Cell Design Tutorial

Product Version 4.4.6
June 2000

Cadence Design Systems, Inc., 555 River Oaks Parkway, San Jose, CA 95134, USA

Contents

Preface

This tutorial introduces you to the Virtuoso® layout editor and the Assura™ interactive verification products.

Each tutorial chapter is divided into several sections. The beginning of each section lists the expectations of what you will learn. The step-by-step instructions help you become acquainted with basic operations of the software. After you complete the tutorial, you will be better prepared to use the other Cadence manuals and to attend Cadence training classes.

In Chapter 2, you create a layout design using many of the layout editor commands. You use the same design throughout most of the tutorial. The design that you start with at the beginning of a chapter is built on the preceding chapters. If you do not want to complete the chapters in sequence, however, you can use the prepared files in the tutorial database.

This tutorial assumes that you are familiar with your workstation, operating system, and window manager.

The preface discusses the following topics:

Related Documents

This tutorial is designed to be used with other manuals and training courses about the layout editor and related products. Which book you use depends on the kind of information you need:

If you want to	Look here
Ask questions about the layout editor and practice using it.	Virtuoso layout editor training courses
Find out how to perform different design tasks with the layout editor.	Virtuoso Layout Editor User Guide
Learn how to use automated layout editor commands.	Virtuoso Layout Accelerator User Guide

Look up specific parameterized cell commands.	*Virtuoso Parameterized Cell Reference*
Learn about creating ROD objects.	Virtuoso Relative Object Design User Guide
Install a sample parameterized cell library.	Sample Parameterized Cells Installation and Reference
Look up specific interactive verification commands.	*Assura Diva Verification Reference*

Typographic and Syntax Conventions

This manual refers to the mouse buttons by their positions on the mouse: “Click” means click the left mouse button. “Press middle” means press and hold the middle mouse button. You will be instructed where to move the mouse while holding down the button.

The instructions for starting a command from a menu will be abbreviated. When you are asked to “choose” commands from a menu, you must first click on the menu name, then on the command. For example, “Choose *File – Open*” means to click on *File* to display the menu, then click on *Open* to execute the command.

Instructions for starting a command from the keyboard will use “press,” followed by the key sequence. For example, “Press the `Return` key.”

Other syntax conventions that may be used in this document are described below.

`text`	Indicates text you must type exactly as it is presented.
`z_argument...`	Indicates text that you must replace with an appropriate argument. The prefix indicates the data type(s) the argument can accept, for example *`t_`* for text. The three dots indicate that you can repeat the argument. Substitute one or more names or values. Do not type the data type or underscore.
`variable`	Indicates text that you must replace with text appropriate to your system. An example is: `cd` *`your_install_dir`*`/tools/dfII/bin/layoutPlus`

1

Getting Started with the Cadence Software

In this chapter, you learn about the Cadence® software environment and the Virtuoso® layout editor as you do the following tasks:

When you finish this chapter, you will be able to

- Start and exit the Cadence software
- Display and browse the tutorial libraries
- Open a library and a layout window
- Use the Layer Selection Window to set layer visibility
- Measure shapes in a layout window
- Pan, zoom, and scroll in a layout window
- Go back to an area you viewed previously

- Access online Help
- Quit the layout editor

Prerequisites

Before you begin the tutorial, be sure your system administrator has installed the layout editor and (optionally) the Assura™ interactive verification products. In this tutorial you will run the following tools: Layout, Layout XL and Pcell. You must have these installed to complete all the tasks in the tutorial.

Here are some questions to ask your system administrator:

- Where is the tutorial directory?
- What command do I use to start the Cadence software?

  The command for starting the Cadence software varies depending on the software options your company purchased.

Copying the Tutorial Database

You need to copy the tutorial database, which is located under your Cadence installation directory, to your home directory. The following steps copy the tutorial database directory called `cell_design` under your home directory.

This tutorial does not use or cover the Team Design Manager (TDM). For information about TDM, refer to the *Team Design Manager User Guide*.

The steps in this section assume

- Your environment is already set up to run Cadence software
- You are logged in to your account
- You will copy the tutorial database directory into a non-TDM area

1. Be sure you do not already have a directory named `cell_design` in your home directory by typing

   ```
   cd ~
   ls cell_design
   ```

   If the response is `cell_design: No such file or directory`, continue to step 2.

If the response is a list of files, you have a `cell_design` directory. Rename the `cell_design` directory in your home directory by typing.

```
mv cell_design old_cell_design
```

2. Find the installation directory for Cadence software by typing

   ```
   instdir
   ```

   You see the installation directory; for example,

   ```
   /usr/cadence/tools/dfII
   ```

3. Change directory to *`your_install_dir`*`/samples/tutorials/le`.

   For example, if your installation directory is

   ```
   /usr/cadence/tools/dfII
   ```

   type

   ```
   cd /usr/cadence/tools/dfII/samples/tutorials/le
   ```

4. Copy the `cell_design` directory to your home directory by typing

   ```
   cp -r cell_design ~/
   ```

   The `cell_design` directory should contain these files and subdirectories:

   ```
   .cdsenv
   .cdsinit
   ROD
   cds.lib
   cellTechLib/
   display.drf
   master/
   mux2_strmin.template
   mux2_strmout.template
   pCells/
   skill
   tutorial/
   ```

   If the data was not copied successfully (you get an error message), type

   ```
   cd
   rm -rf cell_design
   ```

   and try copying the `cell_design` directory to your home directory again.

Starting the Cadence Software

The rest of this chapter will take you through some basic tasks using the Cadence software and the Virtuoso layout editor.

This section tells you how to

- Start the Cadence software, including changing to the tutorial working directory and finding the executable filename
- Use the Command Interpreter Window (CIW), which is the main control window for the Cadence software
- Use menus and forms

Setting the Working Directory

The working directory is where you start the Cadence software. Your Cadence initialization file, `.cdsinit`, contains commands and default settings for this tutorial. Any files or libraries you create during your editing session are stored in this directory, unless you specify otherwise.

To set the working directory,

➤ Change to the tutorial database directory by typing

```
cd cell_design
```

The `cell_design` directory is your working directory for this tutorial. All paths in this tutorial start in this directory.

Finding the Executable

To start the Cadence software, you need to know the executable. If you do not know the executable, you can search for it. If you already know the executable filename, you can skip this section.

The layout editor executables are `layout` (layout and interactive design rule checking) and `layoutPlus` (layout, complete verification, and other layout-related tools such as compaction). To do all the exercises in this tutorial, you need to have `layoutPlus`. If you have only `layout`, you will not be able to do some of the verification steps in this tutorial.

To find the executable,

➤ At the UNIX prompt, type

```
which layoutPlus
```

If `layoutPlus` exists and your search path is set correctly, you see a response similar to

your_install_dir/`tools/dfII/bin/layoutPlus`

`your_install_dir` is where the Cadence software is installed. You might want to make a note of its location for future reference.

If you see a response that looks like

```
no layoutPlus in . /usr/bin/X11
/usr/local /usr/ucb /bin /usr/bin /usr/hosts
```

either you do not have `layoutPlus` or your UNIX search path is not set correctly. Ask your system administrator for help.

Loading the Cadence Software

Once you are in the correct working directory (`cell_design`), you can start the Cadence software. The following step shows how to run the software as a background process so that your xterm window is still available to do other tasks while the Cadence software is running.

To load the Cadence software,

➤ Type

```
layoutPlus &
```

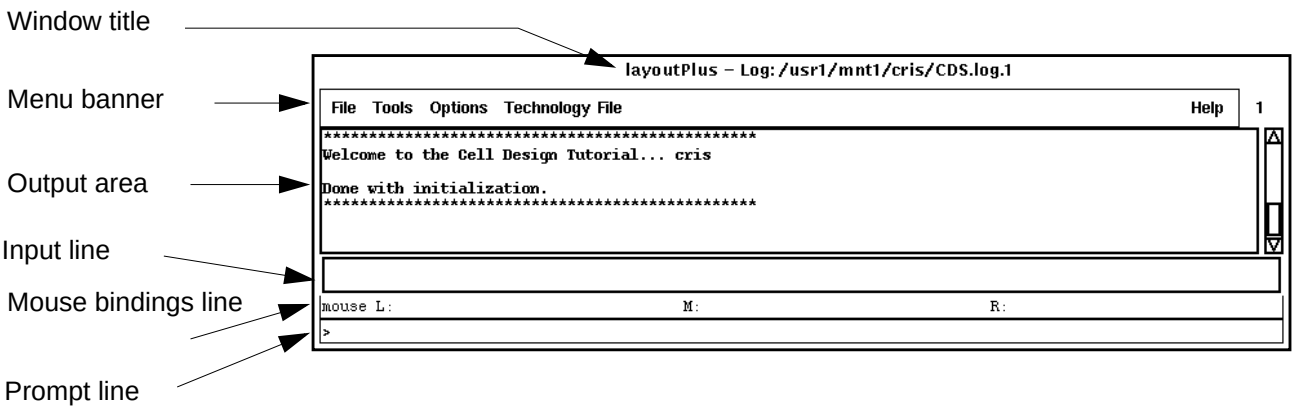
The software might take a few minutes to load. You see a series of messages in your xterm window about the process number and the Cadence products being started. The CIW appears and the following message scrolls into the window:

```
Done with initialization.
```

You can exit the Cadence software at any time, no matter where you are in your work. To exit the software, see “Exiting the Cadence Software” on page 27.

Using the CIW

The CIW is the control window for the Cadence software. The following figure shows the parts of the CIW.



Window title displays the Cadence executable name and the path to the log file that records your current editing session. The log file appears in your home directory.

Menu banner lets you display command menus to access all the Cadence design framework II tools.

Output area displays a running history of the commands you execute and their results. For example, it displays a status message when you open a library. The area enlarges when you enlarge the CIW vertically.

Input line is where you type in Cadence SKILL language expressions or type numeric values for commands instead of clicking on points.

Mouse bindings line displays the current mouse button settings. These settings change as you move the mouse in and out of windows and start and stop commands.

Prompt line reminds you of the next step during a command.

Using Menus and Forms

Most of the menus you use in the Cadence software are pull-down menus. Pull-down menus appear at the top of a window. You click a menu title to pull down the menu and see the commands listed on it. The pull-down menus are the primary place to find commands.

Many commands have forms you use to supply the command with additional information. Some commands have option forms that you do not always need. You can use the *User Preferences* command to set whether or not option forms always appear when commands start. In this tutorial, option forms always appear when you start commands.

To see how the pull-down menus work,

1. In the CIW, click *Options*.

   You see the commands on the *Options* menu. Notice the three dots (...) following the commands. These dots mean a form appears after you click the command.

2. Click anywhere outside the *Options* menu to close it without starting a command.

   If you accidentally choose a command, click *Cancel* in whatever form appears to cancel the command.

3. Click *Options* again.

4. Click *User Preferences*.

   The User Preferences form appears. This form contains settings that control how the Cadence software behaves. It was set for you in the tutorial `.cdsinit` startup file.

5. Click *Cancel* in the form.

   Canceling the command ensures that you do not change the user settings. If you click *Apply* or *OK*, the user settings change to whatever is in the form at that time.

Opening Designs

The Open File form lets you open designs in the libraries.

In this section, you learn

- How to open the Library Browser window
- About the tutorial libraries
- How to look at the contents of the `master` library

The `master` library contains the complete multiplexer design you will create in the following chapters of this tutorial.

Opening the Library Browser

To open the Library Browser,

1. In the CIW, choose *File – Open*.

   The Open File form appears.

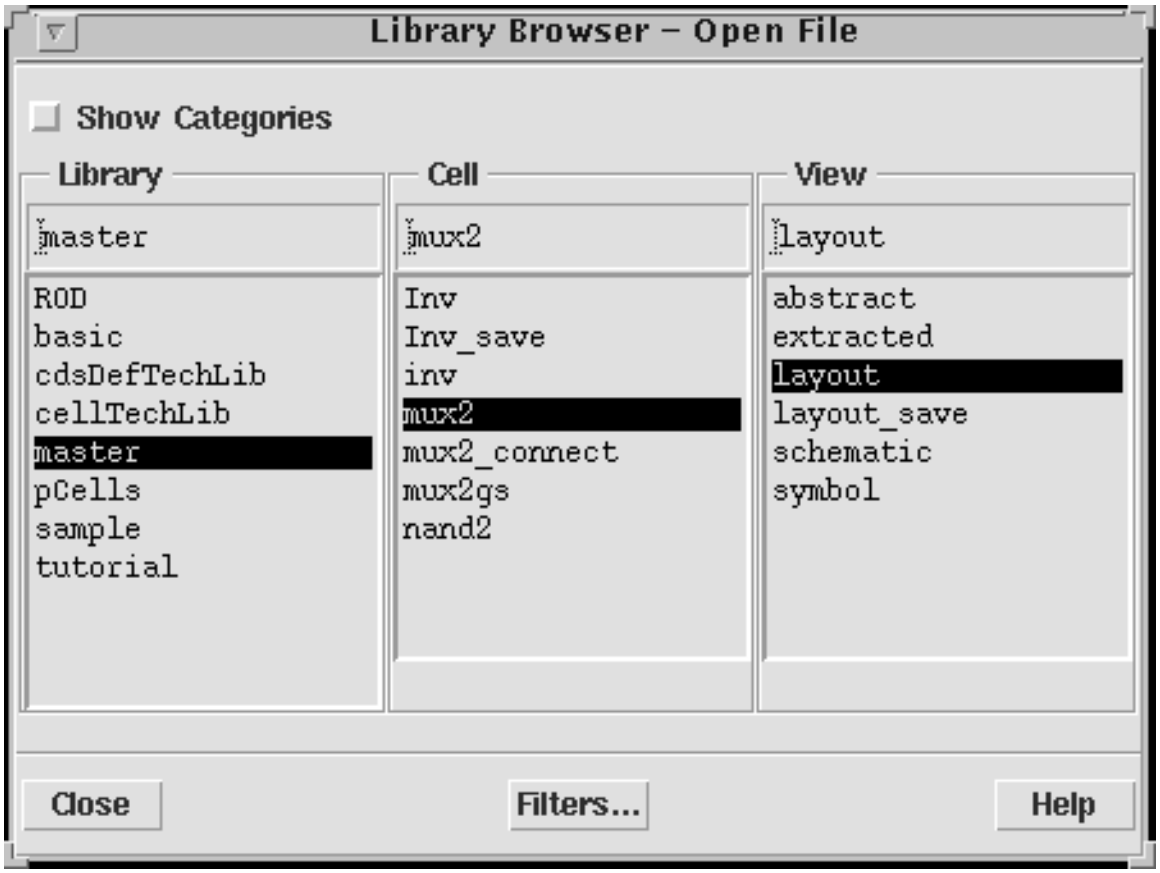
2. Click *Browse* to browse the libraries.

   The Library Browser – Open File form appears.

   The tutorial `cds.lib` file gives you access to all the tutorial libraries. The next section describes these libraries. Depending on the software purchased by your company, your form might show some additional libraries.

   Though you can browse and select libraries, cells, and views in the Library Browser window, you must open the selected cellview from the Open File form.

3. View the contents for the `master` library by clicking *master* in the *Library* column.



About the Tutorial Libraries

The libraries contain the following information:

basic	Reference library supplied with the software. Contains basic symbols, including ground, power, and input and output pins.
cdsDefTechLib	The default technology library supplied with the software. Contains the default technology file.
cellTechLib	Technology library for this tutorial. Contains the technology file, symbolic devices, and rules used by all the other tutorial libraries.
master	Design library supplied with this tutorial. Contains the completed designs for the tutorial.
pCells	Reference library supplied with this tutorial. Contains a collection of Parameterized Cells (pcells).
ROD	Reference library supplied with this tutorial. Contains Relative Object Design (ROD) instances.
sample	Reference library supplied with the software. Contains a sample collection of gates and cells.
tutorial	Design library supplied with this tutorial. You use it to store your designs as you follow the steps in the rest of this tutorial.

Browsing the Master Library

This section lets you explore the tutorial design by displaying the contents of the `master` library. The designs are called *cells*. A cell represents a particular function of a larger design. For example, one of the cells in the `master` library represents an inverter design (*Inv*), one represents a nand2 design (*nand2*), and another represents the multiplexer layout (*mux2*) that contains the inverter and the nand2.

Each cell can have multiple views called *cellviews*. A cellview is a specific representation of a cell; for example, a layout or a schematic.

➤ In the *Cell* column, click *mux2* to display the cellviews for the *mux2* design.

The view names appear.

The *mux2* design has six cellviews.The cellviews and their uses are described here:

- The *abstract* cellview contains an abstract representation of the layout for use by Cadence placement and routing software.

- The *extracted* cellview contains layout with connectivity for use by verification programs.
- The *layout* cellview contains a traditional layout using polygons and other shapes.
- The *layout_save* cellview is a backup of the *layout* cellview.
- The *schematic* cellview contains the logical design for the multiplexer.
- The *symbol* cellview contains a symbol representation of the schematic.

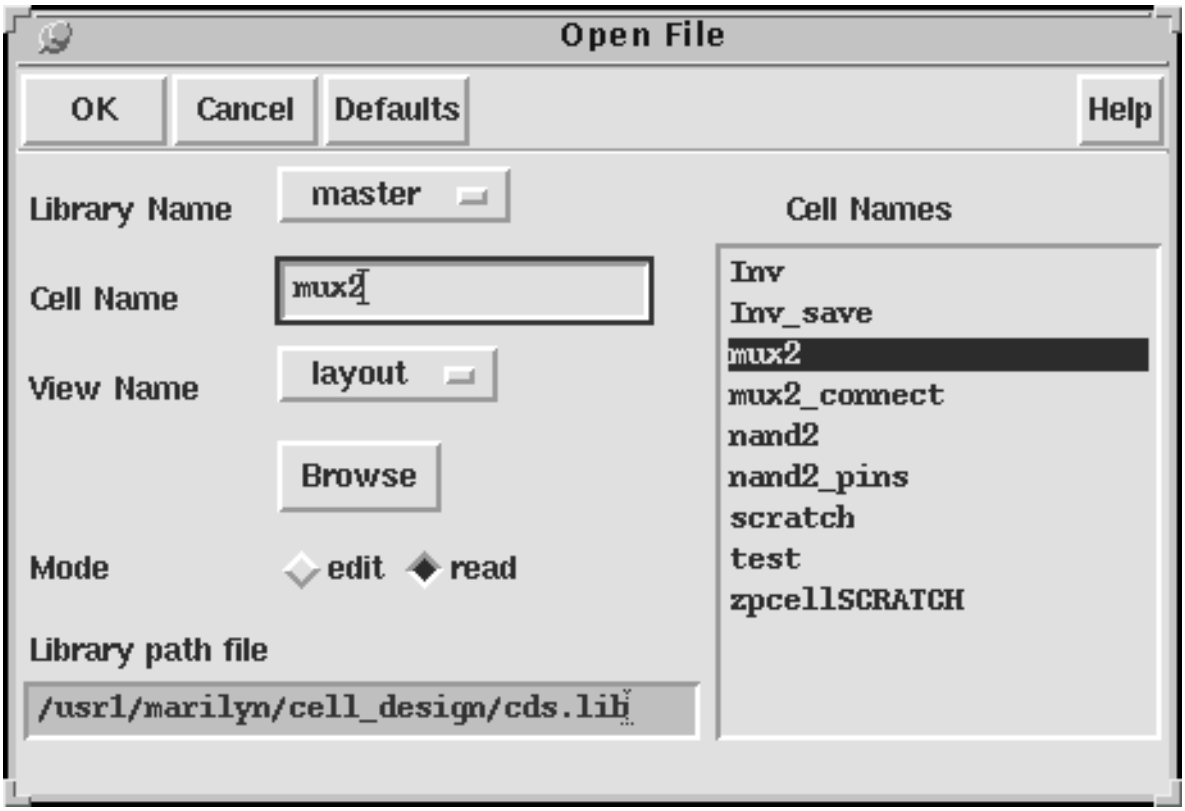
Displaying the mux2 Layout

In this section, you learn to open the layout for the multiplexer design *(mux2)* in read-only mode. This prevents you from making any changes in the `master` library.

1. In the *View* column of the Library Browser window, click *layout*.
2. In the Library Browser window, click *Close*.

   The *Library Name*, *Cell Name*, and *View Name* fields in the Open File form display the information you set in the Library Browser window.

3. In the Open File form, click the *read* radio button to display this cellview in read-only mode.



4. Click *OK*.

   The layout window opens and the *multiplexer* layout appears. The window has several components: icon menu, menu banner, and status banner.

   The icon menu appears (by default) on the side of the window, listing the icons for frequently used commands. The menu banner contains menus that let you execute the layout editor commands. The status banner displays mode and coordinate information. Each component is explained in detail in later sections of this tutorial.

   Another window, called the Layer Selection Window (LSW), appears.

5. To see the commands that you can use from the icon menu, slide the mouse over the icons.

   As you move over an icon, the command name appears.

Note: When you open a cellview in edit mode, more icons appear in the menu.

Setting Layer Visibility

The Layer Selection Window (LSW) lets you

- Choose the layer on which you create objects (called the entry layer)
- Set which layers are selectable
- Set layer visibility

There are several ways to change the LSW to make layers selectable, visible, and valid. In this section, you learn to use the LSW to change the visible layers in the *mux2* window.

To see how layer visibility works,

1. Make the *metal1 dg* layer not visible, by moving the cursor over the *metal1 dg* layer in the LSW and clicking middle.

   The middle mouse button toggles layer visibility. It also automatically sets invisible layers to be unselectable.

   The text layer color disappears to show the layer is invisible. The layer name turns gray to show the layer is not selectable.

   Note: If you click left the *metal1 dg* layer and then click middle, an error message appears saying you cannot set the entry layer to be invisible. To correct this, click a different layer in the LSW, and then click middle again on the *metal1 dg* layer.

2. In the *mux2* window, choose *Window – Redraw*.

   The *mux2* layout is displayed with all layers visible except the *metal1 dg* layer.

