OPUS -- AMS 3.2

Version 4.4.3

Manual

for students designing VLSI integrated circuits at the VLSI laboratory of the DED (V2-324) using the OPUS Design Environment on Sun workstations under the UNIX Operating System.

dr. Peter Gärtner

22.10.2002

Ez a segédanyag megtalálható: \\FERMI\users\gaertner\Opus\opus.doc

Preface

This manual is primarily intended for students designing and testing VLSI integrated circuits or parts thereof at the VLSI laboratory of the DED (V2-324) using the OPUS Design Environment on Sun workstations under the UNIX Operating System.

For doing this work, first of all, you have to acquire from the system manager a personal user account in the Sun Network with UID and password.

This manual consists of three main parts:

- Primer for UNIX, for persons who have not yet worked with UNIX. It provides the minimum necessary knowledge to have some orientation in the operating system and to start OPUS.
- Primer for OPUS, to start the tool and learn the simplest steps for IC design, that is schematic entry, circuit simulation, layout design and check.
- Simplified "OPUS Users´ Manual". This part is introduced by some explanations on the structure of the different tools and on the basic IC design procedure which is supported by them. Then work with the tools is described one by one. The description can not be complete, let us refer here to the many hundred pages of the online help of the program which intends to do that job. However, it is hoped that what is described here can make design work efficient for students under leadership of inspectors.

Eventually it should be mentioned, too, what this manual does not comprise: theory and technology of silicon devices and integrated circuits. This kind of knowledge is prerequisite.

Experience with Windows on PCs is of advantage. In spite of running under UNIX the window system of OPUS shows much similarity with Windows.

Conventions Used

There will be several conventions used in this manual. The mouse on the Sun machines has three buttons. In the following there is some terminology explained which will be used in relation to mouse operations.

click left	press and release the left mouse button (quickly)
click middle	press and release the middle mouse button (quickly)
click right	press and release the right mouse button (quickly)
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.

<...>	press 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.

UNIX Primer

Basic UNIX Instructions

(Unix instructions have to be typed in a command ('shell') window. All instructions have to be terminated with <CR>!)

ls	list:	lists elements of a directory by their names
ls –l	list long:	detailed listing of a directory: access right, owner, length, date, name
ls –a	list all:	list including the hidden files too (beginning with '.')
ls –al	list all long:	detailed long listing of all files
ls –lt		long listing ordered by the time of generation
mkdir *dirname*		make directory named *dirname*
rmdir *dirname*		remove (delete) directory *dirname* (only if the directory is empty)
rm *filename*		remove (delete) the file *filename*
rm -r *dirname*		delete the directory *dirname* with all its contents (hierarchical! USE IT WITH CAUTION!!)
du disk usage		lists the complete hierarchy downwards with size (1 kByte blocks)
cd *subdir*		change directory to *subdir*
cd		change directory to the home directory of the user
textedit *filename*		opens the file filename for editing (new file if filename does not exist)

About UNIX

After logging in you are at the highest level of your user account. This is your *Home Directory*, which can be referred to by the tilde '~' character. UNIX comes up with an *xterm window* which is mainly for the messages of the operating system and does not have a scroll bar. Left click at the left button in the upper right corner so the window becomes an icon in the bottom bar. Then with a left click a menu pops up. Left click **Shells->Cmdtool**. A command shell will be opened with a vertical scroll bar. This window can be your workhorse as long as you are working direct with UNIX.

The directory where you are can be represented by the dot '**.**', the preceding higher level directory by two dots '**..**'.

Typing **ls -al** you will find among others the file *.tcshrc* which contains settings for the operating system. (If it does not yet exist you may open a new one with the editor.) The following three lines show examples for your own usage:

alias lth 'ls -lt | head' If you type **lth** then UNIX will produce a time-ordered list of the ten most recent files - an alias which can be favourably used for checking the recent changes in the directory.

alias ed 'textedit \!*&' Instead of *textedit xxx* you can simply type **ed xxx** and the editor will start with the file *xxx*. The ampersand '&' will make the editor start as a stand-alone process so that your window remains free for other work.

alias lock   'xlock -mode random' With this alias you can lock your screen. As a matter of fact, there is an entry for this very function in the main popup menu of the operating system but sometime it does not work.

Any change in *.tcshrc* will be effective only after your next logging-in.

HINT: If you copy *~gaertner/.tcshrc* to your home directory then you will have these and several other features in your account:

cp ~gaertner/.tcshrc . <CR>

When already copied, you can add other aliases for your personal usage, too.

OPUS Primer

The objective of this part is to teach a quick and easy start into the **Cadence** system without going into details. Going through this primer a simple inverter will be designed making the following steps:

schematic entry,
drawing a symbol,
simulating the circuit with Spice,
designing the silicon layout,
design rule check (DRC),
circuit extraction,
~~electrical rule check (ERC),~~ (not yet realized)
comparison layout versus schematic (LVS).

This primer alone can bring you to the above-mentioned results. It is recommended, however, to first read the introductory chapter of the manual, because the basic background knowledge acquired there will make it easier to follow and understand the procedures.

UNIX preparation

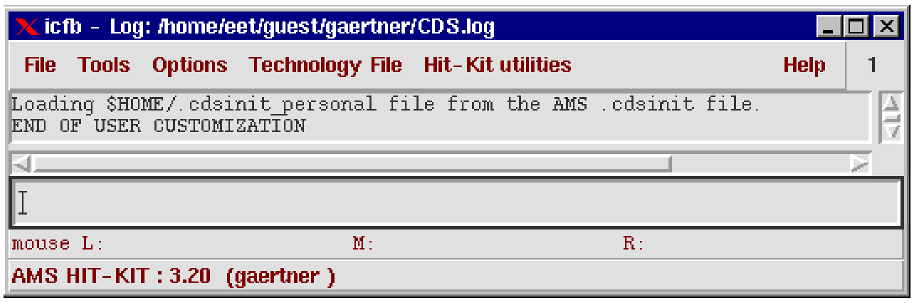
Open a subdirectory for the priming activities. Type **mkdir ams32**. *ams32* will be the home directory for OPUS in your Sun account. All OPUS activities will take place inside this directory.

Starting OPUS

Middle click at an empty place of the screen. The **Eng. Tools** popup window opens. Make a left click at **OPUS->AMS3.2**. A new UNIX shell comes up and asks for the design-directory of OPUS. Type **ams32 <CR>**, the name of the recently established subdirectory.

In return OPUS offers the available technologies. *AMS 0.8 um CMOS (cye)* 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 icon bar with a left click at the upper right corner.

OPUS goes on. The *Log* window appears with some logging messages, then it changes to the *icfb-Log* window which is called the *Command Interpreter Window (CIW)* because it can accept commands which you type in.

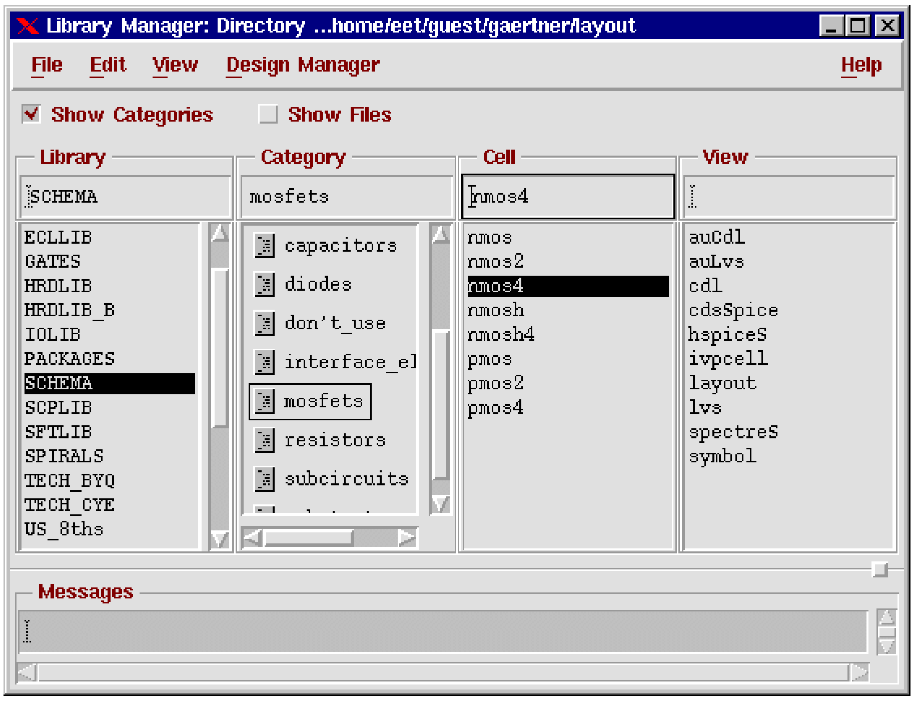

icfb - Log: /home/eet/guest/gaertner/CDS.log
File Tools Options Technology File Hit-Kit utilities Help 1

```
Loading $HOME/.cdsinit_personal file from the AMS .cdsinit file.
END OF USER CUSTOMIZATION
```

mouse L: M: R:
AMS HIT-KIT : 3.20 (gaertner)

1. Fig. Command Interpreter window

It is followed by the library manager window which starts automatically.



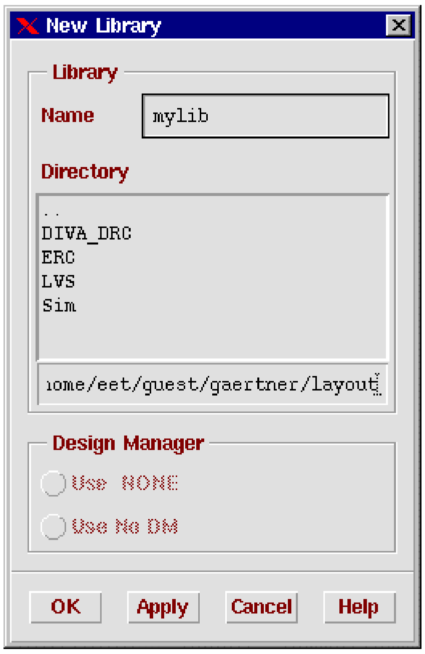
2. Fig. Library Manager window

The left column of the library manager window is a list of the current (accessible) libraries. Among these *SCHEMA* contains the transistors you will need for the inverter.

Left click at **SCHEMA**. The middle column shows the elements of *SCHEMA*. 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.

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.). 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 (*ams32*). 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_CYE** and then **OK**. Your library is created now and you should be able to locate the new library *mylib* in your Library Manager.


New Library

Library

Name: mylib

Directory

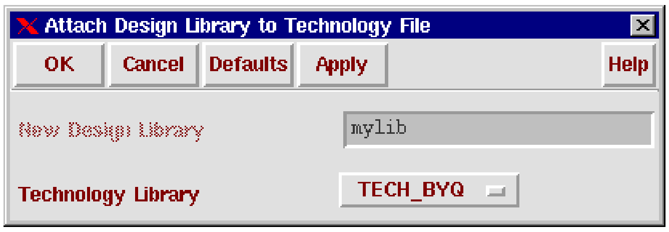
```
..
DIVA_DRC
ERC
LVS
Sim
```

ıome/eet/guest/gaertner/layout

Design Manager

Use NONE

Use No DM


Attach Design Library to Technology File

New Design Library: mylib

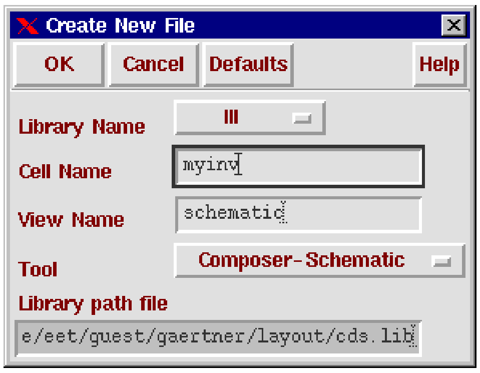
Technology Library: TECH_BYQ

3. Fig. New Library dialog box

4. Fig. 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.). 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. Left click the **OK** button. The *Composer Schematic Editing window* should show up (Fig. 6:).


