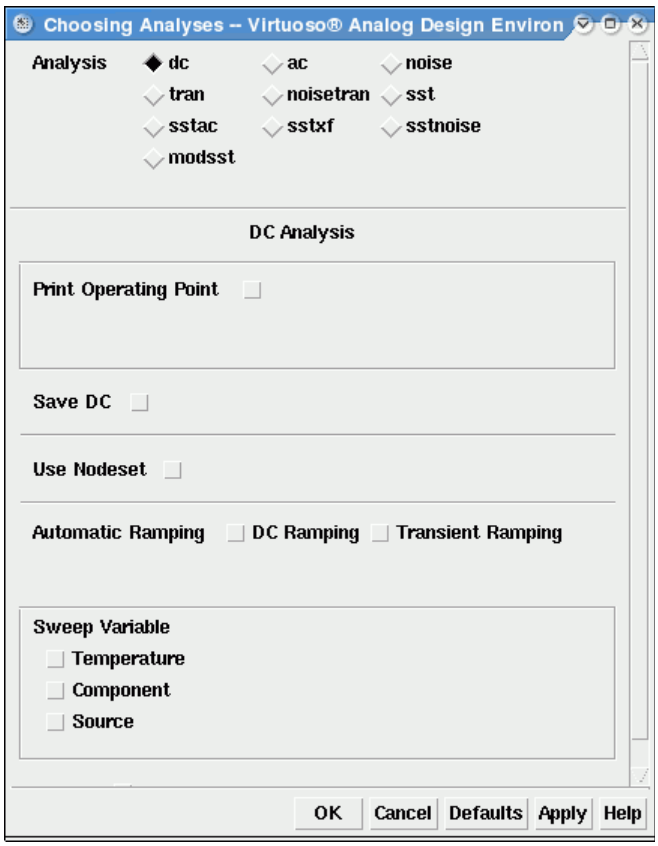


Fig. 18 Selecting simulation parameters

Next we want to select which results should be plotted. Left click on **Analog Design Environment:Outputs->To be Plotted->Select on Schematic**. Click on the wire between your pulse generator and the *in* pin of your inverter bearing the label *input*. Then click on the wire between the *out* pin of the inverter and the out pin of the test bench. Both wires should change color indicating that these voltages will be plotted.

Note: if you want to select a current to be plotted then click on the square of a symbol where the current is flowing through. There will be a circle around the square node indicating that a current is selected. Try it with the *vss* and *vdd* terminals of the inverter. A second click on the same square cancels the selection.

Now you are ready to run the simulation. Left click **Simulation->Run** or click on the green traffic light icon on the right side of the *Analog Design Environment* window. If a dialog box appears now requiring your decision whether some simulation results should be saved then your answer *must be Yes*!

The simulation runs and after a while the results will be plotted in two different ways. The first is the *Waveform Window* of Opus, you should ignore it by simply closing. The second one is the window of the display program *EZwave* which we are going to use. Extend this window to the whole screen and arrange the results neatly side by side.