3. In the LSW, click the *AV* (All Visible) button.

   The colored squares showing the layer color reappear, and the shading on the layer name disappears.

4. In the *mux2* window, choose *Window – Redraw*.

   The *mux2* layout is redisplayed with all layers visible.

   You must redraw the window to see the effect of LSW changes, so you can make several changes before you take the time to redraw a complex design.

Measuring Distances

Most designers must create objects according to precise measurements. The layout editor provides functionality to help designers be as precise as possible. In this section, you learn to

- Enlarge (zoom in) a portion of the cellview
- Create a ruler to measure objects in the cellview

Zooming In on the Multiplexer Design

You can zoom in on the layout to see details. This makes it easier to measure small shapes. You zoom in by creating a box around the area you want to enlarge.

Note: Zooming in does not enlarge objects in the database. It enlarges only what you see.

To see how zoom works,

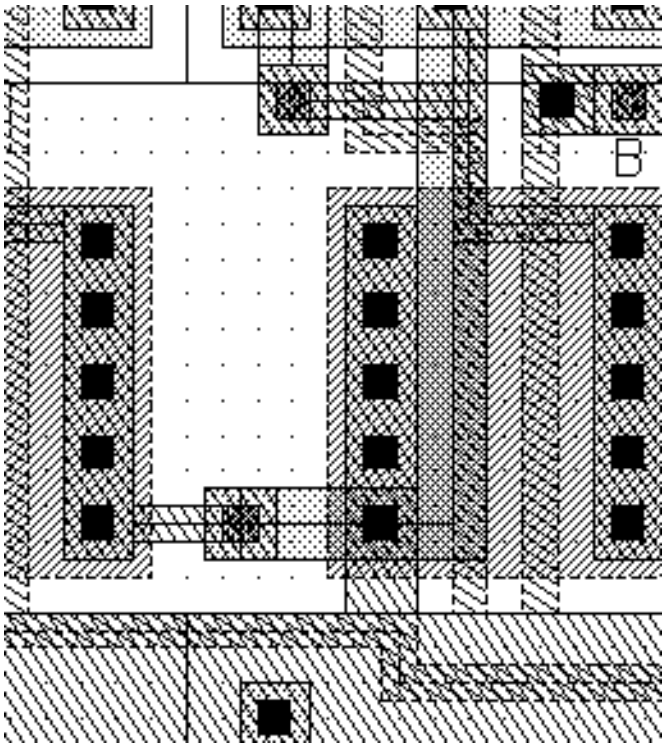
1. In the *mux2* window, choose *Window – Zoom – In*.

   A prompt in the CIW tells you to create a box around the area you want to enlarge.

   To see the details of one of the n-type transistors in the *mux2* layout, you can create a box around the area.

2. Click one corner of the box, then click the opposite corner of the box, as shown in the figure.

The area enclosed by the box is enlarged to fill the window.



Measuring Distances with the Ruler

Now you use a ruler to measure the gate length of the transistor.

1. In the *mux2* window, choose *Window – Create Ruler*.

   The Create Ruler form appears.

2. Click one edge of the *poly1* layer.

   A ruler appears and grows as you move the cursor. The current length of the ruler appears at the end of the ruler.

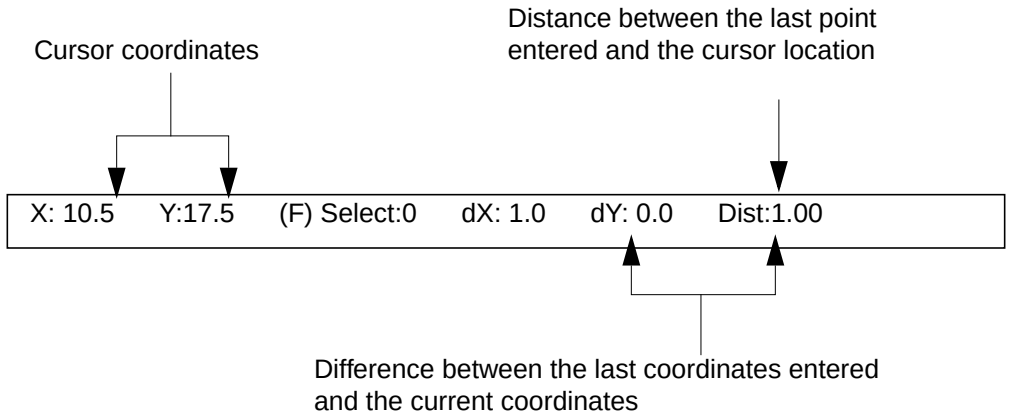
   The ruler measures objects in user units. Your system administrator defines the units in a file called the technology file. User units for this tutorial database are set to microns, which are also the default setting for the Cadence software.

3. Look at the status banner to see the coordinate display.

   The status banner always shows the following information whenever you create any object:

- *X* and *Y* show the current cursor coordinates.
- *Dist* (distance) shows the distance between the last point you entered and the current location of the cursor.
- *dX* (delta in the X direction) and *dY* (delta in the Y direction) show the difference between the last point you entered and the location of the cursor.

You can also use the coordinate display in the status banner to measure the distance between points you enter.



4. Click the opposite edge of the *poly1* layer to complete the ruler.
5. In the Create Ruler form, click *Cancel* to stop the command.

   All layout editor *Create* commands, such as *Create Ruler*, automatically repeat until you cancel them or start a different Create command.
6. Choose *Window – Clear All Rulers* to remove the ruler you drew.

   The ruler disappears. These rulers are not part of the cellview and are not saved. The *Clear Rulers* command removes all rulers in the cellview.

Changing to a Previous Zoom or Pan Image

The part of a design you see displayed in the window is called an image. While you go through this tutorial, you might need to alter the image to see the design more clearly. In the next sections, you learn how to alter the image, including how to

- Fit the entire design into a window
- Return to the previous image

- Reduce the amount of detail you see (zoom out)
- Use the window scroll bars
- Use the mouse to zoom in and out

You also learn to use a bindkey, which is a shortcut to start commands. This tutorial periodically points out the bindkeys to start commands.

Fitting a Design in a Window

To see the entire design,

➤ Choose *Window – Fit All*.

The entire *mux2* design is displayed in the window.

You can use *Fit All* any time you want to return to the full view of a design.

Returning to the Previous Image

The layout editor automatically keeps up to three images in memory. You can easily return to a previous image.

To see the previous view,

➤ Choose *Window – Utilities – Previous View*.

You see the area you zoomed in on earlier.

Using Window Scroll Bars

You can also use the window scroll bars to change the image.

1. Place the cursor on the scroll bar at the bottom of the window.
2. Click and hold, then move the scroll bar to the right.

   The contents of the window scroll to the bottom right section of the design.
3. Place the cursor on the scroll bar at the right of the window.
4. Click and hold, then move the scroll bar up.

   The contents of the window scroll to the top right section of the design.

Zooming Out

To zoom out,

➤ Choose *Window – Zoom – Out by 2*.

The window now shows more of the design at a smaller magnification.

Using the Fit All Bindkey

You can start most of the layout editor menu commands by using one or two keys on your keyboard, called bindkeys. Using the bindkey is the quickest way to start a command. You can find out the bindkey for a layout editor command by looking to the right of the command on the menu.

1. Click the *Window* menu and look at the characters to the right of the command names.

   These characters are the bindkeys for the commands. A lowercase letter means you press just that key; an uppercase letter means you hold down the `Shift` key and then press the key. A caret (^) means you press the `Control` key while pressing the letter key.

2. Close the *Window* menu by clicking in an empty part of the design.

3. Press the `f` key to execute the *Fit All* command.

   The `f` key is bound to the *Fit All* command. You see the entire *mux2* design again.

   Remember to look at the bindkey equivalent when you use a pull-down menu so you learn bindkeys for future use.

Zooming In and Out with the Mouse

Instead of choosing the *Zoom – In* or *Zoom – Out by 2* command, you can use the right mouse button to zoom.

To zoom in,

1. Click and hold right on one corner of the area you want to enlarge.

2. While still pressing right, move the cursor diagonally.

   A box appears and stretches to follow the cursor. Pressing a mouse button and dragging the cursor to create a box is called “drawthrough.”

3. Release the mouse button.

The area enclosed by the box you drew now fills the screen.

Next, create a box in which you want to fit the current window contents.

4. With the cursor inside the *mux2* window, press and hold the `Shift` key.
5. Still pressing the `Shift` key, press and hold right and create a box in which you want to fit the current window contents.
6. Release the mouse button.

   The window zooms out. You see more of the design, but less detail.

Using Help

The *Help* button on forms and in windows displays information about the layout editor.

- To display a page of information about the command you are using, click *Help* in the command form or options form.
- To display the *Virtuoso Layout Editor User Guide Table of Contents* from which you can navigate to the information you want to see, click *Help* in a layout design window and choose *Contents*.

Another way to display Help is to use the `F1` key at the top of your keyboard. `F1` is the bindkey for Help.

- To display a page of information about a command, press `F1` while your cursor is in the design window and the command is running.
- To see the *Virtuoso Layout Editor User Guide Table of Contents*, press `F1` while your cursor is in a layout design window.

Using the Help Button

In this section, you learn to use the Help system. You can use Help as you perform any of the steps in this tutorial.

1. Click *Help* in the *mux2* layout window.

   The *Help* menu appears.
2. Choose *Contents*.

   In a moment, the *Virtuoso Layout Editor User Guide Table of Contents* appears.

3. Choose *Help – Contents* (at the top of the Help window) to read details about using the OpenBook® online documentation library's help.

   A second window appears. You can click the topics in this window to find out more about the Help system.

4. Click *Close* to close the FrameViewer – Help window.

5. In the *Virtuoso Layout Editor User Guide Table of Contents*, click any blue text to read about that topic.

6. Click *Close* to close the Help window.

Exiting the Cadence Software

You can exit the Cadence software at any time; it does not matter where you are in your work. If you have unsaved work, the software prompts you to save your work before you exit.

1. In the CIW, choose *File – Exit* to exit the software.

   An Exit dialog box appears.

2. Click *Yes* in the dialog box to confirm that you want to exit.

   The Cadence software stops this session and closes the CIW.

   Note: If you leave any Cadence tools, windows, or forms open at the time you exit, the software closes them also.

3. In the xterm window, press `Return`.

   You see the exit message

```
[1]     Done                    layoutPlus
```

Summary

In this chapter, you learned the fundamentals of using Cadence software and of viewing designs. Specifically, you

- Logged in and started the Cadence software
  - Learned about the CIW
- Browsed libraries with the Library Browser window
  - Displayed the libraries, cells, and views

- Opened a design in read-only mode

- Changed the image for the design
  - Fit the design to the window
  - Returned to the previous image
  - Zoomed in and out
  - Scrolled
- Learned to start commands different ways
  - Used pull-down menus
  - Used bindkeys
- Learned how to display Help pages
- Learned how to exit the Cadence software

2

Creating the Inverter Layout

This chapter introduces you to the Virtuoso® layout editor as you perform the following tasks:

When you finish this chapter, you will be able to

- Create new cellviews
- Create instances of existing cellviews
- Create rectangles
- Use the Layer Selection Window to set entry layers
- Create paths
- Create polygons
- Copy and mirror objects
- Run the design rule checker
- Save your design

If You Have Not Completed the Previous Chapter

You can follow the steps in this chapter without completing the previous chapter. However, you might want to read the previous chapter summary before starting this chapter, so you know what was covered.

It is possible to run out of resources, such as memory, if you run multiple layout editors. Before you start the software, check whether the software is already running.

To check whether the layout editor is already running,

1. Type the following in an xterm window:

   ```
   ps auxw | grep layout
   ```

2. If the layout editor is listed in the standard output of the `ps` command, use *File – Exit* in the Command Interpreter Window (CIW) to exit the software.

3. Type the following in an xterm window to start the layout editor:

   ```
   cd ~/cell_design
   layoutPlus &
   ```

Starting a New Layout Design

In this section, you learn to

- Create and open a new cellview
- Set your image and drawing grid

As introduced in the previous chapter, a design cell is stored in a library. The cell can have several views, such as schematic or layout, that are different representations of the design. Before you can open a new design, you must have a library and it must contain definitions of the types of views you use. For this tutorial, the library and views have already been created for you.

Note: You will be instructed to move the cursor to specific X and Y coordinates as you build your design. These coordinates may differ slightly from the coordinates on your system. This should not be a problem in building the design.

Creating a New Cellview

1. In the CIW, choose *File – New – Cellview* to create a new file.

   The Create New File form appears.

2. Set *Library Name* to *tutorial*.

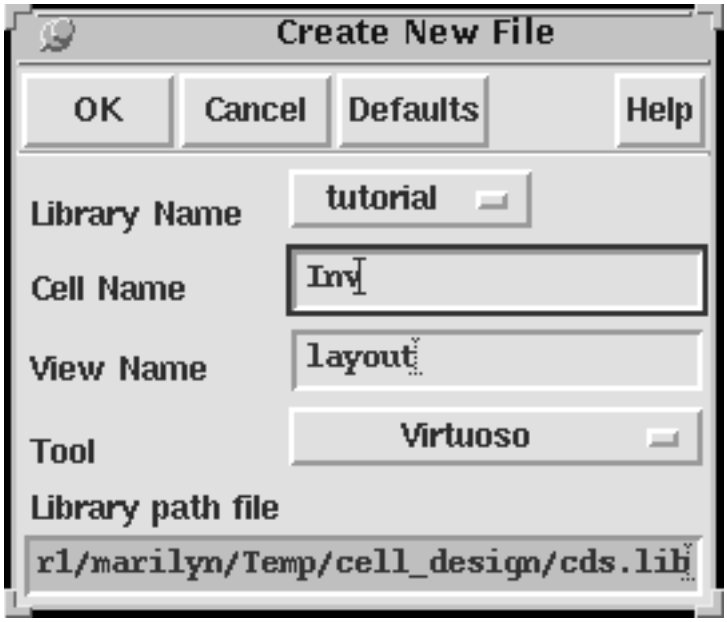
3. Type `Inv` in the *Cell Name* field.

4. Press the `Tab` key and type `layout` in the *View Name* field.

5. Press the `Tab` key.

   Because the view name `layout` is a recognized name for the layout editor, the *Tool* field changes to *Virtuoso*.

   The completed form should look like this:



6. Click *OK*, to create the new cellview and close the form.

   After a few moments, the *Inv layout* cellview appears in a layout window. The window is empty, and you can see the axes and the grid points.

Displaying the Grids

There are two grid displays: the minor (small) grid and the major (large) grid. Usually, you enter points using the minor grid. The tutorial library is set to show minor grid points at every micron and major grid points at every 5 microns. (For details about changing the grid point spacing with the *Display* command, refer to the *Virtuoso Layout Editor User Guide*.)

The minor grid points are white. The major grid points are green. You might need to look closely to see the difference in color. Notice that a major grid point (green) appears in place of every fifth minor grid point (white).

To see the green, major grid points,

1. Press `Shift-z` in the cellview to zoom out by 2.

   The green, major grid points become visible.

You can repeat the last command by clicking right.

The right mouse button information in the CIW reads

```
R: hiZoomRelativeScale(h ...
```

This is the Cadence® SKILL language command for *Zoom – Out by 2*. (The three dots mean you cannot see the rest of the line.)

Note: Once started, most layout editor create and edit commands repeat automatically; you do not have to click right.

2. Click right to repeat *Zoom – Out by 2*.

   The spacing between grid points becomes larger instead of smaller.

   The display shows only the major grid. This is because you are zoomed out too far to see the minor grid.

3. Choose *Window – Zoom – To Grid*.

   Zoom – To Grid zooms out to the smallest magnification at which the minor grid points are visible. Again, you see all the grid points.

4. Choose *Window – Zoom – In by 2* or press `Control-z` to set the grid so it is easy to place instances.

Using Pan to View the Positive Quadrant

Most designs are created in the positive (upper right) quadrant. You can pan the window so you can view just the positive quadrant.

To view the positive quadrant,

1. Choose *Window – Pan*.

   The prompt in the layout window and CIW reads

   ```
   Point at the center of the desired display:
   ```

2. Click in the upper right corner of the window.

   The image changes so the axes are in the lower left corner.

Creating Instances for N- and P-Transistors

Now you are ready to create objects in your design window. In this section, you learn to

- Place a copy of another cell into this cellview
- Use a layout editor options form
- Stop a layout editor command that automatically repeats

Placing a copy of a cell is commonly called placing an instance of that cell.

In this section, you place instances of two parameterized cells (pcells). A parameterized cell is designed to let you change some of its features whenever you place an instance of the cell. You will learn how to create pcells later in this tutorial.

The pcells you place in this section are designed to change dimensions based on the length, width, and number of gates. These changeable dimensions are the parameters of each cell. You enter parameter values when you place the pcell instances.

Creating the N-Transistor Instance

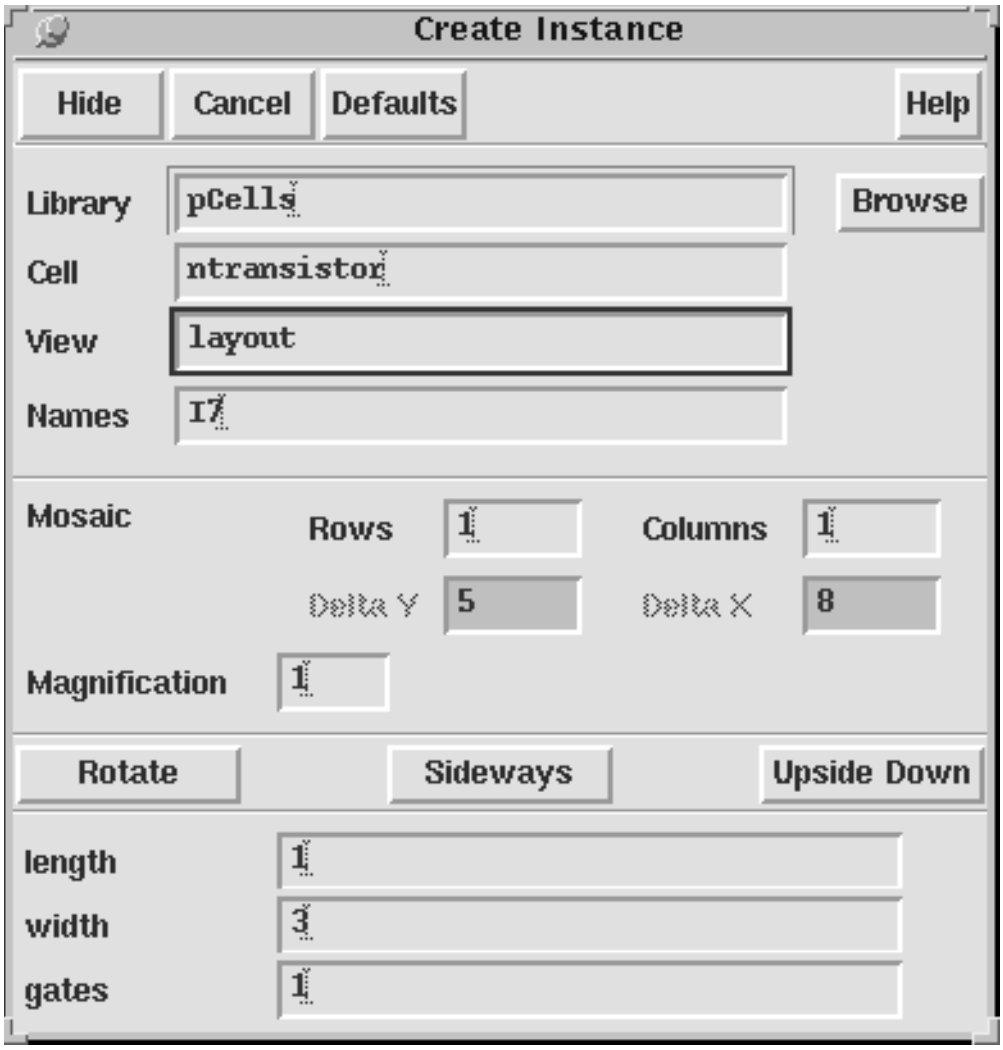
To create the n-transistor,

1. Choose *Create – Instance*.

   The Create Instance form appears.

   Note: This form is called a layout editor options form, which means it appears for the duration of the *Create Instance* command. Look at the buttons on the form; instead of *OK*, there is a *Hide* button. The *Hide* button lets you hide the form while you continue to use *Create Instance*.

2. Type the library and cell names:



You can double-click in a text field to highlight and type over a word. Press the `Tab` key to move to the next form field. The *View* and *Names* fields are automatically assigned.

3. Press the `Tab` key to display the parameter fields.

The parameter names and values appear at the bottom of the form. You are going to change these parameters for this exercise.

4. Move the cursor into the cellview window.

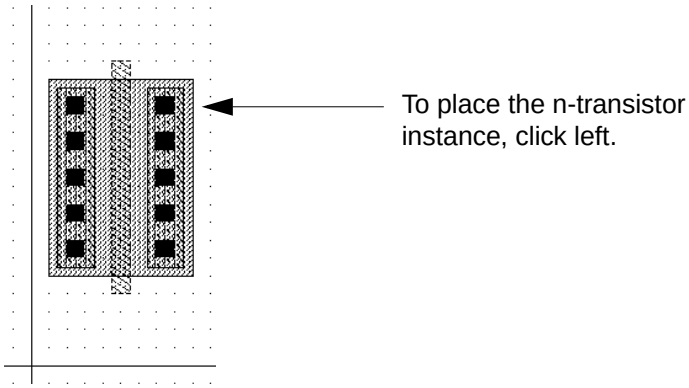
Outlines of the shapes in the n-transistor follow the cursor. The outlines help you align the cell as you place it.

5. In the Create Instance form, change the *width* parameter to `11`.

6. Move the cursor back into the cellview window.
7. Move the cursor to X = 5, Y = 6.5, and click left to place the instance.

   Remember to look in the window banner for the X and Y coordinates.