Create New File

Library Name: lll

Cell Name: myinv

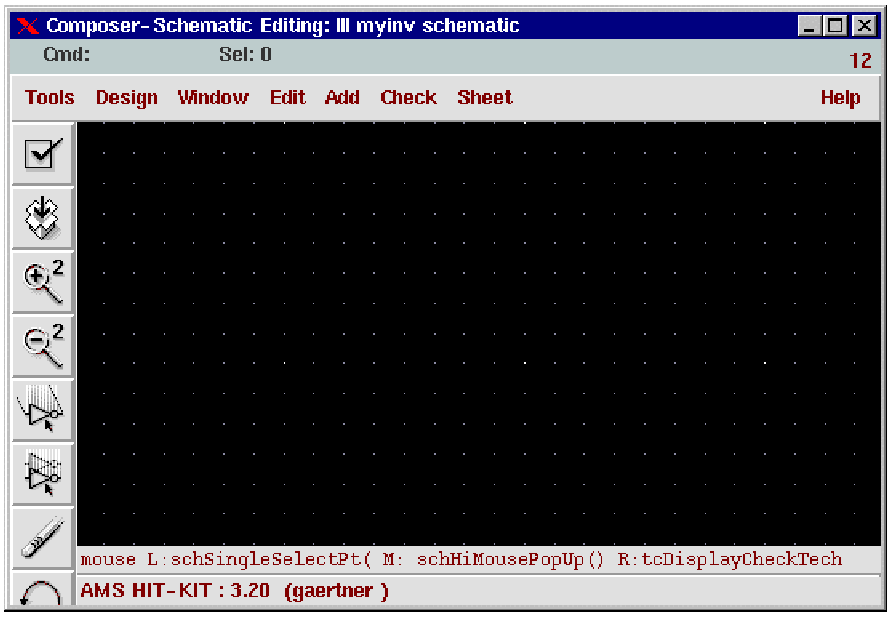
View Name: schematic

Tool: Composer-Schematic

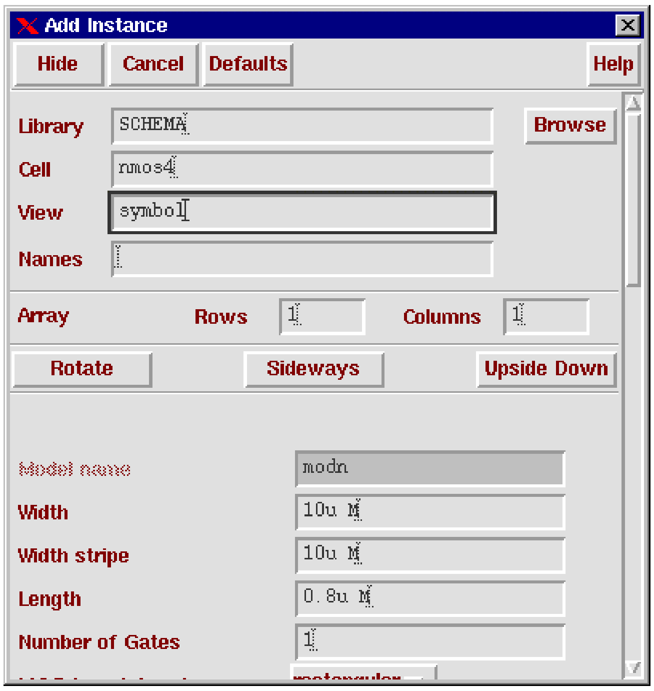
Library path file: e/eet/guest/gaertner/layout/cds.lib

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

Left click **Composer-Schematic Editing:Add->Instance**. The *Add Instance* dialog box appears (Fig. 7.). Type **SCHEMA** 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.


Composer-Schematic Editing: III myinv schematic
Cmd:
Sel: 0
12
Tools Design Window Edit Add Check Sheet Help
mouse L:schSingleSelectPt(M: schHiMousePopUp() R:tcDisplayCheckTech
AMS HIT-KIT : 3.20 (gaertner)

6. Fig. Schematic Editing window



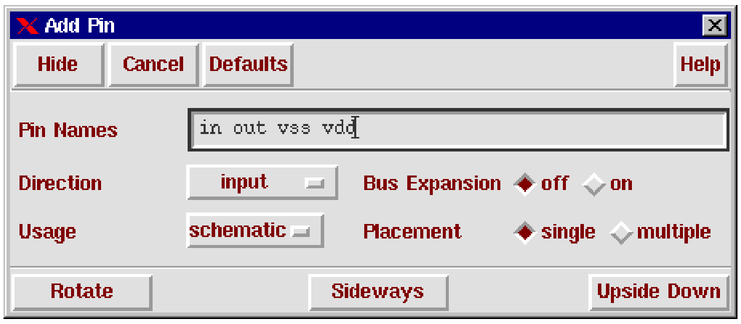
Library SCHEMA
Cell nmos4
View symbol
Names
Array Rows 1 Columns 1
Model name modn
Width 10u M
Width stripe 10u M
Length 0.8u M
Number of Gates 1

7. Fig. Add Instance – specifying a transistor

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 **4u** (four microns).

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 left click to place it. Having placed the first component, change the name of the transistor in the dialog box to **pmos4** and the width to **9u**. 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.

Next we will add the external pins for the inverter. Left click **Add->Pin**. The *Add Pin* dialog box appears (Fig. 8.). Type **in out vss vdd** in the *Pin Names* field for the four 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.)



Pin Names: in out vss vdd
Direction: input
Bus Expansion: [✓ off] on
Usage: schematic
Placement: [✓ single] multiple

8. Fig. Add Pin dialog box with pin specification

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, the next one is *out*. 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.

As it comes to the power supply pins, change the direction to *inputOutput* in the dialog box. The next pin is *vss* which should be placed under the NMOS transistor. In order to prepare the pin for a vertical connecting wire, left click the **Rotate** button in the dialog box and then place the pin in the schematic with a left click. The last pin, *vdd*, should be placed over the PMOS transistor. To prepare it for a downward connection click the **Rotate** button twice (2x90 degrees) and then place it. Left click **Add Pin:Cancel** or type <ESC> to return to the idle state of the editor.

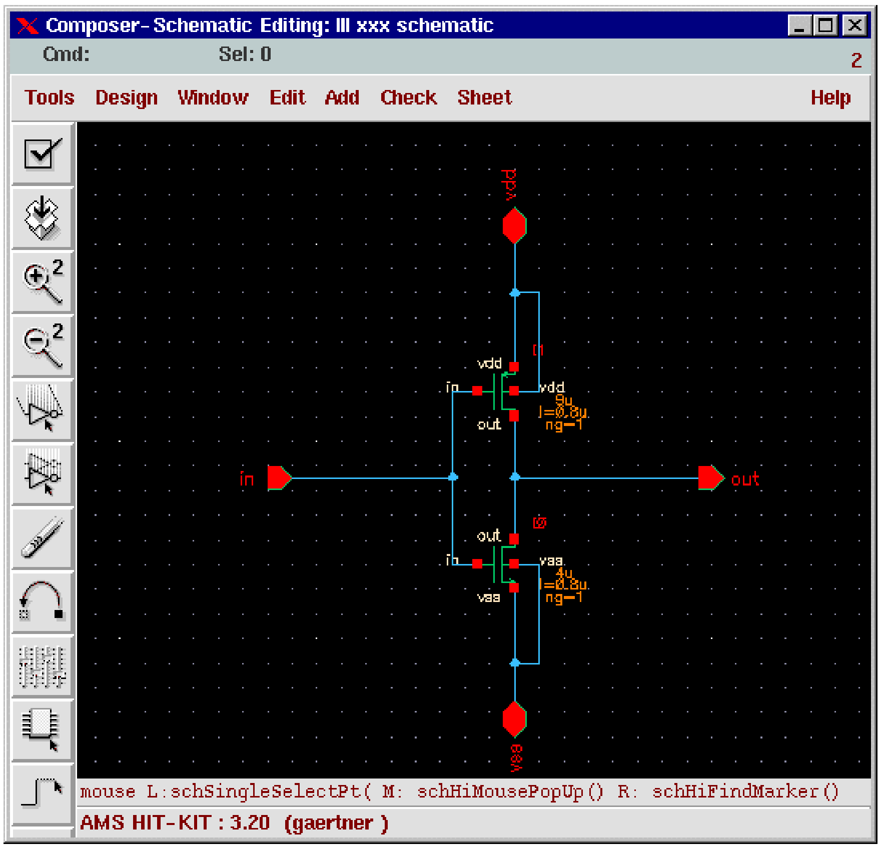
Now we'll add the wires to make things work. Click **Editing:Add->Wire**. The *Add Wire* form should appear. Just click its Hide button. Notice that as you get closer to one pin than another (including those on devices), a small diamond will show up inside of or around that pin. That is where you can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

To begin with, left click the diamond in the *vss* pin, then left click on the source terminal of the PMOS transitor. 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 *vss* pin. 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.

What remains is 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 the *vdd* pin. 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.).

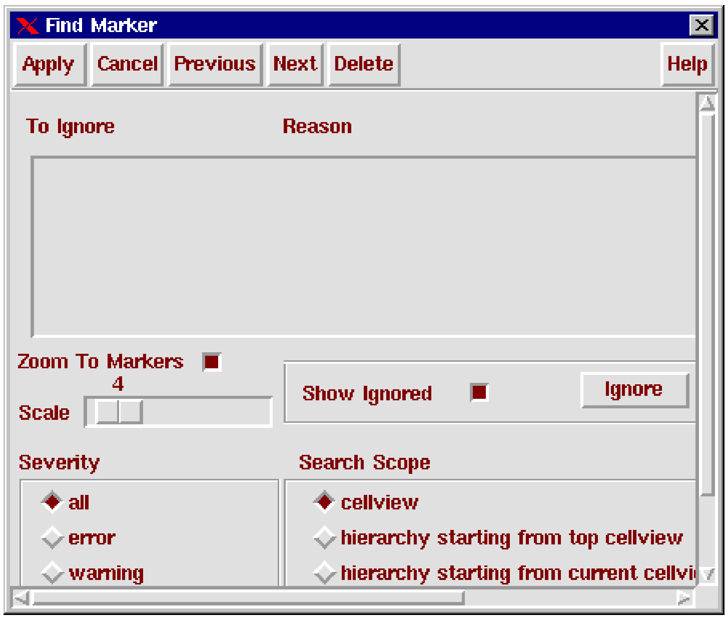
Once you have done editing, left click the **check mark** icon on the left side of the screen. This will check your work for connection errors and will save your cell (more exactly: its schematic view!) in the library. You can accomplish the same by left clicking **Editing: Design->Check and save**.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



mouse L:schSingleSelectPt(M: schHiMousePopUp() R: schHiFindMarker()
AMS HIT-KIT : 3.20 (gaertner)

9. Fig. Complete circuit diagram of the inverter

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.

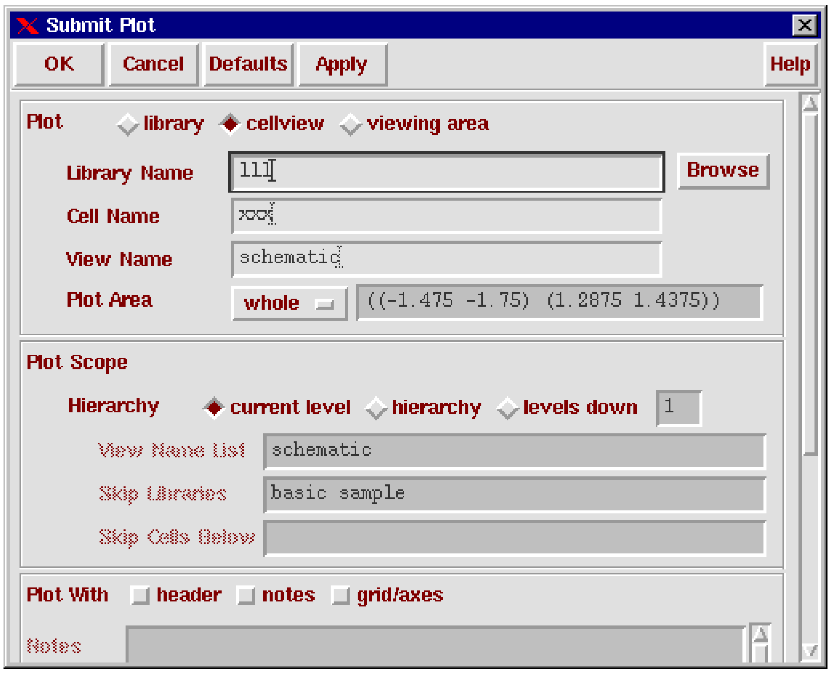


Zoom To Markers ■
Scale 4
Show Ignored ■
Severity: [✓ all] error warning
Search Scope: [✓ cellview] hierarchy starting from top cellview hierarchy starting from current cellvi

10. Fig. 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.). The default settings usually comprise your schematic and the plotter nearby, so a click on the **OK** button will do it. Ensure that the *Header* button is *NOT* selected. This option would only produce an extra page with general information on your plot like name and size etc..



