ZENI

Full Custom IC Design Flow Workshop

CEC Huada Electronic Design Co., Ltd.

EDA Division

http://www.zeni-eda.com

Table of Contents

1. About the Workshop

Welcome to Zeni Full Custom IC Design Flow Workshop. The document includes three parts. First part we have offered a simple circuit for brief tour. It contains some tools needed to investigate the Zeni. The simple circuit is in chapter 5 *A Brief Tour*. Through the tour you would have an overview of Zeni. Second part we have introduced most of menu commands and let you do some exercises special for each command by yourself, which will help you understand each command deeply. This part is in chapter 6 *Command Exercises*. At the end of the document, we have attached a guide of VCell template in chapter 7 *Appendix ---VCell Template*

2. About Zeni

Zeni - a high performance EDA tool provides a full-solution from front-end to back-end of full custom IC design. It integrates Schematic Editor, Layout Editor, Layout Verification, Parasitic Extractor and Signal Integrity analyzer in one common design platform, and the design data could be exchanged by the other EDA tools smoothly.

Zeni can be used easily for its friendly interface and high performance. Several tools such as Schematic Editor, Layout Editor, Layout Verification and Parasitic Extractor have outstanding advantages over other EDA tools. With Fast and effective localized technical support and service, Zeni has gained trust and high praise from our customers. It is being used widely by domestic and international IC design companies.

Please go to www.zeni-eda.com to get more information about Zeni.

3. Starting Zeni

3.1. Set up environment of Zeni

Step 1: Please make sure that the license sever of zeni is running.

Step 2: Before starting zeni, you should run an initialization file to set environment variables.
If you use CShell, please type command:
% source <ZENI_INSTALL_PATH>/setup.csh

If you use Bash, please type command:
% source <ZENI_INSTALL_PATH>/setup.bash

Here, "ZENI_INSTALL_PATH" is Zeni's install path.

Step 3: (Omit this step if you only have one account) Set your HOME with following alternative command.

% setenv ZENI_USER_HOME xxx (c shell)
% export ZENI_USER_HOME=xxx (bash)

Here, “xxx” is a path. An absolute path is recommended.

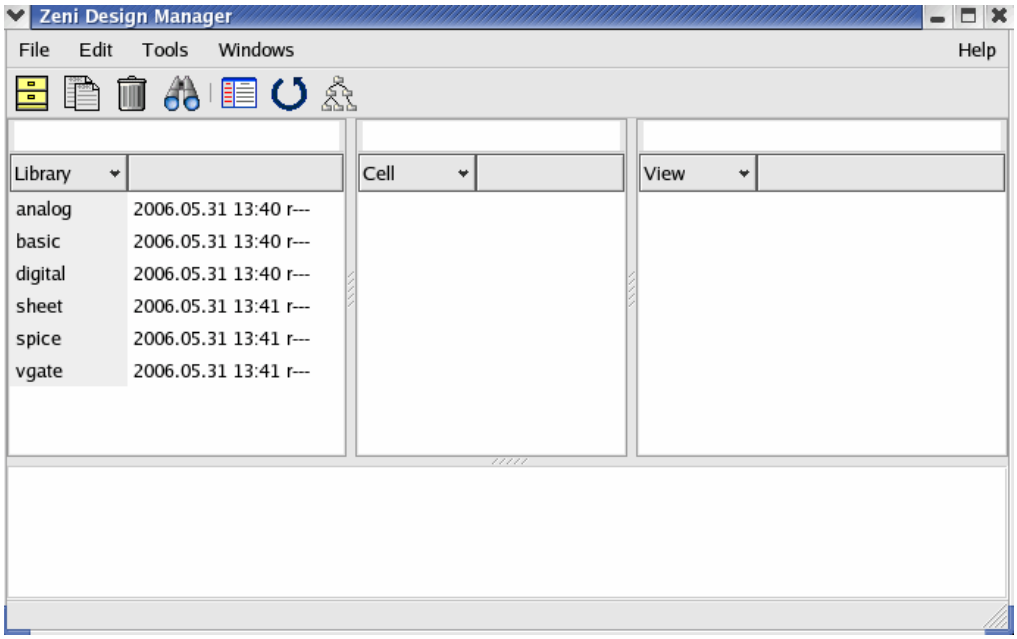
Some resource files created by Zeni will be stored under directory “.zeni”. The directory “.zeni” is under the path “xxx” you specified just now. If you don’t define environment variable “ZENI_USER_HOME”, the directory “.zeni” is under the path of $HOME.

3.2. Run Zeni Tool

Step 4: cd $WORK_DIR

Step 5: dm &

Step 6: Zeni Design Manager Window (ZDMW) appears as below.



While running Zeni Design Manager, it loads a file "zeni.lib" to finish initialization. The initialization file records library names and their locations.

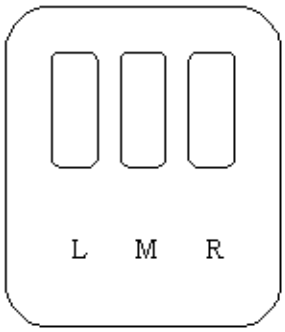
- Firstly, ZDM loads file "$ZENI_INSTALL_PATH/etc/zeni.lib". Here, $ZENI_INSTALL_PATH" is an environment variable of ZENI tool. This "zeni.lib" comes along with the tool package, and it denies any modifications from users. It records design libraries provided along with tool package, such as "analog", "basic", "spice", "sheet", "digital", "vgate", etc.
- Secondly, ZDM loads file "$WORK_DIR/zeni.lib ". Here, "WORK_DIR" represents the current working path of user. This "zeni.lib" records user-specified design libraries in

"[library map]" section. In "[hide]" section records libraries not to be shown in Zeni Design Manager Window.

4. Working With Basic

- **Mouse**

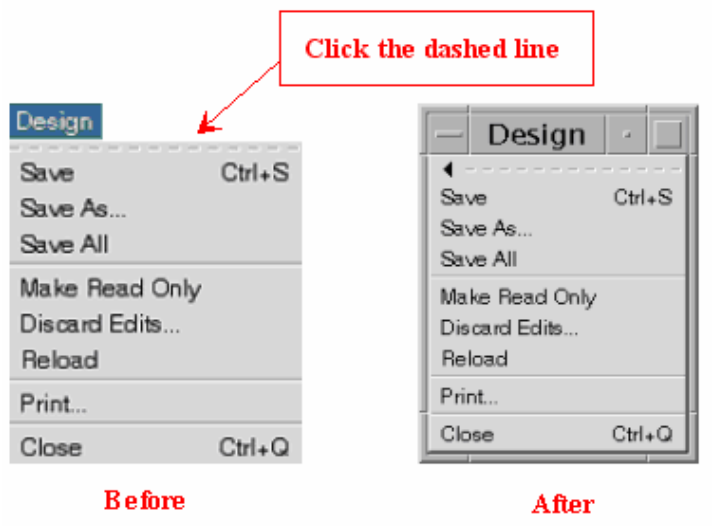
The system supports standard 3 button mouse. In general, button assignment are the following.



Left Mouse Button (LMB): Object selection.
Middle Mouse Button (MMB): Pop up the object-sensitive menu.
Right Mouse Button (RMB): Repeat the last command or command; stroke.

- **Menu**

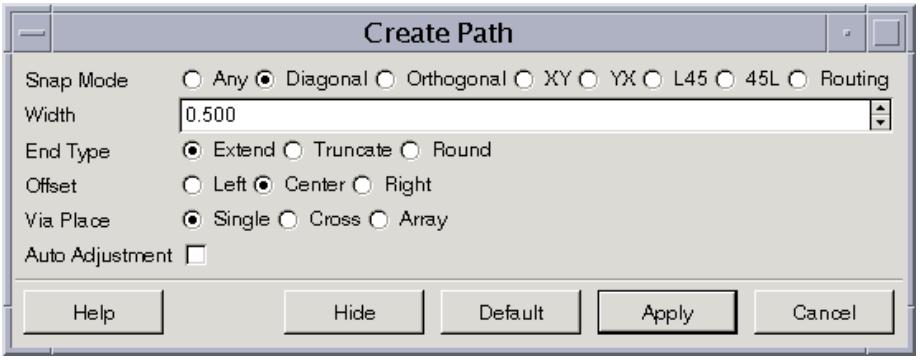
The system supports tear off line menu. If you want to use one or more commands of a menu frequently, you can click left mouse button on the dashed line, the menu will change to a menu window. Click on the menu window, you can select command directly.



- **Command Form**

A form is a window that appears when you use a command. You use the form to change

command settings. For example, in the *Create Path* form, you can change the snap mode to any of the options listed in the *Snap Mode* field.



There are two type forms in the system.

1. **Standard form** lets you change command settings before execution commands. It appears automatically when you start a command.
2. **Options form** lets you change command settings while you are running commands. When you turn off the option *Options->Editor->Popup Option Form->As Need* in Schematic/Symbol Editor or *Options->Generic-> Auto Popup Form* in Layout Editor, it does not appear automatically unless you press F3 key.

- **Form Buttons**

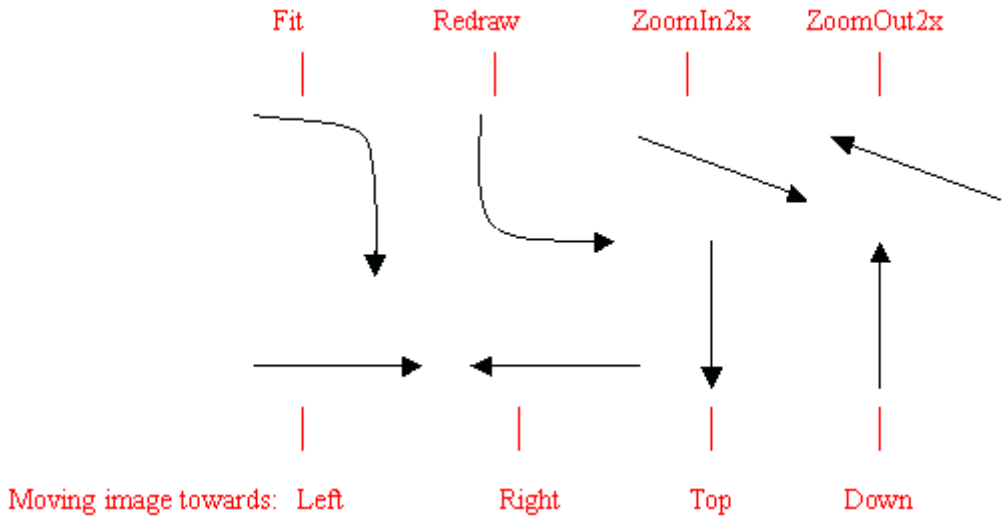
Here lists some buttons on the command form.

Help	Open the manual viewer and display help information of current command.
Hide	Close the current form but the command is going on.
Apply	Accept all current settings and execute the command.
Default	Reset all parameters to default values.
OK	Execute current command with options and terminate the command.
Cancel	Cancel current command.
Close	Close the current form without executing command.

- **Stroke**

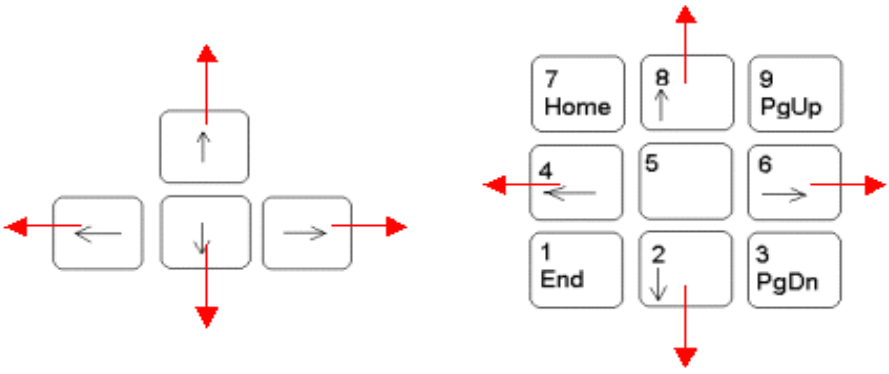
A stroke is a unique shape you draw in a cellview by right click mouse and drag it. As the figure shows below, start a command with a stroke, you should,

1. Click at the starting point of the arrow.
2. Hold and right-click mouse while move it along the direction of the arrow.
3. Release the right mouse button at the ending of the arrow.



- **Pan**

The arrow keys on the keypad let you pan the cellview in direction of left/right/top/down.



For example, press ‘6’ on the keypad, the image moves towards left.

5. A Brief Tour

This tour has been designed to allow anyone with a rudimentary knowledge of Zeni tools to complete the full flow of tools. If you keep in mind the following it will enhance your experience.

- The beginning of each exercise will contain a statement of what is "In this section…" that will let you know what you will be investigation in that module. A summary is also included at the end of the exercise. The statement looks like this:

In this section…

This heading means the text in this spot is a description of what you will be investigating in this section.

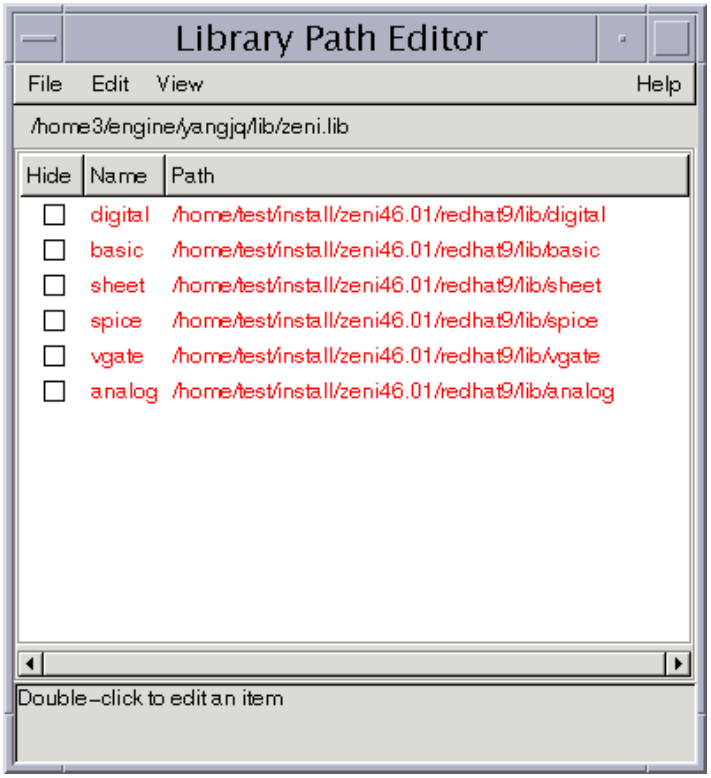
- Steps that you will need to perform at the workstation are highlighted in this manner:
  Step 1: This is something you need to do at the workstation.

- In this document, you will be directed to the correct window and action on take in that window. For example:
  Step1: In the Zeni Design Manager window LMB click File->Exit.

  Means: Find the window labeled "Zeni Design Manager" and click the Left Mouse Button (LMB) on the menu command "File" then the command "Exit" underneath it. (See example below). Note that many of the menu entries are bound to keys that may be used as a shortcut to the menu system. In this example File->Exit is bound to the "Ctrl+X" key. Pressing Ctrl and X while the cursor in the schematic window will result in execution of the File->Exit command.

5.1. Importing demo library: INV

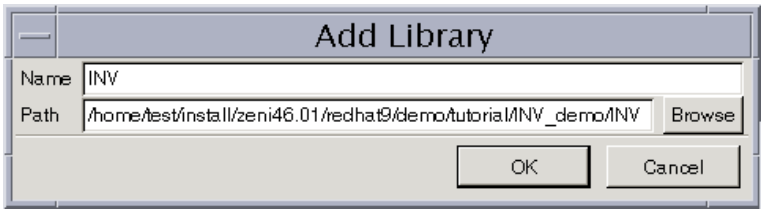
Step 1: In ZDMW, LMB (Left Mouse Button) click Tools->Library Path Editor. The Library Path Editor form appears as below.



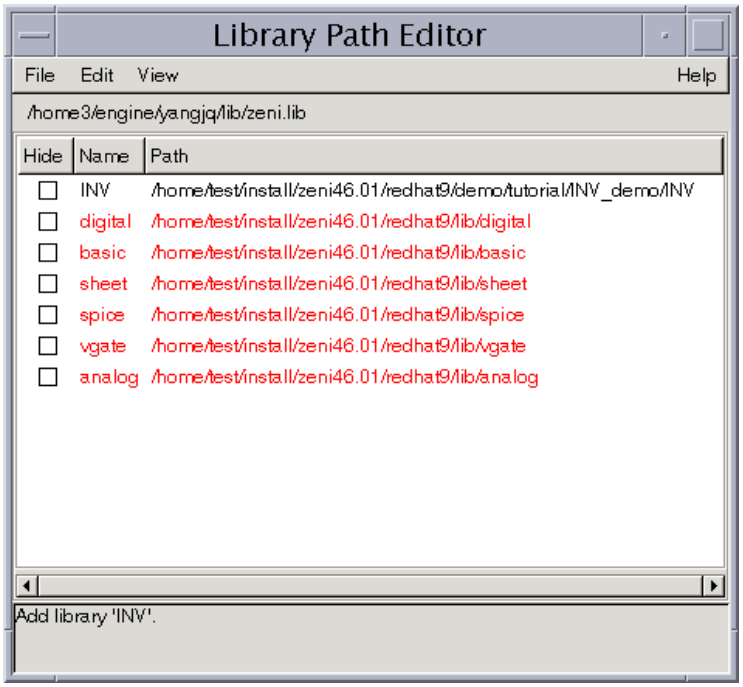
Step 2: In Library Path Editor form, LMB click Edit->Add.

Step 3: In Add Library form, click “Browse” to choose INV library under the path $ZENI_INSTALL_PATH/demo/tutorial/INV_demo.

Step 4: Ok the File Selection form. The Add Library form looks like below.



Step 5: Ok the Add Library form. A new library INV is added in Library Path Editor form.

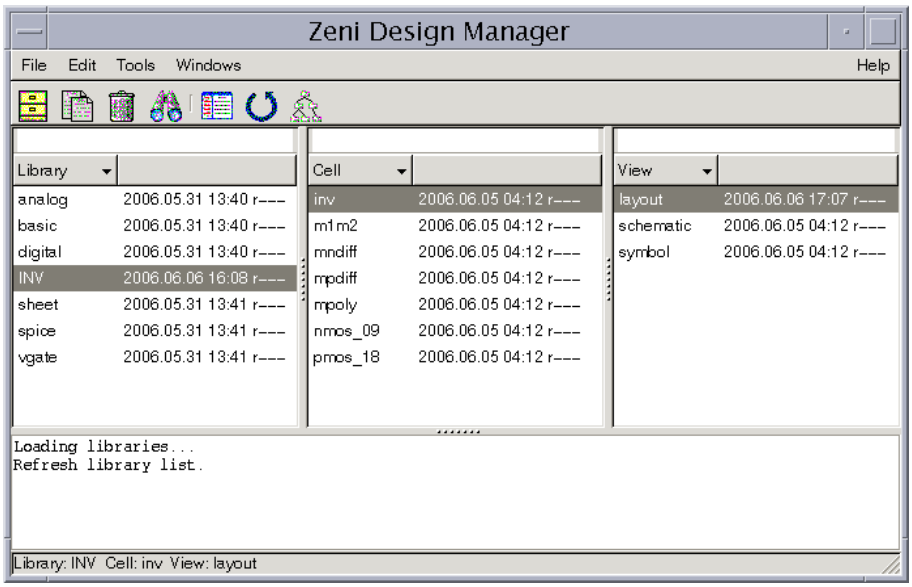


The six library definitions in red represent are System Library. The Normal Library is in black color.

Step 6: In Library Path Editor, LMB click File->Save.

Step 7: Exit this Library Path Editor by File->Quit.

You can find the demo library INV is displayed in the Library List of the ZDMW.

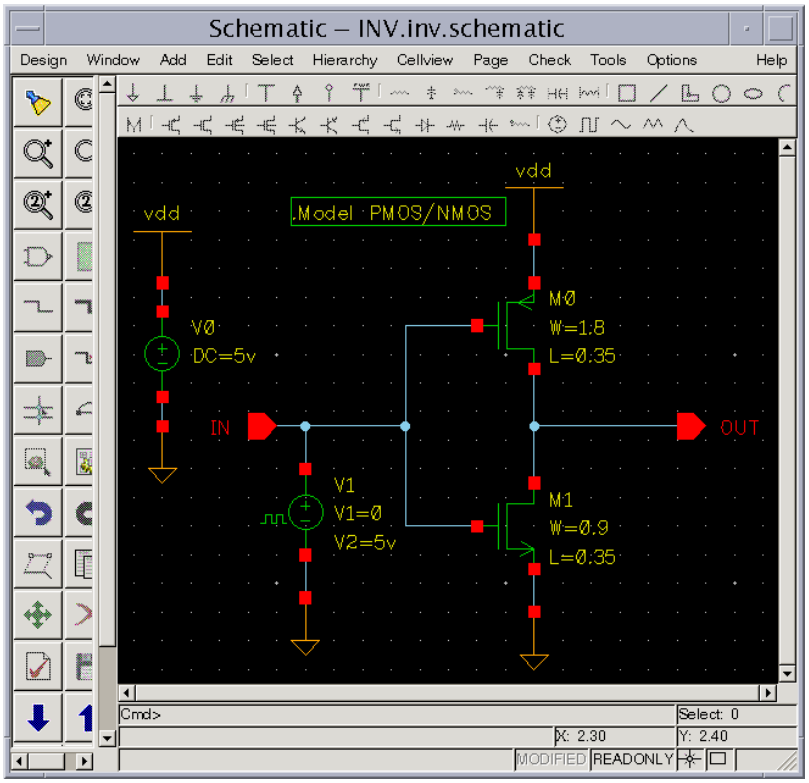


Step 8: In the ZDMW, from left to right, LMB click on:
INV

inv

schematic (double-click to open the schematic cellview)

The schematic cellview of inverter is opened as shown in the figure below.



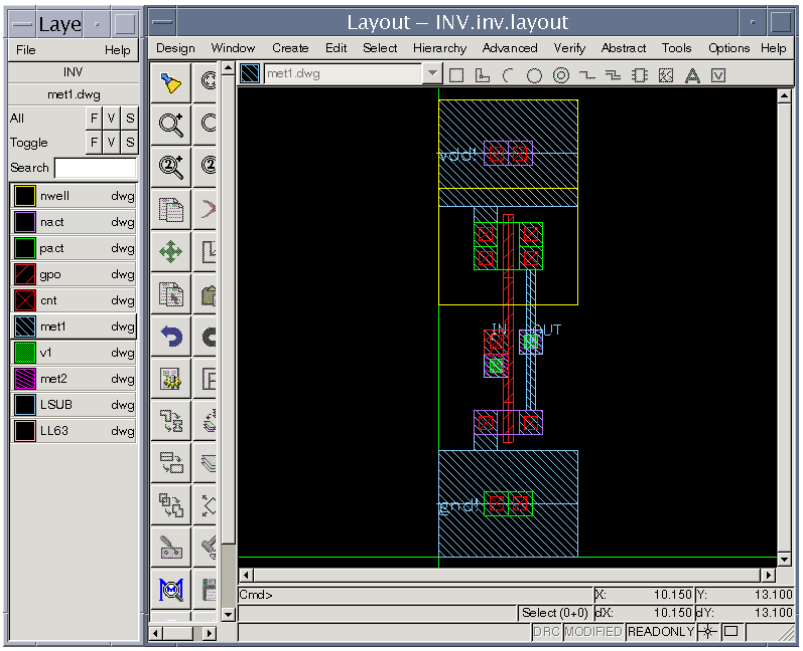
Step 9: LMB click again on:

INV

inv

layout (double-click to open the layout cellview)

The layout cellview of inverter is opened. Press “Ctrl+F” key to show design with full hierarchy. The figure is shown as below.



You can find that these two cellviews are read-only mode because the library comes along with the tool package, it is not accessible to you for any modifications.

In this workshop, creating above two cellviews is your first step for preparation.

5.2. Schematic Capture

Zeni Schematic Editor (ZeniSE) is a powerful schematic editing tool supporting hierarchical design method and multi-window, multi-page editing design environment. It is capable of exporting schematic data in various standard netlist formats and translating the CDL or Verilog netlist to schematic data. ZeniSE also supports importing and exporting of EDIF file, and this feature helps you seamlessly exchange the design data between Zeni and other EDA tools. Furthermore, with Analog Simulation Deck, Pre-defined Parasitic Loads, and many other useful features, ZeniSE offers you a complete circuit design environment to finish sophisticated IC designs quickly and easily.

In this section…

We will create a new library “INV1”, cell “inv” and cellview “schematic” in ZDMW first, and develop the schematic topology of an inverter circuit in ZeniSE .

5.2.1. Start software (Jump if you have already started Zeni)

Step 1: cd $WORK-DIR

Step 2: dm &

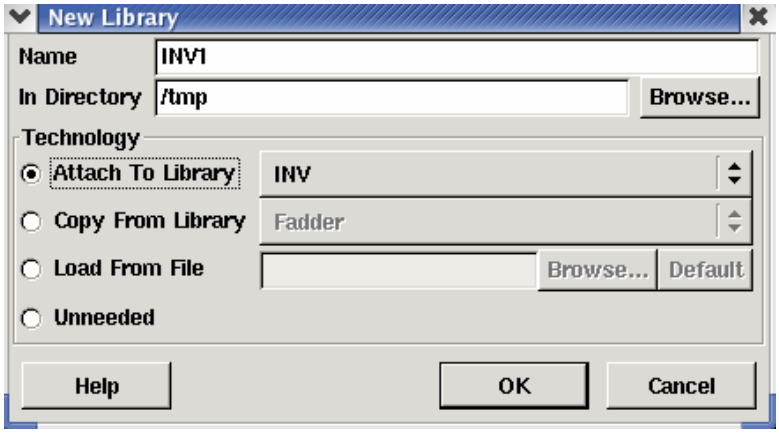
The ZDMW will appear.

5.2.2. Create a new design

We will create a new, clean schematic cellview for this section. The following steps are used to create this view.

Step 3: In ZDMW, LMB click File->New->Library.

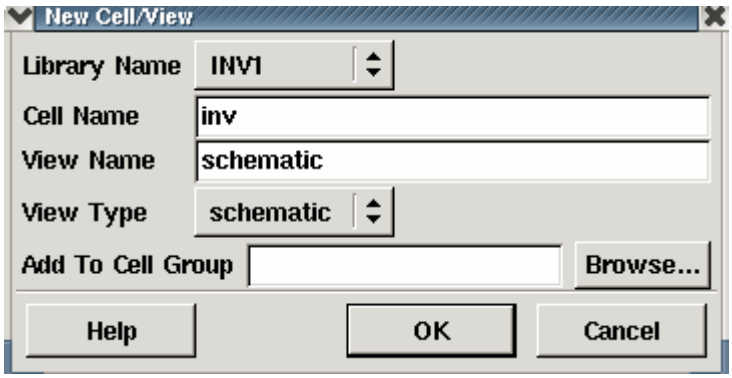
Step 4: Fill out the New Library form with the following information as shown in the figure below.
Library Name: INV1
In Directory: /tmp
Technology : attach to library INV



Step 5: Ok the New Library form.

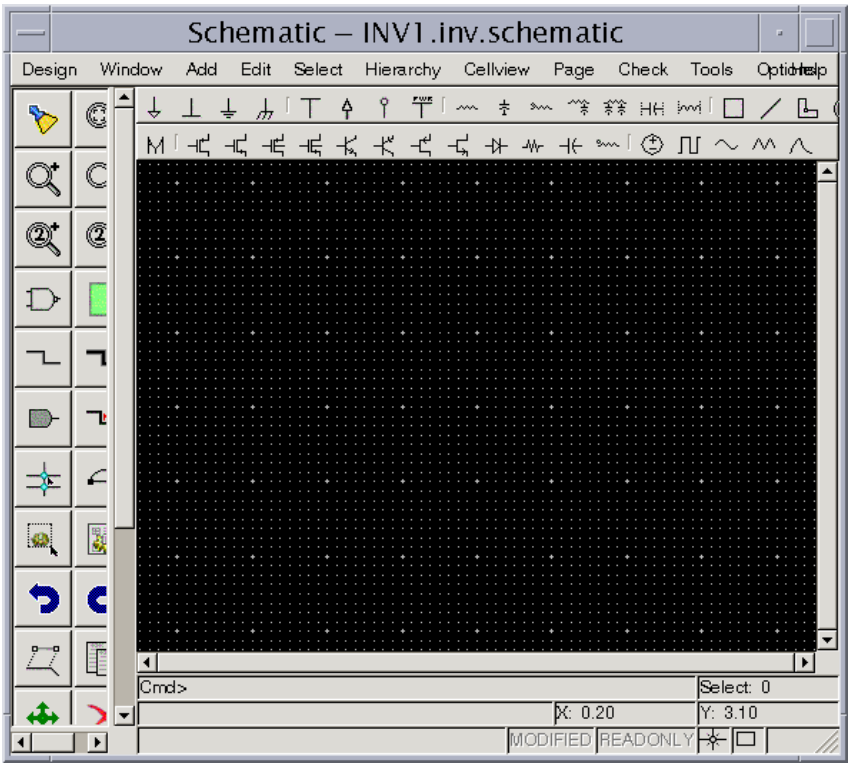
Step 6: In ZDMW, RMB (Right Mouse Button) click on the “INV1” library, select “New Cell/View” command from the submenu list.

Step 7: Fill out the New Cell/View form with the following information as shown in the figure below.
Library Name: INV1
Cell Name: inv
View Name: schematic
View Type: schematic



Step 8: Ok the New Cell/View form.

Step 9: The Schematic Editor window will appear automatically while the new schematic cellview is created.



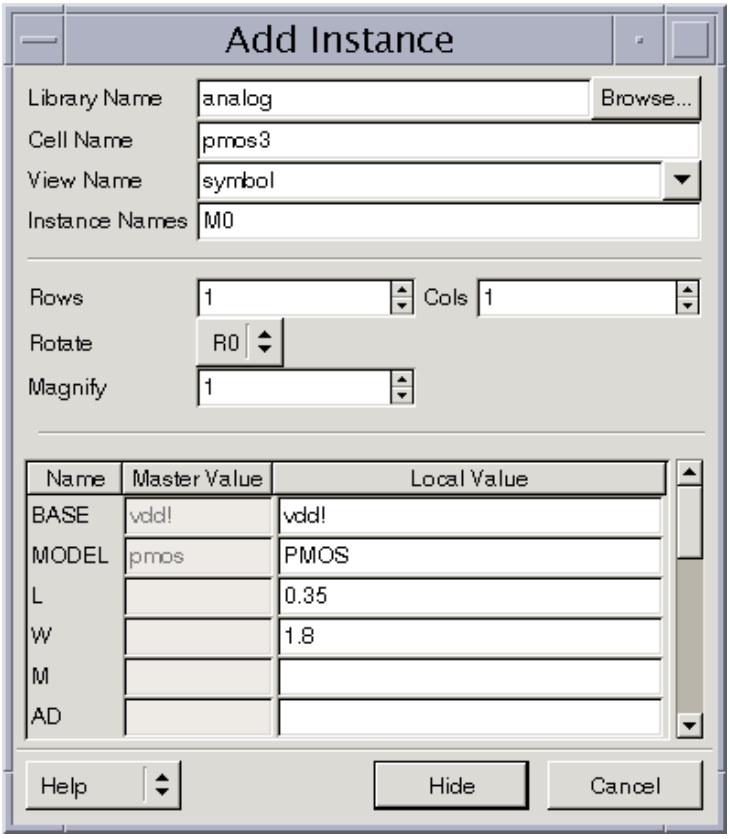
5.2.3. Design a schematic topology of inverter

In this section, we will design a schematic topology of inverter with basic features provided by Zeni Schemaitc Editor.

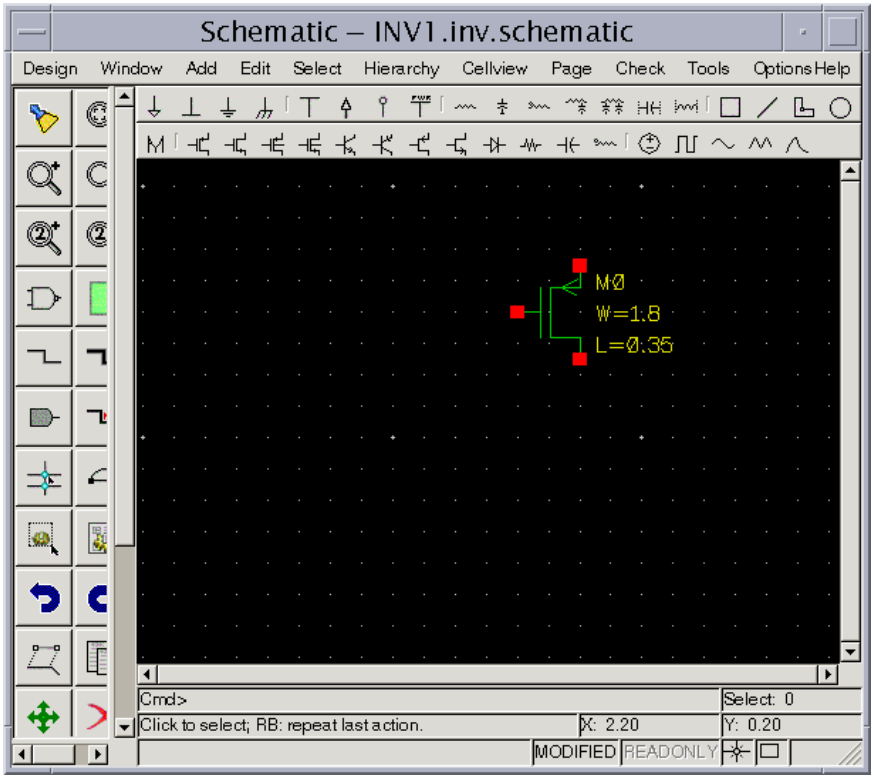
Step 10: Click on “pmos3” icon  from banner menu.

At this time, you can find an “Add Instance” form appears in the top of the screen, which means the pmos3 is from analog.pmos.symbol. In fact. “analog” library is a System Library that is provided along with the tool package. Other system libraries are “basic”, “spice”, “sheet”, “digital”, and “vgate”. These system libraries are not accessible to you for any modification, and they are recoded in $ZENI_INSTALL_PATH/etc/zeni.lib. Contrasted with the system library, it is called Normal Library, such as INV1. The normal library is recoded in $WORK_DIR/zeni.lib.

Step 11: In Add Instance form, type “M0” in Instance Names entry, type the following information to parameter list.
BASE: vdd!
MODEL: PMOS
L: 0.35
W:1.8



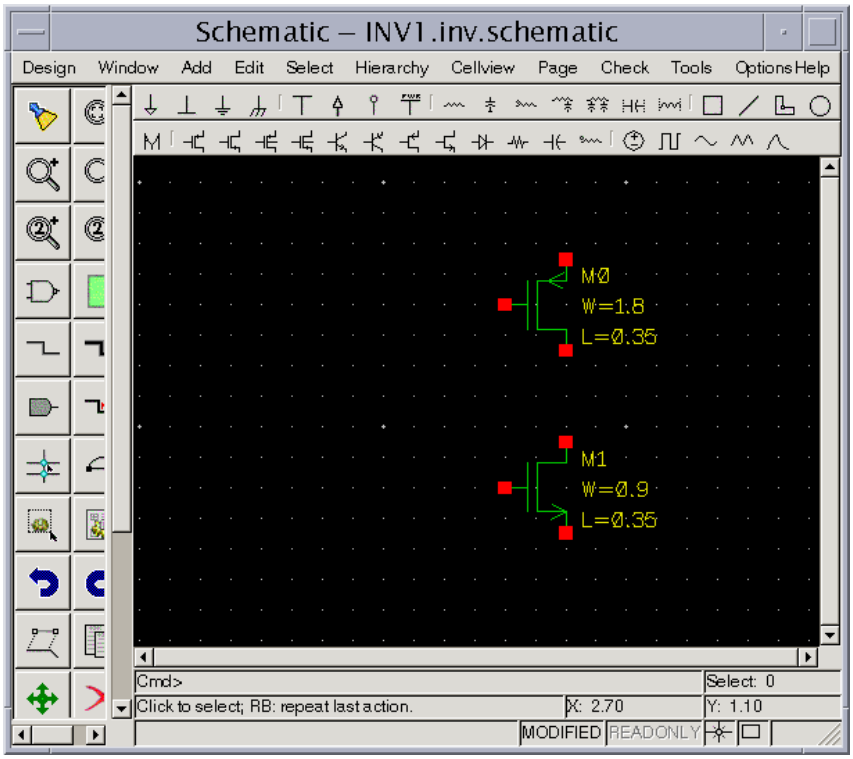
Step 12: Click Apply button in Help dropdown list and move your cursor back into Schematic Editor window. Click LMB anywhere in the Schematic Editor window to place the pmos3.



Step 13: Like the Step 10, choose “nmos3” icon from banner menu and fill out the “Add

Instance" with the following information.
Instance Name: M1
BASE: gnd!
MODEL: NMOS
L:0.35
W:0.9

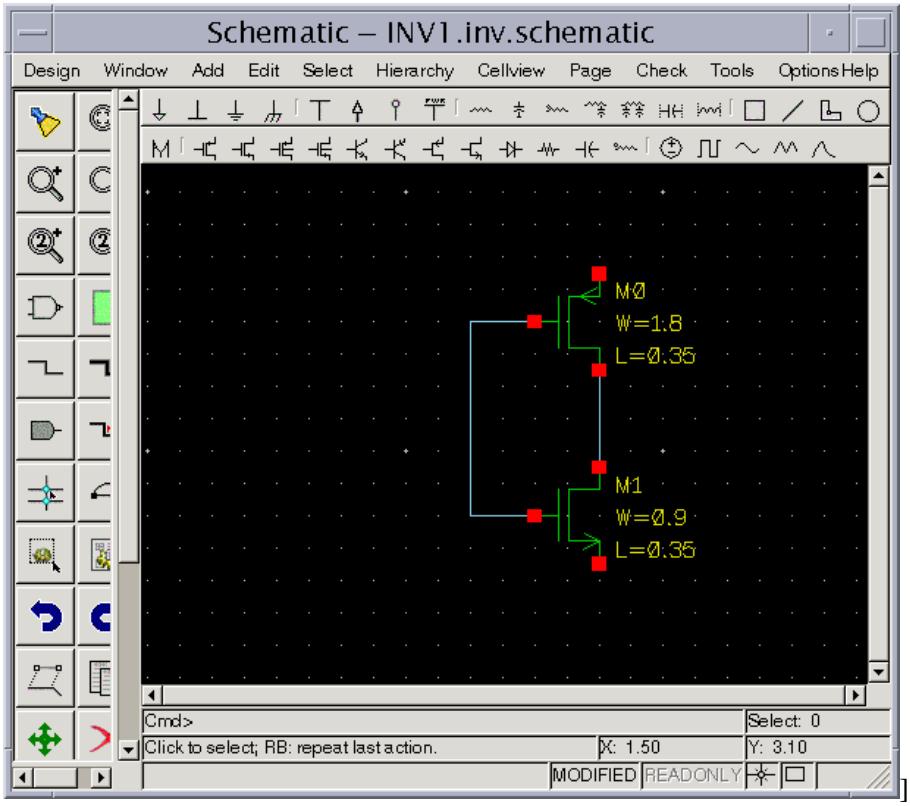
Step 14: Place nmos3 to the Schematic Editor window as below.



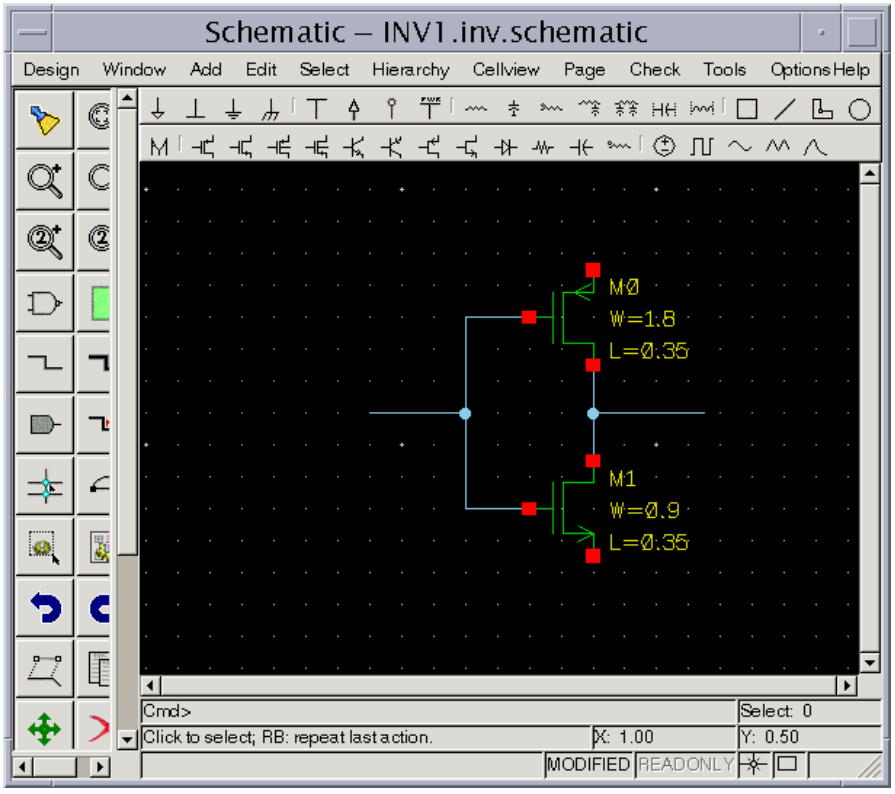
Step 15: Cancel the "Add Instance" form.

Step 16: LMB click the wire icon or click Add->Wire.

Step 17: Connect the drains between M0 and M1 through LMB clicking drain of M0 and moving cursor to drain of M1, LMB click on this drain to terminate the wire. In the same way, connect the gates of M0 and M1. During moving cursor, you may find the nearest pin to the cursor will have a small yellow diamond, pressing the "control-LMB" or "shift-LMB" key will snap route the current wire to that point. Use "Esc" key to cancel Add Wire command.

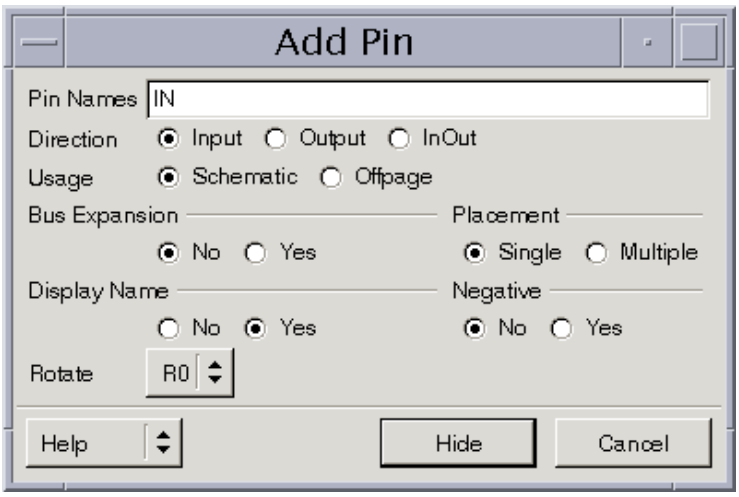


Step 18: Respectively draw two wires as the figure below.

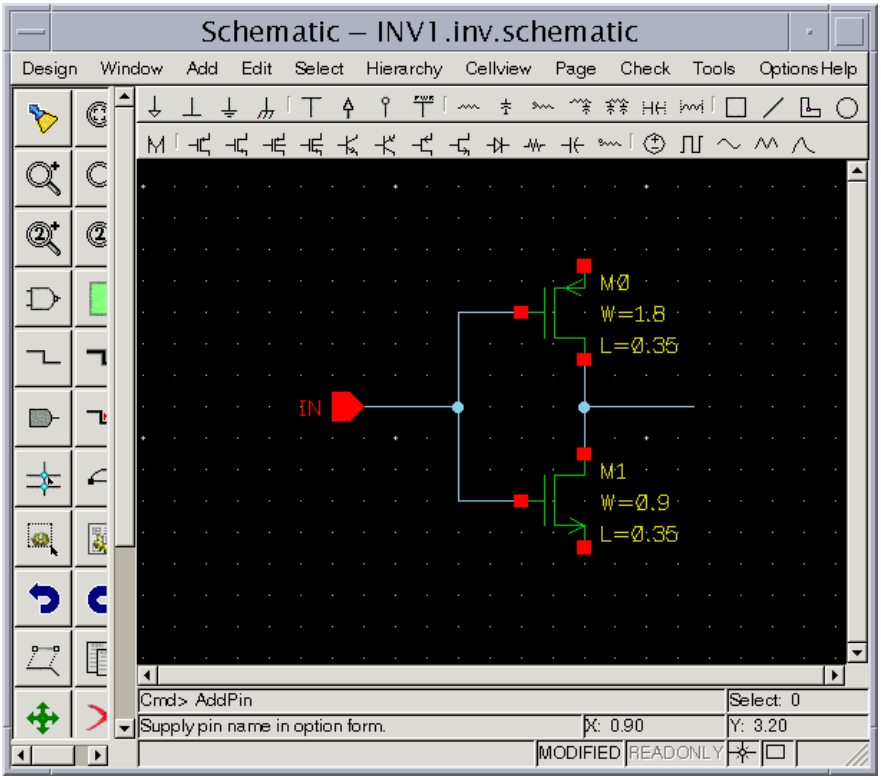


Step 19: LMB click the Pin icon  or click Add->Pin.

Step 20: In “Add Pin” form, fill out the Pin Name with “IN”; set pin direction to “Input”.

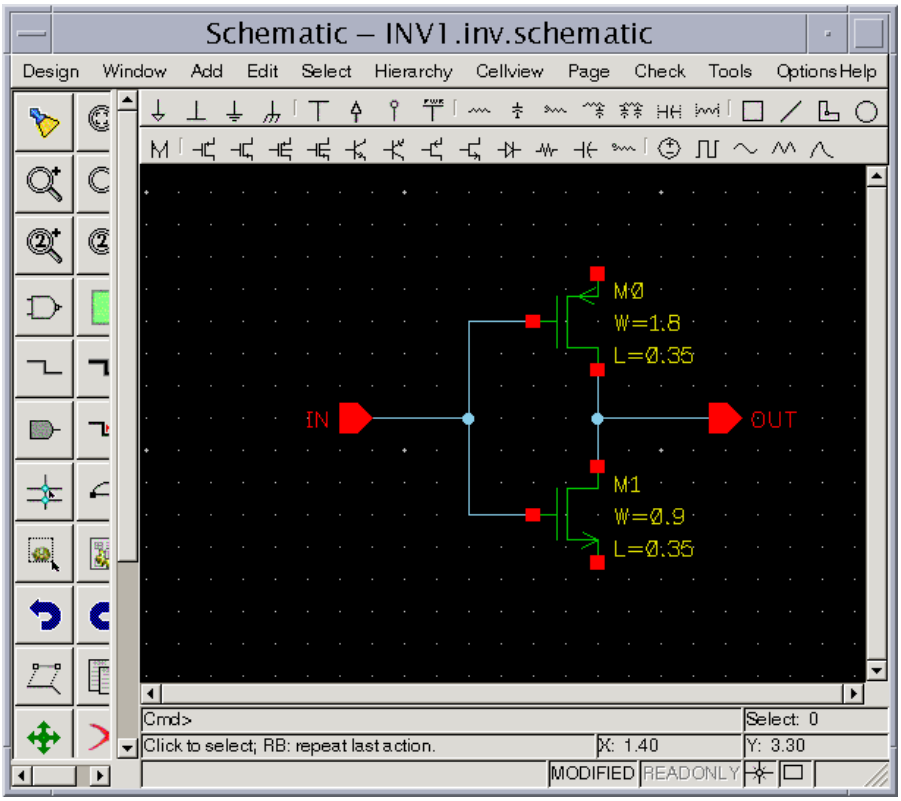


Step 21: Place the pin to a position to connect the wire. The figure is shown as below.

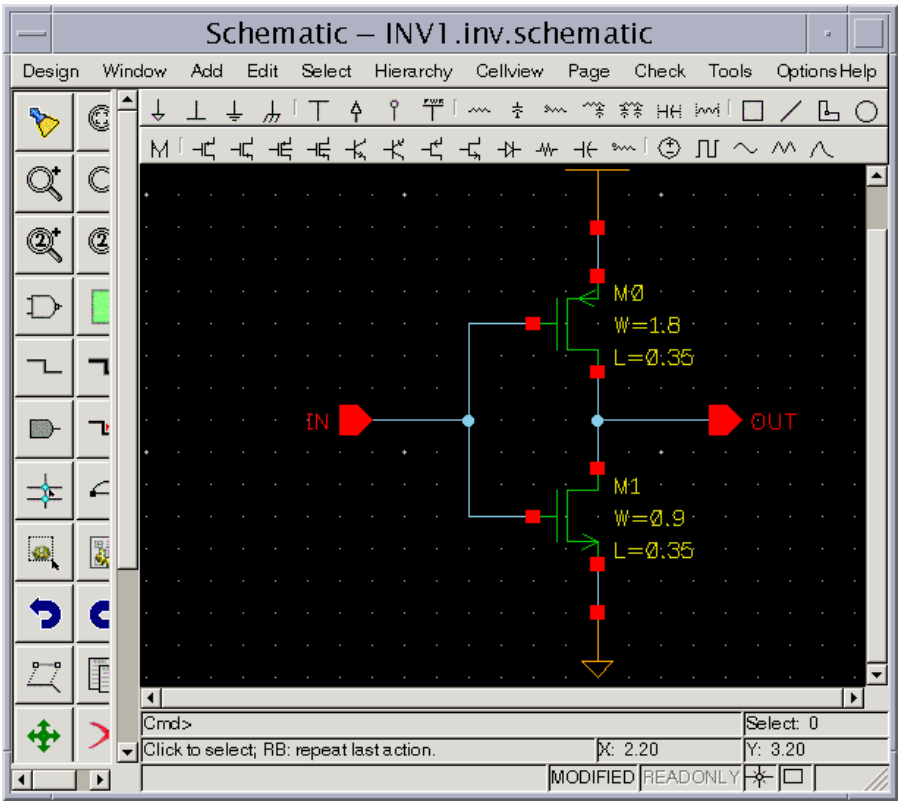


Step 22: In “Add Pin” form, fill out the pin name with “OUT”; set pin direction to “Output”.

Step 23: Place the “OUT” pin to connect the wire. The figure is shown as below.



Step 24: LMB click the vdd icon ( ) and ground icon ( ) from the banner menu., place them to appropriate location and connect them to MOS pin with wire.



Step 25: Up to now, we have finished a design of inverter. LMB click Design->Check and Save to save this design.

In this section, we learned how to create a new schematic cellview in Zeni Design Manager and how to create a schematic topology with simple operation commands in Zeni Schematic Editor.

5.3. Simulation in Zeni Schematic Editor

Circuit simulation is a highly valuable activity and ensures the success of a design project. In this workshop, we use spice3 simulator for circuit analysis.

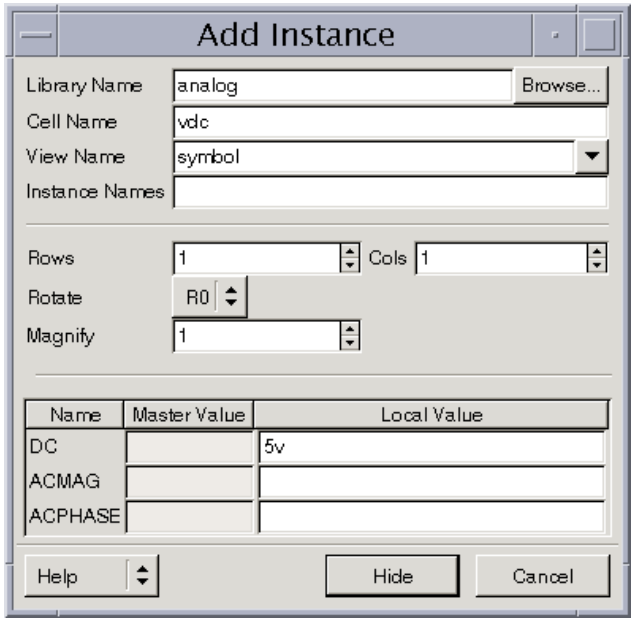
In this section…

We will add stimulates on input pins, and use simulator spice3 to analysis circuit performance. Finally, we can take a look at the waveform in the Zeni Waveform Viewer. If your simulator is ngspice, the flow for ngspice is the same as spice3.

5.3.1. Add stimulates on the input pins

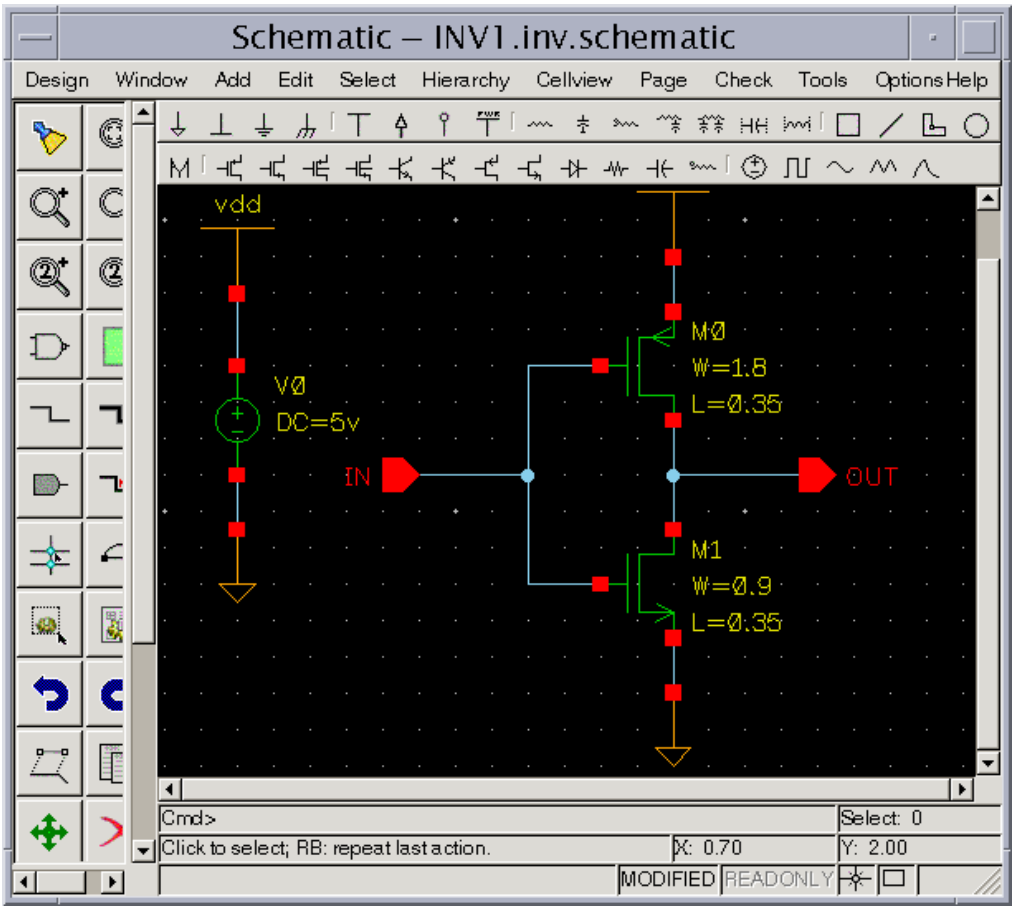
Step 1: In Zeni Schematic Editor window, LMB click  icon from banner menu.

Step 2: Fill out the Add Instance form with the following information as shown in the figure below.
Library Name: analog
Cell Name: vdc
View Name: symbol
DC: 5v



Step 3: Move your cursor back into Schematic Editing window and click LMB anywhere in the schematic to place a vdc.

Step 4: Connect the vdc with vdd and gnd.



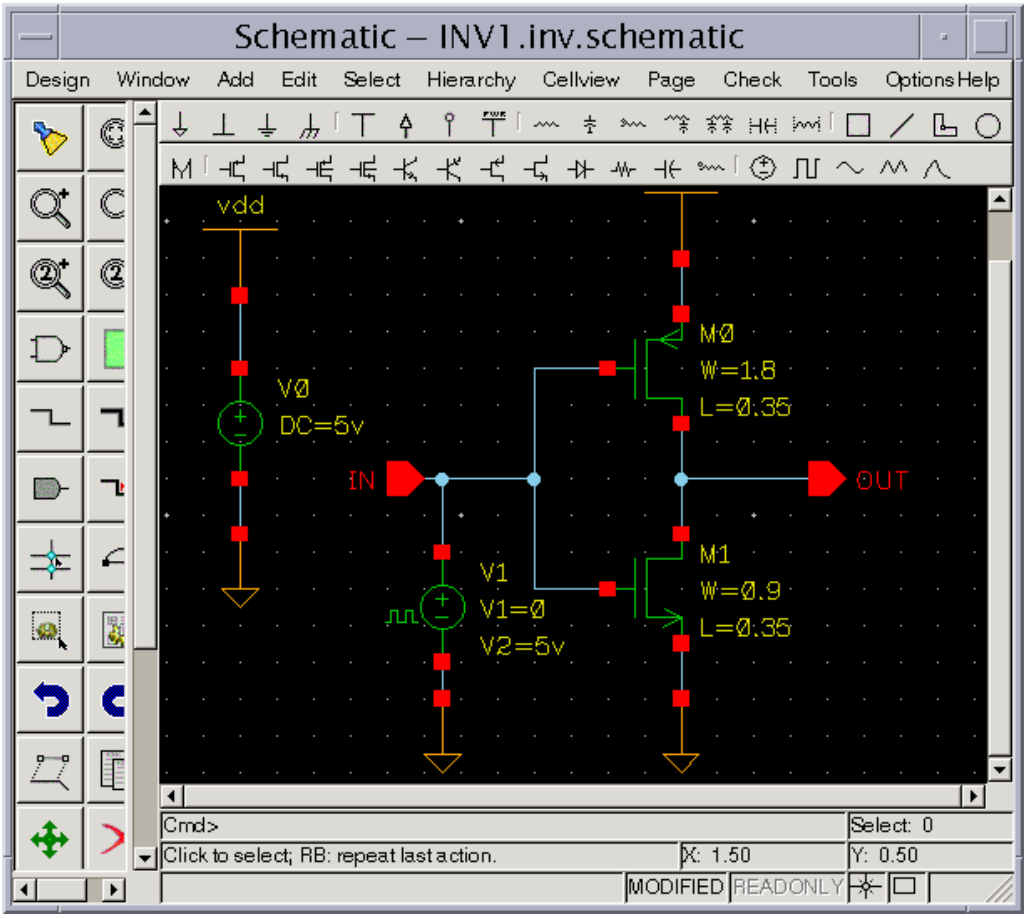
Step 5: LMB click  icon from banner menu.

Step 6: Fill out the Add Instance form with the following information.

Library Name: analog
Cell Name: vpulse
View Name: symbol
Instance Name: V1
DC: 0
V1: 0v
V2: 5v
TD: 0
TR: 1ns
TF: 1ns
PW:20ns
PER: 40ns

Step 7: Or, LMB double click the cellview: “INV.inv.schematic” from Zeni Design Manager window, in “INV.inv.schematic” cellview window, LMB click Edit->Copy and select the instance V1, drag the V1 to “INV1.inv.schematic” cellview window and place it anywhere. Close “INV.inv.schematic” window.

Step 8: Connect the vpulse to the input pin “IN”.

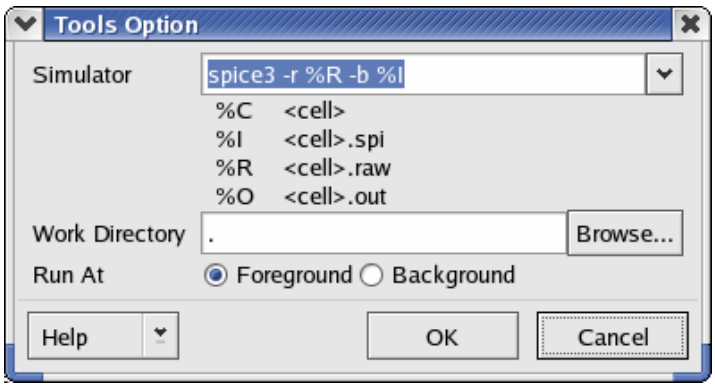


Step 9: LMB click Design->Check and Save. If there is any error, you should fix it before simulation.

5.3.2. Set up Simulation Environment

Step 10: In the Schematic Editor window, click on *Options->Tools…*. The “Tools Option” form appears.

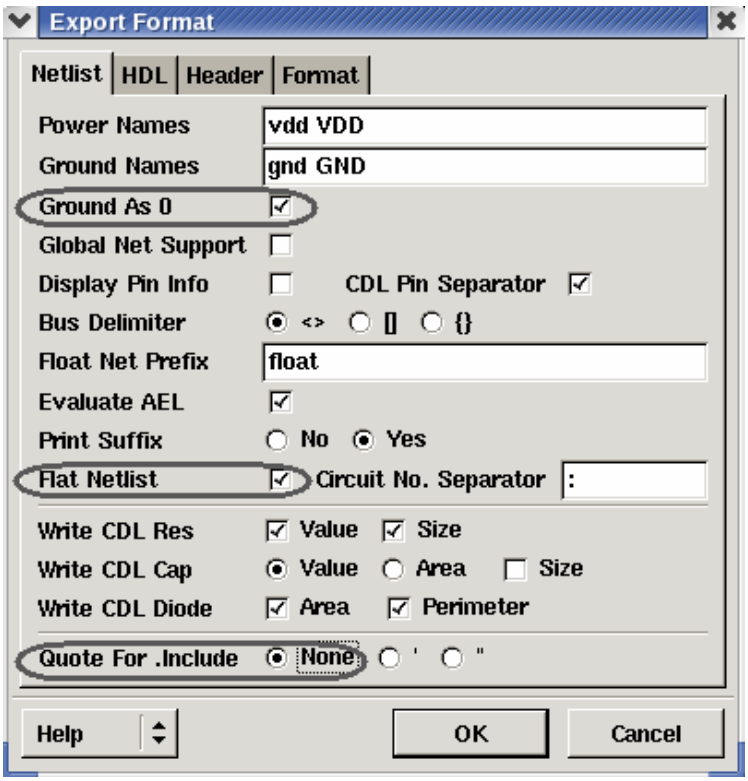
Step 11: Select simulator “spice3” from Simulator dropdown list. The figure is shown as below.



Zeni Schematic Editor pre-provides 6 kinds of simulator interfaces. If your simulator doesn't in that list, you still can invoke your simulator by typing command in *Simulator* field. For example, if your simulator is "ngspice", please type "ngspice –r %R –b %I" in *Simulator* field.

Step 12: Ok the "Tools Option" form.

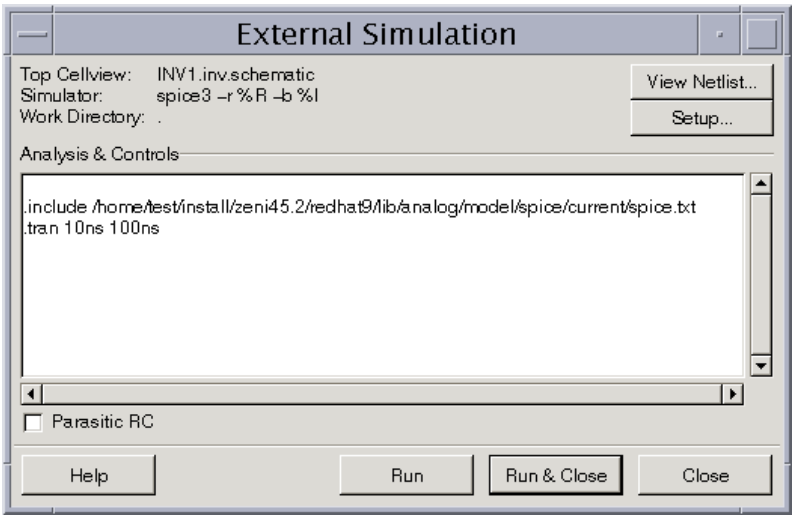
Step 13: As one simulator only accepts the netlist format of its own, according to the simulator you used, click on Options->Export Format->Netlist to configure the netlist format. For spice3 or ngspice, we configure the netlist format as below.



Step 14: Ok the Export Format form.

Step 15: Click the Tools->External Simulation. The External Simulation form appears.

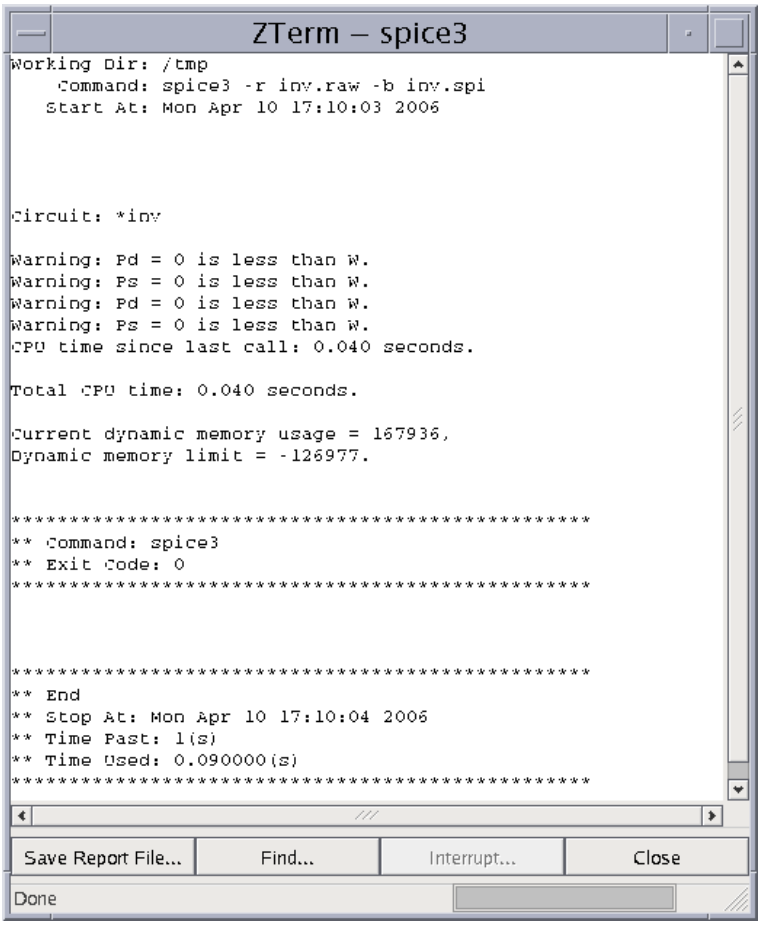
Step 16: In Analysis&Controls section, fill out the circuit analysis type and/or control expression. Here, we type the information as below.



Step 17: Please replace the location of modal file to the path you used.

Step 18: Before running simulator, you can also click View Netlist… button to view the netlist generated by Zeni Schematic Editor.

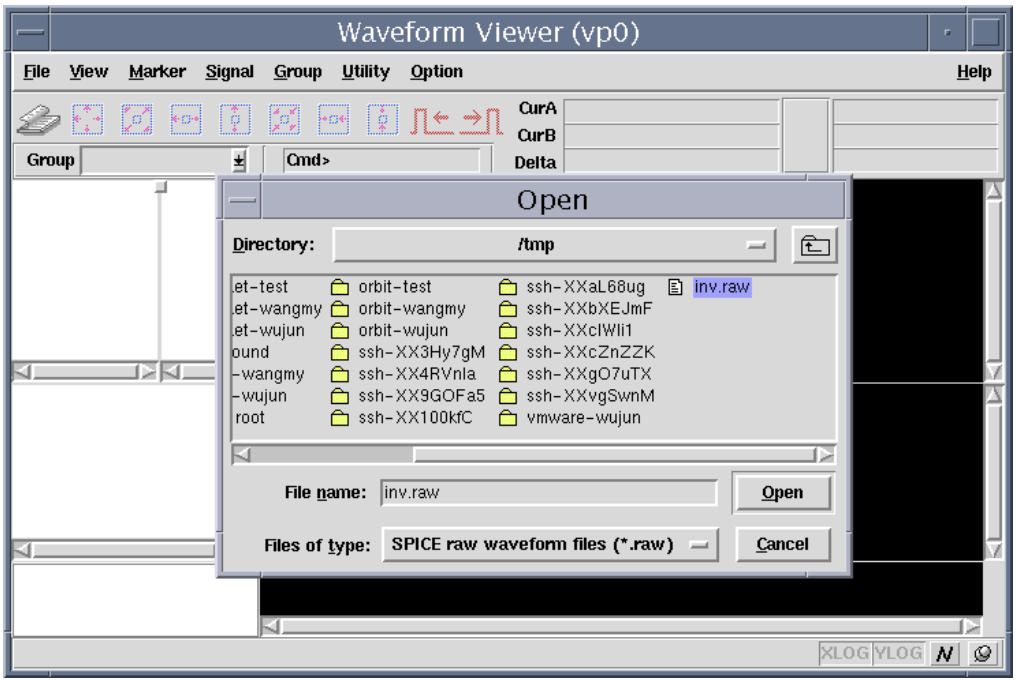
Step 19: In External Simulation form, LMB click Run&Close button to call spice3 simulator and run it. A ZTerm-spice3 form appears as below.