   After you place the instance, the n-transistor appears with the layer colors filled in.



8. Move your cursor to the right.

   The outline of another identical n-transistor follows the cursor because the *Create Instance* command is a repeating command.

Stopping Repeating Commands

The Create Instance form is still open, ready to place another instance. By default, commands on the *Create* and *Edit* menus always repeat.

To stop the *Create Instance* command,

➤ Press the Escape key.

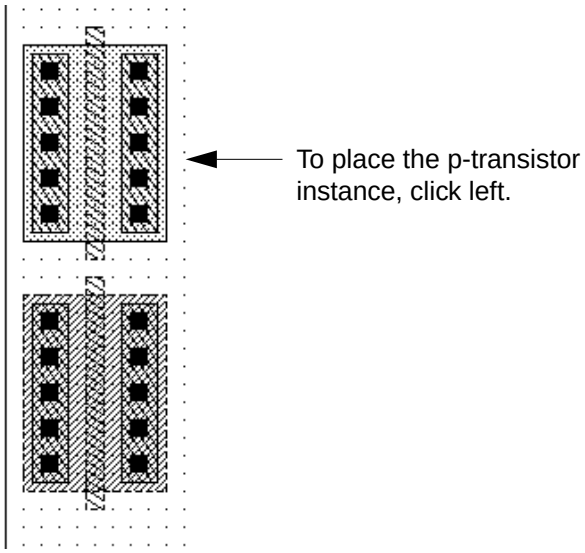
You can stop repeating commands whenever you want to ensure that you do not change your design by mistake.

Note: If a repeating command has an options form, you can also stop the command by clicking *Cancel* in the form. You already practiced this with the *Create Ruler* command in the previous chapter.

Creating the P-Transistor Instance

Next, you place an instance for the p-transistor.

1. Choose *Create – Instance* or press the `i` key.
2. In the Create Instance form, change the *Cell* field to `ptransistor`.
3. Press the `Tab` key to see the p-transistor parameters you can change.
4. Change the *width* parameter to `11`.
5. Click X = 5, Y = 20.5 to place the instance.



Your design now contains a p- and an n-transistor.

6. Press the `Escape` key to stop the *Create Instance* command.
7. Press the `f` key to see both transistors.

Connecting the Inputs and Outputs

Now that you have placed the n- and p-transistors, you need to create connections to tie them together and create an inverter. In this section, you learn to

- Choose the correct layer for creating the connection
- Create a rectangle, a path, and a polygon
- Correct an editing mistake
- Practice selecting objects
- Copy shapes

- Use infix mode to reduce the number of mouse clicks needed to create an object

Selecting Layers in the LSW

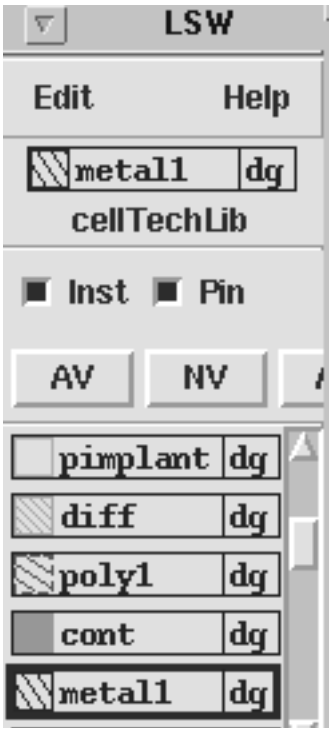
Before you create the connections to build the inverter, you need to select the correct entry layer.

Every layer in your library is assigned a purpose, such as *net* or *drawing*. Most layouts use layers with purposes of *drawing*, so by default the Layer Selection Window (LSW) shows all the layers that are defined as *drawing* in your library technology file. The abbreviation *dg* after each layer name means *drawing*.

➤ Click the *metal1 dg* layer in the LSW.

Note: Be sure to use the left mouse button to select the entry layer. If you click a different mouse button on a layer in the LSW, you will change the selectability or visibility of the layer.

The *metal1* layer is outlined in bold and appears at the top of the LSW. This tells you *metal1 dg* is the current entry layer. The layout editor prompts refer to this as the entry layer.



Connecting the Output

To connect the output of the two transistors,

1. Choose *Create – Rectangle*.

   The Create Rectangle form appears.

   The prompt in the layout window and CIW reads

   ```
   Point at the first corner of the rectangle:
   ```

2. Click X = 7.5, Y = 15.5 to create the first corner of the rectangle.

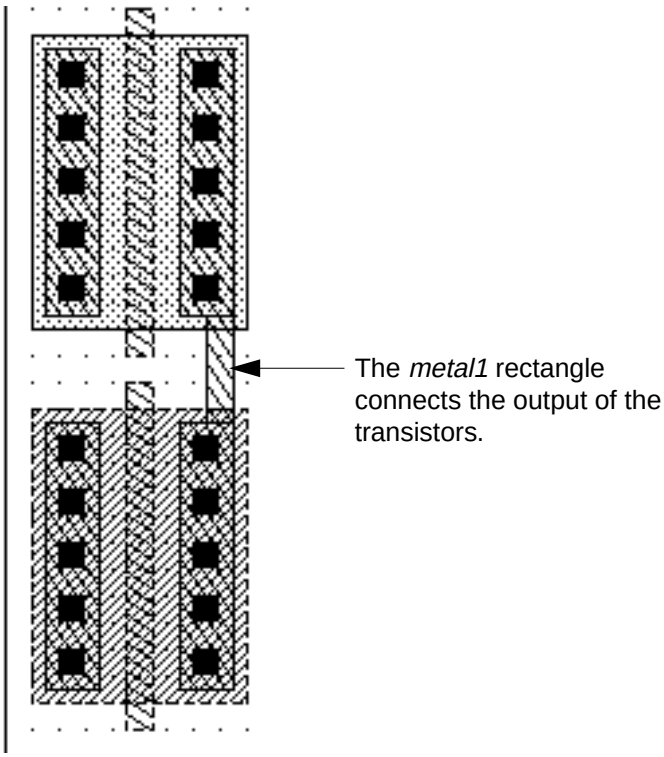
   The prompt in the layout window and CIW reads

   ```
   Point at the opposite corner of the rectangle:
   ```

   A rectangle appears and stretches as you move the cursor.

3. Click X = 8.5, Y = 19.5 to create the opposite corner of the rectangle.

You have completed the rectangle that connects the output of the two transistors.



The layout window and CIW still display a prompt because repeat mode is on.

Nesting a Command

Nesting means to run one command while another command is still running. For example, if you make a mistake while using the *Create Rectangle* command, you can use *Edit – Undo* to undo the previous command. This does not stop the *Create Rectangle* command but does nest the *Undo* command.

Note: You cannot nest repeating commands. For example, you cannot nest the *Copy* command while using the *Create Rectangle* command.

To nest a command,

1. Choose *Edit – Undo*.

   The rectangle disappears.

2. Choose *Edit – Redo.*

   The rectangle reappears.

You can undo up to 10 previous commands. Use *Redo* if you used *Undo* by mistake.

Connecting the Input

You are ready to connect the inputs. Use the *Create Path* command to connect the inputs for the transistors on the *poly1* layer.

Paths are shapes defined by a centerline and a width. You set the width in the Create Path form and create the path centerline.

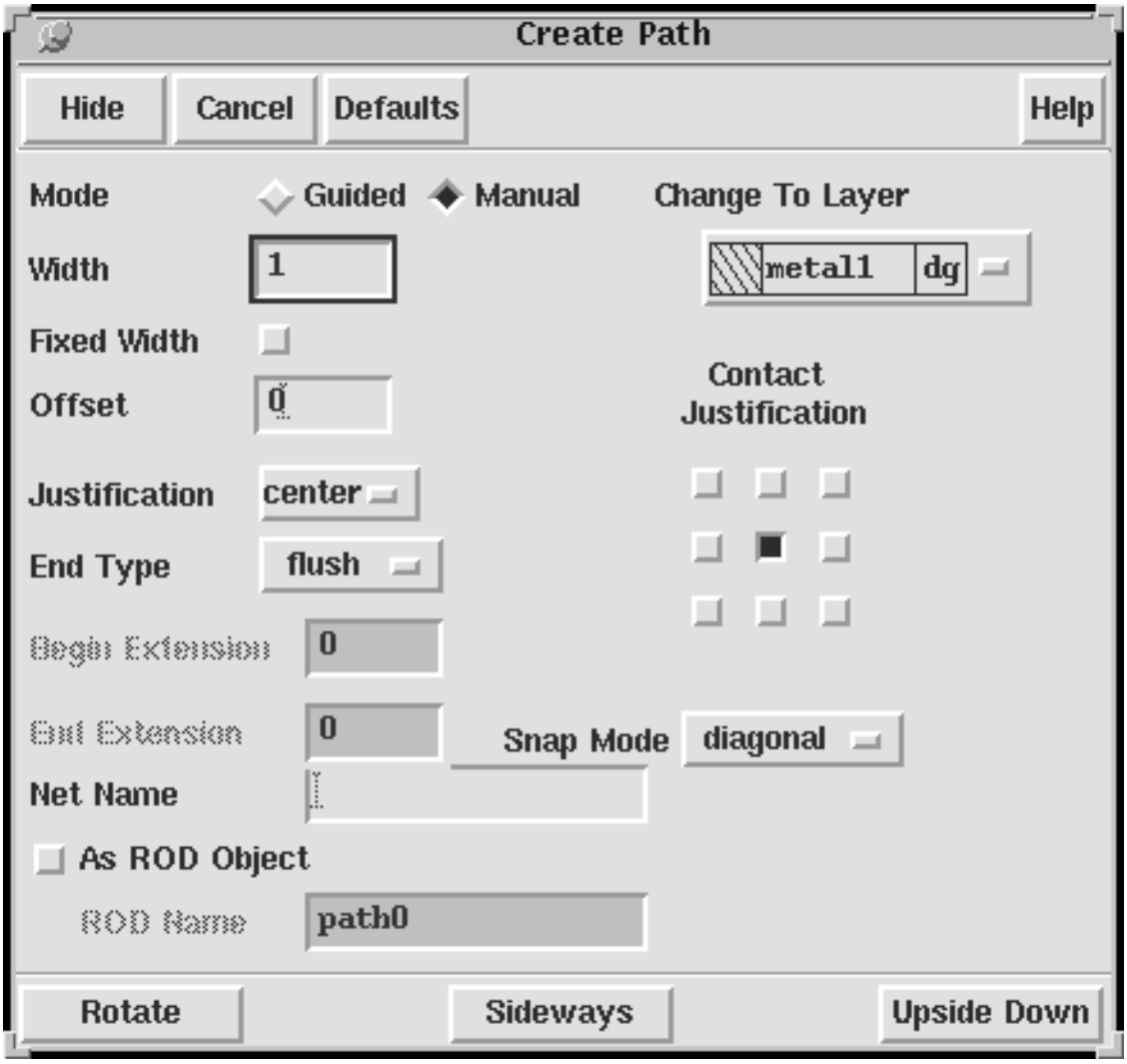
1. In the LSW, click the *poly1 dg* layer.

   The *poly1* layer appears at the top of the LSW.

   The system still prompts you to create rectangles because repeat mode is on. You do not need to manually stop the *Create Rectangle* command. It stops automatically when you choose any other *Create* or *Edit* command.

2. Choose *Create – Path*.

The Create Path form appears. The width is set to 1 micron, which is the minimum width for the *poly1* layer.



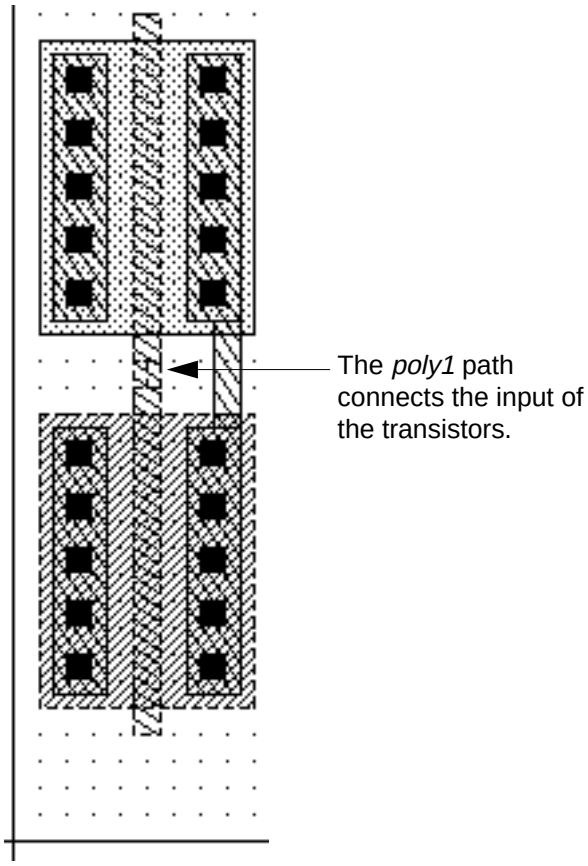
The prompt in the layout window and CIW reads

```
Point at the first point of the path:
```

3. Click X = 5.0, Y = 17.0.

4. Double-click X = 5.0, Y = 18.0.

The path appears. Double-clicking tells the system where to end the path.



Adding the Power Connection

Use the *Create Polygon* command to create the power connection.

Polygons are shapes defined by any number of points. Polygons must be closed; that is, the first point and the last point must be the same. The layout editor can close the polygon for you automatically.

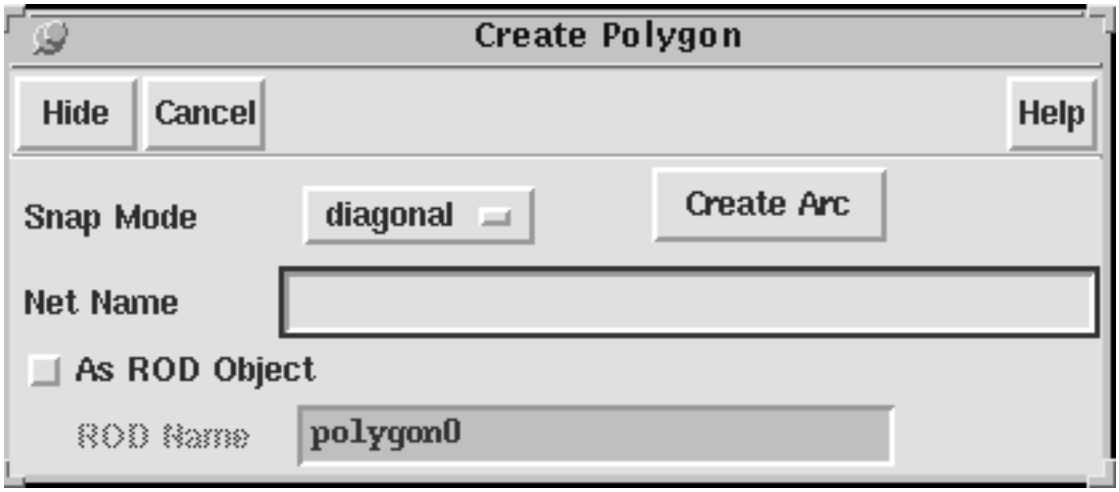
To add the power connection,

1. In the LSW, click the *metal1 dg* layer.

   The *metal1* layer appears at the top of the LSW.

2. Choose *Create – Polygon*.

The Create Polygon form appears.



The prompt in the layout window and CIW reads

```
Point at the first point of the polygon:
```

3. In the Create Polygon form, set *Snap Mode* to *L90XFirst*.

   The snap mode controls the way segments snap to the drawing grid as you create the polygon. *L90* creates two segments at right angles to each other between every pair of points you enter. *XFirst* means the first segment is parallel to the X axis.

4. Press the Tab key to start the *Pan* command to view the area above the p-transistor.

5. Click X = 5.0, Y = 28.0 to complete the pan, and to display the area above the p-transistor.

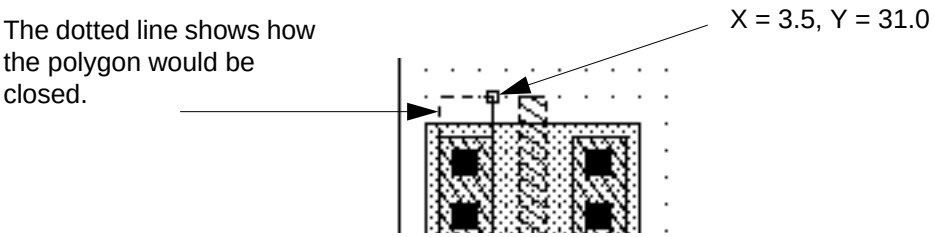
6. Click X = 1.5, Y = 29.5 to create the first point of the polygon.

7. Click X = 3.5, Y = 31.0.

   The layout editor creates

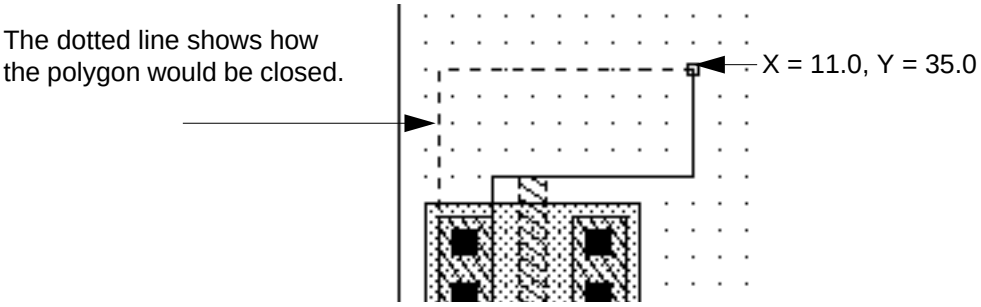
   - Two solid lines at right angles to each other between the points you entered

- Two dashed lines at right angles to each other attached to the two points you entered. The dashed lines show how the layout editor would close the polygon if you click twice on the second point you entered.



You do not want to complete the polygon at these points.

8. Click X = 11.0, Y = 35.0 to complete the polygon.



Undoing Points

If you make a mistake while creating an object, you do not have to start over. You can back up any number of points by pressing the Backspace key.

The last point you entered is not correct. The *metal1* shape should be one micron outside the boundary of the p-transistor. Because you are using L90 snap mode, the system entered two points for you, so you must back up two points.

1. Press the Backspace key.

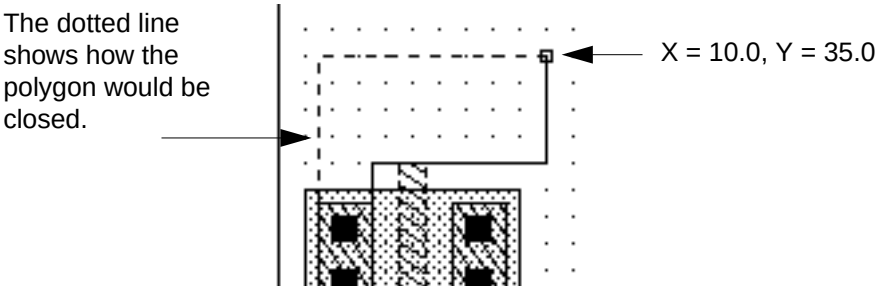
   The last point you entered is undone.

2. To undo the other point, press Backspace again.

Finishing the Polygon

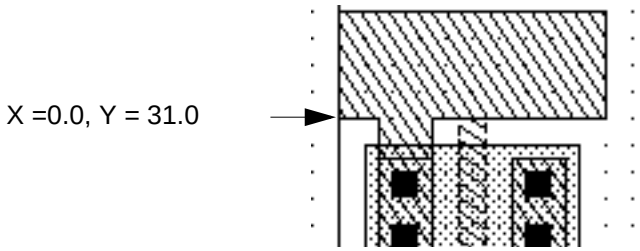
Now you can enter the correct point and finish the polygon.

1. Click X = 10.0, Y = 35.0 to enter the correct point.



The polygon is 1 micron outside of the p-transistor.

2. Double-click X = 0.0, Y = 31.0.



The polygon is complete. You clicked twice to tell the system to close the polygon.

3. Press the `Escape` key to stop the *Create Polygon* command.
4. Press the `f` key to fit the design in the window.

Selecting Objects

After you create objects, you can edit them. To edit an object, first you need to select it. There are two selection modes: full and partial. In full mode, you select the entire object. In partial mode, you can select an entire object or just an edge or corner of an object. You use the `F4` key to toggle between selection modes. The selection mode is displayed in the status banner of the window.

Full mode:

Partial mode:

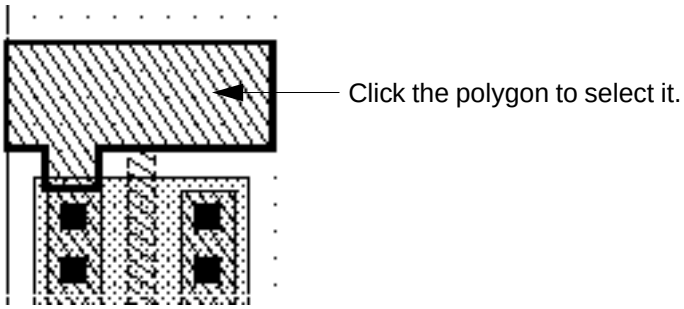
(P) Select: 0

To select an object, set the selection mode and click the object. You also use the left mouse button to deselect objects by clicking in an empty part of the design. You can select one or several objects at a time. In this section, you practice selecting objects in the full mode (the default) before you go on to edit the inverter design.

Note: As you select and deselect objects in this section, notice how the pointer changes. When you select an object, then move the cursor within the selected object, you see the pointer change to four arrows (✥), indicating you can move the object. When you select the edge or vertex (corner) of an object, you see the pointer change to an arrow (⇥ or ↗), indicating you can stretch the edge or corner. Refer to the *Virtuoso Layout Editor User Guide* for detailed descriptions about moving and stretching objects.