Plot: ◇ library [✓ cellview] ◇ viewing area

Library Name: 111
Cell Name: xxx
View Name: schematic
Plot Area: whole ((-1.475 -1.75) (1.2875 1.4375))

Plot Scope

Hierarchy: [✓ current level] ◇ hierarchy ◇ levels down 1
View Name List: schematic
Skip Libraries: basic sample
Skip Cells Below:

Plot With: ☐ header ☐ notes ☐ grid/axes

Notes

11. Fig. Sending a circuit diagram to the plotter

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.

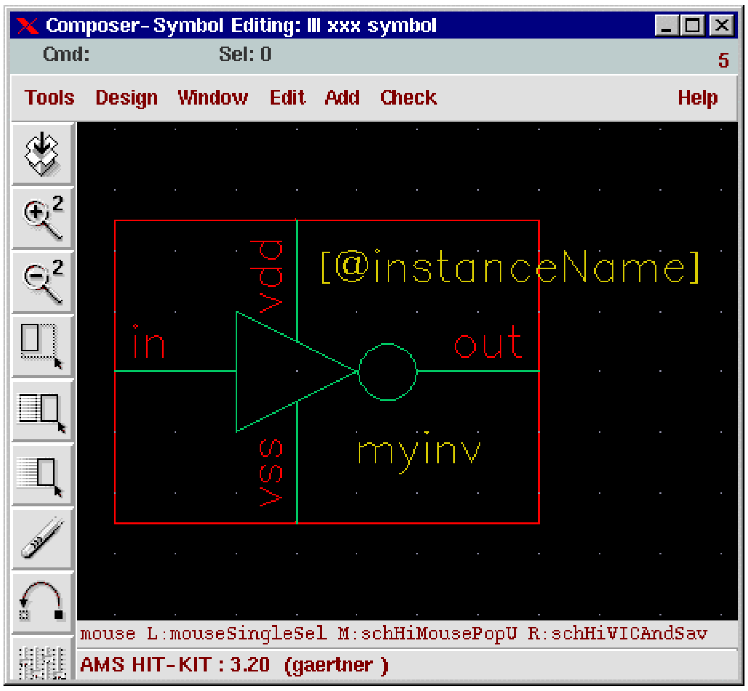
Create a symbol for the inverter

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 *Composer Symbol Editing* window should show up. (Fig. 12. shows it with the would-be result.) 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.

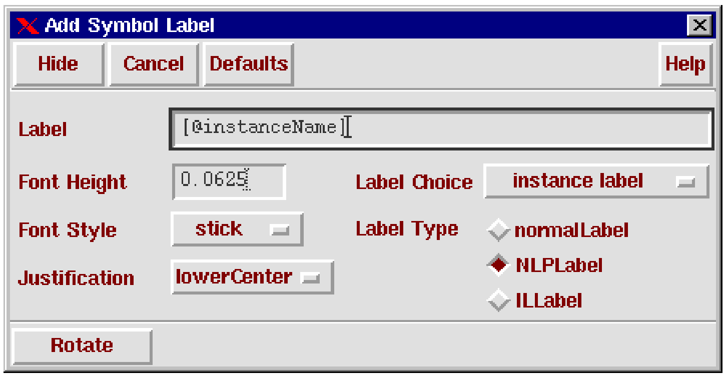


mouse L:mouseSingleSel M:schHiMousePopU R:schHiVICAndSav
AMS HIT-KIT : 3.20 (gaertner)

12. Fig. Symbol Editing window with the symbol

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.

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 vss vdd**. Do not forget to set the correct direction for each pin 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.



Label: [@instanceName]
Font Height: 0.0625
Font Style: stick
Justification: lowerCenter
Label Choice: instance label
Label Type: normalLabel [✓ NLPLabel] ILLabel

13. Fig. 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

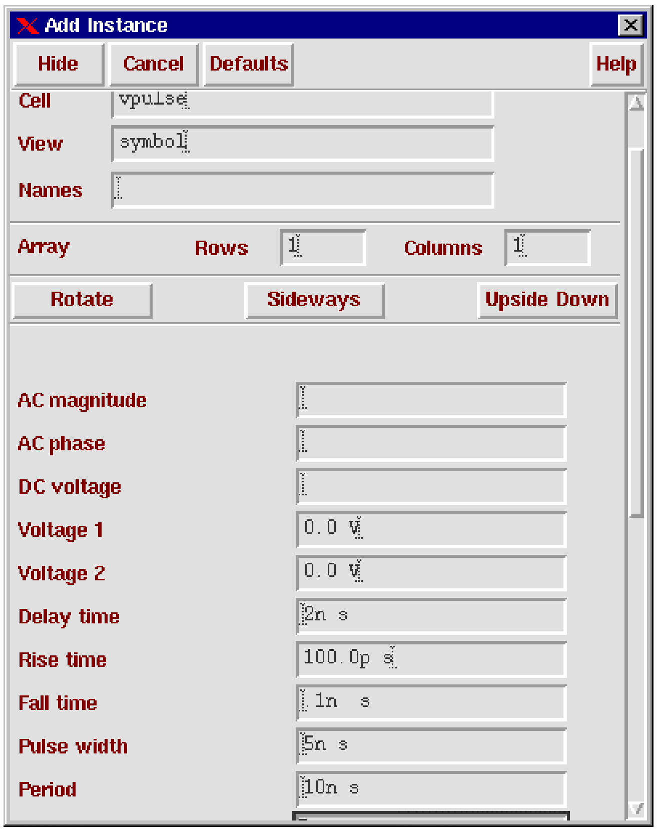
The functionality of an integrated circuit is verified by simulation. For a simulation the following things are needed:

- netlist
- power supply
- input signal source
- stimuli

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 6). Then open a new cell with schematic view, e.g. *test_inv*. The first element of the



Cell	vpulse
View	symbol
Names	
Array Rows	1
Columns	1
AC magnitude	
AC phase	
DC voltage	
Voltage 1	0.0 V
Voltage 2	0.0 V
Delay time	2n s
Rise time	100.0p s
Fall time	1n s
Pulse width	5n s
Period	10n s

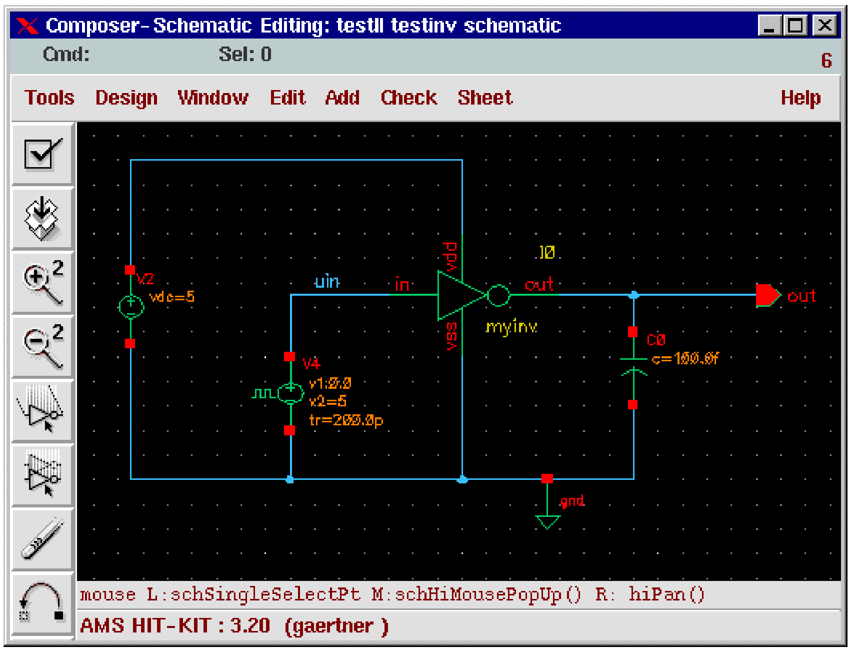
14. Fig. Specifying a pulse generator

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.

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 Volt. 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 *driv*, 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 four components. Next connect 5V to the *vdd* terminal of the inverter, *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.

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.).

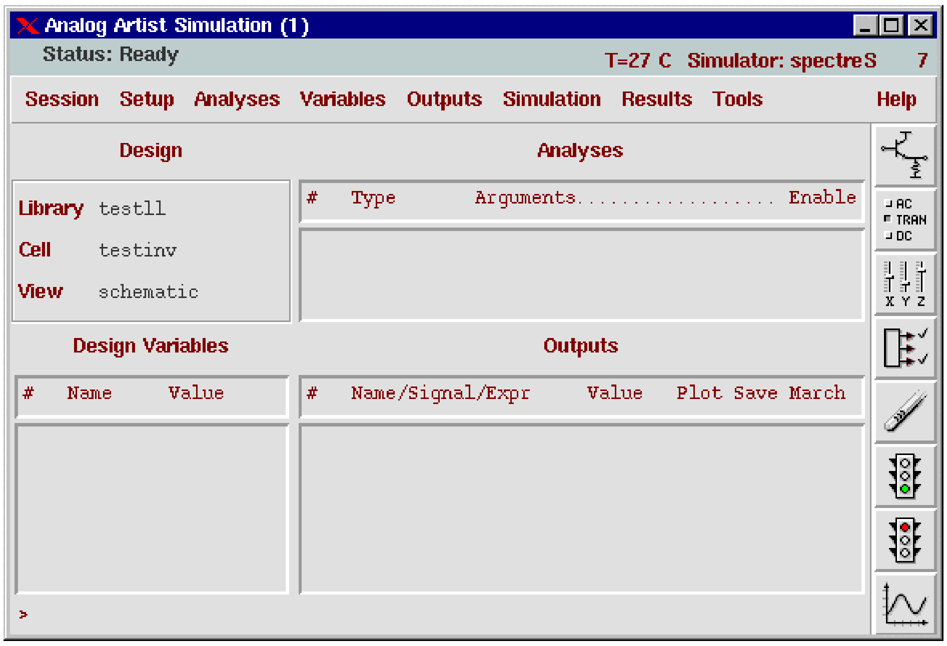


```
mouse L:schSingleSelectPt M:schHiMousePopUp() R: hiPan()
AMS HIT-KIT : 3.20  (gaertner )
```

15. Fig. Testbench – test environment for the inverter

Spectre (Spice) simulation

We are going to analyse the DC transfer characteristics and the transient behaviour of the inverter. Open the schematic *test_inv* in the library *testlib*. Left click **Tools->Analog Artist**. The *Analog Artist Simulation* window should appear(Fig. 16.).


Analog Artist Simulation (1)

Status: Ready   T=27 C Simulator: spectreS   7

Session Setup Analyses Variables Outputs Simulation Results Tools   Help

Design

Library	testll
Cell	testinv
View	schematic

Analyses

#	Type	Arguments.................	Enable

Design Variables

#	Name	Value

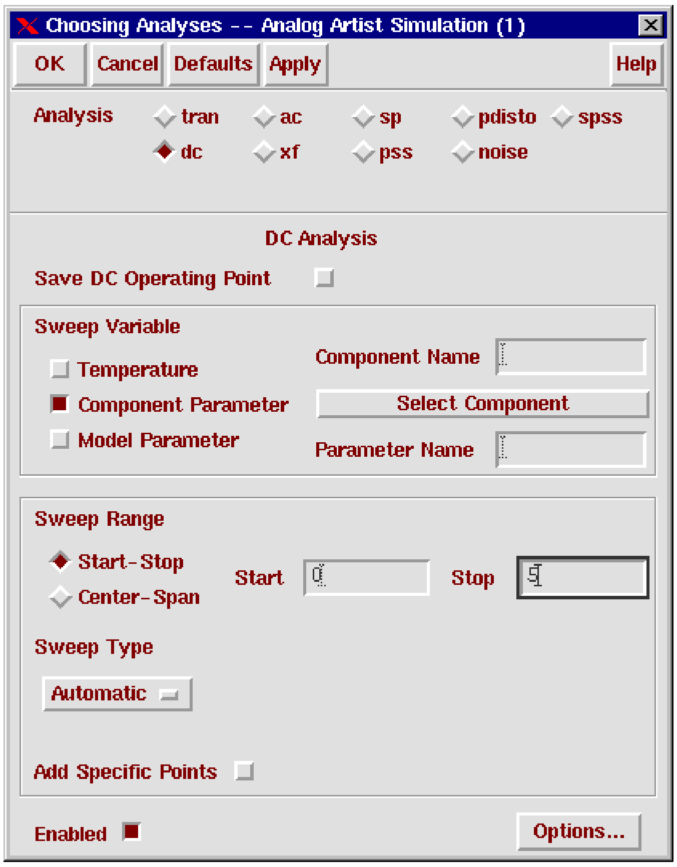
Outputs

#	Name/Signal/Expr	Value	Plot	Save	March

>

16. Fig. Analog Artist -- main control panel

Left click **Analyses->Choose**. The *Choosing Analyses* dialog box comes up (Fig. 17.).