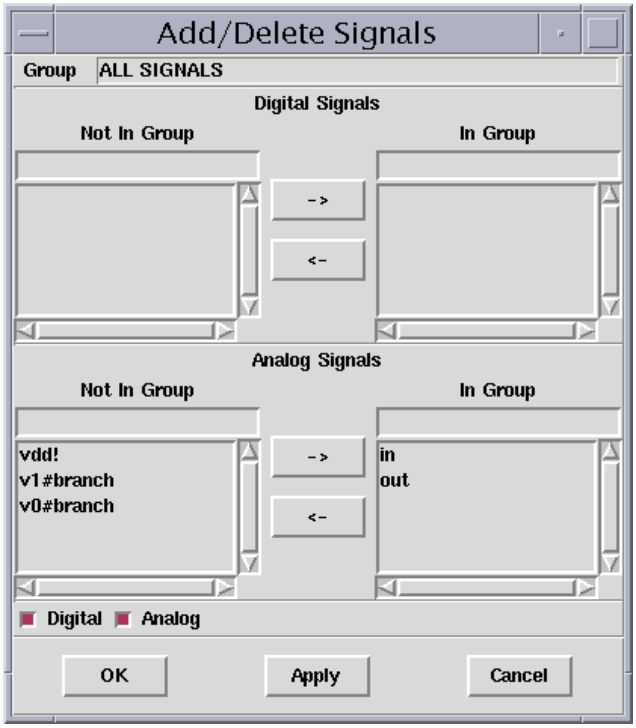
Step 20: Close this Zterm.

Step 21: In Zeni Design Manager window, LMB click Tools->Waveform Viewer. The Waveform Viewer window appears.

Step 22: In Waveform Viewer window, LMB click File->Open Plot, open file “inv.raw”.

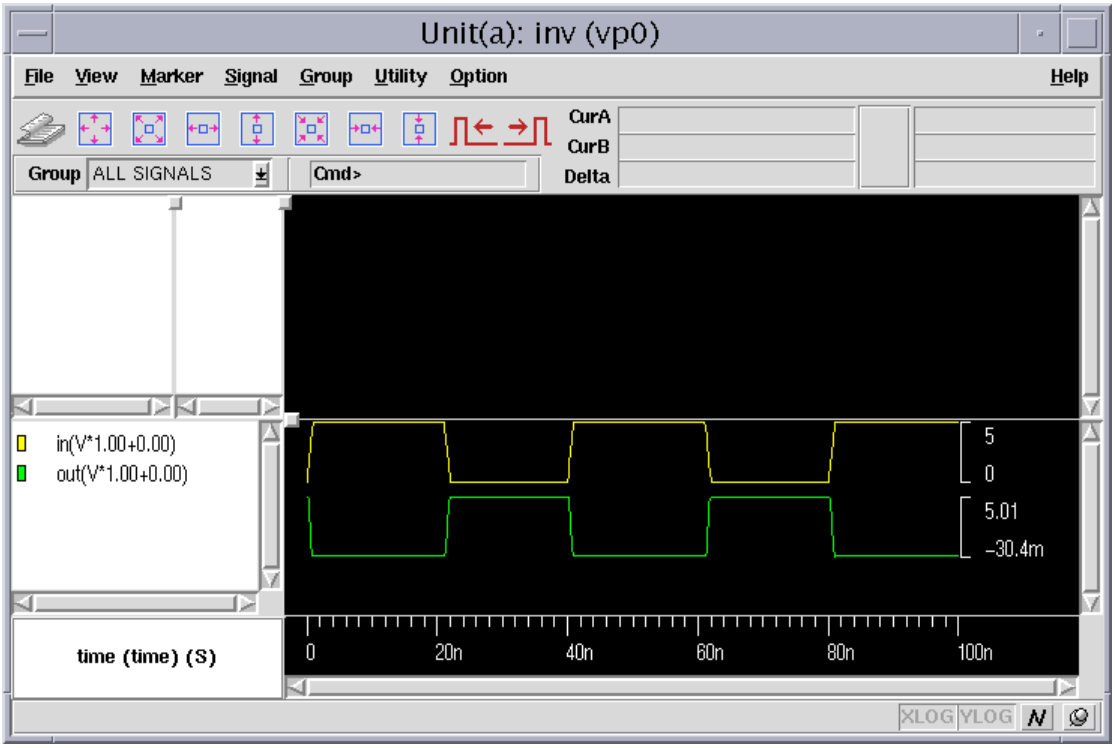


Step 23: LMB click Signal->Add/Delete Signals…. Move valuable signals to In Group list.



Step 24: Ok the Add/Delete Signals form.

Step 25: In Waveform Viewer form, you can see two signals overlapped. LMB click Option->Custom->Analog Signal, turn on the Stack option to separate them.



In this section, we learned how to add stimulus to the input pins, and used spice3 simulator to analysis circuit performance. Finally, we took a look at the waveform of inverter in the Zeni Waveform Viewer.

5.4. Layout Editing

Layout designer begins with a set of schematics and processes to build up a physical representation of the circuit with Zeni Physical Design Tool (ZeniPDT). ZeniPDT is a fully hierarchical multi-window editing environment. It supports physical implementation of custom digital, analog and mixed-signal designs at device, cell and block levels. It also offers support for physical design debug and verification by embedded physical verification tools.

In this section…

We will create two components of inverter, for time saving, other cells will be called from library INV.

5.4.1. Start software (Jump if you have already started Zeni)

Step 1: cd $WORK-DIR

Step 2: dm &

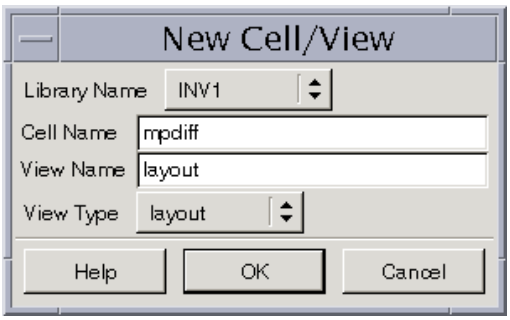
Step 3: The ZDMW will appear.

5.4.2. Design a layout of inverter

Step 4: RMB click INV1 library and select “New Cell/View” submenu.

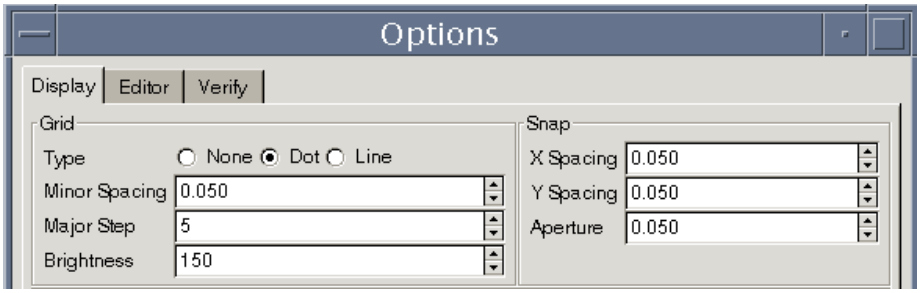
Step 5: Fill out the New Cell/View form with the following information as shown in the figure below.

Library Name: INV1
Cell Name: mpdiff
View Name: layout
View Type: layout



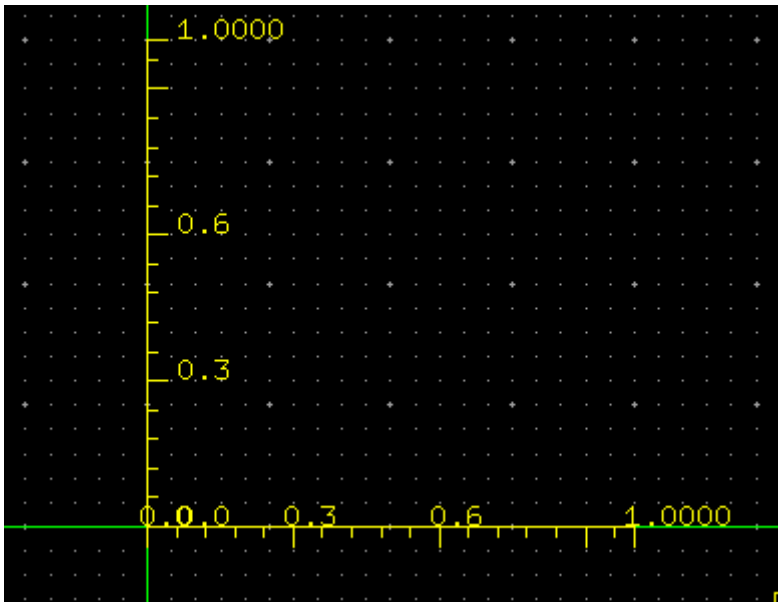
Step 6: Ok the New Cell/View form, the layout editor window appears.

Step 7: LMB click Options->Generic; modify spacing value in Display tab.



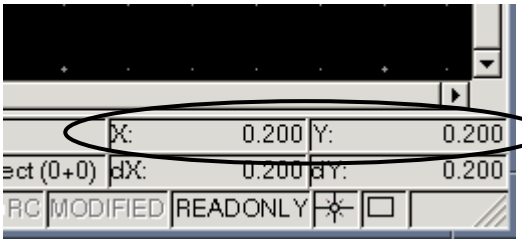
Step 8: Save and close the Options form.

Step 9: Zoom in the display area and LMB click Tools->Ruler or click  icon to draw coordinate from the origin. Press “Esc” to terminate the command.

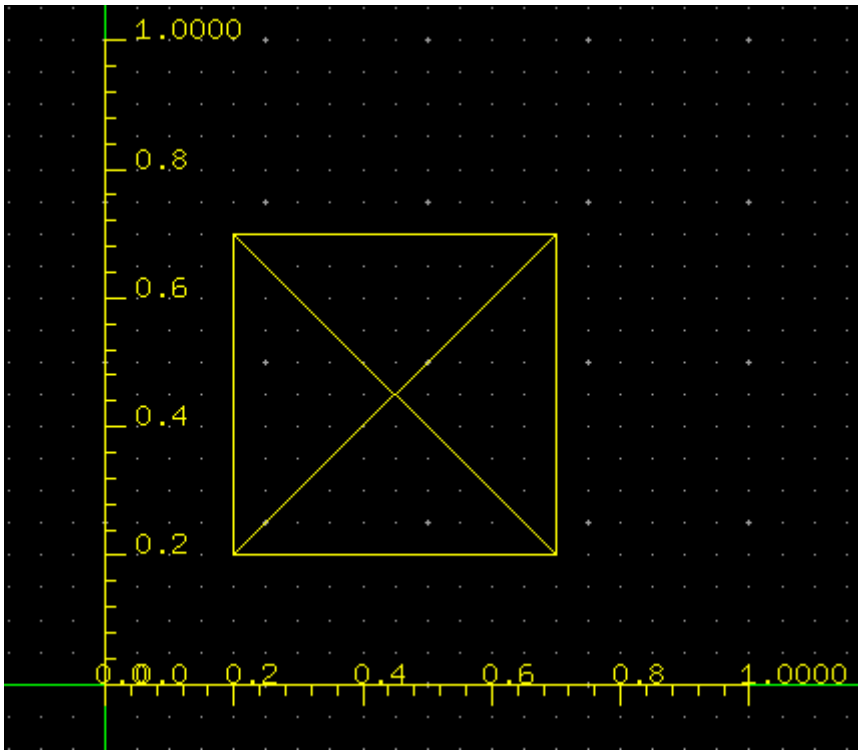


Step 10: Choose “cnt.dwg” from the Layer palette.

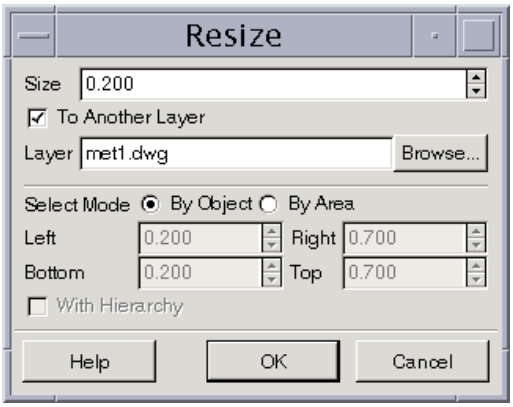
Step 11: LMB click Create->Rectangle or click  icon to draw a rectangle from (0.2, 0.2) to (0.7, 0.7). Current cursor coordinate is shown in the right bottom of layout editing window as figure below.



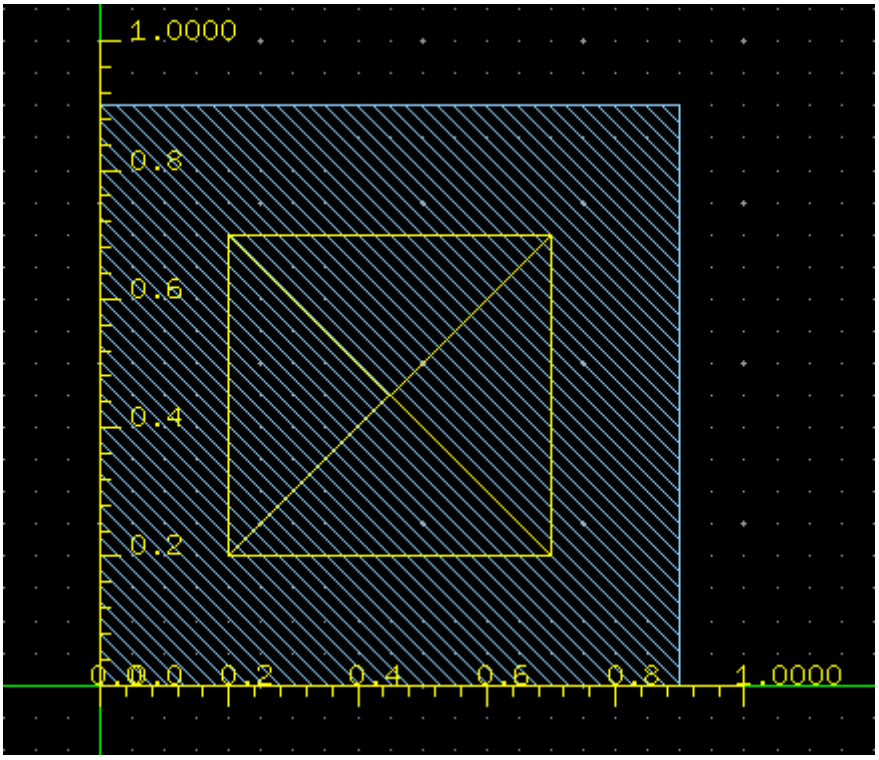
Step 12: Press “Esc” to terminate the command. The figure is shown as below.



Step 13: Select the rectangle first, LMB click Advanced->Resize, the Resize form appears. Fill out the form as the following figure.

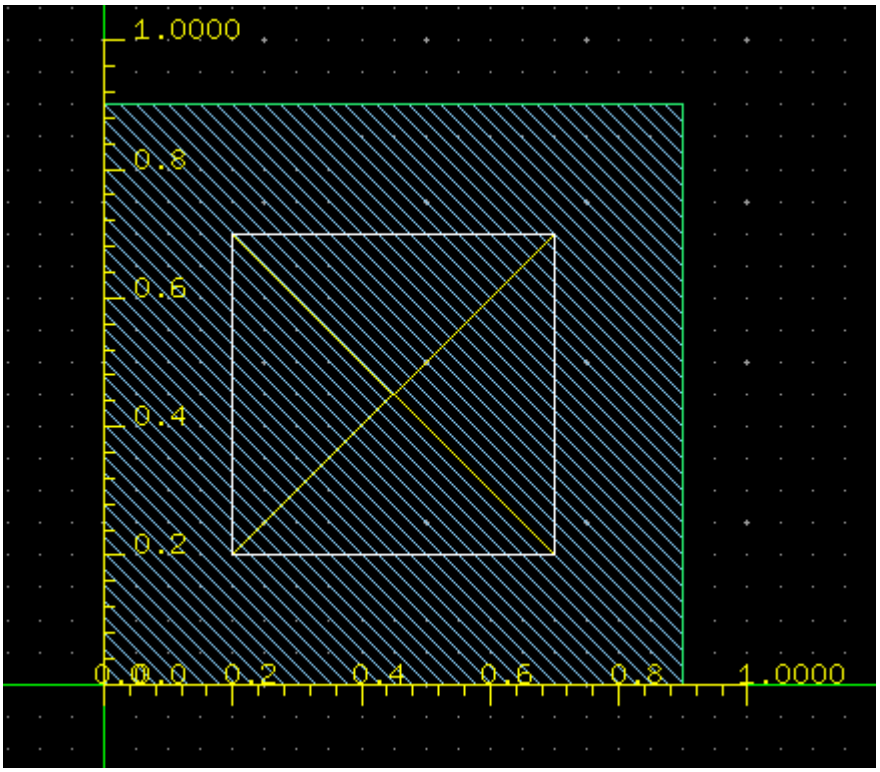


Step 14: Ok the Resize form. The layer “met1.dwg” is created automatically with enlarge 0.2 micron around the layer “cnt.dwg”.

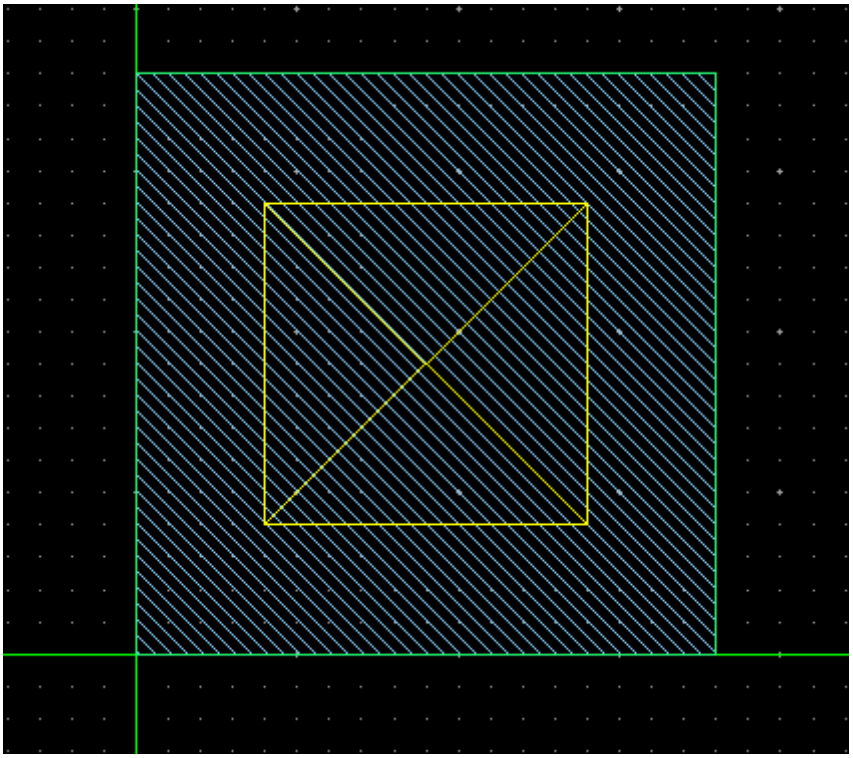


Step 15: Of cause, as Step 11, you can create the “met1.dwg” with drawing rectangle directly.

Step 16: In the same way, you need to create layer “pact.dwg” with the same size as “met1.dwg”.



Step 17: LMB click Tools->Clear Ruler or click  icon to remove the coordinate. The cellview “mpdiff” is shown as below.

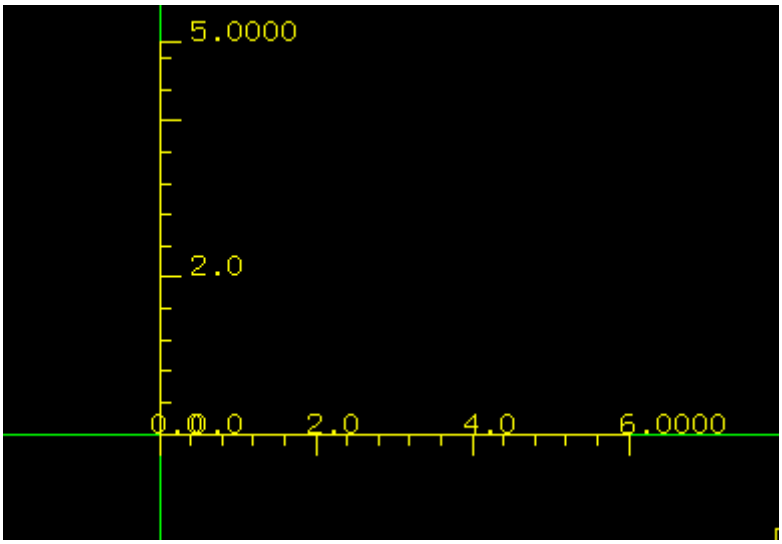


Step 18: LMB click Design->Save or click  icon to save the cellview.

Step 19: LMB click Design->Close to close the cellview.

In the next steps, we will create another cellview: pmos_18.

Step 20: As Step 9: drawing ruler like figure below.



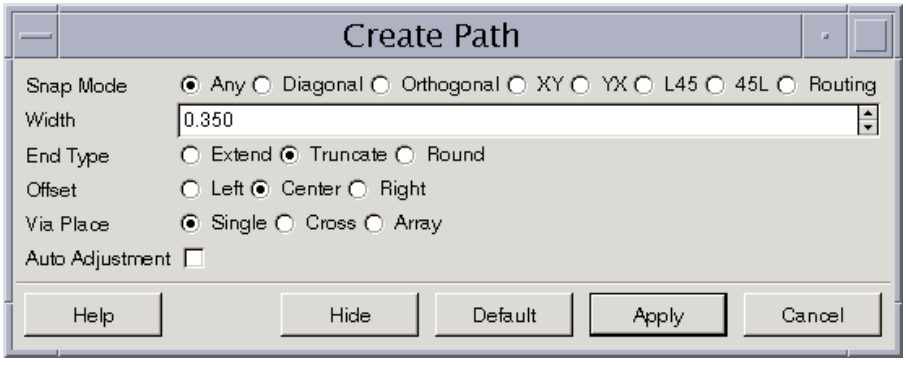
Step 21: Choose “pact.dwg” from Layer palette.

Step 22: LMB click Create->Rectangle or click icon to draw a rectangle from (1.3, 1.3) to (3.9, 3.1).

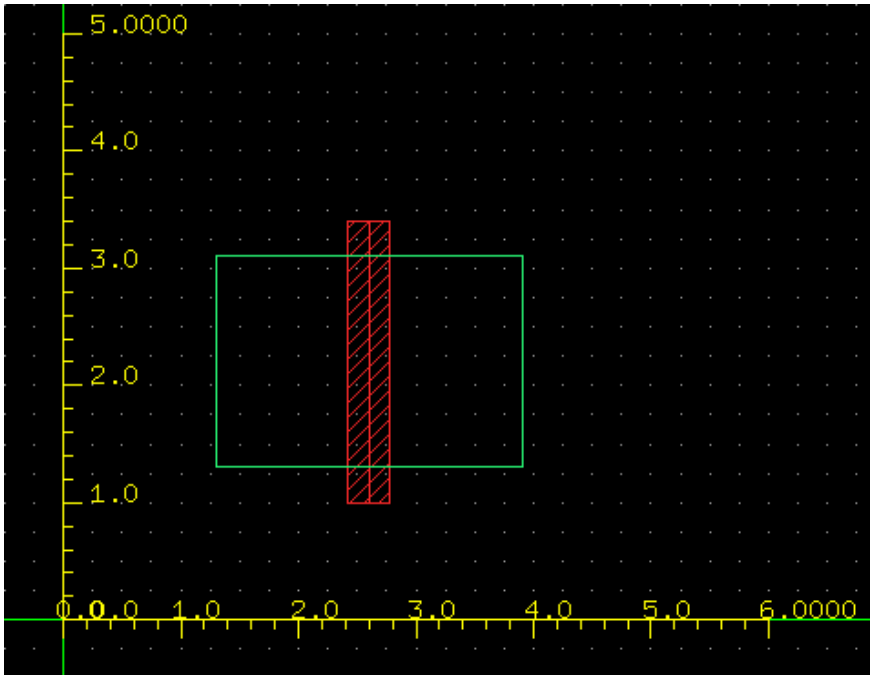
Step 23: Choose “gpo.dwg” from Layer palette.

Step 24: LMB click Create->Path or click icon, a Create Path form appears. (If the form doesn’t appear, please press F3 to wake it up. Or, turn on the option Options->Generic->Editor->Auto Popup Form)

Step 25: Fill out the form with the information as shown in the figure below.

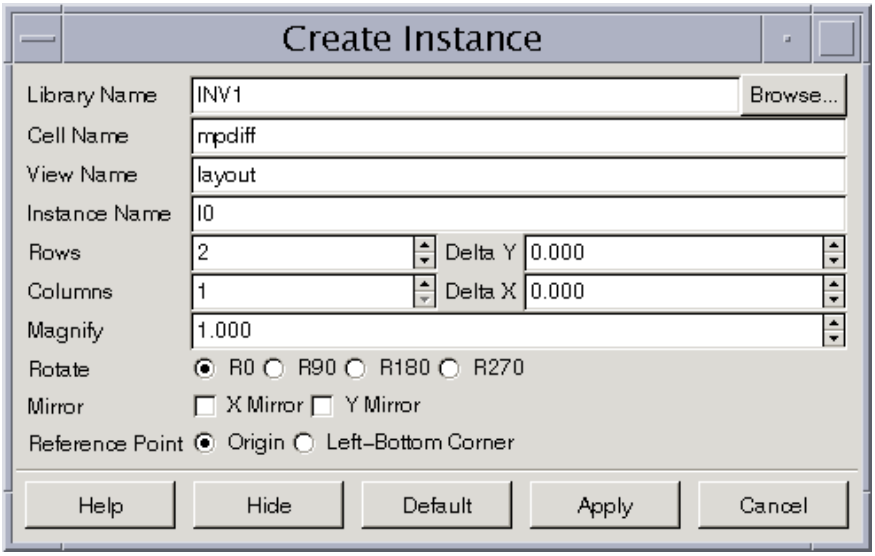


Step 26: Draw a path from (2.6, 3.4) to (2.6, 1.0) with “gpo.dwg”. Double click to terminate this command.



Step 27: LMB click Create->Instance or click  icon, Create Instance form appears.

Step 28: Fill out this form as shown in the figure below.



Step 29: Move the cursor to layout editing area.

Step 30: An instance array (2 instances) is attached to the cursor. LMB click on point (1.3, 1.3) to place these instances.

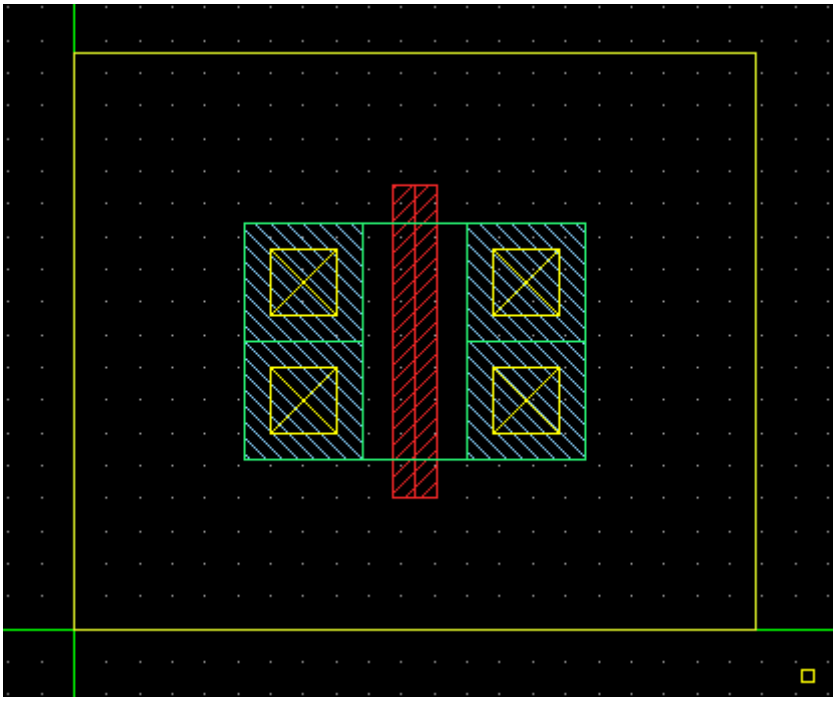
Step 31: LMB click on point (3.0, 1.3) to place these instances again.

Step 32: Press “Esc” to terminate the command.

Step 33: Create a rectangle from (0, 0) to (5.2, 4.4) with “nwell.dwg”.

Step 34: Clear the ruler.

Step 35: The cellview “pmos_18” is shown as below.



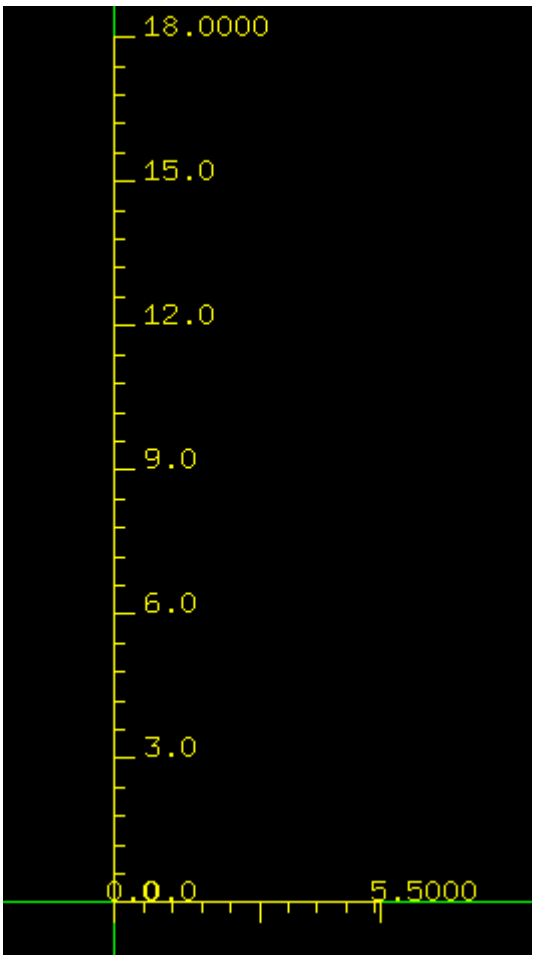
Step 36: Save and close the cellview.

For time saving, we will call the other cellviews from library “INV” to finish the “inv” cell.

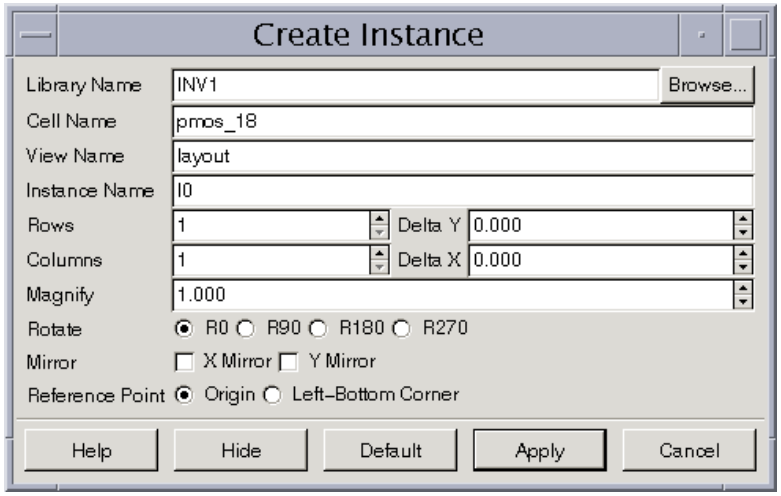
Step 37: In ZDMW, RMB click the cell “inv” in library INV1 and select “New View” submenu.

Step 38: Create layout view for cell “inv”.

Step 39: Draw ruler as figure below.



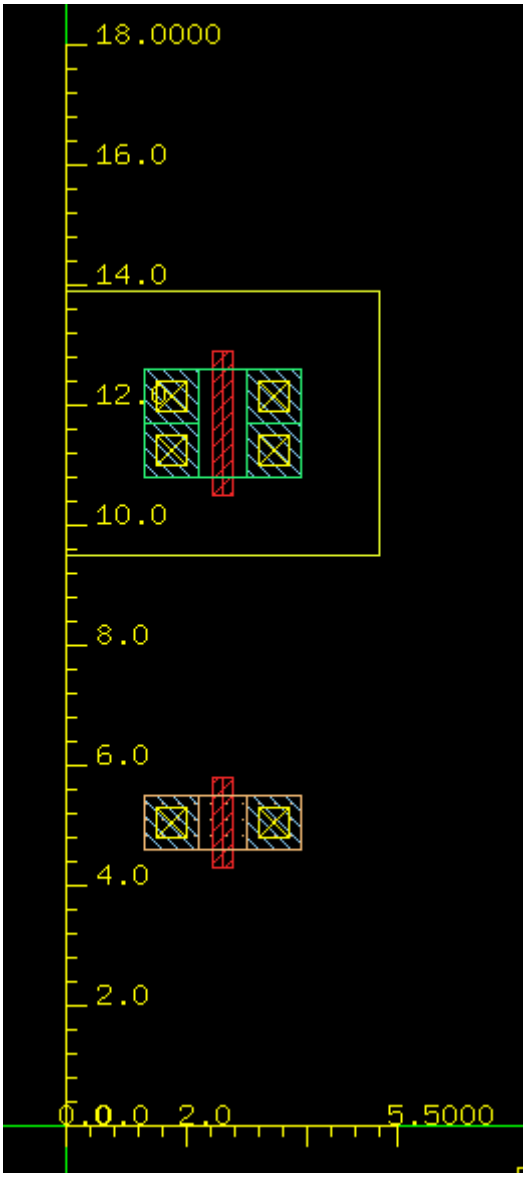
Step 40: LMB click  icon to add instance of cell “pmos_18”.



Step 41: Move the cursor to layout editing area and LMB click on point (0, 9.5) to place the instance of cell “pmos_18”.

Step 42: As Step 40:, add instance of cell “INV.nmos_09.layout”.

Step 43: LMB click on point (1.3, 4.3) to place the instance of cell “noms_09”.



Step 44: Choose “gpo.dwg” and create a path from (2.6, 10.5) to (2.6, 5.8) with 0.35um width.

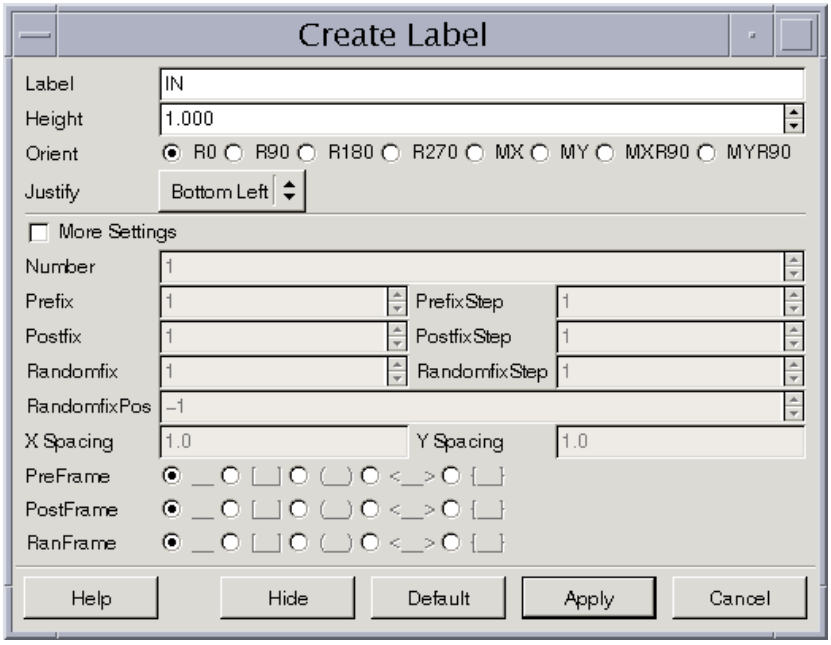
Step 45: Choose “met1.dwg” and create a path from (3.45, 10.8) to (3.45, 5.5) with 0.35um width.

Step 46: Create instance of cell “mpoly” from library “INV” and click on point (1.7, 7.65) to put it.

Step 47: Create instance of cell “m1m2” from library “INV” and click on points (1.7, 6.75) and (3.0, 7.65) to put them.

Step 48: LMB click Create->Label or click  icon. The Create Label form appears.

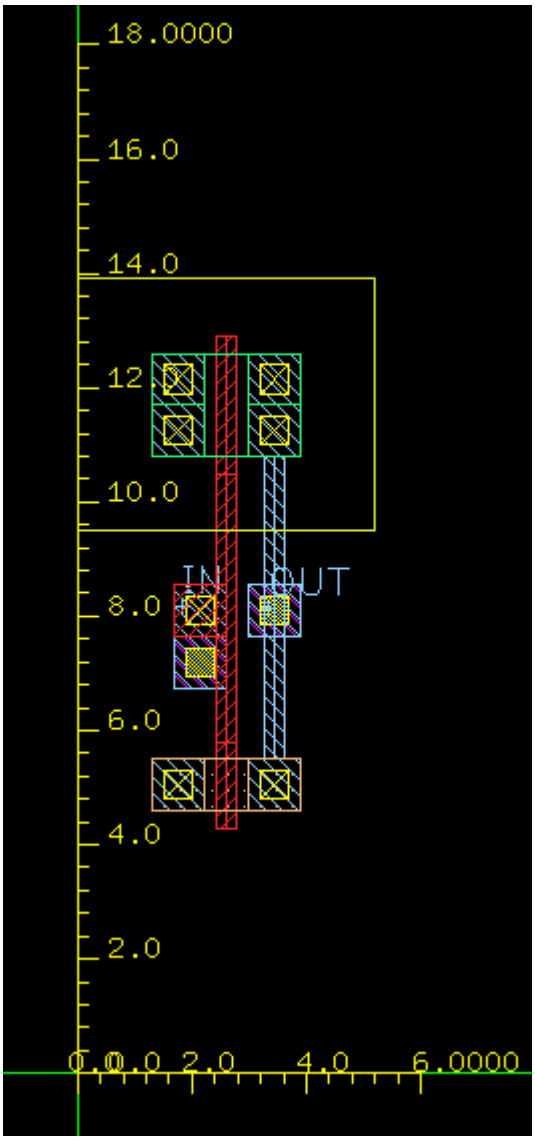
Step 49: Fill out the form as figure below.



Step 50: Move the cursor to the layout editing area, the label “IN” is attached to the cursor.

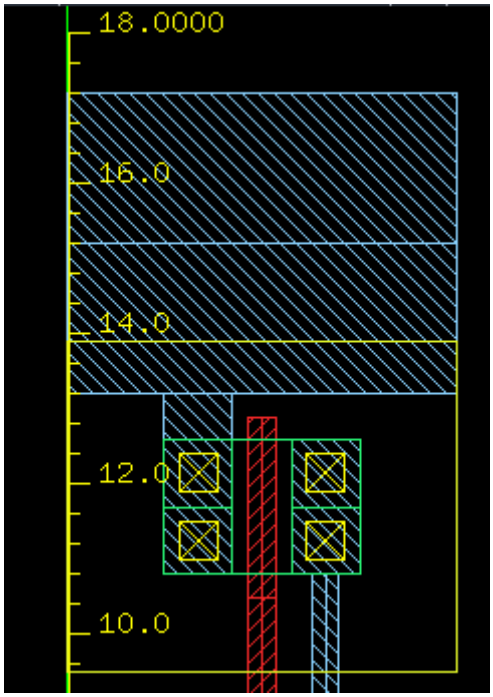
Step 51: Click on a point inside of the instance of cell “mploy” to make sure the symbol “+” is in instance of cell “mploy” area.

Step 52: In the same way, add label “OUT” inside of the instance of cell “m1m2”.



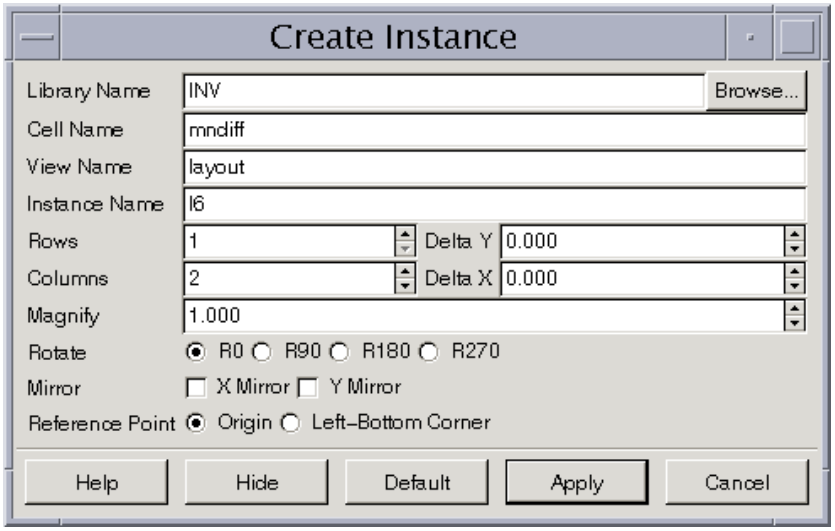
Step 53: Choose “met1.dwg” and create a path with 4um width , and start from (0,15.2) to (5.2, 15.2).

Step 54: Choose “met1.dwg” again and create a rectangle from left bottom point (1.3, 12.6) to right top point (2.2, 13.2). The figure is shown as below.



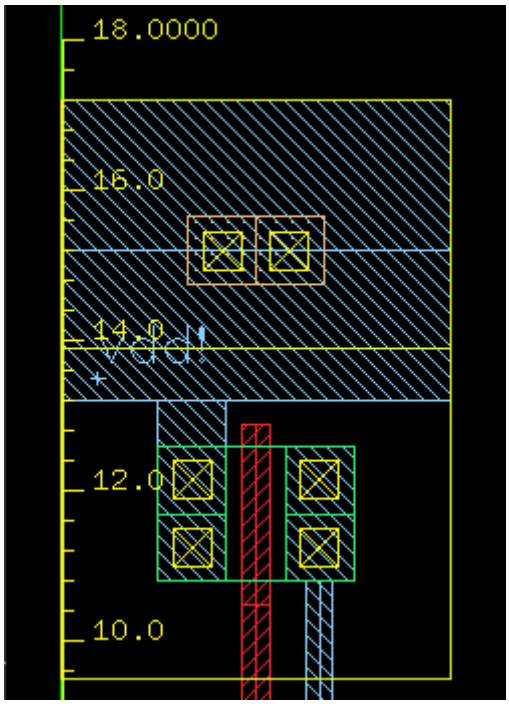
Step 55: Choose “nwell.dwg” and create a rectangle from (0,13.9) to (5.2, 17.2).

Step 56: Add instance of cell “mndiff” from library INV with one row and two columns.



Step 57: LMB click Hide and put the left bottom of the instance array to point (1.7, 14.75).

Step 58: Choose “met1.dwg” and create label “vdd!” that font height is 1.0 in “met1.dwg” area.



Step 59: Move cursor to “met1.dwg” area and press “space” key in your keyboard, the current valid layer will be changed to “met1.dwg” automatically.

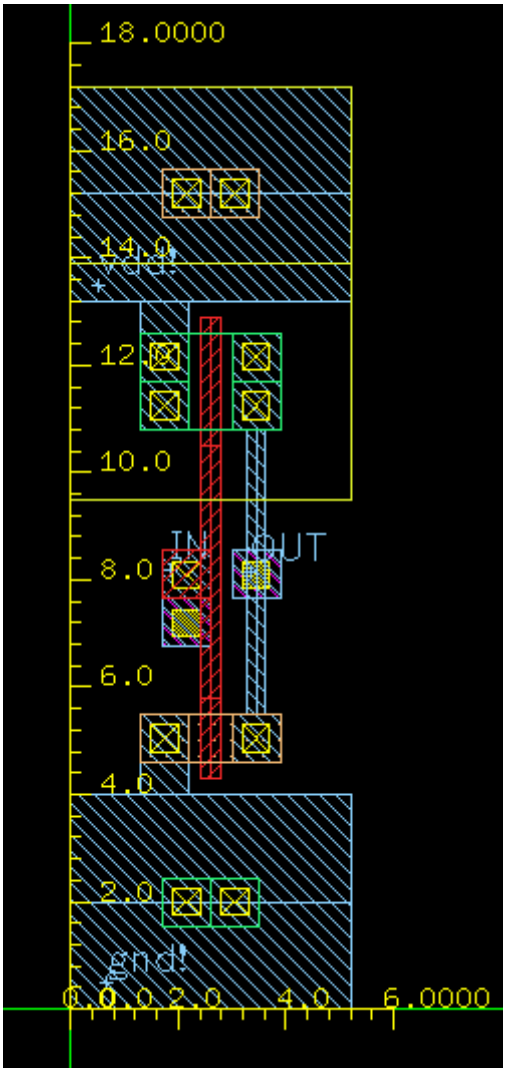
Step 60: In “met1.dwg”, create a path with 4um width, and start from (0,2) to (5.2, 2).

Step 61: In “met1.dwg” again, create a rectangle from left bottom point (1.3, 4) to right top point (2.2, 4.6).

Step 62: Add instance array of cell “mpdiff” from library INV1 and put the left bottom of instance array to point (1.7, 1.55).

Step 63: Choose “met1.dwg” and create label “gnd!” whose font height is 1.0 in “met1.dwg” area.

Step 64: Ok, the whole layout cellview “inv” is created as figure below.



Step 65: Finally, clear the ruler with clicking  icon.

In this section, we learned how to create path, rectangle and instance in order to design a layout cellview of inverter. Hope you've got it.

5.5. Layout DRC

Design Rule Checks (DRC) is run on a layout to check for spacing and construction violations that may have been introduced during layout. The Zeni DRC tool can be run interactively or in batch mode with library database and GDS format.

In this section…

We'll run Zeni DRC on a purposely flawed layout and investigate the tools available to find, display, and manage layout errors found by the tool.

5.5.1. Start software (Jump if you have already started Zeni)

Step 1: cd $WORK-DIR

Step 2: dm &

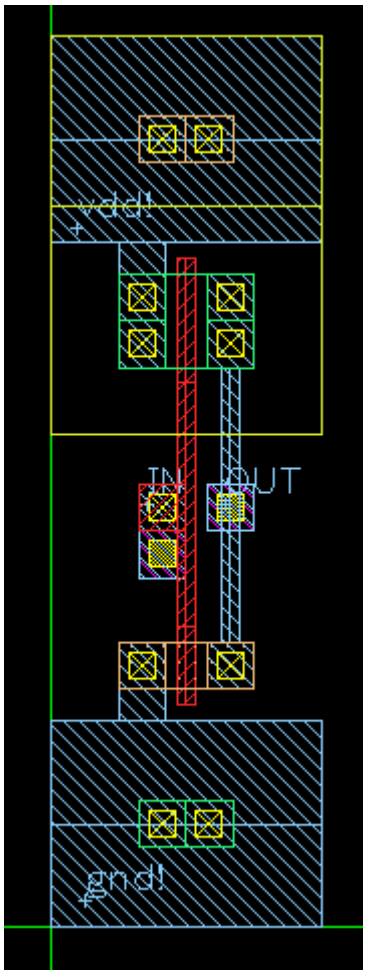
The ZDMW will appear.

5.5.2. Open the design

We'll use a layout of inverter which was created by yourself just now.

Step 3: In the ZDMW, from left to right, LMB click on:
INV1
inv
layout (double-click to open the layout cellview)

The layout cellview of inverter is opened as shown in the figure below.



5.5.3. Run DRC from the User Interface

Step 4: LMB click Options->Generic command, and modify the Verify->Working Path to "/tmp" or using "Browse" to set any directory you want. The figure is shown as below.

Step 5: Save and ok the Options form.

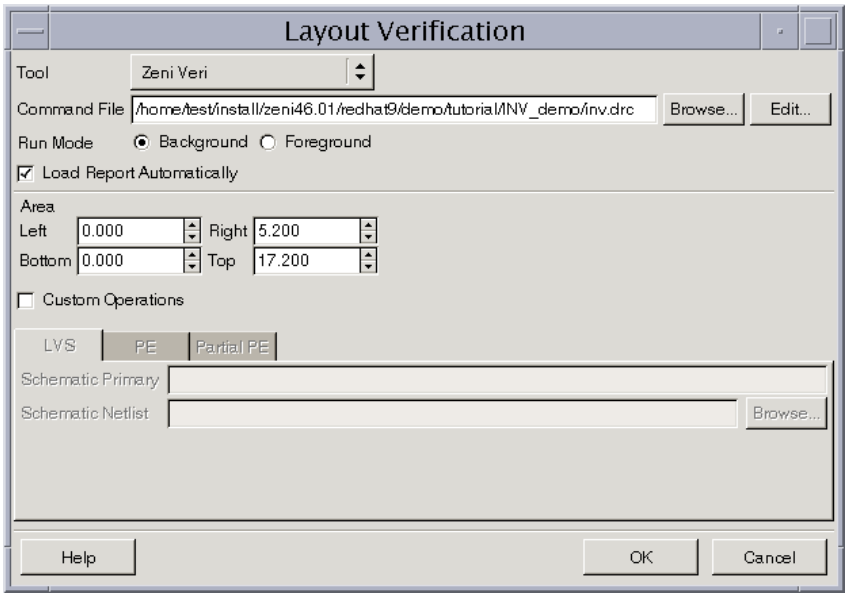
Step 6: LMB click Verify->Layout Verification, a Layout Verification form appears.

Step 7: Select "Zeni Veri" in "Tool" dropdown list.

Step 8: Fill out the Command field with "$Zeni-install-path/demo/tutorial/INV_demo/inv.drc". Here, $Zeni-install-path is Zeni install path.

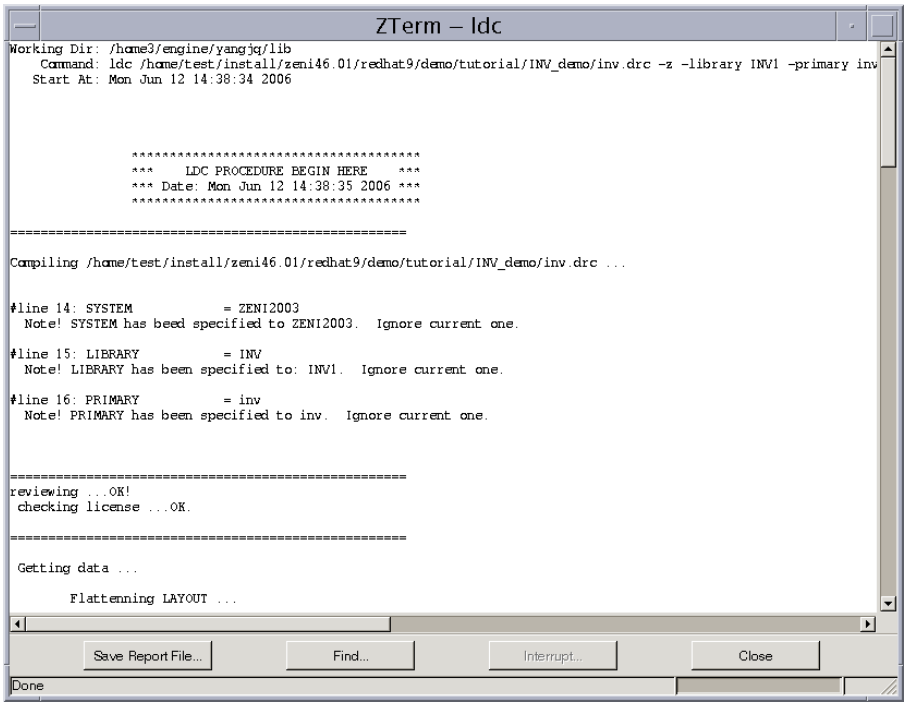
Step 9: You can also click “Edit” button to view the command file.

Step 10: Verify the content of the “Layout Verification” form as below.



Step 11: Click “OK” to start the DRC job. “Background” makes the DRC job running in background process, for larger DRC job, it’s useful for you to do other tasks while a DRC is running. “Load Report Automatically” loads DRC results automatically when the DRC job is finished. If you don’t set the “Load Report Automatically” on, you should load DRC results using “Verify->Load Report” when the DRC job is finished.

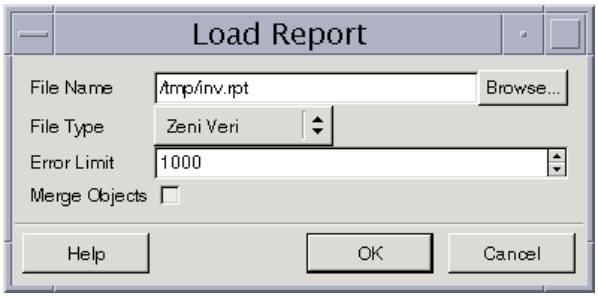
Step 12: The “Zterm-ldc” form appears as below.



Step 13: Close the Zterm.

5.5.4. Examine and Browse the DRC Violation

Step 14: If the “Load Report Automatically” is off in the “Layout Verification” form, click Verify->Load Report to load the DRC violations. Notice the DRC result file “inv.rpt” is saved in the working path that you defined. In this case, the file is saved under directory “/tmp”. Make sure the “File Type” is Zeni Veri and click “OK” to load the DRC violations.

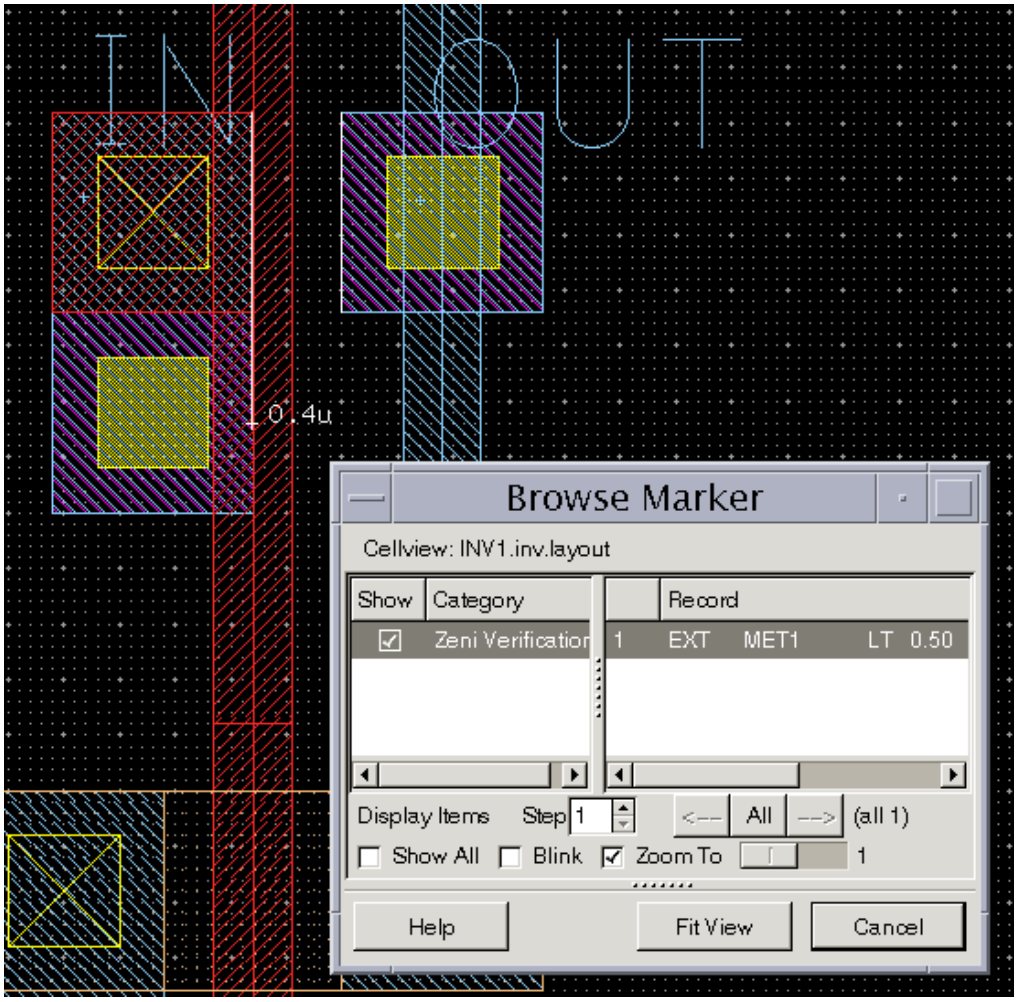


Step 15: LMB click Tools->Browse Marker, the Browse Marker form appears.

Step 16: The form is divided into two halves. The left half of the form lists the verification tools, that is, Zeni can load the DRC result of other EDA verification tools, such as Dracula, Calibre and Hercules. The right hand section lists all DRC violations using verification tool listed in the left side. In this case, there is one DRC violation only.

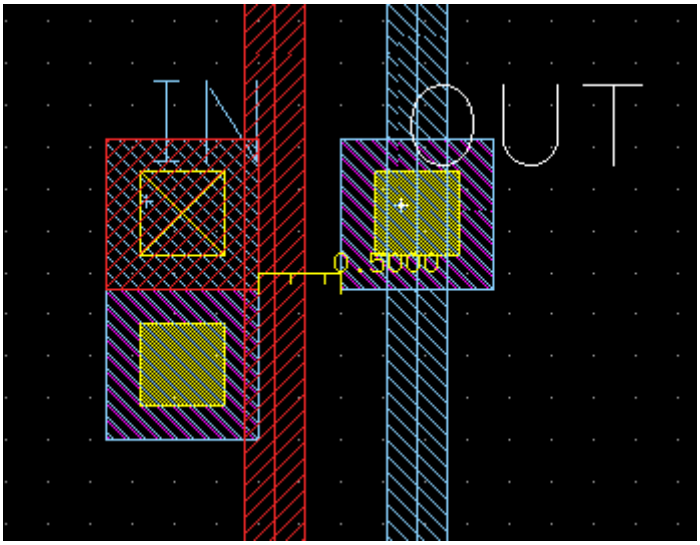
Step 17: It’s better move the “Browse Marker” form to right bottom of your screen so that you can view the whole layout view without overlap.

Step 18: LMB click the DRC violation “EXT MET1 LT 0.50 OUTPUT err9 01”, the related DRC error is displayed in layout editing window with a white marker. The figure is shown as below.



Step 19: This DRC violation means the external spacing of “met1.dwg” layers should be more than 0.50 micron. Now in this layout view, the spacing is 0.4 micron. So you need to enlarge the spacing by moving the instances of cell “mpoly” and “m1m2”.

Step 20: Move the instances of cell “mpoly” and “m1m2” 0.1 micron to the left. The below figure shows the spacing between two layers of “met1.dwg”.



Step 21: Save this cellview and re-run DRC , you can find there is no DRC violations.

In this section, we learned how to use Zeni DRC tool to run DRC check, display and examine layout errors found by the DRC. Hope you've got it.

5.6. Layout LVS

Layout-versus-Schematic verification is used to compare the circuit captured and simulated in the schematic editor with the circuit captured in the layout editor.

Zeni tool performs a circuit extraction on the layout firstly. The layout is examined, and then device structures and the connections between them are identified. From this a netlist representing the circuit found in the layout is generated. A second netlist is generated from the schematic. If there is some differences compared with the netlist generated from layout, Zeni LVS tool will find, display, and manage correction of the errors.

In this section…

We will use interactive LVS to do isomorphic comparison between the layout cellview and schematic cellview of inverter. And use graphic LVS debugger-LDX to find and display the layout error.

5.6.1. Start software (Jump if you have already started Zeni)

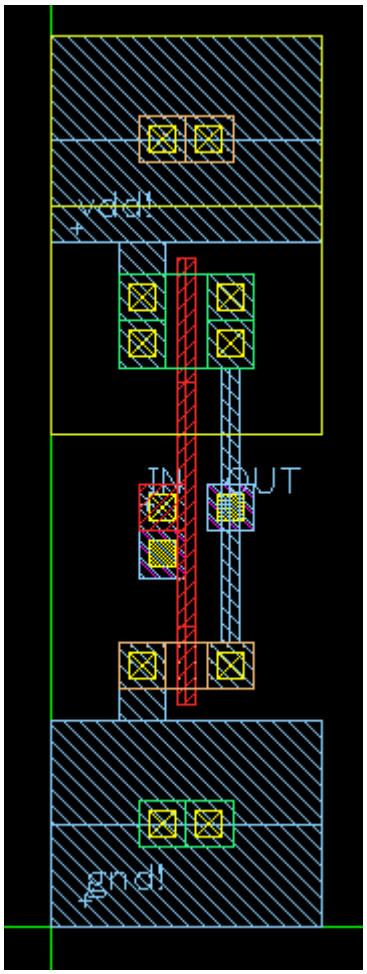
Step 1: cd $WORK-DIR

Step 2: dm &

Step 3: The ZDMW will appear.

Step 4: In the ZDMW, from left to right, LMB click on:
INV1
inv
layout (double-click to open the layout cellview)

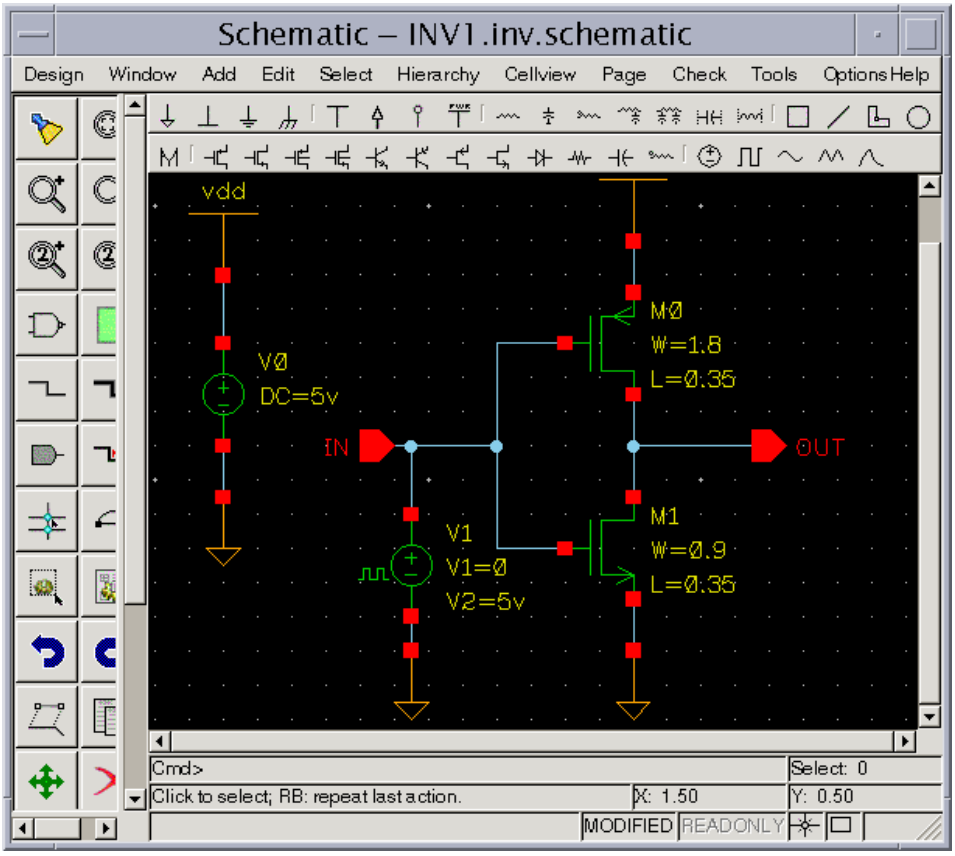
The layout cellview of inverter is opened as shown in the figure below.



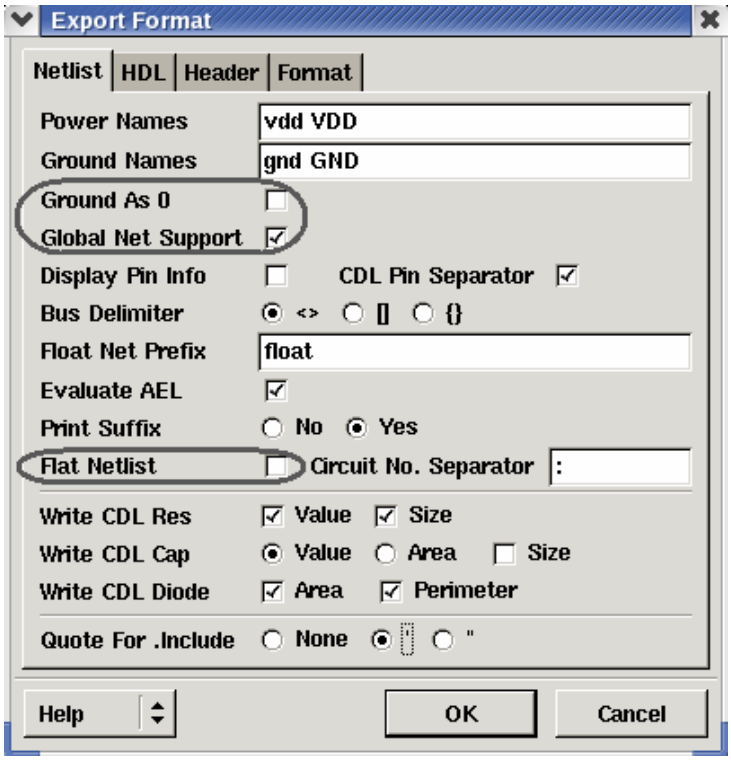
We’ll use this case to compare its layout netlist with its schematic netlist to examine whether they are isomorphic or not.

5.6.2. Export cdl netlist from schematic cellview

Step 5: Open the schematic cellview “INV1.inv.schematic”.

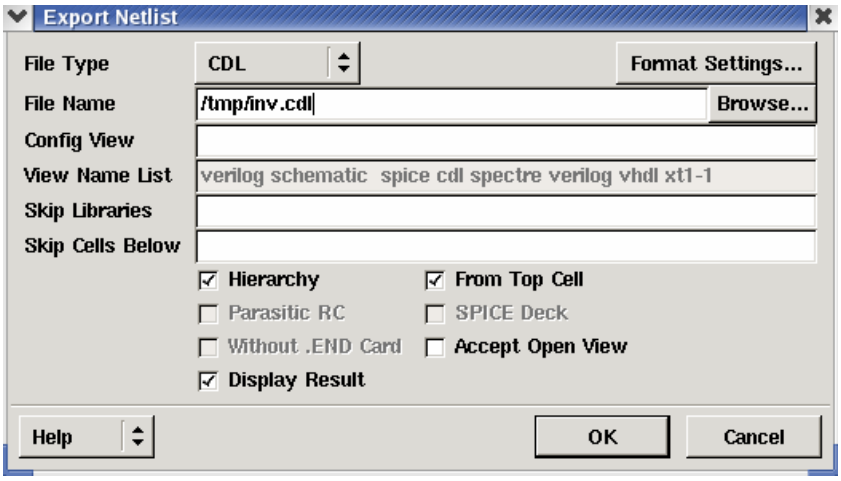


Step 6: LMB click Options->Export Format->Netlist. Turn off the option “Ground As 0”, and turn on “Global Net Support”. Turn off “Flat Netlist”. The figure is shown as below.



Step 7: Ok the form.

Step 8: LMB click Tools->Export Netlist. Fill out the form with the following information as shown in the figure below.

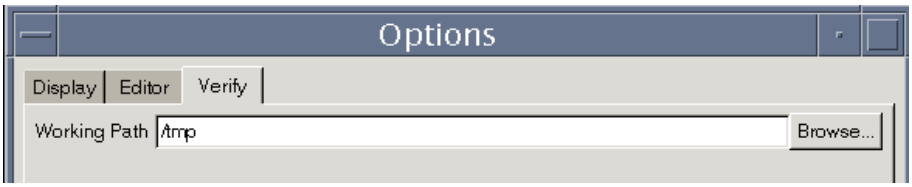


Step 9: Ok the form. The cdl netlist is created in a text editor window. Close this window.

5.6.3. Run LVS from the User Interface

Step 10: Open the cellview "INV1.inv.layout". (Jump this step if this cellview have been opened.)

Step 11: LMB click Options->Generic command, and modify the Verify->Working Path to "/tmp" or using "Browse" to set any directory you want. The figure is shown as below.



Step 12: Ok the Options form.

Step 13: LMB click Verify->Layout Verification, the "Layout Verification" form appears.

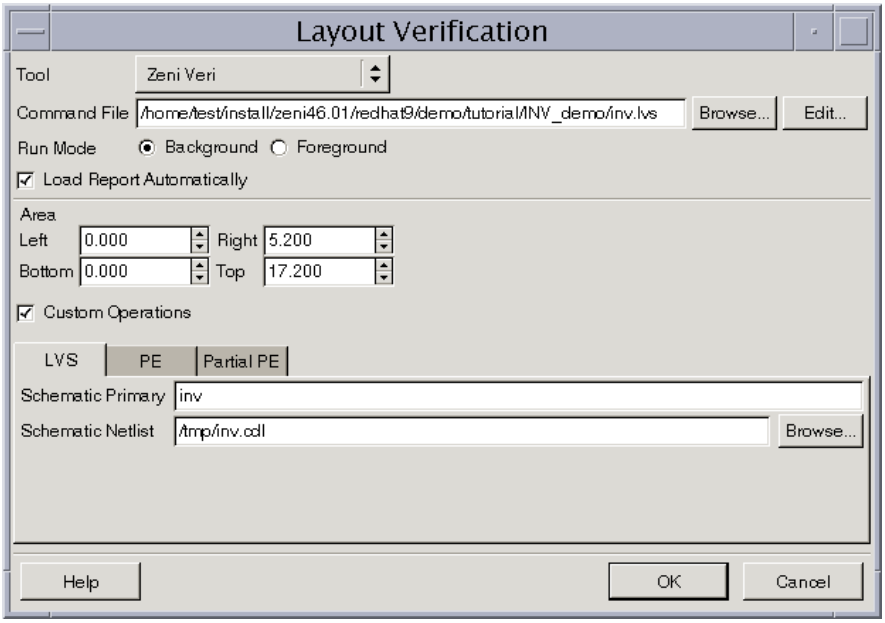
Step 14: Select "Zeni Veri" in "Tool" drop-down menu.