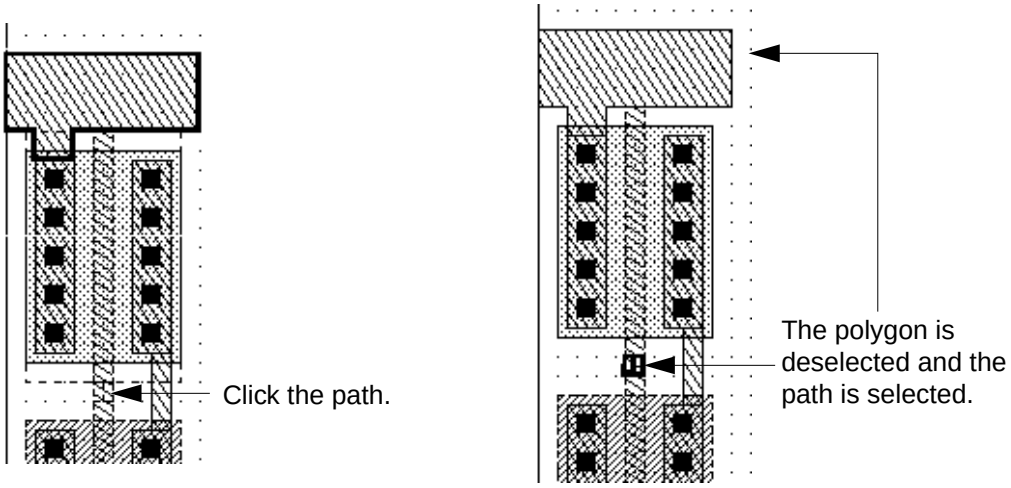
1. Click inside the polygon you just created to select it.

   The polygon is highlighted in a white line, showing it is selected.



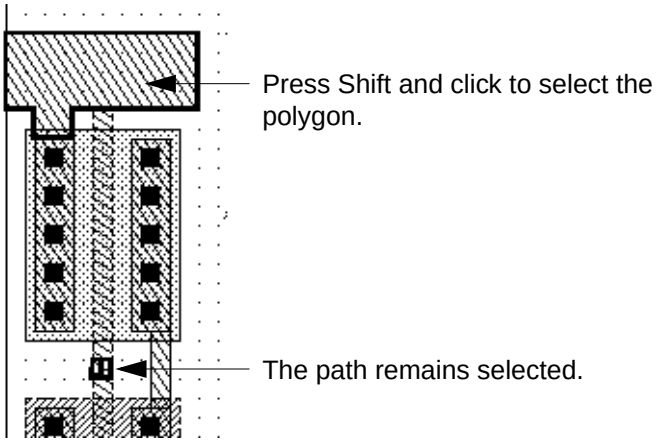
2. Click the path you created earlier.

To make it easier to click the path, you can zoom in. The polygon is deselected, and the path is selected.



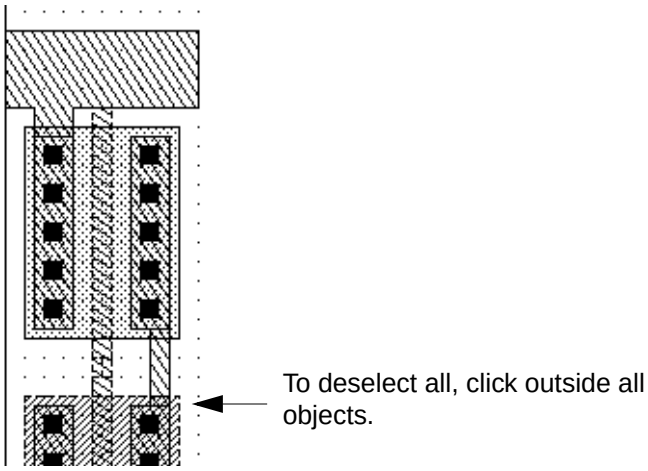
3. Press `Shift` and click the polygon again.

Both the polygon and the path are selected. Pressing `Shift` lets you add objects to the group of selected objects.



4. Click an empty part of the design.

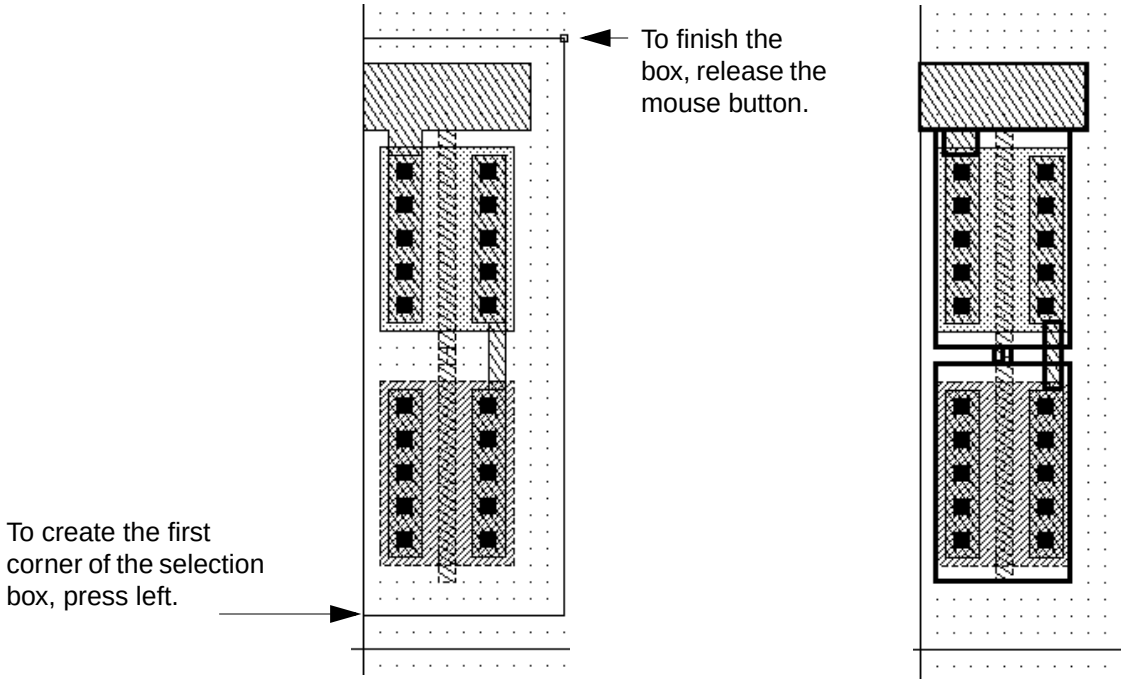
Both the polygon and the path are deselected. Clicking outside all objects deselects any objects that are selected.



5. Select a group of objects by doing the following:

- Move the cursor to the lower left corner of the image.
- Press and drag the cursor to create a box around all of the objects in the image.
- Release the mouse button.

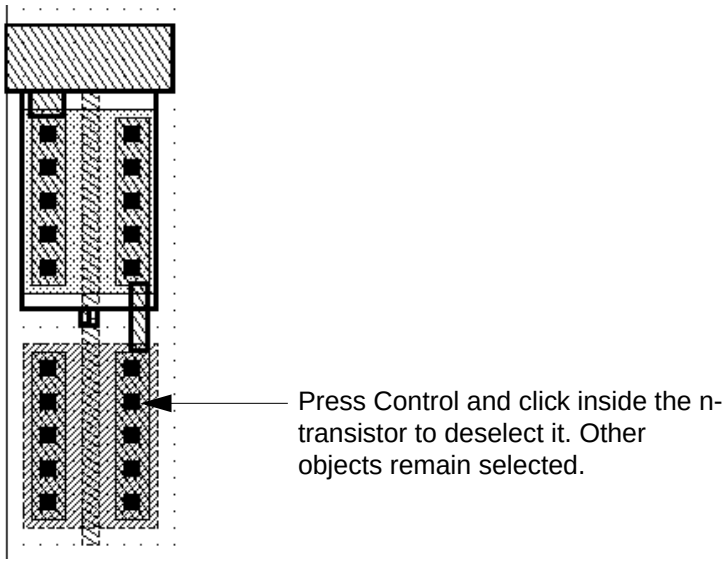
All objects entirely inside the selection box are highlighted and selected.



The objects inside the n- and p-transistors are not highlighted. Because the transistors are cells, only their borders are highlighted.

6. Press `Control` and click the n-transistor.

The n-transistor is deselected; all other objects remain selected. Pressing `Control` lets you deselect one object from a selected group.



7. Click an empty part of the design.

   All objects are deselected.

You can select objects either before or after you start any editing command. In this exercise, you practiced selecting before starting a command. In the rest of this tutorial, you usually select objects after starting a command because this is more typical for new users.

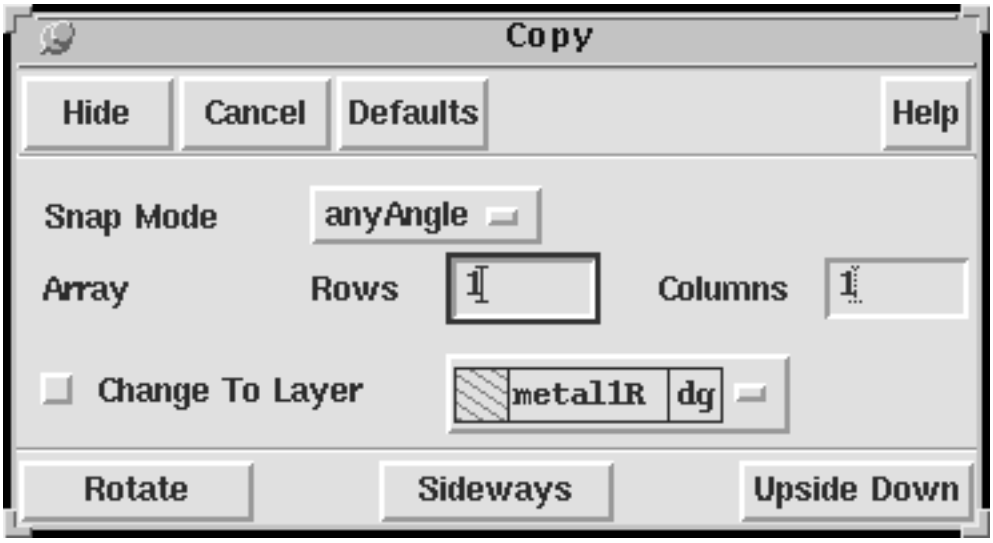
For detailed information about selecting and deselecting objects, refer to the *Virtuoso Layout Editor User Guide*.

Adding the Ground Connection

Use the *Copy* command to copy and mirror the shape you just created to build the ground connection.

1. Choose *Edit – Copy.*

The Copy form appears.



The prompt in the layout window and CIW reads

```
Select the figure to be copied
```

2. Click the *metal1* polygon you just created.

   The polygon is selected.

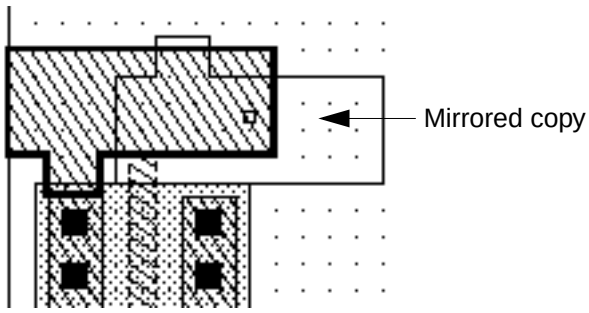
3. Move the cursor slightly to the right.

   A copy of the *metal1* polygon follows the cursor.

You can mirror the copy of the polygon and use it for the ground wire at the bottom of the inverter.

1. In the Copy form, click *Upside Down*.

   The copy of the polygon is mirrored around the X axis.



Panning the Cellview

Before you move the mirrored copy of the polygon, you can pan the image to see the bottom of the design better.

1. Press the `Tab` key to pan.

   The `Tab` key is the bindkey for the *Pan* command.

   The prompt in the layout window and CIW reads

   ```
   Point at the center of the desired display:
   ```

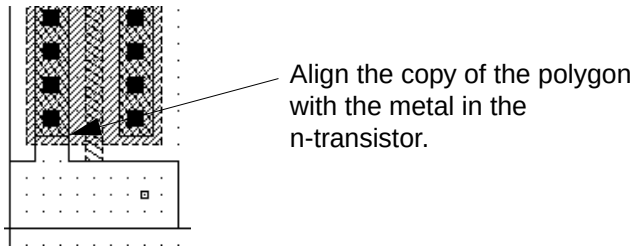
2. Click the top of the n-transistor.

   The image changes so the top of the n-transistor is in the center. The *Pan* prompt disappears from the layout window and CIW, and the *Copy* prompt reappears.

Completing the Copy

Now you can finish moving the mirrored copy of the polygon.

1. Move the mirrored copy of the polygon so it aligns with the metal in the n-transistor.



2. Click left to place the copy of the polygon.

   The copy of the polygon appears on the *metal1* layer.

3. Press the `Escape` key to stop the *Copy* command.

Adding the Well

Next, you create a rectangle on the *nwell* layer to create the well.

In this section, you use the bindkey to start creating the rectangle. You also use a special entry mode called infix, which reduces the number of mouse clicks needed to create shapes.

When infix is on, no click is necessary for the first point of a command. The cursor location at the start of the command is used as the first point. Infix works only with commands started from bindkeys or pop-up menus.

Using Infix

To create the well using infix,

1. From the CIW, choose *Options – User Preferences*.

   The User Preferences form appears.

2. Click the box next to *Infix* to set infix on.

3. Click *OK* to apply the change and close the form.

4. Press the `f` key to fit the design in the window.

5. In the LSW, click *nwell dg*.

   The *nwell* layer appears at the top of the LSW.

6. Move the cursor to X = 0.0, Y = 31.0.

   Because infix is on, this location will be the first corner of your rectangle.

7. Press the `r` key to start the *Create Rectangle* command.

   The rectangle starts at X = 0.0, Y = 31.0. Notice the prompt asks you for the second (opposite) corner of the rectangle:

   ```
   Point at the opposite corner of the rectangle:
   ```

8. Move the cursor to X = 10.0, Y = 18.0.

   An outline of the rectangle stretches with the cursor.

9. Click to complete the rectangle.

   The rectangle appears on the *nwell* layer.

   The prompt in the layout window and CIW reads

   ```
   Point at the first corner of the rectangle:
   ```

   Infix works only for the first rectangle you create after pressing the bindkey.

10. Press the `Escape` key to stop the *Create Rectangle* command.

Turning Infix Off

The remainder of this tutorial does not use infix mode. (You can experiment with it on your own, if you like.)

To turn infix off,

1. From the CIW, choose *Options – User Preferences*.
2. Click the box next to *Infix*.
3. Click *OK*.

Checking Design Rules

Before saving the inverter, check the design against your design rules. Interactive design rule checking is part of Assura™ interactive verification products.

The interactive Design Rule Checker (DRC) uses rules defined in the `divaDRC.rul` file. For the tutorial, these rules have been defined for you. For details about the SKILL functions used to write these rules, refer to the *Assura Diva Verification Reference* manual.

DRC flags the errors it finds by creating polygons around the errors. The polygons are created on a layer reserved for markers. The marker layer usually appears as a blinking layer and is not selectable. DRC removes the error-flag polygons automatically after you correct the errors and run DRC again.

Finding Out if You Can Run DRC

You might not have a license to run interactive verification even if you can use the layout editor. To find out whether you have a license to run interactive verification,

➤ Click the *Verify* menu.

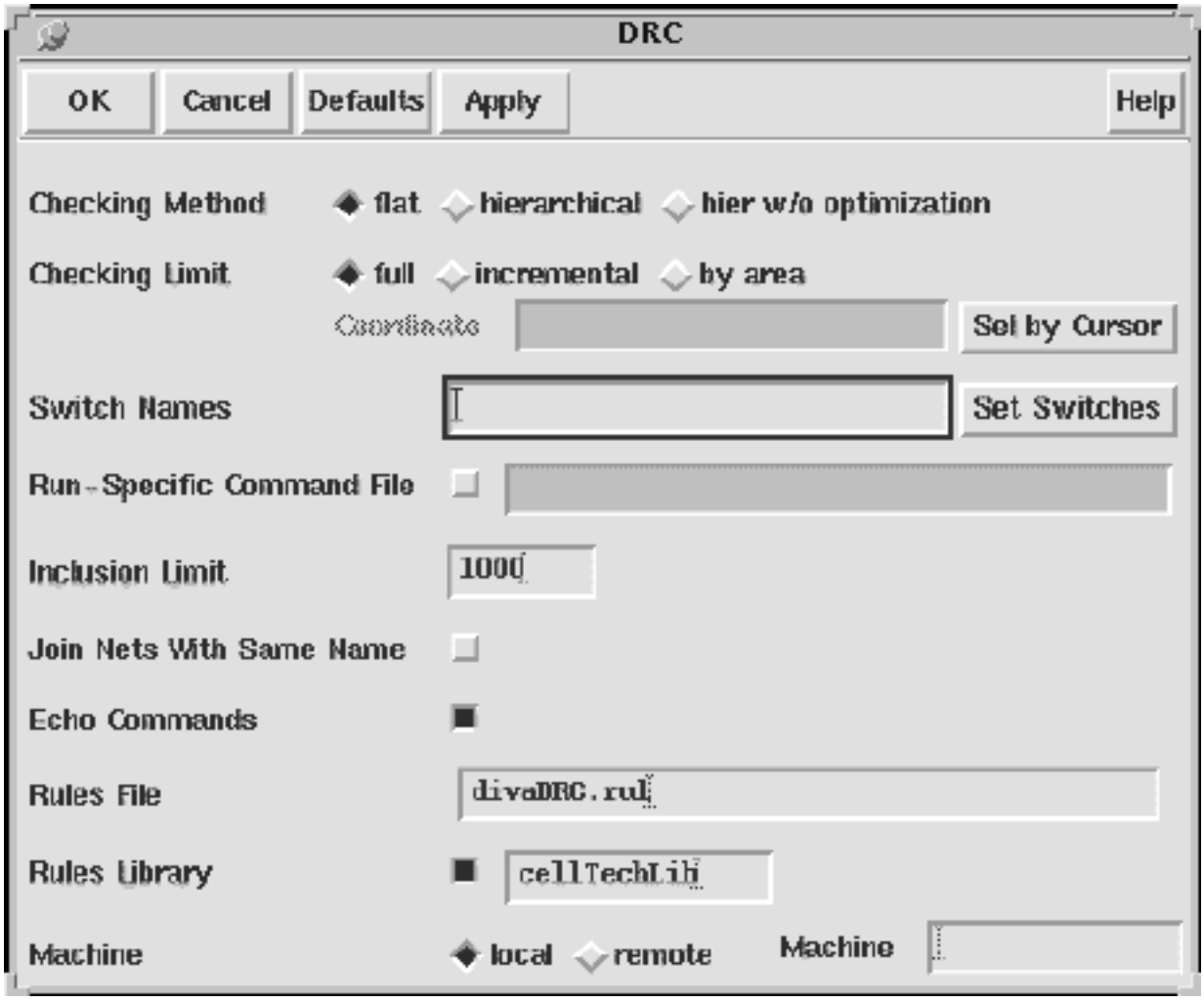
If the commands under *Verify* appear shaded, you do not have a license to run interactive verification. Go on to the "Saving Your Design" on page 56.

If the commands are not shaded, you have a license to run interactive verification. Go to the next section, Running the Design Rule Checker.

Running the Design Rule Checker

1. Choose *Verify – DRC*.

The DRC form appears.



2. Click *OK* to start the Design Rule Checker.

   If you followed the previous steps exactly, there should be no errors. If you have errors, you see a blinking polygon marking the first error. Follow the next steps to fix any errors.

Deleting Objects

The easiest way to correct errors is to delete the objects you created that were in error and recreate them using the original instructions in this chapter.

To delete objects,

1. Click the object you want to delete.

When you want to delete an object, it is easier to select the object before starting the *Delete* command. If you select the wrong object, move the cursor and click again.

2. Press the `Delete` key.

   Now you can recreate the object using the original instructions in this chapter.

Saving Your Design

You have completed the inverter layout. However, the layout exists only in virtual memory. You need to save the design to disk.

To save the design,

1. Choose *Design – Save*.

   Save copies the design in virtual memory to disk under the library directory specified when the design was created. This design is saved to the `tutorial` library. You use this design in the next chapter.

2. Choose *Window – Close*.

Summary

In this chapter, you learned how to use the layout editor to create, select, and edit shapes. Specifically, you

- Set up an environment for entering points
  - Created a new cellview
  - Learned how to zoom the display
  - Learned about grid displays
  - Panned the image
- Created instances of n- and p-transistor parameterized cells (pcells)
  - Learned how to place cell instances
  - Learned how to assign parameter values to pcells
- Created and edited shapes
  - Set the entry layer using the LSW

- Created a path
- Created a rectangle
- Created a polygon
- Learned how to select shapes
- Copied a polygon

- Learned how to correct mistakes
  - Learned how to undo both *Create* and *Edit* commands
  - Deleted a point you created
- Learned how to run the Design Rule Checker
- Learned how to save your design and close the window
- Used bindkeys for
  - Fit All [f]
  - Pan [Tab]

3

Creating the Multiplexer Layout

In this chapter, you learn to use the Virtuoso® layout editor to create a hierarchical design for a multiplexer by doing the following tasks. You must run Virtuoso® layout accelerator to create part of the design. The instructions for starting layout accelerator are in Creating a Guard Ring on page 89. You also continue to use basic editing commands and learn some shortcuts.

When you finish this chapter, you will be able to

- Copy cell instances to create a hierarchical design
- View different levels of hierarchy
- Edit a cell while viewing it in the context of the surrounding design
- Stretch an area
- Flatten instances
- Use path stitching
- Create labels
- Create pins
- Create a multipart path

If You Have Not Completed the Previous Chapters

This chapter assumes you have followed the steps in the previous chapters. If you did not follow the steps in the previous chapters but want to go through this chapter, you can copy a completed design from the `master` library as explained in the following steps.