Choosing Analyses -- Analog Artist Simulation (1)

OK Cancel Defaults Apply   Help

Analysis: tran ac sp pdisto spss [✓ dc] xf pss noise

DC Analysis

Save DC Operating Point ☐

Sweep Variable

☐ Temperature
■ Component Parameter
☐ Model Parameter

Component Name
Select Component
Parameter Name

Sweep Range

[✓ Start-Stop] Center-Span   Start 0   Stop 5

Sweep Type

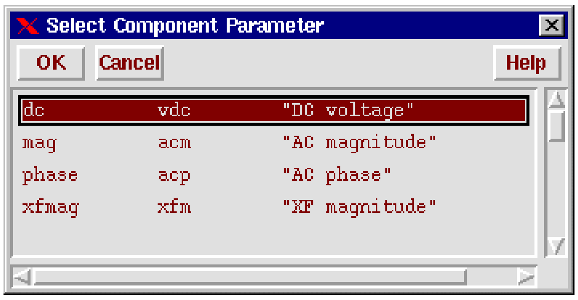
Automatic

Add Specific Points ☐

Enabled ■   Options...

17. Fig. Setting up sweeping DC simulation

Now specify the simulation. Select *Analysis*: **dc**, *Sweep Variable*: **Component Parameter**. Left double-click **Select Component**. The schematic comes into the foreground offering you the possibility to select the component by mouseclick. Select the pulse generator at the input of the inverter. The *Select Component Parameter* box appears (Fig. 18.). Here you have to choose the first option **DC Voltage** by a left click and then clicking at the **OK** button returns to the *Choosing Analyses* dialog box. The next item to specify is *Sweep Range*. Fill in *Start* **0** and *Stop* **5** and click on **OK**. In the *Analyses* field of the simulation window you can see the specified data.



```
dc          vdc          "DC voltage"
mag         acm          "AC magnitude"
phase       acp          "AC phase"
xfmag       xfm          "XF magnitude"
```

18. Fig. Selecting simulation parameters

The transient simulation can be specified similarly. Left click **Analyses->Choose** and select *tran*. The dialog box changes and asks for the *stop time* which you can set to **25n** then click on the **OK** button. The second analysis is specified.

Next we want to select which results should be plotted. Left click on **Analog Artist Simulation:Outputs->To be Plotted->Select on Schematic**. Click on the wire between your pulse generator and the *in* pin of your inverter. 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.

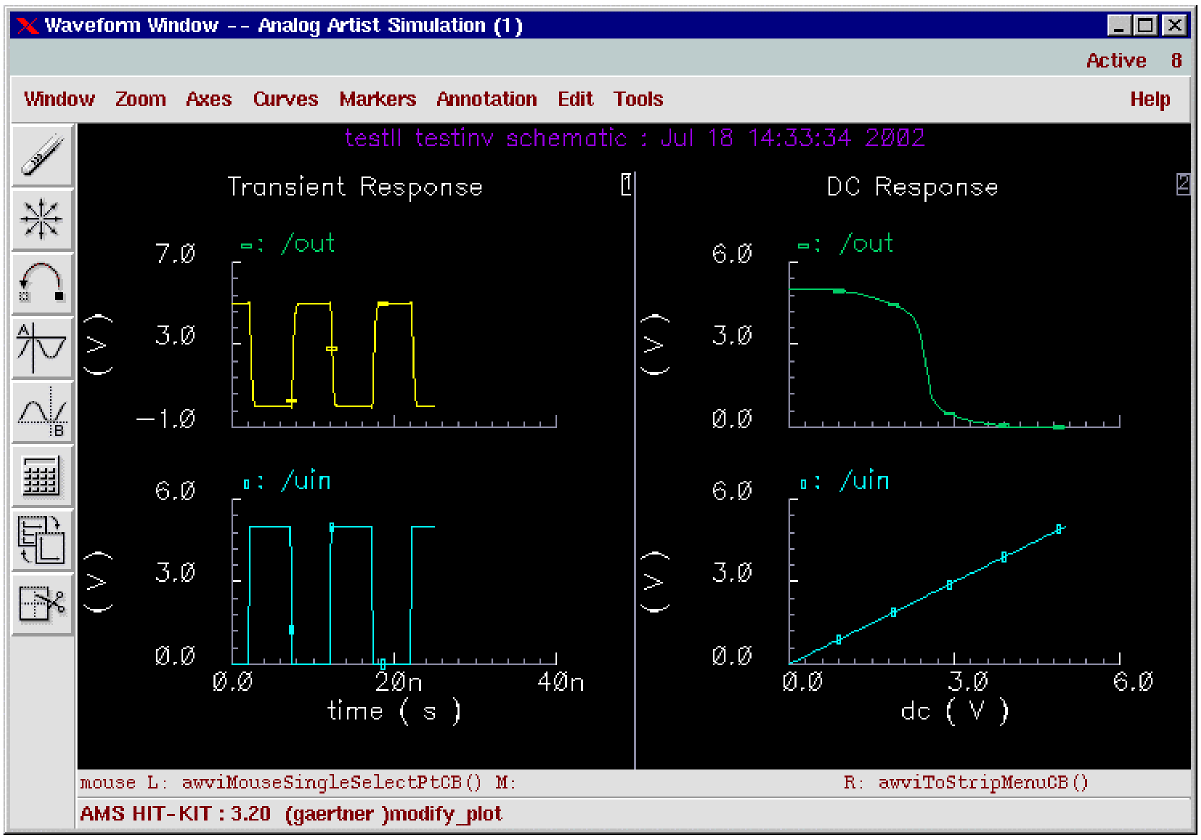
Now you are ready to run the simulation. Left click **Simulation->Run** or click on the green traffic light icon on the right side of the *Analog Artist* window. Again a dialog box appears requiring your decision if some simulation results should be saved. Your answer *must be Yes*!

The simulation runs and after a while the results will be plotted in the Waveform Window (Fig. 19, on the next page). Left click **Axes->To Strip** to separate the different curves. To finish it exit the *Cadence Spice*. Left click on **Analog Artist: 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 Artist:Analyses-> Enable** or *Disable* (or even *Delete*).

Create the layout of the inverter

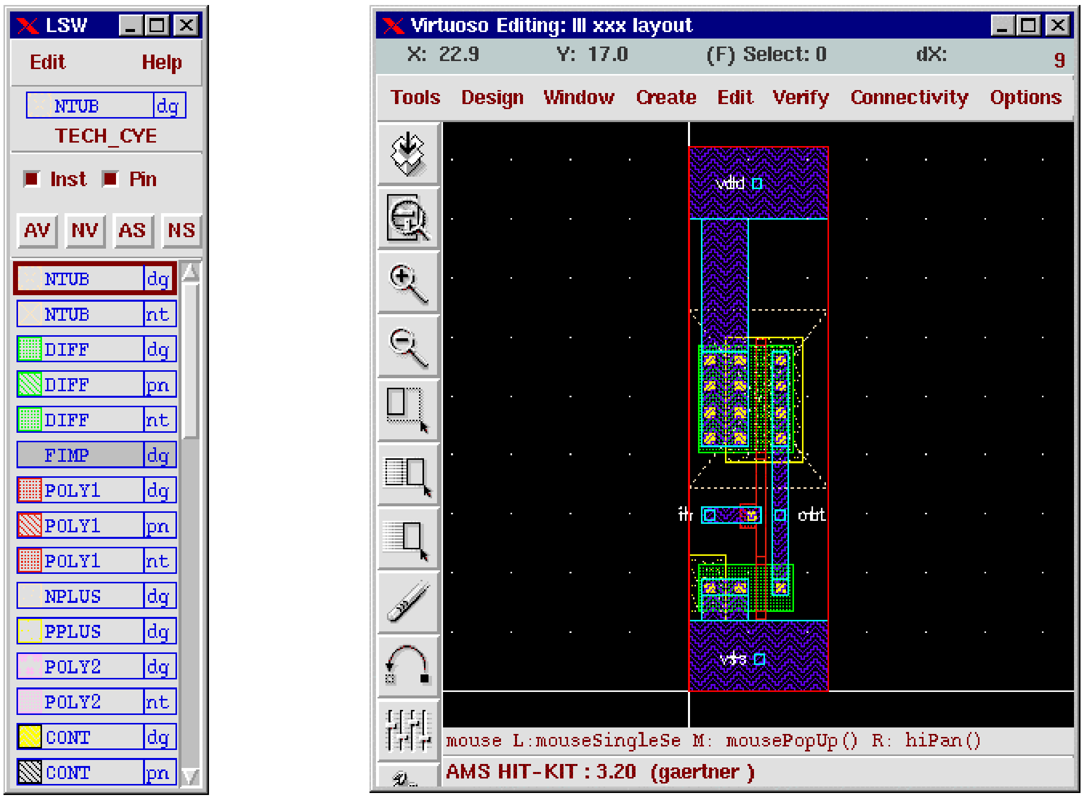
The tool for layout creation is called *VIRTUOSO*. Select in the *Library Manager mylib* and then *myinv*. In the *View* column the already existing views *Schematic* and *Symbol* appear. Left click **Library Manager:File->New->Cell View**. The *Create New File* dialog box shows up with *myinv* in the *Cell Name* field. Left click on Virtuoso in the popup menu of the tool selecting block. *layout* will be automatically filled in for *View Name*. Having clicked at the **OK** button two windows will appear, the *Virtuoso Editing* window and the *LSW* (*Layer Select*) window. Drag the LSW window to the left edge of the screen and adjust the layout editing window to the rest of the screen (Fig 20. Shows both windows, next page).



mouse L: awviMouseSingleSelectPtCB() M: R: awviToStripMenuCB()
AMS HIT-KIT : 3.20 (gaertner)modify_plot

19. Fig. Simulation results

The *LSW* window is the one you will use to choose the different layers for the IC design. It contains a list of the possible layers. Each entry is divided in three categories which are colour, abbreviated name and purpose. Colour shows the appearance of the layer in the layout. The abbreviation is the official name of the layer for Virtuoso, it can appear in messages, etc.



mouse L:mouseSingleSe M: mousePopUp() R: hiPan()
AMS HIT-KIT : 3.20 (gaertner)

20. Fig. Layout editing windows showing the target layout cell

The purpose can be *dg* for drawing and *pn* for pin, you will almost always need *dg*.

A layer can be activated for editing by a left click in the list. It gets a brown frame as highlighting and will be displayed at the uppermost part of the *LSW* window (current entry for shape processing commands). Underneath the current entry the Inst and Pin toggle buttons turn cell and pin instances selectable (on) and unselectable (off).

The four buttons in the next line control the visibility and selectability of all shape elements: AV = all visible NV = none visible AS = all selectable NS = none selectable When switching on and off, the elements in the list will be greyed. Single layers can be controlled by middle click.

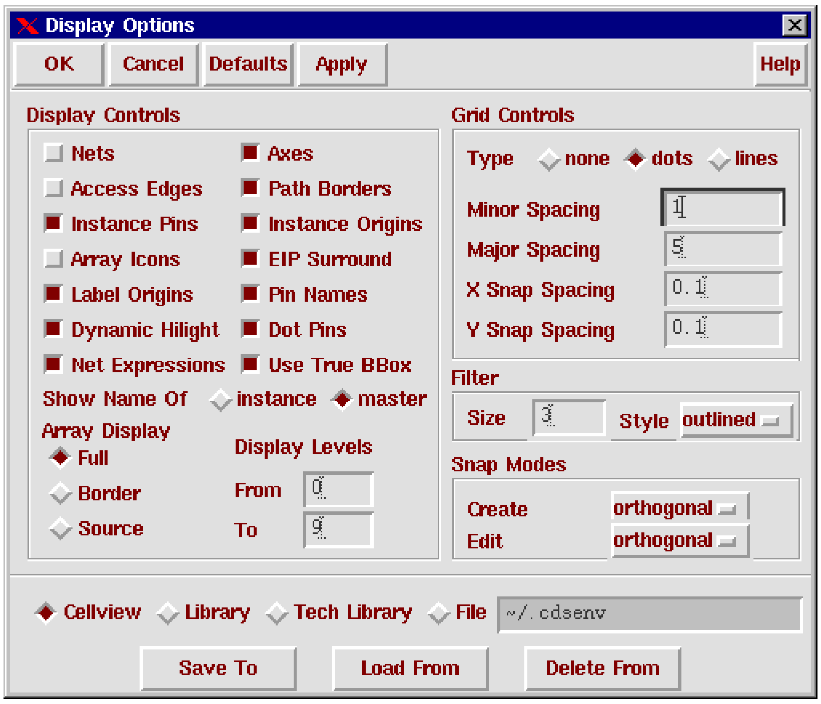
Some general information to the editing window:

The buttons on the left side speak for themselves. For *zoom-in* a rectangle can be drawn by right drag. You can return to the *previous view* by typing <w> (or left click **Window->Utilities-> Previous View**). Typing <f> will produce a *full picture*.