Step 15: Fill out the Command file with "$Zeni-install-path/demo/tutorial/INV_demo/inv.lvs". Here $Zeni-install-path is Zeni install path.

Step 16: You can also click "Edit" button to view and modify the command file.

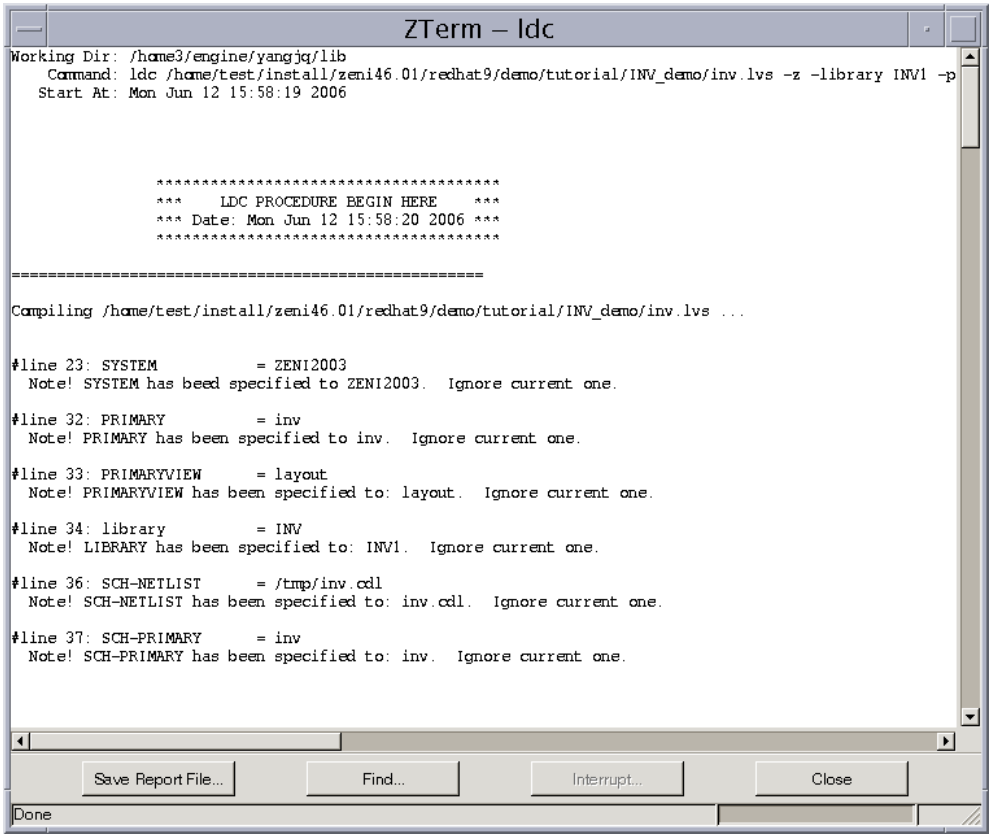
Step 17: Turn on "Custom Operations".

Step 18: Type “inv” and “/tmp/inv.cdl” respectively in LVS card. The figure is shown as below.



Step 19: Ok the form.

Step 20: The “Zterm-ldc” form appears as below.



Step 21: In the end of this form, you can find the text looks like the text below; it means the two netlists are isomorphic.

```
.....................................................

    @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
    @@@@    The two netlists are isomorphic.    @@@@
    @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@


Loading schematic table ... done
Loading layout table ... done
Start parameter or substrate checking ...


    @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
    @@@@     Device substrate is in consistence     @@@
    @@@@      Device subtype is in consistence      @@@
    @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

=================================================


Outputting ...

                  done!

=================================================
```

Step 22: Close this form.

If you have much time, we'll purposely generate an error in following steps. For data safety, we will save a copy of this layout.

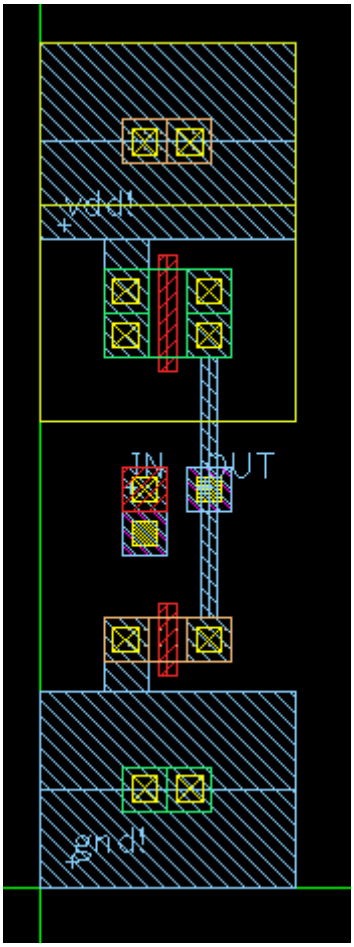
Step 23: Open the cellview "INV1.inv.layout".

Step 24: LMB click Design->Save As.

Step 25: Fill out the form with the following information.
Library: INV1
Cell: invlvsok
View: layout

Step 26: Ok the Save As form.

Step 27: In "INV1.inv.layout" cellview window, click on the layer of "gpo.dwg" and delete it by press Del key. The cellview is shown as below.



Step 28: Save the cellview.

Step 29: Run LVS again as above steps.

Step 30: Close the “Zterm-ldc” window.

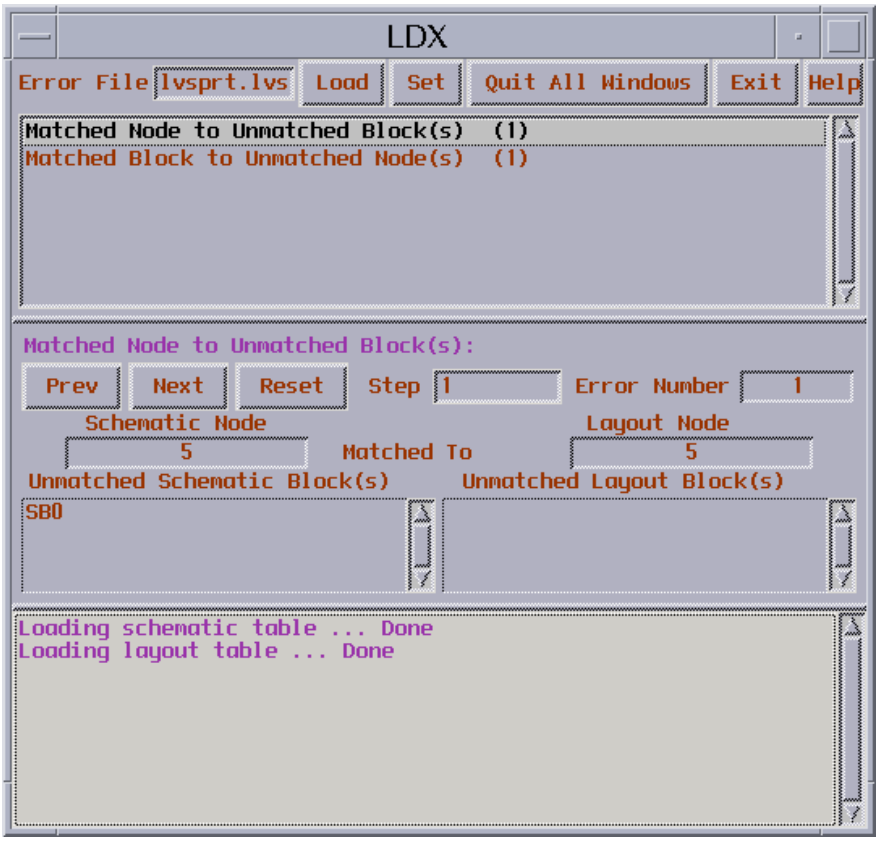
Step 31: LMB click Verify->LVS Debugger. The LDX form appears.

Step 32: Type “lvsprt.lvs” to Error File. Where the prefix of “lvsprt.lvs” is defined in “inv.lvs”file , just is “PRINTFILE = lvsprt”. You can also define it to what you want.

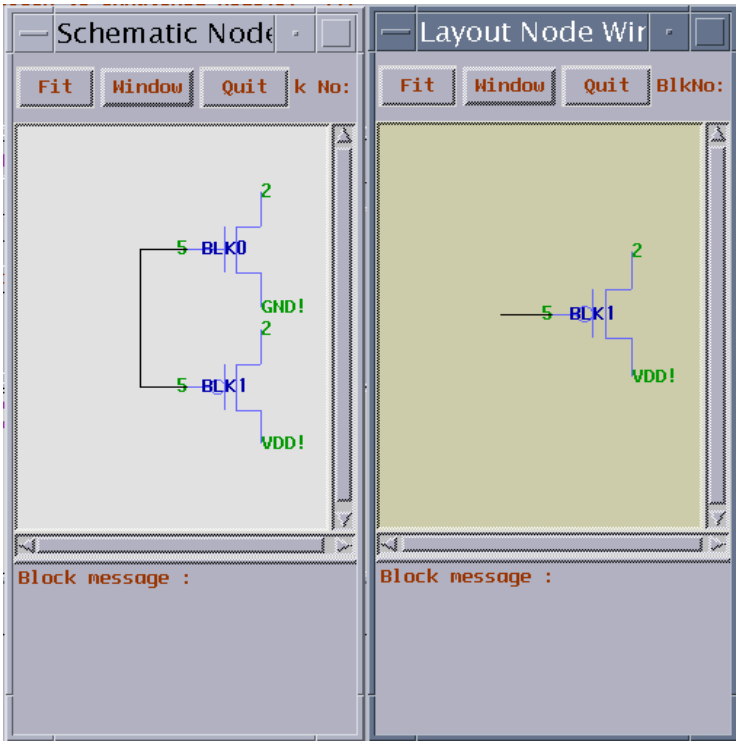
Step 33: LMB click Load button in LDX form.

Step 34: There are 2 unmatched objects.

Step 35: Double click the first item. The related information is displayed as below.

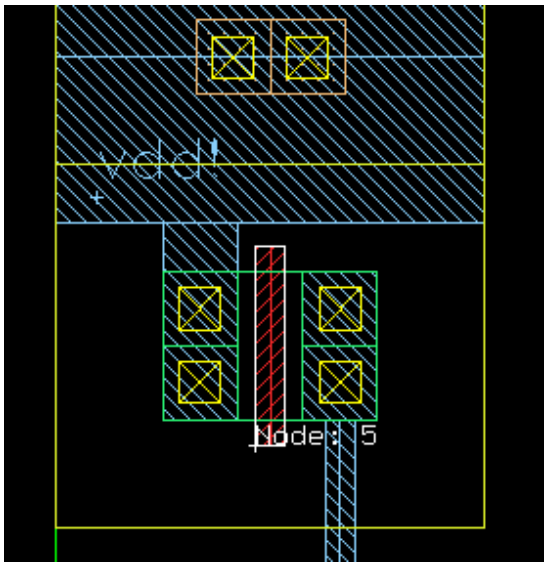


Step 36: Respectively MMB (Middle Mouse Button) click on the node “5” in “Schematic Node” and “Layout Node”, a popup menu appears. Select “Show Node” in this menu, the two schematic topologies that are extracted from schematic cellview and layout cellview are shown as below.

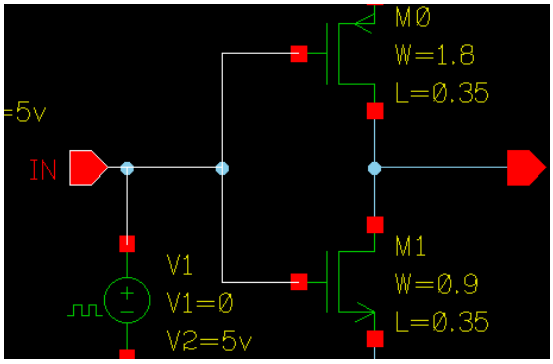


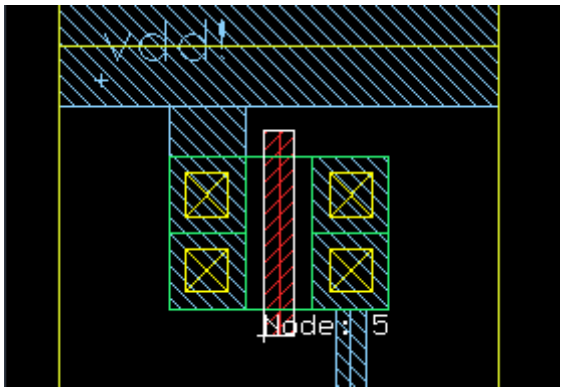
Step 37: The left schematic topology is extracted from the schematic cellview, another is extracted from layout cellview. Comparing both figures, you can clearly see that a NMOS is lost in right figure because a wire which connects two gates of PMOS and NOMS is lost.

Step 38: At the same time, the layout cellview window appears at the top of screen and the error markers are displayed in this cellview.



Step 39: Respectively MMB click on the node “5” in “Schematic Node” and “Layout Node”, select “Show Error” from popped up menu. The schematic cellview and layout cellview are automatically opened, and the error marker is clearly added to each cellview.





In this section, we learned how to run ZeniLVS in interactive mode. With the assistance of LVS debugger- LDX, we clearly found the layout error. Hope you've got it.

6. Command Exercises

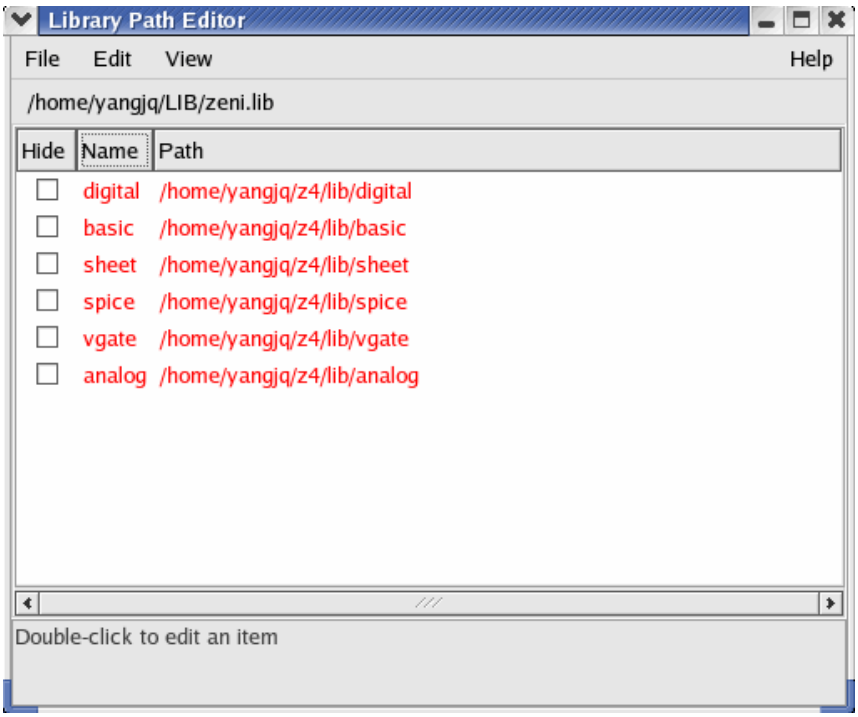
This chapter introduces the usages of most menu commands. These commands are important or are used frequently. After each exercise you will understand the commands deeply.
Currently, commands in the following tools will be introduced.

- Design Manager
- Schematic Editor
- Symbol Editor
- Layout Editor

6.1. Design Manager

6.1.1. Import demo library "INV" (Jump if you have already imported)

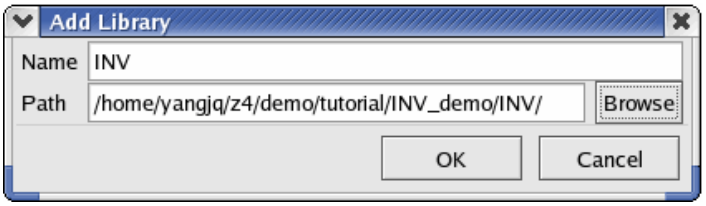
Step 10: In ZDMW, LMB (Left Mouse Button) click *Tools->Library Path Editor*. The Library Path Editor form appears as below.



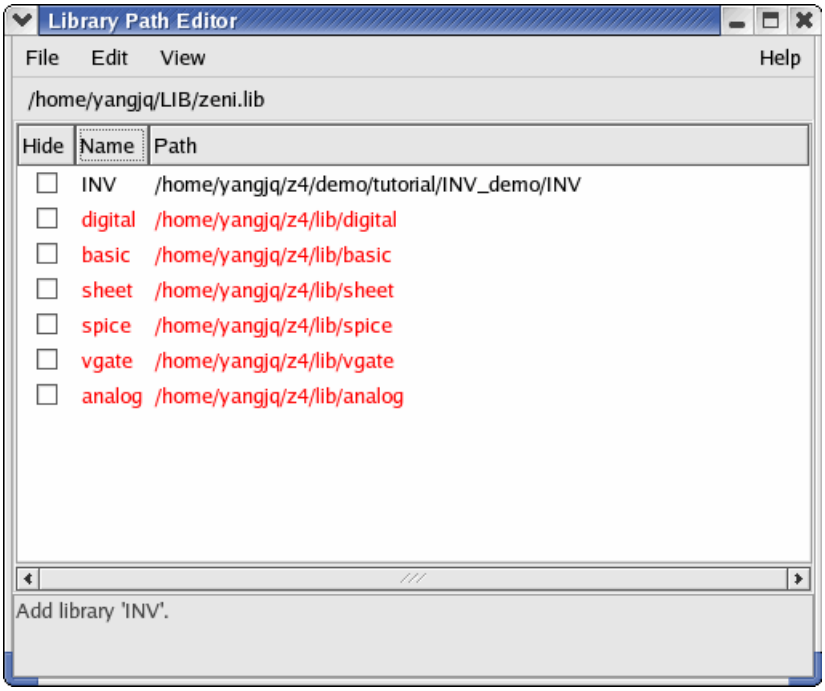
Step 11: In Library Path Editor form, LMB click *Edit->Add*.

Step 12: In Add Library form, click "Browse" to choose INV library under the path $ZENI_INSTALL_PATH/demo/tutorial/INV_demo.

Step 13: Ok the File Selection form. The Add Library form looks like below.



Step 14: Ok the Add Library form. A new library INV is added in Library Path Editor form.

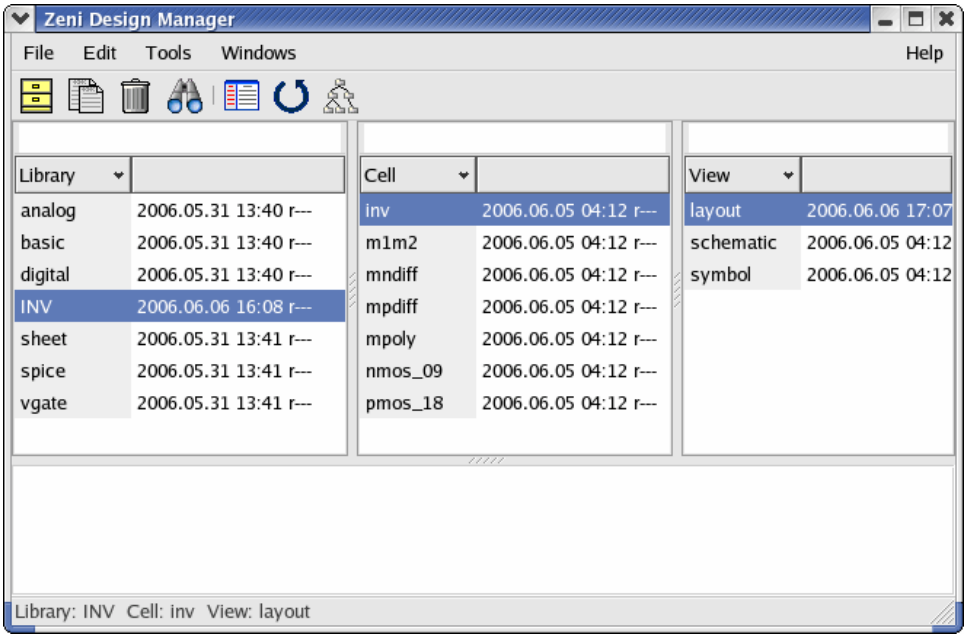


Six library definitions in red are System Library that provided by Zeni tool package. You cannot modify them. Normal Library is in black color.

Step 15: In Library Path Editor, LMB click File->Save.

Step 16: Exit this Library Path Editor by File->Quit.

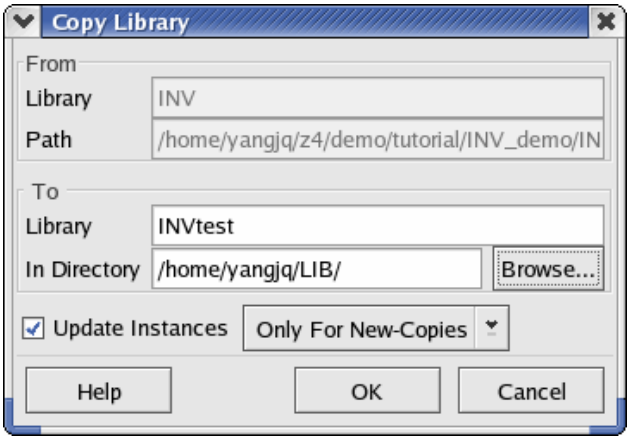
You can find the demo libraries “INV” is displayed in the Library List of ZDMW.



6.1.2. Copy library

Step 1: Select library INV in Design Manager Window, click middle/right button and choose "Copy" command from popup menu.

Step 2: Copy "INV" to "INVtest" in your working directory.



Step 3: Ok the form.

6.1.3. Exercises

Exercise - 1 Difference among "File->Refresh Library", "File->Refresh Library List" and "File->Refresh All"

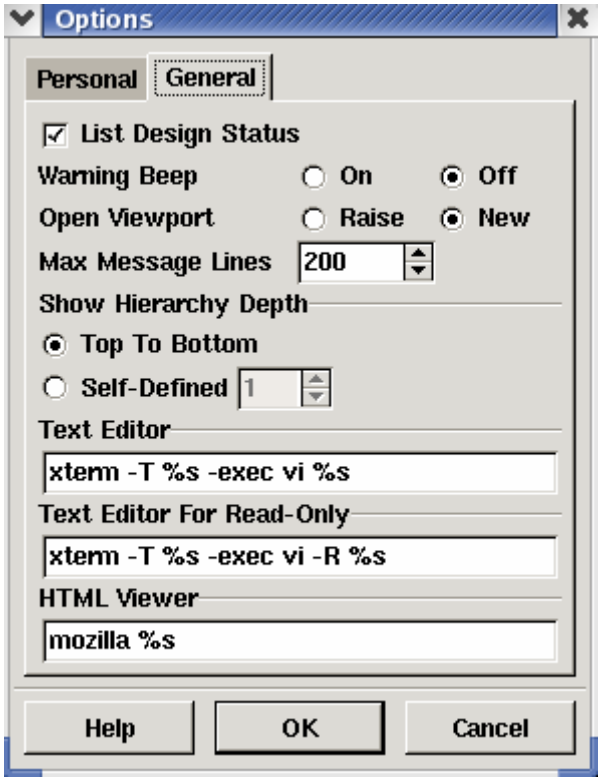
In File menu, there are three commands relate to refresh design. The difference among three commands is as below.

- **Refresh Library** only refreshes selected library.
- **Refresh Library List** reloads all libraries recorded in file “zeni.lib”. It only re-initializes library name, time created and access permit.
- **Refresh All** reloads all libraries recorded in file “zeni.lib”. It re-initializes library list, cell group list, cell list, view list and design hierarchy.

Note:

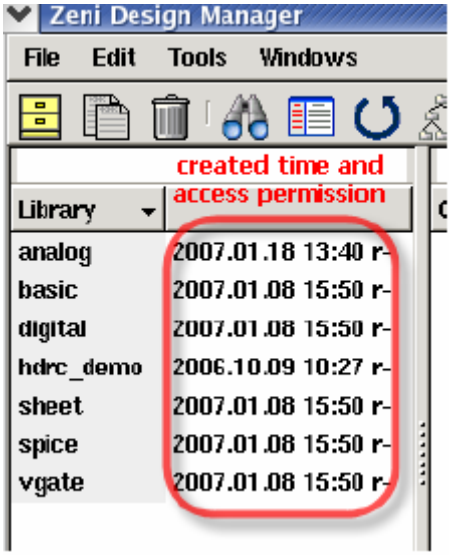
Command *Refresh All* will spend much time to create hierarchy information for all libraries.

Exercise - 2 File->Options->General



This form provides global options during design.

- **List Design Status** The option controls whether "created time and access permission" list is displayed in Design Main Window or not. Not display as default.



- **Warning beeps**
  -***On***: DM beeps when warning or error message appears in Design Manager.
- **Open Viewport**
  - ***Raise***: If you open a view, but the view had been opened, DM will raise the opened window to the top of the screen for convenient working on it.
  -***New:*** DM will open a view in a new window no matter whether the view had been opened.
- **Max Message Line** specifies the maximum number of message displayed in Design Manager Window. These messages include Note, Warning and Error. 0~10000 is valid.
- **Show Hierarchy Depth** specifies the limitation of displaying hierarchy depth in Tools->Show Tree form. Please refer to *Exercise - 6 Tools->Show Tree* on page 68.
  -***Top to Bottom*** displays full hierarchy of design.
  -***Self-Defined*** depends on user-specified depth. 1~100 is valid.
- **HTML Viewer** specifies the viewer to show Zeni online manual. “netscape %s” is default. You can also modify it according to Operation System you used, for example, “mozilla %s”.

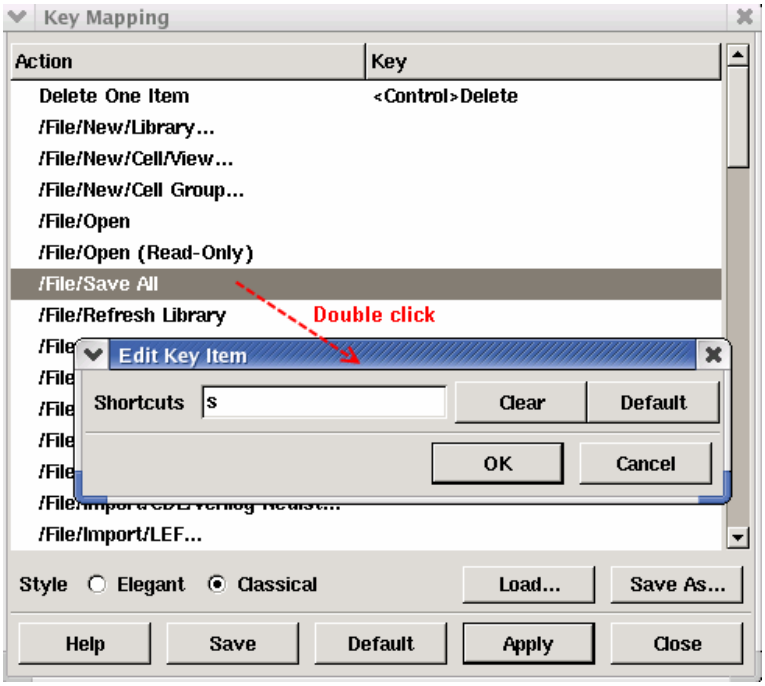
Exercise - 3 File->Key Mapping

This command lets you define the bindkey as you like.

Do the following steps to add bindkey ‘s’ to “Save All” command.

Step 1: In “Key Mapping” form, please turn on Classical option and double click item “/File/Save All ”.

Step 2: In “Edit Key Item” form, press “s” and ok the form.

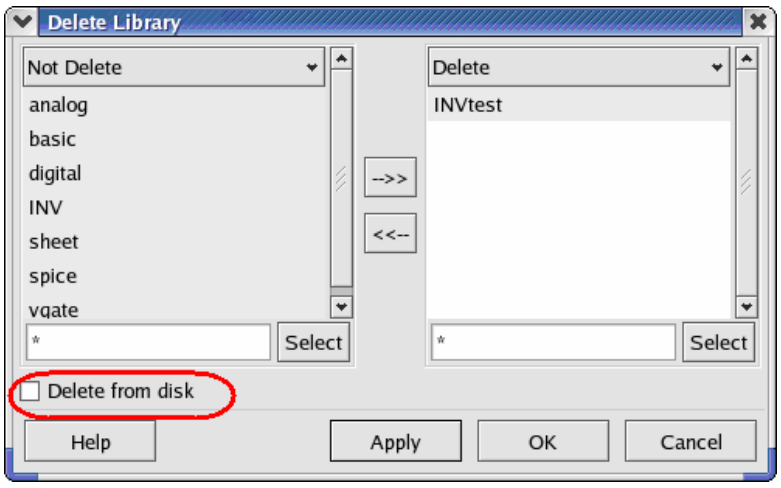


Step 3: Save and close the “Key Mapping” form.

Exercise - 4 Edit->Delete

This command lets you delete an item you selected on Design Manager Window such as library, cell-group, cell and view.

Here, we suppose you selected a library “INVtest”, click right mouse button on the library name, choose “Delete” from popped up form, the Delete Library form appears.



The selected library “INVtest” is automatically added to “Delete” list. Of cause, you can use button -->> to add the libraries you want to delete to “Delete list”.

Note:

At the left bottom of the form, there is an option *Delete from disk.*

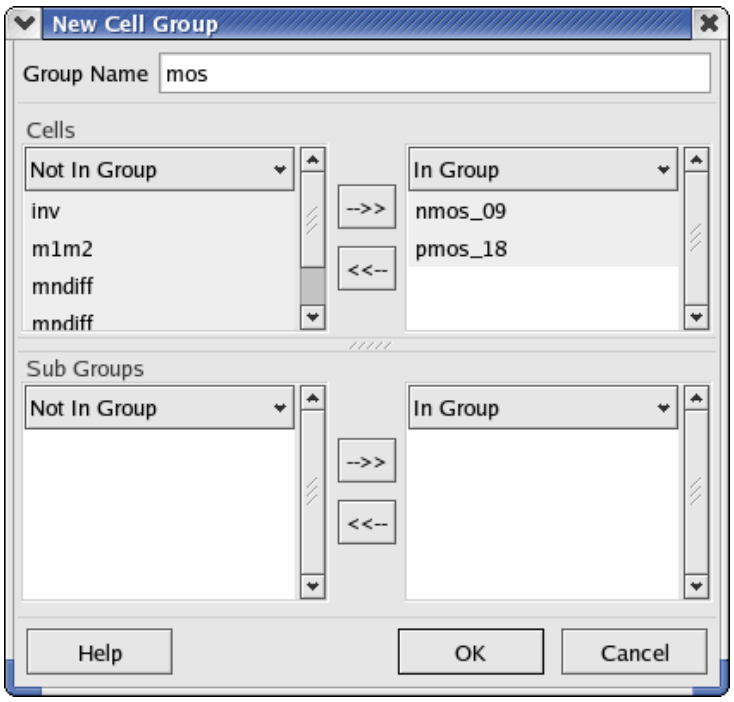
- Turn on this option, remove this library definition from “zeni.lib” and really delete it from disk.
- Turn off this option, only remove this library definition from “zeni.lib”. The library data is still in disk.

Exercise - 5 Edit->Cell Group

This command lets you categorize all of cells as you like. Each cell group can contain sub-group.

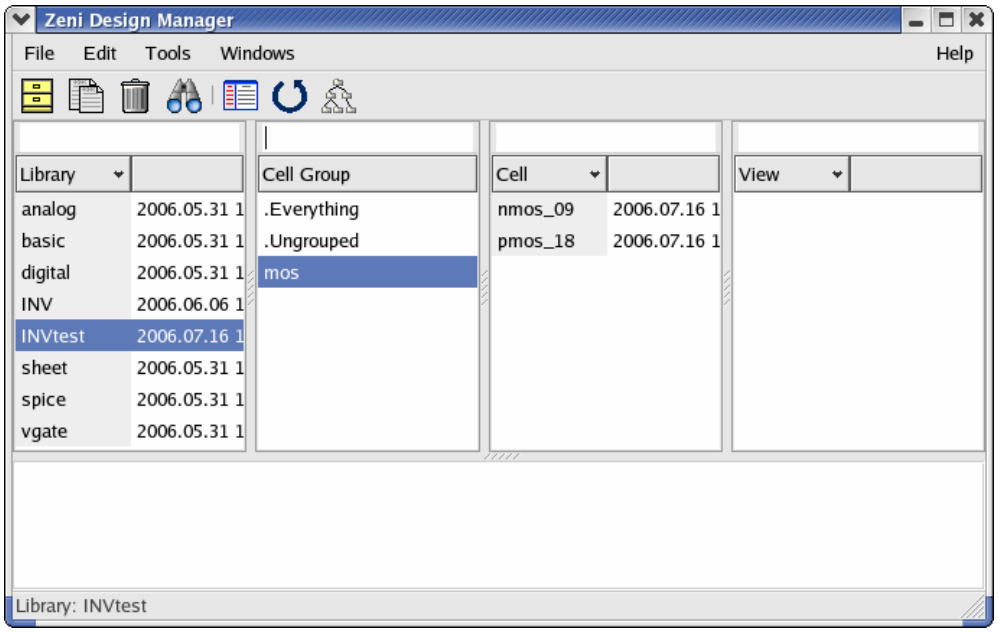
Step 1: Select the library INVtest, click middle/right button and choose “New Cell Group” command from popup menu.

Step 2: In New Cell Group form, fill out the information as the figure shown below.



Step 3: Ok the form.

Step 4: In Design Manager, Cell Group list is shown automatically.



In the cell group list, there are 3 groups ".Everyting", ".Ungrouped" and "mos".

- ".Everything" is a default cell group, it contains all cells of current library.
- ".Ungrouped" is also a default cell group; it only contains those cells that are not in any group.
- "mos" is created by yourself just now.

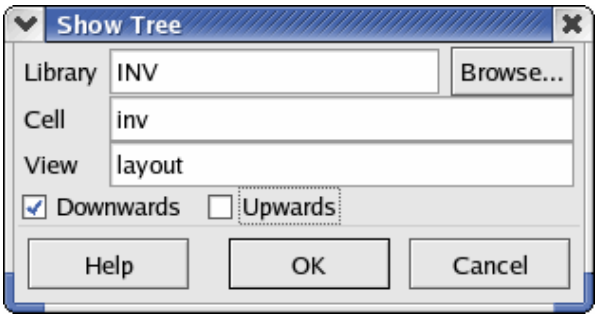
Step 5: Click icon  to hide the Cell Group window.

Exercise - 6 Tools->Show Tree

This command shows the hierarchy information of a cellview.

Step 1: In DMW, from left to right, click INV, inv and layout.

Step 2: Click middle/right button at layout cellview and choose "Show Tree" from popup menu. Show Tree form appears as below.



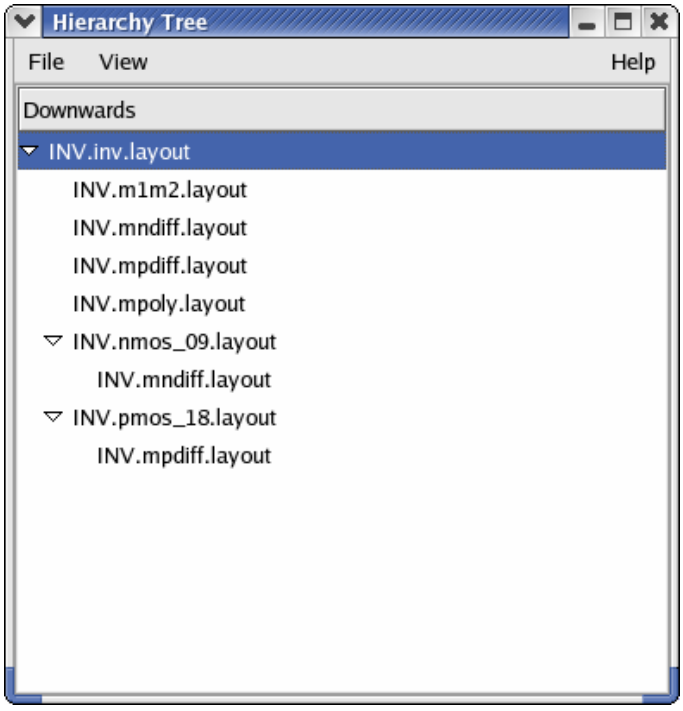
In this form, two important options need to be paid attention to.

- **Downwards** only shows the cellviews that are referenced by current cellview. We called it top-down style.
- **Upwards** only show the cellviews that reference current cellviews. We called it down-up style.
- If you turn on both two options, DM shows both styles.

Note:

If you turn on *Upwards* option, it spends much time to create hierarchy information for all libraries.

Step 3: Ok the form, the hierarchy tree window appears.



In this form, use View->Expand or View->Collapse to open or close the hierarchy tree; use File->Save to store the hierarchy tree information to a text file.

Note:

Showing hierarchy information depth depends on option *File->Options->General->Show Hierarchy Depth.* Please refer to *Exercise - 2 File->Options->General* on page 64.

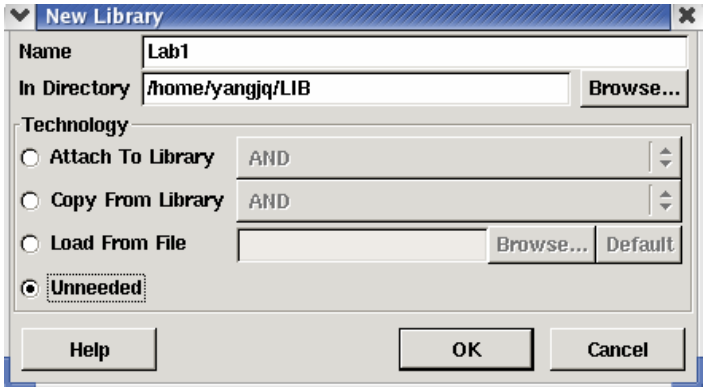
6.2. Schematic Editor

6.2.1. Import demo libraries

Please refer to *6.1.1 Import demo library* on page 61 to import demo libraries “Adder” and “ViewBind” from “$ZENI_INSTALL_PATH/demo/SE” to Design Mananger window.

6.2.2. Create a new library

Step 1: Select File->New->Library to create a new library “Lab1” in your working directory.

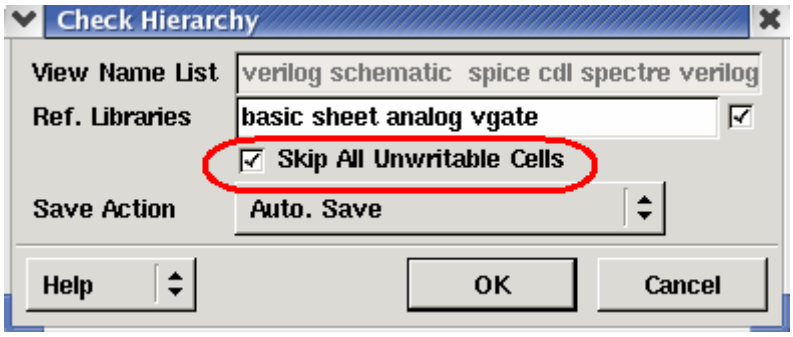


Step 2: Ok the form.

6.2.3. Basic Exercises

Exercise -7 Difference among "Design->Check and Save", "Check->Current Cellview" and "Check->Hierarchy"

- **Design->Check and Save** checks and saves current cellview.
- **Check->Current Cellview** only checks current cellview.
- **Check->Hierarchy** checks all schematic type cellviews listed in *View Name List* in the hierarchy. About View Name List, please refer to *Exercise - 28 Options->Editor* on page 91.



The option "Skip All Unwritable Cells" used to un-check the cells which you have no write permission.

If the circuit has error, warning and error markers would be shown in the corresponding position. You can use *Check->Find Marker/Delete Marker/Delete Markers* to find each marker or delete them.

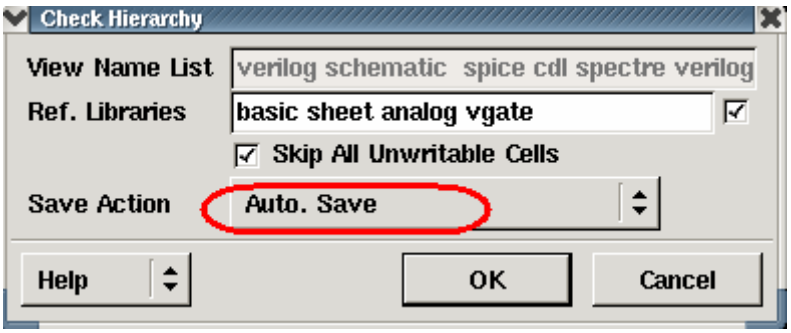
Note:

For "save" the schematic operation, system will create a file "schematic.dat.timestamp" automatically under the path "library_name/cell_name/schematic_name/current" This file only contains a number. Notice that any modification after you "save" one schematic, even you move one line, the number in "schematic.dat.timestamp" will be changed after the second "save" operation; but if you don't do any modification between two "save" operations, the number in the file will not be changed.

For "check" the schematic operation, system will also create a file "netlist.dat.timestamp" automatically under the path "library_name/cell_name/schematic_name/current". This file contains a number too.

To export netlist after checking, the two numbers in "schematic.dat.timestamp" and "netlist.dat.timestamp" must be the same. Otherwise, netlist can't be exported. If "save" and "check "at the same time, or there is no modification on the schematic between the two operations "save" and "check", two numbers in the two files will be the same. Then system exports netlist

successfully. So, you'd better use "check and save" command to check and save current cellview. For hierarchical check, it is simple because “Check Hierarchy” form has an option *"Save Action”*, you can set it to *“Auto.Save”* in order to automatically save full hierarchy cellviews while checking. “Check Hierarchy” form is shown as figure below.



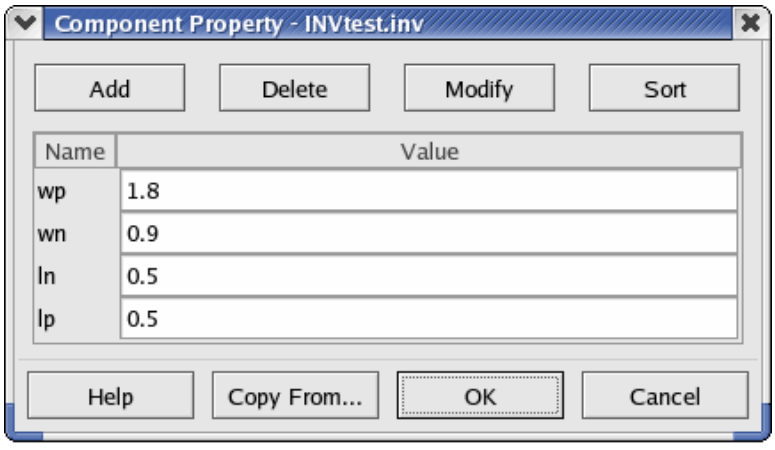
Exercise - 8 Design->Component Property

This command lets you add parameters to this cell as component parameters. When this cell is referenced by another cell, you can modify these parameters value according to different instantiation. A cell should have and only a set of component parameters. So these parameters will be inherited by another schematic cellviews and symbol cellviews of this cell.

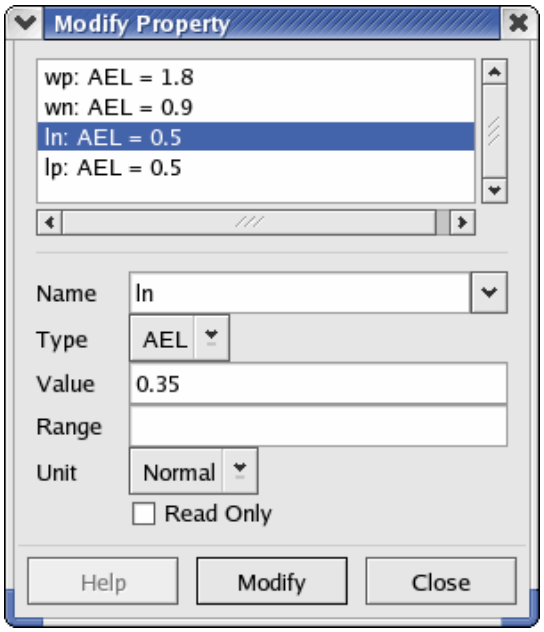
Do the following steps you can understand it more details.

Step 1: Open the “INVtest.inv.schematic” cellview.

Step 2: Select Design->Component Property. In Component Property form, there are 4 parameters with default value. .



Step 3: Click Modify button to modify ln and lp to 0.35.



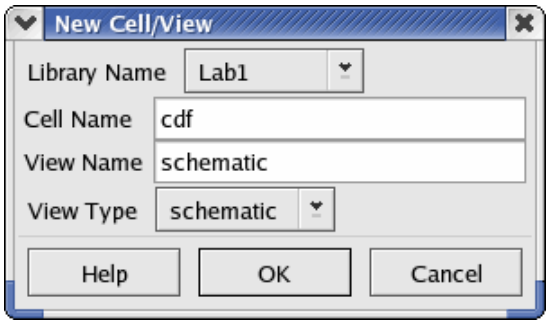
Step 4: Close the Modify Property form.

Step 5: Ok the Component Property form.

Step 6: In Design Manager window, select library “Lab1”.

Step 7: Click middle/right mouse button and choose “New Cell/View” command from popup-menu.

Step 8: Fill out the information in New Cell/View form as the figure below.



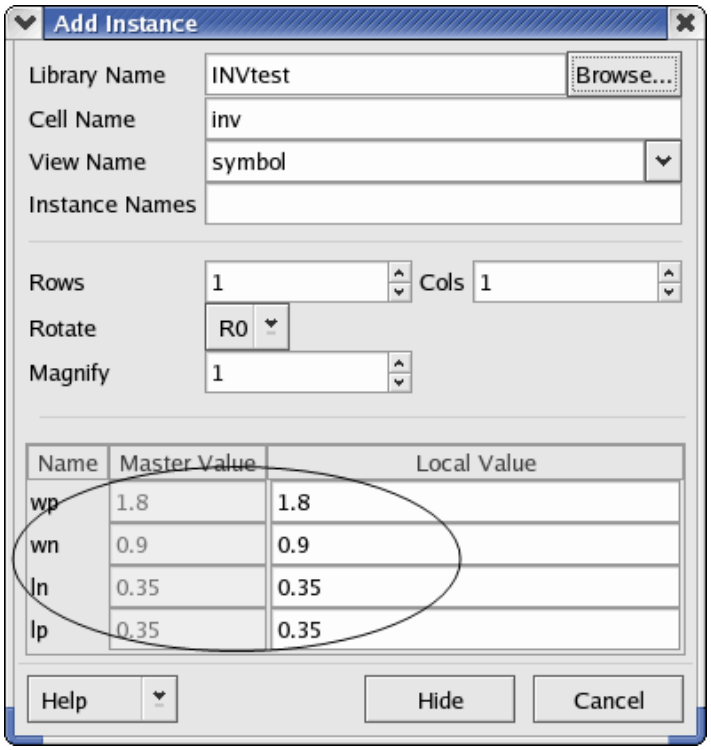
Step 9: Ok the form.

Step 10: In “Lab1.cdf.schematic” editor window, select Add->Instance or click icon  .

Step 11: In Add Instance form, use Browser to select “INVtest.inv.symbol”.

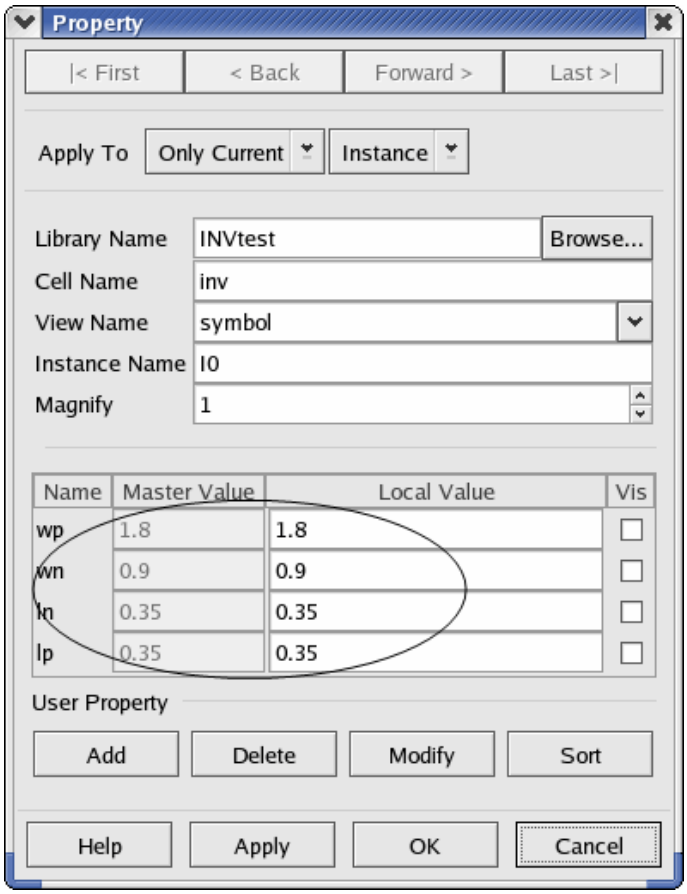
Step 12: Add Instance form pops up as below. The ellipse line marks Component Property (parameters) of cell “inv”. Of cause, you can modify these parameters value with Local

Value in this form.



Step 13: Move the cursor back into the Schematic Editor window, right-click mouse to rotate the instance outline. Left-Click mouse anywhere in the schematic editor window to place this instance.

Step 14: After placing this instance, select Edit->Property or click icon  to see this instance property. You can also modify the component parameters value in Local Value field

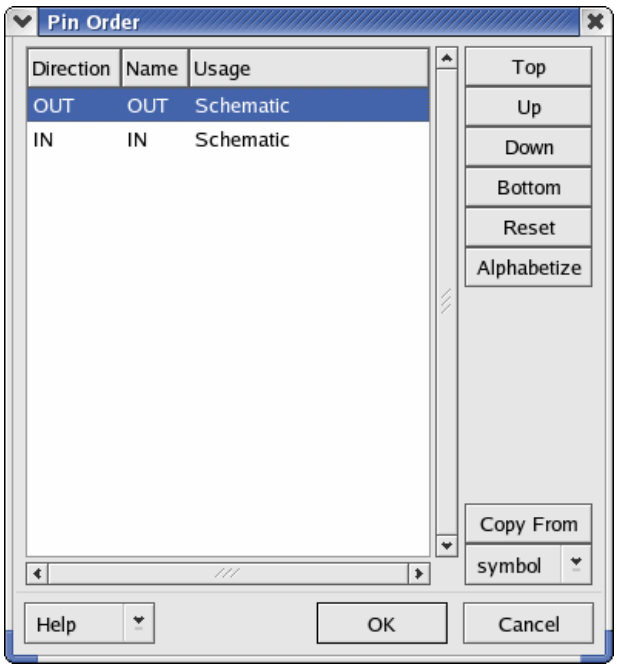


Step 15: Click icon  to save the cellview.

Step 16: Close the cellview.

Exercise - 9 Design->Pin Order

Pin order form lists the order of pin. The order depends on the sequence of pin creation. You can modify the order by Top, Up, Reset, etc.



Note:

The pin order of schematic cellview should be the same as Symbol Editor's, otherwise, Zeni will give an error message when you check schematic cellview or symbol cellview.

Exercise - 10 Design->Discards Edit

This command lets you discard all modifications since the last time it was saved.

Note:

The commands ***Undo*** and ***Redo*** will not work after this command.

Exercise - 11 Design->Reload

This command lets you reload schematic data from disk and refresh current schematic view. It is useful for a project team to synchronize design data.

Exercise - 12 Window->Birds-eye View

This command shows the current whole view in a mini window. The bindkey "1" is default.

Exercise - 13 Add->Wire/Wide Wire

This command lets you add a wire to schematic view.

Step 1: Open the cellview "INVtest.inv.schematic", click icon or to start a wire. If "Add Wire" form doesn't appear, please press "F3" to popup it. While moving cursor, you may find the nearest pin to the cursor will have a small yellow diamond, press

"Ctrl+left-click mouse" or "Shift+left-click mouse" will snap route the current wire to that point.

Step 2: Close without saving.

Note:

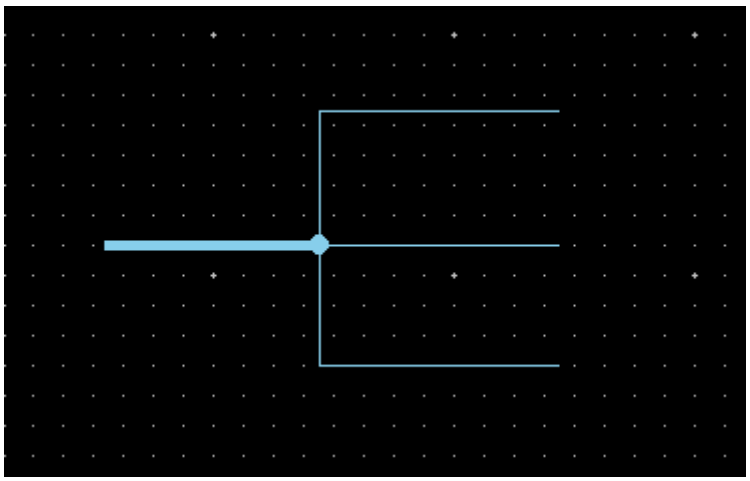
1. In *Options->Editor* menu, (please refer to *Exercise - 28 Options->Editor* on page 91*)* turn on the "Always" in "Popup option form" option, the command form will be popped up without pressing "F3" key.
2. In *Options->Editor* menu, turn off the "Wire Snapping" option, the small yellow diamond will not appear.
3. While creating a wire, you can right-click mouse to change the wire mode. If the "Add Wire" form still exists, the value of Draw Mode will be changed with each right-click mouse.

Exercise - 14 Add->Wire Name

This command lets you name a wire. You can give multiple names separated by space. You can also use a bus expression to name an array.

Step 1: Create a new cellview "Lab1.wirename.schematic".

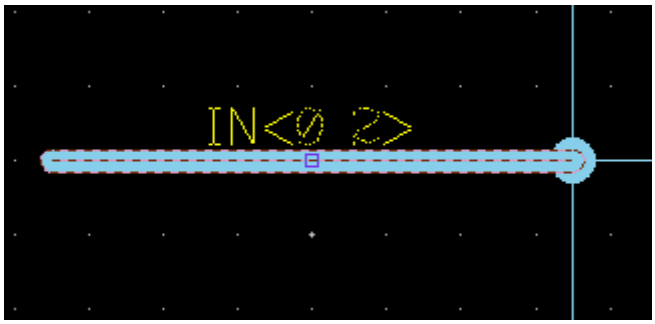
Step 2: In cellview "Lab1.wirename.schematic" window, click icon or to create wires as the figure below.



Step 3: Click icon , fill out the Add Wire Name form with the following information.

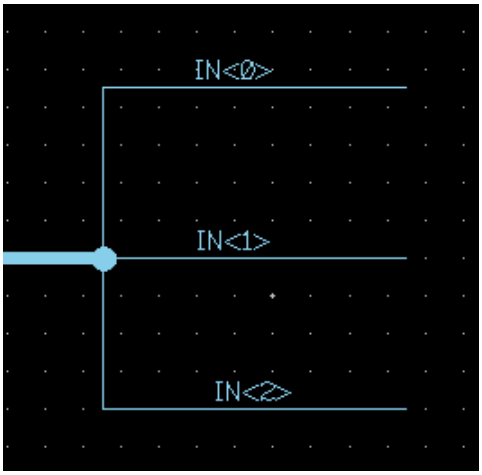
Wire Names: IN<0:2>
Bus Expansion: No
Placement: Single

Step 4: Directly move your cursor into the wire segment as the figure below. Click left mouse button to add the wire name to this wire segment.



Step 5: Move your cursor into “Add Wire Name” form again, fill out the “IN<0:2>” again in Wire Names field and modify the option Bus Expansion “No” to “Yes”.

Step 6: Directly move your cursor into Schematic Editor window. Click left mouse button to add the wire name to these tree wire segment one by one.



Step 7: Select icon  to undo added wire name IN<2>, IN<1> and IN<0>.

Step 8: Click icon  again, fill out the Add Wire Name form with the following information.

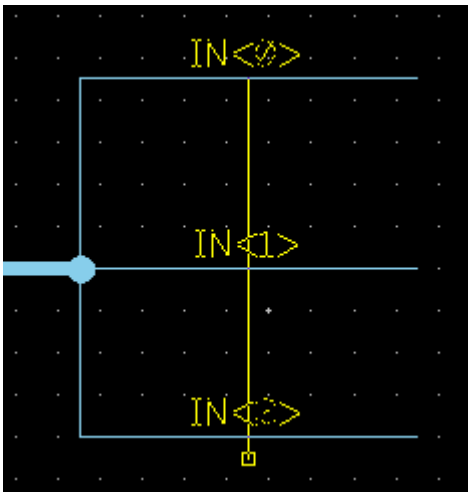
Wire Names: IN<0:2>

Bus Expansion: Yes

Placement: Multiple

Step 9: Hide the “Add Wire Name” form, move the mouse to the first wire segment and click left mouse button to fix it.

Step 10: Drag and move the cursor cross the other two wire segments. You can find the other names IN<1> and IN<2> appear in the cross point automatically as shown in the figure below.



Step 11: Click left mouse button to fix them.

Step 12: Save and close this schematic view.

Note:

To do this operation successfully, you must make sure that “Repeat” option is turned on in Command field of menu command Options->Editor.

Exercise - 15 Add-> Pin

This command lets you add a schematic pin or offpage pin to current schematic cellview.
In Add Pin form, there are 3 options need to be paid attention to.

- **Usage -*Schematic*** specifies the pin is a hierarchy pin, that is, it is an interface of input or output data.

  -Offpage specifies the pin is a page connector to connect each page. Please refer to *Exercise - 26 Page->Make Multiples page* on page 87

- **Display Name**

  -No doesn’t show pin name while creating pin. You can show the name by using Add->Property Label. Please refer to *Exercise - 16 Add->Property Label on page 80*.

- **Negative**

  -Yes specifies the pin is a negative logic with a short line added to the top of the wire name.

Note:

You can right-click mouse to rotate a pin.

Exercise - 16 Add->Property Label

This command shows property name of a specified object.
For example, if you set the option Display Name to No in *Exercise - 15 Add-> Pin* on page 79, you can show the wire name with the following steps.

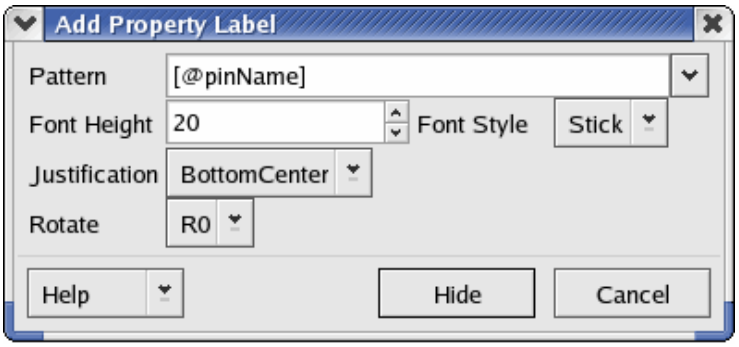
Step 1: Create a new cellview “Lab1.prolabel.schematic”.

Step 2: In cellview “Lab1.prolabel.schematic” window, select Add->Pin to create an input pin named “In” but the name is not shown.

Step 3: Select Add->Property Label.

Step 4: Select the pin in Schematic Editor window.

Step 5: The Pattern entry becomes valid in “Add Property Label” form, and select [@pinName] in dropdown list.

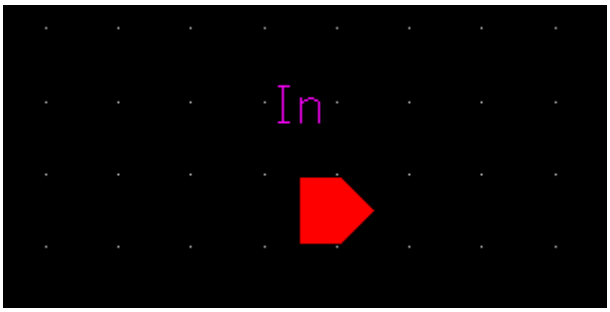


Step 6: Hide the “Add Property Label” form.

Step 7: You can find the string “[@pinName]” is attached with a line start from the object.

Step 8: Left-click mouse anywhere you want.

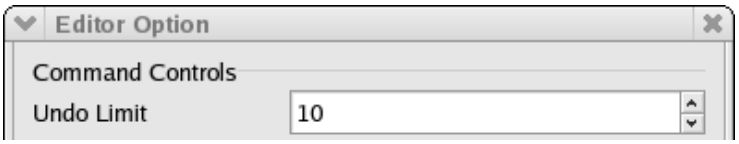
Step 9: The string “[@pinName]” is replaced by real pin name “In”.



Step 10: Save and close this schematic view.

Exercise - 17 Edit->Undo

This command lets you reverse the action of previous edition commands. You can set the undo limit from 0 to 10000 in "Option->Editor->Undo Limit" option. Please refer to *Exercise - 28 Options->Editor* on page 91.

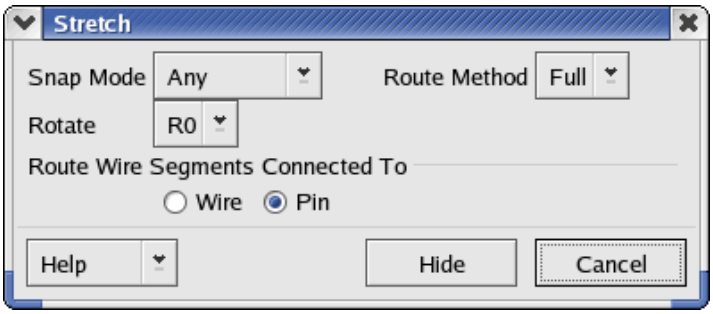


Note:

After doing "Check and Save", "Save", "Check Hierarchy" or "Discard Edits", *Undo* command doesn't work.

Exercise - 18 Edit->Stretch

This command lets you move the selected object to another location without losing connectivity between wire and pin.

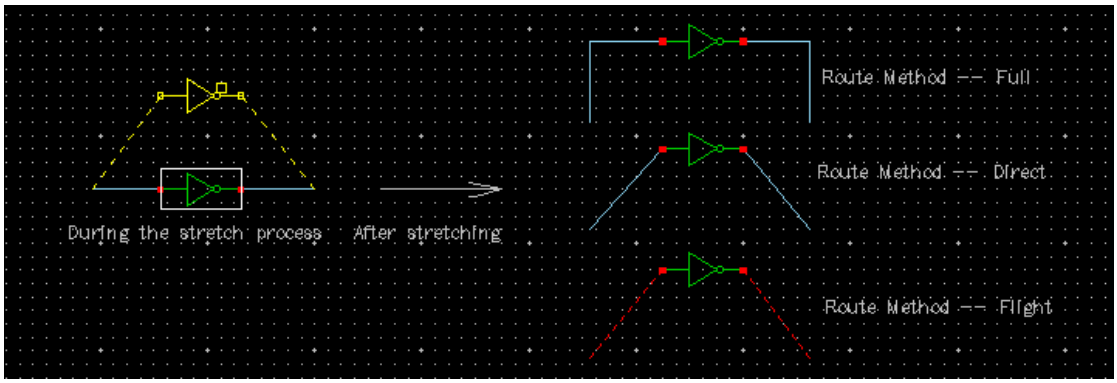


In "Stretch" form, there is 2 options need to be paid attention to.

1. **Route Method** specifies how the editor routes wire when you stretch an object to other position.

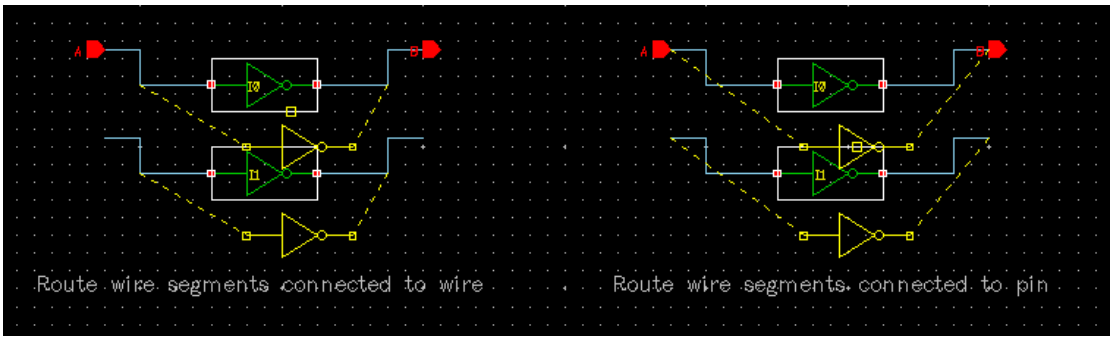
-Full automatically routing by editor.
-Direct uses straight line to connect two points
-Flight uses dashed line to connect two points.

The following figure shows the difference among 3 methods.



2. **Route Wire Segments Connected To** specifies the location rubber-band line begins from
   -Wire: the rubber-bind line begins from the last segment of the wire.
   -Pin: the rubber-bind line begins from a pin, a solder dot, or a wire ending.

Suppose that we stretch two instances at the same time, you can find the difference between two options from the following two figures.



Exercise - 19 Edit->Move/Copy

Two points of this command need to be paid attention to.
1. You can move or copy object from a cellview window to another cellview window.
2. When you drag mouse to move or copy object, you can right-click mouse to rotate the destination object.

Exercise - 20 Edit->Rotate

This command lets you rotate objects by left-click mouse each time till press “Esc” key to terminate the command.

Rotate direction depends on the setting in the “Rotate” form (Press “F3” to popup form)

To rotate preselected objects, do the following:
Step 1: Select objects first.
Step 2: Click Edit->Rotate.
Step 3: Point at a reference point around the objects.
Step 4: Left-click mouse time by time to rotate objects.
Step 5: Press “Esc” key to terminate the command.

To rotate postselected objects, do the following:
Step 1: Click Edit->Rotate.
Step 2: Select objects.
Step 3: Point at a reference point around the object.
Step 4: Left-click mouse time by time to rotate objects.
Step 5: Press “Esc” key to terminate the command.

Exercise - 21 Edit->Renumber Instance

This command lets you renumber all the instance names from number “0”.

Note:

If the prefix name of an instance is different from the prefix is defined in Design->Design Property->Instance Prefix field, system doesn’t renumber it.

Exercise - 22 Edit->Hide Label/Reset Invisible Label

These two commands let you hide or show the labels displayed in current cellview.
The labels are the names of pin, wire, instance, instance parameter, etc.

To hide labels, do as follow:
Step 1: Select Edit->Hide Label.
Step 2: Click on a label you want to hide. The label is invisible immediately.
Step 3: Click on another label to hide it till pressing “Esc” key to terminate command.

To show the label, do as follow:
Step 1: Select Edit->Reset Invisible Label.
Step 2: All of hidden labels will blink in the current cellview.
Step 3: Click on the blinking label you want to show one by one, or use area selection blinking

labels to make them display again.

Exercise - 23 Select->Area/Line Select

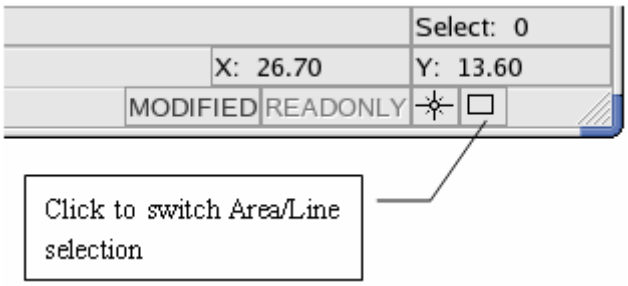
Area selection selects more objects in an area
Line selection helps you precisely select objects.

For line selection, do as follow:
Step 1: Click at a point, don't release mouse, and drag to draw a line in any direction.
Step 2: Release the mouse at the object you want to select.

If the line crosses some objects, these objects are selected as well.

Note:

You can click the icon at the right bottom of Schematic Editor window to switch area selection or line selection.



Comparing with menu command method, this method is different. Using menu command Select->Area/Line select only works on current session. When the command is finished, the default selection mode still depends on the icon command located at the right bottom of the Schematic Editor window.