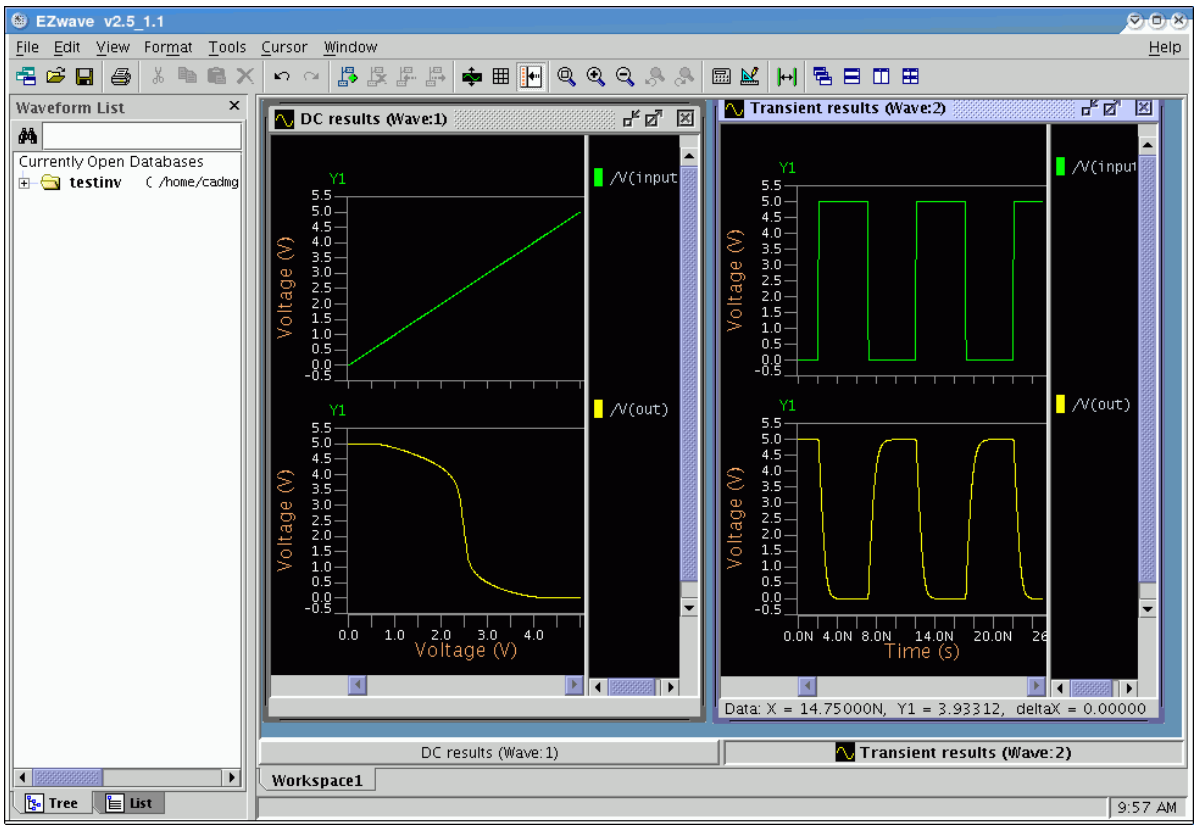


Fig. 19 Simulation results in the EZwave window

The results are plotted in the same coordinate-system. You can separate them by a right click inside the window and activate *Split* in the pop-up menu (Fig. 19). On the right side of the curve you see the colour-code and the name of the curve. To unite curves you can select the curve by left-click at its name and then left-drag the highlighted name to another one.

To finish the simulation exit *eldoD Spice*. Left click on **Analog Design Environment: Session->Quit**. Remember *NOT to save* the current state. If you choose to save then several hundreds of megabytes will be used in order to save your last simulation.

Note: the specified simulations in the *Analyses* field can be enabled and disabled one by one. To do so select the simulation by a left click and then left click **Analog Design Environment:Analyses-> Enable** or *Disable* (or even *Delete*).

Exercise using the *cursors* and the *slope* function

Read the section **Simulation: EZwave Window, Cursors and Slope** (page 35) and do the following exercise. Start the *DC Sweep* simulation of the inverter again, with the curves *out* and *uin*. When the *EZwave* window opens with the curves make it to form a large square on the screen. Now find three characteristic points of the output curve of the inverter:

1. The *inflexion point* where the slope of the curve, and so the *voltage gain* of the stage, is maximal
2. Determine the *noise immunity* of the inverter by finding those two points of the curve where the slope is just -1. The distance of these points from the end of the curve (0 or VDD, respectively) can be regarded as the *noise margin*. (One of them is shown in Fig. 20.)
3. Determine the transient times of the inverter: the propagation delay at 50% signal value and rise and fall time at the output between 10% and 90%. Arrange the transient signals in one coordinate system. So the propagation delay(s) can be determined. Apply one cursor to each curve and bring them to 2.5V. Read the time difference of the cursors at the bottom of the plot Then delete the input curve and apply both cursors to the output curve. Set them so that they are

at 0.5V and 4.5V, respectively. So you can read the rise and fall time values (Fig. 20.)

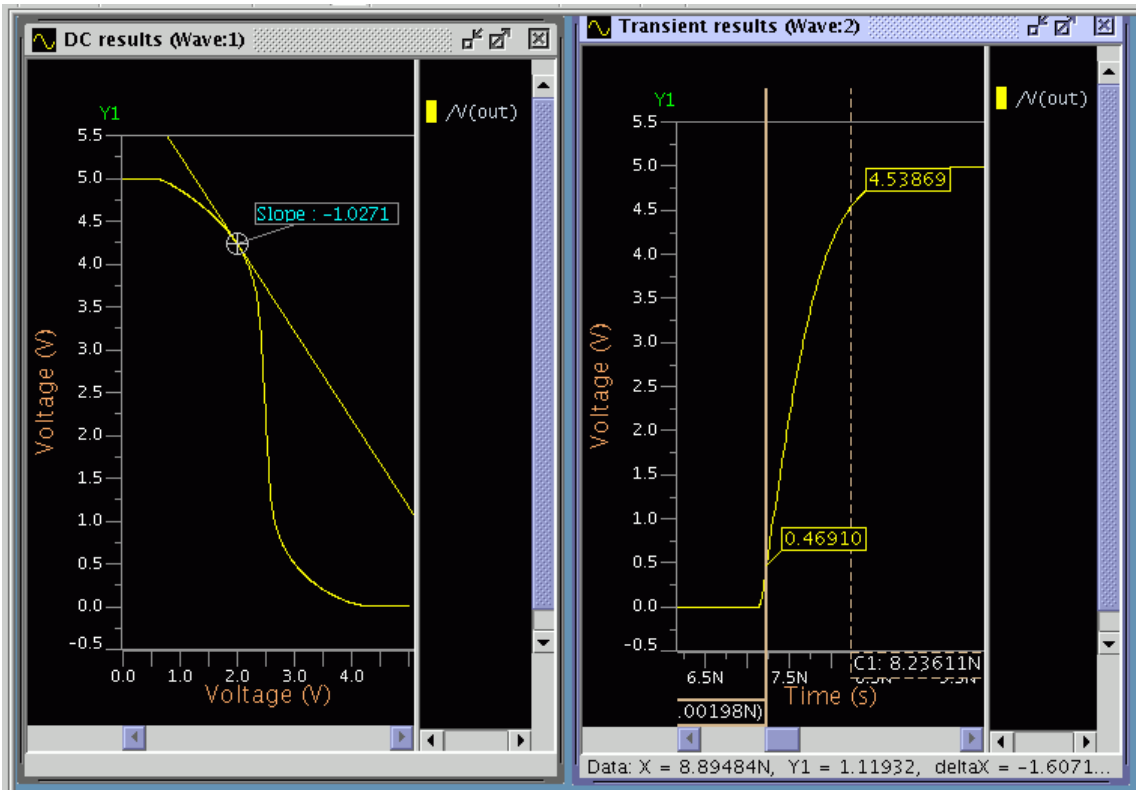


Fig. 20 Finding voltage gain and rise time