When the mouse is inside of the editing window then the X and Y coordinate values of the cursor are displayed between the title bar and the menu bar. Notice the dX and dY items, too. When creating shapes, these show distances relative to the last click.

If you make a mistake at any time you can left click on **Editing:Edit->Undo**. You can refresh the drawing by a left click on **Editing:Window->Redraw**.

Before starting prepare the battlefield for your work. Left click **Editing:Options->Display**. The *Display Options* dialog box opens containing usual default settings. In the *Display Controls* block switch on the buttons *Instance Pins* and *Pin Names*. For *Display levels* fill in *To* **9**. Underneath switch on the button *Cellview* and click on the button *Save To* and then *OK*. This means that these settings will be stored in the cell view file, so that they will be automatically set whenever the cell view will be opened.



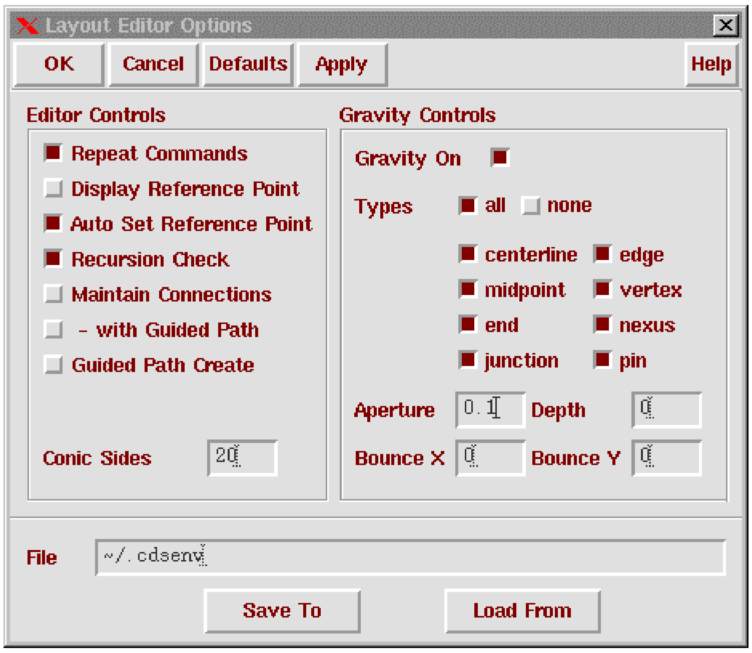
Display Controls
[ ] Nets [✓] Axes
[ ] Access Edges [✓] Path Borders
[✓] Instance Pins [✓] Instance Origins
[ ] Array Icons [✓] EIP Surround
[✓] Label Origins [✓] Pin Names
[✓] Dynamic Hilight [✓] Dot Pins
[✓] Net Expressions [✓] Use True BBox
Show Name Of instance [✓ master]
Array Display [✓ Full] Border Source
Display Levels From 0 To 9

Grid Controls
Type none [✓ dots] lines
Minor Spacing 1
Major Spacing 5
X Snap Spacing 0.1
Y Snap Spacing 0.1
Filter
Size 3 Style outlined
Snap Modes
Create orthogonal
Edit orthogonal

[✓ Cellview] Library Tech Library File ~/.cdsenv

21. Fig. Setting up display properties

One more setting: Left click **Editing:Options->Layout Editor**. The *Layout Editor Options* dialog box opens containing usual default settings (Fig. 22.). In the *Gravity Controls* block set *Aperture* to **0.1** and then left click **OK**.



Editor Controls

- ■ Repeat Commands
- □ Display Reference Point
- ■ Auto Set Reference Point
- ■ Recursion Check
- □ Maintain Connections
- □ - with Guided Path
- □ Guided Path Create

Conic Sides: 20

Gravity Controls

Gravity On ■

Types: ■ all □ none

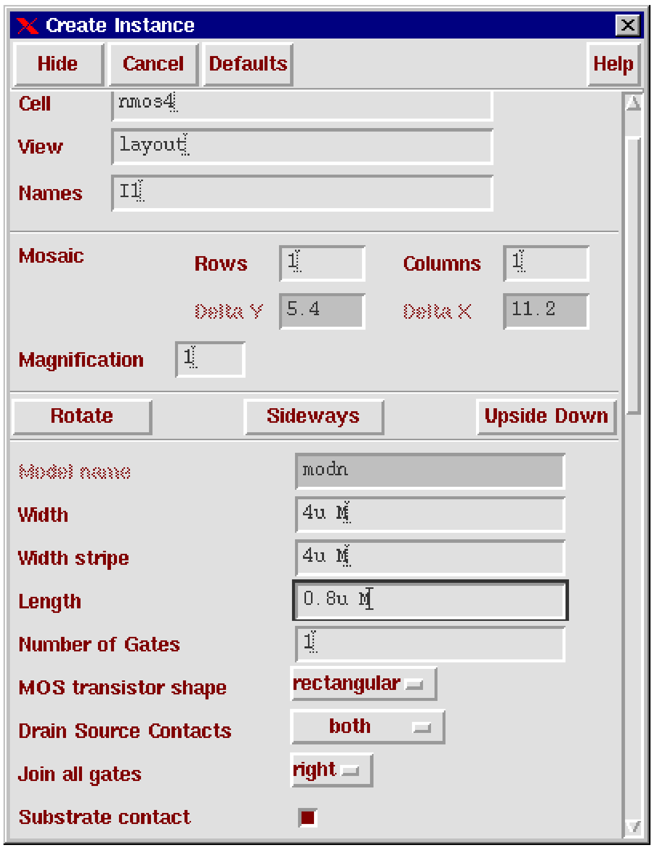
■ centerline	■ edge
■ midpoint	■ vertex
■ end	■ nexus
■ junction	■ pin

Aperture	0.1	Depth	0
Bounce X	0	Bounce Y	0

File: ~/.cdsenv

22. Fig. Layout Editor Options dialog box

Now it is time to start building the layout of your inverter *myinv*. The main elements are the transistors which are quite complex structures consisting of pieces of different layers. You might draw these pieces one by one, carefully complying with all design rules. However, you can make advantage of the library where "prefabricated" parametrisable transistor instances can be found.



Cell	nmos4
View	layout
Names	I1

Mosaic

Rows	1	Columns	1
Delta Y	5.4	Delta X	11.2

Magnification: 1

Model name	modn
Width	4u M
Width stripe	4u M
Length	0.8u M
Number of Gates	1
MOS transistor shape	rectangular
Drain Source Contacts	both
Join all gates	right
Substrate contact	■

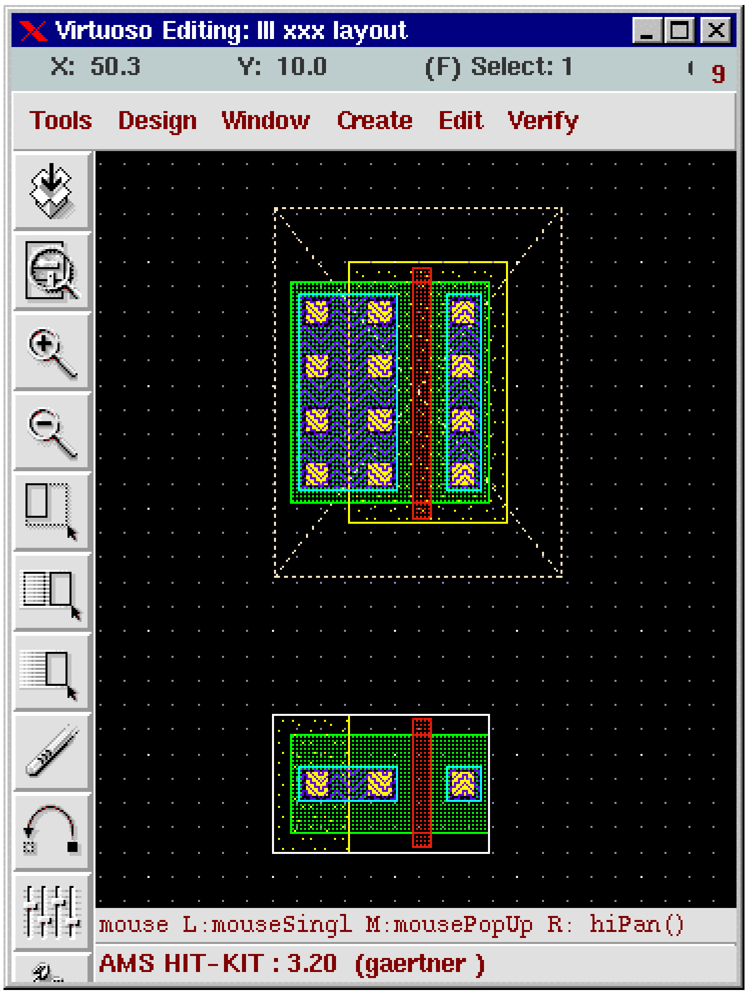
23. Fig. Create Instance dialog box with a transistor

They contain all details in compliance with the design rules, you only have to tailor them to your need (i.e. length and width).

Left click **Editing:Create->Instance**. The *Create Instance* dialog box opens (Fig 23.). If you exactly know then you may fill in *Library* and *Cell Name* of the cell you need. If not, click on the **Browse** button and find and select it in the library. Now we need (in accordance with the schematic) *nmos4* from the library *SCHEMA*. As soon as *Virtuoso* notices that a parametrizable instance has been specified, it extends the *Create Instance* dialog box for specifying the size of the transistor. Fill in *width*=**4u**, *length* is defaulted to *0.8u*. Activate the button **Substrate Contact**. Shifting the cursor to the editing window the contour of the transistor appears. You can recognise the gate as a narrow horizontal strip. With three right clicks you can turn it 3x90 degree so that the gate is vertical and the substrate contact is on the left side. Place the transistor in this position.

Repeat the procedure with a *pmos4* transistor. Set the width to **9u** and switch on the substrate contact. Again, with three right click bring it into a favourable position. It has a similar structure to the NMOS transistor but it is surrounded by an *N-well* (the layer is called *NTUB*). Place it above the NMOS transistor so that

- it exactly matches the horizontal position of the NMOS,
- it has a distance the from the NMOS nearly equal to the height of that (see Fig 24.).


Virtuoso Editing: III xxx layout
X: 50.3
Y: 10.0
(F) Select: 1
9
Tools
Design
Window
Create
Edit
Verify
mouse L:mouseSingl M:mousePopUp R: hiPan()
AMS HIT-KIT : 3.20 (gaertner)

24. Fig. Placement of the transistors of the inverter

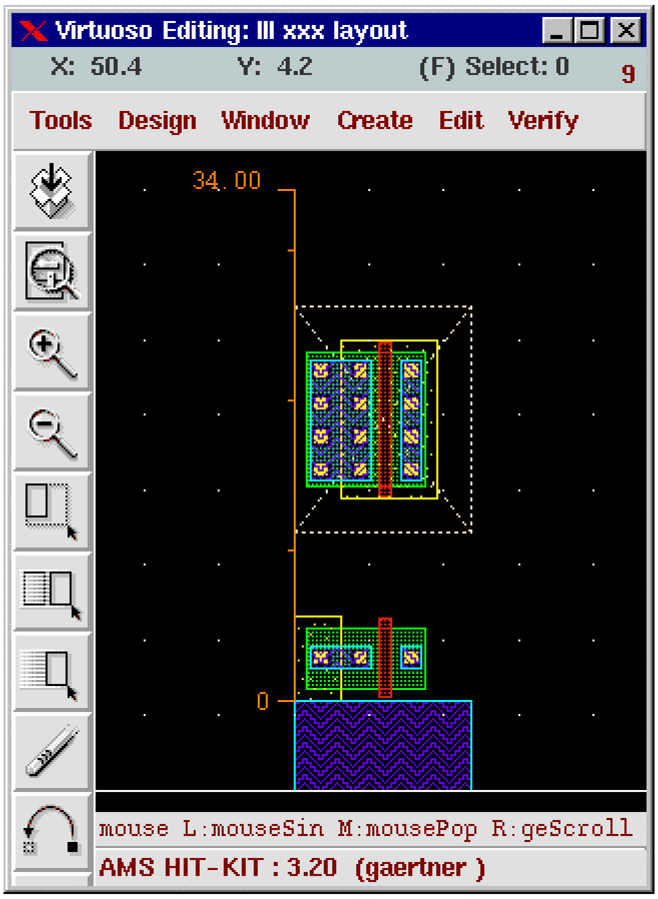
The transistors of the inverter have to be connected with each other: gate with gate and drain with drain. Zoom in so that these areas can well be seen and processed. In the *LSW* window left click on the layer **MET1**. Then left click **Editing:Create->Rectangle**. With two left clicks or one left drag make a *MET1* box between the drains with the same width as those. Now change the entry layer to *POLY1* and repeat the action for the gate terminals.