If you did follow the steps in the previous chapters you can skip to "Creating a Hierarchical Layout" on page 60.

Note: It is possible to run out of resources, such as memory, if you run multiple layout editors. Before you start the software, you need to check whether the software is already running.

1. Type the following in an xterm window to check whether the layout editor is already running:

   ```
   ps auxw | grep layout
   ```

2. If the layout editor is running, choose *File – Exit* in the Command Interpreter Window (CIW) to exit the software.

3. Type the following in an xterm window to start the layout editor:

   ```
   cd ~/cell_design
   layoutPlus &
   ```

4. Choose *File – Open*.

   The Open File form appears.

5. Type the library, cell, and view names as follows:

Library Name	`master`
Cell Name	`Inv_save`
View Name	`layout`

6. Click *OK*.

   The inverter cell from the `master` library opens.

7. In the cellview window, choose *Design – Save As*.

   The Save As form appears.

8. In the Save As form, type the library and cell names as follows:

Library Name	`tutorial`
Cell Name	`Inv`

9. Click *OK*.

   The inverter cell is copied to the `tutorial` library but the inverter cell from the `master` library remains on your screen.

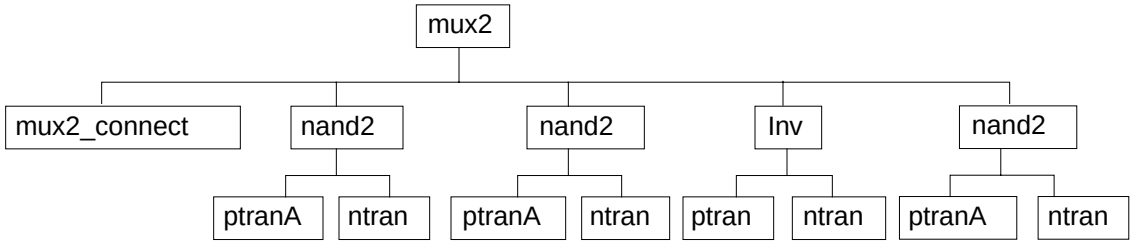
10. In the cellview window, choose *Window – Close* to close the inverter cellview from the `master` library.

    You use the inverter from the `tutorial` library later in this chapter.

Creating a Hierarchical Layout

In this section, you create a *hierarchical* design. A hierarchical design is one containing instances of other cells. Those cells, in turn, can contain instances of cells.

You create the multiplexer in this section by placing instances of several cells inside the multiplexer cellview, *mux2*, as shown in the following figure.



You place the following cells from the `tutorial` library: *mux2_connect*, *nand2*, and *Inv,* the inverter you created in the previous chapter or just copied from the `master` library. Most of these cells contain other cells.

Later in this section, you flatten the *mux2_connect* cell, so its contents appear in the top *mux2* cellview and it is no longer an instance.

In the following sections, you learn to

- Create the multiplexer layout, *mux2*

- Create and copy the nand instance
- Create the inverter instance

Creating a New Cellview

In this section, you learn to create a layout view for the *mux2* cell.

1. In the Command Interpreter Window (CIW), choose *File – New – Cellview.*

   The Create New File form appears. If you have continued from Chapter 2, the last file you created (*Inv layout*) is displayed in the form. If you have just started, no cell name is displayed, and you must set the library, cell, and view names as follows:

Library Name	`tutorial`
Cell Name	`mux2`
View Name	`layout`
Tool	`Virtuoso`

2. Click *OK* to create the *mux2* layout.

   A window appears containing just the cellview axes and grid points.

Opening a Schematic for Reference

You can open the schematic view of the *mux2* design for reference as you build the layout view of *mux2*.

1. Choose *File – Open.*

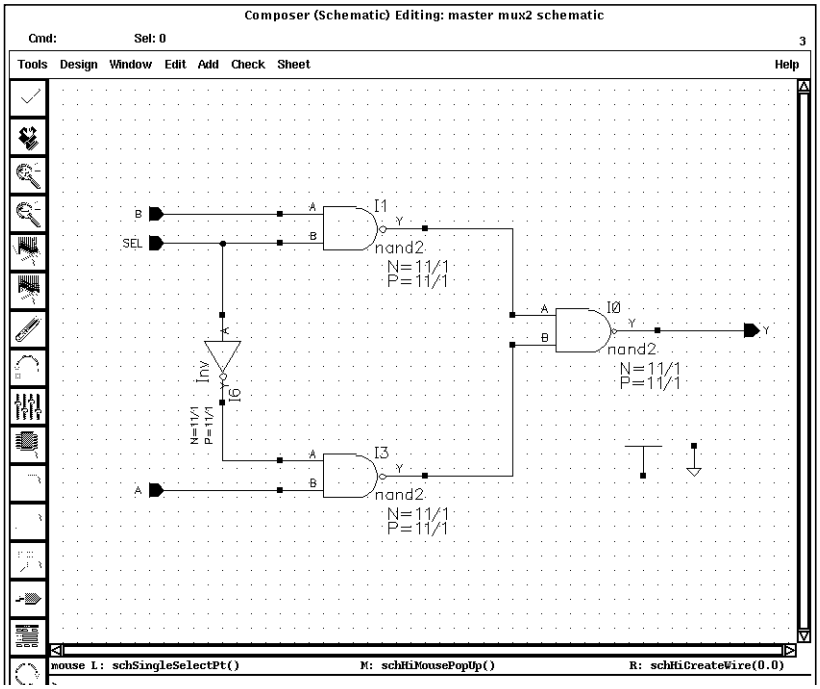
   The Open File form appears.

2. Set the library, cell, and view names as follows:

Library Name	`master`
Cell Name	`mux2`
View Name	`schematic`

3. Click *OK* or press the `Return` key.

The schematic view of the *mux2* design appears.



Note: For a better fit of all your windows on your screen, click and hold any corner of the schematic window and move the mouse until the window is a smaller size. Then press the `f` key in the schematic window to fit the schematic drawing within the resized window.

Creating the First nand2 Instance

You are now ready to place the first instance of the *nand2* cell into the *mux2* cellview.

1. Move the cursor inside the *mux2* layout window and press the `i` key.

   The Create Instance form appears. If you have continued from Chapter 2, the last cell you placed (*ptransistor*) is displayed in the form. If you have just started, no cell name is displayed.

2. Type the library, cell, and view names as follows:

Library Name	`master`
Cell Name	`nand2`
View Name	`layout`

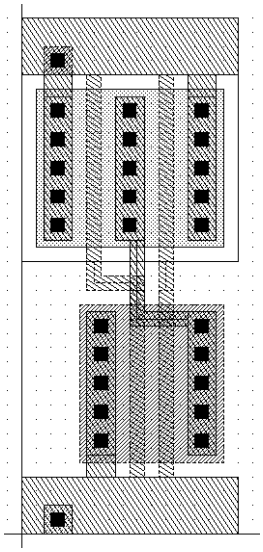
3. Click X = 0, Y = 0 to place the first *nand2* instance.

   The *nand2* instance appears in the *mux2* cellview.

4. Press the Escape key to stop the *Create Instance* command.

   The Create Instance form disappears.

5. Press the f key to fit the design in the window.



6. You need to move to the right of the nand2. You do this with the *Pan* command. Press the Tab key to start the *Pan* command.

7. Click X = 18.0, Y = 18.0 to move to the right of the nand2.

Copying the nand2 Instance

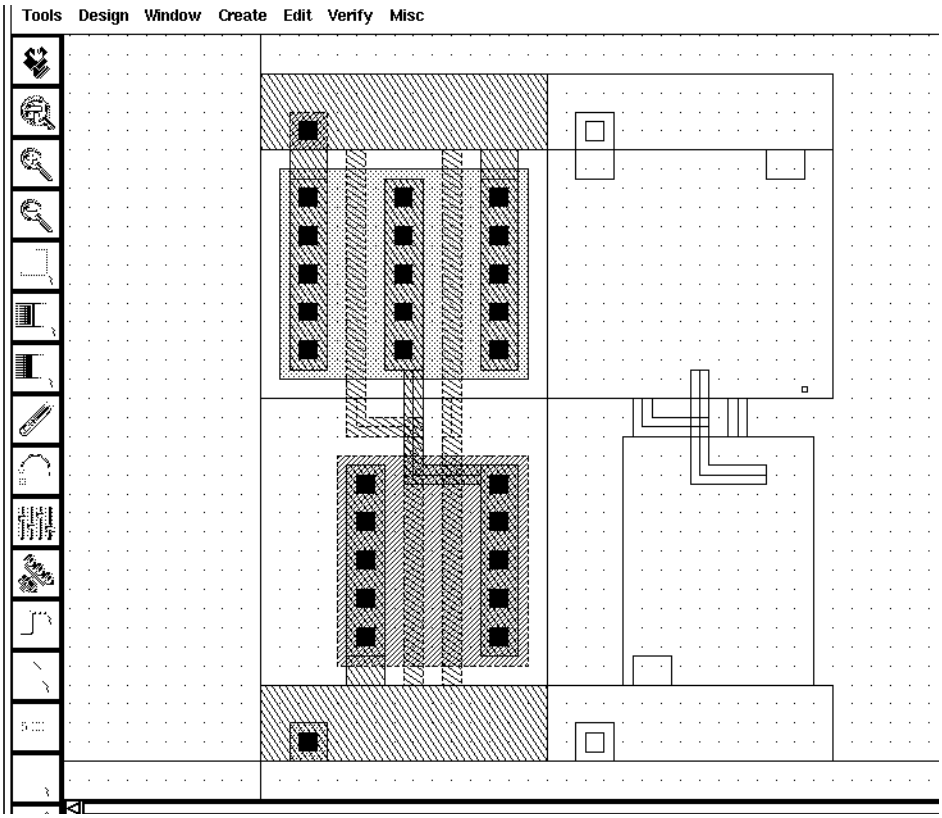
You copy the first nand instance to create the second nand instance.

1. Press the c key to start the *Copy* command.

2. Click anywhere inside the first instance.

   The outline and shapes inside the first instance are highlighted.

3. Move the cursor to the right until the second instance aligns with the first.



4. Click to place the copy.

Creating the Inv Instance

Now you place an instance of the inverter layout you created in the previous chapter or copied from the `master` library.

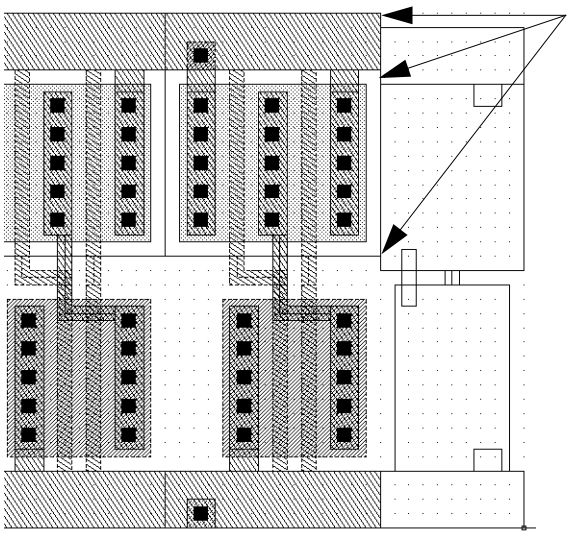
1. Press the `Tab` key, and click X = 32.0, Y = 18.0 to pan right.
2. Press the `i` key.

   The *Copy* command is canceled and the Create Instance form appears.
3. In the Create Instance form, type the library, cell, and view names as follows:

Library Name	`tutorial`
Cell Name	`Inv`
View Name	`layout`

4. You need to mirror the inverter along the X axis. Click *Sideways* to mirror the inverter.
5. In the cellview, click X = 40, Y = 0 for the instance origin.
6. Press the Escape key to stop the *Create Instance* command.

   The inverter appears next to the nands.



The inverter is not aligned properly with the nands. You correct this error in the following section.

Editing the Inverter in Place

The inverter you placed in the previous section does not align properly with the nands. You can correct this error by stretching the top half of the inverter up by 1 micron. But you cannot edit the inverter now because you are looking only at an instance of the inverter cell inside the multiplexer.

You could open the inverter cell layout in a separate window and edit the inverter there, but then you would not be able to see how the inverter aligns with the other instances in the multiplexer.

The *Edit in Place* command lets you edit the inverter master cell while viewing it inside the multiplexer layout. This way, you can edit the inverter instance as it appears in this cellview and see how it aligns with the nand next to it.

In the following sections, you edit in place to correct the inverter cell. You learn to

- Open the inverter cell to edit in place
- Stretch the top of the inverter
- Return to editing the *mux2* cellview

Opening a Cell to Edit in Place

Open the inverter cell for editing using *Edit in Place*.

1. Choose *Design – Hierarchy – Edit in Place*.

   The layout editor prompts you to point to a shape in the design to be edited.

2. Click the *metal1* polygon at the top or bottom of the inverter.

   Note: The transistors are (pcells) parameterized cells. You cannot edit pcells in place because they must be created by compiling (you learn how to compile pcells later in this tutorial). If you accidentally click on a shape inside one of the pcells, a message says you cannot edit pcells in place.

   When you successfully open the inverter using *Edit in Place*, you might not notice any change in the display. Look at the window title to see that you are editing the *Inv layout*.

3. Choose *Window – Fit Edit*.

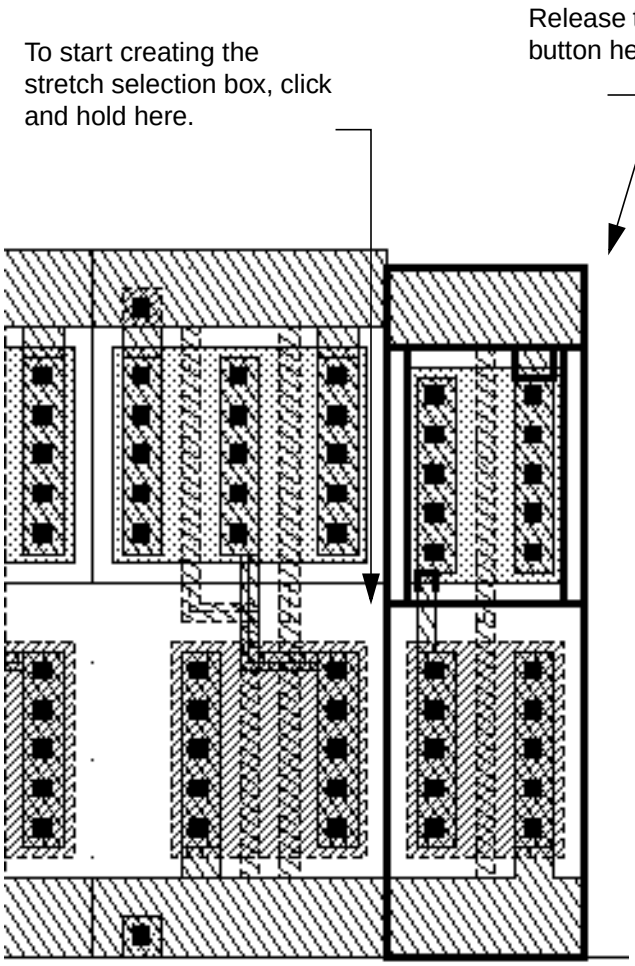
   The *Inv layout* data is fitted to the window and the border of the inverter is highlighted in light brown. This confirms you are editing the inverter cell. You still see the surrounding multiplexer data, but you cannot edit it because it is at a different level of the hierarchy.

Stretching an Area

Use *Stretch* to stretch the top of the inverter.

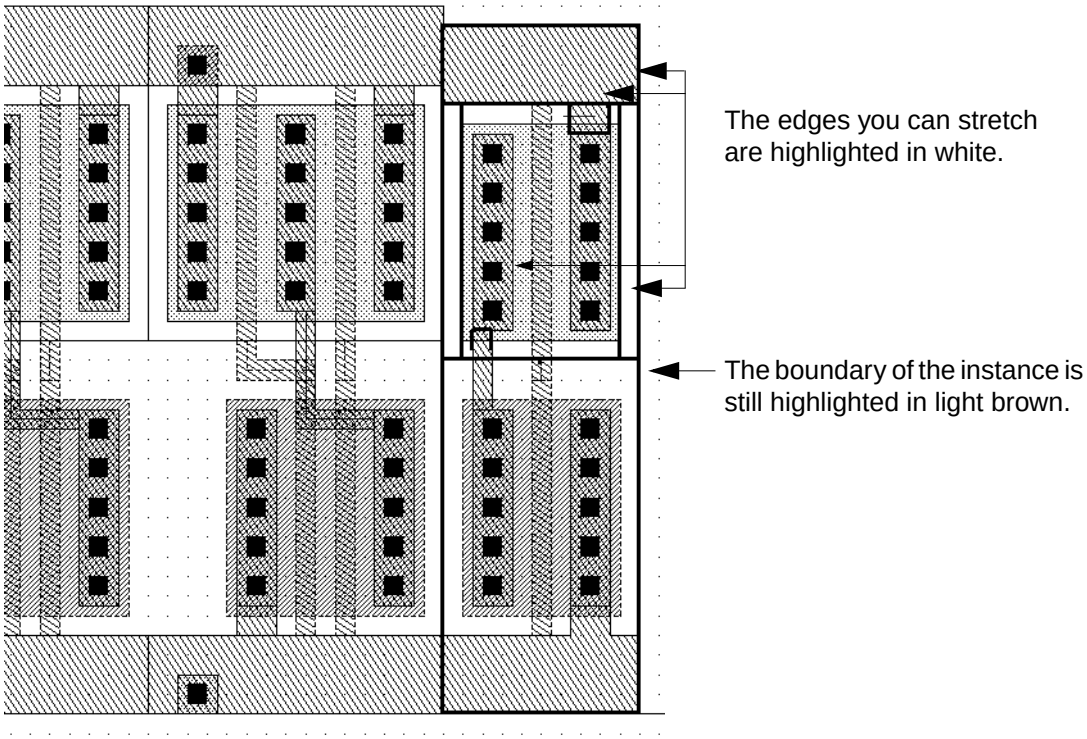
1. Choose *Edit – Stretch*.

2. You need to define the area you want to stretch. Click X = 29, Y = 18 and drag the box to a point above and to the right of the area to be stretched.



3. Release the mouse button.

The edges you can stretch are highlighted.

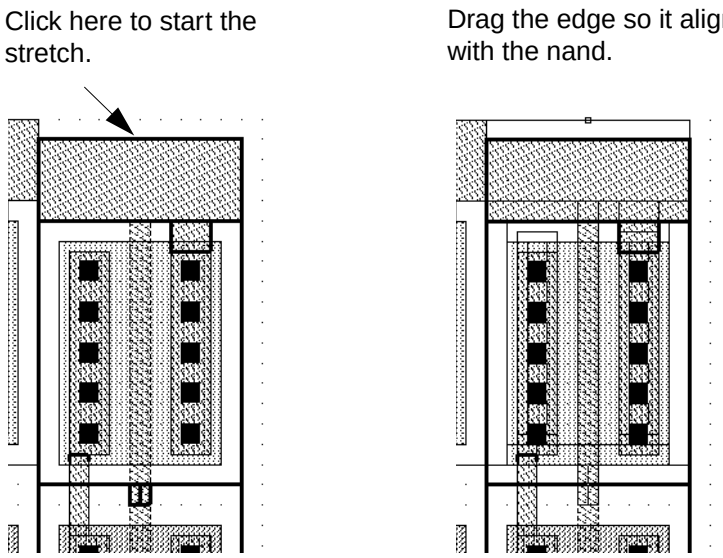


The prompt in the CIW reads

```
Point at the reference point for the stretch
```

The layout editor often asks for a *reference point* as you use editing commands. The reference point is the startpoint for the command; for example, the starting point from which you move a group of objects. In this case, the reference point is the starting point for the stretch.

4. Click the top edge of the polygon for a reference point. Move the cursor up until the edge of the inverter aligns with the nand.



5. Click to complete the stretch.

   The inverter and the nand instance now align correctly.

   The inverter is still highlighted in light brown to remind you that you are editing this cell, not the multiplexer.

6. Press the Escape key to stop the *Stretch* command.

Returning to the Multiplexer

While you edit a cell in place, you cannot edit the surrounding data (in this case, the other objects in *mux2*). When you finish editing the inverter in place, you must return to editing the multiplexer cell.

1. Choose *Design – Hierarchy – Return*.

   Return takes you back to the previous editing level. It also checks whether you have saved or not. Because you did not save yet, a dialog box appears asking if you want to save your changes to *tutorial Inv layout*.

2. Click *Yes* to save your changes.

The inverter is saved, and the window title bar shows that you are now editing the multiplexer (*mux2*) again.

Displaying Hierarchy Levels

You might want to verify the instances you have created so far. In this section, you learn to

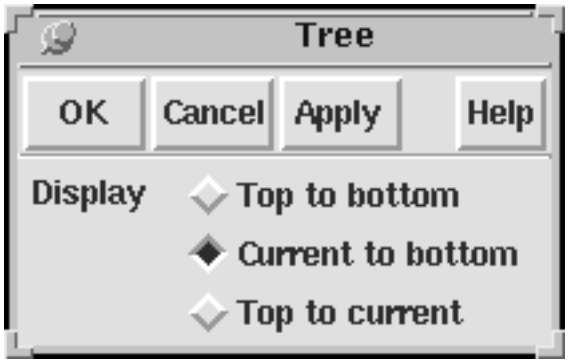
- Use the *Tree* command to list the names of the cell instances in *mux2*
- Change the amount of detail displayed in the *mux2* cellview, so you see only the instance names

Listing the Cells in the Multiplexer

Use the *Tree* command to list all the cell instances placed in the multiplexer.