Exercise - 24 Select->Trace Net

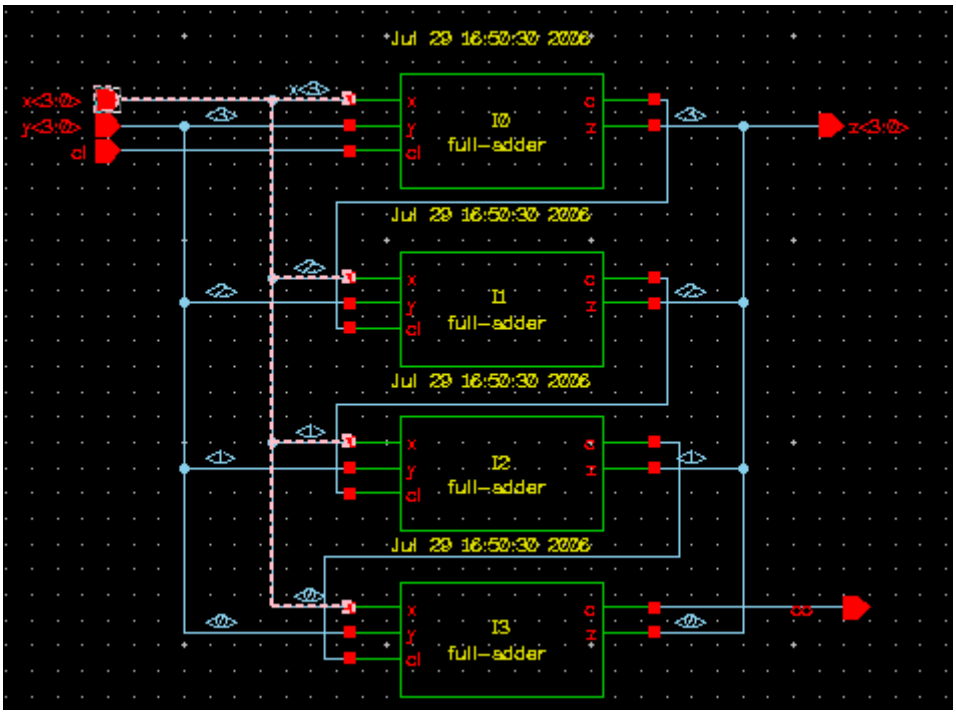
This command helps you trace a net with electrical connectivity by hierarchical.

To trace a net, do as follow:
Step 1: Open the cellview of "Adder.adder_4.schematic".

Step 2: In Schematic Editor window, select Check->Hierarchy to check full hierarchy first.

Step 3: Choose "Select->Trace Net".

Step 4: Click on any segment of a net or a hierarchy pin, for example, click on pin "x<3:0>", the whole net connected with the pin "x<3:0>" is highlighted in a colored wide dashed



Step 5: Click icon then left-click “IO” instance, choose “schematic” item from popup menu list to enter the schematic cellview of instance “I0”. In schematic cellview “Adder.full_adder.schematic” window, the net connected with the pin “x” is highlighted as well.

Note:

1. Before tracing net, you must check hierarchy cell.
2. You can trace more nets one by one till pressing “Esc” key.
3. Using *Select->Remove Trace* or *Select->Remove All Traces* to remove the highlight.
4. The color of highlighted dashed is specified in *Option->Color*.

Exercise - 25 Cellview->Create From Cellview

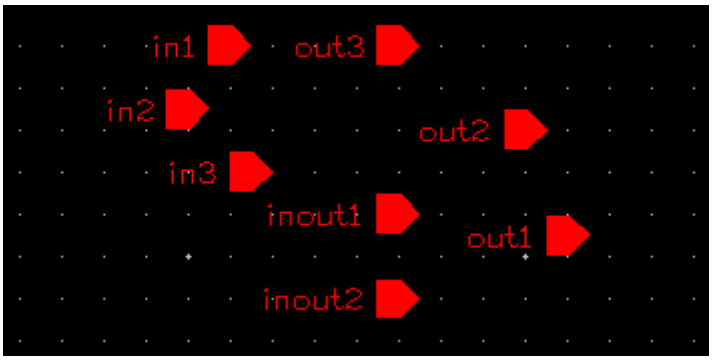
This command automatically creates a new cellview using the pins in current cellview. For example, create a symbol cellview for this cell from current schematic cellview.

To create a symbol cellview, we should create a schematic for preparation, do as follow:

Step 1: Create new cellview “Lab1.crtsym.schematic”.

Step 2: In cellview “Lab1.crtsym.schematic” window, select Add->Pin to create following pins and place them anywhere.

- Input pins— in1, in2 , in3
- Out pins—out1, out2 ,out3
- InOut pins – inout1, inout2

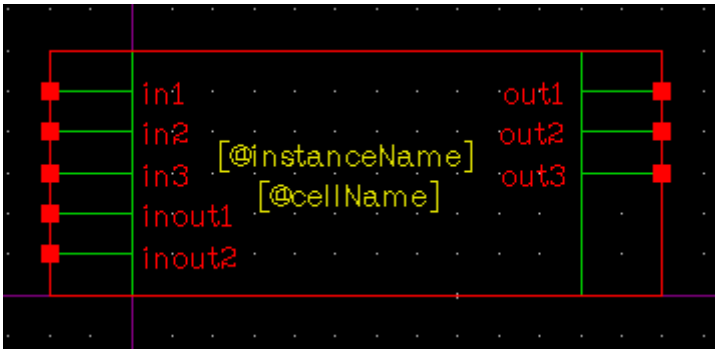


Step 3: Save the cellview.

Now create a symbol as follow:

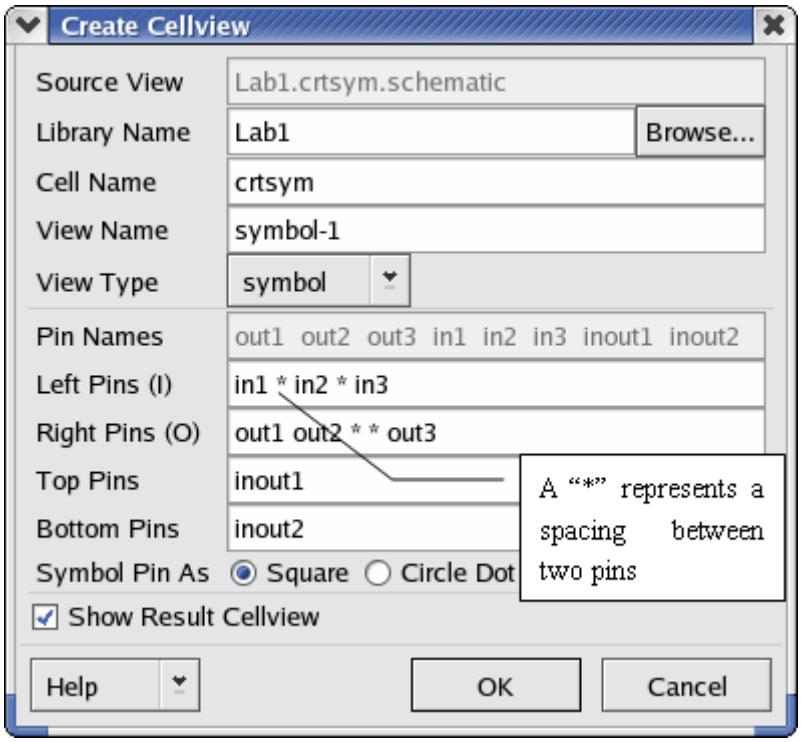
Step 4: Select Cellview->Create From Cellview.

Step 5: Directly ok the “Create Cellview” form. The symbol view is automatically created and is shown as below.

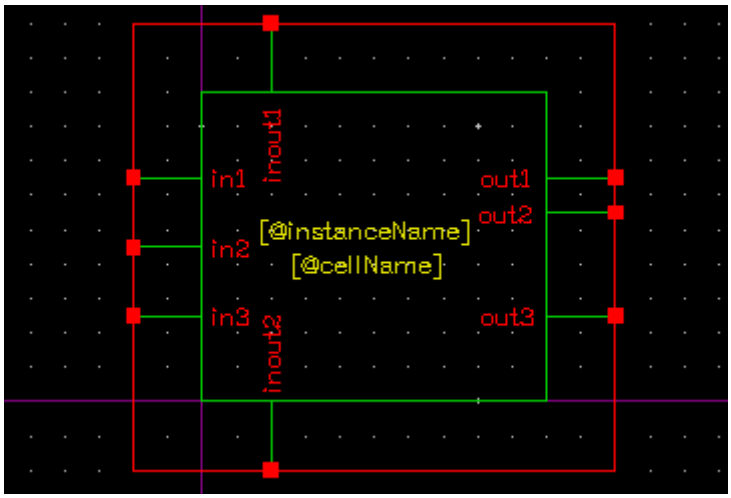


Step 6: The symbol cellview consists of a green rectangle, red rectangle, all of pins and two labels. Green rectangle is a real symbol; the red rectangle is selection box; all input and inout pins are at the left-side of the green rectangle, all of output pins are at the right-side.

Step 7: In cellview “Lab1.crtsym.schematic” window, fill out the Create Cellview form with the following information. Please note that modify View Name to “symbol-1”.



Step 8: The symbol cellview "symbol-1" is created as shown in the figure below.



Step 9: Close the symbol view and the schematic view.

Exercise - 26 Page->Make Multiples page

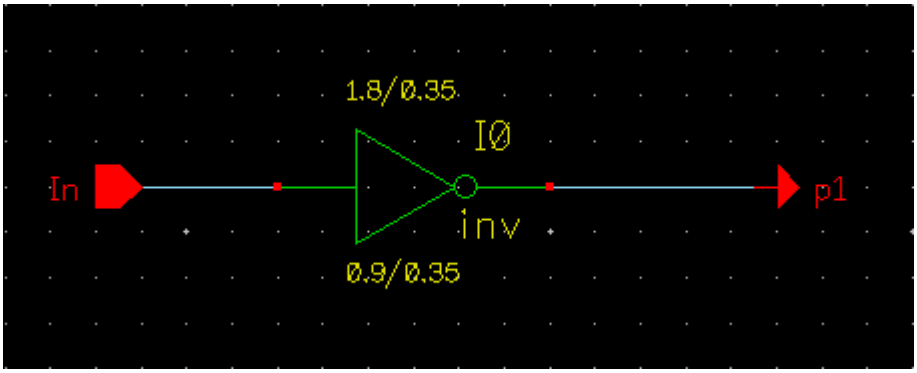
This command lets you use several pages to represent a whole design.

Do the following to understand this command in details.

Step 1: Create a new cellview "Lab1.page.schematic"

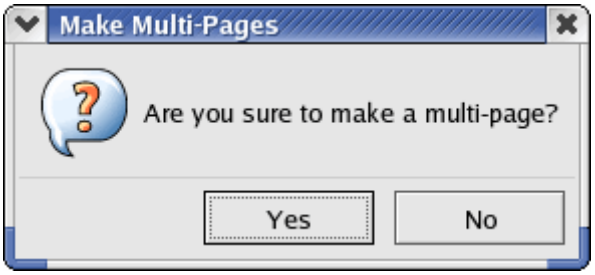
Step 2: In Schematic Editor window, create following objects:

- schematic input pin “In”
- instance of cell “INVtest.inv.schematic”
- offpage output pin “p1”.



Step 3: Save the cellview.

Step 4: Select page-> Make Multiples page. A question form appears as below.

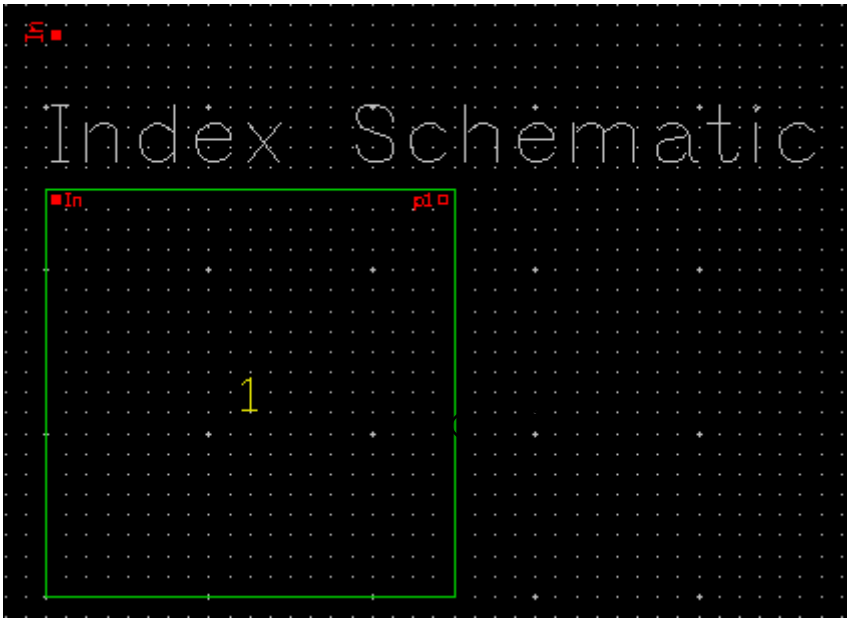


Step 5: Yes the form.

Step 6: The title of current Schematic Editor window becomes to “Lab1.page@1.schematic”. “@1” represents the first page.

Step 7: Close the current Schematic Editor window.

Step 8: Open the cellview “Lab1.page.schematic”, this cellview looks as the figure below.

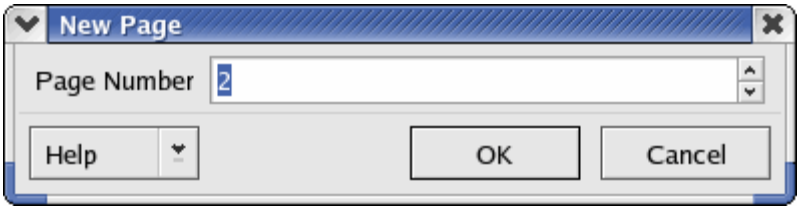


Step 9: This cellview includes an instance of cell “page@1”, and extracts the schematic pin of cell “page@1” as schematic pin of itself.

Step 10: In instance of cell “page@1”,

1. ■ represents the schematic pin
2. □ represents the offpage pin.
3. Input pin is in the left, output pin is in the right

Step 11: In this cellview window, select Page->New Page. A New Page form appears.

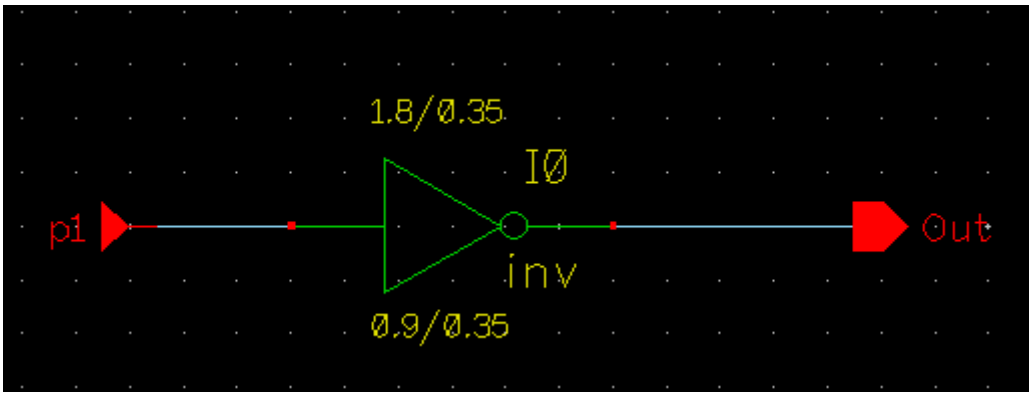


Step 12: System automatically fills out ‘2’ in Page Number to represent the next page is the second one.

Step 13: Ok the form. The cell “page@2” is created in Design Manager window, and the Schematic Editor pops up.

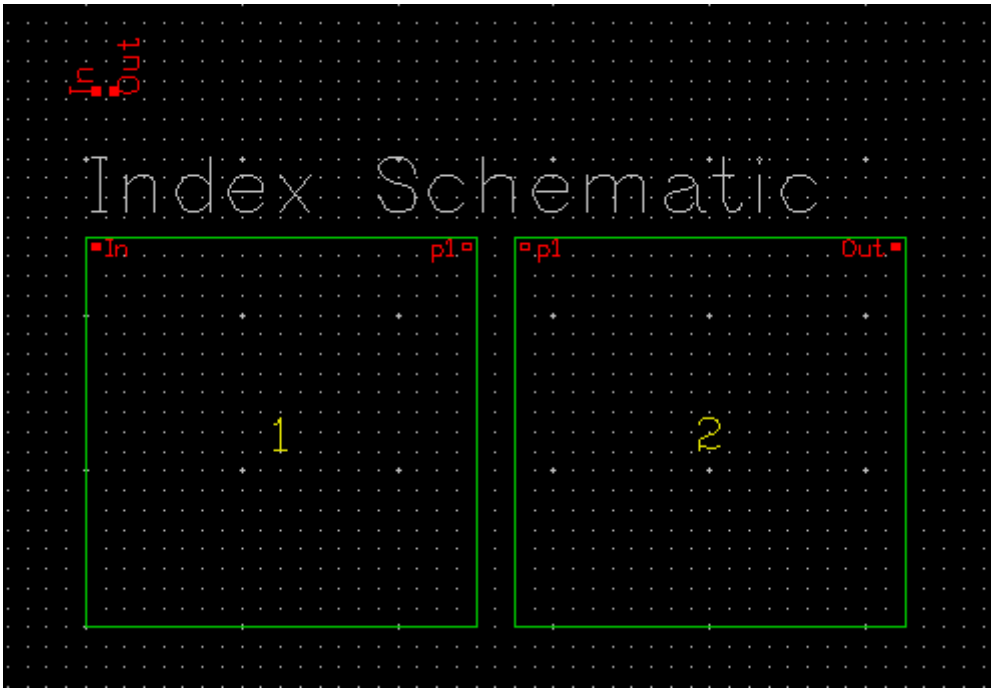
Step 14: In Schematic Editor window, create following objects:

- offpage input pin p1.
- instance of cell “INVtest.inv.schematic”
- schematic output pin out



Step 15: Save the current cellview.

Step 16: Open the cellview “Lab1.page.schematic” again. There are two instances as below.



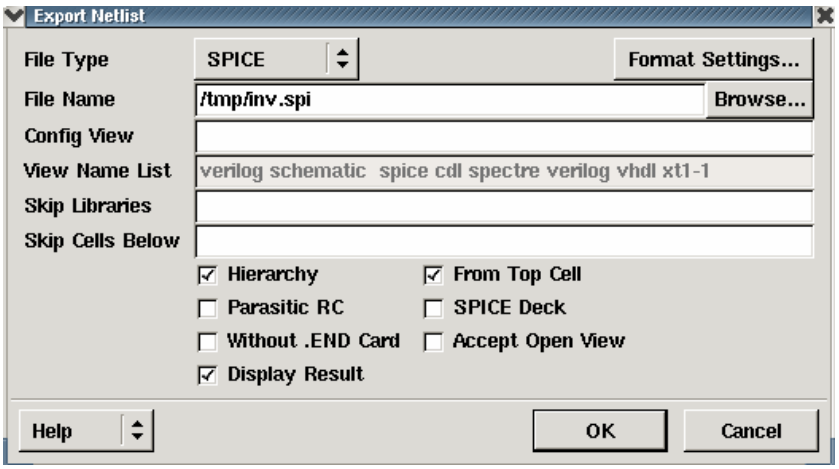
Step 17: Save and close the cellview.

Note:

Offpage pin must appear twice with reverse direction between two pages.

Exercise - 27 Tools->Export Netlist

This command lets you generate 6 types netlist SPICE, CDL, Vhdl, Verilog, Spectre and Mixed-Mode.
Mixed-Mode netlist depends on the configure view.



Click on the *Format Setting* to specify the netlist format. Please refer to *Exercise - 30 Options->Export Format* on page 95 .

View Name List comes from *options->Editor->View Name List*. Here, if you turn on the Hierarchy option, netlist generator searches related view name from this list according to exporting netlist type. For example, if you want to export verilog netlist in hierarchical, netlist generator will search verilog type cellview from the list one by one. Once find it out, netlist generator stop searching and directly add this verilog text to the export netlist.

From Top Cell means you can export hierarchical netlist from toplevel cell when you are in sub-cell view window.

Accept Open View means if there is no suitable view for sub-cell, ZeniSE skip it and continue to export netlist when you turn on the option, and ZeniSE will give warning message on DM window. If you turn off the option, ZeniSE will give error message and stop exporting netlist.

Note:

1. You can fill out File Name field with "%C". For example,

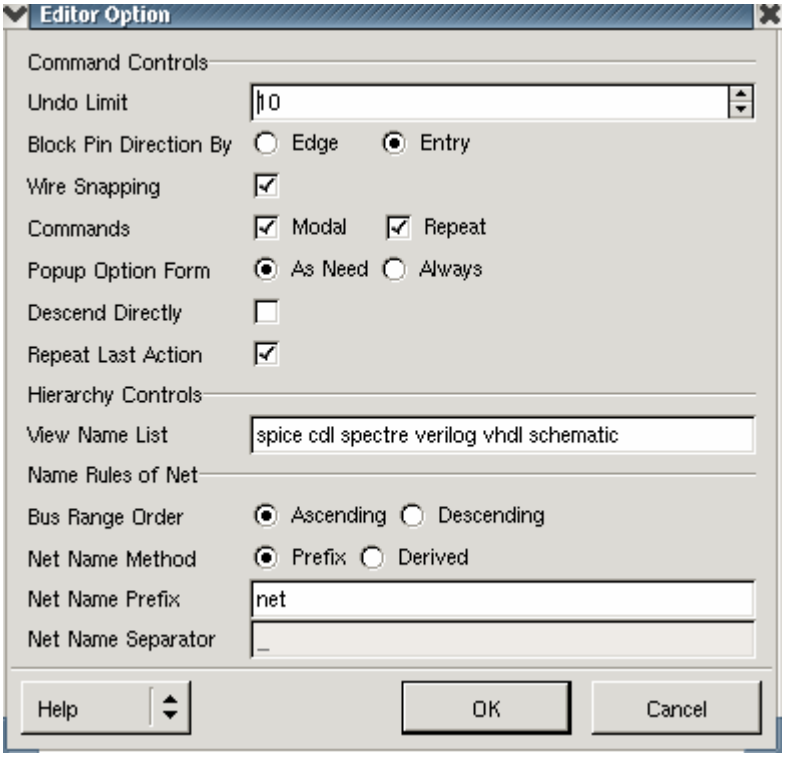
   

   . Here, %C stands for the current cellview name. Suppose that the current cellview name is "inv". The netlist exported will be "/tmp/inv/inv.spi".

2. The design must be checked before exporting netlist.

Exercise - 28 Options->Editor

This form specifies the common setting on editing.



About the form above,
Undo limit specifies the limitation of the undo times. 1~10000 is valid. Please refer to *Exercise - 17 Edit->Undo* on page 81.

Wire Snapping lets you use a key to quickly complete a wire drawing action. If you turn on this radio box , you may find the nearest pin to the cursor will have a small yellow diamond while you create a wire, press “Ctrl+left mouse button” or “Shift+left mouse button” key will snap route the current wire to that point. Please refer to *Exercise - 13 Add->Wire/Wide Wire* on page 76.

Command Controls field set the properties of command.

Commands- Modal specifies whether the previous command is suspended while a new command is executed. Turn on this option, a new command cancels the previous command, otherwise, the previous is suspended till the new command is over.

Commands-Repeat lets you repeat execution a command time by time till you pressing “Esc” key to terminate it.

Popup option form specifies whether the system pops up an associated form when execute some commands.

-As Need: popup associated form only when you press “F3” key.

-Always: popup form automatically when a command is executed.

Descend Directly lets you directly enter the instance design when you double click an instance.

To access which cellview of instance design depends on the following order.

1. If this view has a viewBind property, the system will access this view specified by viewBind. About viewBind property, please refer to *Exercise - 31 graphic, lvsIngore, ercIgnore, drcIgnore, viewBind* on page 99.
2. If there is no viewBind property, system searches view with schematic type in *View Name List*.
3. If there is no view with schematic type, system searches view from the head of *View Name List* one by one.

Repeat Last Action to control repeat executing last command by clicking the right mouse button . But if your mouse is not sensitive enough, please turn off the option because it is always do last command when you want to zoomin or zoomout image with clicking the right mouse button

View name List is a list of views name. It is used to check hierarchy, export hierarchy netlist, or descend directly. According to the order of this list, system decides to check/export/descend which cellview of instance cell.
The order of cellview names is very import. For example, when you check a full hierarchy design, for instance cell, the system will search names in this list one by one till finding out existed schematic cellview name of instance cell. Suppose that cell “a” references an instance of cell “b”. But cell “b” has two schematic cellviews, “schematic1” and “schematic2”. When you check full hierarchy from cell “a”, the system will find the suitable schematic cellview name of cell “b” from view name list and check it. If the “schematic2” is in front of “schematic1” in the view name list, SE will check “schematic2” cellview.

Name Rules of Net field set the naming rule of net while check hierarchy.

Bus Range Order decides to export net name to verilog netlist in ascending order or in descending order. For example, if you name a wire as “In<0>, In<3>, In<2>”, if you turn on “Ascending” option, the wire name is exported to verilog netlist as “wire [1:3] In”, otherwise, export it as “wire [3:1] In”.

Please note that if the bus name is a pin name, it is not controlled by the option. For this case, ZeniSE checks the first second pin name order and decide to export them to netlist in ascending order or in descending order. For example, if pin name is “In<3>,In<1>,In<2>”,ZeniSE will export “Input [3:1] In” to verilog netlist. If the pin name is “In<2>, In<3>,In<1>”, ZeniSE will export “Input [1:3] In ” to verilog netlist.

In additionally, if there are 3 pins “In<1>”, “In<2>” and “In<3>”, ZeniSE checks the first second pin name order from command form “Design->Pin Order” and decide to export them to netlist in ascending order or in descending order

Net Name Method specifies how editor names net which hasn’t been named after be designed.

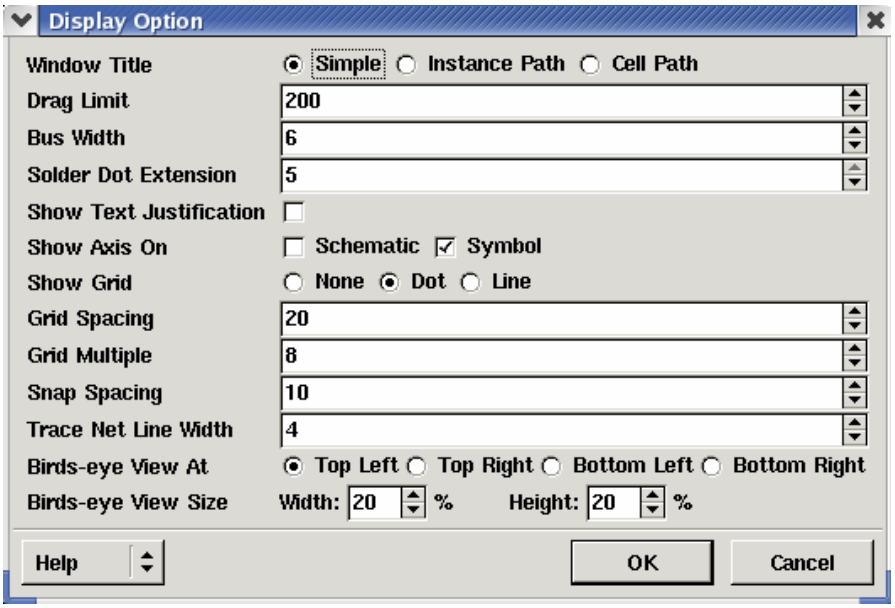
-Prefix uses a prefix string and a number to name the net .The prefix string is specified at the *Net Name Prefix* entry. The number starts from number “1”. For example, name the net

to “net1”.

-Derived uses the instance name, name separator and instance pin name to name the net. The name format is <instance name><net name separator><instance pin name>. The name separator is specified at the *Net Name Separator* entry. For example, name the net to “I2-in”

Exercise - 29 Options->Display

This form specifies the common settings about schematic display.



About the form above,

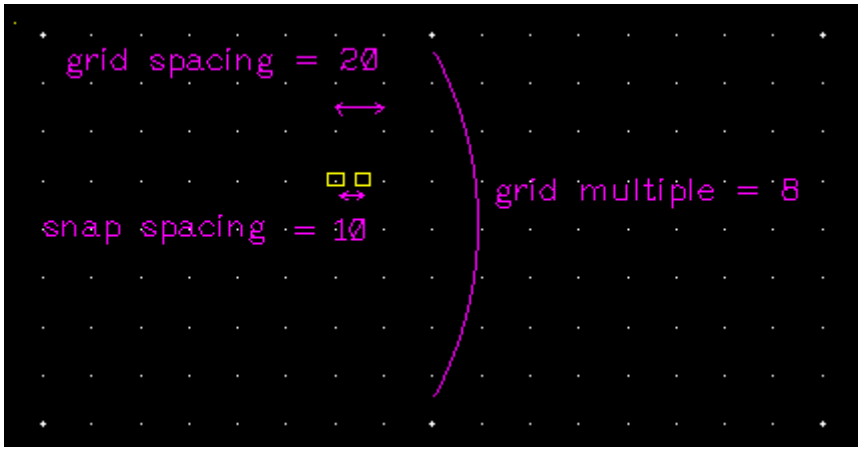
Drag Limit lets you specify a maximum number of objects which will be dragged. If the total number of objects you selected in schematic editor window is larger than the specified number, system will use a boundary box instead of all the selected objects during dragging.

Grid spacing specifies the distance between two grids.

Grid Multiple specifies how many grids constitute a grid matrix.

Snap Spacing specifies the distance of moving cursor for one step.

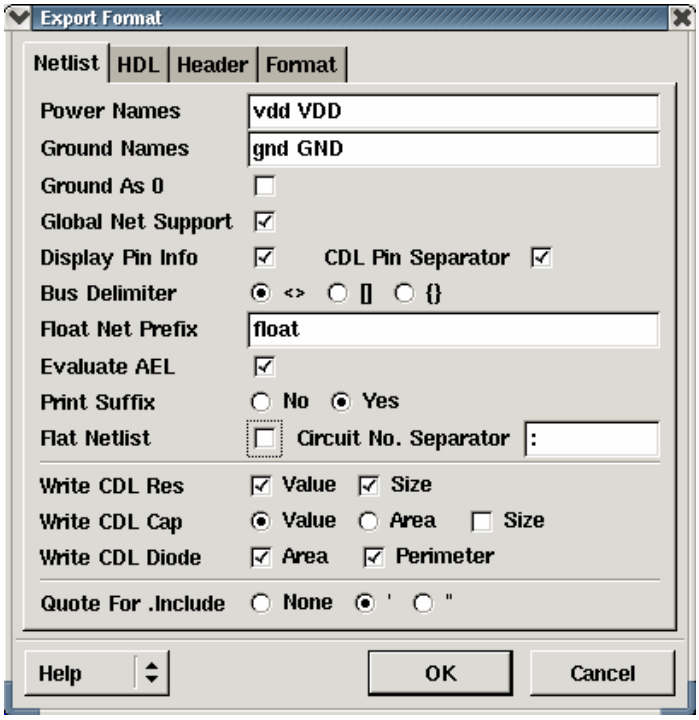
In the following figure, you can understand better of the three options above.



Trace Net Line Width specifies the width of highlight line when you execute command “Trace Net”. About command “Trace Net”, please refer to *Exercise - 24 Select->Trace Net* on page 84.

Exercise - 30 Options->Export Format

Netlist tab specifies the rules for generating SPCIE netlist or CDL netlist.



About the form above,

Power Names specifies the power names. If the pin name or net name is the same as one of the name list, netlist generator regards it as power.

Ground Names specifies the ground names. If the pin name or net name is the same as one of the names in that list, netlist generator regards it as ground.

In general, name must be ended with '!' in *Power Names* and *Ground Names*.If you type the name without '!', the netlist generator will add '!' at the end of the name automatically. But you should assure the net name or pin name with '!' must exist in schematic view.

Ground As 0 lets the netlist generator export '0' instead of the ground signals.

Global Net Support lets netlist generator export statement ".Global" to the netlist file.

Display Pin Info lets the netlist generator export the pin name and pin direction in statement *.PININFO" in CDL netlist.

CDL Pin Separator lets the netlist generator export the separator '/' in CDL netlist, which to distinguish that the pin is either output pin or input pin.

Bus Delimiter specifies the type of bus delimiter. Which delimiter is used depends on your simulator. For example, A<1:2>, A [3:4], A {3:5}.

Float Net Prefix specifies the float net in netlist. The entire name is as "<prefix>#<number>". For example, "float#1".

Evaluate AEL lets the netlist generator calculates the value of AEL in component property. For example, we suppose that the value of "W" is "0.9u", for expression "AD=iPar(W)*0.24u", the value of "AD" is "0.216p".If turn off this option, the value of "AD" is "0.9u*0.24u". About AEL, please refer to *Exercise - 33 AEL* on page 107.

Print Suffix specifies whether to print unit “u” or “p” to netlist or not. Please refer to *Exercise - 33 AEL* on page 107.

Flat Netlist lets the netlist generator export the flat netlist, not hierarchical. Export either the flat netlist or hierarchical netlist depends on your simulator.

Circuti No.Separator is special for generating flat netlist. A number followed the separator specifies the hierarchy of a device in hierarchical. For example, we set this separator to ":", a part of flatten netlist is shown as below.

```
M0:1 xdec_en net0 gnd! gnd! nmos
M1:1 xdec_en net0 vdd! vdd! pmos
M0:2 net3 net0 gnd! gnd! nmos
M1:2 net3 net0 vdd! vdd! pmos
```

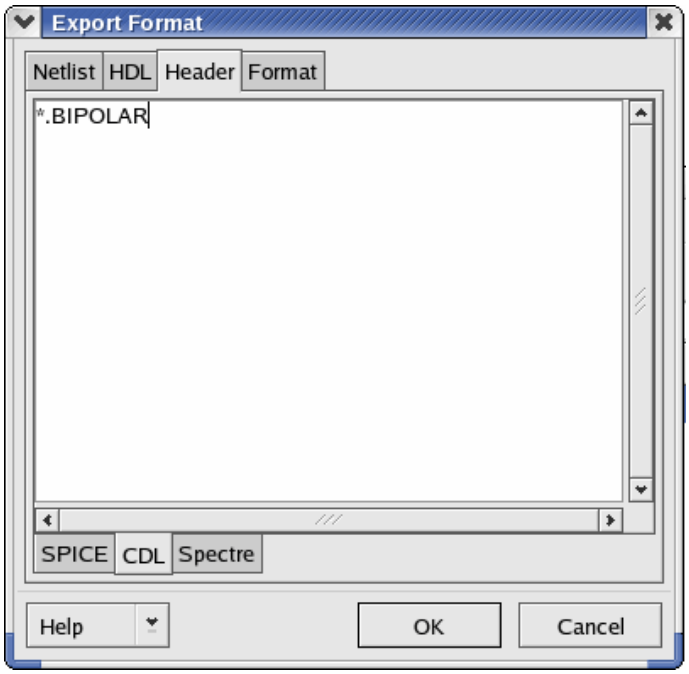
Write CDL Res specifies whether to export the value or (and) size of resistor to CDL netlist or not.

Write CDL Cap specifies whether to export the value or area of capacitor to CDL netlist or not.

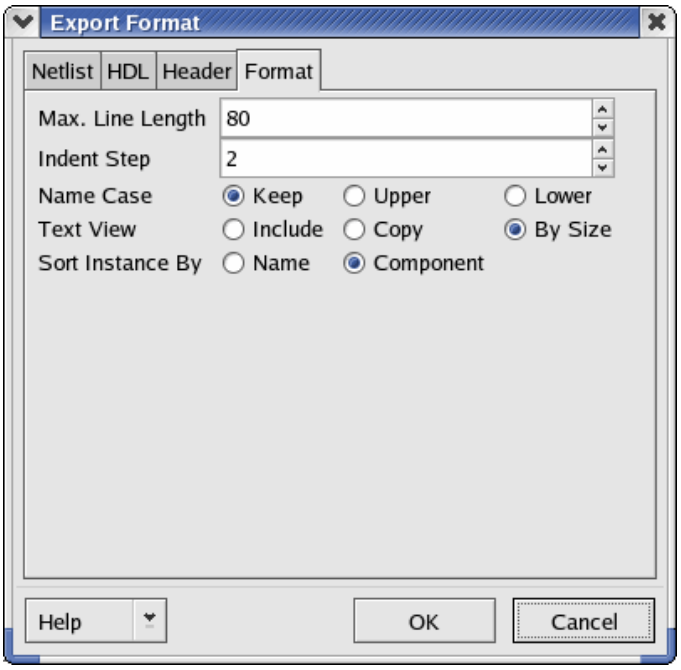
Write CDL Diode specifies whether to export the area ,size (main for capacitor)or (and) perimeter of resistor to CDL netlist or not.

Quote For .Include specifies which quotation mark is used in ".include" sentence according to different simulator.

Header tab lets you type any netlist statement or export spice/cdl/spectre netlist., the netlist generator will add these statements to the head of the netlist.



Format tab specifies the format of the netlist to be generated.



Text view specifies that when you export verilog netlist, if an instance cell has verilog cellview (netlist), netlist generator lets you add the existed verilog netlist to the exported netlist with one of the following methods.

-Include uses 'include' expression to add the existed verilog netlist to exported netlist.

-Copy adds the whole verilog netlist text to exported netlist.

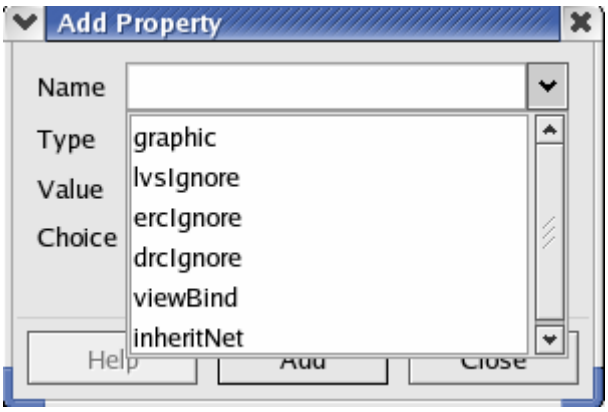
-By size specifies that if the size of existed verilog netlist is equal or larger than 1024 bytes, netlist generator uses *Include* option to generate netlist, otherwise, use *Copy* option to do that.

6.2.4. Advanced Features

Exercise - 31 graphic, lvsIngore, ercIgnore, drcIgnore, viewBind

In general, these properties are used in *Design->Design Property* command in Symbol Editor. If the symbol cellview is referenced as an instance, you can modify or add these properties only for this instance in Property form in Schematic Editor. That is, Property form in Schematic Editor has higher priority.

In Design Property form, click Add button to define the property for this symbol cellview. In *Name* dropdown- list, there are 6 kinds of properties are shown as figure below.



Later, we will introduce each property. But the precondition is that we suppose this symbol is referenced by another schematic cell “top”.

graphic is a boolean parameter. If you set it to True, in schematic cellview ‘top’ window, this symbol is regarded as a graphic, not an instance. So Schematic Editor will neither check this symbol nor export it to netlist.

lvsIgnore is a boolean parameter. If you set it to True, netlist generator will not export this instance to CDL netlist. Or only export the cell that is in the toplevel into the SPICE netlist. Normally, this property is used to voltage source and current source.

Do the following to understand this property more in details.

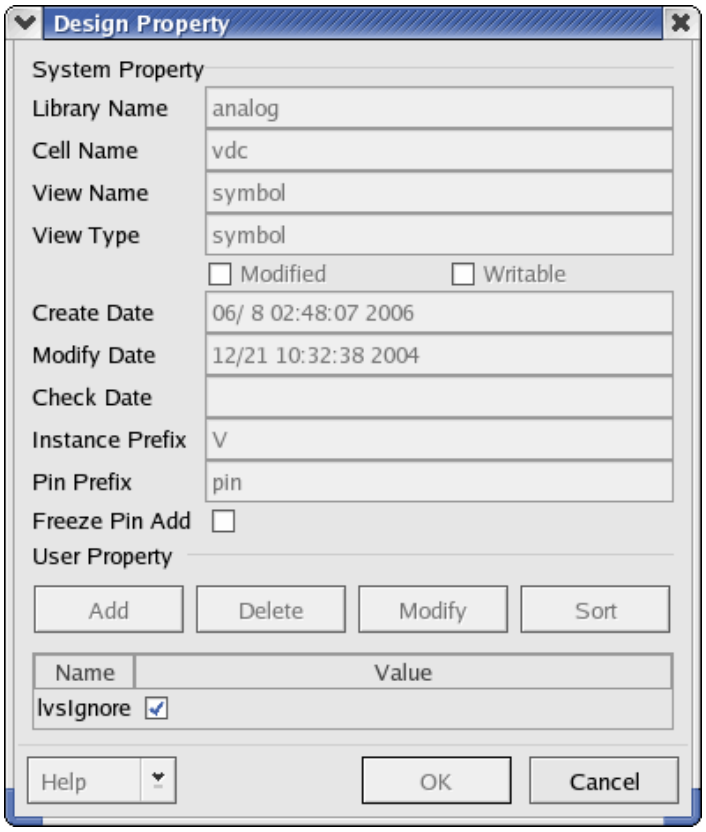
Step 1: Open the cellview “Adder.full-adder.schematic”.

Step 2: Click icon  to enter the schematic cellview of instance “I5”. You can find a voltage source “vdc” is in this cellview “Adder.half_adder.schematic”.

Step 3: In cellview “Adder.half_adder.schematic”, click icon  to enter the symbol cellview of instance V0.

Step 4: In symbol cellview “analog.vdc.symbol”, select *Design->Design Property*.

Step 5: In Design Property form, the “lvsIgnore” property had been set to True by default.



If you export spice netlist from cellview “Adder.full-adder.schematic”, netlist generator only exports voltage source “vdc” that is in the cellview “Adder.full-adder.schematic” to spice netlist because of lvsIgnore property.

ercIgnore is a boolean parameter. If you set it to true, in schematic cellview ‘top’ window, Schematic Editor will not report error massage when it has no pin. This property is used in cell without any pins, such as all of cell in libraries “Sheet” and “Spice”.

drcIgnore is a boolean parameter. If you set it to true, in schematic cellview ‘top’ window, suppose that this instance overlap with another instance, Schematic Editor will not report error message for overlapped instances.

viewBind specifies exporting which cellview to netlist. If a cell has some schematic cellviews and one symbol cellview, when this symbol cellview is as an instance, normally, this symbol represents which kind of cellview depends on View Name List. But you can use viewBind to

specify a cellview for this symbol (instance).

This property is mainly used in Property command in Schematic Editor.

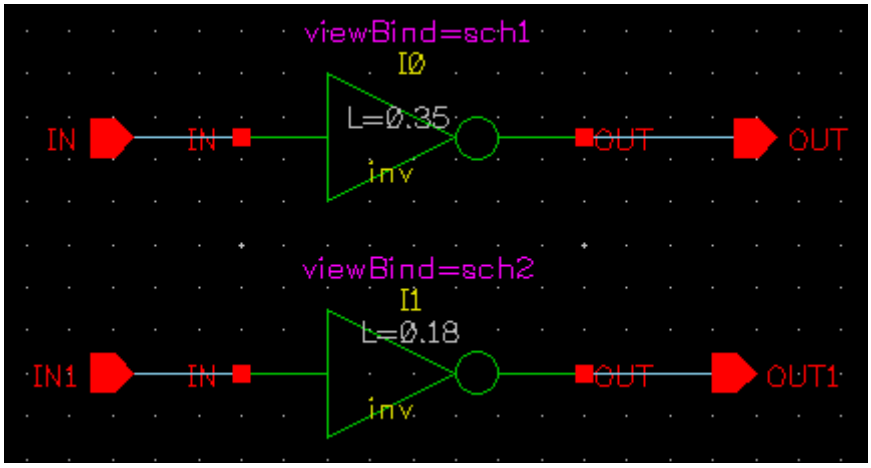
You may understand more in detail with the following steps.

Step 1: In Design Manager window, from left to right, click “ViewBind” in library field and “inv” in cell field.

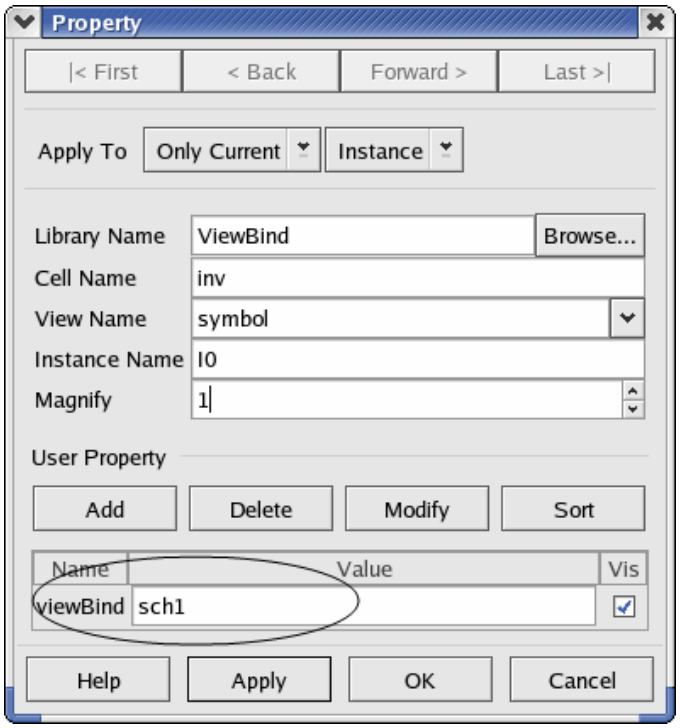
Step 2: In view list, there are 3 schematic type cellviews “sch1”, “sch2” , “schematic, and one symbol cellview named “symbol”.

Step 3: Respectively open “sch1”, “sch2” and “schematic” cellviews, you can find the 3 schematics are different in device parameter.

Step 4: Open the schematic cellview “ViewBind.top.schematic”. The symbol view “ViewBind.inv.symbol” is referenced twice as instance I0 and I1.



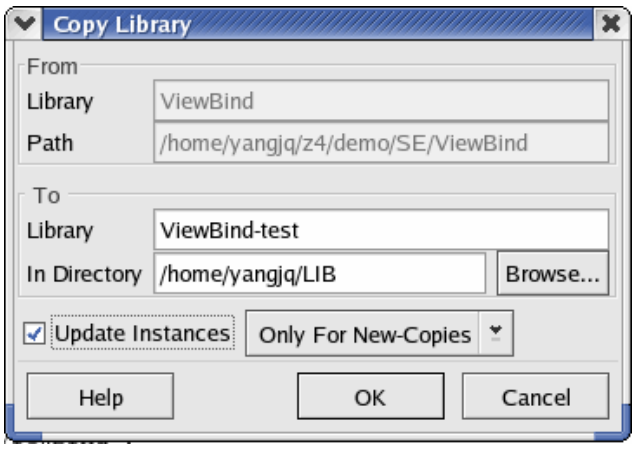
Step 5: Select instance ‘I0’ and click icon  the Property form appears as below.



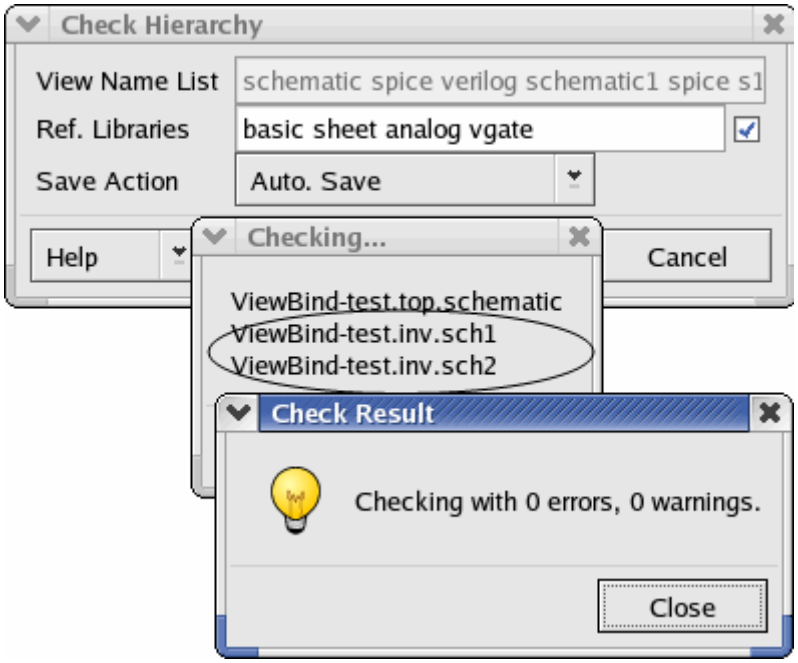
Step 6: Shown as the figure above, we had added viewBind to “sch1” in User Property section. You can find that instance “I1” viewBind has been set to “sch2” in the same way.

Because this library is a demo library, you cannot check it. You need to copy the demo library to another.

Step 7: In Design Manager window, copy the library to “ViewBind-test”.



Step 8: Open cellview “ViewBind-teste.top.schematic”. Select *Check->Hierarchy*, Schematic Editor will check view “sch1” and “sch2” though these two view names are not listed in View Name List.



Step 9: Select *Tools->Export Netlis*t to export spice netlist. The netlist is shown as figure below.

```
*top
*****************************************
* Tool     : ZENI-SE
* Date     : Mon Sep  4 09:52:32 2006
* Top Cell : top
*****************************************

.GLOBAL vdd! gnd!

*****************************************
* Library : ViewBind-test
* Cell    : inv
* View    : sch2
*****************************************

.SUBCKT inv_sch2_ViewBind-test OUT IN
M0 OUT IN vdd! vdd! PMOS L=0.18u W=1.8u
M1 OUT IN gnd! gnd! NMOS L=0.18u W=0.9u
.ENDS inv_sch2_ViewBind-test

*****************************************
* Library : ViewBind-test
* Cell    : inv
* View    : sch1
*****************************************

.SUBCKT inv_sch1_ViewBind-test OUT IN
M0 OUT IN vdd! vdd! PMOS L=0.35u W=1.8u
M1 OUT IN gnd! gnd! NMOS L=0.35u W=0.9u
.ENDS inv_sch1_ViewBind-test

*****************************************
* Library : ViewBind-test
* Cell    : top
* View    : schematic
*****************************************

XI0 OUT IN inv_sch1_ViewBind-test
XI1 OUT1 IN1 inv_sch2_ViewBind-test

.END
```

This netlist illustrates that netlist generator exports “sch1” and “sch2” to netlist. But you may find the sub-circuit name was changed to “inv_sch1_ViewBind-test” and “inv_sch2_ViewBind-test”. It because of netlist generator changes the same sub-circuit name to “cellname_viewname_libraryname”.

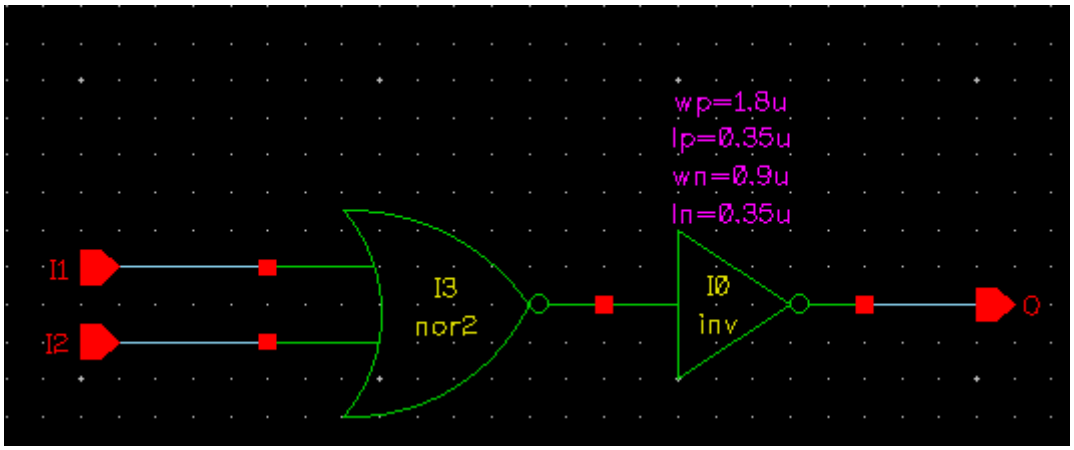
Exercise - 32 pPar( ), iPar( )

These two expressions are used to inherit parameter.

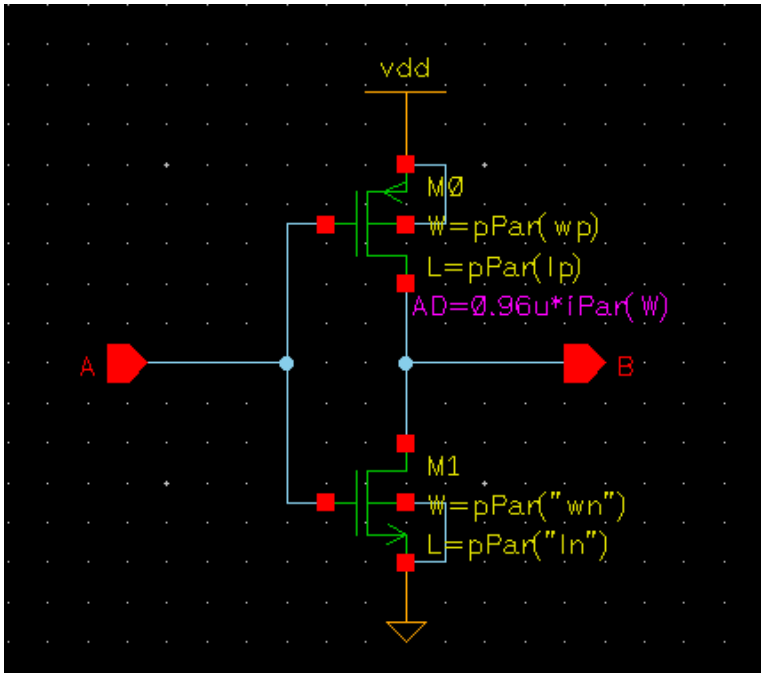
pPar() inherits parameter from up-level cell.
iPar() inherits parameter from its another parameter.

The format of pPar is pPar(“component parameter”). Double quotation marks (“ ”) can be ignored.
The format of iPar is iPar(“another parameter”). Double quotation marks (“ ”) can be ignored.

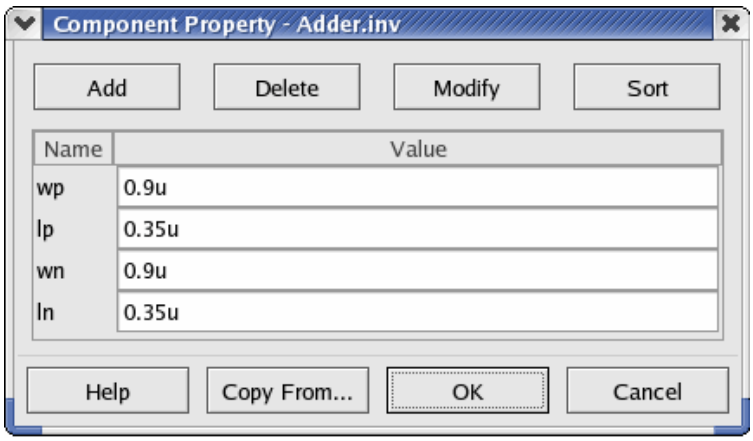
Open view “Adder.or2.schematic”, the schematic window is shown as figure below.



Click icon  to enter the schematic view of instance “I0”.



For instance “M0” and “M1”, set the parameter W and L to pPar() expression, “wp”, “lp”, “wn” and “ln” are the component parameters of this cellview. You can select Design->Component Property to check them.



For instance “M0”, set AD to iPar(W) expression, ‘W’ is its another parameter because the area of drain depends on its “W”.

Export hierarchy netlist of cell “or2”, the sub-circuit inv’s netlist is shown as below.

```
*****************************************
* Library : Adder
* Cell    : inv
* View    : schematic
*****************************************

.SUBCKT inv B A wp=0.9u lp=0.35u wn=0.9u ln=0.35u
M0 B A vdd! vdd! pmos L=lp W=wp AD='0.96u*wp'
M1 B A 0 0 nmos L=ln W=wn
.ENDS inv
```

If you export flat netlist of cell “or2”, the whole netlist is shown as below. The field enclosed by ellipse is inv’s netlist.

```
*****************************************
* Library : Adder
* Cell    : or2
* View    : schematic
*****************************************

M0:1 0 net0 vdd! vdd! pmos L=0.35u W=1.8u AD=1.728p
M1:1 0 net0 0 0 nmos L=0.35u W=0.9u
M0:2 net0 I2 0 0 nmos L=0.35u W=0.9u
M1:2 net0 I1 0 0 nmos L=0.35u W=0.9u
M2:2 net0 I2 net0:2 vdd! pmos L=0.35u W=1.8u
M3:2 net0:2 I1 vdd! vdd! pmos L=0.35u W=1.8u
```

Netlist generator automatically give the value of expression pPar() and iPar() to parameter of device.

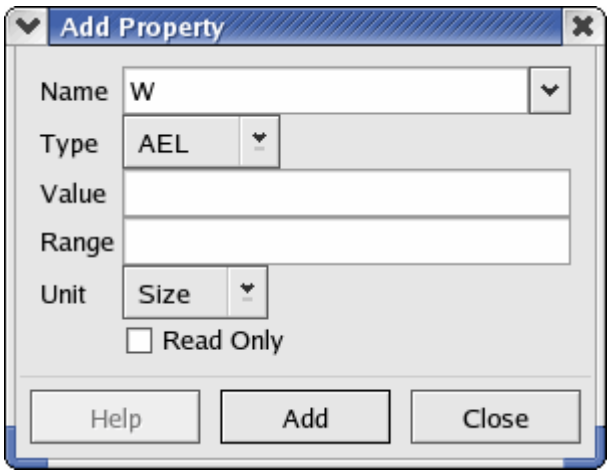
Netlist generator can calculate the expression’s value because the option *Options->Export Format-> Evaluate AEL* is turned on, otherwise, the parameter “AD” cannot be calculated and exported to netlist as below.

```
M0:1 0 net0 vdd! vdd! pmos L=0.35u W=1.8u AD='0.96u*1.8u'
```

Exercise - 33 AEL

AEL is the abbreviation of Analog Expression Language. It is used to create component parameter.

If you want to use W=pPar(xxx)"feature, the parameter "W" must be set to AEL type in *Design->Component Property->Add Property* form.



For AEL type, there are three units Normal, Size and Area.

- Nomal has no unit, the netlist absolutely depends on the parameter value.
- Size's unit is 'u'. If you set the parameter to 5k, then netlist generator will print "5e+09u" to netlist.
- Area's unit is 'p'. If you set the parameter to 5k, then netlist generator will print "5e+15p" to netlist.

For unit of Size or Area, whether netlist generator print 'u' or 'p' to netlist depends on *Options->Export Format->Print Suffix* option (please refer to *Exercise - 30 Options->Export Format* on page 95). If you turn off the option, the netlist generator will not print the 'u' or 'p' to netlist. For Normal, *Print Suffix* option doesn't work.

Suppose that there are three parameters W, L and AD.

- W is set to normal.
- L is set to size.
- AD is set to Area.

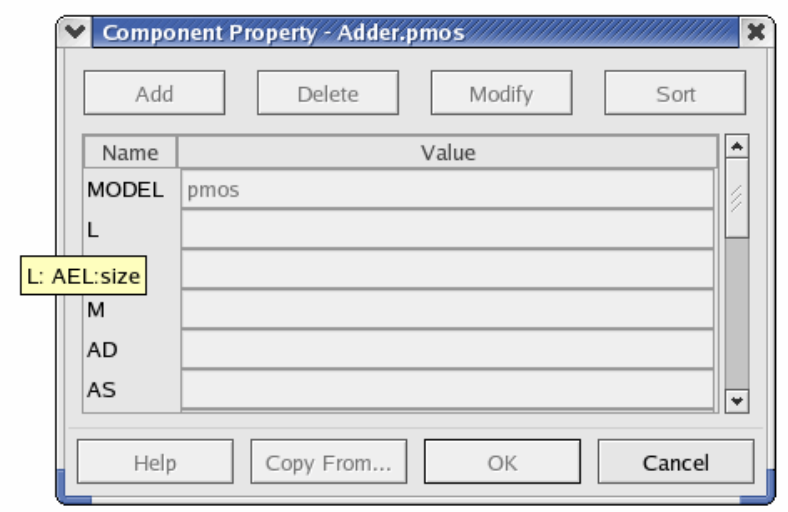
In following tables, we set three values to each parameter, right two columns are the exported value in netlist when turn on/off *Print Suffix* option.

W: Normal Value (W=)	Print Suffix	
	No	Yes
5	5	5
5u	5u	5u
5k	5k	5k

L: Size Value (L=)	Print Suffix	
	No	Yes
5	5	5u
5u	5	5u
5k	5e+09	5e+09u

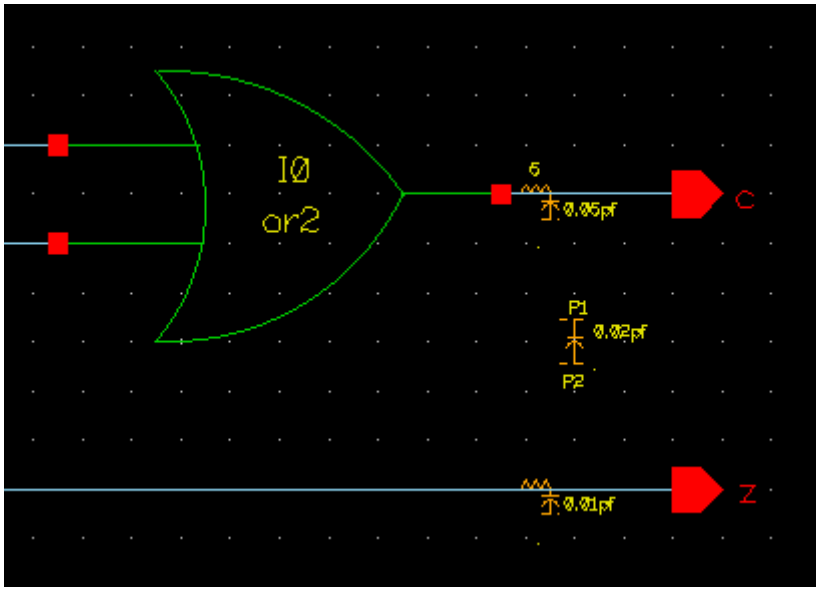
AD: Area Value (AD=)	Print Suffix	
	No	Yes
5	5	5p
5k	5e+15	5e+15p
5kp	5e+15	5e+15p

Open view "Adder.pmos.symbol", *select Design->Component Property.* In "Component Property" form, put your cursor to a parameter name, a yellow small window will popup after a moment. It is convenient to see the parameter's property from this small window.



Exercise - 34 Pre-defined Parasitic Loads

In Zeni Schematic Editor, you can add estimated parasitic loads to schematic design stage and export them to netlist.
Open the cellview "Adder.full-adder.schematic", you can find the following image.

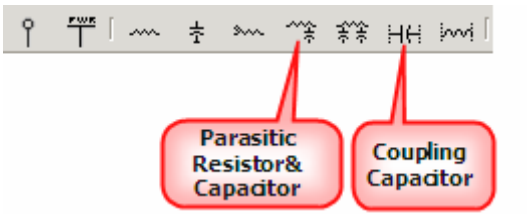


There are two icons  are placed in net ‘c’ and net ‘z’, one icon  is placed anywhere. In face, the icon is regarded as instance.

 represents the parasitic resistor and parasitic capacitor of net

 represents coupling capacitor between two nets

You can find the two icons in the icon bar at the top of the editing area.



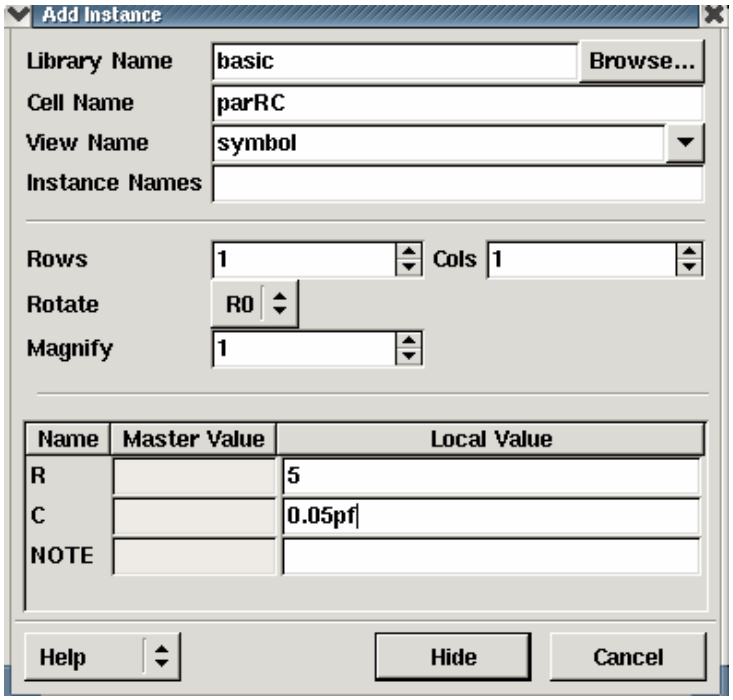
Next, we do an exercise to know how to add parasitic loads in schematic design.

Step 1: Because library ‘Adder’ is a demo library, you cannot modify anything in it. We copy library ‘Adder’ to ‘Adder-1’ first.

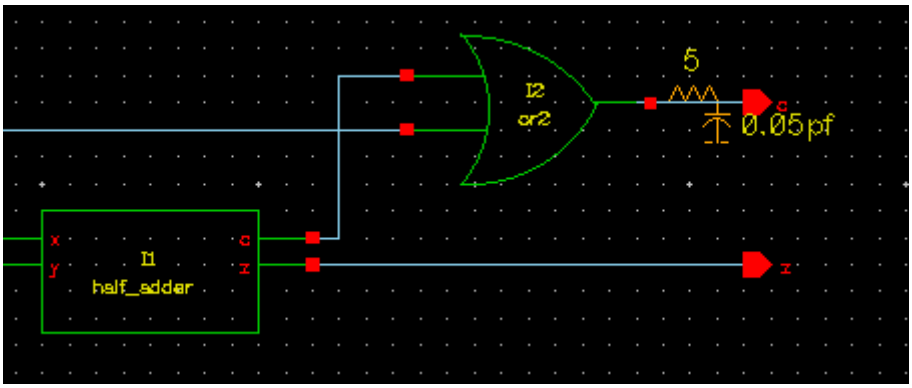
Step 2: Open the cellview “Adder-1.full-adder.schematic”.

Step 3: Click icon  in icon bar, Add Instance form pop up.

Step 4: Fill out the form with value of R and C shown as figure below.



Step 5: Move cursor to schematic window, and click at net ‘c’. The pre-defined parasitic resistor/capacitor which is regarded as an instance is added to the net ‘c’ automatically. It means the pre-defined parasitic resistor of the net is 5 ohm, the pre-defined parasitic capacitor of the net is 0.05 pf. The figure is shown as below.



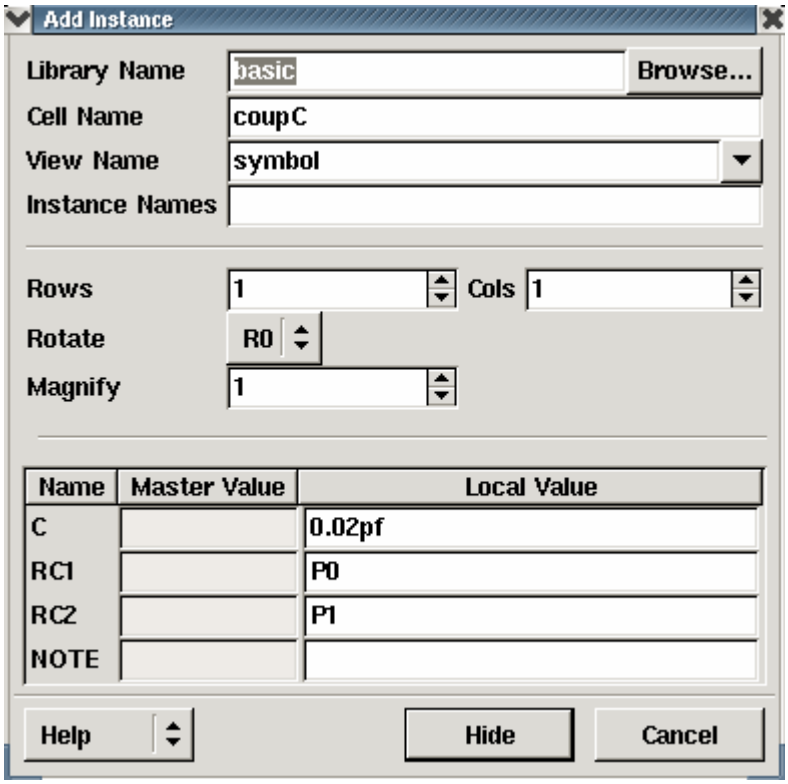
Step 6: With the same way, add another pre-defined resistor and capacitor to net z with following information.

R: 1

C:0.01pf

Step 7: Click icon , Add Instance form pop up.

Step 8: Fill out the form with information shown as below figure.