Next you have to provide power supply. If this inverter will be part of a standard cell design (and we suppose so) then you have to comply with the standard power supply structure

- medium distance 40 um
- width 6 um.

Set the entry layer to *MET1*. Left click **Editing:Create->Rectangle**. Make a left click at the lower left corner of the *PPLUS* layer of the substrate contact of the NMOS transistor. Shift the cursor to south-east until you can read *dX=-6* and *dY=11.8*. Complete the *MET1* box here with a left click. This will be the VSS rail.

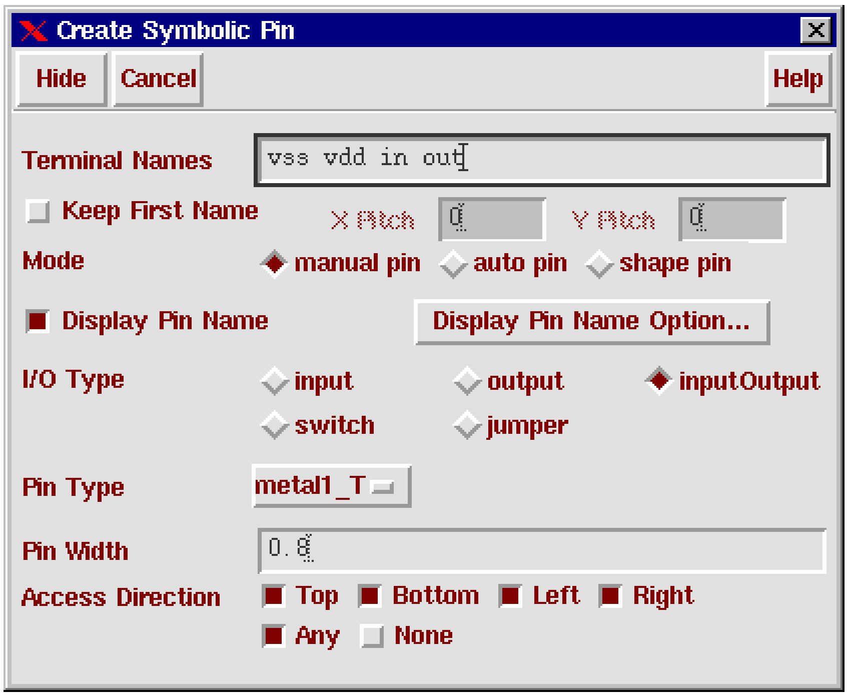
The VDD rail has to be positioned carefully. Left click **Editing: Create Ruler** (or simply type <k>). Make a left click at the upper left corner of the recently created VSS rail. Shifting the cursor upwards a vertical ruler starts. Bring it up to 34um and left click again. This way you have set the location of the lower left corner of the VDD rail (Fig. 25.). Now make another *MET1* box of the same size for VDD and type <K> - then the ruler disappears.


Virtuoso Editing: lll xxx layout
X: 50.4
Y: 4.2
(F) Select: 0
9
Tools
Design
Window
Create
Edit
Verify
34.00
0
mouse L:mouseSin M:mousePop R:geScroll
AMS HIT-KIT : 3.20 (gaertner)

25. Fig. Ruler in the layout editor window

The power rails being there, connect the source and bulk connect areas to them. Contact and a piece of metal 1 are already there, you only have to place two appropriate (4u wide) *MET1* boxes. Now an input terminal pin has to be prepared on *MET1*. Make a *POLY1* box, 1.3u wide, 2u high between the two transistors and joining the poly1 line connecting the gates. Switch to the layer *CONT* (contact) and place a 1u square box in the middle of the poly1. It should have a distance of 0.6u on the right side. Make again a *MET1* box: 1.4u high and 5u long, going left from the contact and having a space of 0.2u around. It should have a distance of 1.0u from the *MET1* bar connecting the drains.

The placement of the pins follows. Left click **Editing:Create->Pin**. The *Create Symbolic Pin* dialog box appears (Fig. 26.). Select *Pin Type* **metal1_T** and enter *Pin Width* **0.8**. Switch on the button *Display Pin Name*. Enter the names of the pins: **vss vdd in out**. *I/O Type* should be


Create Symbolic Pin

Terminal Names: vss vdd in out

Keep First Name: ☐

X Pitch: 0

Y Pitch: 0

Mode: [✓ manual pin] auto pin shape pin

Display Pin Name: ☑

I/O Type: input output [✓ inputOutput] switch jumper

Pin Type: metal1_T

Pin Width: 0.8

Access Direction: ☑ Top ☑ Bottom ☑ Left ☑ Right ☑ Any ☐ None

26. Fig. Dialog boksz for layout pins

defaulted to I/O. Bring the cursor to the middle of the VSS rail and make a left click. The pin will be placed but the pin name, too, has to be placed with another left click. Similarly place now *vdd*. Then change the *I/O Type* to *Input* in the *Create Symbolic Pin* dialog box and place the pin *in* and, eventually, change the *Pin Type* to *Output* and place the pin *out*.

The last layer to add is *CELBOX* (cell-box). Make it the entry layer and draw a rectangle around the cell so that everything is comprised. The origin (0,0) of the cell has to be adjusted to the lower left corner of the cell box. Left click **Editing:Edit->Other->Move Origin**. Now the coordinate axes move together with the cursor. Bring the origin to the lower left corner and place it with a left click.

Click on the **Save** button! You think the layout is completed. It is almost true.

Design Rule Check (DRC)

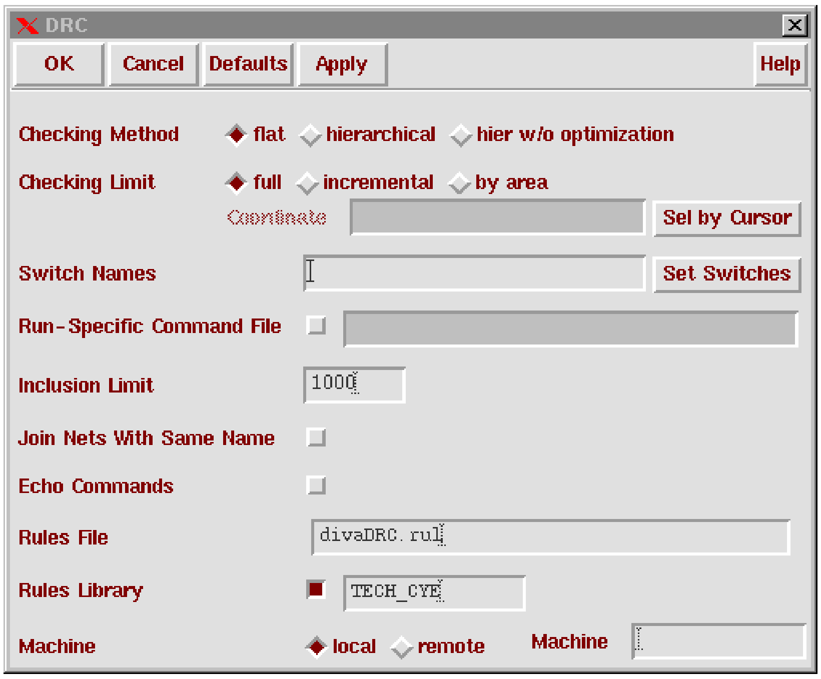
(DRC is the most important examination the layout is subjected to. Safe chip operation can only be guaranteed if there is no violation. As long as there is even only one single violation, the production of the chip must not be started.)

DRC can be started from the layout editor window. Left click **Editing:Verify->DRC**. The DRC dialog box opens (Fig. 27, next page). Check the following settings:

Checking Method:	*flat*
Checking Limit:	*full*
Switch Names:	*empty -- no entry!*
Rules File:	**divaDRC.rul**
Rules Library:	*TECH_CYE*
Machine:	*local*

By clicking on **Apply** or **OK** the DRC could be started now and you would get at least two violations:

- *NTUB* and *FIMP* not identical
- *NPLUS* and *PPLUS* not identical


DRC

OK Cancel Defaults Apply Help

Checking Method [✓ flat] hierarchical hier w/o optimization

Checking Limit [✓ full] incremental by area

Coordinate

Sel by Cursor

Switch Names

Set Switches

Run-Specific Command File

Inclusion Limit 1000

Join Nets With Same Name

Echo Commands

Rules File divaDRC.rul

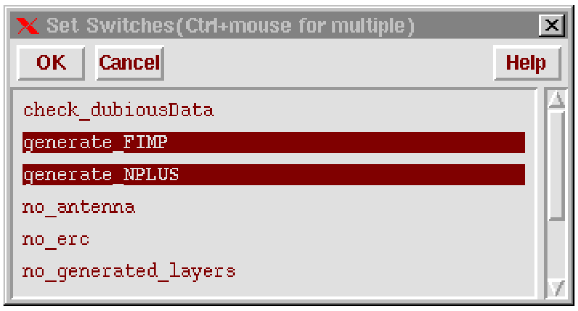
Rules Library [✓] TECH_CYE

Machine [✓ local] remote Machine

27. Fig. Window for starting DRC

Two layers are missing. *FIMP* or *field implantation* has to be there where there is n-well. *NPLUS* means *NO n+ diffusion*, it has to be there where there is *p+ diffusion* (*PPLUS*). Both new layers do not carry new information. They have a certain kind of administrative role. They have to be produced just before DRC and this can be done automatically using the DRC dialog box.

Left click **DRC:Set Switches**. The *Set Switches* box pops up Fig. 28). Keeping the control (CTRL) key depressed left click on **generate FIMP** and **generate NPLUS**. Then left click **OK**. The selected options appear in the field *Switch Names*. Now you can start DRC with **Apply** or **OK**. OPUS first will generate the missing layers and then do DRC. If it is free of errors then **save** it!


Set Switches(Ctrl+mouse for multiple)

OK Cancel Help

check_dubiousData

[✓ generate_FIMP]

[✓ generate_NPLUS]

no_antenna

no_erc

no_generated_layers

28. Fig. Switches for generating FIMP and NPLUS

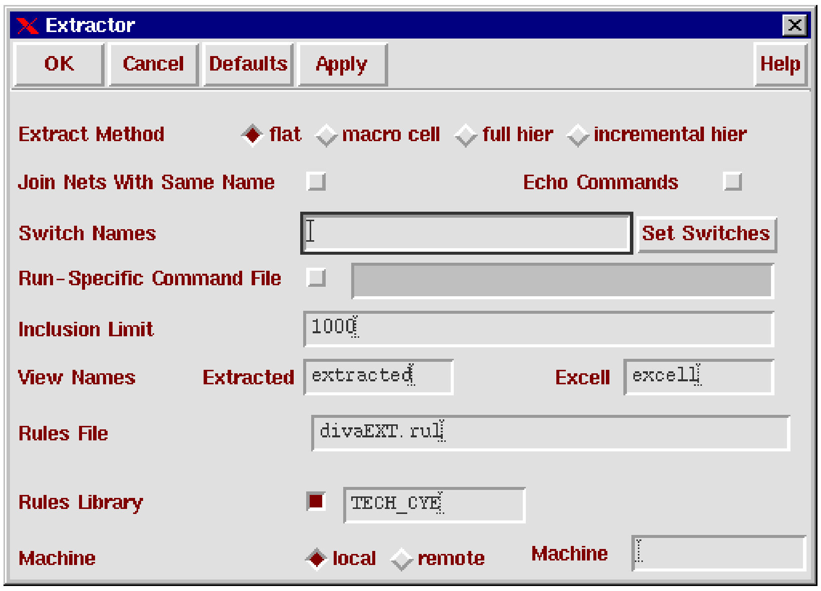
If DRC happens to find errors then they will be reported in the *Command Interpreter Window (CIW)* but you can analyse them one by one. Left click **Editing:Verify->Markers->Find**. The *Find Marker* dialog box appears. Switch on the *Zoom To Markers* button and click on **Next**. The editing window will zoom to the marker and the marker text message box explains the rule violation to you so that you can correct it. You have to iterate with error corrections until you reach zero error.