1. Choose *Design – Hierarchy – Tree*.

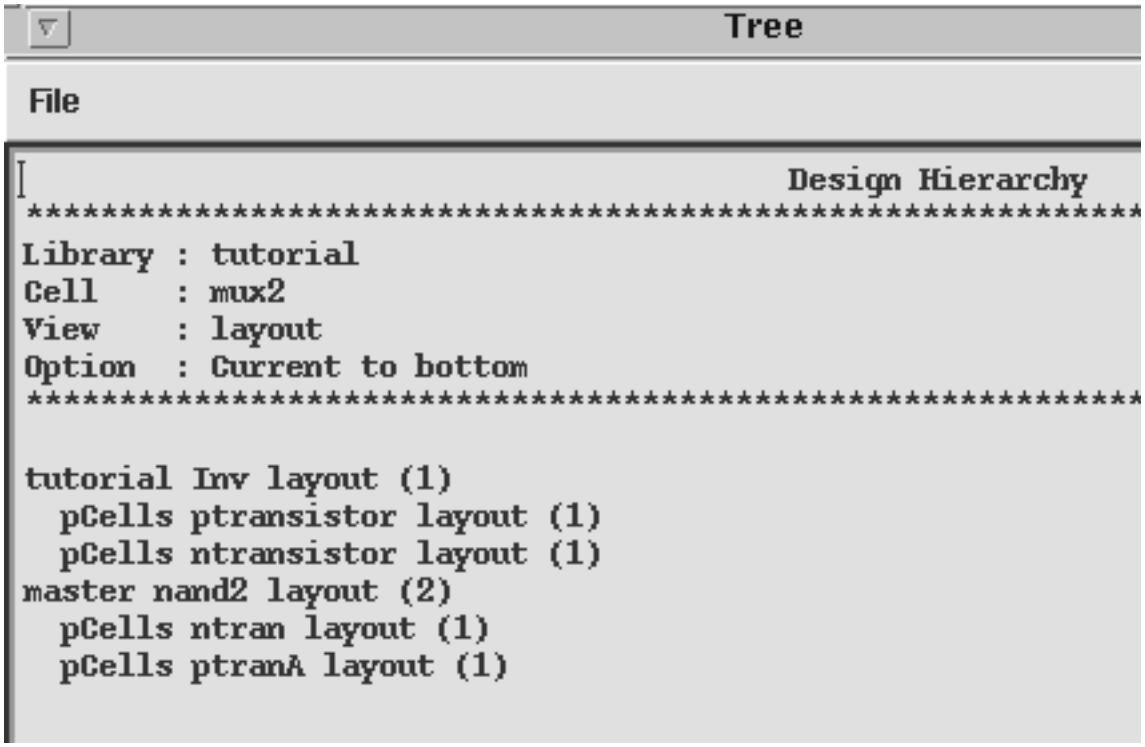
   The Tree form appears.



   The *Display* option lets you choose how much of the hierarchy you want to see. The top is this cellview and the current cell is the cell you are editing. When you were editing the *Inv* cell in place, it was the current cell.

2. Click *OK*.

A window appears, listing the cells inside the *mux2* cellview. The window displayed by *Tree* is a text window. Commands that list information usually display the data in a text window.



Tree shows all the hierarchy contained in *mux2*. It lists all the cell instances placed in *mux2* and any instances inside those cells. It gives the library, cell, and view name for each instance. It also shows the number of instances of a particular cell, in parentheses.

You can use *Tree* whenever you want to review the levels of hierarchy in a design.

3. Click *File* to display the *File* menu.

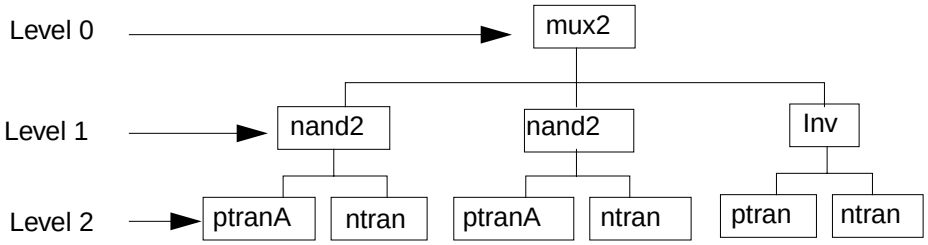
   You can use the *File* menu to save the data in the window to a text file or to search through the text.

4. Choose *File – Close Window* to close the window.

Changing Display Levels

At present, you are viewing all the detail in all instances placed inside the *mux2* cell. You can turn off some of this detail so you see only the outlines of instances and the master cell name for each instance.

To change the amount of detail in a cellview, you indicate the starting level in the hierarchy you want to view and the hierarchy level at which you want to stop viewing detail. The current cellview (in this case, *mux2*) is level 0 in the hierarchy. Any cells directly inside it are level 1. Cell instances inside those cells are level 2.



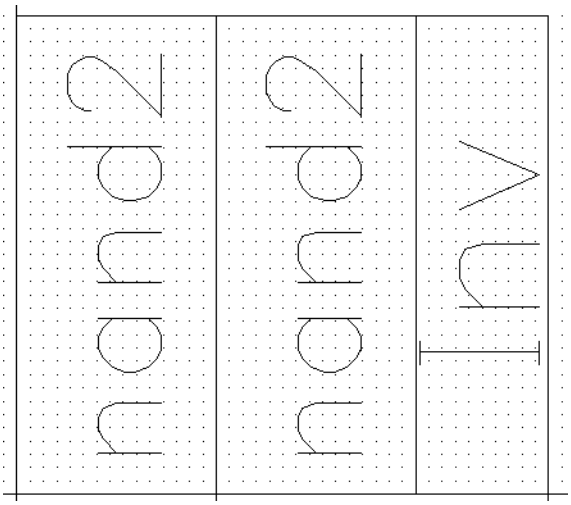
In the following steps, you learn how to display only the top level of the hierarchy (level 0) and how to display data within a range of levels.

1. Press the `f` key to fit the design in the window.
2. Choose *Options – Display*.

   The Display Options form appears.
3. Under *Display Levels*, change the *To* field to `0`.
4. Click *Apply*.

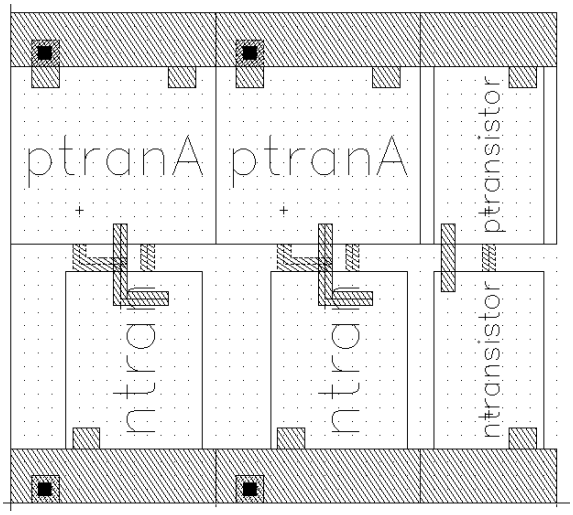
   The multiplexer is redrawn to show only data at level 0. You see only the outlines and master cell names for the instances you placed. These are the only objects at level 0.

The master cell name is not relative to the orientation of the instance; it is just displayed as large as possible inside the outline.



5. Change the *To* field to 1.
6. Click *Apply*.

   The multiplexer is redrawn to show data at levels 0 through 1. You see the objects in the *nand2* and *inverter* layouts but only outlines for the cells at the next level.



7. Change the *To* field to 32.
8. Click *OK*.

   All levels of data are displayed.

Because you change display levels often, you can use the following bindkeys.

- `Shift-f` displays levels 0 through 32.
- `Control-f` displays only level 0.

9. Press `Control-f`.

Only level 0 data is displayed.

10. Press `Shift-f`.

All levels of data are displayed.

Placing and Flattening an Instance

You still need to place the last nand for the multiplexer. You also need to create the connections between the instances. The tutorial database contains a layout cell containing predefined connections for the instances in the multiplexer so you do not have to create all the connections yourself. You create the final connections later in this chapter.

Normally, you would create the connections at the top level (level 0). Because you place these connections using a cell instance (level 1), you flatten the instance after you place it. The *Flatten* command moves the contents of a cell or array up one or more levels in the hierarchy. In this case, flattening the instance moves its contents up to the top level so it is no longer an instance.

In the following sections, you learn to

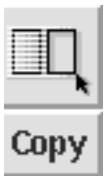
- Copy and place the last nand in the multiplexer
- Place the *mux2_connect* instance containing most of the connections for the multiplexer
- Flatten the *mux2_connect* instance so it is no longer an instance, moving its contents up one level in the hierarchy

Copying the nand2 Instance Again

Now you copy one of the *nand2* instances to the right of the inverter to place the last nand in the multiplexer. You do this just as you did for the second nand as described in "Copying the nand2 Instance" on page 63. If you feel confident, copy the nand again on your own and skip to the "Placing the Connect Cell" on page 76.

1. Press the `Tab` key and click X = 35.0, Y = 18.0 to pan right.

2. Click the *Copy* command icon in the icon menu to start the *Copy* command.

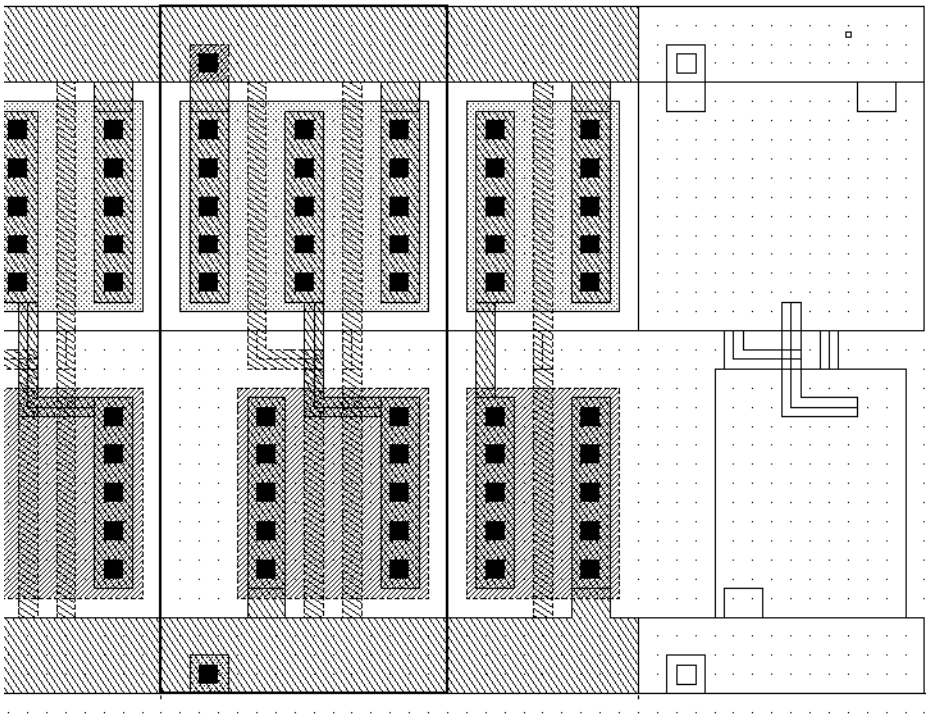


   The Copy form appears.

3. Click anywhere inside the second *nand2* instance.

   The outline and shapes inside the instance are highlighted.

4. Move the cursor to the right until the last *nand2* aligns with the inverter.



5. Click to place the copy.

6. Press the `Escape` key to stop the *Copy* command.

7. Press the `f` key to fit the entire design in the multiplexer window.

Placing the Connect Cell

Use *Create Instance* to place a layout cell containing predefined connections for the instances in the multiplexer.

1. Click the *Create Instance* command icon in the icon menu to display the Create Instance form.

   The Create Instance form appears.

2. Type the library and cell names as follows:

Library Name	`master`
Cell Name	`mux2_connect`

3. Move the cursor into the *mux2* cellview.

   An outline of the connections in *mux2_connect* follows the cursor.

4. Move the cursor to X = 0, Y = 0.

   The *poly1* connections at the top and bottom of the multiplexer should align exactly.

5. Click to place the *mux2_connect* cell.

6. Press the `Escape` key to stop the *Create Instance* command.

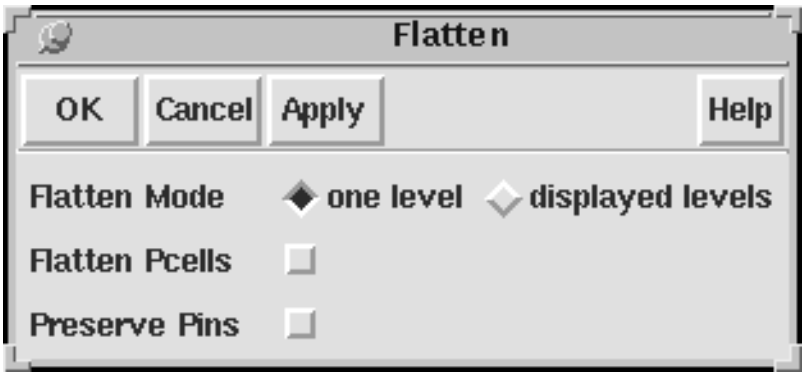
Flattening the Connect Cell

Normally, you would create the connections between the instances in the multiplexer at the top level (level 0); they would not be in a cell. The *mux2_connect* cell was provided to save you time. You use *Flatten* to move the data in *mux2_connect* up to level 0 so it is no longer an instance.

1. Press `Control-f` to display outlines of the instances.

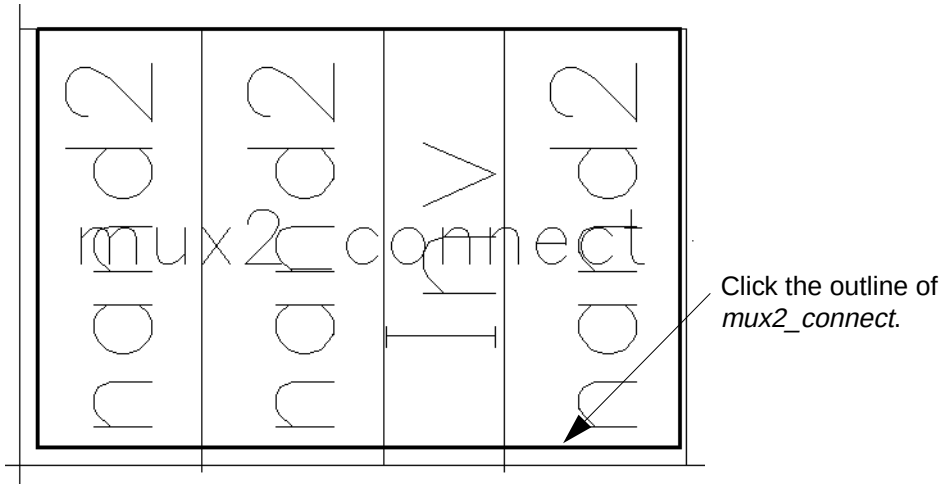
   It is often easier to select a cell instance while viewing only instance outlines.

2. Choose *Edit – Hierarchy – Flatten*.



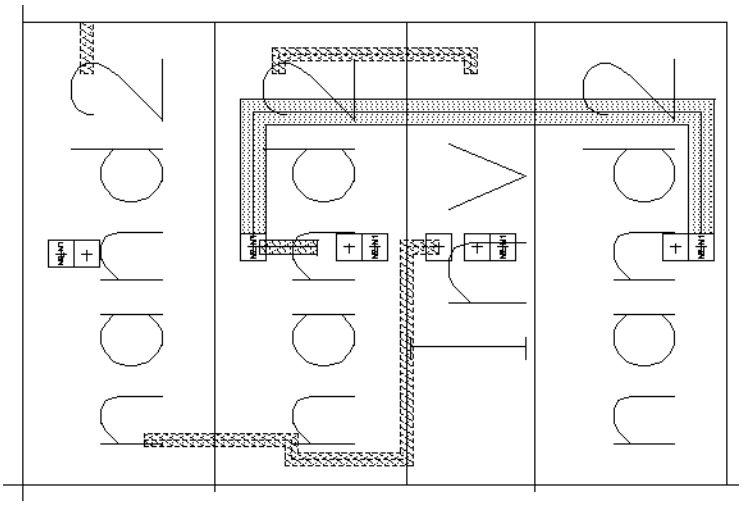
The Flatten form appears. You need to move the contents of *mux2_connect* up just one level (from level 1 to level 0). The default form settings are set correctly.

3. Click the outline of the *mux2_connect* cell.



4. Click *OK*.

The instance outline disappears and the data in the *mux2_connect* cell appears.



5. To display all levels, press `Shift-f`.

Saving the Design

It is always a good idea to save your design periodically.

➤ Press the F2 key to save your design.

The multiplexer is saved to disk.

Using Path Stitching

Not all the connections for the multiplexer were in the *mux2_connect* cell you placed in the previous section. You still need to connect the output of the first *nand2* instance to one of the inputs for the last *nand2*.

You use *Path* to create the final connections. You cannot create a path on a single layer all the way across the multiplexer, so you use *path stitching* to change layers within the path. Path stitching automatically changes the path from one layer to another, placing an appropriate contact to connect the two layers. The *Create Path* command chooses the contact for you from the technology file.

In this section, you learn to

- Turn gravity off so you can create the path

- Start the path on the *metal1* layer
- Switch to the *metal2* layer, automatically placing a metal1-to-metal2 contact
- Switch back to *metal1*, then to *poly1*, each time placing contacts

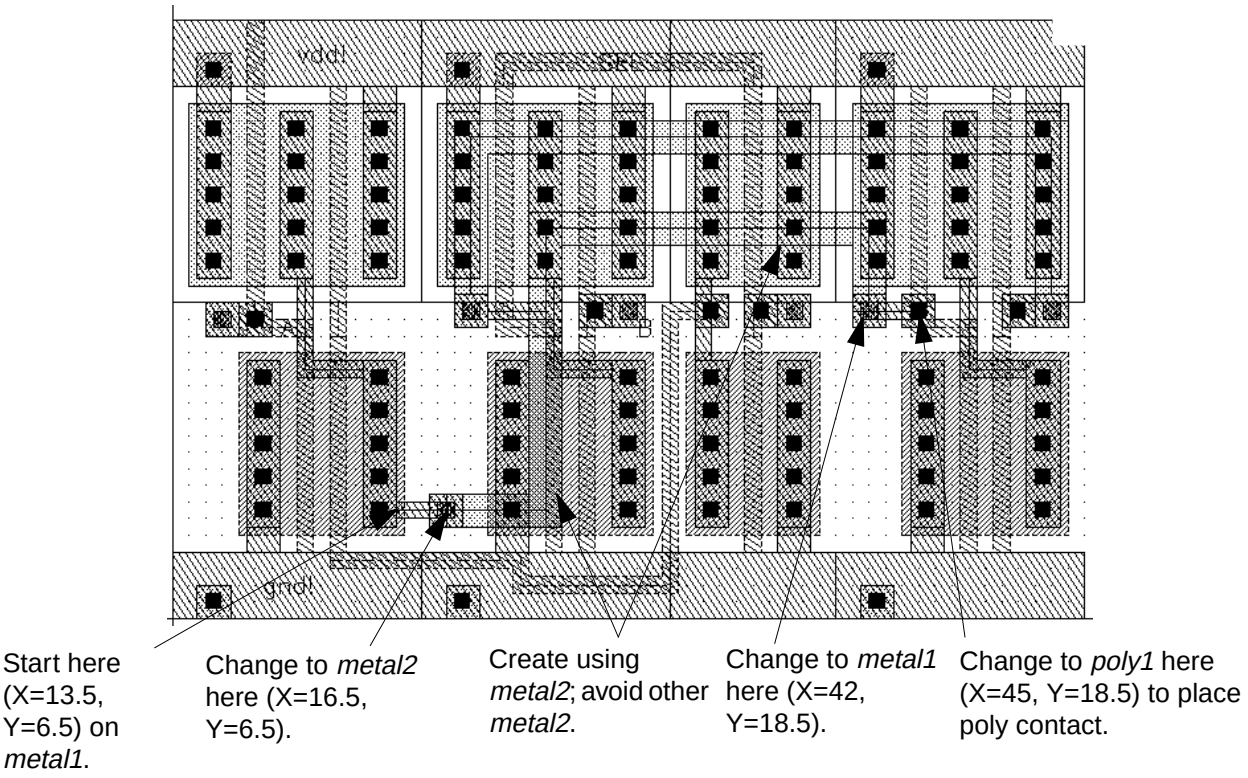
Turning Gravity Off

Snapping the cursor to objects is called *gravity.* Gravity is turned on by default and is helpful when creating instances and devices. In this section, you turn gravity off to make it easier to create a path. You can toggle the gravity on and off using either the Layout Editor Options form or the g bindkey.

➤ Move the cursor inside the *mux2* layout and press the g key to turn gravity off.

Overview of Path Route

The connections you make are shown below. You can use the illustration and the summary table as guides while you go through the following steps.



Note: If you make an incorrect click while path stitching, press the `Backspace` key to cancel the last click, and then continue with your path.

Table 3-1 Summary of Path Route

Number of points	Location	Angle	Layer	Purpose
First	X=13.5, Y=6.5	orthogonal	metal1	Start path
Second	X=16.5, Y=6.5	orthogonal	metal1	Create path
Third	X=16.5, Y=6.5	orthogonal	metal2	Place contact

Table 3-1 Summary of Path Route

Number of points	Location	Angle	Layer	Purpose
Fourth	X=22.5, Y=23.5	L90XFirst	metal2	Create path
Fifth	X=42, Y=18.5	L90XFirst	metal2	Create path
Sixth	X=42, Y=18.5	L90XFirst	metal1	Place contact
Seventh	X=45, Y=18.5	L90XFirst	metal1	Create path
Eight	X=45, Y=18.5	L90XFirst	poly1	Place contact
Ninth	X=45, Y=18.5	L90XFirst	poly1	End path

Starting the Path on metal1

1. In the Layer Selection Window (LSW), click the *metal1 dg* layer.
2. In the cellview window, press the p key.

   The Create Path form appears. Notice that the path width is set to 1 micron. This width is defined in the technology file as a property of the *metal1* layer.
3. Click X = 13.5, Y = 6.5 to start the path.
4. Click X = 16.5, Y = 6.5.