In RC1 and RC2 fields, you need to fill the instance name of pre-defined parasitic load in net ‘c’ and in net ‘z’. So you need to know this two instance names before filling out the above form. To know the instance name, you should

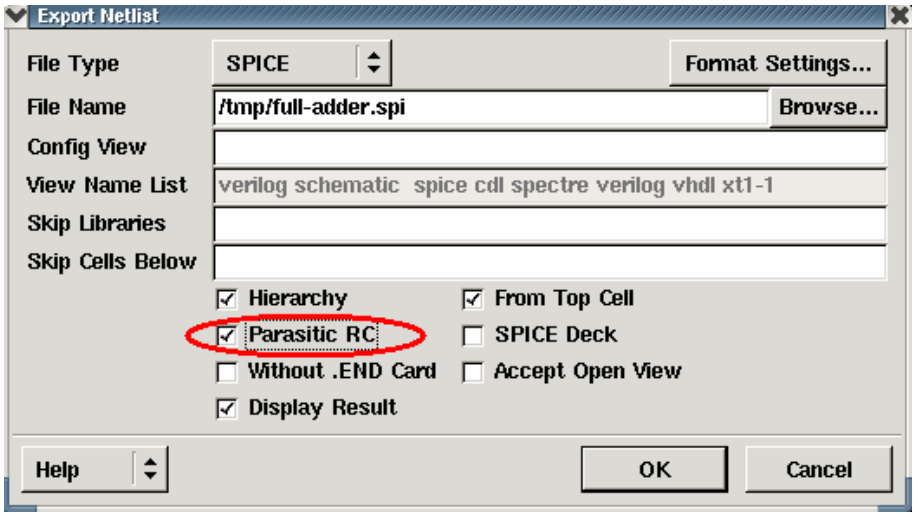
1. Select the parasitic load instance in net ‘c’ first.
2. Click ‘q’ to popup the property form.
3. In this form, you can find the instance name.
4. Close the Property form.
5. Use the same way to get another instance name.

Step 9: Hide the form and move cursor to schematic window, click at anywhere to place the coupling capacitor instance. The instance means the coupling capacitor value between net ‘c’ and net ‘z’ is 0.02pf.

Step 10: Save the design.

Step 11: Select Check->Hierarchy to check whole design first.

Step 12: Select Tools->Export Netlist. In “Export Netlist” form, turn on the option *Parasitic RC* to export parasitic R/C/Coupling C.



In exported netlist, the information of parasitic resistor, parasitic capacitor and coupling capacitor are printed as the following.

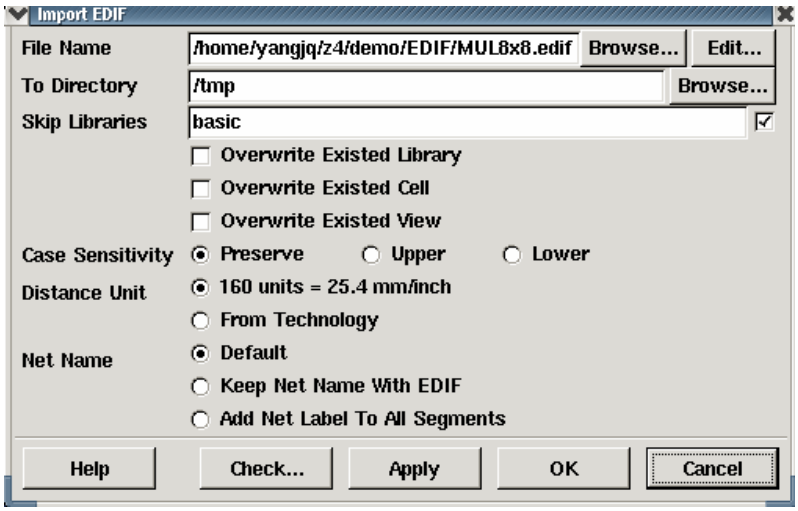
```
* parasitic res/cap/ind
RP0 c#1 c 5
CP0 c 0 0.05pf
RP1 z#1 z 1
CP1 z 0 0.01pf

* coupling cap/ind
CP2 c z 0.02pf
```

Exercise - 35 Edif In

EDIF is the abbreviation of Electronic Design Interchange Format. It is a nonproprietary, standard interchange format that uses text to describe electronic design data.

Edif-In tool translates EDIF file to schematic data. Do following steps to translate an EDIF file to schematic data.

Step 1: In Design Manager window, select File->Import->EDIF, the Import EDIF form appears.



In this form, some options need to be paid attention to.

Skip Library is a library name list. Edif-In tool will not import these libraries to schematic data. Suppose that the library which is defined in EDIF file already exists in Zeni design environment, if you don't skip it, Edif-In will overwrite the old library with the new one. In general, the "basic" library is skipped for that in Cadence environment, "basic" library includes some symbols of fundamental elements, such as, port symbol, power symbol, etc. It has the same name as the "basic" library of Zeni. If you do not skip the "basic" library, EDIF-In will recreate a "basic" library which comes from Cadence tool. It is terrible to overwrite the "basic" library of Zeni. So you'd better skip the "basic" library for safe

Distance Unit specifies the grid spacing in schematic. Select *160units=25.4mm/inch* option to keep the original spacing defined in EDIF file. Select *From Technology* to adjust the original size to fit Schematic Editor.

Net Name specifies that EDIF-In how to name a net.

-***Default*** names the net as same as the original in other EDA tool.

For example, the following is a part of an EDIF file, EDIF-In names net with "a".

```
(net (name a
       (display (figureGroupOverride wire
       (textHeight 12))
       (justify LOWERCENTER)
        (orientation R0)
       (origin (pt -130 220))))
            ....
            ....
```

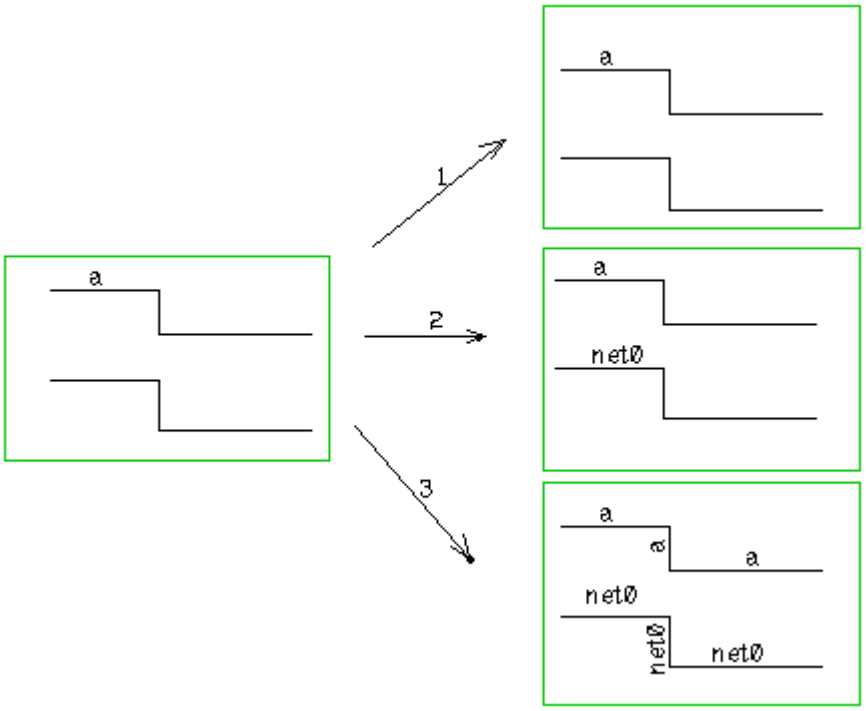
-***Keep Net Name With EDIF*** adds a default name to any segment of the net. The default name, such as net0, net1, is defined in EDIF file.

For example, the following is a part of an EDIF file, EDIF-In names net with "net0".

```
(net net0
       (joined)
       (figure wire (path (pointList
            ...
            ...
```

-Add Net Label To All Segment adds the name to all segments of the net.

The following is a figure to demonstrate the difference among 3 options.



The left figure is original schematic data in other EDA tool environment, the right 3 figures are translated to schematic data from Edif file with difference options by EDIF-In.

- The arrow 1 is for *Default* option. You can find the translated schematic data is the same as the original one.
- The arrow 2 is for *Keep Net Name With EDIF* option. You can find that EDIF-In is not only names the first net with "a" but also names the second net which no name in original schematic data with the name "net0", and places this name to any segment of the second net.
- The arrow 3 is for *Add Net Label To All Segment* option. You can find that EDIF-In is not only names the first net with "a" and places this name to all of segments of this net, but also names the second net which no name in original schematic data with "net0" and places this name to all of segments of this net.

Step 2: Fill out above form with the following information.
File: $ZENI_INSTALL_PATH/demo/EDIF/MUL8x8.edif
To Directory: to your working directory.
Skip Libraries: basic

Step 3: Ok the form.

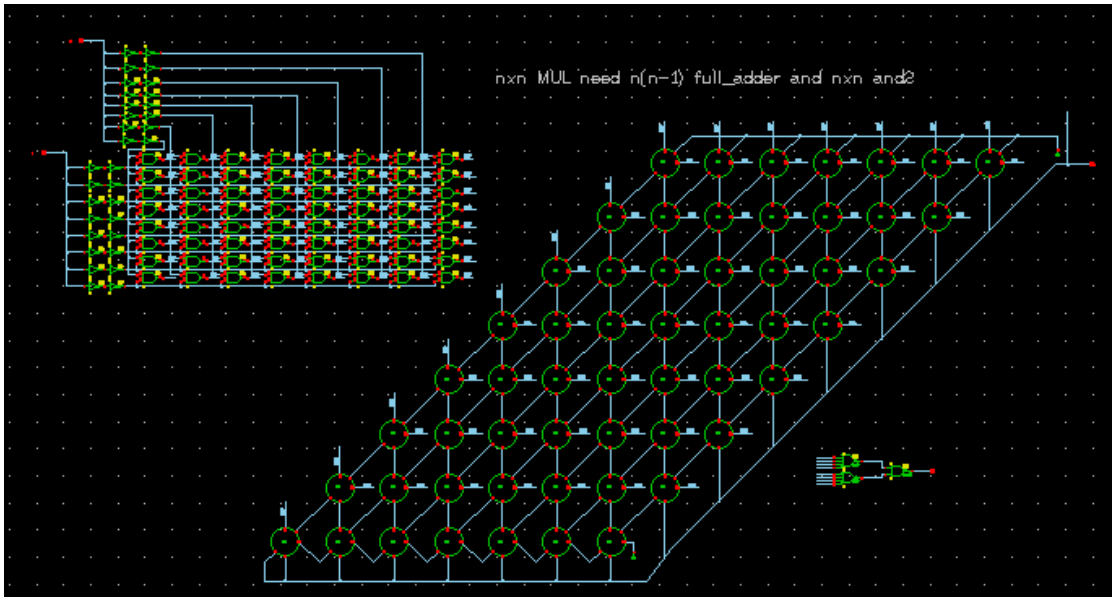
Step 4: The Zterm-edif form appears.

Step 5: Close this fome when it is over.

Step 6: In Design Manager Window, select File->Refresh Library List to refresh libraries.

Step 7: The three libraries “cdsRipLib”, “MUL8X8” and “tec” are added in library list. They were defined in EDIF file.

Step 8: Open toplevel cellview “MUL8X8.MUL8X8.schematic”, the schematic data is translated from edif file is shown as below.

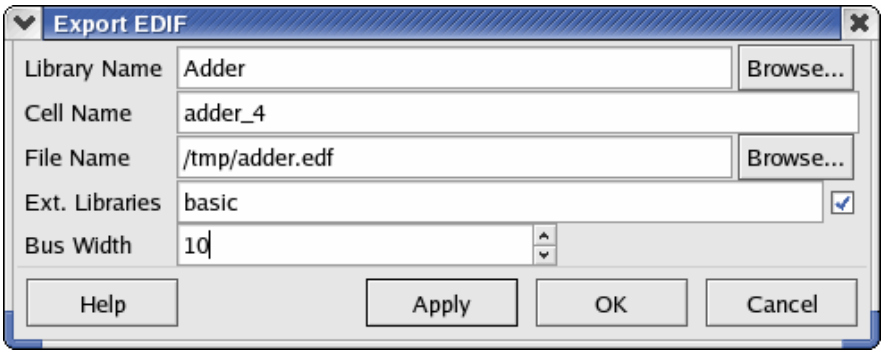


Exercise - 36 Edif Out

Edif-Out tool exports schematic data to EDIF file. You can continue your design with this file in other EDA tools. Even you can import this file to Zeni again with Edif-In if you like .

Do following steps to export cell “Adder.adder_4.schematic” to EDIF file.

Step 1: In Design Manager window, select File->Export->EDIF or directly select File->Export EDIF in schematic cellview “Adder.adder_4.schematic” window. Fill out the form with the following information shown as figure below.



In this form, Ext.Libraries option needs to be paid attention to.
Ext.Libraries is a library name list. Edif-Out tool will export this library with keywords “external” to Edif file.

For example, as the form above, skip the library “basic”, the Edif file looks like below.

```
(external basic
  (EDIFLEVEL 0)
  (technology (numberDefinition))
  (cell EGND (cellType GENERIC)
   (view symbol (viewType SCHEMATIC)
    (interface
```

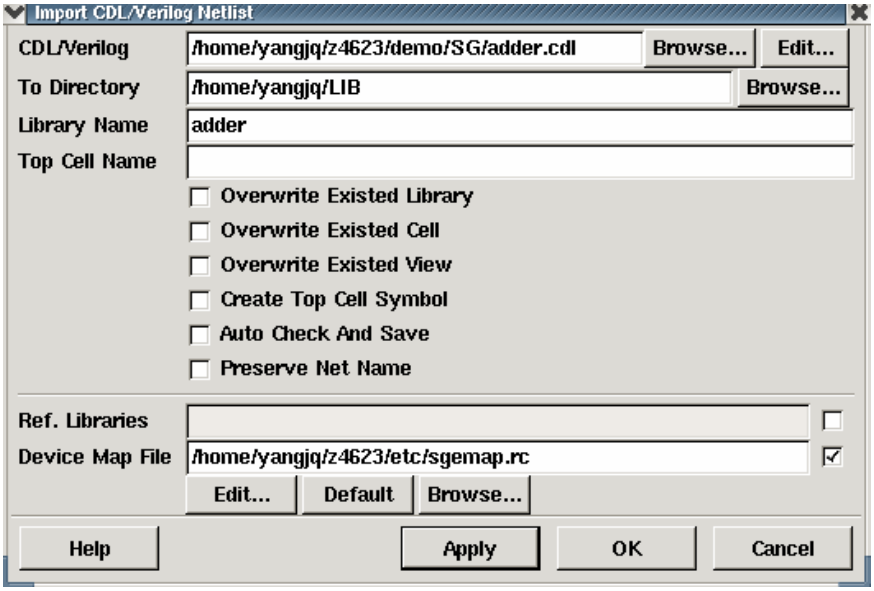
Why to make some libraries "external”? Because that if one library is marked with keyword "external" in edif file, when you edif in this file, no matter in Zeni environment or in other EDA tool environment, the library will not be translated to schematic data. Otherwise, when the library name is the same as one in Zeni or in other EDA tool, this new library will overwrite the original one in Zeni or other EDA tool if you not skip it during edif in.

Step 2: Ok the form, the edif file is exported.

Exercise - 37 CDL-In, Verilog-In

Zeni translates CDL netlist or Verilog netlist to schematic data.

Step 1: In Design Manager window, select File->Import->CDL/Verilog Netlist, the Import CDL/Verilog Netlist form appears.



About the form above,

CDL/Verilog chooses the CDL or Verilog file you want to translate.

To Directory sets a path to locate the library.

Library Name names the library that netlist is translated to.

Top Cell Name regards this cell as the toplevel cell and only translates the cell and its sub-cells to schematic data. Zeni will translate all of cells to schematic data. if the field is empty.

Create Top Cell Symbol creates a symbol view for top cell automatically while importing.

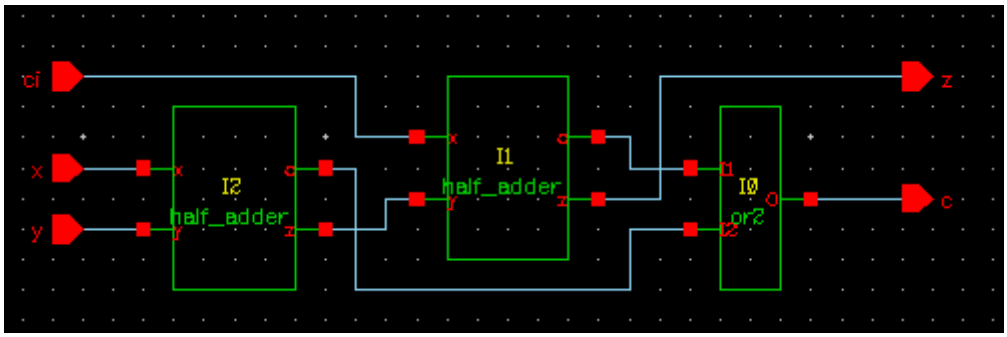
Auto Check And Save checks and save all of cellview after importing. If you turn on the option, don't need to check full design again before export netlist if you don't modify anything.

Preserve Net Name imports the net label according to the netlist.

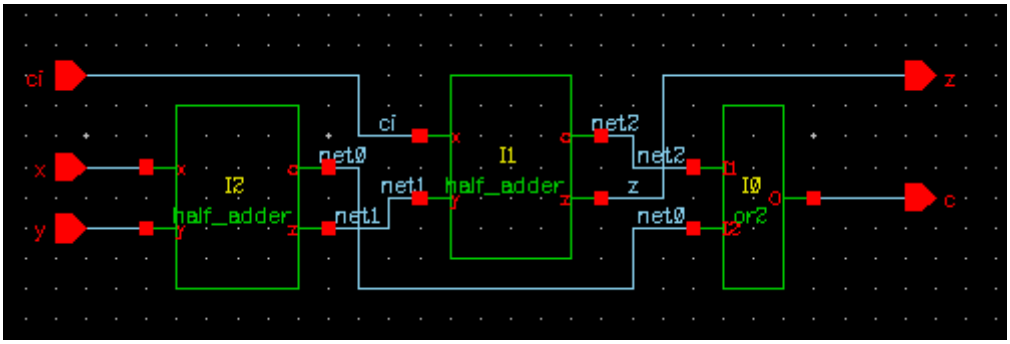
For example, a part of cdl netlist is shown as below.

```
.SUBCKT fulladder c z / ci x y
XI0 c net2 net0 / or2
XI1 net2 z ci net1 / half_adder
XI2 net0 net1 x y / half_adder
.ENDS fulladder
```

Turn off the option "Preserve Net Name", the created schematic is shown as below.



Turn on the option, the created schematic is shown as below.



Ref. Libraries specifies the reference libraries. As there is no any library information in CDL or Verilog netlist, if you turn on this option, Zeni will search cell name from these library one by one to match the cell in netlist. If matching failed or you turn off this option, Zeni will translate cell information to schematic data and create a cell in library.

Device Map File specifies a device mapping table. It is only for translate CDL netlist to schematic data.
Click *Default* button to load the default map file “sgemap.rc”. Click *Edit* button to open this file.
This file defines the mapping relation between device and global signal. Each line is a definition item.

1. Device Mapping

For example, a definition item as below.

analog.pmos.symbol M="pmos Pch" d=D g=G s=S b=B w=W l=L

Here,

- "analog.pmos.symbol" specifies the library name, cell name and view name of device.
- "M" specifies the device is a MOS.
- "pmos Pch" specifies the model list of device.
- "d=D g=G s=S b=B" specifies the mapping relation of pin name. On the left of "=“ specifies the pin of standard MOS, on the right of "=" specifies the pin name of view "analog.pmos.symbol".
- "w=W l=L" specifies the mapping relation of parameter name. On the left of "=" specifies the parameter name in CDL netlist. On the right of "=" specifies the parameter name of "analog.pmos.symbol".

In order to understand clearly, we suppose a definition item as below:

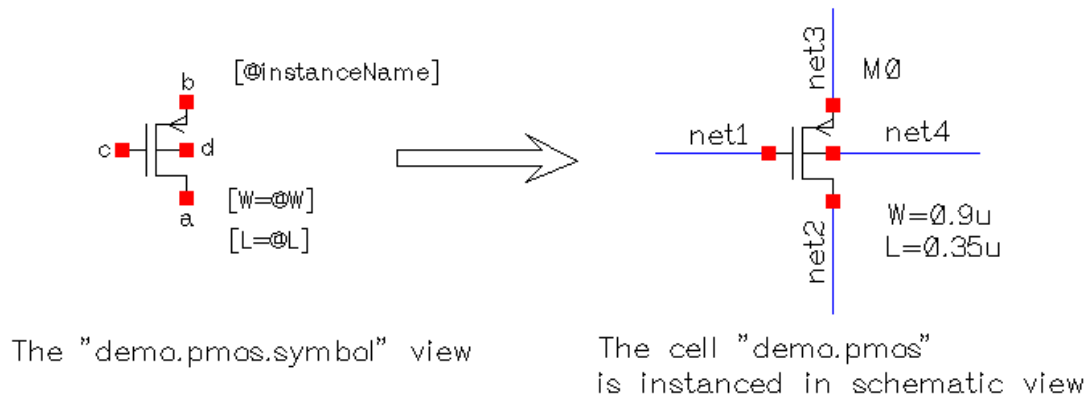
demo.pmos.symbol M="pmos Pch" d=a g=c s=b b=d w=W l=L

As the example shows above, in the CDL netlist, if the device mode is "M" and model name is "pmos" or "Pch", Zeni will reference the device from cell "demo.pmos.symbol". "d", "g", "s" and "b" represent “drain”, “gate”, “source” and “bulk” of device MOS. Zeni is not sensitive to name case on the left of "=". These four names also can be "D", "G", "S" and "B". "a", "c", "b" and "d" are the pin name of "demo.pmos.symbol”. The values of parameter "w" and "l" in CDL view will be past to parameters "W" and "L" of "demo.pmos.symbol".

Suppose that the CDL view and CDL netlist are as follow:

```
/*-------- CDL view ----------*/
>M?    <d> <g> <s> <b>        [@MODEL] [@W:W=%] [@L:L=%] [@M:M=%]
/* end */

/*------- CDL netlist --------*/
......
M0 net2 net1 net3 net4 pmos 0.9u 0.35u
......
/* end */
```

Zeni translates this netlist to schematic data as below.



The left figure above is cellview “demo.pmos.symbol”, according to device map definition “demo.pmos.symbol M="pmos Pch" d=a g=c s=b b=d w=W l=L “, the created schematic data is shown as the right figure.

2. Global signal Mapping

For example, there are two definition items:

```
demo.gnd.symbol ground="gnd! gnd 0"
demo.vdd.symbol power="vdd! vcc"
```

Firstly, Schematic Editor searches string ".gloal" in CDL netlist, if the string is not found, Schematic Editor regards '0' existed in CDL netlist as the Ground and uses "basic.gnd.symbol" in schematic view as default. If the string is found, there will be 3 cases:

1. Form as ".global xx:P", schematic regards "xx" as Power, so it searches string "power" in device map file and matches the "xx" from the power name list specified on the right of "=". If matching is successful, Schematic Editor uses the symbol specified on the left of "power" as Power symbol in schematic view. Otherwise, Schematic Editor uses "basic.vdd.symbol" as Power symbol in schematic view by default.
2. Form as ".global xx:G", Schematic Editor regards "xx" as Ground, so it searches string "ground" in device mapping file and matches the "xx" from the ground name list specified on the right of "=". If matching is successful, Schematic Editor uses the symbol specified on the left of "ground" as Ground symbol in schematic view. Otherwise, Schematic Editor uses "basic.gnd.symbol" as Ground symbol in schematic view by default.
3. Form as ".global xx", Schematic Editor matches "xx" from Power name list and Ground name list respectively. If matching is successful, Schematic Editor uses the specified symbol as Power symol or Ground symbol. Otherwise, Schematic Editor regards "xx" as a global signal, that is, Schematic Editor creates a new pin named "xx!" in schematic view.

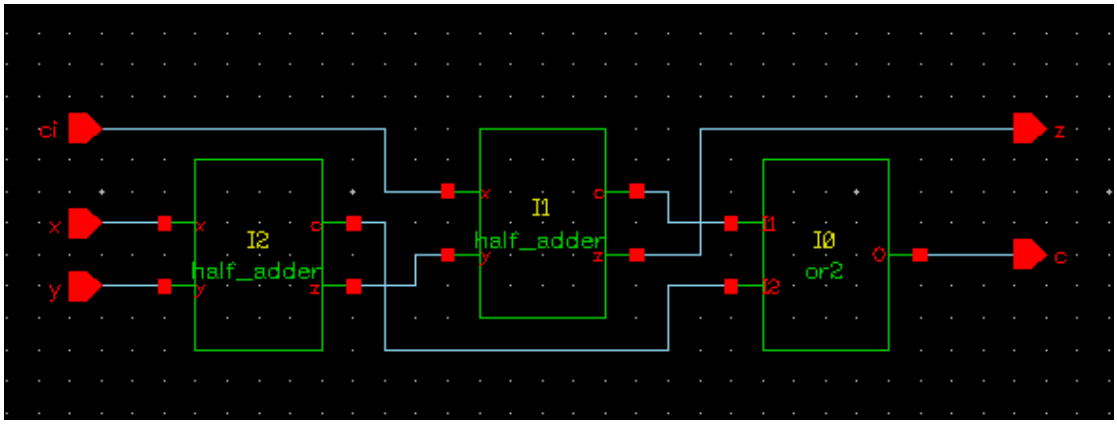
CDL-In

Step 2: Fill out the above figure with the following information.
CDL/Verilog: fill the demo file “$ZENI_INSTALL_PATH/demo/SG/adder.cdl”.
To Directory: fill your working path.
To Library: adder.
Device map file: “$ZENI_INSTALL_PATH/etc/sgemap.rc”.

Step 3: Ok the form. CDL-In begins to run.

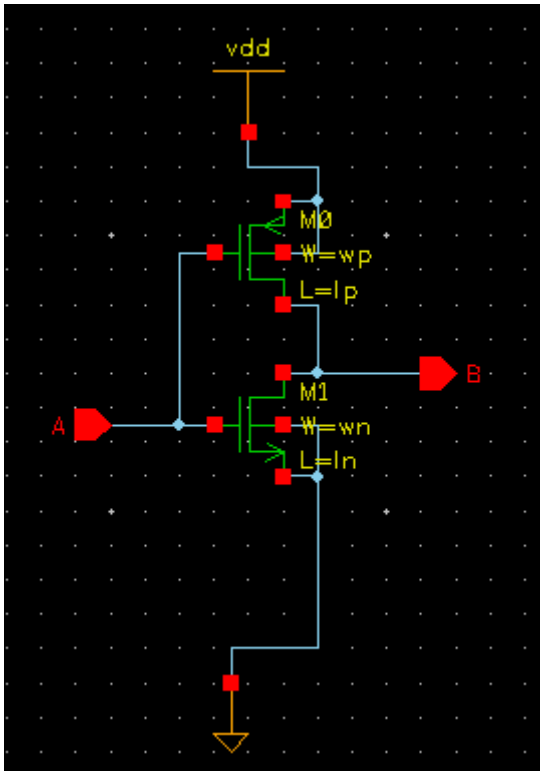
Step 4: The library “adder” appears automatically in Design Manager window after running.

Step 5: Open the toplevel cell “fulladder”. The schematic data is shown as below.



In this figure, default rectangle represents each instance shape.

Step 6: Open the cellview “adder.inv.schematic”, the cell “inv” netlist is translated to schematic data as figure below.



Step 7: Select instance “M0”, and press key ‘q’ to popup its property form.

Step 8: In property form, you can find the pmos comes from “analog.pmos.symbol” because of definition in the device map file.

Step 9: Close schematic window of “inv” and “fulladder”.

Step 10: In Design Manager, select File->Import->CDL/Verilog again.

Step 11: In the form, the information filled last time still exists. Please turn on the *Overwrite Existed Library*, the *Overwrite Existed Cell* and *Overwrite Existed View* are turned on automatically in the meantime.

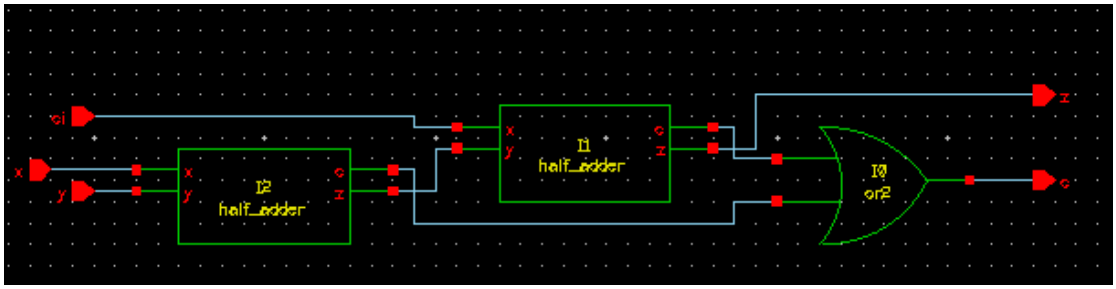
Step 12: Turn on the Ref.Library option and type “Adder” in it.

Step 13: Ok the form.

Step 14: Close the Zterm-sge form.

Step 15: In Design Manager window, select library “adder”, click middle/right button and choose “Refreash” from popup-menu.

Step 16: Open the schematic view of cell “fulladder”, the schematic window appears as below.



This figure illustrates the instances of cell “or2” and “half_adder” come from the library “Adder”. Because the “Adder” is a reference library and it has two cells “or2” and “half_adder”. Zeni directly uses these cells to construct the schematic view.

Step 17: Close the cellview.

Verilog-In

Step 18: As the step 2, modify the Import CDL/Verilog Netlist form with the following information.

CDL/Verilog: $ZENI_INSTALL_PATH/demo/SG/adder.v.
To Directory: working path.
To Library: adder.
Overwrite Existed Library: on
Overwrite Existed Cell: on
OverWrite Existed View:on

Step 19: Ok the form. All cells included in verilog netlist are created in library “adder”, you can open top cell “fulladder” to view it.

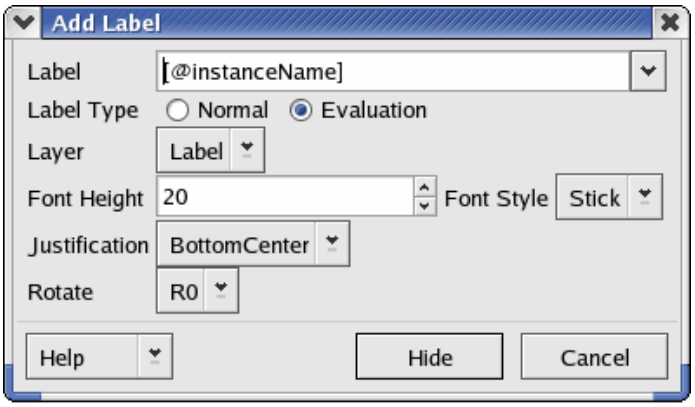
6.3. Symbol Editor

In Symbol Editor, most of the commands are the same as those in Schematic Editor. Here, we mainly introduce some different ones.

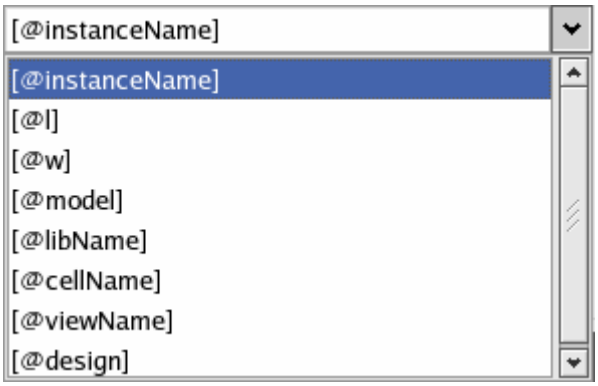
Exercise - 38 Add->Label

This command lets you add parameters text to the symbol.

Select Add->Label or click icon  , "Add Label" form appears.



In label dropdown-list, Symbol Editor pre-defines some labels as figure below.



These parameter labels start with '@'. When you add this symbol cellview into the Schematic Editor, Schematic Editor replaces this parameter name with its value when you turn on *Evaluation* option in "Add Label" form, otherwise, Schematic Editor regards it as normal string.

Layer dropdown-list includes Sheet, Primitive, Device and Lable. This option is only used to change the color of label. These 4 colors are specified in *Options->Color.*

Exercise - 39 Add->Selection Box

This command lets you define an area. This area is used that when you add this symbol cellview to Schematic Editor, Schematic Editor highlights the symbol as long as your cursor is in this area. If you click cursor in this area, this symbol (instance) will be selected. The default selection box is a rectangle that encloses all of the pins and symbol shapes, except Note text.

Exercise - 40 Add->Import Symbol

This command lets you import an existed symbol cellview to current Symbol Editor window.

6.4. Layout Editor

6.4.1. Import demo libraries

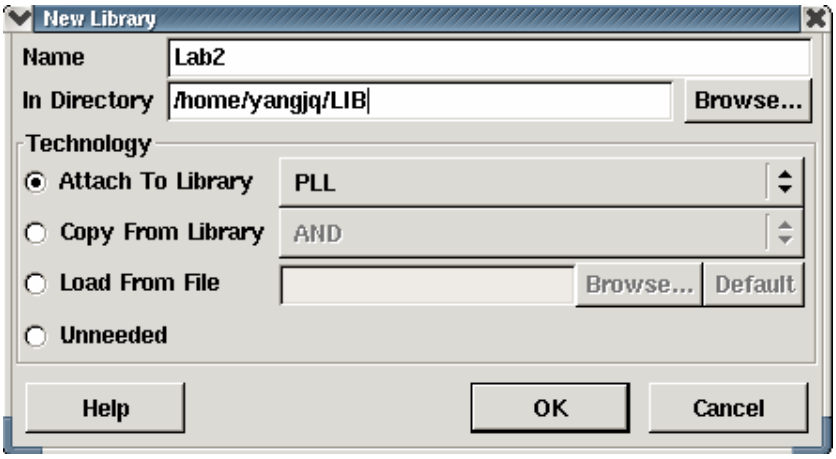
Please refer to *6.1.1 Import demo library* on page 61 to import some demo libraries to Design Manager window.

1. PLL=$ZENI_INSTALL_PATH/demo/tutorial/PLL_demo/PLL
2. PadText=$ZENI_INSTALL_PATH/demo/PDT/PadTextLib/PadText
3. GenArray=$ZENI_INSTALL_PATH/demo/PDT/GenArrayLib/GenArray

6.4.2. Create a library

Step 1: In Design Manager window, select File->New->Library.

Step 2: Fill out Name field with “Lab2” and In Directory field with your working path in New Library form.



Step 3: Use the technology file of library “PLL” as its technology file.

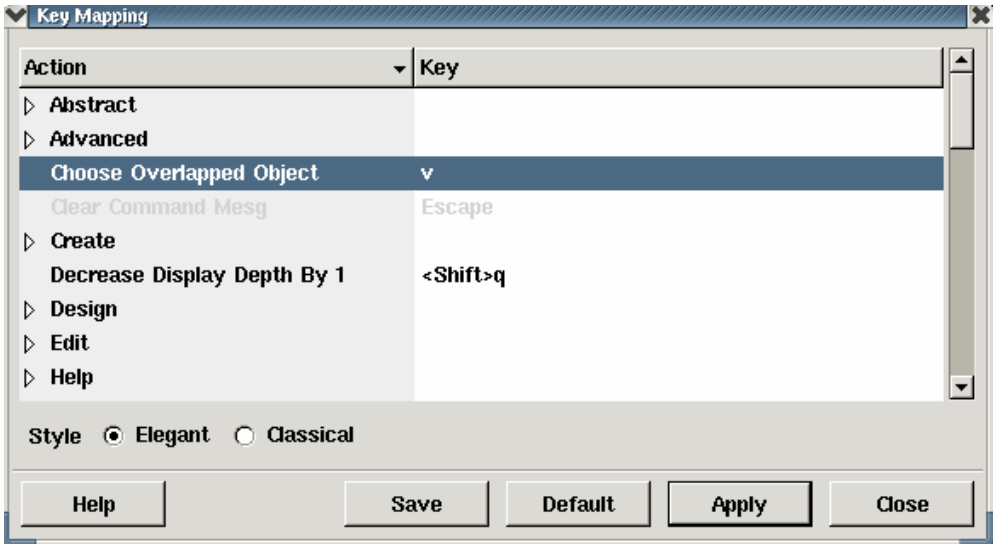
Step 4: Ok the form.

6.4.3. Working With Basic

Exercise - 41 Exactly Select Overlapped Object

Sometimes objects are overlapped each other, which will take trouble to exactly select one of the objects. ZeniPDT lets you press a key to select an object. The system sets the key "v" as default.

You can also modify it by Key Mapping command. The *Key Mapping* form is as below.



To exactly select an overlapped object, you should

Step 1: Slide the cursor in overlap area, ZeniPDT will automatically highlight the nearest object from the cursor.

Step 2: Press "v" key to switch the highlighted object you want to select.

Step 3: Click the left mouse button to select the highlighted object.

Exercise - 42 Locating Precisely

You can set the precise location by input the coordinates when you create, copy or move an object in layout window.

When you create an object, you should

Step 1: Activate a create command.

Step 2: Enter the coordinates of starting point in it. An absolute coordinate can be specified by using one of the following syntax.

(a) (x y), or (x,y), or (x:y)
(b) [x y], or [x,y], or [x:y]
(c) {x y}, or {x,y}, or {x:y}
(d) <x y>, or <x,y>, or <x:y>

Multiple points can be separated by spaces or comma, such as (1 2) (3.5 4), or (1 2), (3.5 4). A relative coordinate can be specified by turning on the Use Relative Coordinate option or by placing a '@' sign at the front of the x or y coordinate. Either of x,y coordinate is specified as relative, the expected point will be classified as relative point. For example, (@0 5) = (0 @5) = (@0 @5).

Step 3: Press "*Enter*".

Step 4: Move the cursor in layout editor window, the cartoon object will follow your cursor.

When you move or copy an object, you should

Step 1: Select an object first.

Step 2: Activate the move or copy command.

Step 3: Left-click at a point in the object and move cursor, a small yellow square around the point you click appears in the cartoon object. The point is a reference point of the object.

Step 4: Press *"Shift+9"* and input the coordinate of reference point you want to place finally.

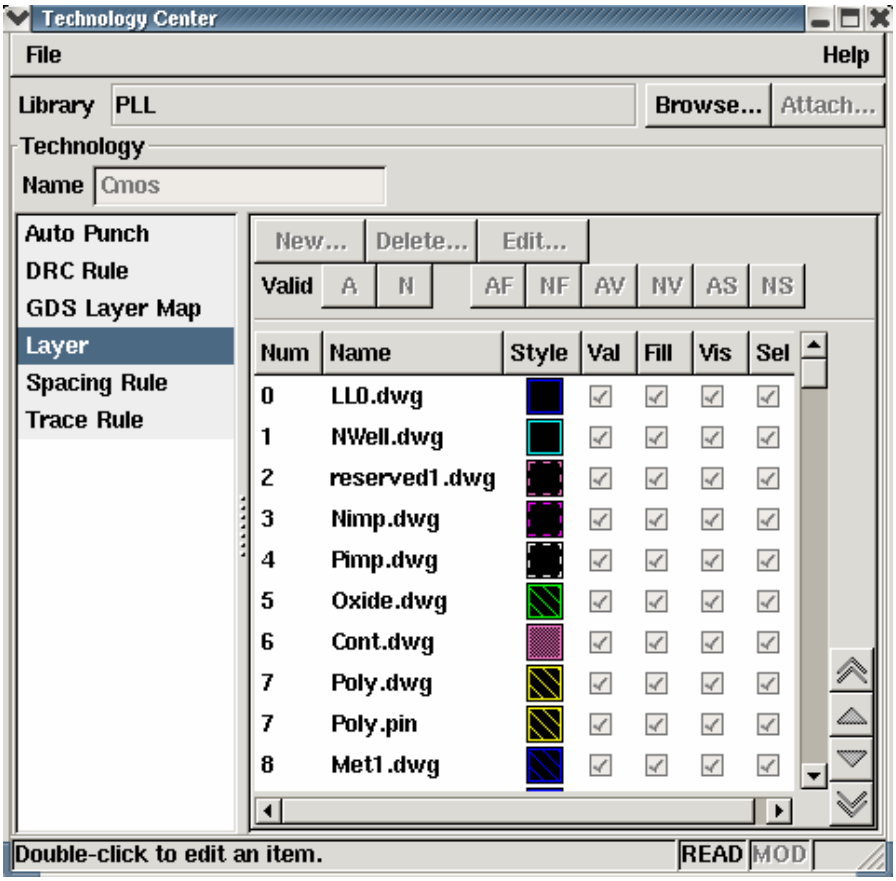
Step 5: Press "*Enter*".

6.4.4. Basic Exercises

Exercise - 43 Technology Center

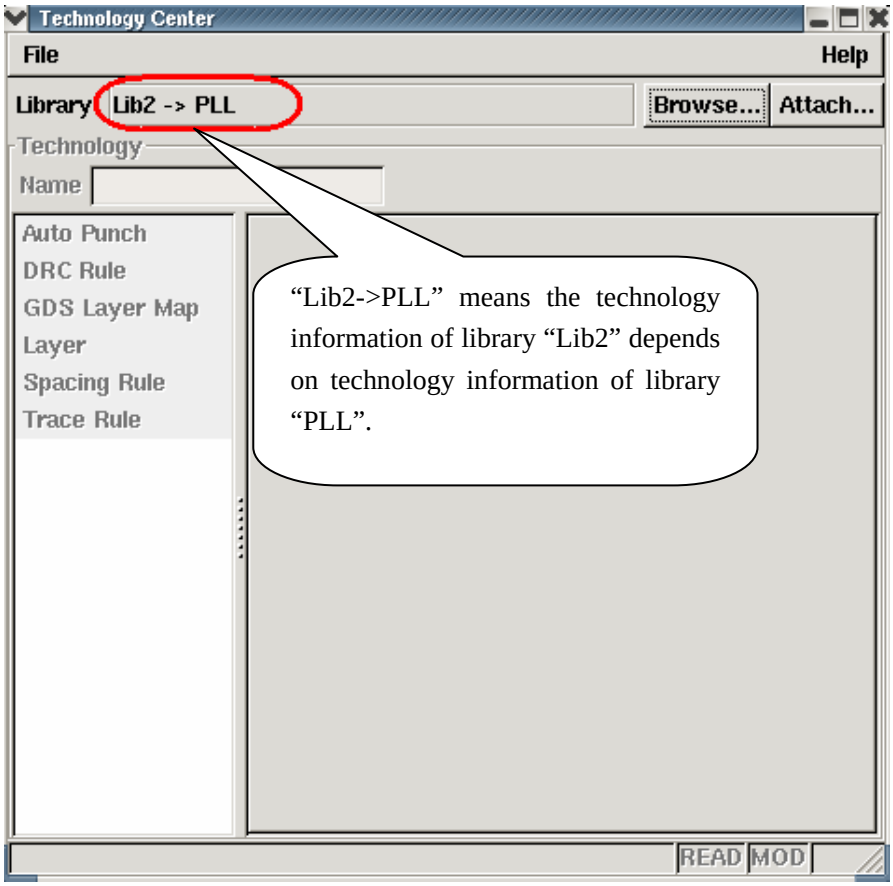
Technology Center edits and displays user-specified vendor technology.

In Design Manager window, click Tools->Technology Center. Technology Center window appears as below.

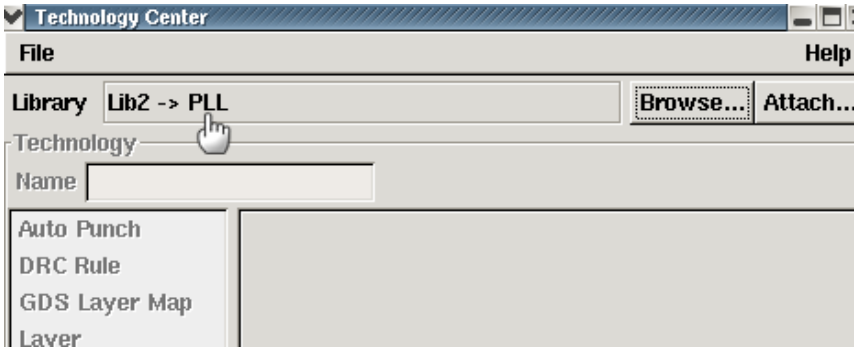


This window contains all technology information of each library. Library drop-down list lists all of libraries. Choose one library from it, technology information related to the library will be displayed in this window at once.

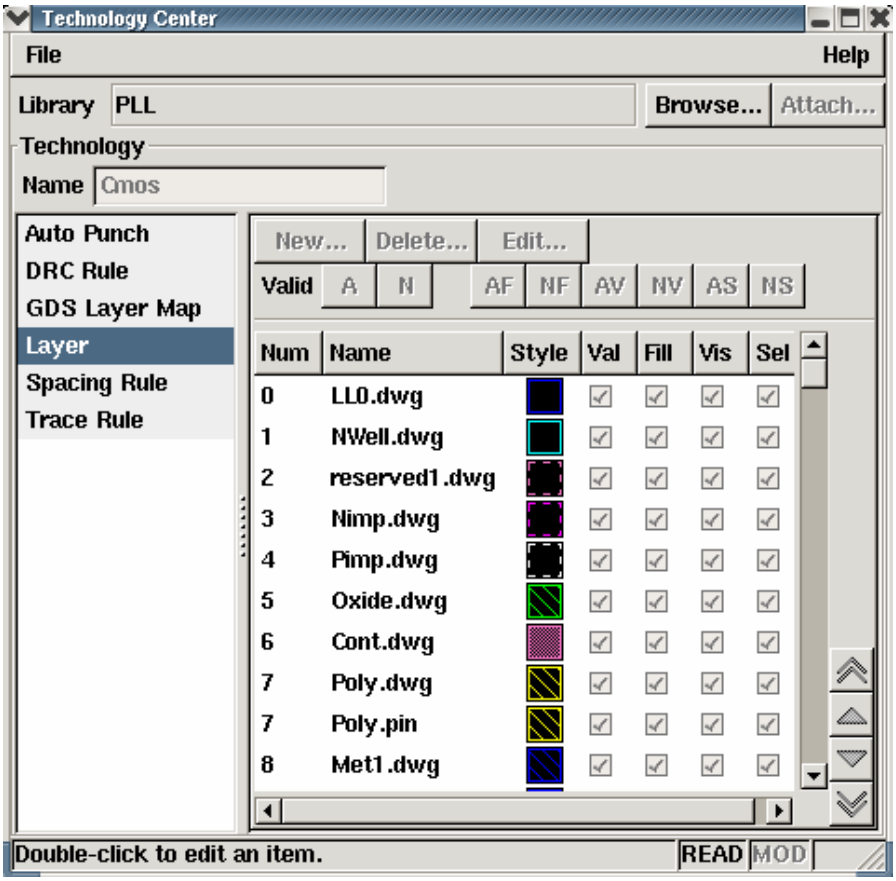
In addition, if the technology information of a library is linked to another library's, such as library "Lib2" created by *6.4.2* on page 125 , the information is shown as below.



When cursor moves to near string "PLL", a small hand shape appears.



Click it, the technology information of library "PLL" is shown as below figure shown.



On the left side of the Technology Center window, six kinds of information are shown. Here, information of Auto Punch, Layer, Spacing Rule and Trace Rule are stored in technology file "tech.dat", DRC Rule and GDS Layer Map's are stored in "drc.rule" and "gds.map" respectively. These three files locate under path "<library_name>/.technology"

Click one item on the left side of window, related information will be shown on the side of the window.

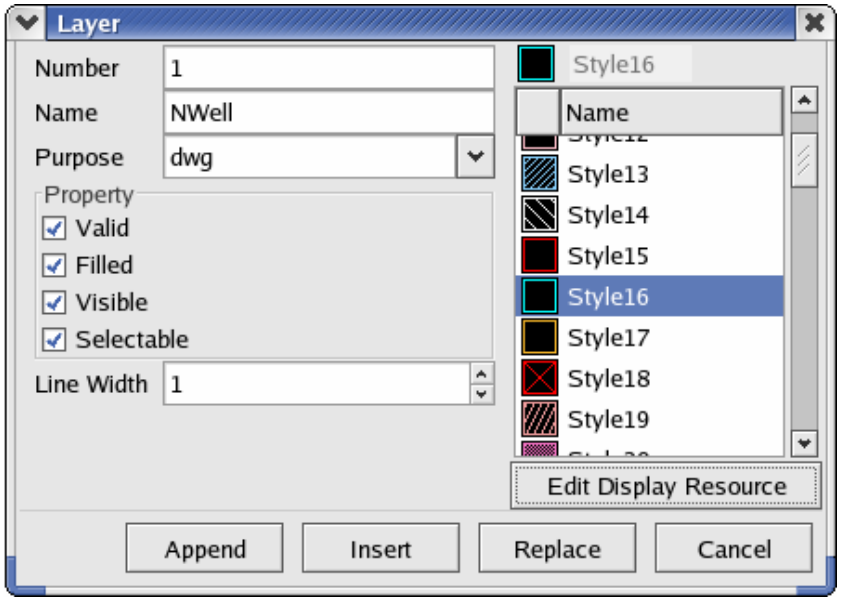
About the six kinds of technology item,

- **Auto Punch** specifies rules to add via cells at the cross of two paths automatically. Please refer to *Exercise - 88 Auto Punch* on page 205.
- **DRC Rule** specifies rules for real-time Design Rule Check (DRC). Please refer to *Exercise - 90 Realtime DRC* on page 209.
- **GDS Layer Map** lets you re-define the layer name and layer number when current layer number is different from the one defined in GDS file of foundry.
- **Layer** specifies layer number, layer name and display rendering.
- **Space Rule** specifies rules for the minimal external spacing between two geometries. Please refer to *Exercise - 89 Automatically Adjust Spacing* on page 208.
- **Trace Rul**e specifies rules for inter-connection detection between nodes. Please refer to *Exercise - 74 Advanced->Trace Net* on page 174.

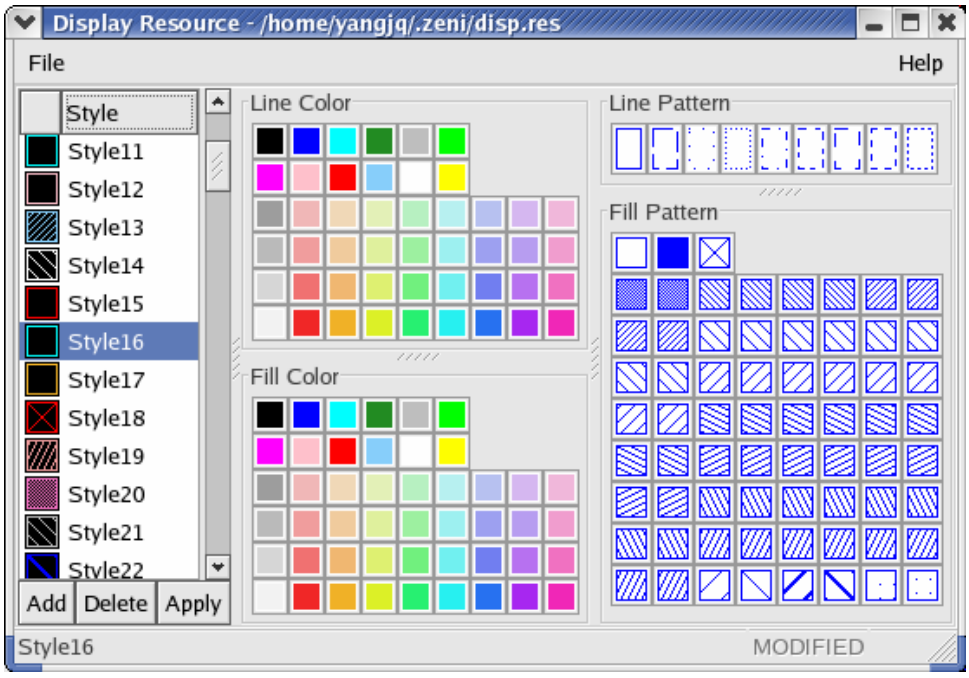
As the figure above, click Layer item on the left side of window, all layer definitions of library

“PLL” are shown on the right side of window.

Double click the second item, Layer form appears.



You can modify or create another layer definition in the above figure.
Here, Style specifies the layer rendering. You can select another layer style as you need. A style is a set of color, fill, line pattern, and it is created by Display Resource. Please refer to *Exercise - 44 Display Resource* on page 132. You can click Edit Display Resource button to edit the selected style in figure below.

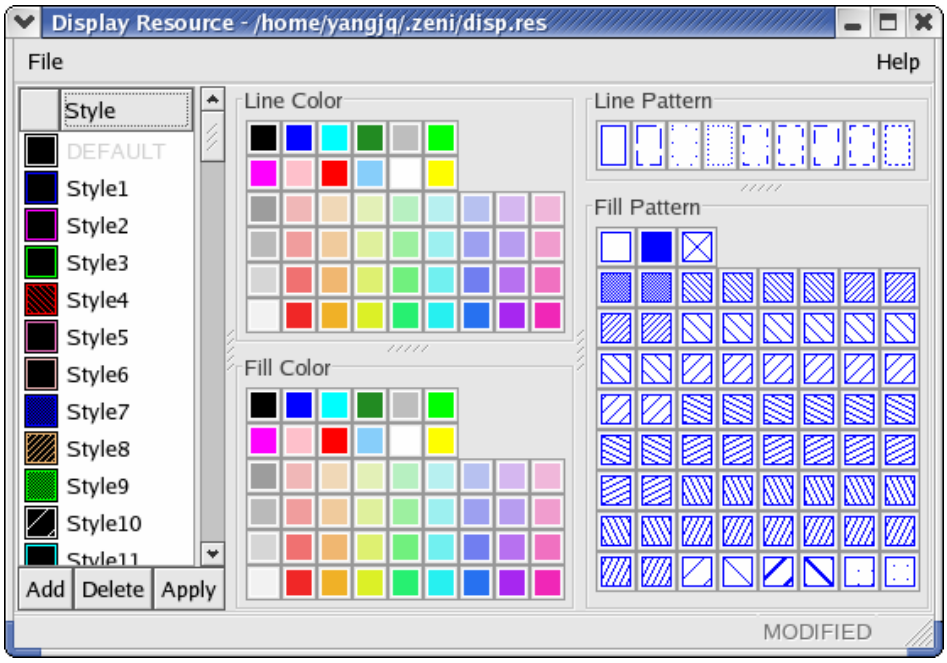


About the File menu in Technology Center window,
File->Save stores technology information to file “tech.dat” under path “<lib_name>/.technology”.
File->Reload aborts any modification and reloads technology information from file “tech.dat”.
File->Import->Zeni ASCII Tech File imports an ASCII technology file created by File->Export and replace current technology file.
File->Import->Zeni Binary Tech File imports a binary technology file of Zeni and replace current technology file.
File->Import->Zeni1.x/2.x Tech translates technology file of Zeni old version to current one.
File->Import->Cadence ASCII Tech File imports an ASCII technology file created by Cadence tool and replace current technology file. Meanwhile, display resource file of Cadence tool is imported too and is merged into display resource file of Zeni if you fill out the Display Resource File field in Import Cadence Technology form.
File->Export exports an ASCII technology file.

Exercise - 44 Display Resource

This command supplies resource for layer definition in Technology Center.

In Design Manager window, click Tools->Display Resource, the form appears as below.



This form displays all existed style names and resources of Line Color, Fill Color, Line Pattern and Fill Pattern. A style is a group of these four resources.
Some operations about style:

- Double click the style name to rename.
- Right-click the style name or icon in front of the name to delete the style or add a new style at the bake of the selected style.

About File menu,

File->Save stores the resources to file “disp.res” under path “$HOME/.zeni” or “$ZENI_USER_HOME/.zeni”.

File->Merge merges another display resource file into current one.

File->Reload re-reads file “disp.res” and replace current one.

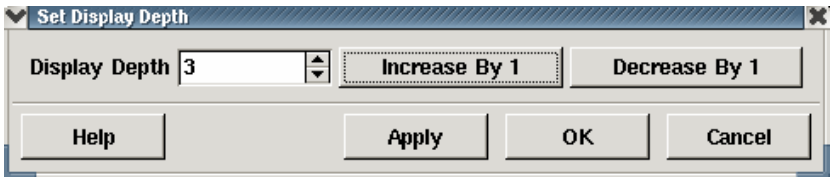
File->Load Default loads file “disp.res” under path “$ZENI_INSTALL_PATH/etc” and replace current one. 65 styles are default.

Exercise - 45 Design->Save Markers

This command lets you save marker to binary file “maker.dat” in directory “../libname/cellname/viewname/current”. The marker is created by commands Layout Verification, Load Report, Trace Net or Find. The command “Browse Marker” translates the marker file to internal database of Zeni and shows markers on Browse Marker form. Please refer to
Exercise - 85 Tools->Browse Marker on page 194.

Exercise - 46 Window->Set Display Depth

This command specifies the depth of image shown in the current window.



If you set “3” in the above form, Layout Editor will display layout images from the toplevel to the 3rd level. “0” represents toplevel.

Step 1: Open layout cellview “PLL.PLL.layout”, the toplevel images are shown only.

Step 2: Select Window->Set Display Depth, or press “Alt+F” key, type “2” in Set Display Depth form. You can find layout images are shown from toplevel to the 2nd level.

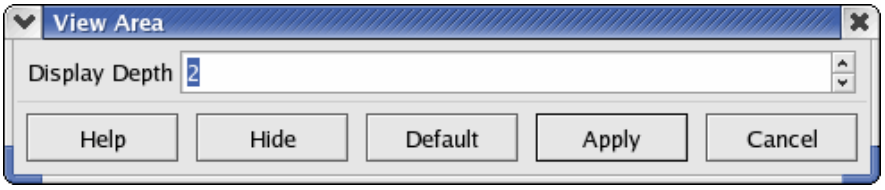
Note:

1. This command is different from *Options->Generic->Display->Initial Display Depth*. The *Initial Display Depth* specifies how deep image is shown when you open a layout cellview. The depth number is stored in “[display]” section of file “le.rc”.

2. Press “Ctrl+F” to display full hierarchy images; press “Shirt+F” to display the toplevel images only.

Exercise - 47 Window->View Area

This command lets Layout Editor display image in an area with hierarchy. How deep image is shown depends on the Display Depth number in View Area form.



Step 1: (Omit this step if this cellview is opened) Open layout cellview “PLL.PLL.layout”.

Step 2: Press “Shift+F” key to display toplevel images.

Step 3: Select Window->View Area->Set and set “2” to View Area form.

Step 4: Click Hide button.

Step 5: Select an area in current window, the images from toplevel to the 2nd level in this area will be displayed.

Step 6: Select Window->View Area->Delete to recover original view status.

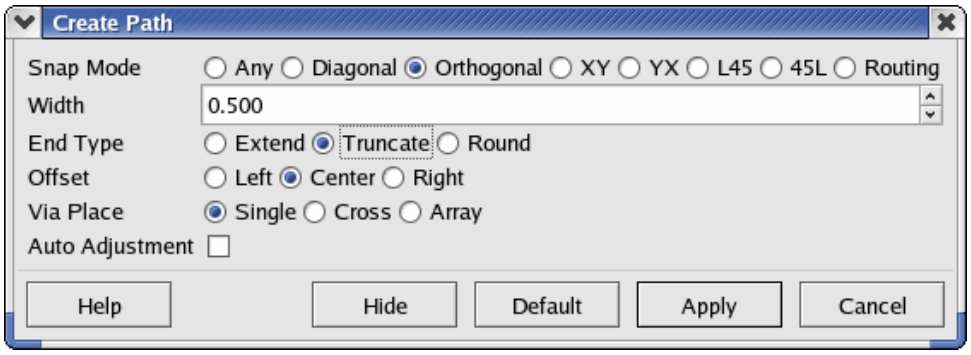
Exercise - 48 Window->Raise Design Manager

This command raises Design Manager window to the top of the screen. It’s very convenient for you to return to Design Manager window directly from schematic editor window or layout editor window. ‘2’ is default bindkey.

Exercise - 49 Create->Path

This command lets you create path.

Select Create->Path or click icon , Add Path form appears (if the form doesn’t appear, please press “F3”)



About the form above,

Snap Mode specifies the moving track while creation path.

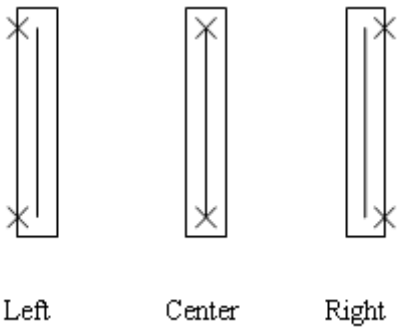
Width specifies the path width.

End Type specifies how the path ends are created.

- ***Extend*** specifies the path ends extend from the path points by one half the path width.
- ***Truncate*** specifies the path ends and path points end at the same point.
- ***Round*** specifies the path ends by an arc. The radius of arc is half the path width.

The following three path ends show path end type.

Offset specifies which edge of the path is offset from the line you create.
For example, the following figure shows three offset modes.

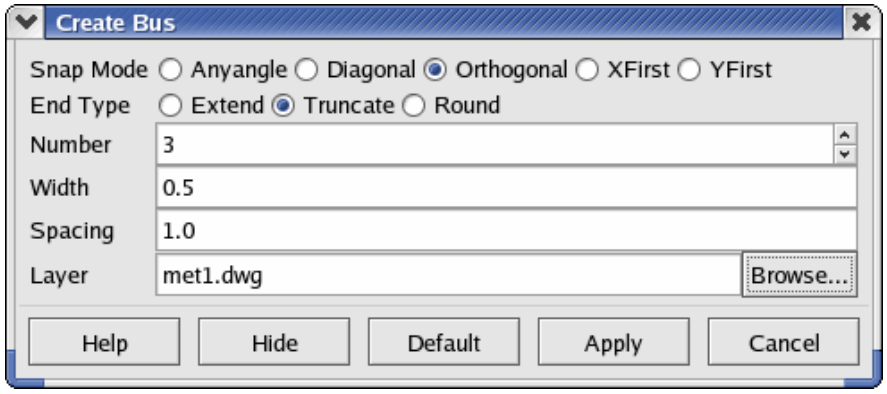


Via Place is used to add via cell automatically at the cross of paths. Please refer to *Exercise - 88 Auto Punch* on page 200 for more details.

Auto Adjustment is used to adjust two paths' distance to meet DRC rule. Please refer to *Exercise - 89* on page 208 for more details.

Exercise - 50 Create->Bus

This command let you create multiple paths at one time.



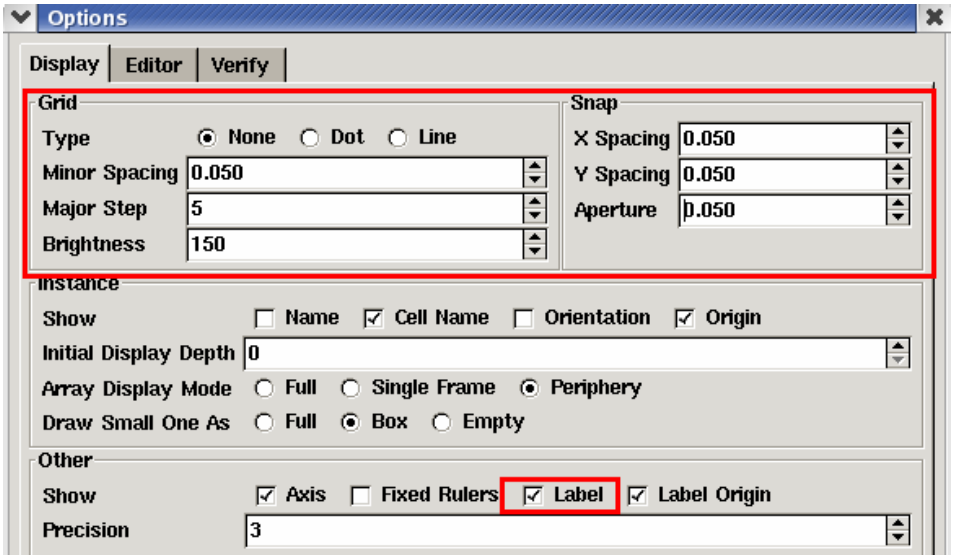
About the form above,
Number specifies the path number you want to create.
Width specifies each path width.
Spacing specifies the distance of two paths.
Layer specifies the layers each path used. Click *Browse* button to select layers. In Layer form, you can select multiple layer names by pressing "Ctrl + left mouse button".

Exercise - 51 Create->Label

This command let you create text in the cellview.

Step 1: Create a new cellview "Lab2.crtlabel.layout".

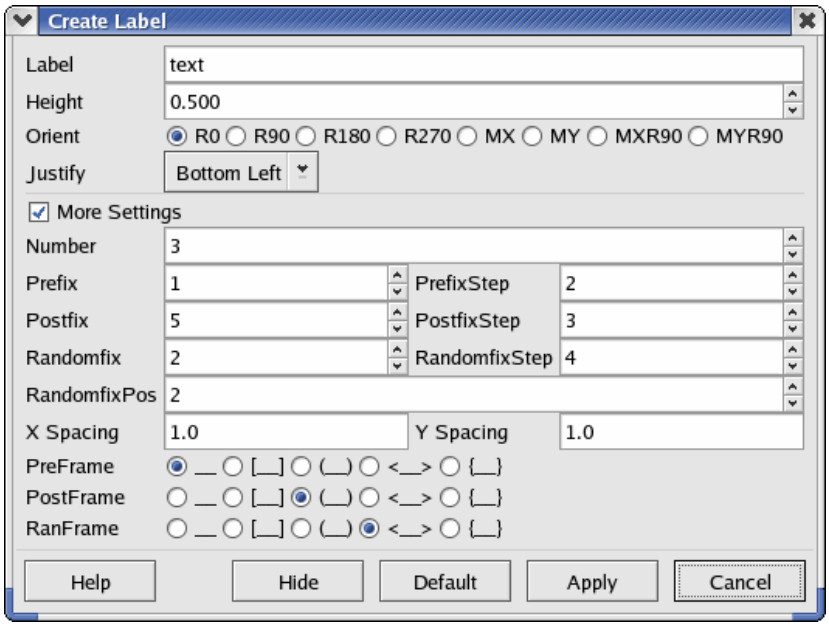
Step 2: In Layout Editor window, click Options->Generic menu to modify spacing value and make sure the show label option is turned on in Display tab.



Step 3: Ok and save the form.

Step 4: In Layout Editor window, select “Met1.dwg”from the Layer Palette on the left side of the editor window. The Layer Palette looks like the below figure.

Step 5: Select Create->Label or click icon  , and fill out the Create Label form with the information as figure below.



About the form above,
Label specifies the text you want to create.

Height specifies the text height. The number that greater than 0.0 and less than or equal to 2000.0 is valid.

Orient specifies the orientation of the text.

Justify specifies the location of the label origin. The origin appears as a square on the label when you place the label.

More Settings specifies some advanced settings, such as creating a label array.

-***Number***: the number of label. 1~1024 is valid.

-***Prefix***: the number as a prefix of the label. -1~1024 is valid. '-1' means system don't add any number in front of the label.

-***PrefixStep***: the step of prefix number.-1024~1024 is valid.

-***Postfix***: the number as a postfix of the label. -1~1024 is valid. '-1' means system don't add any number at the back of the label.

-***PostfixStep***: the step of postfix number. -1024~1024 is valid.

-***Randomfix***: a number is added among the label string. The position of the label string is specified by RandomfixPos. -1~1024 is valid. '-1' means don't add any number to the label.

-***RandomfixStep***: the step of Randomfix.-1024~1024 is valid.

-***RandomfixPos***: if adding a number to the label, RandomfixPos specifies the position of the label. The position number is greater than or equal to -1 and less than or equal to the

length of the label. For example, if the length of the label is 6, set the RandomfixPos to 4, it will add Randomfix inumber at the back of the 4th letter of labels. '-1' means system don't add any number to the label.

-X Spacing: the distance of two labels in X direction.

-Y Spacing: the distance of two labels in Y direction.

-PreFrame: the frame of prefix.

-PostFrame: the frame of postfix.

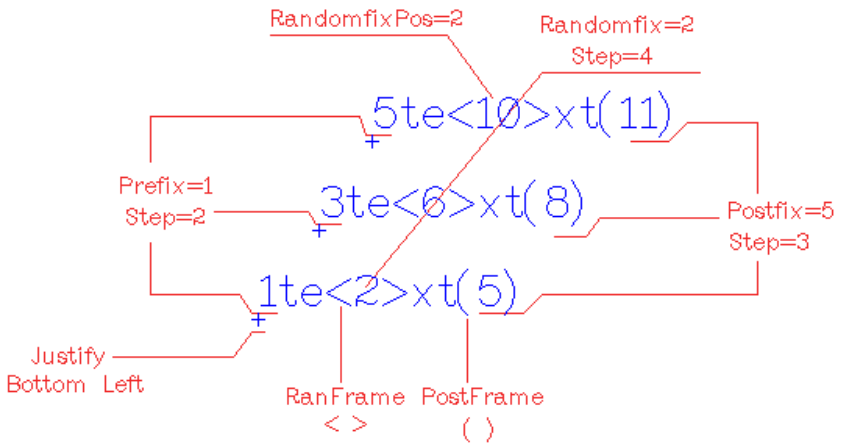
-RanFrame: the frame of randomframe.

Step 6: Apply and Hide this form.