Layout extraction

DRC guarantees only the manufacturability of the structures on the chip. However, the functionality of the structures has to be checked, too. For this purpose the structures will be analysed and a netlist will be constructed - this is called *extraction*.

Left click **Editing:Verify->Extract**. The *Extractor* dialog box appears (Fig. 29). Check the default settings:

• Extract Method:	*flat*
• Switch Names:	*empty -- no entry!*
• Run-Specific Command File:	*off*
• Rules File:	**divaEXT.rul**
• Rules Library:	*TECH_CYE*
• Machine:	*local*



Extract Method	[✓ flat] macro cell full hier incremental hier
Join Nets With Same Name	☐
Echo Commands	☐
Switch Names	
Run-Specific Command File	☐
Inclusion Limit	1000
View Names Extracted	extracted
Excell	excell
Rules File	divaEXT.rul
Rules Library	■ TECH_CYE
Machine	[✓ local] remote
Machine	

29. Fig. Dialog box for starting the layout extraction

Start the extraction by a click on **Apply** or **OK**. In the *CIW* window you get a long report which should end with *Total errors found: 0*. If there are errors such as structures which cannot be interpreted as a correct device then these will be highlighted and you can do *find and explain markers* similarly to the case with DRC errors.

Afterwards you can check in the library manager that for the cell *myinv* the new view *extracted* appears.

Comparison Layout Versus Schematic (LVS)

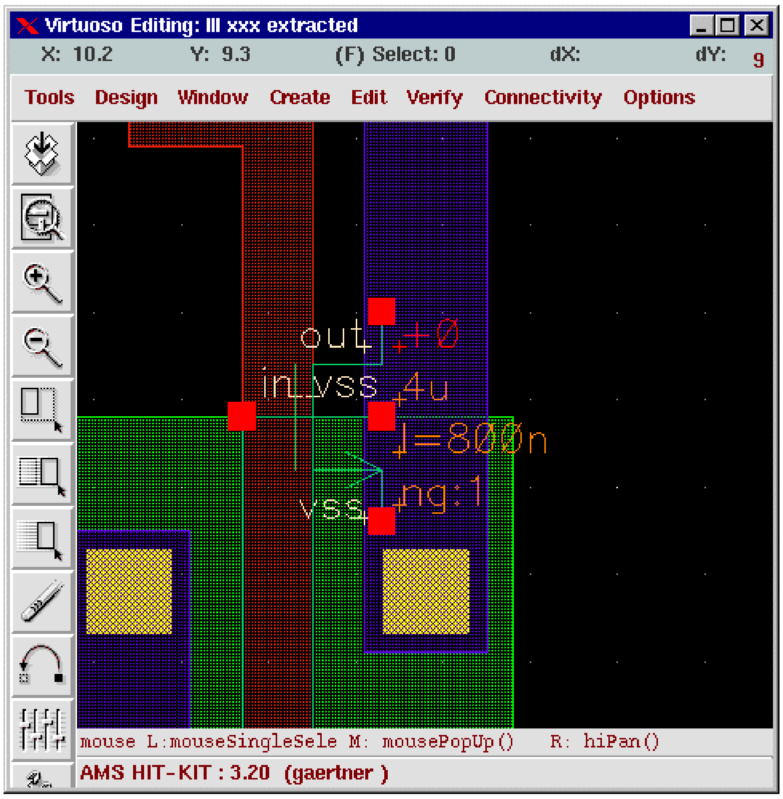
Open the extracted view of *myinv*. What you see is quite similar to the layout. If you zoom in you can find the schematic symbols of recognised and extracted circuit elements somewhere at the edge of the corresponding layout pieces (Fig. 30, on the next page). The question is if the realised circuit extracted from the layout is identical to the schematic which was the basis of its design.

Left click **Editing:Verify->LVS**. The *LVS* dialog box opens (Fig. 31, on the next page) with data for the extracted netlist already filled in. Either you fill in data for the schematic (Library, Cell, Schematic) or click on the browser button and select the schematic from the library by the mouse. Check other default data:

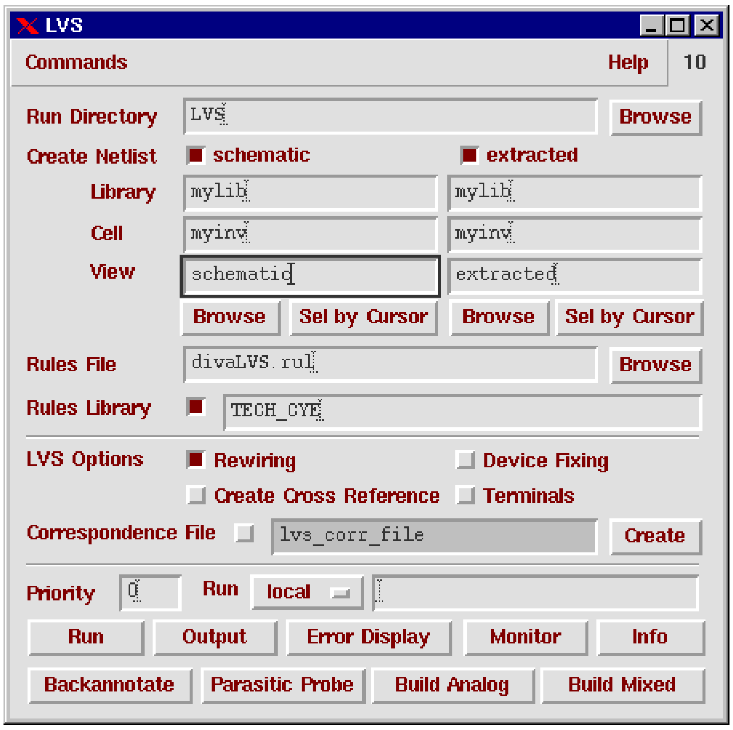
• Run Directory:	*LVS*
• Rules File:	*divaLVS.rul*

- Rules Library: *TECH_CYE*

Left click at the Run button. The comparison starts. If you have the performance meter (UNIX!) on the screen then you can observe the CPU activities, otherwise you don't see


Virtuoso Editing: lll xxx extracted
X: 10.2
Y: 9.3
(F) Select: 0
dX:
dY:
Tools
Design
Window
Create
Edit
Verify
Connectivity
Options
out
+0
in
vss
4u
l=800n
vss
ng:1
mouse L:mouseSingleSele M: mousePopUp() R: hiPan()
AMS HIT-KIT : 3.20 (gaertner)

30. Fig. Extracted transistor in the layout



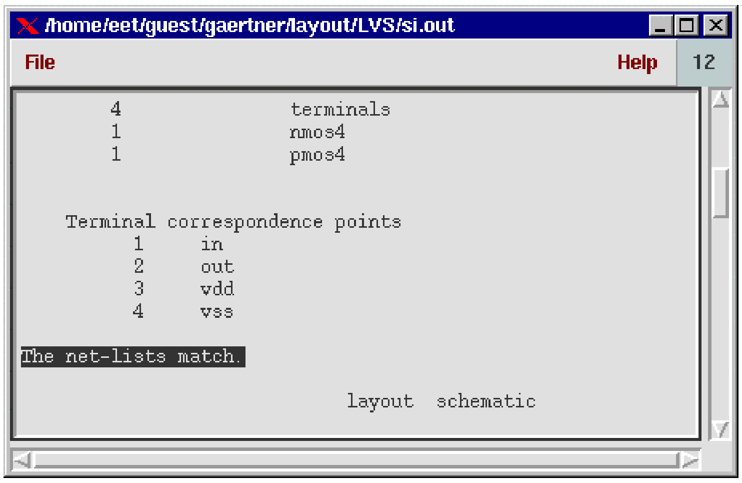
Run Directory	LVS	
Create Netlist	■ schematic	■ extracted
Library	mylib	mylib
Cell	myinv	myinv
View	schematic	extracted
Rules File	divaLVS.rul	
Rules Library	■ TECH_CYE	
LVS Options	■ Rewiring	☐ Device Fixing
	☐ Create Cross Reference	☐ Terminals
Correspondence File	☐ lvs_corr_file	
Priority	0	Run local

31. Fig. Window for starting LVS

anything for a while. Then the *Analysis Job Succeeded* message box appears and notifies you about success or failure of the LVS. Click on **OK** or **Cancel** and afterwards on **LVS:Output**.

A quite long report of the LVS job appears (Fig. 32, on the next page) with statistics and some details about possible faults. If you can find the line

The net-lists match



```
         4                terminals
         1                nmos4
         1                pmos4


    Terminal correspondence points
           1      in
           2      out
           3      vdd
           4      vss

The net-lists match.

                               layout  schematic
```

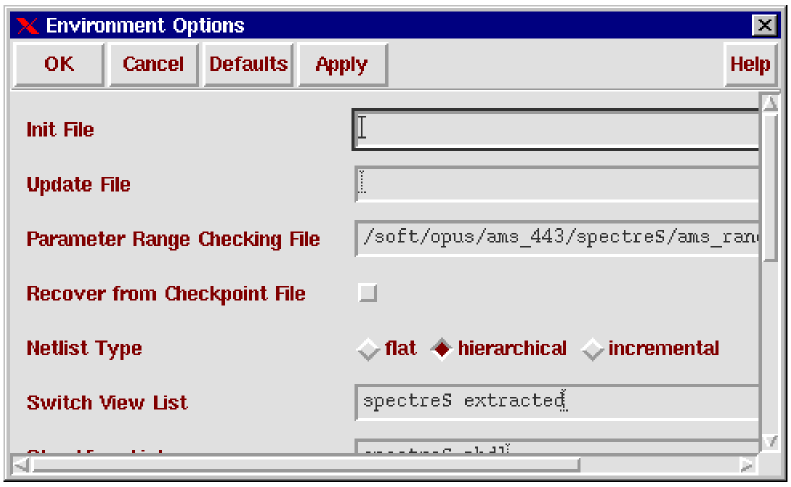
32. Fig. LVS report file

that means full success for you -- supposed that the schematic is correct.

Simulation of the extracted netlist

The last step is to check that not only the extracted netlist matches the schematic but also the simulation results are (nearly) identical. They should be but the stray capacitance of the wiring may influence the transient response.

After simulating the schematic we will change the content of the inverter symbol in the testbench. Left click on **Analog Artist Simulation: Setup->Environment**. The *Environment Options* dialog box opens (Fig. 33.). Change the block *Switch View List* to **spectreS extracted** and then click on **OK**.



Init File	
Update File	
Parameter Range Checking File	/soft/opus/ams_443/spectreS/ams_ran
Recover from Checkpoint File	☐
Netlist Type	flat [✓ hierarchical] incremental
Switch View List	spectreS extracted

33. Fig. Switching to layout simulation

Now you can start the simulation and check if it brings the expected results.

Simplified OPUS Users' Manual

Introduction

Cells and Views

In OPUS the construction unit is cell. A cell does not necessarily constitute a complete logical or circuit function. It is just a set of several (or even only one) circuit elements which are on whatever basis kept together. In different phases of circuit and layout design different kinds of data are required and processed. For instance geometrical (*picture*) information for a circuit diagram, electrical information for a netlist (*parts and their connections*), and another set of geometrical data for the layout of the very same cell. These data are collected and stored in the different *views* of the cell, such as:

Schematic:	circuit diagram (what it looks like).
HspiceS, Spectre:	contain Spice information for the element.
Layout:	silicon structure (geometrical).
Extracted:	netlist generated by analysing the structure of the layout.
Abstract:	contains an abstract representation of the layout for use by Cadence place & route software.
Behavioral:	the Verilog description of the cell.

For using cells as hierarchic elements they have the view:

Symbol:	"black box" representation of the cell for usage in higher levels of the circuit hierarchy.

Parametric Cells

Cells may have numeric or textual parameters. Filling in these parameters when instancing the cell may result in different shapes, size, etc. A typical case for parametric cells are the transistors used in OPUS Primer.

Cell Hierarchy

Cells are building blocks. They can be used as parts of other cells. It is always an instance of the cell, represented by a black-box symbol, with a unic instance name which is placed into the higher level cell. Neither the number of instances nor the depth of the hierarchy is limited.

File System

When starting, OPUS asks for a Home Directory which is not necessarily identical with your user's home directory. OPUS will do all its jobs in this directory, mostly in subdirectories of this one. There are only a few exceptions such as the logfiles *CDS.log* and *PIPO.LOG*. They are placed into the user's home directory. The most important subdirectory in OPUS' home directory is the *Working Library*.