Changing to metal2

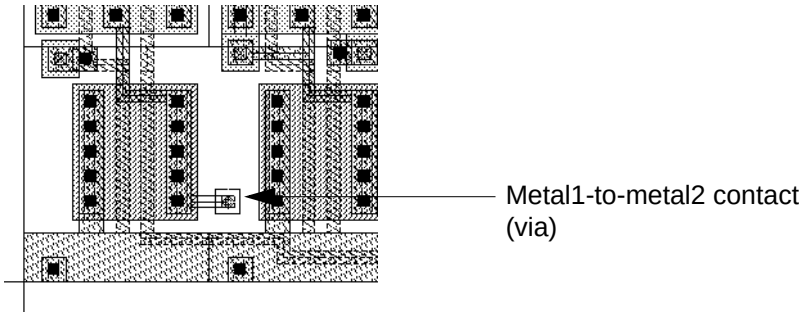
You use the *Change To* cyclic field in the Create Path form to change from *metal1* to *metal2*.

1. In the Create Path form, click and hold *metal1 dg* in the *Change To* field.

A list of layer names appears. These are the layers you can change to from *metal1*, based on the *metal1* contacts defined in your technology file.

2. Slide the cursor to *metal2* and release.
3. Move the cursor back into the cellview.

   A metal1-to-metal2 contact appears on top of your cursor.



4. Click again on X = 16.5, Y = 6.5 to anchor the contact.

   The entry layer shown at the top of the LSW changes to *metal2*. You are now entering points on *metal2*. The path width changes to 2 microns, the width set in the technology file for the *metal2* layer.
5. In the Create Path form, change the *Snap Mode* to *L90XFirst*.
6. Move the cursor back to the cellview.
7. Click X = 22.5, Y = 23.5.
8. Click X = 42, Y = 18.5.

Completing the Path

To complete the path, you change the path layer two more times: to *metal1* and then to *poly1*.

1. In the Create Path form, choose *metal1* from the *Change To* field.
2. Move the cursor back into the cellview.

   A metal2-to-metal1 contact appears on top of your cursor.
3. Click again on X = 42, Y = 18.5 to anchor the contact.
4. Click X = 45, Y = 18.5.
5. In the Create Path form, choose *poly1* from the *Change To* field.

   A metal1-to-poly1 contact appears on top of your cursor.
6. Click again on X = 45, Y = 18.5 to anchor the contact.
7. Check that your cursor is on X = 45, Y = 18.5 and press `Return` to complete the path.

   The connections for the multiplexer are now complete.
8. Press the `Escape` key to stop the *Create Path* command.
9. Press the `F2` key to save your design.

Creating Pins

Now that all the connections are complete, you need to add some information used by other Cadence® tools.

You need to add net labels to help you diagnose problems found by the Layout Versus Schematic (LVS) program. You run LVS in the next chapter.

You also need to create pins. Pins are used by Cadence tools as follows:

- Pins define the connectivity between hierarchy levels. That is, a pin indicates where this cell can connect to routing or to other instances when the cell is placed into a larger design.
- Pins specify the access directions for Cadence routing tools.
- LVS checks to see if you have placed labels that conflict with the nets you define for the pins.

In this section, you

- Create pins and label them
- Save the multiplexer design to disk

About Pins

Pins show what areas of the multiplexer can connect to routing or other cells when you place an instance of the *mux2* into another design cell.

Note: You create pins coincident with shapes in the instances placed in *mux2*. If you make a mistake, it is easier to select and correct pins if the instances in *mux2* are unselectable. If you need to make instances unselectable during the following steps, click the button next to *Inst* (Instances) in the LSW so it is empty.

Creating Pins

You create six pins in the *mux2* cellview. The pins have different characteristics. This table summarizes the characteristics of the pins you create in the following exercise.

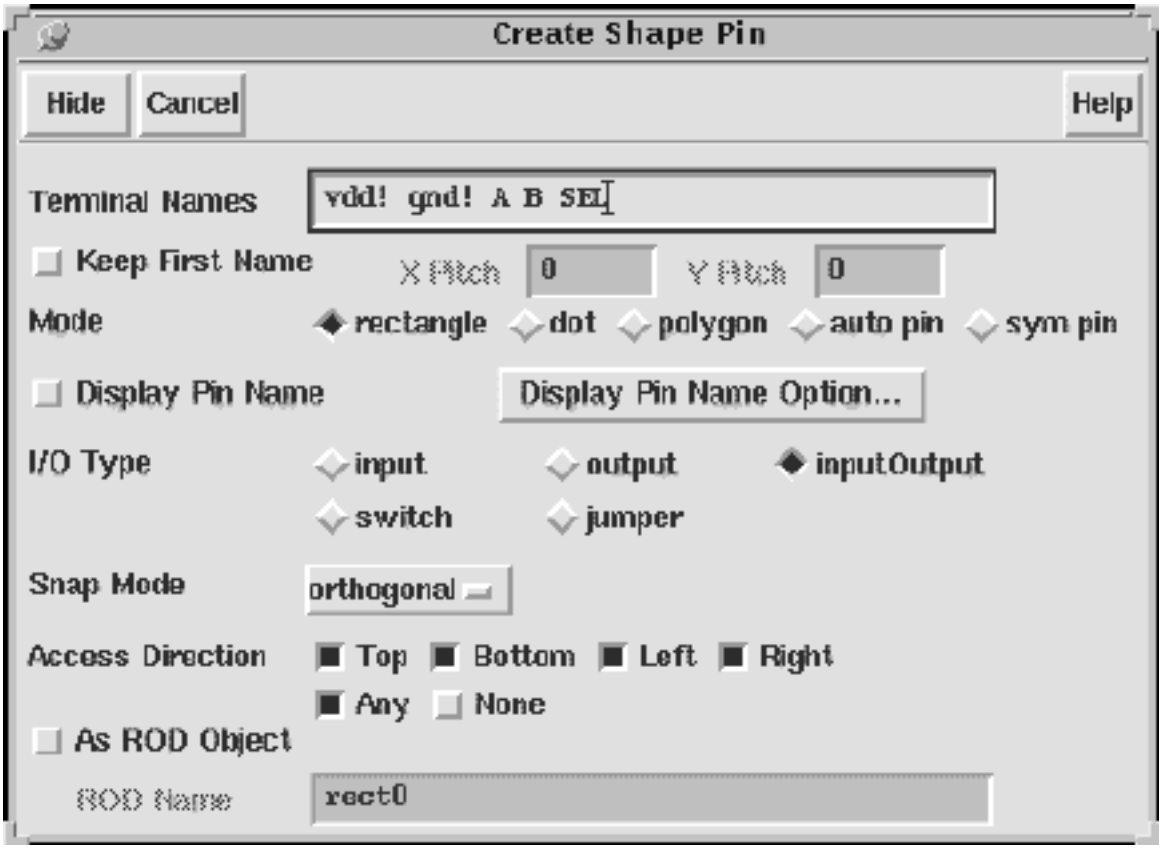
Pin	Input/Output Type	Access Direction	Layer
vdd!	I/O	top, left, right	metal1 dg
gnd!	I/O	bottom, left, right	metal1 dg
A	input	top, bottom	metal2 dg
B	input	bottom	metal2 dg
SEL	input	bottom	metal2 dg
Y	output	right	metal1 dg

1. In the LSW, click the *metal1 dg* layer.
2. Choose *Create – Pin*.

   The Create Symbolic Pin form appears. You use shape pins in this tutorial, not symbolic pins.

3. Click the button next to *shape pin* to open the Create Shape Pin form.

The Create Shape Pin form appears.



4. In the Create Shape Pin form, type the following in the *Terminal Names* field.

```
vdd! gnd! A B SEL Y
```

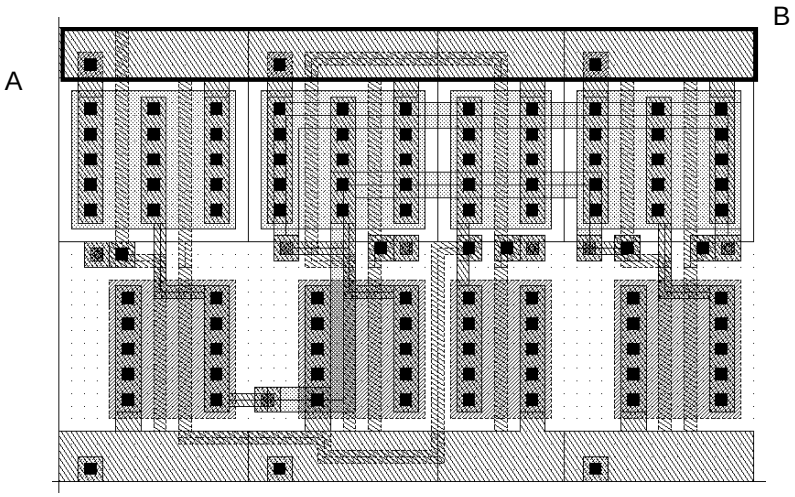
You can type any number of names in the Create Shape Pin (or Create Symbolic Pin) form. Each pin you create is assigned the next name, from left to right, in the *Terminal Names* field.

5. Click *Display Pin Name* to associate the name with the pin.

6. Set the access direction to *Top*, *Left*, and *Right* by clicking *Bottom* to turn it off.

7. Create the rectangle for the *vdd!* pin coincident with the power line at the top of the multiplexer.

- Start the *vdd!* pin at corner A.
- Finish the *vdd!* pin at corner B.



The name of the pin (*vdd!*) appears near the cursor after you click the second corner.

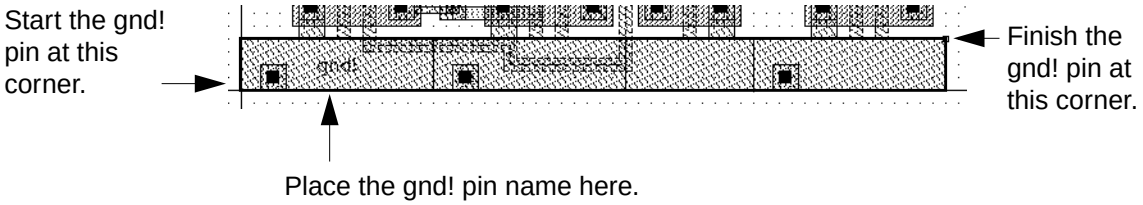
A dashed line extends from the first corner of the pin to the name, showing the pin name is attached to the pin. If you move or delete the pin, the attached label will also be moved or deleted.

8. Move the cursor so the *vdd!* text appears near the left side of the pin, then click.

The name *vdd!* disappears from the Create Shape Pin form. The first name listed is now *gnd!* (the ground pin).

9. In the Create Shape Pin form, turn off the *Top* access direction and turn on the *Bottom* access direction.

10. Create the rectangle for the *gnd!* pin coincident with the ground line at the bottom of the multiplexer.



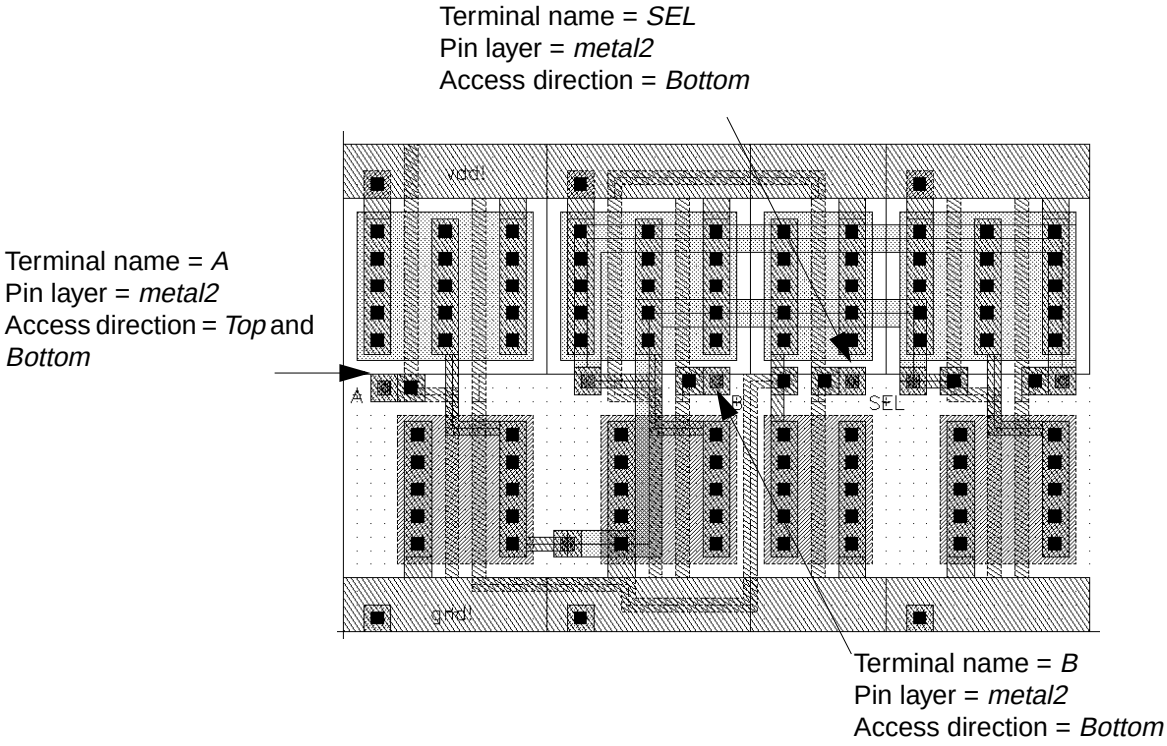
11. In the LSW, click *metal2 dg*.

12. In the Create Shape Pin form, set the *I/O Type* to *input*.

13. Type the following information in the Create Shape Pin form before you create each input pin for terminals *A*, *B*, and *SEL*.

For Pin. . .	Set the Access Direction to. . .
A	*Top* and *Bottom*
B	*Bottom*
SEL	*Bottom*

Use this illustration to determine the location for each pin.



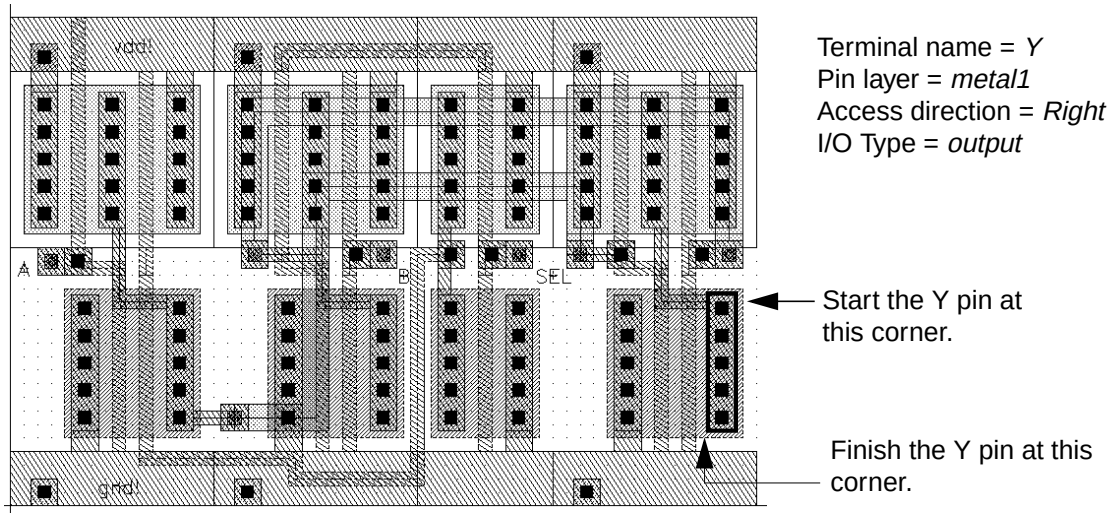
After you create the pins shown in the illustration, the pin form now shows only one pin to create: the Y pin for the multiplexer output.

14. Click the *metal1 dg* layer in the LSW.

15. Change the I/O type to *output* in the Create Shape Pin form.

16. Turn off all access directions except *Right*.

17. Create the Y output pin as shown below.



Place the Y pin name anywhere near the pin.

18. Press the Escape key to stop the *Create Pin* command.

Saving the Design

➤ Press the F2 key to save your design.

The multiplexer is saved to disk.

Closing the mux2 Schematic

Because your design is finished, you can close the *mux2* schematic.

➤ To close the *mux2* schematic, in the schematic window, choose *Window – Close*.

Creating a Guard Ring

In this section, you use Virtuoso® relative object design (ROD) functionality to create a guard ring around the *mux2* design using a multipart ROD object. ROD functionality is used for defining simple and complex layout objects and their relationships to each other.

For more information about ROD, see the *Virtuoso Relative Object Design User Guide*.

In this section, you

- Start layout accelerator
- Learn about multipart paths (MPP)
- Move the *mux2* design to make room for the guard ring
- Define values for the guard ring in the Create Multipart Path and ROD Subpart forms
- Save the values to a template in the technology file for future use
- Draw a guard ring around the multiplexer cell in the layout cellview

Starting Layout Accelerator

To start layout accelerator,

1. Choose *Tools - Layout XL* from the menu banner.

   The Define Connectivity Reference form appears.

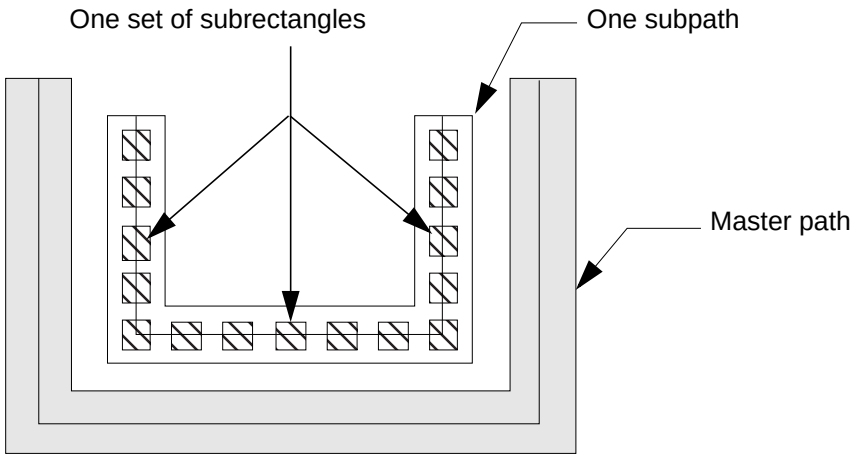
2. Click *Cancel* to close the form.

   The banner becomes Virtuoso® XL Layout Editing: tutorial mux2 layout.

About Multipart Paths

You create a multipart path as the guard ring for the *mux2*. A multipart path is a single ROD object consisting of one or more parts at level 0 in the hierarchy on the same or on different layers. A multipart path consists of a single *master path* and one or more *subparts*. The master path is an ordinary path; however, it is the defining part of a multipart path; all subparts are based on the master path.

For example, the multipart path shown below has one subpath and one set of subrectangles. Both the subpath and the set of subrectangles are offset from the master path



Moving the Design

Before you create the multipart path, you must make room for it. You move the *mux2* design up and to the right.

To move the design,

1. Choose *Edit – Select – Select All*.

   All the elements of the *mux2* are highlighted.

2. Choose *Edit – Move*.

   You are prompted in the CIW to point at the reference point for the move.

3. Click X = 0, Y = 0.

   You are prompted in the CIW to point at the new location for the move.

4. Click X = 7, Y = 9.5.

   The designs moves to the right.

5. Press f to fit the design in the window.

Creating the Multipart Path

The best way to create a multipart path is to enter values for the MPP in the Create Multipart Path form and save the values to a template for future use. Creating templates for multipart paths allows you to reuse the information to

- Create additional paths
- Edit the path by changing the template

The multipart path you create for this design has a master path, a enclosure subpath, and a set of subrectangles.

To create the template for this path,

1. Click the *diff* layer in the LSW.

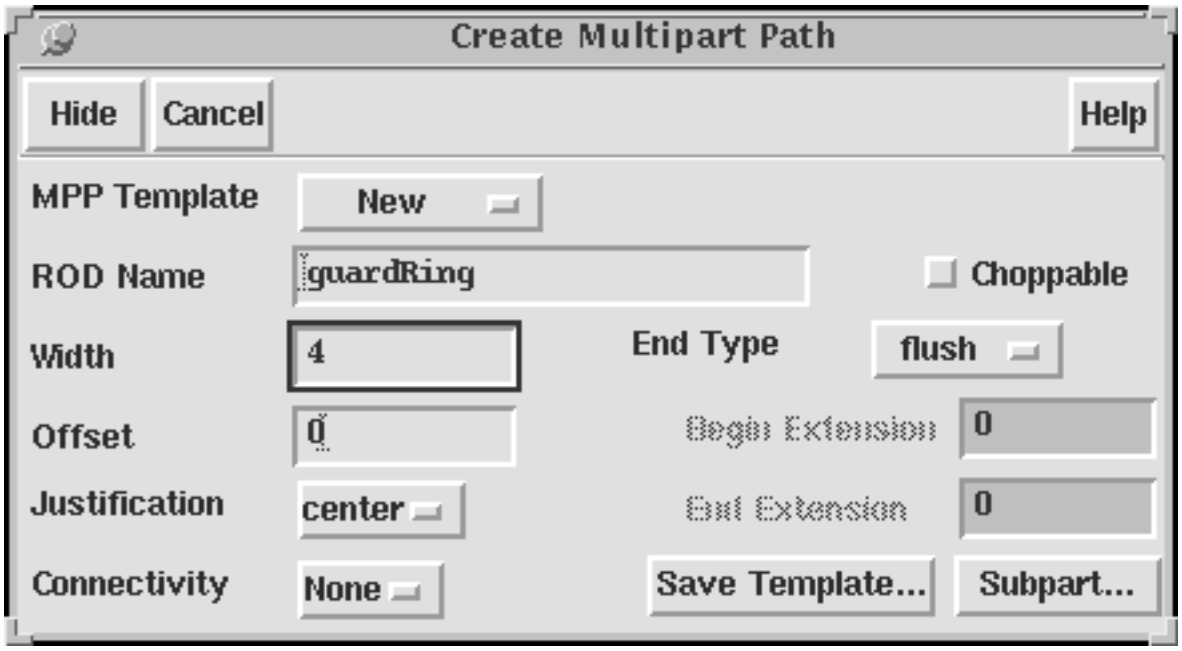
   The diffusion layer is the master path layer.