Step 7: Use stroke (Please refer to *4 Working With Basic*) to magnify an area till you can see the label attached the cursor clearly.

Step 8: Click at a point on the Layout Editor window to place these labels.

The label “text” added in cellview is shown as the figure below.



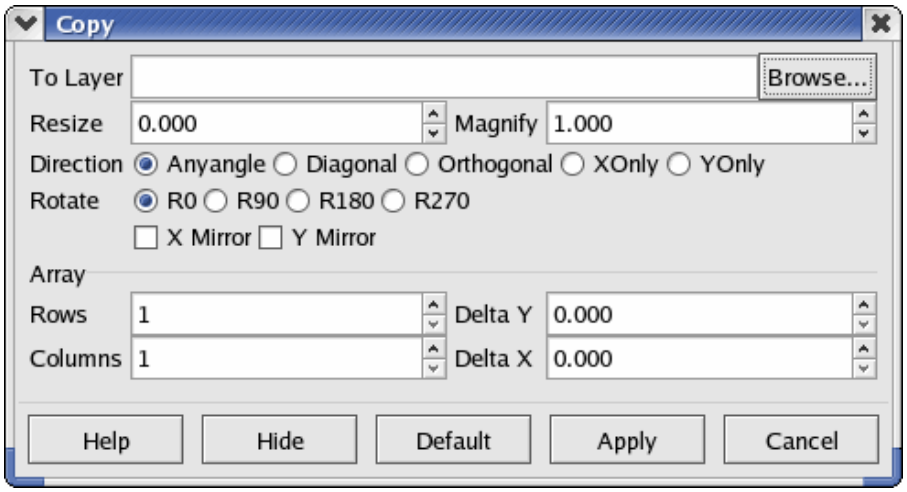
Exercise - 52 Create->VCell

This command automatically creates foundational device cells through translating vcell template file. Please refer to *Exercise - 87 VCell* on page 200 for details.

Exercise - 53 Edit->Copy

This command let you copy objects in the same cellview or to another cellview. You can also copy object from a read-only cellview to another writable cellview.

Click Edit->Copy and press “F3” to popup the related form.



About the form above,

To Layer lets you copy objects to another layer.

Resize increases or decreases the size of destination object. If you set it to “2”, system increases 2 micron to destination object compared with original one; if set it to “-2”, the system decreases 2 micron to destination object compared with original one.

Magnify lets you magnify or shrink destination object. If you set it to 2, the size of destination objects is twice as original one; if set it to 0.5, the size of destination object is half of the original one.

Direction controls the direction in which you can move the duplicate object.

Rotate controls rotation of duplicate object.

Rows and **Columns** let you create an array of duplicate objects.

Delta Y and **Delta X** specify the spacing between array objects.

Note:

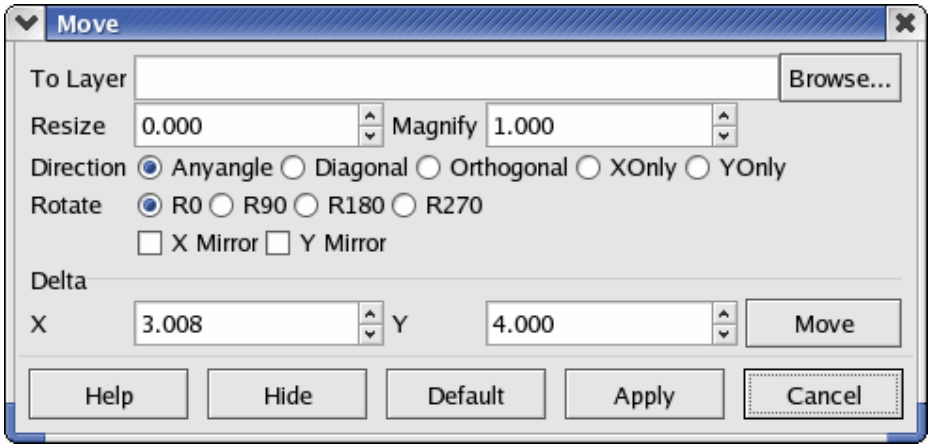
1. If you turn on the option Options->Generic->Editor->Keep selected, you can copy more objects continuously without selecting each object again before next copy.

2. An easy way to rotate duplicate object is that right click the duplicate object before placing it.

Exercise - 54 Edit->Move

This command let you move objects in the same cellview or to another cellview.

Click Edit->Move and press “F3” to popup the related form.



About the form above,

To Layer lets you move objects to another layer.

Resize increases or decreases object in size. If you set it to “2”, system increases 2 micron to the object compared with original one; if set it to “-2”, system decreases 2 micron to the object compared with original one.

Magnify lets you magnify or shrink the object. If you set it to 2, the object size is the twice than original one; if set it to 0.5, the object size is half of the original one.

Direction controls the direction in which you move object.

Rotate controlst the way rotate the moved object.

Delta X and **Delta Y** specify the relative position of moved object compared with original position. After fill out the value of both them, you need click **Move** button to implement it.

Note:

1. If you turn on the option Options->Generic->Editor->Keep selected, you can move more objects continuously without selecting each object again before next moving.

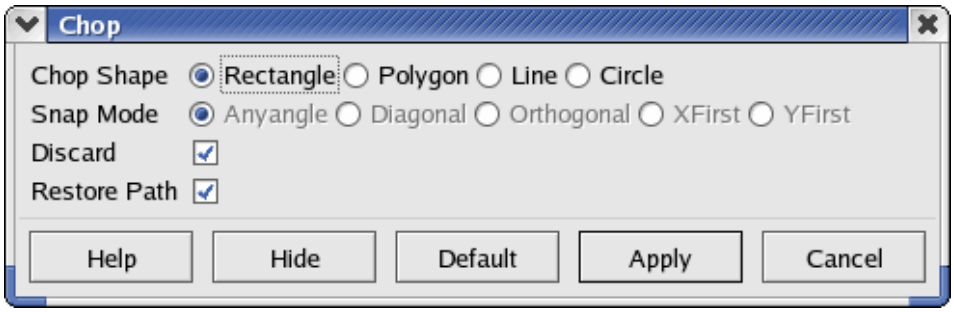
2. An easy way to rotate moved object is that right click the object before placing it.

Exercise - 55 Edit->Chop

This command lets you remove a selected portion of objects.

Step 1: Select object first.

Step 2: Click Edit->Chop and press “F3” to popup related form.



About the form above,

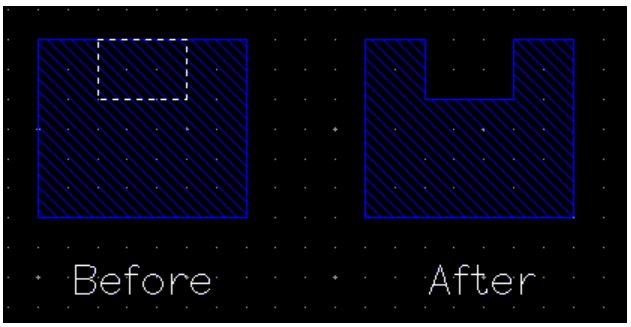
Chop Shape specifies the shape of chop.

Snap Mode is used only when chop shape is set to *Polygon* or *Line*. It controls the direction of tangent.

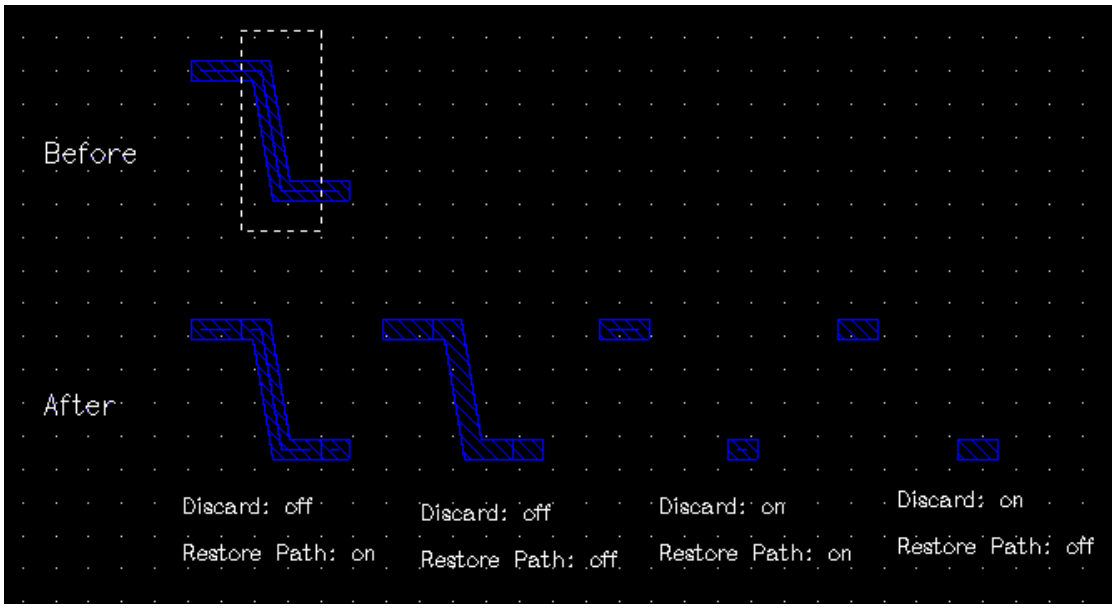
Discard specifies whether to discard extracted objects or not.

Restore Path specifies that if the object to be chopped is a path, whether to keep path property or not when tangent direction is horizontal or vertical.

Example 1: Chop a rectangle.



Example 2: Chop a path.



Exercise - 56 Edit->Stretch

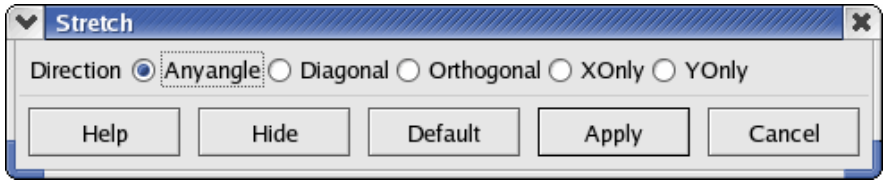
This command lets you stretch edges or corners of an object.

Do following steps to stretch objects.

Step 1: Select corners or edges, the selected portion are highlighted.

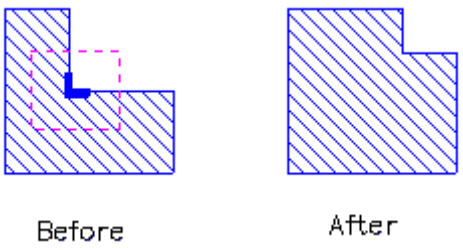
Step 2: Click at a point as reference point

Step 3: Move the pointer to stretch the corners or edges. The moving direction depends on options in Stretch form.

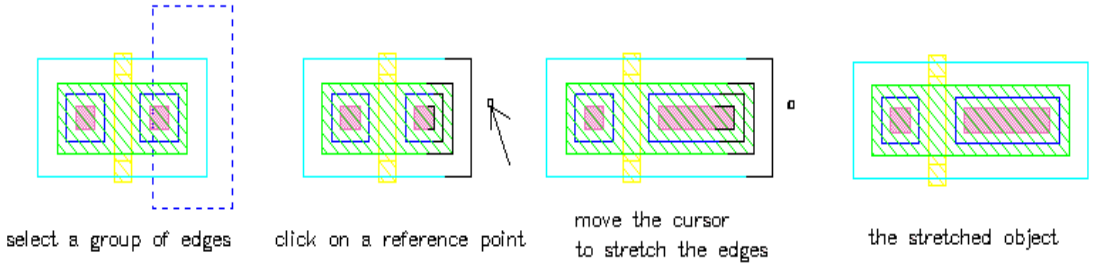


Step 4: Release the mouse button.

Example 1: stretch corner



Example 2: stretch edges



Exercise - 57 Edit->Reshape

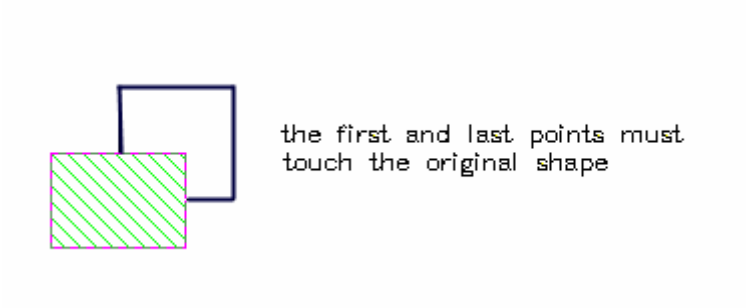
This command lets you change the shape of selected rectangles, polygons and paths.

Do following steps to reshape rectangles or polygons.

Step 1: Select the rectangle or polygon first.

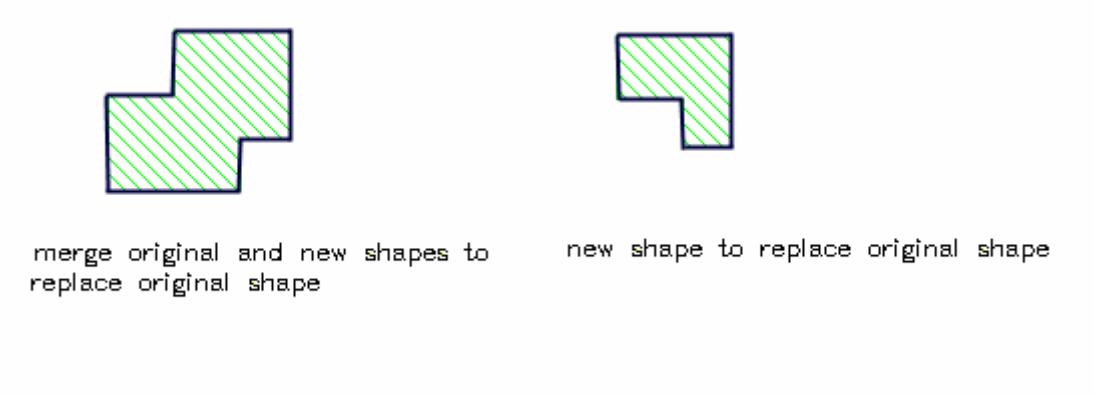
Step 2: Click Edit->Reshape or click icon  .

Step 3: Click at a point on the edge of selected object and move mouse to create the new section of the object.



Step 4: When the last point touches the edge of object, the new shape is highlighted.

Step 5: Right click to toggle between highlighting the new shape and highlighting both the original and new shape.



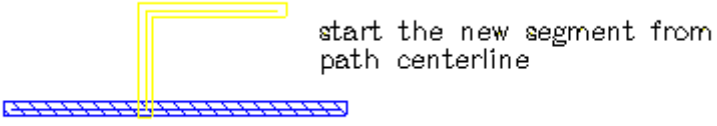
Step 6: Left click when the shape you want is highlighted. Then you will get the new shape.

Do following steps to reshape a path.

Step 1: Select the rectangle or polygon first.

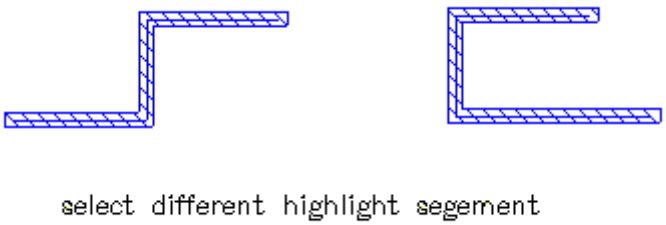
Step 2: Click Edit->Reshape or click icon  .

Step 3: Click at the centerline of path to create new section.



Step 4: Double click when you finish entering points.

Step 5: Right click to toggle the highlighted path segment.



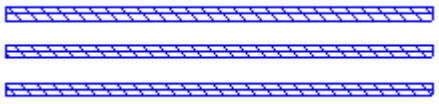
Step 6: Left click when the shape you want is highlighted. Then you will get the new shape.

Exercise - 58 Edit->Split

This command let you split a section of the paths and stretching the splited section.

Step 1: Create a new cellview “Lab2.splitbus.layout”.

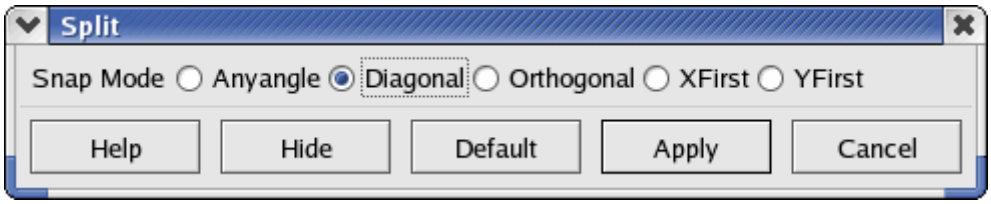
Step 2: Click icon to create 3 paths with width.0.5. Please refer to *Exercise - 50 Create->Bus* on page 136.



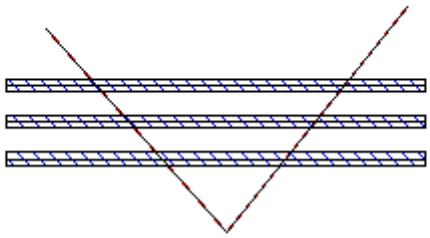
Step 3: Select all of paths first.

Step 4: Choose Edit-> Split.

Step 5: Press F3 to popup related form. Select *Diagonal* in below form.



Step 6: Click a point to create a split line that crosses each path twice.

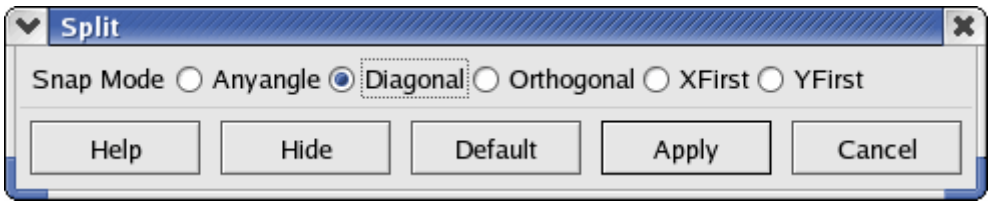


create a split line at a 45-degree angle through the segments to prevent the path segments overlapping when you stretch them

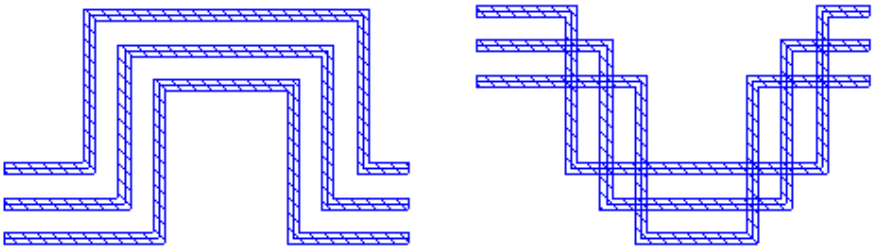
Step 7: Double click at the last points of the split line.

Step 8: Right click to toggle highlighted section.

Step 9: Click on a point and stretch highlighted section along direction specified in *Split* form. Of cause, you can also modify the direction in this form.



Step 10: Left click where you want to stretch the segments.



select different highlight section and stretch them upwards

Step 11: Save and close the cellview.

Exercise - 59 Edit->Find

This command lets you find instance, label and object recursively throughout a design hierarchy.

Step 1: Create two new cellview “Lab2.find1.layout” and “Lab2.find2.layout”.

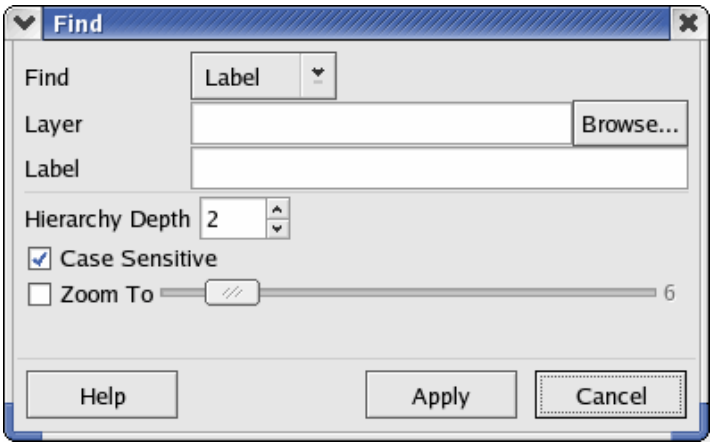
Step 2: In “Lab2.find1.layout”, create a label “label1” with “Met1.dwg”.

Step 3: Save the cellview.

Step 4: In “Lab2.find2.layout”, add instance of cellview “Lab2.find1.layout” first, then create a label “label2” with “Met2.dwg”. The figure is shown as below.

Step 5: Save the cellview.

Step 6: In “Lab2.find2.layout”, select Edit->Find. Find form appears. Set Find to Label, the figure is shown as below.



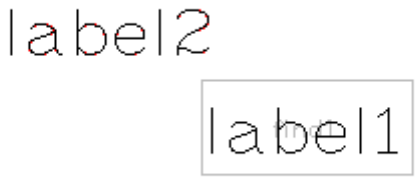
About the form above,

Layer lets you fill layer name that label belongs to. If you don’t fill it, system will search label on all layers.

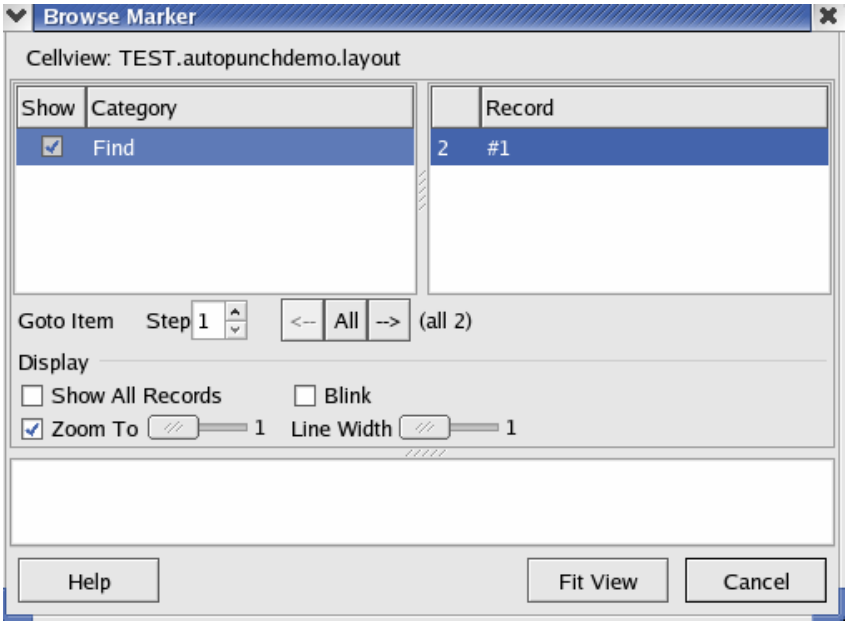
Label lets you fill the label name you want to search. If you don’t fill it, system will search all labels on specified layer.

Hierarchy Depths specifies system finds label within design hierarchy depth you defined.

Step 7: Don’t fill Layer field and Label field, click Apply directly. All labels on any layers will be found out as figure shown below.



You can also click Tools->Browse Marker （*Exercise - 85　Tools->Browse Marker* on page 194） or click icon  to look over the found labels from Browse Marker form below.



In Browse Marker form, left side lists the marker is created by Find command. Right side lists 2 markers are created. Click "->" to look over each marker in layout editor window.

If you want to delete the markers, you can

1) Right-click the Find item in Browse Marker form, choose Delete from popped up menu.
2) Click Tools->Delete All Markers.

Step 8: Close the cellview.

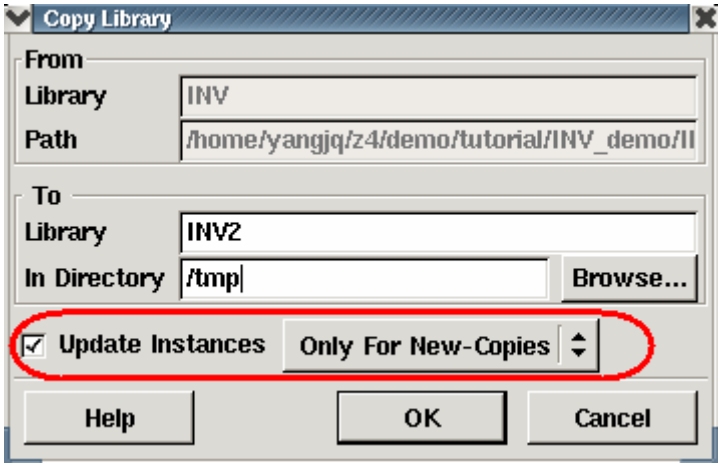
Exercise - 60　Edit->Replace

This command lets you not only replace instance and property, but also find the invalid cell which is invoked by another cell but the cell was deleted before.

If you want to find an invalid cell which is invoked by another cell but the cell was deleted in your design library, you can do the following:

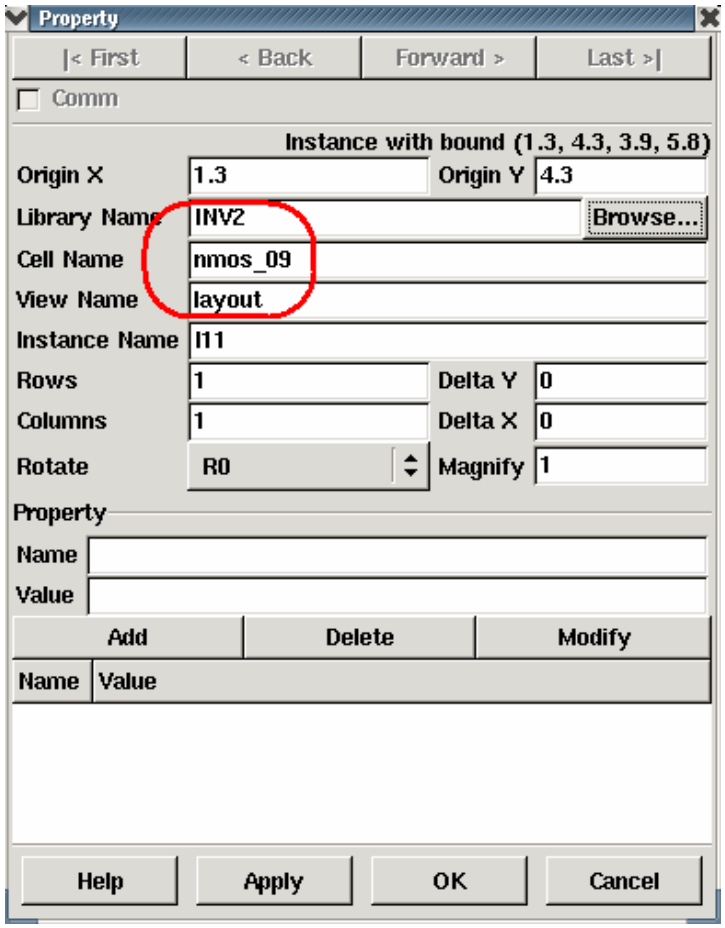
Step 1: As *5.1 Importing demo library: INV* on page 7, import the library INV (jump if you have already imported)

Step 2: Copy library “INV” to “INV2”. The copy form setting as below figure shown. Please note that you need to turn on the option “Update Instances”.



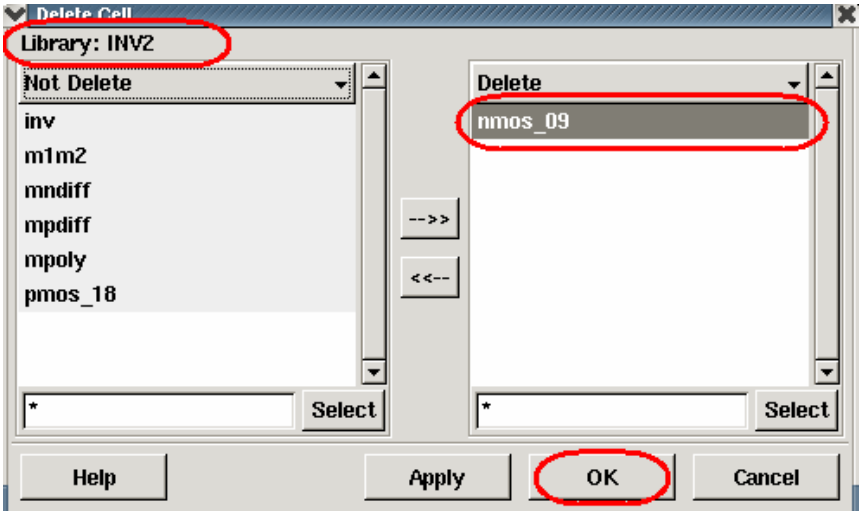
Step 3: Open layout cellview “INV2.inv.layout”.

Step 4: Select instance cell “nmos_09” in layout window, and press key “q” to popup the Property form. Please make sure the cell “nmos_09” comes from library “INV2”.



Step 5: Close layout cellview “INV2.inv.layout”.

Step 6: In Design Manager window, delete cell “nmos_09” from library “INV2” deliberately.

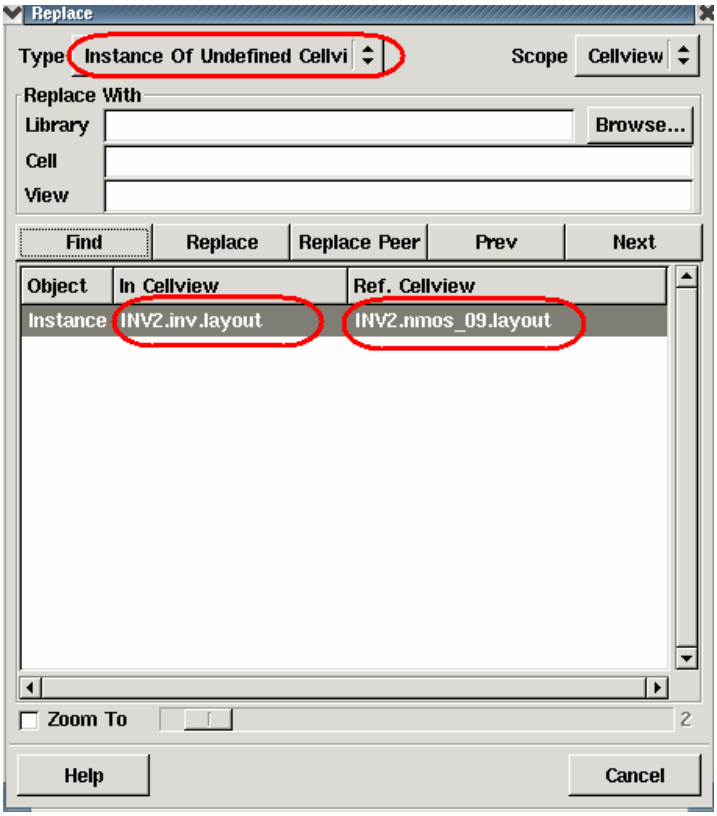


Step 7: Open again layout cellview “INV2.inv.layout”.

Step 8: Select Edit->Replace. In Type drop-down menu, choose “Instance Of Undefined Cellview”.

Step 9: In Scope drop-down menu, choose “Cellview”.

Step 10: Click “Find” button directly. The result of searching will be shown in following figure.

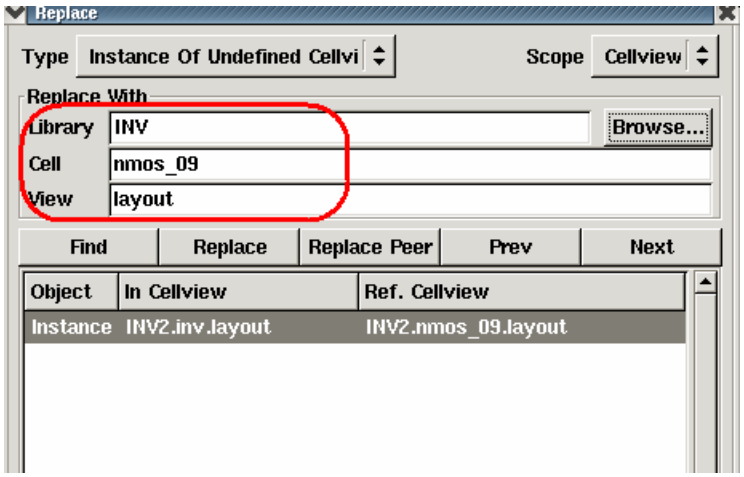


Step 11: As the above figure shown, cellview “INV2.inv.layout” invokes cellview “INV2.nmos_09.layout”, but the cellview “INV2.nmos_09.layout” was deleted just now..

Step 12: According to this information, you can replace the instance with a valid cellview.

Step 13: In Replace With section, click Browse button.

Step 14: In Design Browser form, from left to right, select INV, nmos_09, layout.



Step 15: Click Replace button to replace the instance cell "INV2.nmos_09.layout" with "INV.nmos_09.layout".

Note:

What's the difference between "Replace" and "Replace Peer"?

--"Replace" only replaces the cellview you selected with a valid cellview.

--"Replace Peer" replaces all cellviews that are same as the cellview you selected with a valid cellview.

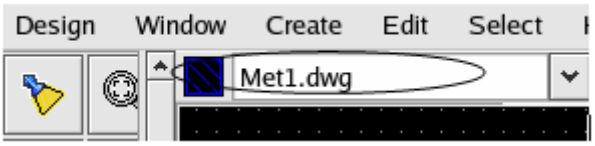
Exercise - 61 Edit->More->Change Layer

This command lets you change selected objects from one layer to another.

Step 1: Select objects first.

Step 2: Click Edit->Move->Change Layer or click icon .

Step 3: System changes selected objects to current active layer directly. Current active layer is shown as figure below.



Exercise - 62 Edit->More->Convert To Polygon

This command lets you convert selected rectangles, paths, circles and arcs to polygons.

Do following steps to convert path or rectangle to polygon.

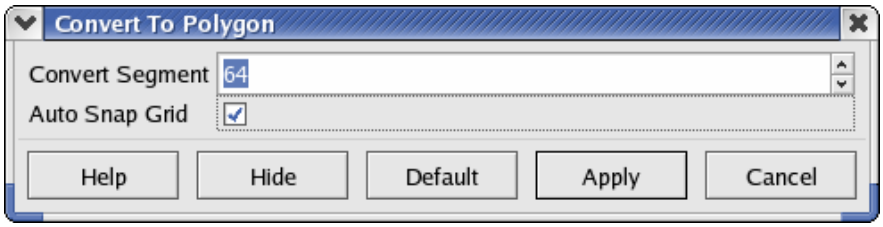
Step 1: Select a path or (and) a rectangle object.

Step 2: Click Edit->More->Convert To Polygon or click icon  .

For circle and arc, you can define how many edges the polygon has in Convert To Polygon form.

Step 1: Click Edit->More->Convert To Polygon or click icon  first.

Step 2: Press F3 to popup the related form and fill out the Convert Segment field in the form. 6~128 is valid. 64 or a value set last time is default.

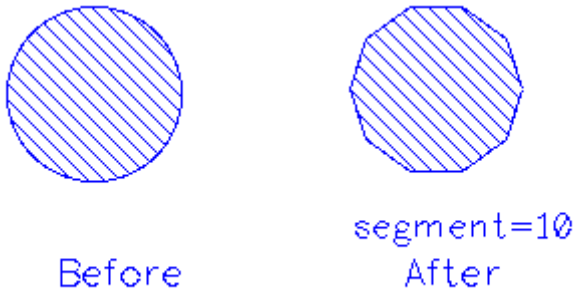


Step 3: If you want to snap polygons on grid automatically after converting, please turn on option Auto Snap Grid.

Step 4: Hide the form.

Step 5: Click objects you want to convert.

The following figure illustrates converting circle to polygon with 10 segments.



Exercise - 63 Edit->More->Connect Path

This command makes two paths merge. The final path inherits the property of the first path, including path layer, path width, etc.

To connect two paths, do the following.

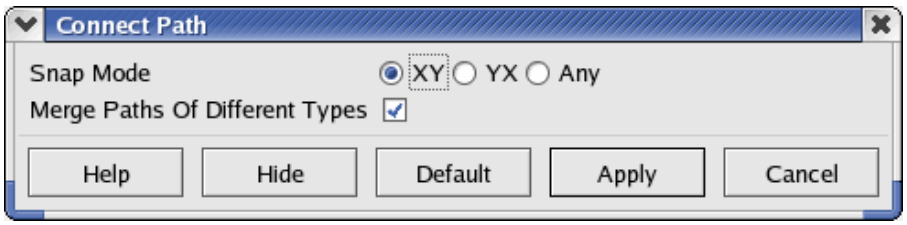
Step 1: Create a new cellview “Lab2.cntpath.layout”.

Step 2: Zoomin window to see grid clearly.

Step 3: Create a path which layer is Met1.dwg; width is 0.5; end type is Extend.

Step 4: Create another path which layer is Met3.dwg; width is 0.7; end type is Truncate. The two paths are shown as below figure (1).

Step 5: Click Edit->More->Connect Path, press F3 to popup related form as below.



About the form above,

Snap Mode specifies the connected path direction.

Merge Paths Of Different Types controls whether to merge two paths with different path width and end type.

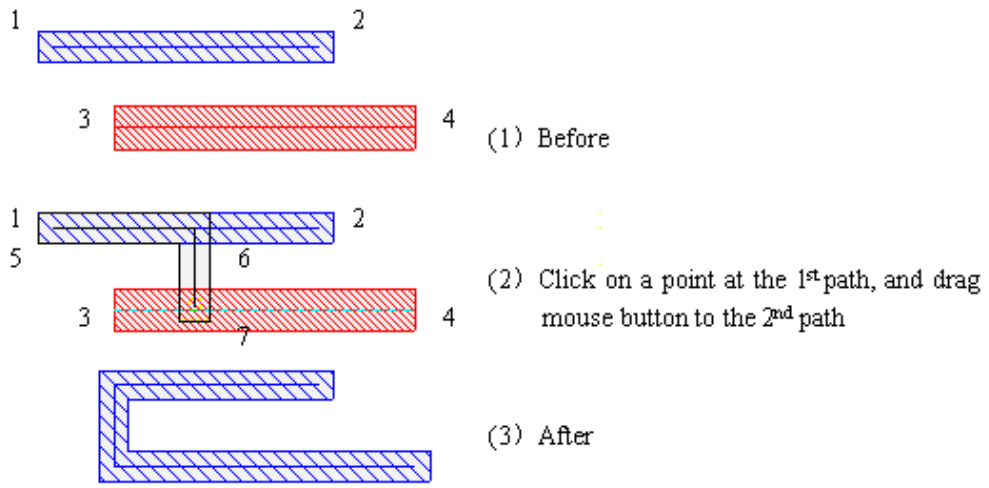
Step 6: Apply the form.

Step 7: As the below figure (2), click at a point nearby point 1, a yellow path appears starting from point 1, drag mouse button to point 7 via point 6. So the yellow path (cartoon path) track is go through point 5, point 6 to point 7.

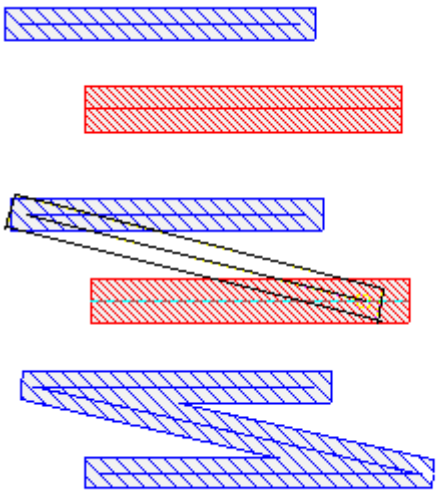
Step 8: Release mouse button at point 7, the final path is created by point 2, point 3 and point 4. Final path is shown as below figure (3).

Step 9: You can find the final path inherits the first path's properties, such as width and end type.

Why the shape of final path looks like figure (3). It is because system stores points of two paths are point1, point2, point3, point4 in database. The yellow path track is point 5, point 6 and point 7, the point 5 is the same as point 1, so the point 1 and point 5 are removed from database; the point 7 is nearby point 3, then system regards point 7 as point 3. So the valid path points are point 2, point 3 and point 7.

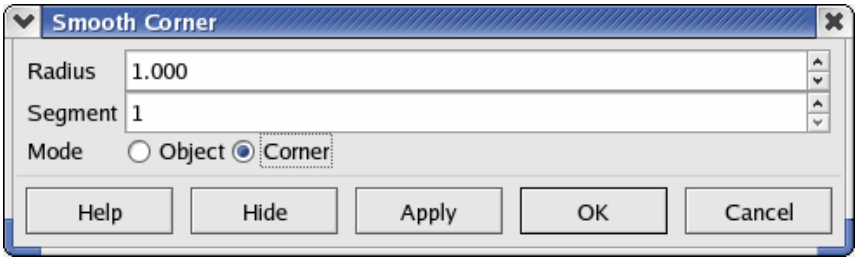


If the snap mode is "Any", the final path is created as below figure (3).



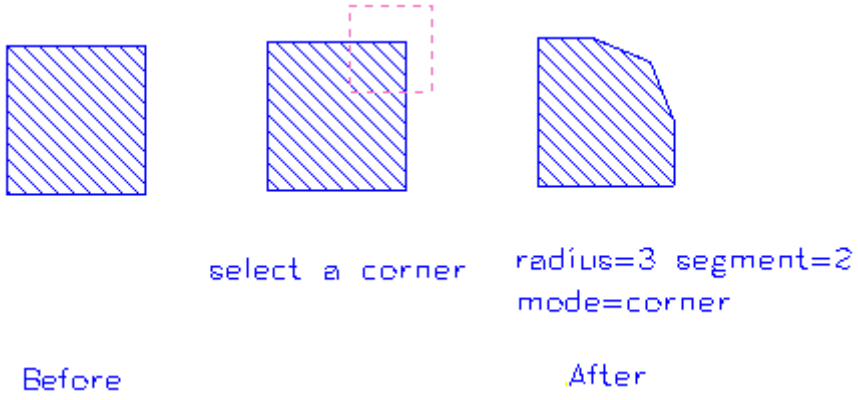
Exercise - 64 Edit->More->Smooth Corner

This command lets you make corners of rectangle or polygon into rounds with multi-segment edges.

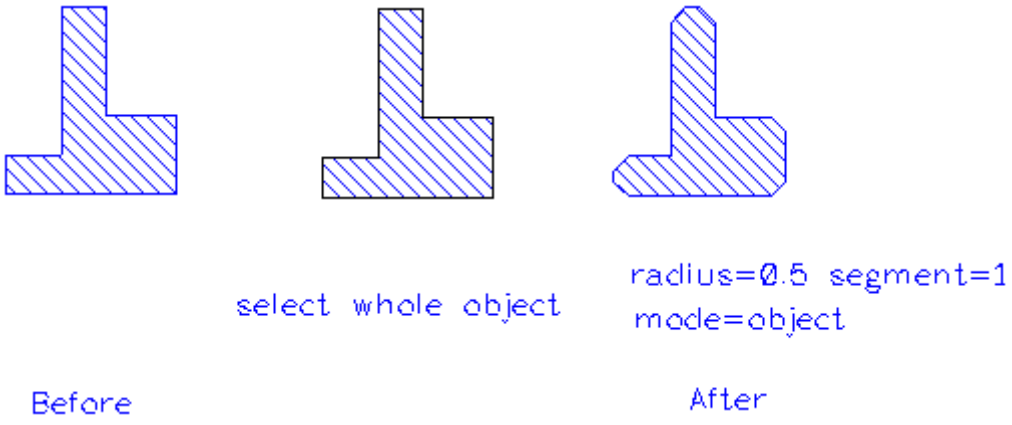


About the form above,
Radius specifies a radius of the resultant multi-segment edge.
Segment specifies how many segments will be used.
Mode specifies the target object reshape one corner or all of corners.

To smooth a corner of an object



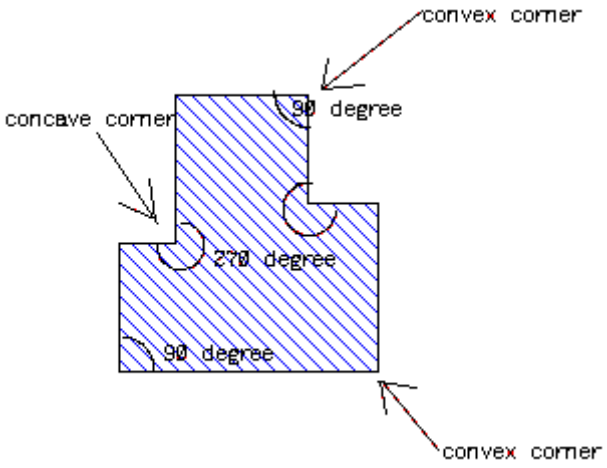
To smooth all corners of an object.



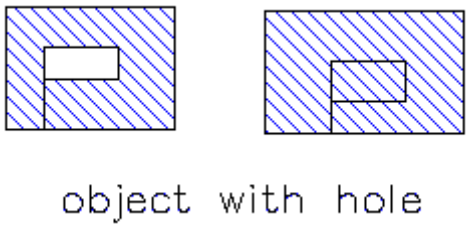
Note:

1. System doesn't operate on concave angle. Convex corner is the corner that less than 180 degree, concave angle is the corner that great than 180 degree.

   The following figure shows what are convex corner and concave angle.



2. System doesn't operate on an object with hole(s) when mode is set to *Object*.

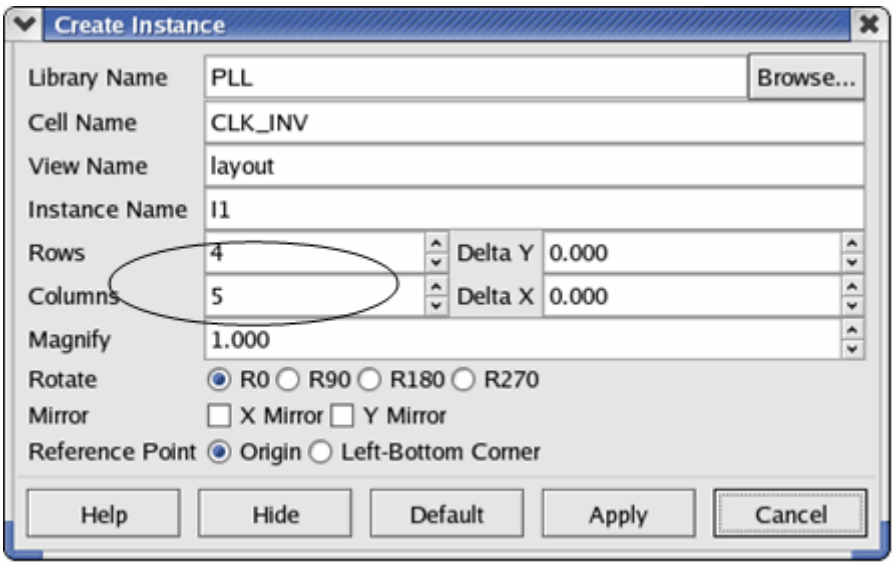


Exercise - 65 Edit->More->Split Array

When you create an instance array, the instance array is regarded as a whole object, so you cannot select or remove one of instances. Split Array lets you divide the instance array to independent instance or to sub-array

Step 1: Create a new cellview "Lab2.splitarray.layout.

Step 2: Add instance of cell "PLL.CLK_INV" with 4 rows and 5 columns.

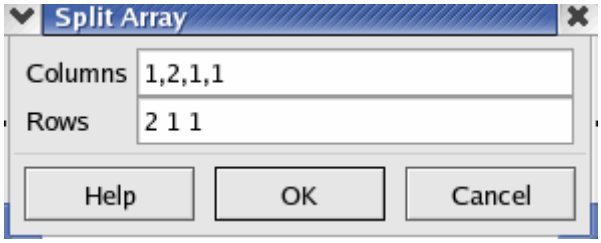


Step 3: Hide the form and put the instance array to layout editor as below figure.

Step 4: Select whole instance array.

Step 5: Click Edit->More->Split Array and press “F3” to popup related form.

Step 6: Fill out the form is shown as figure below.



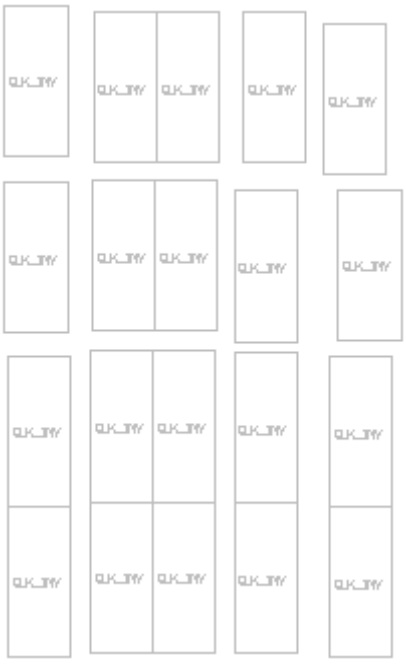
About the form above,

Columns specifies that how many sub-array columns you want to divide the whole instance into. Each sub-array columns is separated by space or comma. Sum of number is equal to instance array's column number.

Rows specifies that how many sub-array rows you want to divide the whole instance into. Each sub-array rows is separated by space or comma. Sum of number is equal to instance array's row number.

As the above figure shows, we divide whole array into 4 sub-array columns: the 1st sub-array columns includes 1 column, the 2nd sub-array columns includes 2 columns, the 3rd and the 4th sub-array columns include 1 column. In additional, we separate the whole array into 3 sub-array rows: the 1st sub-array row includes 2 rows, the 2nd and the 3rd sub-array rows include 1 column.

Step 7: Ok the form, the whole instance array is divided into 12 independent partitions. Moved each partition, you can see each partitions as figure below.



Step 8: Save and close the cellview.

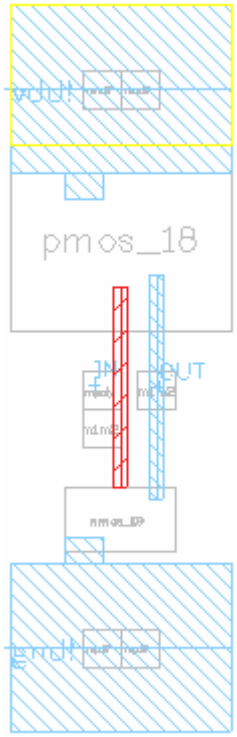
Note:

If you want to divide a whole instance array into independent instance each other, using Smash command is an easy way. Please refer to *Exercise - 68 Edit->More->Smash* to get more details.

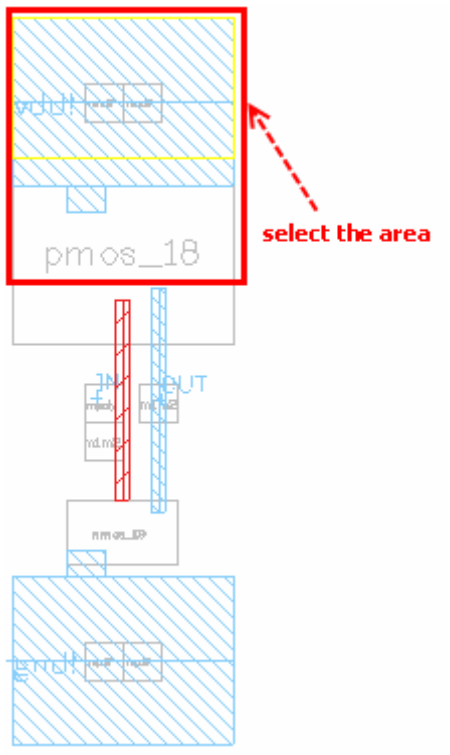
Exercise - 66 Edit->More->Yank/Paste

This command allows you copy object or portion of object to clipboard. Use *Edit->More->Paste* command removes the object from clipboard and copies them to layout editor.

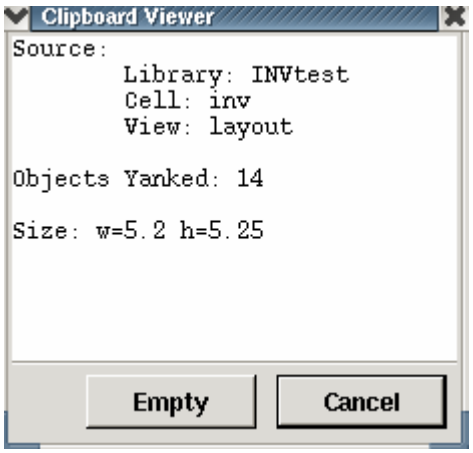
Step 1: Open cellivew “INVtest.inv.layout”. The layout is shown as below figure.



Step 2: Click Edit->More->Yank or click icon  . Select a rectangle area marked as below figure.



Step 3: Click Tools->Clipboard Viewer. Clipboard Viewer window pops up. The form records information you selected objects just now. Here, “Objects Yanked” means the number of objects you selected. “Size” means the size of selected area.

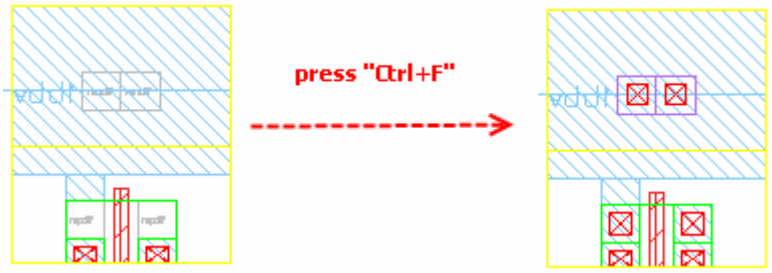


Please note that if whole instance is in selection area, Zeni regards the instance as one object, and command “Paste” will copy the whole instance to layout editor. If a part of instance is in selection area, Zeni smashes the instance to each object, and command “Paste” will copy the objects enclosed in selection area to layout editor.

Step 4: Cancel the form.

Step 5: Click Edit->More->Paste or click icon  , and click anywhere at the layout editor.

The objects selected by command Yank are copied to here. The copied objects look like below figure shown.



Step 6: Close the cellview without save.

Note:

Commands Yank and Paste must be used by couple.

Exercise - 67 Edit->More->Make Cell

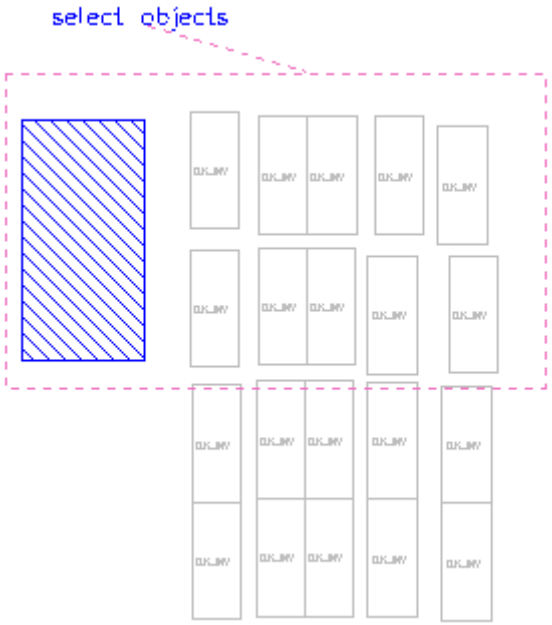
This command makes selected objects into a new cell. The new cell is shown as an instance in current cellview window.

Use the instance created in *Exercise - 65* as an example.

Step 1: Open the cellview “Lab2. splitarray.layout”.

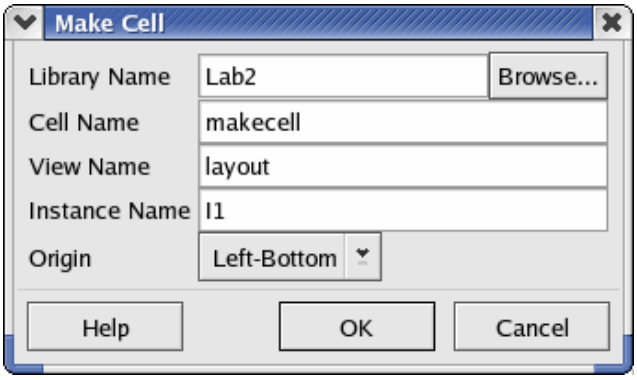
Step 2: Create a rectangle in anywhere.

Step 3: Select objects you want to make them to a new cell first.

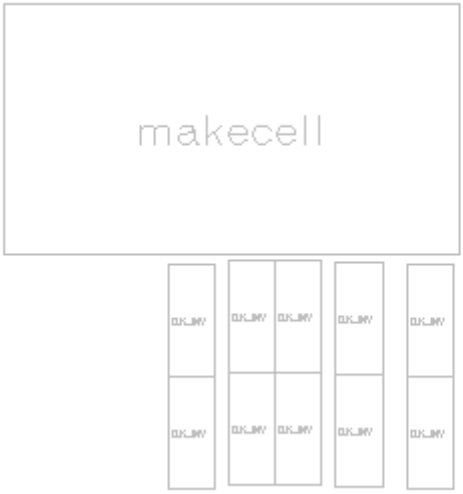


Step 4: Click Edit->More->Make Cell or click icon  . Make Cell form appears.

Step 5: Fill out the form as figure below.



Step 6: Ok the form, the cellview becomes as figure below.



Step 7: Save as the cellview to “Lab2. makecell_test.layout”.

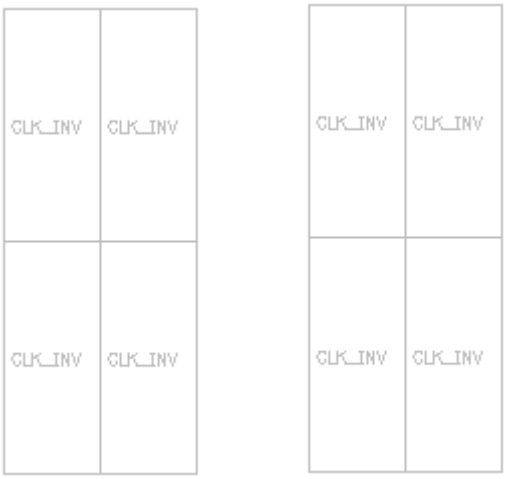
Step 8: Close the current cellview without saving.

Exercise - 68 Edit->More->Smash

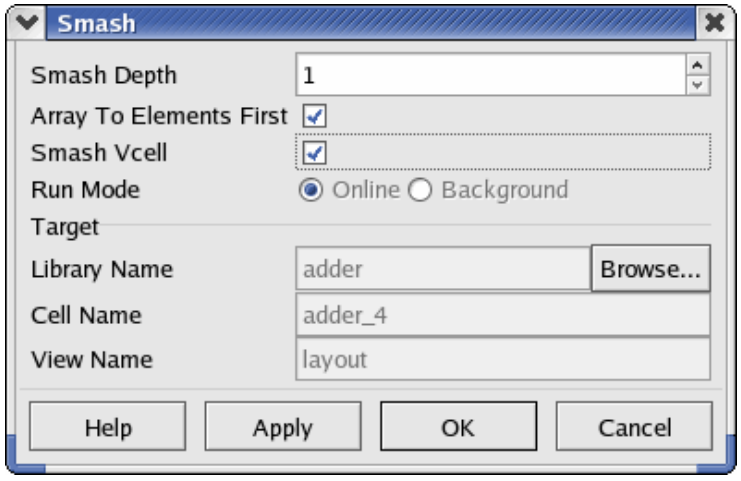
This command flattens selected cells. The objects of flattened cells are moved up to the parent level. This command cannot be reversed by Undo command.

Step 1: Create a new cellview “Lab2.smash.layout”.

Step 2: Add instance of cell “PLL.CLK-INV” with 2 columns and 2 rows twice.



Step 3: Click Edit->More->Smash or click icon  . Smash form appears.



About the form above,
Smash Depth controls flatten how many depths.
Array To Elements First specifies divide instance array into independent instance each other first if Smash Depth is 1. This is an easy way to split instance array. Please refer to *Exercise - 65 Edit->More->Split Array* on page 157.
Smash Vcell controls whether flatten vcell. About Vcell please refer to *Exercise - 87 VCell* on page 200.

Step 4: Fill out the form with the following information.
Smash Depth: 1
Array To Elements First**:** on

Step 5: Don't turn off the form, select instance array in left side first.

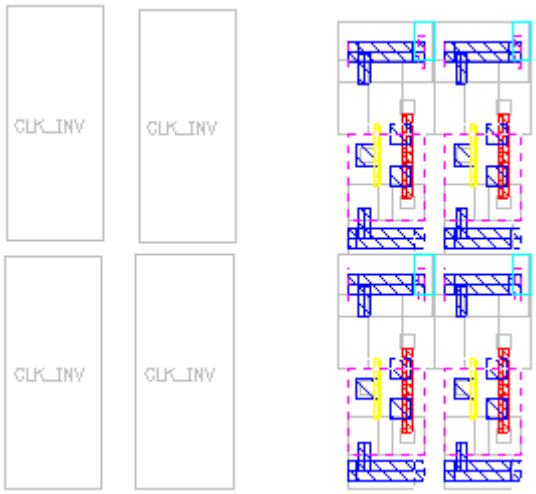
Step 6: Apply the Smash form.

Step 7: The instance array is divided into independent instance, don't flatten them. You can move each instance.

Step 8: Turn off *Array To Elements First* option, select instance array in right side again.

Step 9: Ok the Smash form. The objects of the first level in cellview "Lab2.CLK_INV.layout" are flattened and shown in current cellview.

Through smashing two instance arrays with different option *Array To Elements First*, we can get the different results.



Step 10: Save and close cellview.

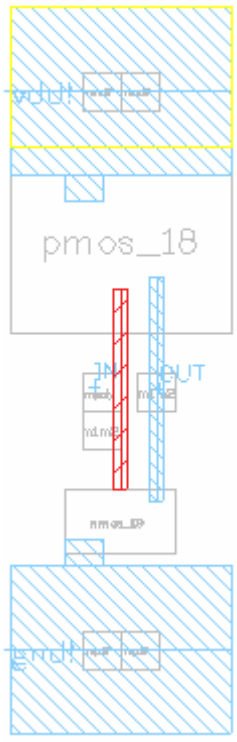
Note:

This command cannot be reversed by the Undo command. You can use *Design->Discard Editor* to restore to original.

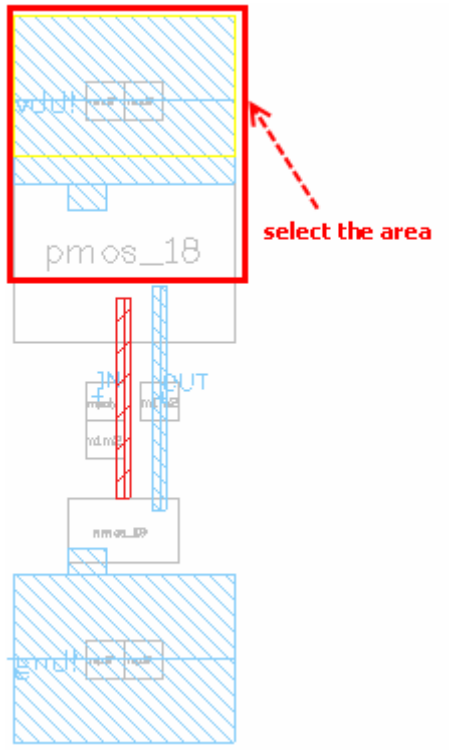
Exercise - 69 Edit->Property Notepad

This command lets you look over or modify properties of objects you selected.

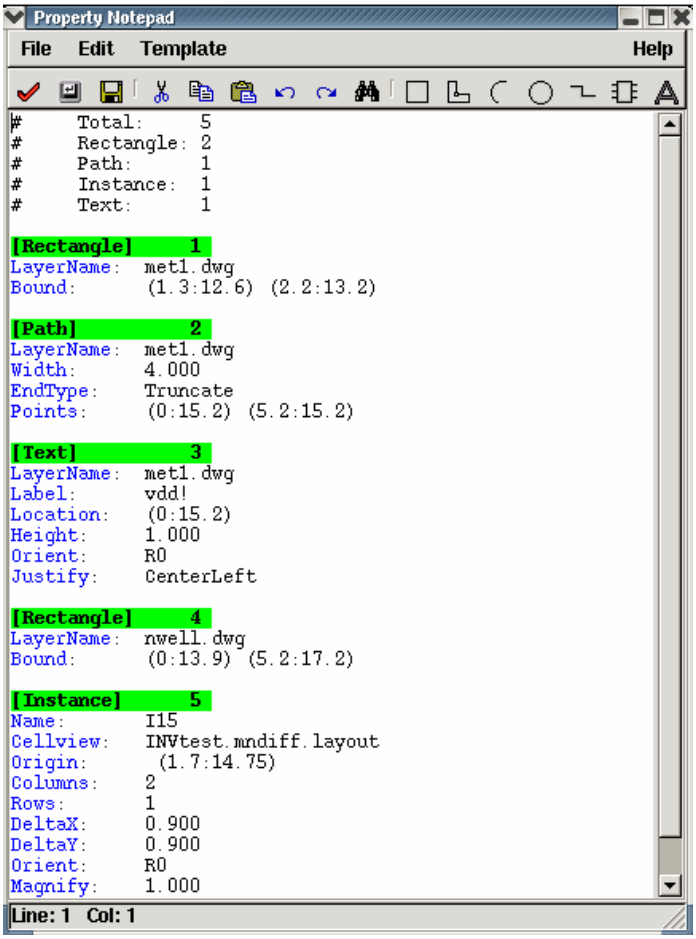
Step 1: Open cellivew “INVtest.inv.layout”. The layout is shown as below figure.



Step 2: Select an area shown as below figure.



Step 3: Click Edit->Property Notepad, Property Notepad window pops up.



The window displays properties of objects in your selected area just now. Here, you can modify any object property, for example, rectangle coordinates, path width, path coordinates, instance location, etc. After modification, click File->Apply and Return or click icon  in the window, modification will be effected if there is no any syntax error.
It also provides templates to create new objects, for example, path, rectangle, label, instance, etc.

Step 4: Modify instance columns from 2 to 1, origin coordinates from (1.7:14.75) to (2.25:14.75).

Step 5: Click File->Apply and Return or click icon  .

Step 6: The Property Notepad window closes and two contact cells becomes one cell. The figure is shown as below.

Step 7: Close the cellview without save.

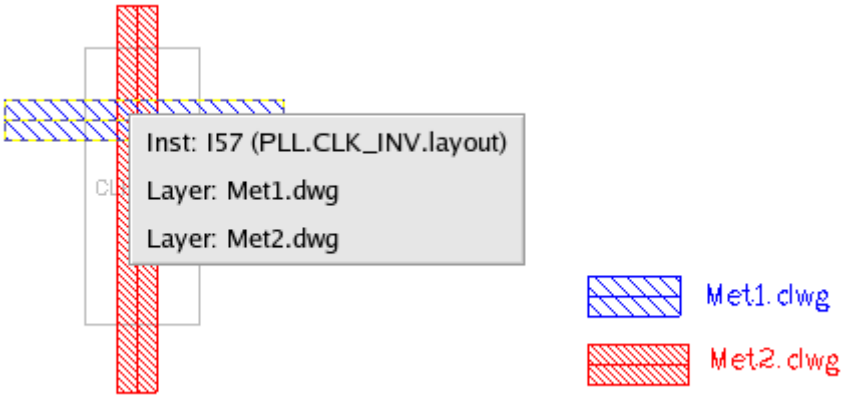
Exercise - 70 Select->Select By Layer

This command helps you select a layer or an instance from a group of overlapped objects in different layers.

As the below figure shows, 3 objects are overlapped: an instance and two paths in different layers.

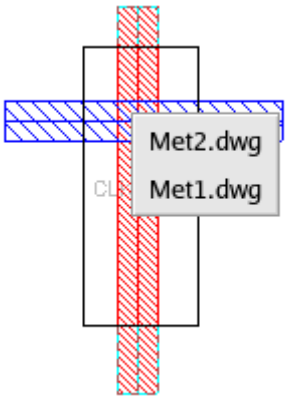
Step 1: Click Select->Select By Layer and click at a point in overlapped area.

Step 2: A small menu pops up as below figure. The menu lists all of layers and instances in that point.



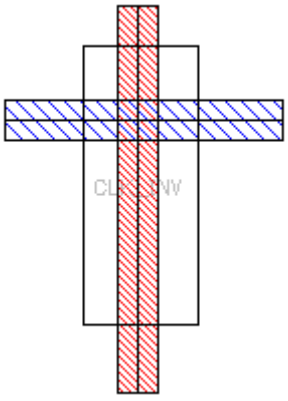
Step 3: Choose the first item. The related instance will be selected.

Step 4: Continue to click this point, small menu popup again. The menu lists the remained two layers.



Step 5: Choose the first item. The layer “Met2.dwg” will be selected.

Step 6: Continue to click this point, small menu doesn’t appearance, system directly selects layer “Met1.dwg”.



Note:

If one object is selected before, this command will not select it again, that is, small menu doesn’t

list the object.