A cell as such does not exist. It only exists in its views. Accordingly, in the working library, where all the cells with their data are stored, each cell is represented by a *subdirectory* having the name of the cell. The subdirectory contains the *views* of the cell which, again, are *subdirectories* containing several data files of the view concerned.

The great majority of the files are binary. They cannot be read or processed by text editor. Their existence may be checked on the Unix level but they **must not** be moved, copied, deleted or operated on by the operating system. All these tasks are organised and performed by the *Library Manager* of OPUS.

Design flow with OPUS:

OPUS is organised in accordance with the general design flow of integrated circuits which looks like this:

(idea)

(specification)

circuit entry

circuit verification

layout design

layout verification

(test design)

Idea and specification do not need computer aid although it is very important that a well founded specification is made before the circuit design begins. (The designer must know exactly what he/she wants to design!) Test design is only mentioned here for the sake of completeness but it is not dealt with in this manual. The design procedure is iterative, i.e. at each stage it may become necessary to return to a previous stage and make corrections.

The circuit design begins with the circuit entry (or capture). You can draw a circuit diagram on the screen using the

Schematic Editor (Composer)

The Schematic Editor is furnished with simple check functions, too. Complex circuits usually have a hierarchical architecture. To place a subcircuit at a higher circuit level a symbol of the subcircuit is needed. For this purpose OPUS offers the

Symbol Editor (Composer)

The schematics has to be verified whether it complies with the specification. This is done by simulation. For this purpose OPUS puts a tool to disposal which organises the simulation job:

Analog Artist

Analog Artist is only a framework, it enables a choice among several simulators which are attached to OPUS. These are: *spectre, spectre S, cds Spice, hspice S*, and the same simulators for *Verilog*. Usually *spectre S* is selected, it is the default choice.

The verified schematic has to be transformed into silicon. This is done by the layout editor

Virtuoso

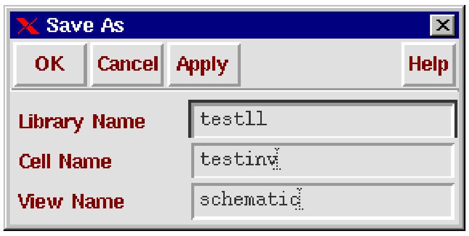
The layout editor is also furnished with devices for the verification of the created layout: design rules check (*DRC*), network extractor (*Extract*), electrical rules check (*ERC*) and comparison of layout with schematic (*LVS* - layout versus schematic).

LVS is necessary but not sufficient for layout verification. The circuit simulation, which was done for the verification of the schematic, has to be repeated with the extracted netlist since this contains the real capacitances of the circuit.

Data Sheets (documentation)

The cell design is only complete with data sheets containing all available data of the cell. Of course it is the specification which is the basis of the data sheets, for it was the basis of the design work, too. (Some data of the specification may have been modified in the course of the iterative design steps.) But there are other properties of the circuit which have not been specified. For instance: sensitivity of some parameters from supply voltage and/or temperature. Such data are only by-products of the design "as they are", but they, too, have to be determined by additional simulations and documented.

Windows of OPUS



Library Name testll
Cell Name testinv
View Name schematic

34. Fig. Typical OPUS window

The windows of OPUS have similarities in appearence and functionality to those in MS-Windows (Fig. 34.). They all have a title bar at the top having the usual 3 buttons for iconifying, making full screen and closing. Dialog boxes only have the closing button. The closing buttons (X) often do not react, in such a case some other button has to be used. Windows usually have a menu bar with drop-down menus. Functionality of some general purpose buttons:

OK	Do it and close the window
Apply	Do it and do not close the window
Cancel	Do not do anything, close the window
Close	The same as cancel
Hide	The settings are valid and the window is iconified
Help	The help system is called

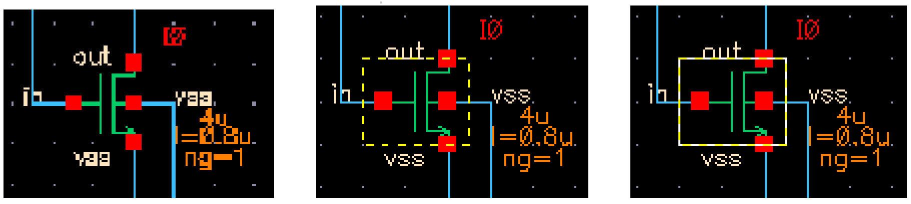
You may execute commands in different ways resulting in the same action. In case of a complete window like that of Fig. 6. You may use the roll-down menu at the top of the window, the buttons with icons on the left, or hot keys like the ones in OPUS Primer referres as <char>.

The ESCAPE function

If an editing function is activated then it remains active until another will be switched on. If, instead, on any reason, no active function is needed then type <ESC> and the system returns to an idle (waiting) state.

Selecting objects

1. Move the cursor to the edge of the object or place it over it. The contour of the object changes its colour and forms a dashed yellow line. (Sometime the dashed yellow line just appears because in "normal" state it is not displayed such as the selection box of an instance, see Fig. 35.) With a left click it turns to a solid white line that indicates the selected state. In case of uncertainty about what has been selected type <q>.



35. Fig. Selecting a transistor instance

A property sheet appears showing among others the name of the object, too.

2. Making a left drag a yellow rectangle can be drawn on the screen. All items being completely inside of the yellow rectangle will be selected by this method.

Property sheets

Cells consist of objects, be it a piece of wire, a rectangle, a transistor or an instance of another cell. They can be selected by mouseclick. When depressing <q> the *Edit Object Properties* window pops up (property sheet) showing the properties of the selected object and offering the possibility to change them. For instance the size of a rectangle, the location or the name of an instance can be changed. Changing the name of an instance means replacing it by another one. The change will be performed only after *OK* or *Apply* is clicked at. After *Apply* another change may be made but, more important, if in this situation another object will be selected, then the property sheet automatically brings up the properties of the newly selected object.

Searching

A favourable aid in editing layout as well as schematic is the function *Search* or *Search and Replace*. From the main menu it can be invoked by clicking at **Edit->Search**. A dialog box opens with many different buttons and edit fields for specifying the search (Fig.36). The object matching the criteria will be selected. Using wild characters such as * multiple search can be done. If an object is selected it can either be edited by hand or else some property can be changed by means of the Replace function.

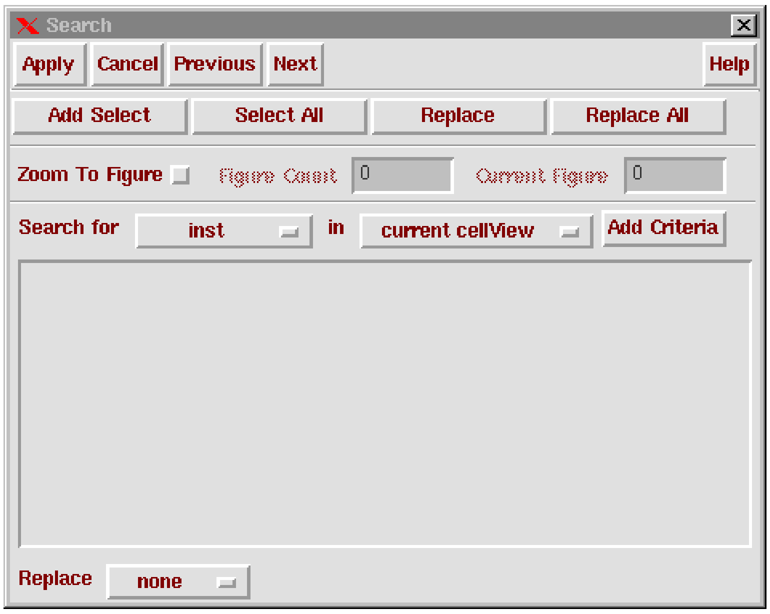
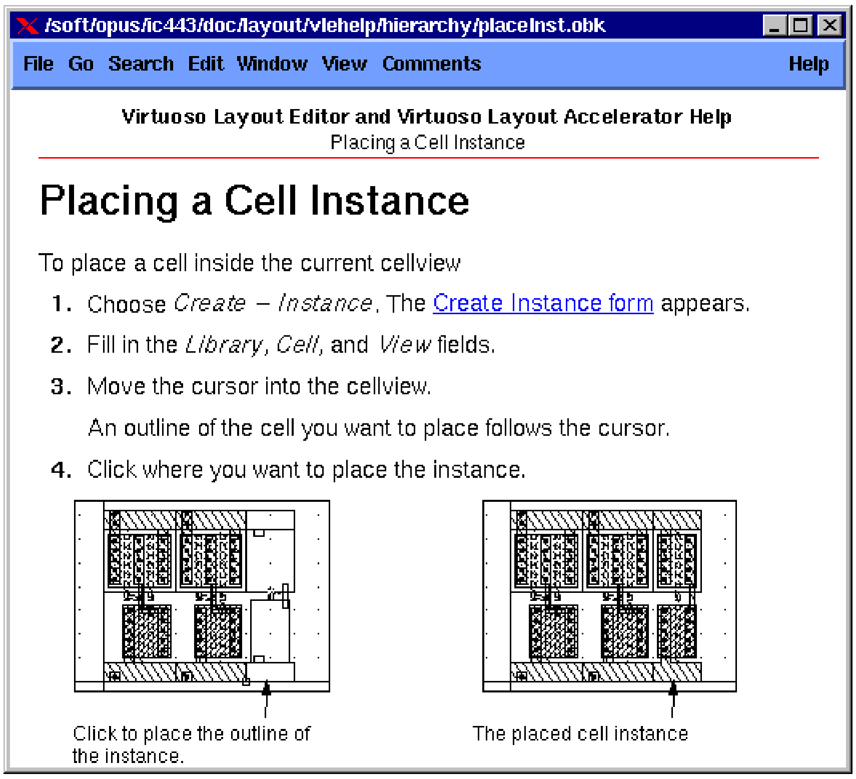


Figure Count 0
Current Figure 0
Search for inst in current cellView
Replace none

36. Fig. Search dialog box

HELP

OPUS has a quite extensive help system consisting of many books. Clicking at the Help button of a window invokes the appropriate book with the relevant information. However, to get the help some patience is required. After the click there is no sign of the internal activities of the computer and considerable time elapses till the help window comes up (Fig. 37.). If the first attempt does not bring the expected information, you can click at **Help:Goto->Contents** or **Goto->Index** and search. The information almost always exists but sometime finding it may be tiresome.


/soft/opus/ic443/doc/layout/vlehelp/hierarchy/placeInst.obk
File Go Search Edit Window View Comments Help

Virtuoso Layout Editor and Virtuoso Layout Accelerator Help
Placing a Cell Instance

Placing a Cell Instance

To place a cell inside the current cellview

1. Choose *Create – Instance*. The Create Instance form appears.
2. Fill in the *Library*, *Cell*, and *View* fields.
3. Move the cursor into the cellview.

   An outline of the cell you want to place follows the cursor.
4. Click where you want to place the instance.

37. Fig. Help window

GATE FOREST STYLE SEMICUSTOM LAYOUT

The *Gate Forest* semicustom *ASIC* (*Application Specific Integrated Circuit*) manufacturing system is based on the "sea of transistors" philosophy. It was developed at the Institute of Microelectronics Stuttgart (**IMS**) for ASICs of small series. The internal part (core) of the chip consists of a matrix of CMOS transistors forming alternate rows of p- and n-channel devices. A family of several masters from 2000 to 120000 pairs is prefabricated with connection pads around the core. So the transistors are there, for a given function the chip has to be personified by an individually designed metalisation on two levels which establishes appropriate connections between them.

Of course this semicustom system is a trade-off of cost and design work. The designer can choose neither the size of the chip nor the parameter of the transistors. On the other hand, to personify a chip only connections on two metal layers with four masks have to be designed and manufactured. This is considerable saving in time, effort and cost as well.

Such a system is especially advantageous for digital circuits although it can be extended for mixed-signal chips as well. For the elementary logic functions (gates, flipflops, etc.) a library of cells is developed consisting only of the metalisation for these functions. They can be freely placed on the core of the chip following the periodic structure of the transistor matrix.

Width of metal leads and their distances (routing pitch), size and distance of contact holes, etc. are quantised. It is recommended that the main grid pitch be equal to the routing pitch (i.e. the minimum pitch of the metal leads). The grid points in between should subdivide it so that (nearly) all minimal sizes of the layout elements lay on grid points, while the snap area is adjusted to the grid. With such settings the whole layout is "sitting on the grid" and by moving the cursor only the grid points can be selected. Such a "grid-oriented style" is correct per construction.