Exercise - 71 Hierarchy->Enter/Descend

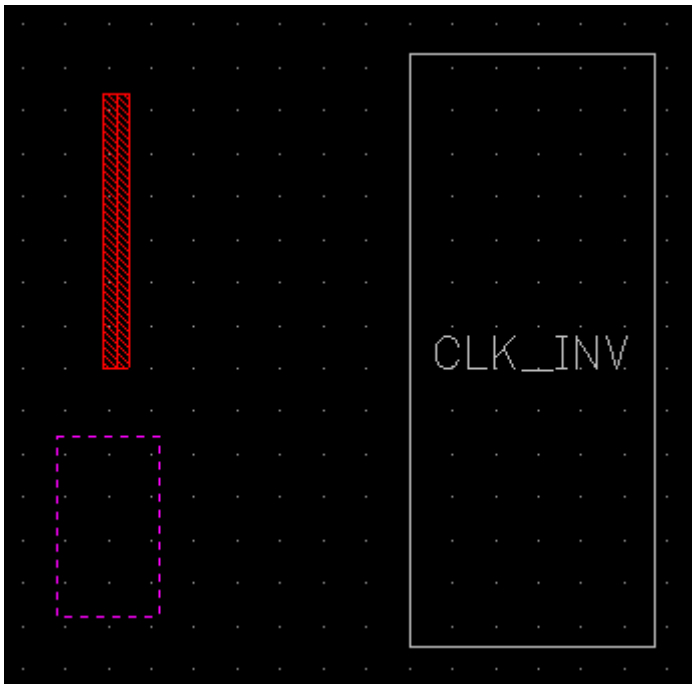
Enter command lets you enter a cell or object at lower level of design hierarchy. The adjacent cells or objects in the higher hierarchy level are kept visible. But Descend command enters lower level cell or object directly without viewing its surrounding higher level object.

You can turn on *Options->Generic->Editor->Shadow Surrounding Objects* to gray the adjacent cells or objects in higher hierarchy level when you use Enter command. Please refer to *Exercise - 86 Options->Generic* on page 195.

Step 1: Create a new cellview “Lab2.enter.layout”.

Step 2: Add an instance of cell “PLL.CLK_INV”.

Step 3: Create some objects around the instance anywhere. Below figure shows an example.

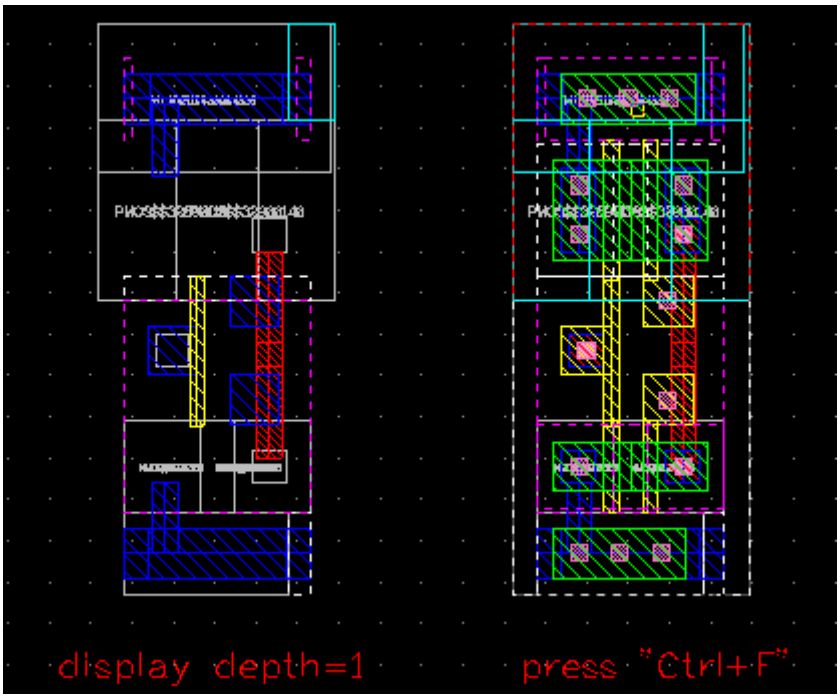


Step 4: Turn on *Options->Generic->Editor->Shadow Surrounding Objects*.

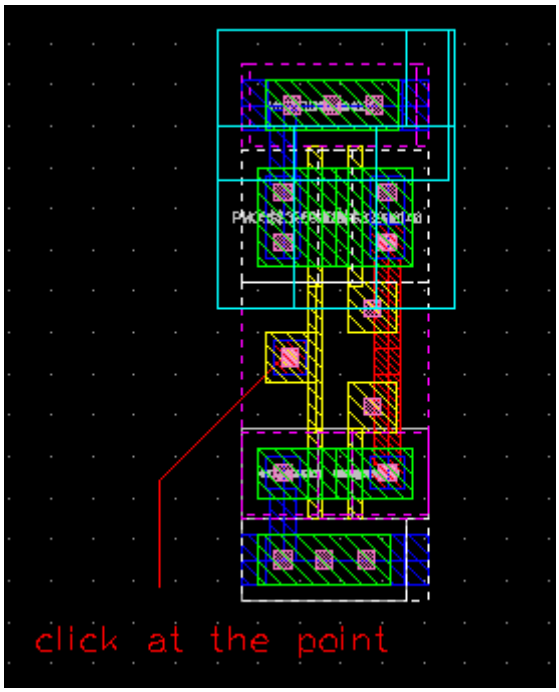
Step 5: Press “Alt+F” to set display depth to 1.

Step 6: You can find the instance includes other sub-cells.

Step 7: Press “Ctrl+F” to show full hierarchy objects in current window.



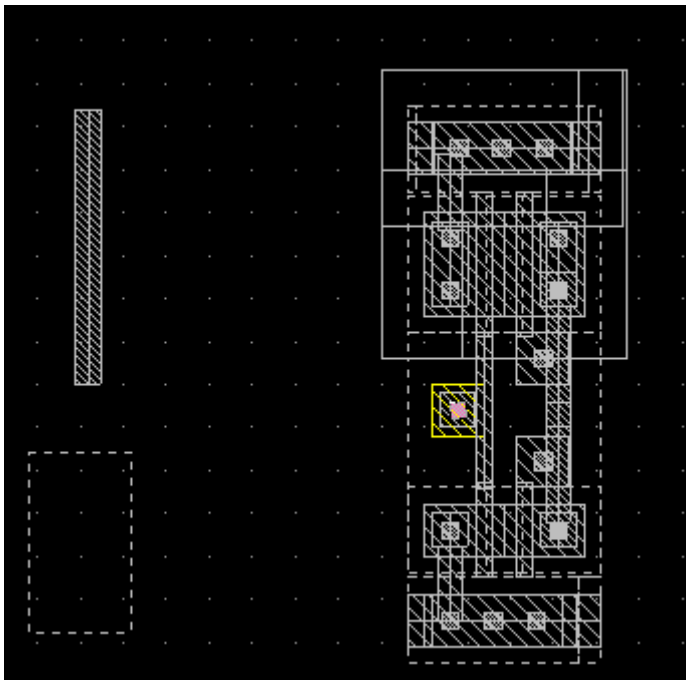
Step 8: Click Hierarchy->Enter, and click at a point as below figure shows.



Step 9: System enters cellview “PLL.M1_POLY$$32249900.layout” directly and shows it with blink. Editor window’s title becomes to 

. Surrounding higher level objects still are visible in grey, but you cannot modify them because current cellview is

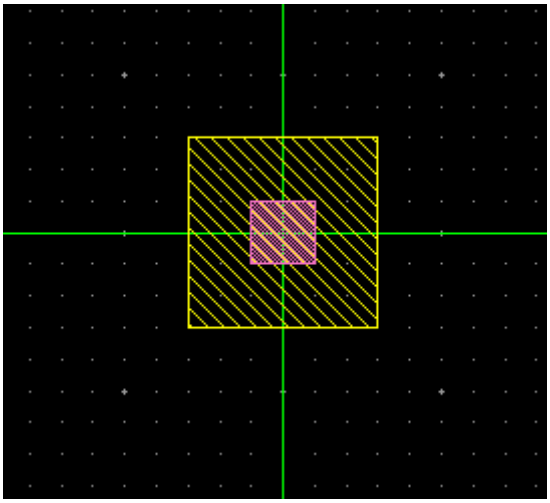
"PLL.M1_POLY$$32249900.layout" after all, you can only modify objects in cellview "PLL.M1_POLY$$32249900.layout". But in this case, library PLL is a demo library, you cannot modify this cellview.



Step 10: Click icon  to return to upper level. Current cellview is "PLL.CLK-INV.layout".

Step 11: Click icon  again to return to the toplevel cellview "PLL.enter.layout".

Step 12: If you select Hierarchy->Descend or click icon  , and click that point as Step 8, system will enter cellview "PLL.M1_POLY$$32249900.layout" directly without viewing its surrounding higher level objects.



Note:

Before using these commands, the content of the cell or objects should be visible with proper settings of view depth or using “Ctrl+F”.

Exercise - 72 Hierarchy->Enter one Step/Descend one Step

These two commands are similar to Enter and Descend. The difference is entering cell or object just one level once.
For example, refer to *Exercise - 71 Hierarchy->Enter/*Don page 170, click that point as Step 8, then enters cellview “PLL.CLK_INV.layout” only.

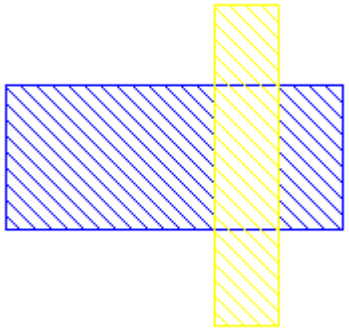
Exercise - 73 Advanced->Generate Slot

This command helps you add slot to polygon and rectangle.

Step 1: Create a new cellview “Lab2.slot.layout”.

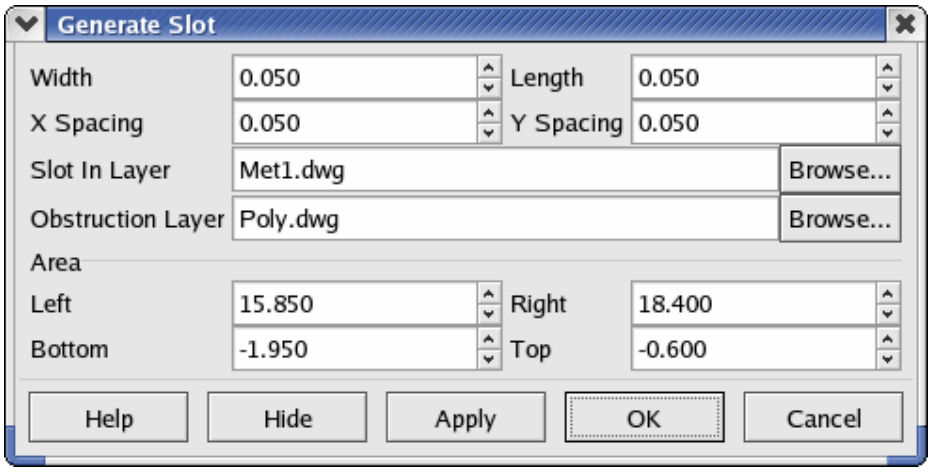
Step 2: Zoomin the cellview.

Step 3: Create two rectangles with “Met1.dwg” and “Poly.dwg” as below figure.



Step 4: Click Advanced->Generate Slot, related form appears.

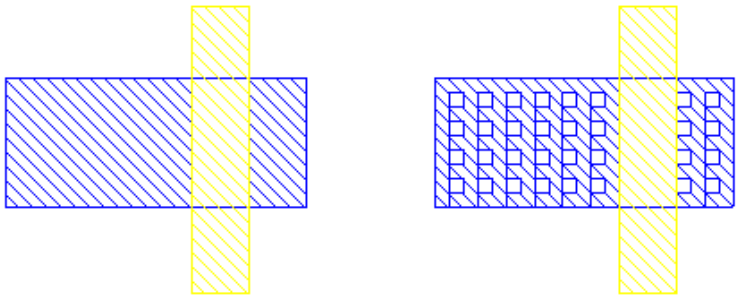
Step 5: Fill out the form with the information as figure shown below.



Step 6: In the above form, you can fill out the fields of Width, Length, X Spacing and Y Spacing as you need.

Step 7: Move cursor into editor window and select an area enclose the whole object on Met1.dwg. The coordinates of area you selected is shown in Area filed.

Step 8: Ok the form. Slots are added to object on Met1.dwg, the area overlapped with Poly.dwg doesn't add any slot because layer "Poly.dwg" is regards as obstruction layer.



Step 9: Save and close the cellview.

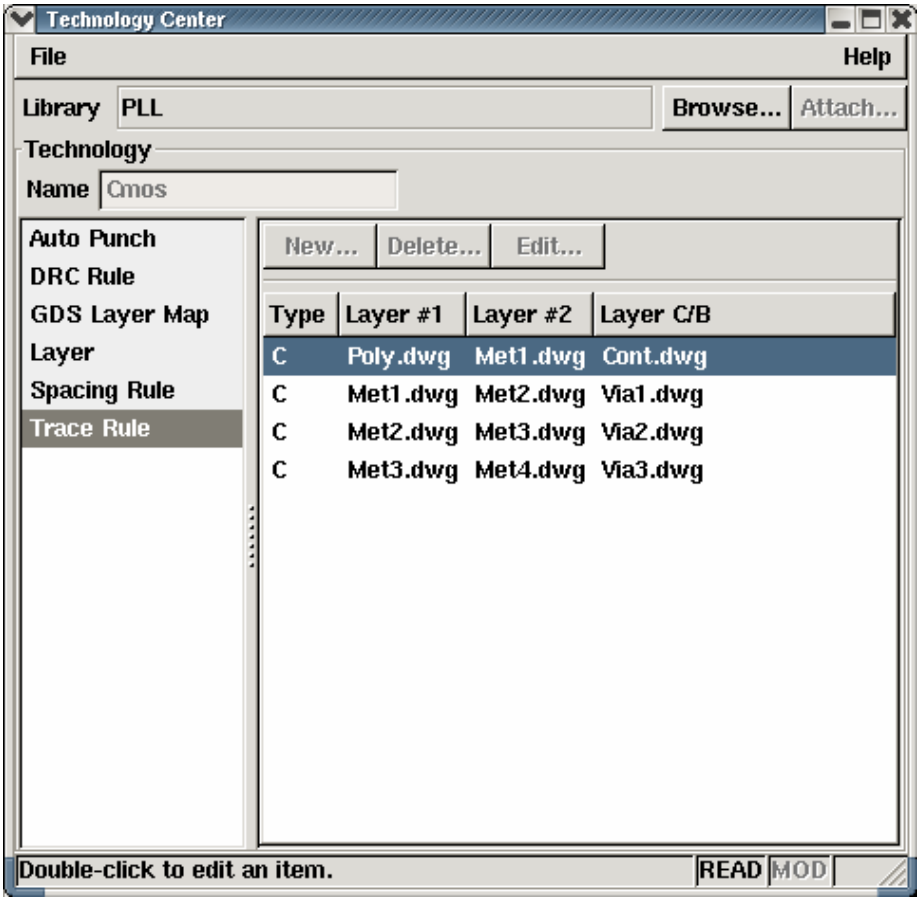
Exercise - 74 Advanced->Trace Net

This command lets you trace and highlight all electrically connected objects in all visible levels. Before using this function, you need to create *Trace Rule* in *Technology Center*.

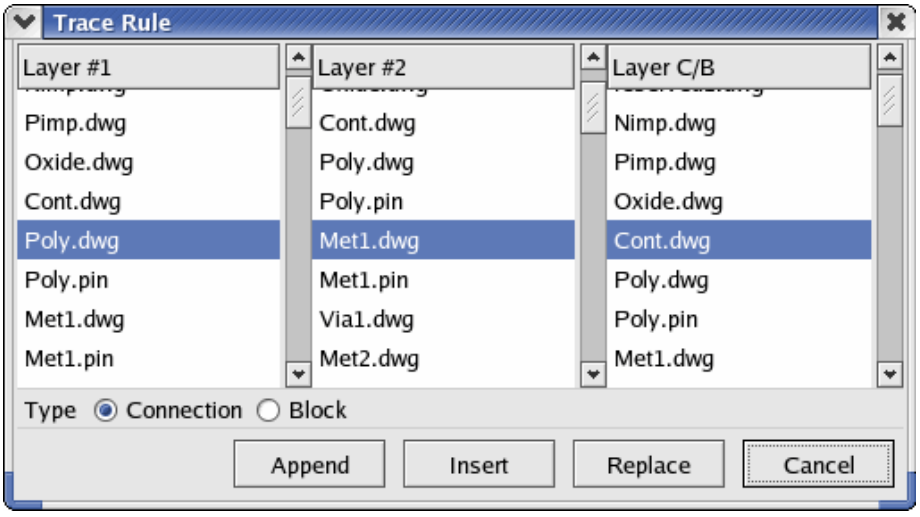
Step 1: Open cellview "PLL.DFFC.layout".

Step 2: Click File->Technology Center in Lay Palette window. Technology Center form appears.

Step 3: Click Trace Rule item, all of trace rules are shown in the right side of window.



Step 4: The four rules were created before. Double click at the first item, a form pops up. You can modify or add item in this form

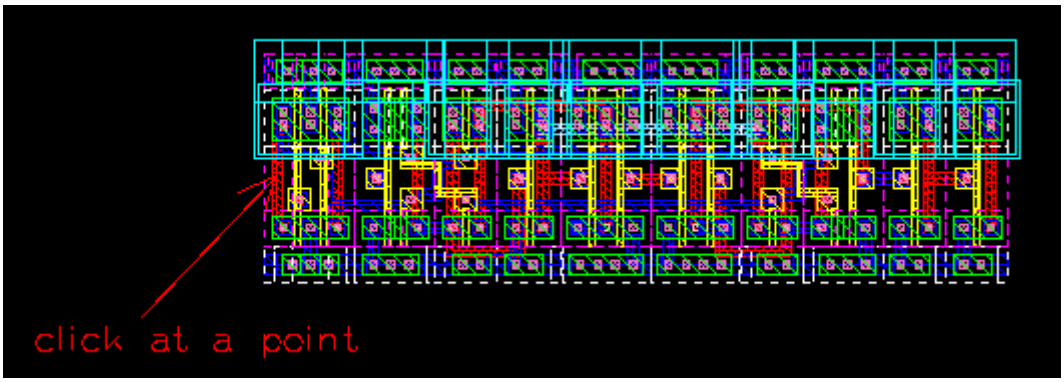


Step 5: Cancel the form and close Technology Center form.

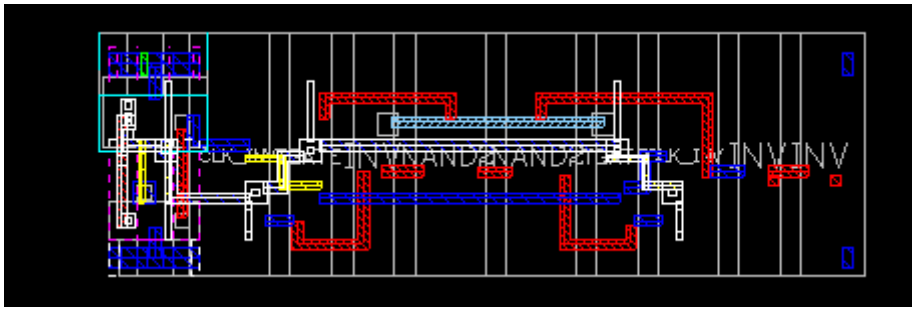
Step 6: Move cursor into editor window.

Step 7: Press “Ctrl+F” to display full hierarchy images.

Step 8: Click Advanced->Trace Net and click at a point on Met2.dwg.



Step 9: All of connected objects are highlighted. Press “Shift+F” to see these highlighted objects clearly.



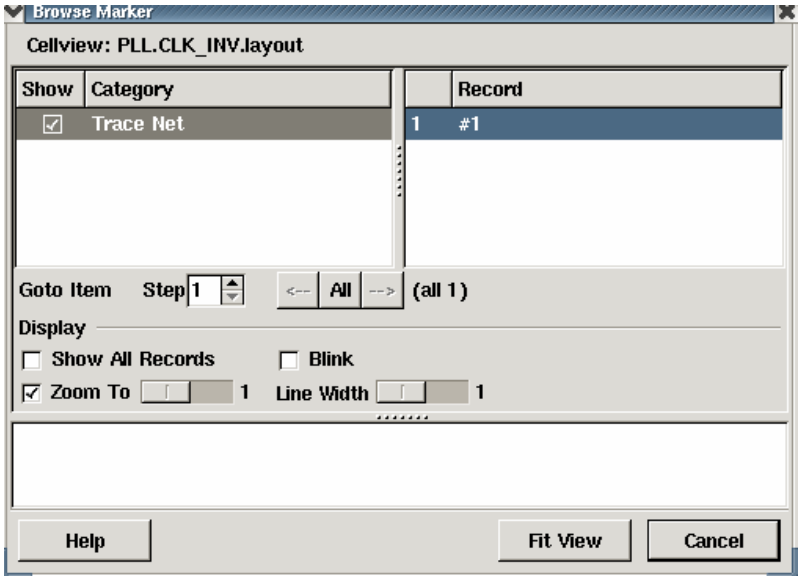
Step 10: Click Tools->Delete All Markers to remove highlight.

Step 11: Close current cellview.

Note:

You can trace multiple nets one by one till pressing “Esc” key. The highlighted color is specified by Options->Color.

You can also use Tools->Browse Marker to look over the highlight with blink. Please refer to *Exercise - 85 Tools->Browse Marker* on page 194.

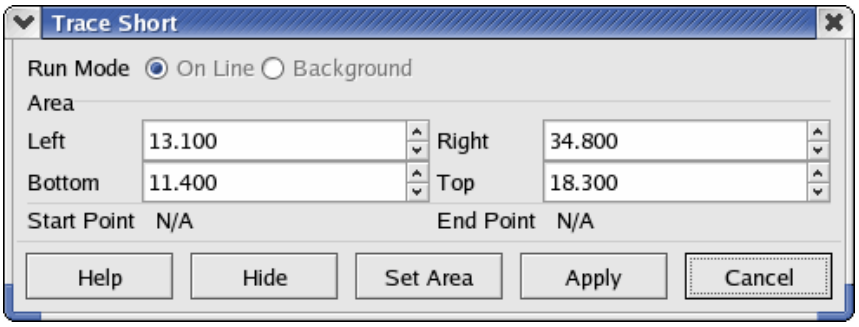


Exercise - 75 Advanced->Trace Short

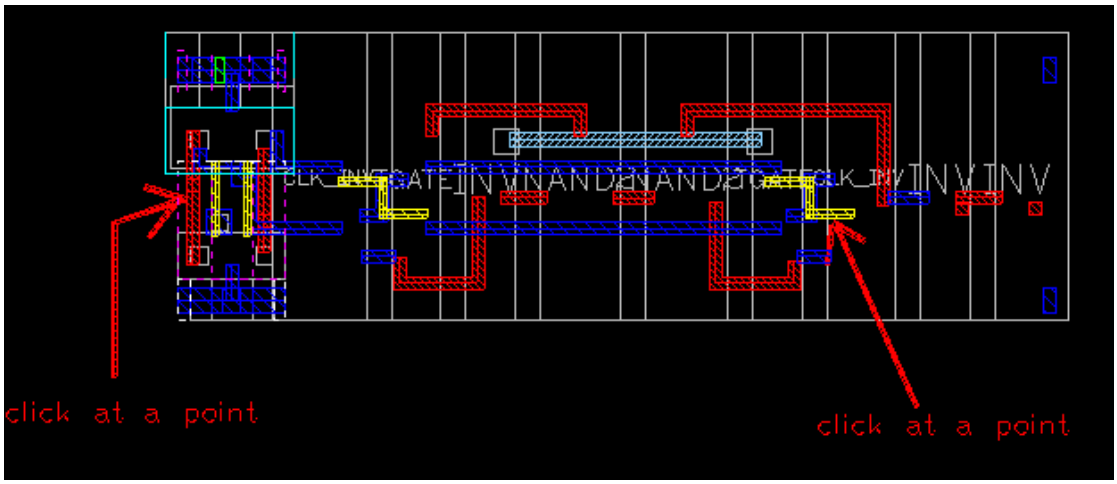
This command helps you check if your design has short circuit between two points.
The usage of this command resembles Trace Net. One difference is Zeni only highlights one "connected" path. This connected path represents that there is a short circuit between two points. You need to correct your design until Zeni cannot find out such path.

Step 1: Open cellview “PLL.DFFC.layout”.

Step 2: Click Advanced->Trace Short. Trace Short form appears.



Step 3: Click at start point and end point as below figure shows.



Step 4: Press “Ctrl+F” to show all objects in full hierarchy.

Step 5: Apply the Trace Short form.

Step 6: A connected path is highlighted. Press “Shift+F” to see the highlighted path clearly.

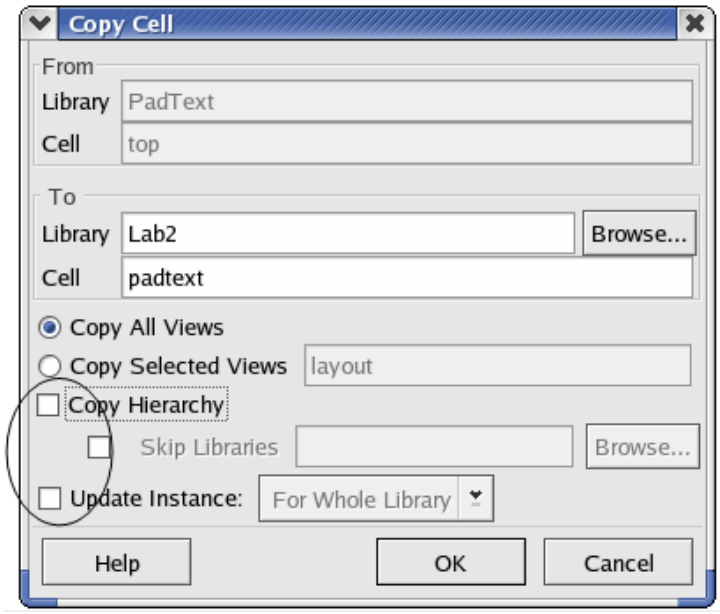
Note:

If you select an area in editor window before applying the Trace Short form, system will search connected path within this area. It’s good for speeding trace process, but maybe cannot find out any short circuit path if the area is too small.

Exercise - 76 Advanced->Pad Text

This command helps you add pad label easily and fleetly. This command need to be supported by a pad text file. This file records all names of pad labels. Each label name in an independent line.

Step 1: Copy demo cell “PadText.top” to “Lab2.padtext” without hierarchy.



Step 2: Open cellview “Lab2.padtext.layout”.

Step 3: Press “Ctrl +F” to show all objects in full hierarchy. In this cellview, eight pads are included. Each pad should be added with a label. All labels’ name are written in a text file. Open file “$ZENI_INSTALL_PATH/demo/PDT/PadTextLib/padtext.txt”, you can find these labels as below.

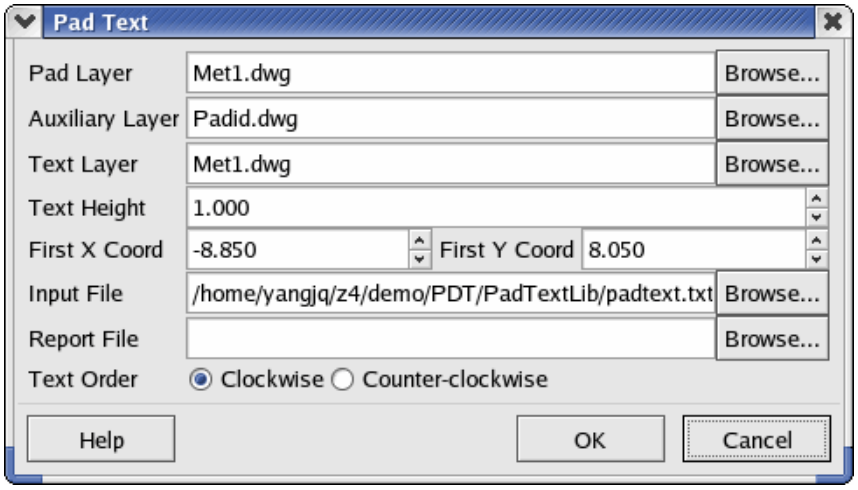
```
VDD
IN1
IN2
IN3
GND
OUT3
OUT2
OUT1
```

In this file, the first label must be in first line and each label string had better starts from the first columns.

Step 4: Click Advanced->PadText, Pad Text form appears.

Step 5: Fill out the form with following information.

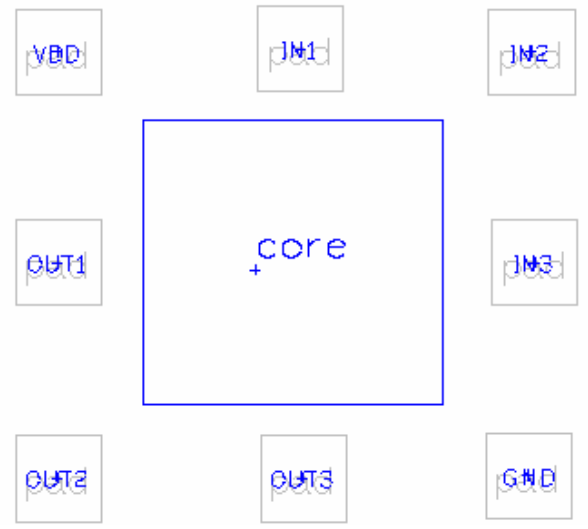
Pad Layer: Met1.dwg
Auxiliary Layer: Padid.dwg
Text Layer: Met1.dwg
Text Height: 1.000
Input File: $ZENI_INSTALL_PATH/demo/PDT/PadTextLib/padtext.txt
Text Order: Clockwise



Step 6: Click at a point on a pad sub-cell in left top corner. The coordinates of the point are shown in First X/Y Coord fields. This point represents the position of first pad. System adds labels in clockwise direction or Counter-clockwise direction beginning from this pad.

Step 7: Ok the form. Pad labels are added into each pad clockwise.

Step 8: Press “Shift+F” to show object in the toplevel only. You can see the pad labels clearly.



Step 9: Save and close the cellview.

Note:

If pads are placed irregularly, system cannot distinguish clockwise or counter-clockwise direction, so this command doesn’t work well.

Exercise - 77 Advanced->Generate Array

This command helps you create objects array. Compared with instance array, this command can create an array with two kinds of cells. Why system only supports two kinds of cells? Because this command is used to design memory cell. In memory cell, there are only two sub-cells, 0 and 1.

This command need to be supported by a rule which specifies how to place each object.

Step 1: Copy demo library “GenArray” to “GenArray-test”. This library contains two cells 0 and 1. These two cells are just demo cells, not real memory cells.

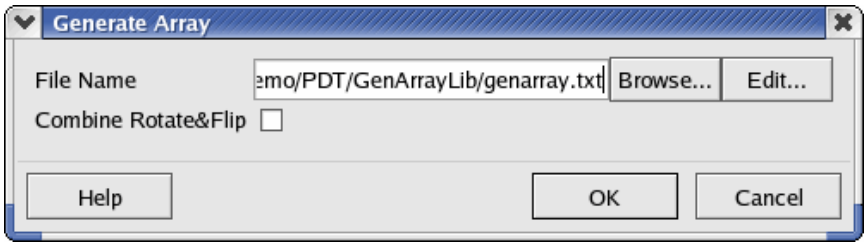
Step 2: Open cellviews “GenArray-test.0.layout” and “GenArray-test.1.layout” to view.

Step 3: Close these two cellviews.

Step 4: Open cellview “GenArray-test.mem.layout”.

Step 5: Select Advanced->Generate Array. Generate Array form appears.

Step 6: Fill out the form with file name “ $ ZENI_INSTALL_PATH/demo/PDT/GenArrayLib/genarray.txt”.



Step 7: Click Edit button to view the file as below.

```
cellname 1 0;
mode 1;
xtrack 3, 3 2, 4 3, 5 2;
ytrack 3, 3 1, 4 2, 5 3;
#define row 1 to 16
vacancy 8, B101, C10F, D110, A101, A100, E010, D110, B001;
#define row 17 to r32
#vacancy 8; F000, C000, A000, C000, F000, E000, B000, A000;
r90 x, 1, 3, 6;
xflip x, 1, 3, 6;
place 0, 0;
size 8, 16;
```

About the file format.

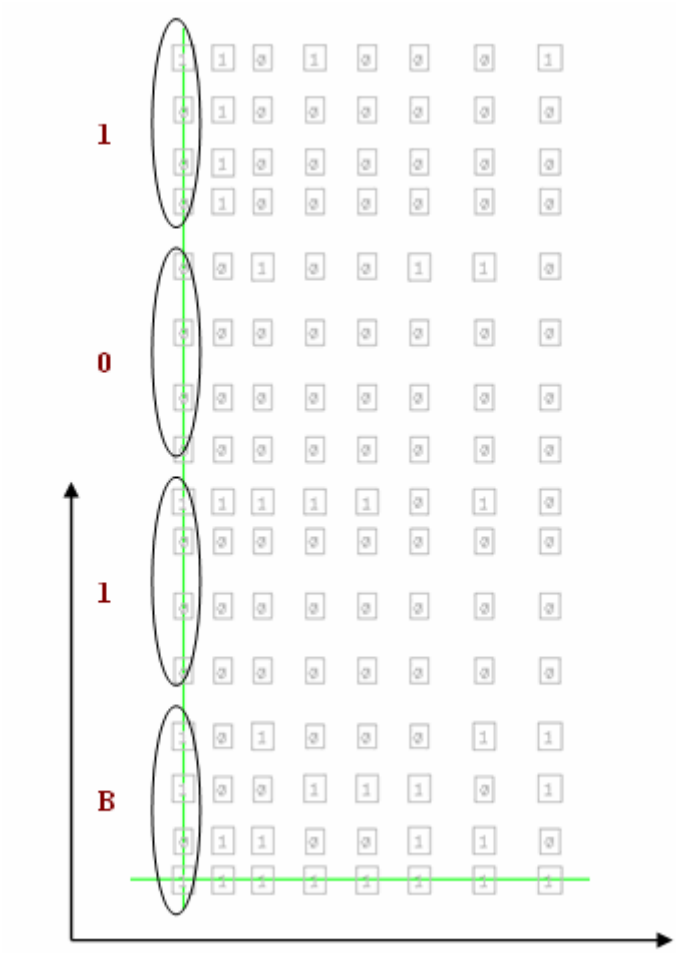
Rule	Explanation
Cellname <cellname1> [<cellname0>]	"cellname1" and "cellname0" are two cell names in current library. They are used in *vacancy* rule. When the value is 1 in vacancy rule, "cellname1" is added into object array; when the value is 0 in *vacancy* rule, "cellname0 "is added into object array. If this command has no "cellname0", leave object vacant
Model 0/1	It specifies how to place the whole array. '0' specifies array starting from left top; '1' specifies array starting from left bottom. 1 is default.
Xtrack <number>,<spacing> <step>,…,<spacing> <step>;	It specifies distance among cells in X / Y axis. <number> specifies the amount of spacing definition. "<spacing> <step>" is one of spacing definitions. Each spacing definition is separated with ",". <spacing> specifies the distance of two cells. <step> specifies times uses the spacing rule. For example, "xtrack 3, 3 2, 4 3, 5 2;" means that there are 3 definitions. The first is "3 2", the second is "4 3" and the third is "5 2". Here, "3 2" means the distance is 3 microns between two cells and this rule is used twice, that is, there are 3 cells, the distance of each two cells is 3 microns.
Vacancy <number>,<hex>,…,<hex>;	This command specifies location of each cell.<number> specifies amount of hex digital. Each <hex> represents 16 locations in one row or one column. When Mode is 1, <number> specifies the number of column, each <hex> represents 16 locations in one column. Otherwise, <number> specifies the number of row, <hex> represents 16 locations in one row. For example, "Mode 1" "vacancy 3 B101 C10F D110" The above rules specifiy that there are 3 colomns and 16 rows. Here, for example "B101", each number is a hex digital. So system extends the 4 numbers to16 bits "1011000100000001". In the 16 bits. '1' represents "cell1" and '0' represents "cell0".

	And the 16 cells are placed from down to up as mode is 1.
R90/R180/R270 <x/y> <number>, …, <number>;	Rotate cells. When mode is 1, x specifies to rotate cells in row. Y specifies to rotate cells in column. <number> specifies the number of row. When mode is 0, x specifies to rotate cells in column. Y specifies to rotate cells in row. <number> specifies the number of column
Xflip/Yflip/X90Flip/Y90Flip <x/y> <number>, …, <number>;	Flip cell in X axis or Y axis. X90Flip/Y90 Flip specifies to rotate counter-clock 90 degree first.
Size <column>,<row>;	This command specifies the number of column and row you want to place.
Place x,y;	This command specifies center location of the first cell.

Step 8: Close the text.

Step 9: In Generate Array form, Combine Rotate&Flip specifies whether rotating or flipping cell based on previous rotate/flip rule or not If turn off the option, system omit the previous rotate/flip rule.

Step 10: Ok the form. The cell array is created as below.

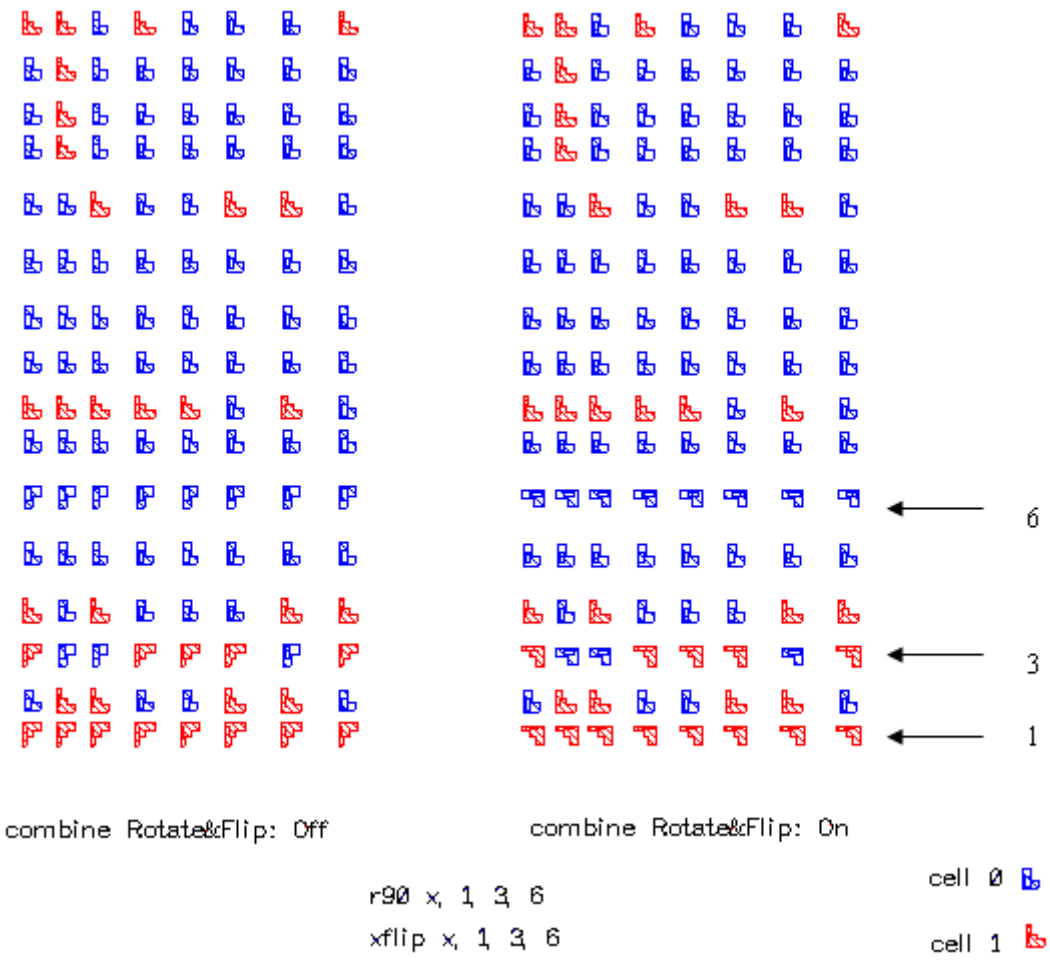


As the figure above shows, the first cell starts from point (0,0). Take an example of the first column, cells are placed from down to up as mode is 1. Cell ‘1’ and cell ‘0’ are placed with “B101”.

Step 11: Press “Ctrl+A” to select all cells and delete them.

Step 12: Select Advanced->Generate Array again and turn on option *Combine Rotate&Flip.*

Step 13: Ok the form. The figure below shows the difference of turning on or turning off the option *Combine Rotate&Flip.*The left side of figure shows rule “r90 x, 1, 3, 6” doesn’t work for cells in rows 1, 3 and 6.The right side of figure shows two rules working on cells in rows 1, 3, and 6.



Step 14: Save and close the cellview.

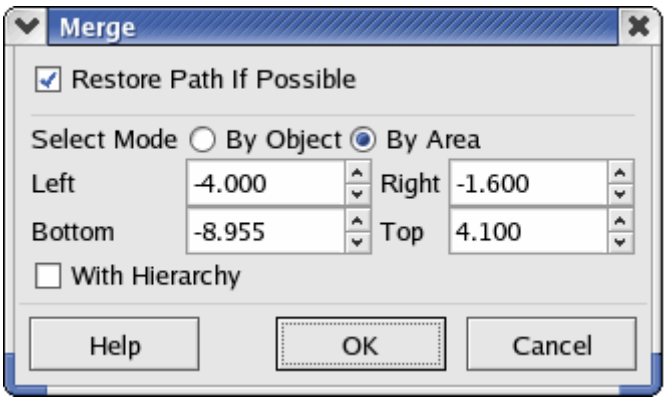
Exercise - 78 Advanced->Merge

This command lets you merge selected objects in the same layer into one polygon.
All geometric objects will be converted to polygons after merging. For path, it still keeps special property when the center lines of two (or more) paths whose width are equal are touched.

If you want to keep path property after you merge path, you must obey following rules.

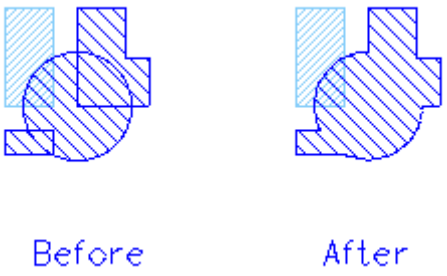
1. The paths width must be the same.
2. The paths must be touched and not overlapped.

Step 1: Select Advance->Merge or click icon  , Merge form appears as below.

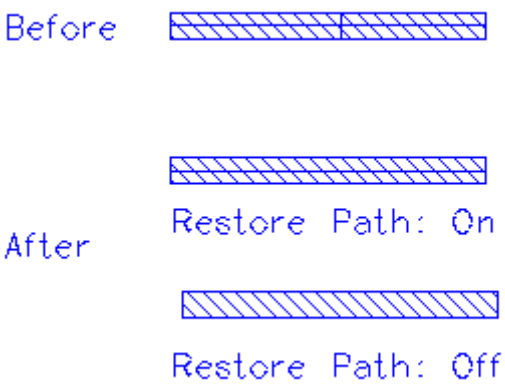


Step 2: Select some objects one by one or use Area Select to choose some objects in an area.

Step 3: Ok the form, objects in the same layer will be merged into one polygon.



When you merge more paths, set Restore Path If Possible to on, final object will be one path. Otherwise, the final object is a polygon.



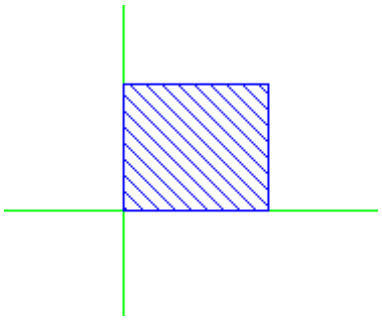
Exercise - 79 Advanced->Resize

This command lets you resize selected objects. The resized objects can be output to another

layer.

Step 1: Create a new cellview “Lab2.resize.layout”

Step 2: Create a rectangle with “Met1.dwg” as figure shown below. The left bottom point of rectangle is at the origin.

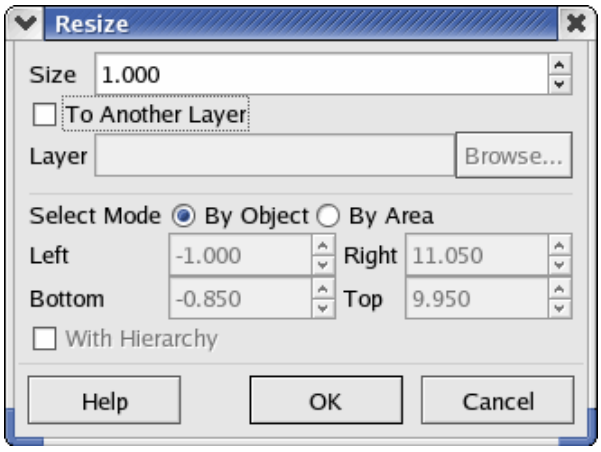


Step 3: Select Advanced->Resize or click icon  , Resize form appears.

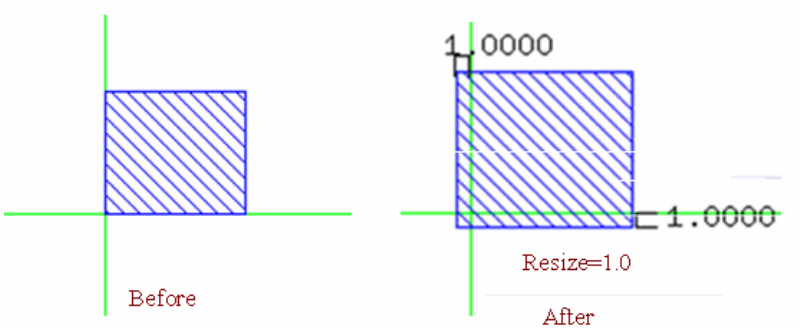
Step 4: Fill out Size field with a number to increase or decrease the objects.

Step 5: Select the rectangle from editor window

Step 6: Fill out the form with the following information as shown in the figure below.
Size: 1.000
Select Mode: By Object.

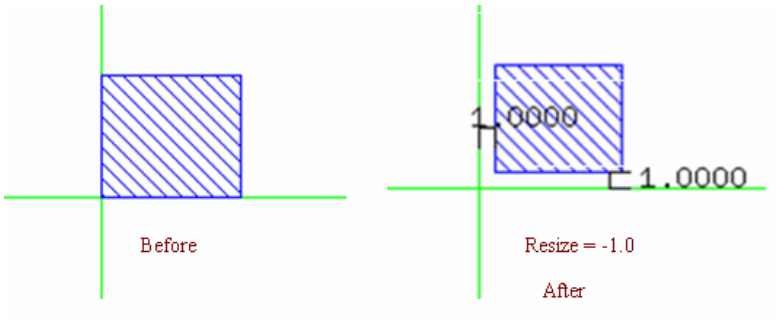


Step 7: Ok the form. Each edge of the rectangle is increased 1 micron as shown in the figure below.



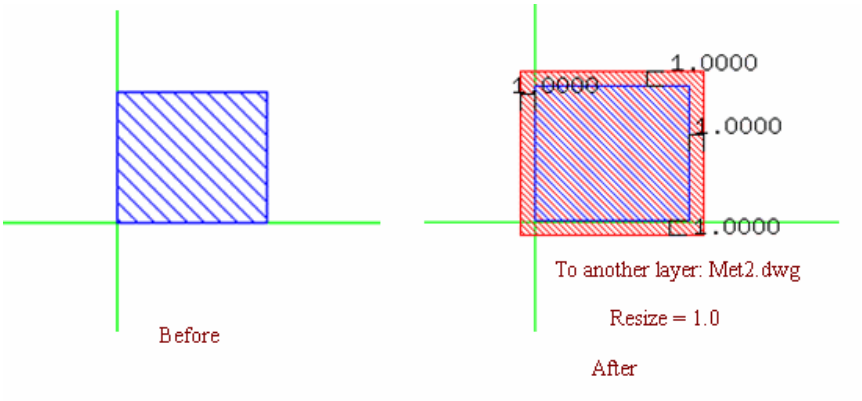
Step 8: Undo previous command and run Resize again.

Step 9: Fill Size filed to -1.000. Each edge of the rectangle will be decreased 1 micron as shown in the figure below.



Step 10: Undo previous command and run Resize again.

Step 11: Fill 1.000 to Size field and turn on To Another Layer and fill “Met2.dwg” to Layer field, system will create a new rectangle whose each edge is 1 micron larger than old one in “Met2.dwg”.



Step 12: Save and close current cellview window.

Note:

If you set Select Mode to By Area and turn on With Hierarchy, system will increase or decrease all visible images within an area in hierarchy.

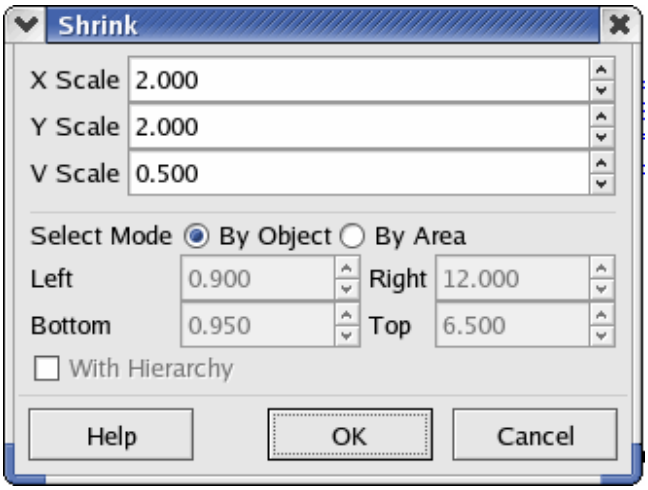
Exercise - 80 Advanced->Shrink

This command lets you magnify or shrink coordinates of object.

For example, there is a path 2 micron in width, center of path starts from at (3.0, 3.0) and ends at (6.0, 3.0).

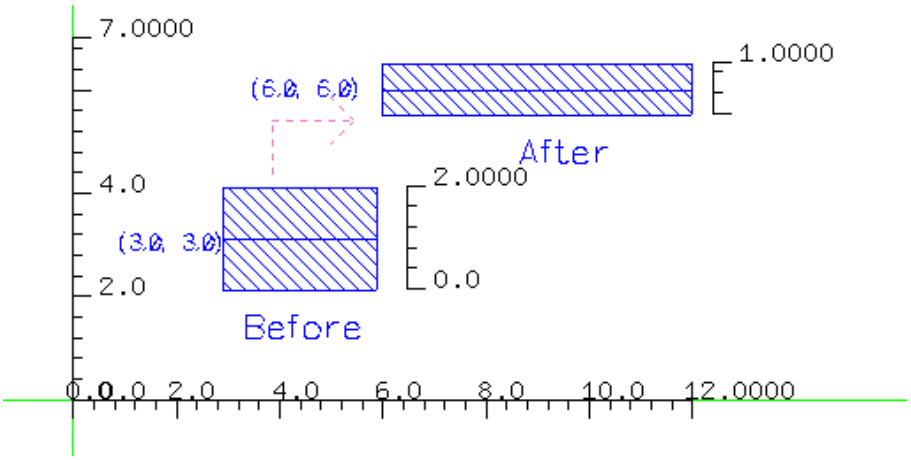
Step 1: Select this path and select Advanced->Shrink, Shrink form appears.

Step 2: Fill out the form with the following information as shown in the figure below.
X Scale: 2.0
Y Scale: 2.0
V Scale: 0.5
Select Mode: By Object.



About the form above,
X Scale specifies the multiple of magnify or shrink coordinate in X axis.
Y Scale specifies the multiple of magnify or shrink in Y axis.
V Scale specifies the multiple of magnify or shrink in width of path, radius of circle, width and degree of arc.

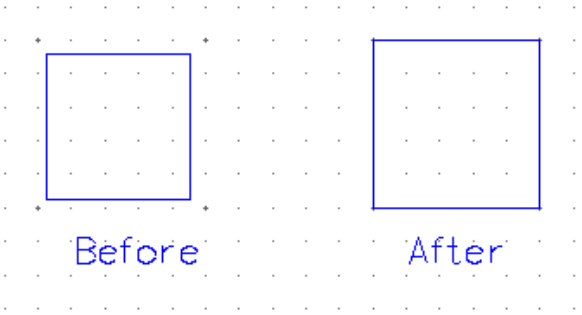
Step 3: Ok the form. The coordinate of center line of path are magnified one time. Width of path is shrunk one time.



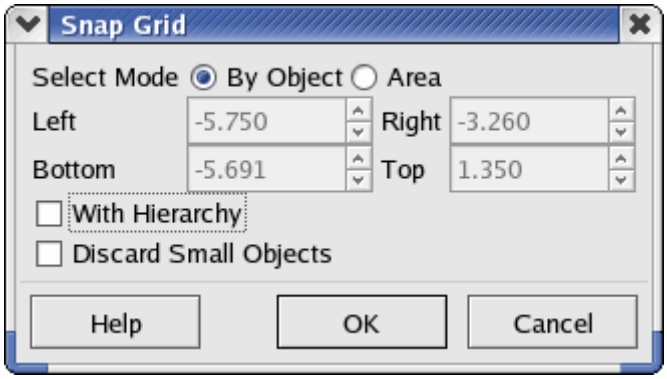
Exercise - 81 Advanced->Snap Grid

This command lets you snap object fit to grid point. Each vertex of object will be snapped to the nearest grid dot.

For example, there is a rectangle whose angle point is not on grid point as shown in left side figure below.



Step 1: Select Advance->Snap Grid, Snap Grid form appears.



Step 2: Set Select Mode to By Object in form above.

Step 3: Select the rectangle from cellview editor window.

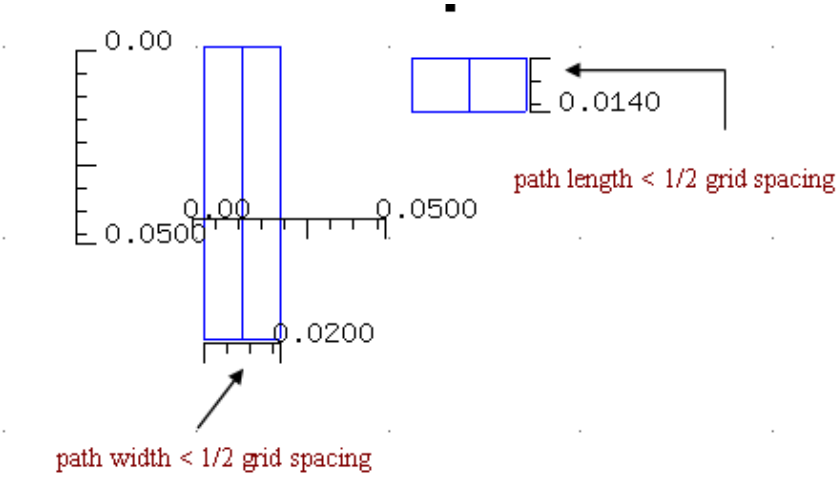
Step 4: Ok the form. The shape of rectangle is adjusted to fit grid point. Please refer to right side figure above.

Note:

When you turn on Discard Small Objects, system will remove the object if the resultant object is unreasonable. Otherwise, system doesn't work on the object.

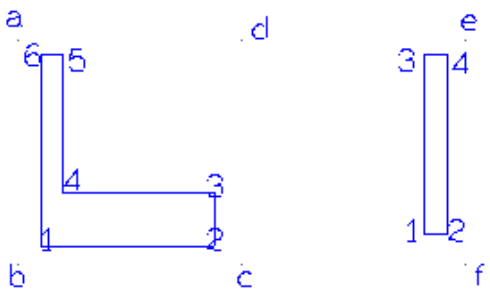
Following objects maybe cause unreasonable resultant object after snapping grid.

1. Width or length of path is less than half of grid spacing as figure shown below.



2. For polygon or rectangle, if some points are snapped to the same point, which will cause an unreasonable final image.

   For example, in the figure below, there are 6 grid points, a polygon and a rectangle. 6 grid points are marked with a, b, c, d, e and f. For polygon, points 1 and 4 are snapped to grid point b, points 2 and 3 are snapped to grid point c, points 5 and 6 are snapped to grid point a. Image on grid points a, b and c doesn't a reasonable object. For rectangle, points1 and 2 are snapped to grid point f, points 3 and 4 are snapped to grid point e, so image on grid points e and f doesn't a reasonable object.



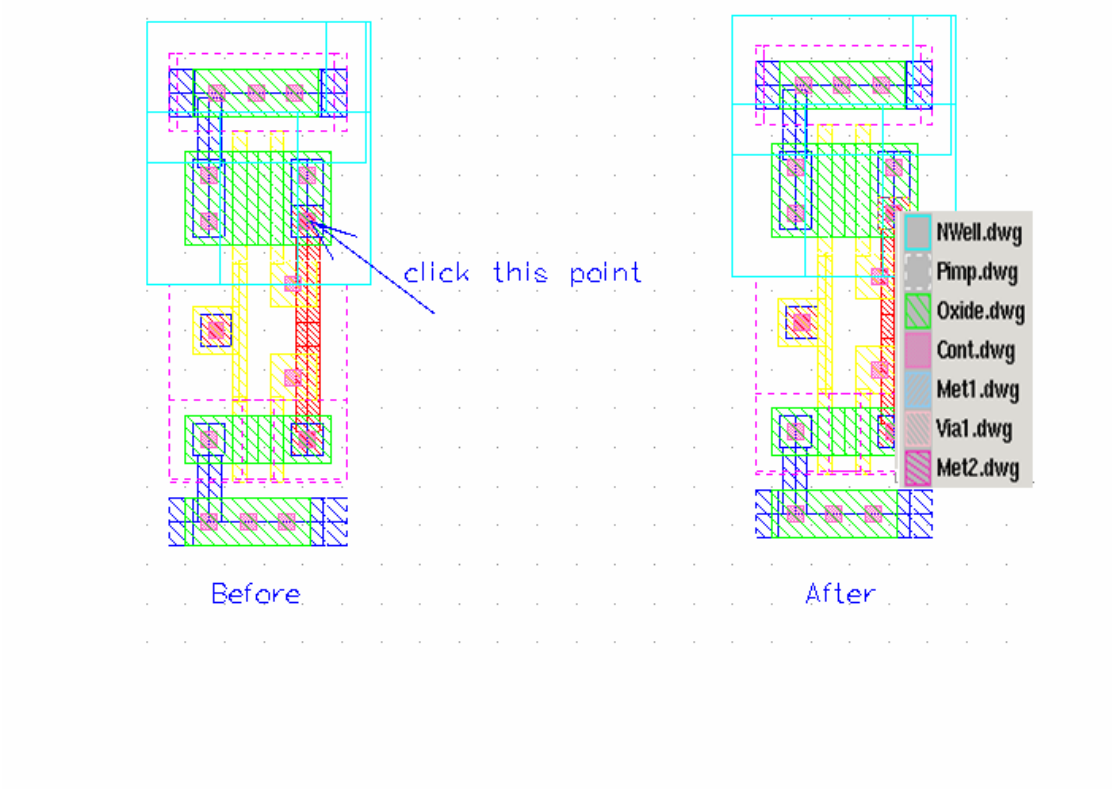
Exercise - 82 Tools->Probe Layers

This command helps you check how many different layers are overlapped at a specified point with full hierarchy.

Step 1: Open cellview “PLL.CLK_INV.layout”.

Step 2: Press “Ctrl+F” to see all objects.

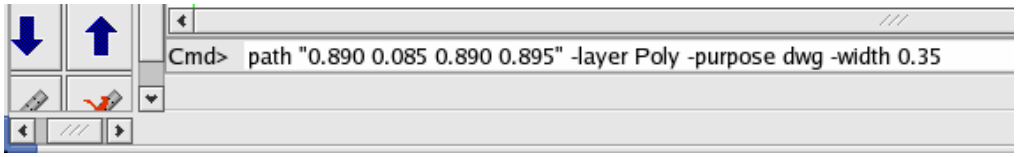
Step 3: Left-Click at a point indicated on the left side of figure below. A small menu will popup after clicking. The menu lists all of different layer names overlapped at this point from top-level to bottom-level. This list is only for looking over, you cannot choose any item from it to select related object in the layer.



Step 4: Close the cellview.

Exercise - 83 Tools->Command Line

This command lets you create an object by typing command script in prompt line at the bottom of the editor window.

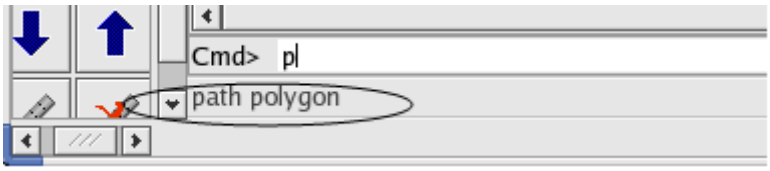


Currently, system provides 4 kinds of command as following.

1. rectangle "*xl yb xr yt*" [-layer *lay_name*] [-purpose *purpose*] [-pin *name*] [-io input/output/inout/switch][-ad all/left/right/top/bottom]
2. polygon "*x1 y1 [x2 y2] [x3 y3]…*" [-layer *lay_name*] [-purpose *purpose*] [-pin *name*] [-io input/output/inout/switch][-ad all/left/right/top/bottom]
3. path "x0 y0 [x1 y1] [x2 y2]…" [-layer lay_name] [-purpose purpose] [-width *wid_num*]
4. instance "*x y*" –lib *lib_name* –cell *cell_name* –view *view_name*

Note:

1. When you fill the prompt line, system helps you match keyword speedily. For example, when you type "p", "path polygon" will appear as below.



Type "a" again and click "Enter", string "path" is added into the command line automatically.

When you type "-", all options about this command will appear too.

2. You can press "up-arrow" key to reload previous commands.

Exercise - 84 Tools->Run Script

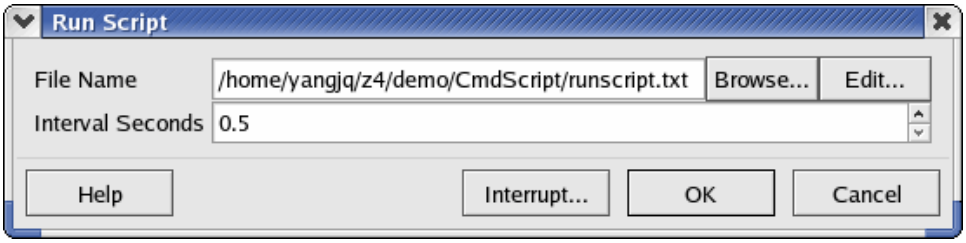
Refer to *Exercise - 83 Tools->Command Line* on page 192, we can gather all command scripts into a file .This command lets you create some objects with this file at one time.

Step 1: Create a new cellview "Lab2.runscript.layout".

Step 2: Zoomin the area around origin to see grid dots clearly.

Step 3: Select Tools->Run Script. Run Script form appears.

Step 4: Fill out the form with the following information as shown in figure below.
File Name : $ZENI_INSTALL_PATH/demo/CmdScript/runscript.txt
Interval Seconds: 0.5

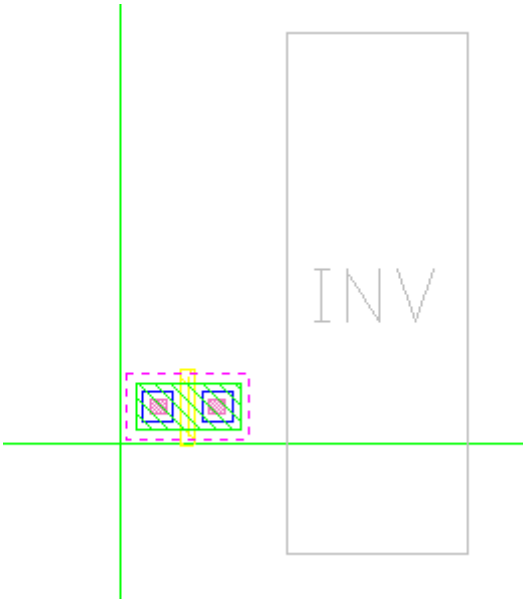


Interval seconds specifies the time interval of execution each command.

Step 5: Click Edit button to open the file as below.

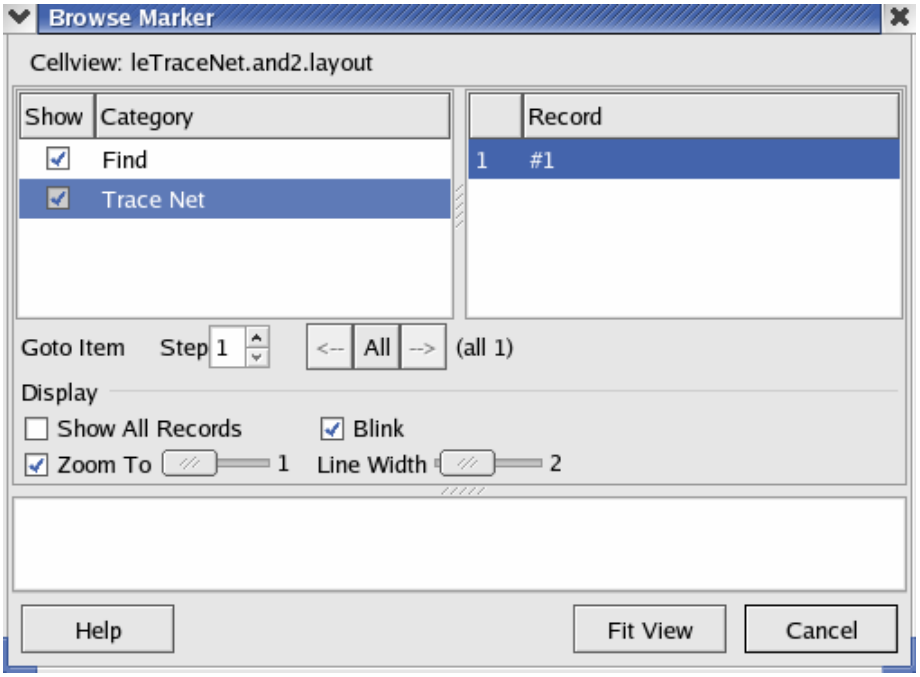
```
rectangle "0.2 0.2 1.58 0.8" -layer Oxide -purpose dwg
path "0.890 0.085 0.890 0.895" -layer Poly -purpose dwg -width 0.2
rectangle "0.4 0.4 0.6 0.6" -layer Cont
rectangle "1.18 0.4 1.38 0.6" -layer Cont
polygon "0.3 0.3 0.7 0.3 0.7 0.7 0.3 0.7" -layer Met1 -purpose dwg
polygon "1.08 0.3 1.48 0.3 1.48 0.7 1.08 0.7" -layer Met1 -purpose dwg
rectangle "0.07 0.061 1.706 0.946" -layer Nimp
instance "2.5 -1.4" -lib PLL -cell INV -view layout
```

Step 6: Ok the form. You can find objects are created one by one within time interval as shown in figure below.



Exercise - 85 Tools->Browse Marker

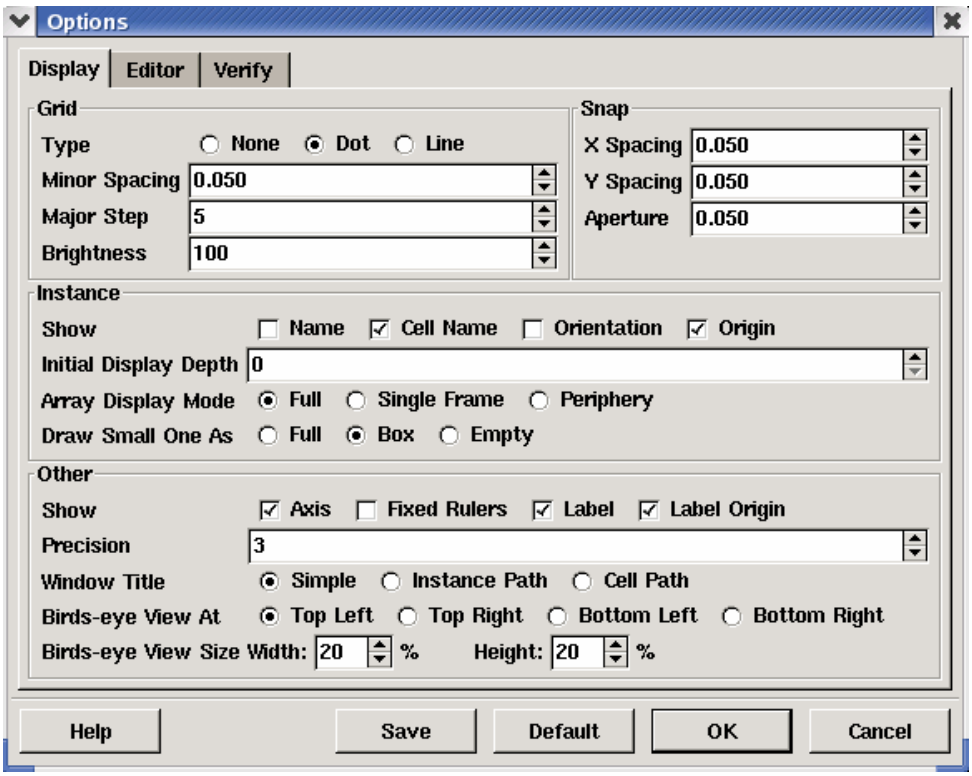
This command lets you look over markers created by Trace Net, Find, DRC result file of Zeni or other EDA tools. If markers are not saved by Design->Save Markers before closing the cellview window, no marker item is shown in Browse Marker form when you open this cellview in the next time. Please refer to *Exercise - 45 Design->Save Markers* on page 133 and *Exercise - 59 Edit->Find* on page 146.



Exercise - 86 Options->Generic

This command lets you set initialization parameters for current design session. You can save all settings to file, which is as default setting for current library.

Display Tab

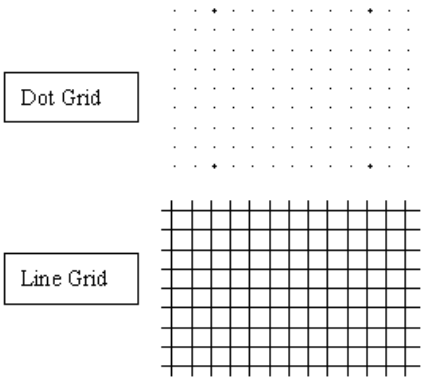


About this form,

Grid field specifies grid shape and distance between two grid points.

Type specifies whether system shows grid in editor or not, or show which type of grid.

- *None* does not show grid
- *Dot* shows grid dot as grid
- *Line* shows line as grid



Minor Spacing specifies the minor spacing between grid points.

Major Step specifies how many grids constitute a grid matrix.

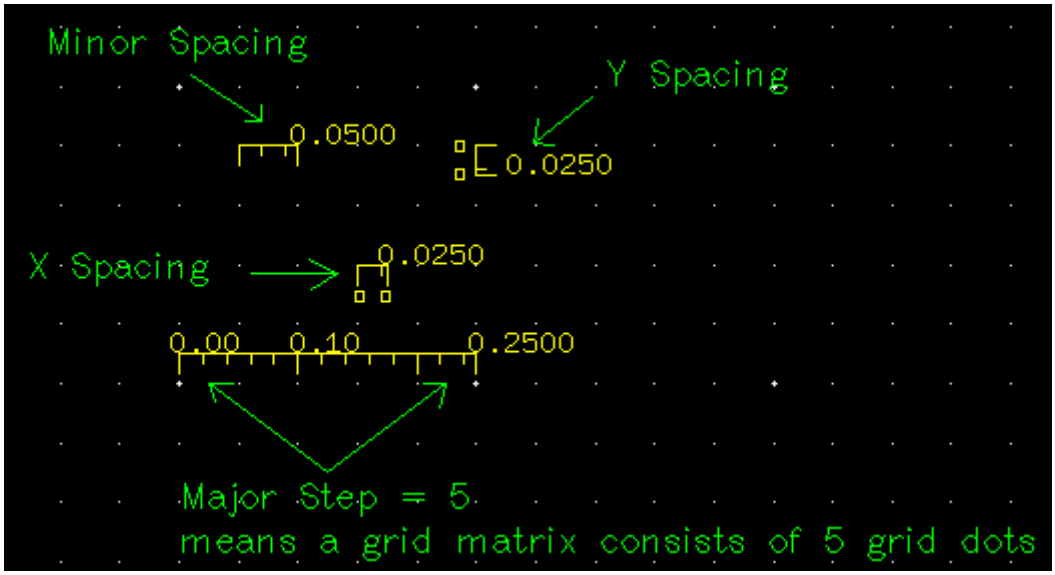
Brightness specifies the brightness of the grid. 100~255 is valid.

Snap field specify the cursor step in editor window.

X Spacing specifies the distance of moving cursor for one step in X direction.

Y Spacing specifies the distance of moving cursor one step in Y direction.

In the following figure, you can understand better of the options above.



Aperture specifies the distance between cursor and object. When the distance between cursor and objects is less than or equal with this value you specify, the object is highlighted, at this moment, if you click the left mouse button, the object will be selected.

Instance field specifies how to display instance.

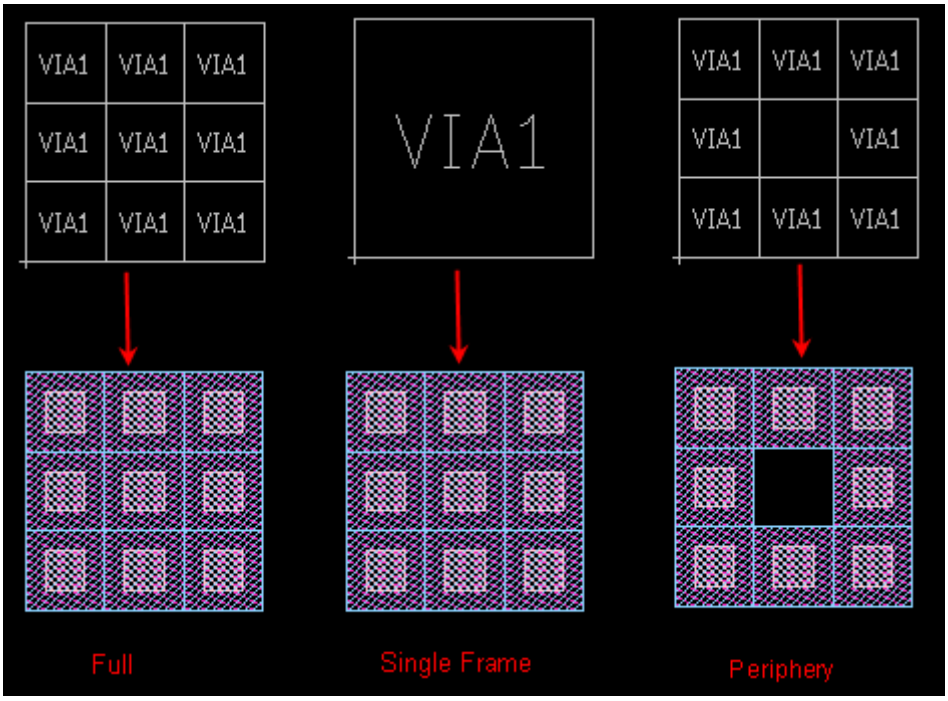
Show specifies whether to display instance properties. These properties are instance name, cell name, orientation and origin.

Initial Display Depth specifies the number of the highest level in the hierarchy visible in the next opening. When you open a cellview, editor will flatten display the cellview till the hierarchy depth you have specified. But a precondition is that you must save this form to file, editor will display object within hierarchy depth according to this file when to open the cellview each time.

Array Display Mode specifies how to display instance array.

- *Full* shows all instances.
- *Single Frame* shows all instances.
- *Periphery* shows the instance around the outside edge of array. But please note that it just do not display the center instance, the content of the instance still exist.

For getting more details about such 3 options, please refer to below figure.



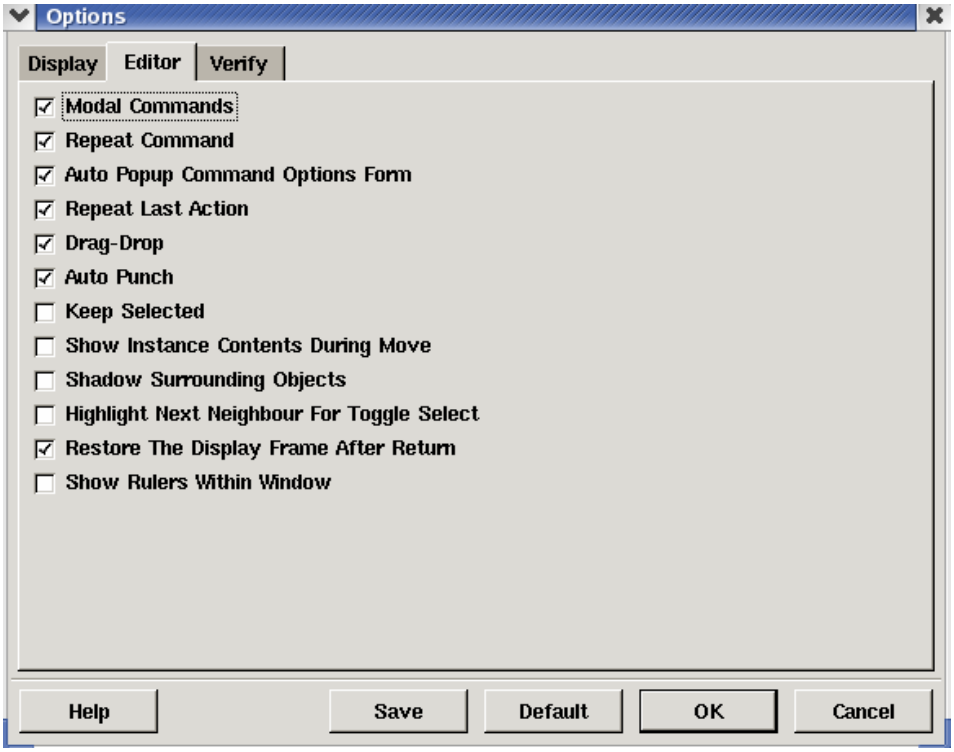
Draw Small One As specifies how editor displays small instances.

- *Full* shows all objects within instance.
- *Box* shows only a boundary of instance.
- *Empty* don't show it.

Other field specifies other items.

Show specifies whether to display 4 kinds of objects in layout editor window. There are *Axis*, *Fixed Rules*, *Label* and *Label Origin*. Here please make special note of “Label” option. If you turn off the option, you can not see label after you creating label.
Precision specifies the precision of the display coordinate.

Editor Tab



Modal Commands specifies whether the previous command is suspended while a new command is being executed or not. Turn on this option, a new command cancels the previous command, otherwise, the previous is suspended till the new command is over.

Repeat Command lets you repeat execution a command till you pressing “Esc” key to terminate it.

Auto Popup Command Option Form specifies whether the system pops up an associated form when execute some commands. If turn off it, press “F3” to popup the form.

Repeat Last Action to control whether to click the right mouse button to repeat executing last command. But if your mouse is not sensitive enough, please turn off the option because it is always do last command when you want to zoomin or zoomout image with clicking the right mouse button.

Drag-Drop controls whether user can drag drop the objects in cellview or not.

Auto Punch controls whether to add via during creating path or not. Please refer to *Exercise - 49*

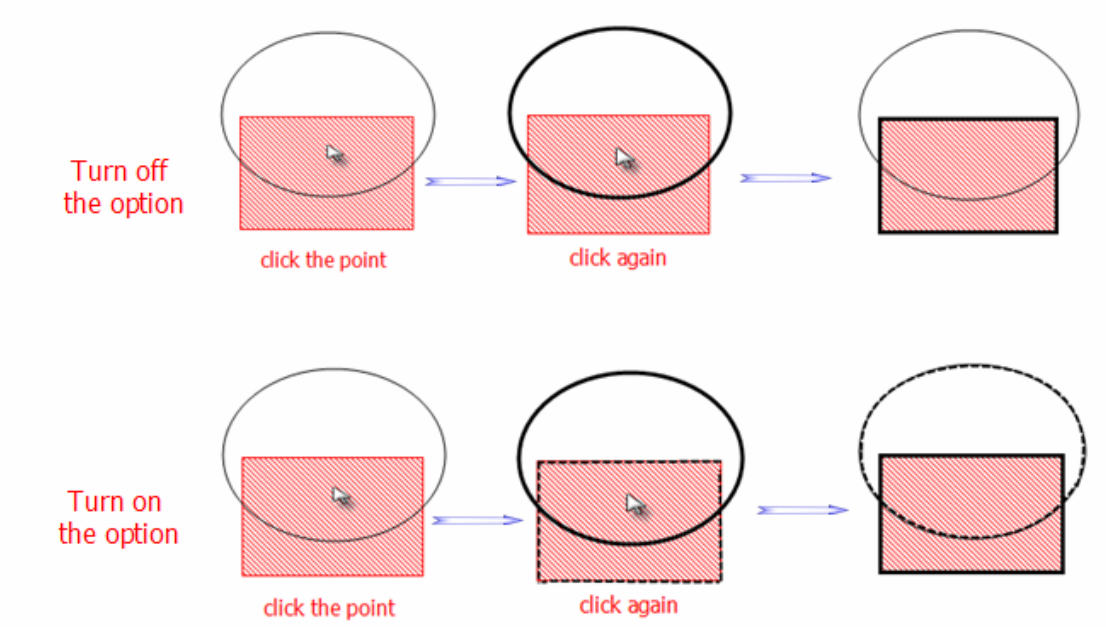
Create->Path on page 134 and *Exercise - 88 Auto Punch* on page 205 .

Keep Selected controls whether to make the selected object keep selected status after an operation or not. For example, copy or move some objects at one time. Please refer to *Exercise - 53 Edit->Copy* on page 139 and *Exercise - 54 Edit->Move* on page 140.

Show Instance Contents During Move controls whether to display all objects within the instance during moving an instance or not. If you turn off it, editor will move a boundary without showing any object within the instance.

Shadow Surrounding Objects controls whether shadow objects out of selected object when you do *Enter/Enter One Step*, the shadow color is set in *Options->Color->Shadow*. Please refer to *Exercise - 71 Hierarchy->Enter/Descend* on page 170 and *Exercise - 72 Hierarchy->Enter one Step/Descend one Step* on page 173.

Highlight Next Neighbour For Toggle Select controls whether to display the next object which will be highlighted with dashed line when you click left mouse button continuously to select overlapped object one by one.
Look over the following case, there are two objects overlap, when you turn off this option., click left mouse at the common area, an object will be selected, click again, another object will be selected. When you turn on this option, an object will be highlighted with dashed line while another object is selected.

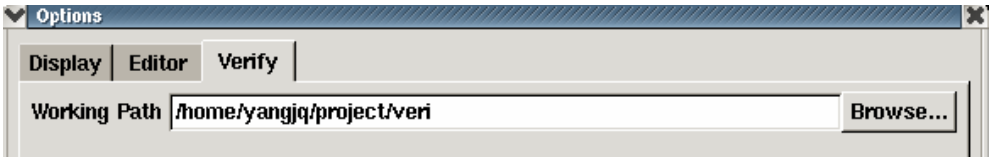


Restore The Display Frame After Return controls whether to fit the upper level cell when you return from a lower level cell whose view is changed.

Show Rules Within Window controls whether to display ruler which is created in lower level cell in upper level cell window. For example, when you use command “Hierarchy->Enter” or

“Hierarchy->Enter One Step” to enter into a lower level cell and make measure in this lower level cell, if you turn on this option, the ruler will be displayed in upper level cell when you return to the upper level cell.

Verify Tab defines the working path of layout verification.



6.4.5. Advanced Features

Exercise - 87 VCell

VCell is an abbreviation of Variable Cell. It is a special cell and is created by translating vcell template file. The vcell template file consists of TCL script and build-in commands of Layout Editor.
Vcell is used to create foundational device cells, such as MOS, Resistor, Capacitor, Inductor and so on. You can automatically create different device cell through modifying the device parameters in graph user interface.

To understand Vcell more in details, please do the following.

1) Create a vcell

Step 1: In Design Manager, select library “PLL”.

Step 2: Select Tools->Vcell Template.

Step 3: In Vcell Template form, select PMOS item and click Edit button to read it.

Step 4: The file “PMOS.vcell” is opened in a text editor.

Step 5: Look at the title of text editor, “PMOS.vcell” locates in directory “../libname/.vcell”.

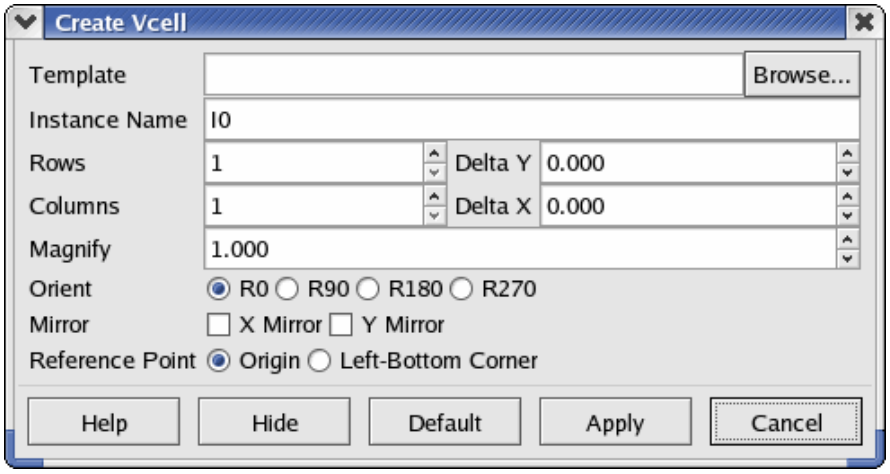
Step 6: Viewing whole file, the file includes two “proc” functions. The first function defines device parameters, the second one defines how to create a PMOS device.

Step 7: Close this text editor.

Step 8: Close the Vcell Template form.

Step 9: Create a new layout celliview “Lab2.vcell.layout”.

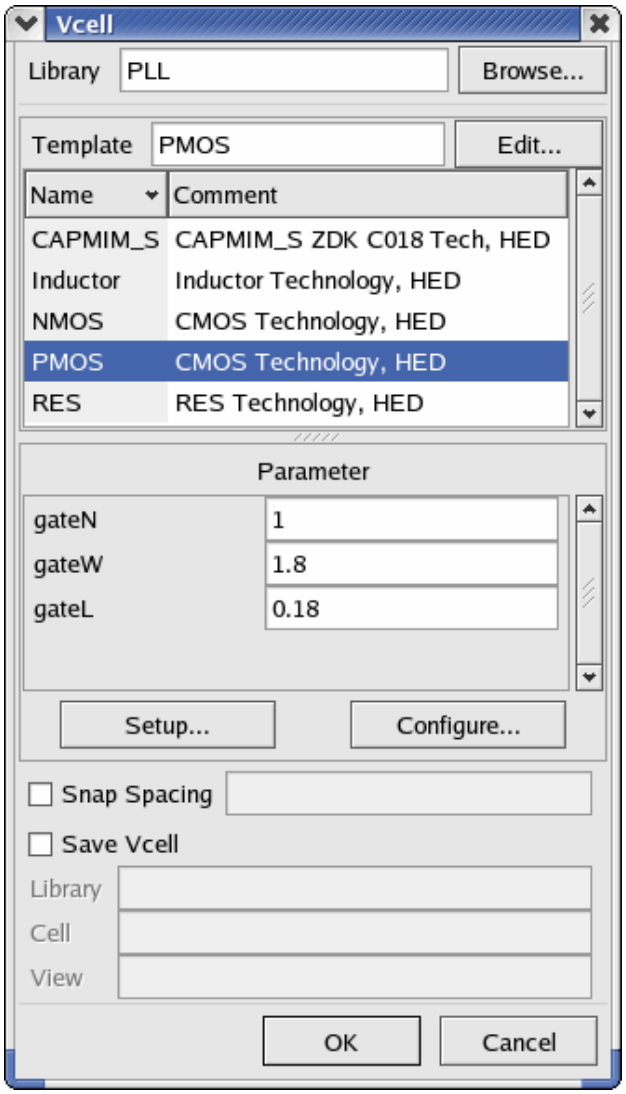
Step 10: Select Create->Vcell or click icon , the Create Vcell form appears.



Step 11: Click Browse button to select a vcell template file.

Step 12: In Vcell form, Library field is set to current library “Lab2” automatically. Because there is no any vcell template file in “Lab2”, click Browse button to select library “PLL”.

Step 13: All vcell templates are added in Vcell form. Click PMOS item, the parameters defined in file “PMOS.vcell” are shown by GUI.



Step 14: Click Edit button to open the PMOS.vcell file. You can find the parameters in form come from the definition of the vcell template file.

Step 15: Click Setup button, Setup Vcell form appears. All of layer definitions are shown in this form.

Step 16: Cancel the Setup Vcell.

Step 17: Click Configure button, Configure Vcell form appears. All of connection modes defined in vcell template file are shown in this form.

Step 18: Cancel this form.

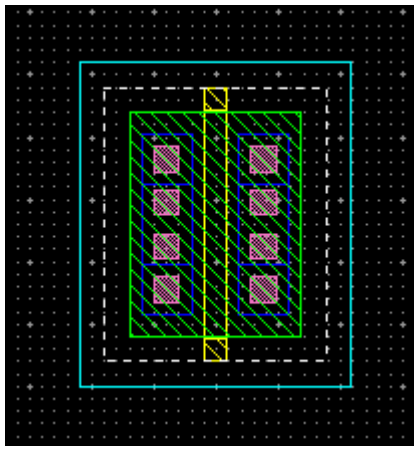
Step 19: Ok the Vcell form.

Step 20: In Create Vcell form, Template field is set to “PLL.PMOS”.

Step 21: Hide the form.

Step 22: Click at a point to place the vcell “PMOS” in cellview window.

Step 23: Press “Ctrl+F” to show all of the layers of vcell “PMOS”.



Because vcell is only a virtual cell, not a really cell, you cannot modify it directly. Select Edit->Vcell->Re-Parameters to modify it.

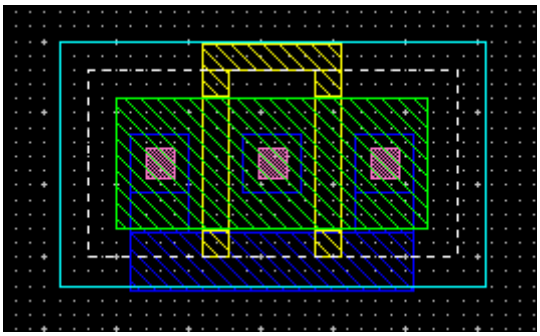
2) Modify Vcell parameters

Step 24: Select the vcell first.

Step 25: Select Edit->Vcell->Re-Parameters, the Re-Parameterize Vcell form appears.

Step 26: Modify the parameters as the following information:
gateN: 2
gateW: 1.8
gateL: 0.18
Configure: gate-connect:Top
sd-connect:BottomLeft

Step 27: Ok the Re-Parameterize Vcell form, the vcell PMOS is changed as shown below.

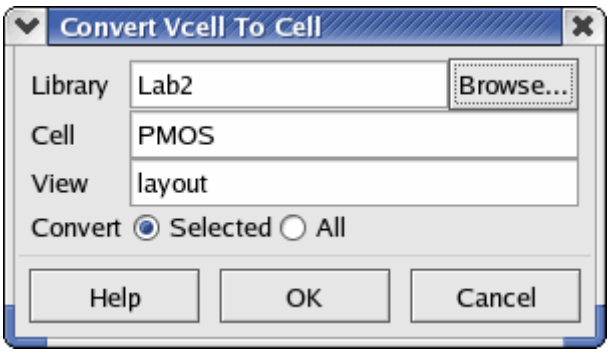


You can also convert the vcell PMOS to normal cell in order to modify it.

3) Convert Vcell to Normal Cell

Step 28: Select vcell PMOS first.

Step 29: Select Edit->Vcell->Convert To Cell. The Convert To Cell form appears.

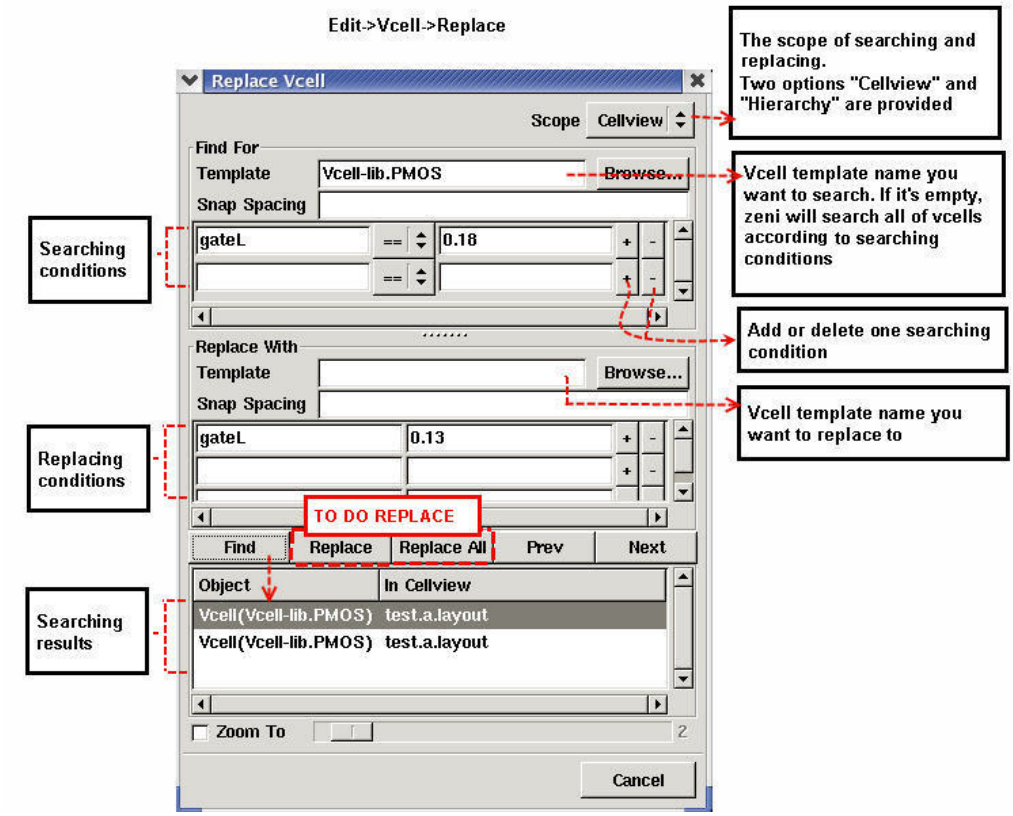


Step 30: Ok the form to create a normal cellview in current library “Lab2”. In this form, you can also decide which vcell is converted. *Selected* only converts the selected vcell; *All* converts all of the vcell created from the same vcell template by the same parameters.

Step 31: Now, the vcell PMOS in the cellview window becomes an instance of cell “Lab2.Pmos”.You can click icon to enter and modify the cell directly.

4) Find & Replace Vcell

Command “Edit->Vcell->Replace” to find or replace vcell according to conditions you defined.



Note:

1. All vcell template files in a library is stored under path "../libname/.vcell".
2. The prefix must be the same as the string in front of the first function name "SetParameter" in vcell template file. For example, the vcell template file named "PMOS.vcell", open this template file; the first function name must be "PMOSSetParameter".
3. The suffix of vcell template file must be ".vcell".
4. Please refer to *7 Appendix ---VCell Template* on page 215 to know more.

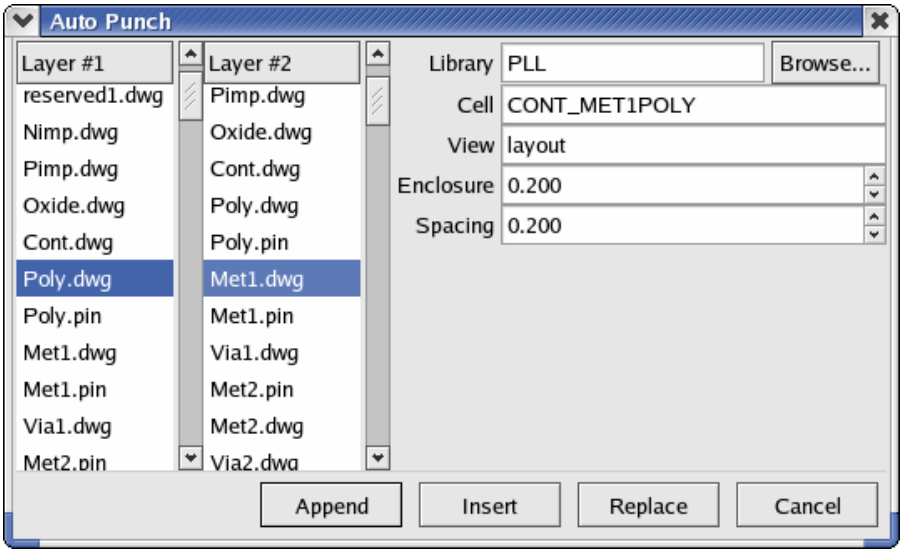
Exercise - 88 Auto Punch

This command lets system add via or contact cell at the cross of two paths automatically.
Please note that this command can work only under the circumstances that Auto Punch rules are specified in Technology Center.

Step 1: Create a new layout cellview "Lab2.autopunch.layout".

Step 2: In Layer Palette form, on the left side of the editor window, select File->Technology Center. The Technology Center window appears.

Step 3: In the window, there are four rules of “Auto Punch”. Select the first rule item and double click it, an Auto Punch form appears.



The figure illustrates adding a contact cell “PLL.CONT_MET1POLY.layout” at the cross of layers “Poly.dwg” and “Met1.dwg”. The distance between contact and layer is equal to or less than 0.200; if you want to add more contact at the cross of paths, the distance between both contacts is equal to 0.200.

Step 4: You can modify or add auto punch rule in this form.

Step 5: Cancel Auto Punch form and close the Technology Center window.

Step 6: In Layout Editor window, turn on the option Options->Generic->Auto Punch.at Editor tab.

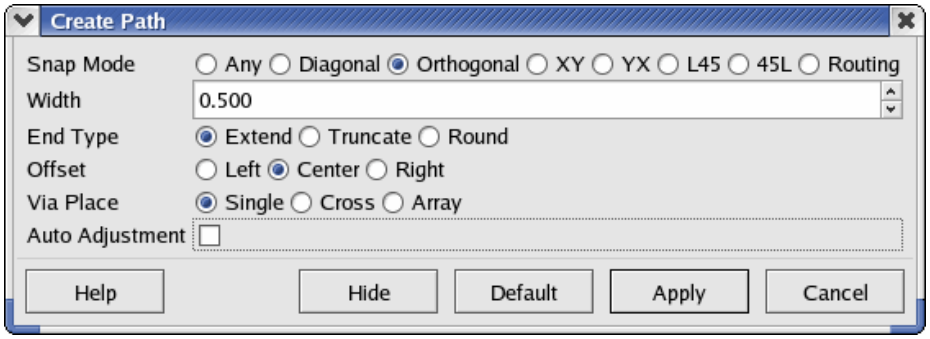
Step 7: Zoomin the window.

Step 8: Select “Poly.dwg” from Layer Palette.

Step 9: Click icon and press “F3” to popup related form.

Step 10: Fill out the Create Path form with following information and is shown as the figure below.

Snap Mode: Orthogonal
Width: 0.500
End Type: Extend
Offset Center
Via Place: Single
Auto Adjustment: off

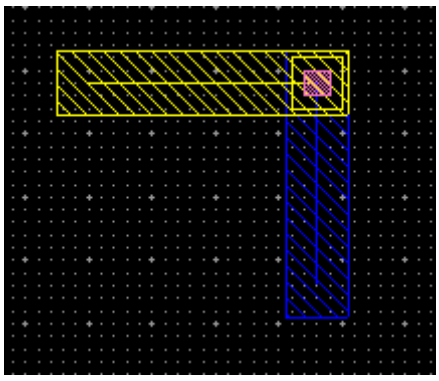


Step 11: Hide the form.

Step 12: Click a point to start path and move cursor along X or Y axis.

Step 13: Don't release mouse at the end of the path, click middle button and choose "Met1.dwg" from popped up the little menu.

Step 14: A contact cell is added at the cross of paths automatically. At this time, the active layer becomes "Met1.dwg".



Step 15: Continue move mouse to create path with "Met1.dwg". Click middle button at the end of the path, a little menu pops up. This menu includes two layers "Poly.dwg" and "Met2.dwg", which because "Met1.dwg" is related to two rule of Auto Punch.

Step 16: Select "Met2.dwg" from little menu, a via cell "VIA1" is added.

Step 17: Continually move mouse to create path with "Met2.dwg" and double click to terminate the command at the end of the path.

For adding more via or contact cells, you should,

Step 18: Increase the path width to 2.5.

Step 19: Set *Via Place* to *Cross* or *Array* in Create Path form.

Step 20: With the above methods to add more via or contact cells at the cross two paths.

Step 21: Save and close the cellview.

 **Note:**

You must turn on Options->Generic->Editor ->Auto Punch before auto punch.

Exercise - 89 Automatically Adjust Spacing

When the distance between two paths is less than the DRC rule, this command helps you increase spacing to meet the rule.

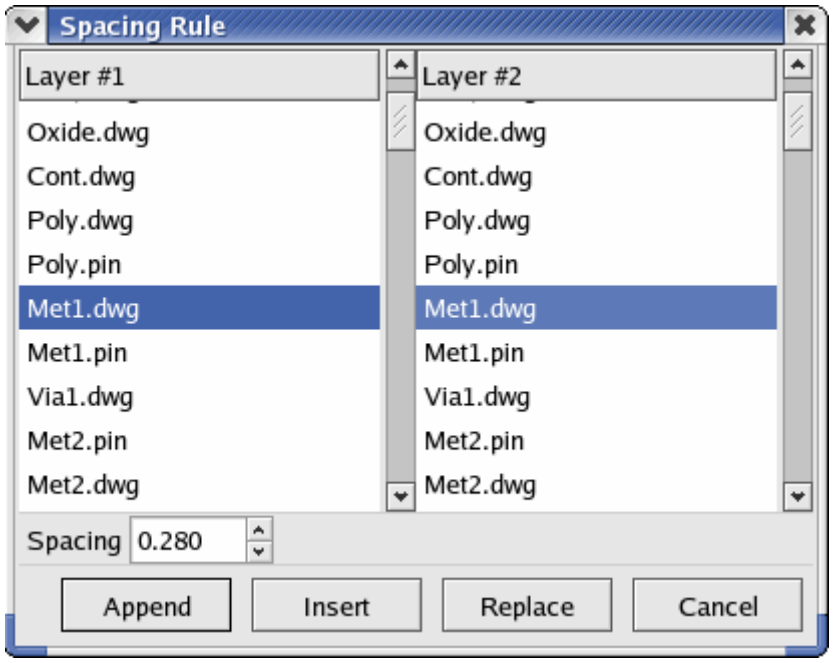
This command can work only under the circumstances that Spacing Rule is specified in Technology Center window.

Step 1: Create a new cellview “Lab2.adjustspacing.layout”.

Step 2: In Layer Palette form, on the left side of the editor window, select File->Technology Center. The Technology Center window appears.

Step 3: Select Spacing Rule item, the related rules show in the right of the window.

Step 4: Double click the first item, Spacing Item form appears as below. The form lets you add or modify a spacing rule



Step 5: Cancel Spacing Item form and close the Technology Center.

Step 6: In Layout Editor window, create a path with “Met1.dwg”.

Step 7: Don’t cancel this command and continue to click at a point to create the 2nd path.

Step 8: If the distance between both paths is less than 0.28 microns, the 2nd path will move to suitable position automatically when you click at the end of the 2nd path while creation.

Step 9: Save and close the cellview window.

Exercise - 90 Realtime DRC

This command helps you check simple DRC violations while you editing.
Please note that this command can work only under the circumstances that the DRC Rule is specified in Technology Center.

Step 1: Create a new cellview “Lab2.rtdrc.layout”.

Step 2: In Layer Palette form, on the left side of the editor window, select File->Technology Center. The Technology Center window appears.

Step 3: Click DRC Rule item, the rules for DRC are shown on the right of the form. You can write other DRC rules in this field directly. Click Clear to clean up all of the rules.

Currently, system supports 6 kinds of the DRC commands. They are AREA, WIDTH, ENC, INT , EXT and LENGTH.

The parameters of each command are shown as table below.

Command	Parameter
AREA	EQ, NE
WIDTH	LT, GT, LE, GE, EQ, NE
EXT / ENC / INT	LT, LE, EQ
LENGTH	LE, LT, GT, GE

The system is not sensitive to name case of command and parameter.
You can omit the suffix of layer name. The default suffix is ".dwg".

For example, following three DRC commands are equal.

- "WIDTH Met1.dwg LT 0.28",
- "width Met1.dwg lt 0.28"
- "width Met1 lt 0.28" are valid.

Step 4: Click Verify->Start Realtime DRC

Step 5: Select Met1.dwg from Layer Palette, and click icon  to start Create Path command.

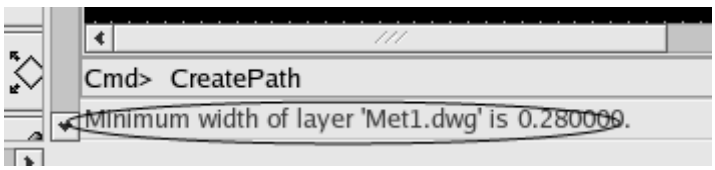
Step 6: Press "F3" to popup the related form and modify the path width to 02.

Step 7: Hide the form.

Step 8: Zoomin an area.

Step 9: Click at a point to start a path.

Step 10: While mouse moving, the outline of the path is highlighted and a message appears in the bottom of the window. The message tells you this operation violates which DRC rule.



Step 11: Click Verify->Stop Realtime DRC to terminate this command.

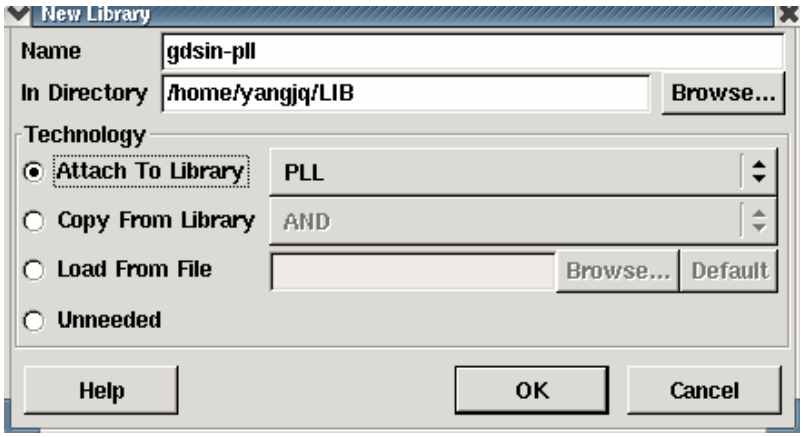
Step 12: Save and close the cellview window.

Exercise - 91 GDS-IN

This tool translates Graphical Design Stream (GDS) format to Zeni database format. We call this

tool "GDS-In" for short.

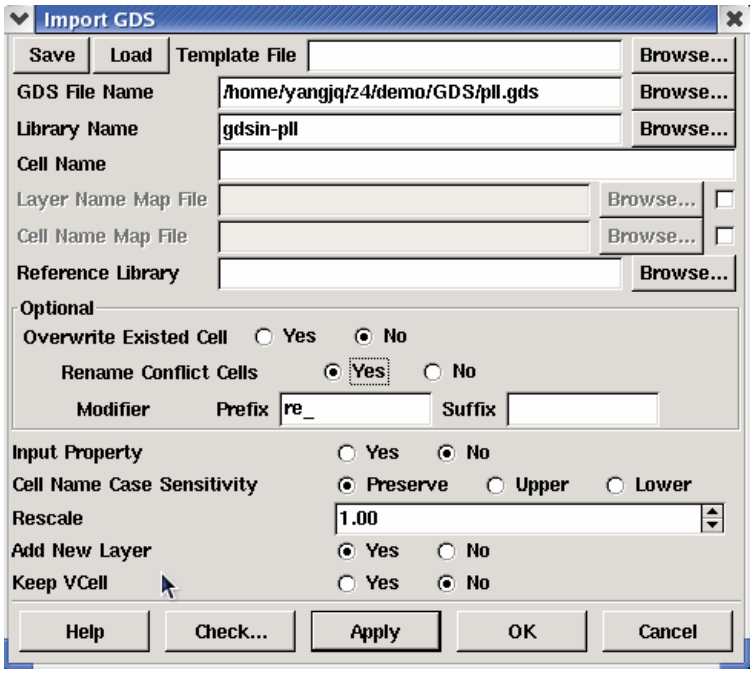
Step 1: Create a new library “gdsin-pll” with the technology file of library “PLL”.



Step 2: In Design Manager window, left click library “gdsin-pll”.

Step 3: Click File->Import->GDS, and fill out the GDS form with the following information.

GDS File Name: $ZENI_INATLL_PATH/demo/GDS/pll.gds
Library Name: gdsin-pll



Step 4: Ok the form. The process of gds-in is shown in Zterm-gdsin window. In this window, you can look over the summary of translation. Click Save Report File to store the information to a file.

Step 5: Close the window.

Step 6: Open cellview “gdsin-pll.PLL.layout” to see the top level cellview.

 **Note:**

1. If you need import a layer map file at the same time, please turn on the Layer Name Map File option and fill out the field. Under Zeni environment, the layer map file named “gds.map” is in path “<target library> /.technology”.

2. If you need import a cell map file at the same time, please turn on the Cell Name Map File option and fill out the field. Under Zeni environment, the cell map file named “cell.map” is in path “<target library> /.technology”.

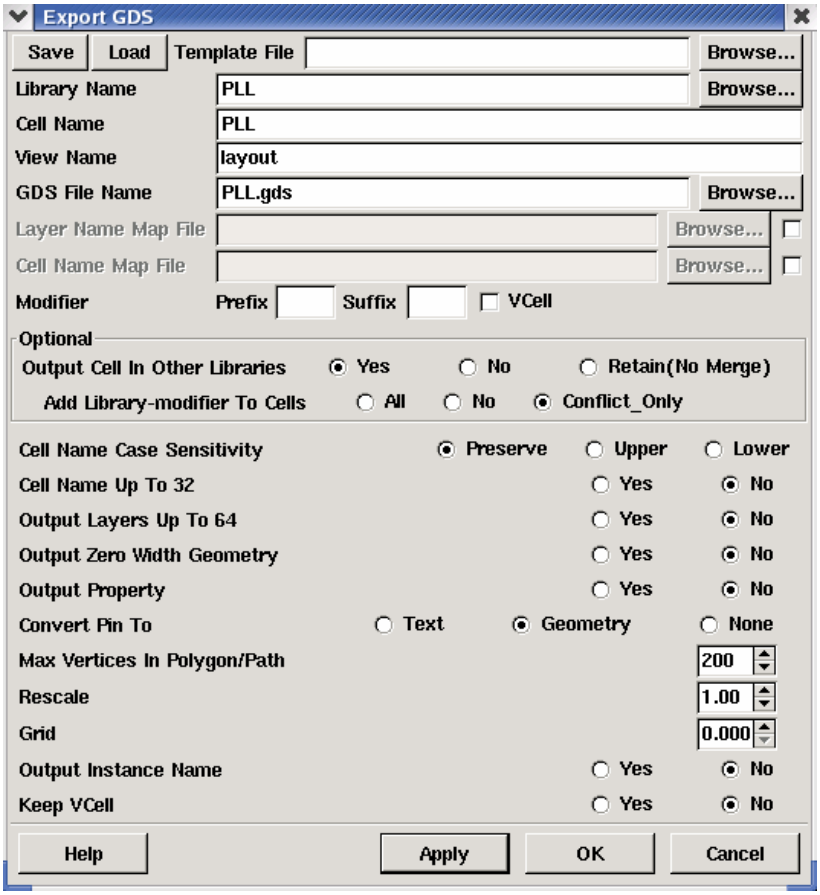
3. If there exists instance referenced relationship only, not any instance of sub-cell information, GDS-In will match the sub-cell name from cells of library specified by *Reference Library* field first one by one. If matching failed, GDS-In will match it again from library specified by *Library Name* field. Please refer to *Retain(No Merge)* option in *Exercise - 92 GDS-OUT* below.

Exercise - 92 GDS-OUT

This tool translates Zeni database format to Graphical Design Stream (GDS) format. We call this tool "GDS-Out" for short.

Step 1: In Design Manager window, click “PLL.PLL.layout”.

Step 2: Click Files->Export->GDS.



About the form above,

If you want to export layout data according to Layer Name Map File or (and) Cell Name Map File, please turn on the two options and fill out them. Layer name map file can be created in Technology Center and is named “gds.map”, saved in path <soure library> /.technology”. <source library> is a library name specified in Library Name field. Please refer to *Exercise - 43 Technology Center* on page 127. Cell name map file can be edited by yourself with following formats:

1. Comment line starts with “;” or “#”.
2. A cell map definition item is a line.
3. Each definition item looks like “Zeni_lib_name Zeni_cell_name Zeni_view_name gds_cell_name”.

Modifier specifies GDS-Out adds the prefix or (and) suffix to all of cell names and export it to GDS file.

Output Cell In Other Libraries specifies whether system exports information of sub-cell in other library to GDS file or not.

- *Yes* specifies system exports content of sub-cell and reference relationship.
- *No* specifies system exports neither content of sub-cell nor reference relationship.
- *Retain(No Merge)* specifies system export the reference relationship only. About this option, please refer to *Reference Library* option in *Exercise - 91 GDS-IN* on page 210.

When you selected *Yes* or *Retain(No Merge)* options, system will add the prefix " <library>_" to sub-cell name. Here, the <library> is a name of library that the sub-cell is in. For example, if the sub-cell "a" is from library "LibA", system will rename sub-cell name to "LibA_a" in GDS file.

- *All* specifies system adds the prefix to all of sub-cell in other library.
- *No* specifies system does not to add the prefix.
- *Conflict_Only* specifies system adds the prefix only for sub-cell that name is same as cell in up-level.

Step 3: Ok the form. The process of gds-out is shown in Zterm-gdsout window. In this window, you can look over the summary of translation. Click Save Report File to store the information to a file.

Step 4: Close the window. The gds file "PLL.gds" will be created.

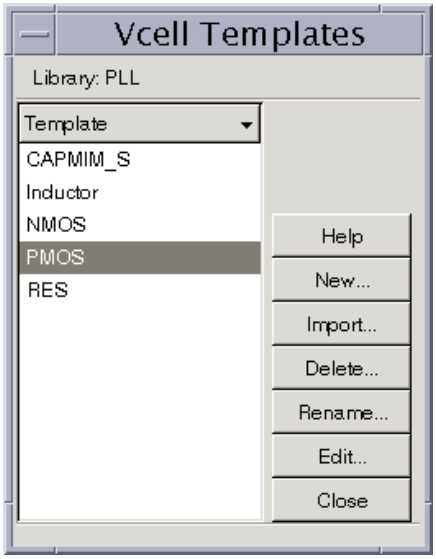
7. Appendix ---VCell Template

7.1. About VCell

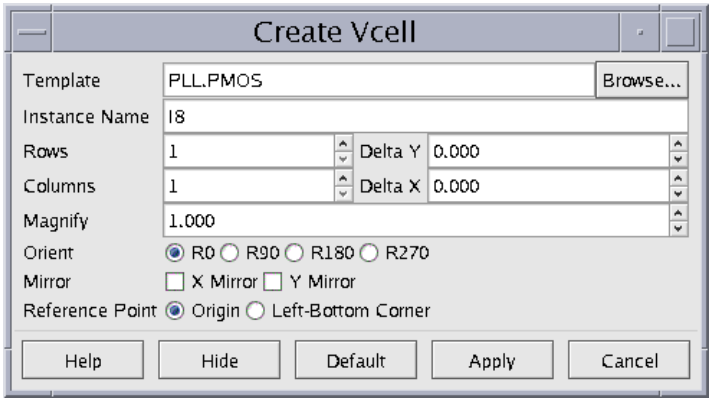
VCell is an abbreviation of Variable Cell. ZeniPDT translates vcell template file to create device cell automatically. Vcell template is based on TCL language and build-in command of Zeni. It defines the device parameters and device internal structure. You can modify the parameters to meet different technology requirement.

7.2. Using VCell

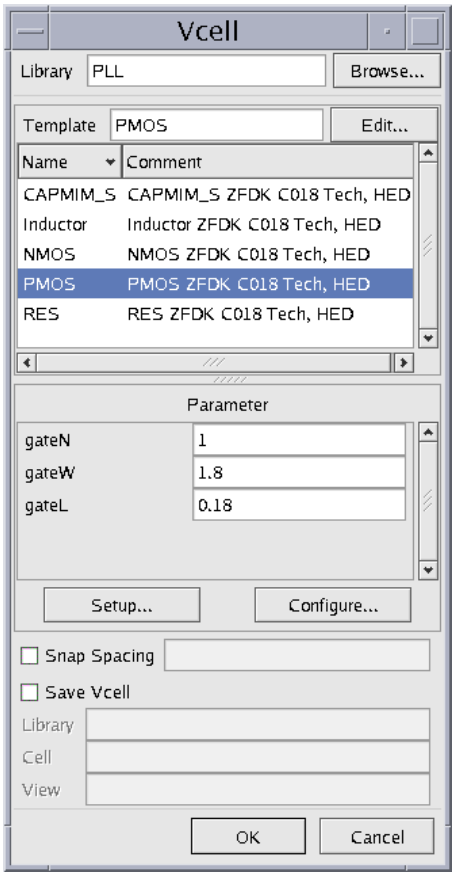
All vcell template files of a library are stored in folder ".vcell" under library path. Each template name ends with ".vcell" as its postfix. For example, "PMOS.vcell", "RES.vcell".
In Zeni Design Manager window，Tools->Vcell Templates lists all vcell templates of library selected from Design Manager window. You can modify these templates by the GUI.



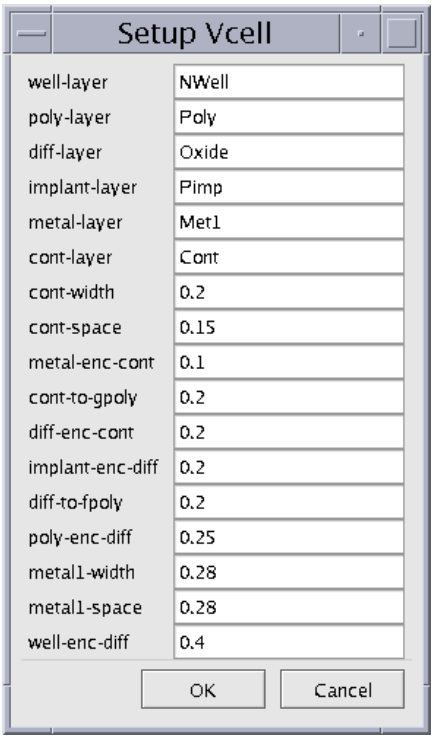
In ZeniPDT, select Create->Vcell. Create Vcell form appears as below.



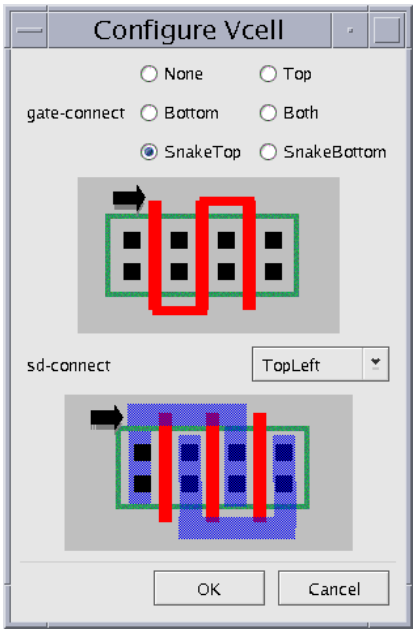
Click Browse... button to select a vcell template.



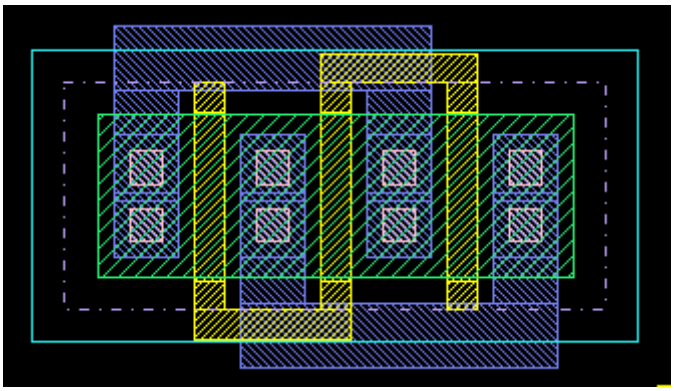
In the above form, select a template item, all parameters of the template will be shown in GUI mode. Click Setup... button to look over or modify the information about technology layers and design rules.



Click  button in Vcell form to select device internal connection mode.



For example, modify parameters gateN = 3，gateW=3，gateL=0.18 and click Apply in Create Vcell form, a PMOS layout is created in ZeniPDT automatically.



Note:

Variable cell cannot be modified as it is a particular cell. But you can convert it to normal cell by command Edit->Vcell->Convert To Cell first.

7.3. VCell Template Component

VCell template is based on Tcl language and build-in command of Zeni. It is composed of two parts: device parameter and device internal structure.

7.3.1. Device Parameter Definition

The part defines 5 kinds of variables as following:

1. VcellComment -- comment information
2. VcellParameter -- default device parameter
3. VcellSetup -- device layers and design rules
4. VcellConfig -- device internal connection mode
5. VcellFunction -- device implementation function

This 5 kinds of variables are defined in one function as following.

proc *Vcell_Name***SetParameter {} {**

global VcellComment VcellParameter VcellFunction VcellSetup VcellConfig
 # set default value for each parameter
set VcellComment "*Technology and author information***"**

set VcellParameter(*index***)** **"***Para_Name Value***"**
set VcellParameter(type_ *index***)** **"radio", "option" or "check"**
set VcellSetup(*index***)** **"***Tech_layer Layer_name***"**
set VcellSetup(*index***)** **"***Design_Rule Rule_Value***"**
set VcellSetup(type_ *index***)** **"radio", "option" or "check"**
set VcellConfig(*index***)** **"***Config cfg1 cfg2 cfg3***"**
set VcellConfig(type_ *index***)** **"radio", "option" or "check"**

set VcellConfig(icon_ *index _cfg1***)** **"***icon1..gif***"**
set VcellConfig(icon_ *index _cfg2***)** **"***icon2.gif***"**
set VcellConfig(icon_ *index _cfg3***)** **"***icon3.gif***"**

set VcellFuntion **"***Vcell_Function_Name***"**
}

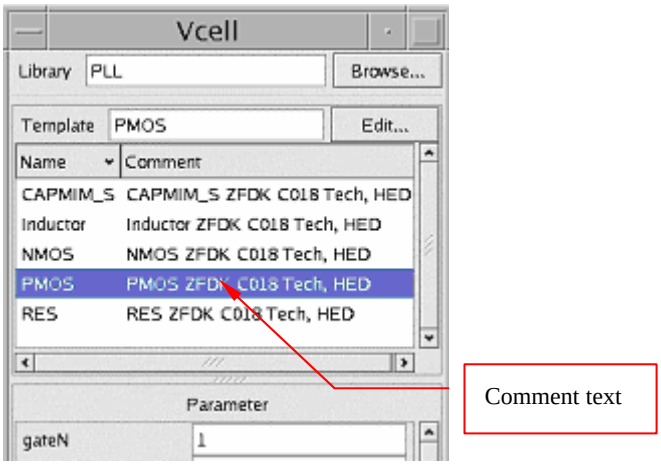
Note:

1. Comment line starts with "#".
2. In the above function, **black-bold** font is key words,you cannot modify it. *Italic* font can be modified.
3. *Vcell_Name* is device name, it must be the same prefix as template file name. For example, the function name is "PMOSSetParameter", the vcell template file must be "PMOS.vcell".
4. **VcellParameter** and **VcellFuntion** must be defined, other variables can be ignored.

7.3.1.1. VcellComment

This variable stores comment text. For example technology name and author name. The information will be shown in Vcell form. "VcellComment" variable can be ignored in vcell template file.

For example： set VcellComment " PMOS ZFDK C018 Tech, HED"



7.3.1.2. VcellParameter

This variable must be an array data. It stores device parameters. These parameters will be shown in "Parameter"list in Vcell form. This variable must exist in vcell template file.

Definition of variable format:

set VcellParameter(*index***)** "*parameter value*"

or

set VcellParameter(*index***)** *"parameter value1 value2 … valuen"*
set VcellParameter(type_*index*) **"radio", "option" or "check"**

" index " as the subscript of array starts from ' 1 '.

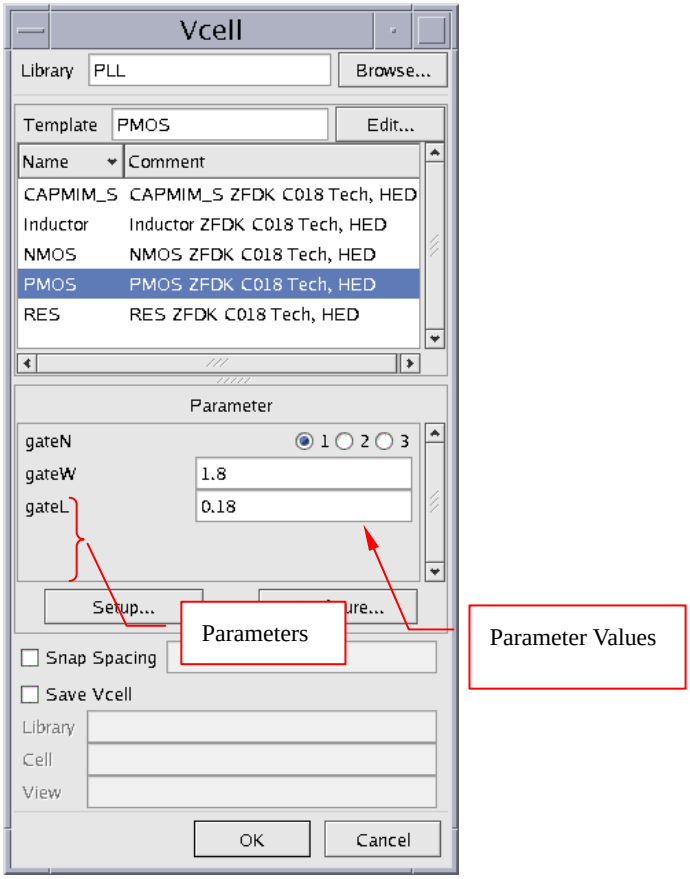
For example,
set VcellParameter(1) "gateN 1"
set VcellParameter(2) "gateW 1.8"
set VcellParameter(3) "gate L 0.18"

If a parameter has multiple values, list all the values with blank characters separate them.
For example,
set VcellParameter(1) "gateN 1 2 3"

But this mode needs "**VcellParameter(type_***index*)" to specify the placement of values. Here "radio" and "option" are available.
For example,
set VcellParameter(type_1) "radio"
or
set VcellParameter(type_1) "option"

If there are following definitions, the Vcell form shows them as below figure.

```
set VcellParameter( 1 )          "gateN  1 2 3"
set VcellParameter(type_1)       "radio"
set VcellParameter( 3 )          "gate L   0.18"
set VcellParameter( 2 )          "gateW   1.8"
```



7.3.1.3. VcellSetup

This variable must be an array data. It stores layer information and design rules. In Vcell form, click  button to look over these information. This variable can be ignored.

Variable format is as below:

set VcellSetup(*index***)** "*parameter value*"

or

set VcellSetup(*index***)** "*parameter value1 value2 … valuen*"

set VcellSetup(type_*index*) **"radio", "option" or "check"**

" index " as the subscript of array starts from ' 1 '.

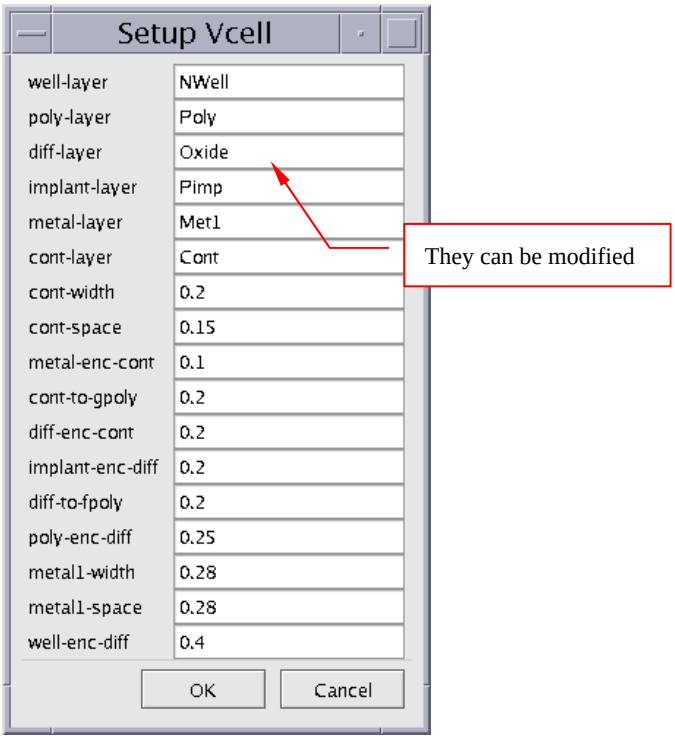
For example:

set VcellSetup(1) "cathode_comp comp"
set VcellSetup(2) "cathode_nplus nplus"
set VcellSetup(3) "anode_comp comp"

VcellSetup also defines multiple value for a parameter as VcellParameter. "**VcellSetup (type**_*index*)" specifies the placement of values.

If there are following definitions, Setup Vcell form shows them as below figure.

```
set VcellSetup(1) "well-layer              NWell"
set VcellSetup(2) "poly-layer              Poly"
set VcellSetup(3) "diff-layer              Oxide"
set VcellSetup(4) "implant-layer           Pimp"
set VcellSetup(5) "metal-layer             Met1"
set VcellSetup(6) "cont-layer              Cont"

set VcellSetup(7)   "cont-width            0.2"
set VcellSetup(8)   "cont-space            0.15"
set VcellSetup(9)   "metal-enc-cont        0.1"
set VcellSetup(10)   "cont-to-gpoly        0.2"
set VcellSetup(11) "diff-enc-cont          0.2"
set VcellSetup(12) "implant-enc-diff       0.2"
set VcellSetup(13) "diff-to-fpoly          0.2"
set VcellSetup(14) "poly-enc-diff          0.25"
set VcellSetup(15) "metal1-width           0.28"
set VcellSetup(16) "metal1-space           0.28"
set VcellSetup(17) "well-enc-diff          0.4"
```



7.3.1.4. VcellConfig

This variable must be an array data. It defines device internal connection mode. In Vcell form, click 

button to look over these information. This variable can be ignored.

Varable definition format is as below:

set VcellConfig (*index***)** " *parameter value1 value2 … valuen*"

set VcellConfig (type_*index***) "radio", "option" or "check"**

set VcellConfig (icon_*index_value1***)** "*image1*"

set VcellConfig (icon_*index_value2***)** "*image2*"

. . . . . .

. . . . . .

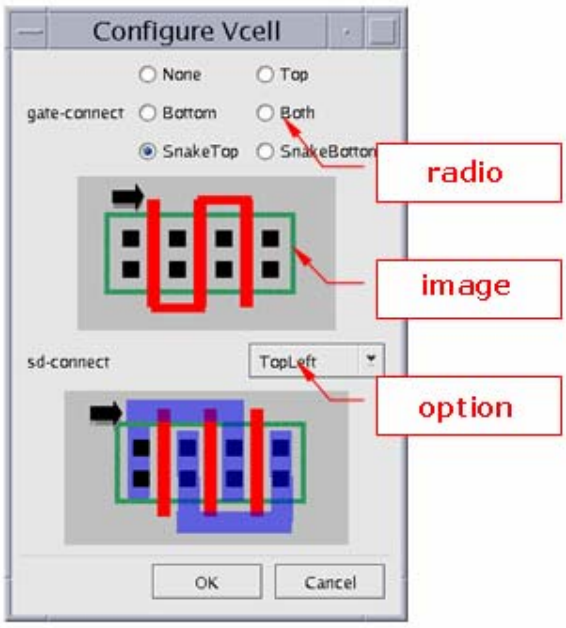
set VcellConfig (icon_*index_value n***)** "*imagen*"

Can be ignored

" index " as the subscript of array starts from ' 1 '.

If there are following definitions, Configure Vcell form shows them as below figure.

```
set VcellConfig(1)  "gate-connect       None Top Bottom Both SnakeTop
SnakeBottom"
set VcellConfig(2)  "sd-connect         None TopLeft BottomLeft HTopLeft
HBottomLeft"

set VcellConfig(type_1)                 "radio"
set VcellConfig(type_2)                 "option"

set VcellConfig(icon_1_None)            "mulparall.gif"
set VcellConfig(icon_1_Top)             "top.gif"
set VcellConfig(icon_1_Bottom)          "bottom.gif"
set VcellConfig(icon_1_Both)            "both.gif"
set VcellConfig(icon_1_SnakeTop)        "snaketop.gif"
set VcellConfig(icon_1_SnakeBottom)     "snakebottom.gif"

set VcellConfig(icon_2_None)            "mulparall.gif"
set VcellConfig(icon_2_TopLeft)         "topleft.gif"
set VcellConfig(icon_2_BottomLeft)      "bottomleft.gif"
set VcellConfig(icon_2_HTopLeft)        "hitopleft.gif"
set VcellConfig(icon_2_HBottomLeft)     "hibottomleft.gif"
```



7.3.1.5. VcellFuntion

This variable defines the name of implementation function name. This variable must appear in vcell template file.

Variable definition format is as below.
set VcellFunction *Function_ name*
There is no any parameter behind function name. Please refer to 7.3.2 Device Internal Structure Definition below.

7.3.2. Device Internal Structure Definition

This part defines how to create a device layout with parameters defined in function *Vcell_Name***SetParameter.** All implementation methods are included in function below.

proc Function_name { par1 par2 par3 … parN } {
……..
……
}

Function_name is defined in 7.3.1.5 VcellFuntion.

"par1 par2 … parN " are formal parameters of function. The real parameter values are passed by following order:
1. parameters in VcellParameter.
2. parameters in VcellSetup.
3. parameters in VcellConfig.

ZeniPDT create internal values through calculatation and executation command of Tcl and draw layout image with build-in creation object commands of Zeni. There create object commands contain Rectangle，Polygon，Path and Instance.

7.3.3. Build-in Creation Object Command

7.3.3.1. Create Rectangle

Format of this command:
rectangle "*xl yb xr yt*" [**-layer** *lay_name*] [**-purpose** *purpose*]
[**-pin** *name* **–io** *io_type* **–ad** *ad_type*]

xl yb xr yt：left bottom and right top coordinates of rectangle
-layer *lay_name*：layer name.
-**purpose** *purpose*：purpose name. "dwg" is defalult.
-**pin** *name*：pin name. Option "–pin" specifies Zeni regards the rectangle as pin.

-io *io_type*：pin type. *Io_type* must be one of ”in”，”out”，”inout” and”switch”.
-ad *ad_type*：access direction. *ad_type* must be one of “all”，”top”，”bottom”，”left” and ”right”.

For example：

rectangle “0.2 0.2 1.58 0.8” –layer oxide –purpose dwg –pin A –io in –ad all

7.3.3.2. Create Polygon

Format of this command:

polygon “*x1 y1 [x2 y2] [x3 y3]…*” [**-layer** *lay_name*] [**-purpose** *purpose*]
[**-pin** *name* **–io** *io_type* **–ad** *ad_type*]

x1 y1 [x2 y2] [x3 y3]：all of point coordinates of polygon.
-layer *lay_name*：layer name.
-purpose *purpose*：purpose name. ”dwg” is defalult.
-pin *name*：pin name. Option “–pin” specifies Zeni regards the rectangle as pin.
-io *io_type*：pin type. *Io_type* must be one of ”in”，”out”，”inout” and”switch”.
-ad *ad_type*：access direction. *ad_type* must be one of “all”，”top”，”bottom”，”left” and ”right”.

For example:

polygon “0.3 0.3 0.7 0.3 0.7 0.7 0.3 0.7” –layer Met1 –purpose dwg

7.3.3.3. Create Path

Format of this command:

path “*x0 y0 [x1 y1] [x2 y2]…*” [**-layer** lay_name] [**-purpose** *purpose*] [**-width** *wid_num*]

x0 y0 [x1 y1] [x2 y2]：middle line coordinates of path.
-layer *lay_name*：layer name.
-purpose *purpose*：purpose name. ”dwg” is defalult.
-width *wid_num*：path width.

For example：

path “0.890 0.085 0.890 0.895 “ –layer Poly –purpose dwg –width 0.2

7.3.3.4. Create Instance

Format of this command:

instance “*x y*” **–lib** *lib_name* **–cell** *cell_name* **–view** *view_name*

x y：reference point coorinate.
–lib *lib_name* **–cell** *cell_name* **–view** *view_name*：the source of the cell which is

called.

For example：

instance “0.01 0.02” –l tstlib –cell pmos –view layout

7.3.3.5. Create Temporary Cell ---dump

This command creates a temporary cell with definitions in front of dump. This temporary cell can be called by instance command (please refer to 7.3.3.4 Create Instance on page 226. Commands dump and instance must be used at the same time.

Format of this command:

dump -cell *cell_name*

instance “*x y*” **–cell** *cell_name*

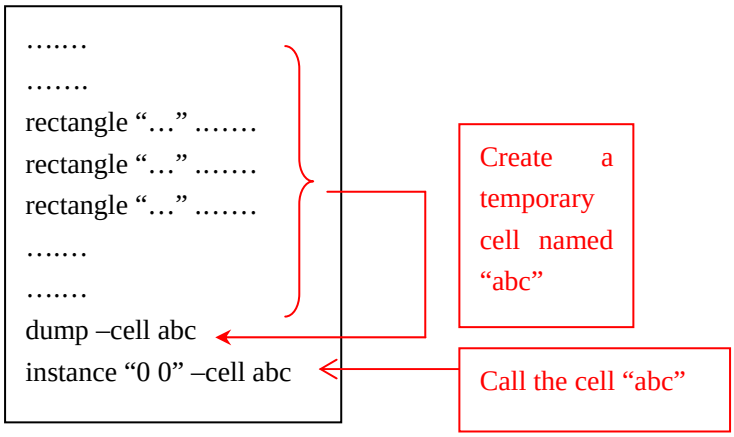
cell_name：temporary cell name.

x y：offset based on original coorinate.

Note：

Commands dump and instance must be used at the same time. If use dump command only, the temporary cell will not appear in device layout.

For example:



7.3.3.6. Print Message --- vcellPrint

vcellPrint print message in Design Manager. Message ends with “\n”：

Format of this command:

AvcellPrint “*msg*”

For example：

vcellPrint “PMOS will be of no gate. \n”