2. Choose *Create – Multipart Path.*

   The Create Multipart Path form appears.

3. Type these values in the form. The CIW displays warnings. These warnings do not affect your work.

MPP Template	*New*
ROD Name	`guardRing`
Choppable	off
Width	`4`
End Type	*flush*
Offset	`0`
Begin Extension	`0`
End Extension	`0`
Justification	*center*
Connectivity	*None*



You are ready to create the *metal1* layer as the enclosure subpath.

1. Click *Subpart* in the Create Multipart Path form.

The ROD Subpart form appears.

2. Click *Enclosure Subpath.*

   The enclosure subpath fields appear.

3. Enter these values in the form:

Layer	*metal1*
Choppable	on
Begin Offset	`-0.6`
Enclosure	`0.6`
End Offset	`-0.6`
Connectivity	*Pin*

   When you choose *Pin*, new fields appear. Enter these values in the new fields:

Net Name	`gnd!`
I/O Type	*inputOutput*
Access Direction	*Bottom, Left, Right*
Display Pin Name	*No*
Reference Handle	*centerCenter*
Offset X	`0`
Offset Y	`0`

4. Click *Add* to register this data as subpath parameters.

   The data for the enclosure subpath appears in the scroll window at the top of the ROD Subpart form.

5. Click *Apply* in the ROD Subpart form to add this data to the template.

   The ROD Subpart form remains open.

You are ready to create the contacts as subrectangles.

1. In the ROD Subpart form, click on *Subrectangle*.

   The subrectangle fields appear.

2. Enter these values in the form:

Layer	*cont*
Choppable	on
Begin Offset	`-1.2`
Width	`1`

   By setting *Begin Offset* and *Width*, the system assigns the default values to *End Offset*, *Length*, and *Space*.

3. Click *Add* to register this data as subrectangle parameters.

   The data for the subrectangles appears in the scroll window at the top of the ROD Subpart form.

4. Click *OK* in the ROD Subpart form to add this data to the template.

   The ROD Subpart form closes.

Why You Save Changes to a Template

Save your MPP form values as a template **before** you draw the MPP in your layout cellview. Once you enter the last point for the master path, you can no longer make any changes to the MPP.

However, if you save the values as a template, you can create a similar MPP by loading the template and changing the values as desired. When you load the template, all the fields display the template data **except** the *Net Name* field, which you must add every time you load a template. If you do not add the net name, the system beeps and a message appears in the CIW telling you to add the net name.

Saving Changes to the Template

To save all your MPP form values as a template,

1. In the Create Multipart Path form, click *Save Template*.
2. Type `guardRing` in the *Template Name* field,
3. Click *OK*.

Saving the Template to the Technology File

When you save an MPP template, the system updates the temporary version of your technology library in virtual memory. You still need to save your technology library changes to disk before you exit the software, or you will lose your template changes.

To make the changes to the temporary version of your technology file permanent,

1. In the CIW, choose *Technology File – Save*.

   The Save Technology File form appears.

2. Choose *cellTechLib* as the target technology file.

3. Click *OK*.

   A dialog box appears asking you to confirm saving the technology file to disk.

4. Click *Yes*.

You have completed saving the MPP values as a template in your binary technology library for future use. The Create Multipart Path form remains open. All the values you entered remain in this form until you close it. Now you are ready to draw a guard ring in your layout cellview.

Drawing the Guard Ring

With the multipart path parameters set and the template saved to the technology file, you are ready to draw the guard ring around the *mux2*.

In the cellview window, click at these points. For the last point, X=0, Y=50, either double click or press `Return`.

First click: X = 2, Y = 48

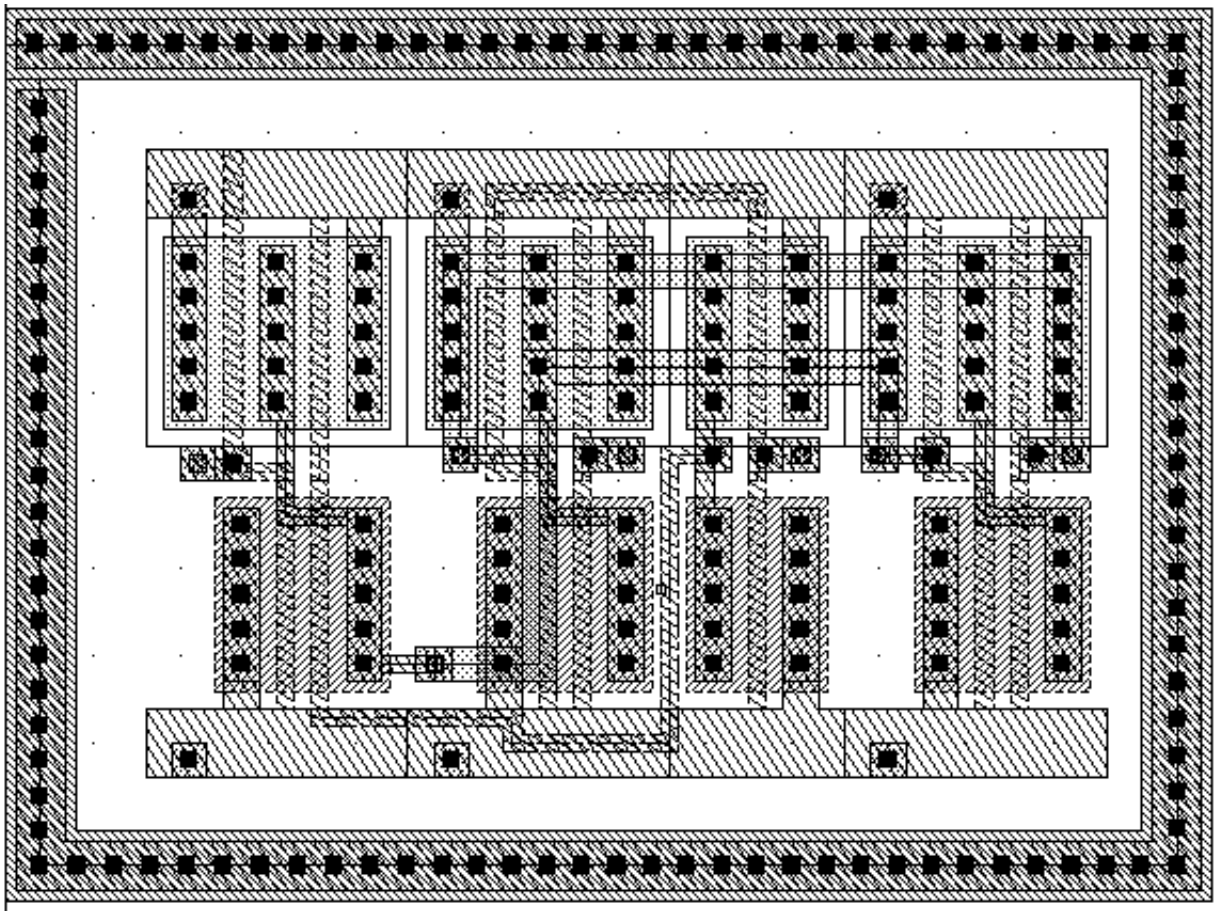
Second click: X = 2, Y = 3

Third click: X = 67, Y = 3

Fourth click: X = 67, Y = 50

Fifth click: X = 0, Y = 50

The completed design:



Editing the Guard Ring

In this section, you edit the guard ring by

- Making it choppable
- Stretching the right side to a new X location

To edit the guard ring,

1. Select the guard ring in your layout window.
2. Choose *Edit – Properties*.

   The Edit Properties form appears.
3. Click *Choppable*.

4. Click *Apply*.

   The master path is now choppable.

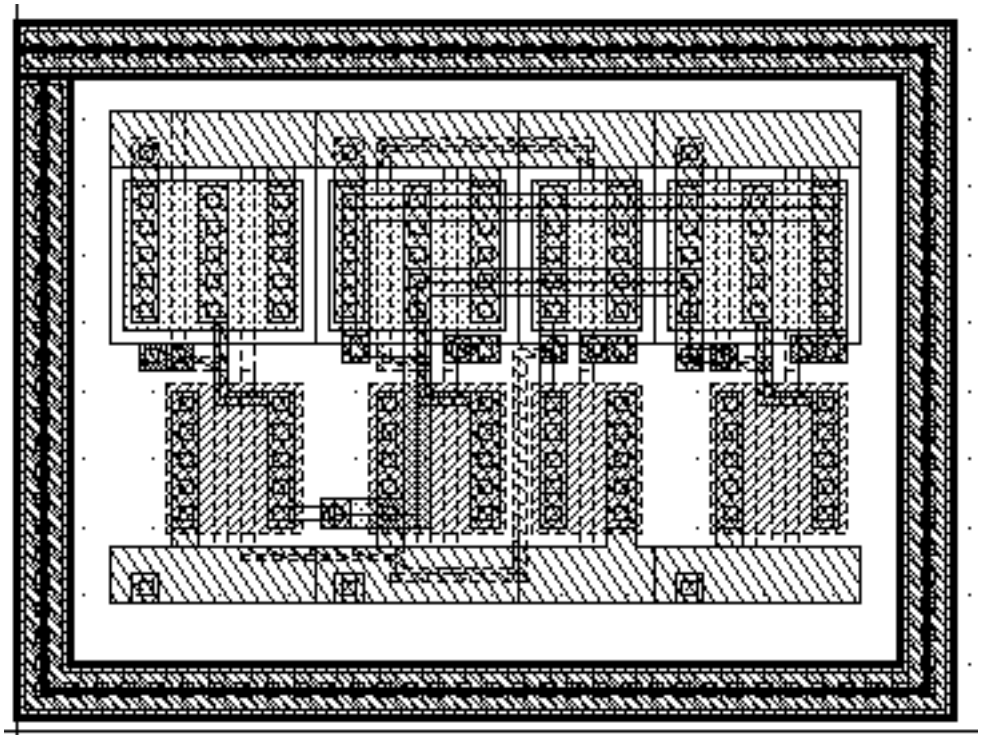
5. Deselect the guard ring by clicking in an empty part of the window.

6. Choose *Edit – Stretch*.

7. Click on any of the contacts on the right side of the guard ring.

   The entire master path highlights in yellow.

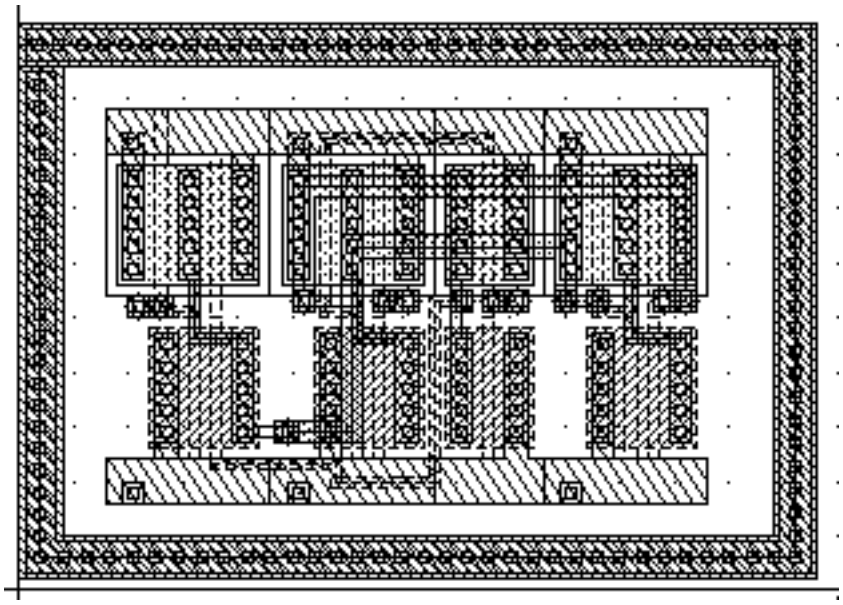
   You are going to change the points for the path by stretching the path. Look at the points displayed in the *Points* field. They should reflect the points you entered to draw the guard ring.



8. Click on X = 71 to stretch the path to the right.

9. Press the Escape key to end the stretch.

The guard ring stretches to the right.



10. Click on the guard ring to select it.

11. Examine the Edit Properties form for the revised point list. The third set of points should show (71 3).

12. Click OK to close the Edit Properties form.

13. Save and close your design.

Summary

In this section, you learned how to use the layout editor to create hierarchical designs. You also learned more about layout editor create and edit commands. Specifically, you

- Created a hierarchical layout
  - Created instances
  - Copied instances
  - Mirrored instances
- Used the *Edit in Place* command
  - Opened a lower-level cell for editing
  - Returned to the previous level

- Saved during *Edit in Place*

- Stretched an area
  - Selected an area
  - Used a reference point
- Displayed different hierarchy levels
  - Displayed a list of cells in the current cellview
  - Viewed instance outlines and master names only
  - Displayed a range of levels
  - Used bindkeys to change display levels
- Flattened the hierarchy in one cell
- Created paths using path stitching
  - Turned gravity off
  - Changed layers
  - Automatically placed contacts
- Created labels
- Created pins
- Created a multipart path
- Saved the multipart path to a template
- Edited the multipart path
- Saved the design
- Used bindkeys
  - Create Instance [`i`]
  - Fit All [`f`]
  - Pan [`Tab`]
  - Copy [c]
  - Display Levels 0–20 [`Shift-f`]
  - Display Levels 0–0 [`Control-f`]

- Gravity toggle [g]
- Save [F2]

- Used the icon menu
  - Copy
  - Create Instance

4

Verifying the Multiplexer Layout

This chapter introduces you to interactive verification. You will perform two different tests in the Virtuoso® layout editor while using Assura™ interactive verification products. One test uses the Design Rule Checker (DRC) to compare your design against the design rule, and the other test uses Layout Versus Schematic (LVS) software to check your design's connectivity. You will be

When you finish this chapter, you will be able to

- Run a design rule check and view errors
- View and correct DRC errors
- Run extraction on a layout
- View a schematic
- Cross-probe between a layout and a schematic
- Rerun verification after correcting an error

Finding Out if You Can Run Interactive Verification

You might not have a license to run the interactive verification products.

➤ Click the *Verify* menu to find out whether you can use interactive verification.

If the commands under *Verify* appear shaded, you do not have a license to run interactive verification. You can either read this chapter to get an idea about how interactive verification works, or you can go on to the next chapter.

If You Have Not Completed the Previous Chapters

This chapter assumes you have followed the steps in the previous chapters. If you have, you can skip this section and go to the "Creating a Test Case for Checking Errors" on page 104. If you did not follow the steps in the previous chapters, you must copy a completed design from the `master` library so you can go through this chapter. The following steps show you how to copy the completed design from the `master` library.

It is possible to run out of resources, such as memory, if you run multiple layout editors. Before you start the software, you need to check whether the software is already running.

1. Type the following in an xterm window to check whether the layout editor is already running:

   ```
   ps auxw | grep layout
   ```

2. If the layout editor is running, choose *File – Exit* in the Command Interpreter Window (CIW) to exit the software.

3. Type the following in an xterm window to start the layout editor:

   ```
   cd ~/cell_design
   layoutPlus &
   ```

4. Choose *File – Open.*

5. Type the library, cell, and view names as follows:

Library Name	`master`
Cell Name	`mux2`
View Name	`layout`

6. Click *OK*.

   The *mux2* cell from the `master` library opens.

7. In the cellview window, choose *Design – Save As*.

   The Save As form appears.

8. In the Save As form, type the library and cell names as follows:

Library Name	`tutorial`
Cell Name	`mux2`

9. Click *OK*.

   The *mux2* cell is copied to the `tutorial` library.

10. In the *mux2* cellview, choose *Window – Close* to close the cellview.

11. In the CIW, choose *Open – File* to open the *mux2* layout you just saved.

12. Type the library, cell, and view names as follows:

Library Name	`tutorial`
Cell Name	`mux2`
View Name	`layout`

13. Click *OK*.

    The *mux2* cell from the `tutorial` library opens.

    Note: Another way to open a cellview is with the *Open* command. Using *Open* replaces the current window with the new window. To use the *Open* command, choose *Design – Open*. The Open File form appears. Set the library, cell, and view names to the cellview you want to open, and click *OK*. The current cellview is replaced with the new cellview.

Creating a Test Case for Checking Errors

If you followed the instructions in the last chapter exactly or copied the *mux2 layout* from the `master` library, the multiplexer design does not generate any verification errors. In this section, you will make a small, deliberate error on the *metal1* layer so you can learn how to display and correct errors.

In this section, you learn to

- Turn off visibility of all layers except *metal1* so it is easier to see the path you edit
- Make an error by stretching the end of a path

- Turn visibility of all layers back on

Displaying Only the metal1 Layer

Open your *mux2* layout if you closed it after the previous chapter. To make it easier to see the path you want to edit, you turn off visibility of all layers except *metal1.*

1. In the Layer Selection Window (LSW), press `Shift` and click middle on the *metal1 dg* layer.

   The *metal1* entry layer is now the current entry layer. The layer names in the LSW are all shaded gray to show they are invisible, with the exception of *metal1*.

   The *mux2* layout window does not change. You must redraw the window to see any changes you make in the LSW.

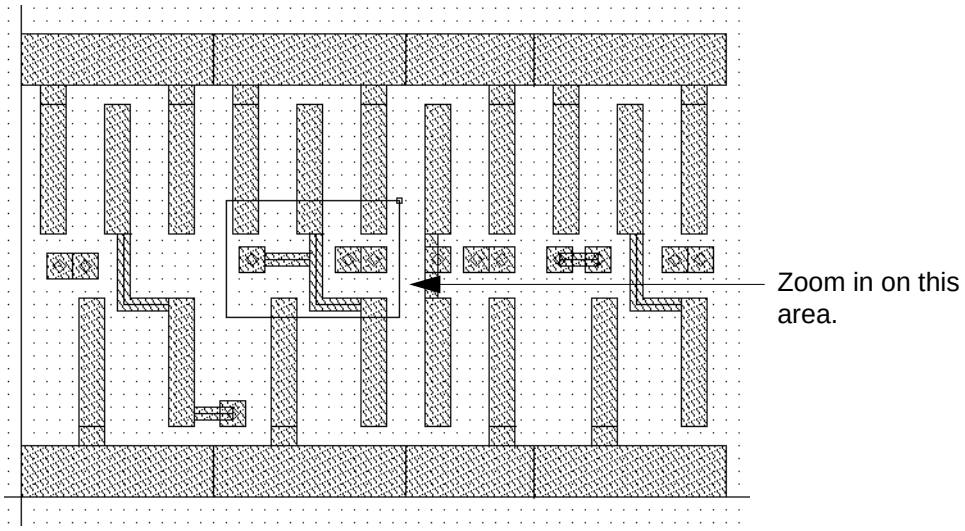
2. Move the cursor to the layout window and press `Control-r`.

   Now you see only *metal1* objects in the layout window.

Stretching a Path

In this section, you learn how to stretch a path.

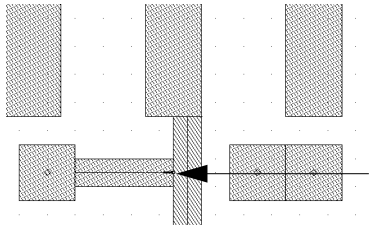
1. Zoom in on the area shown below.



2. Press the `s` key to open the Stretch form.

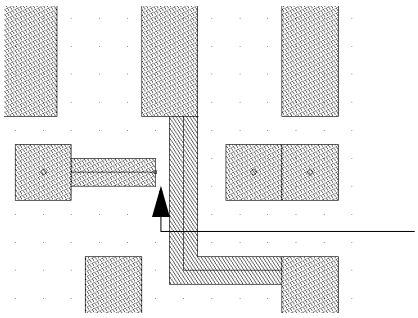
3. Click the right end of the *metal1* path.

   The entire path is highlighted. Bolder highlighting appears at the endpoint. The bold mark shows you selected the end of the path.



4. Stretch the path so there is a 0.5-micron gap by clicking X = 32, Y = 28.

   The display grid points are each 1 micron apart. The cursor snaps to the grid every 0.5 microns. The gap you create is one-half of one visible grid space.



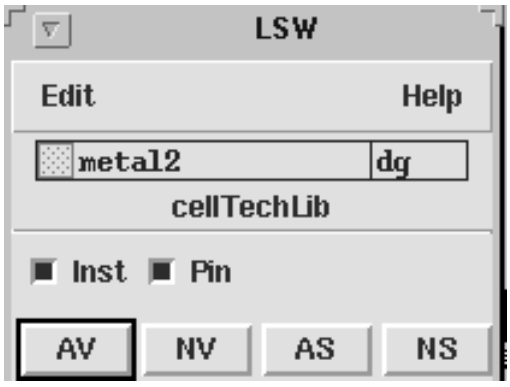
5. Press the Escape key to stop the *Stretch* command.

You just created an error by stretching the path. Later, you use DRC to find this error.

Redisplaying All Layers

It is not necessary to view all the layers when you run DRC. However, errors are easier to see with all layers visible.

1. In the LSW, click *AV* (All Visible).



2. Move the cursor into the layout window and press `Control-r` to see all layers.

Performing a Design Rule Check

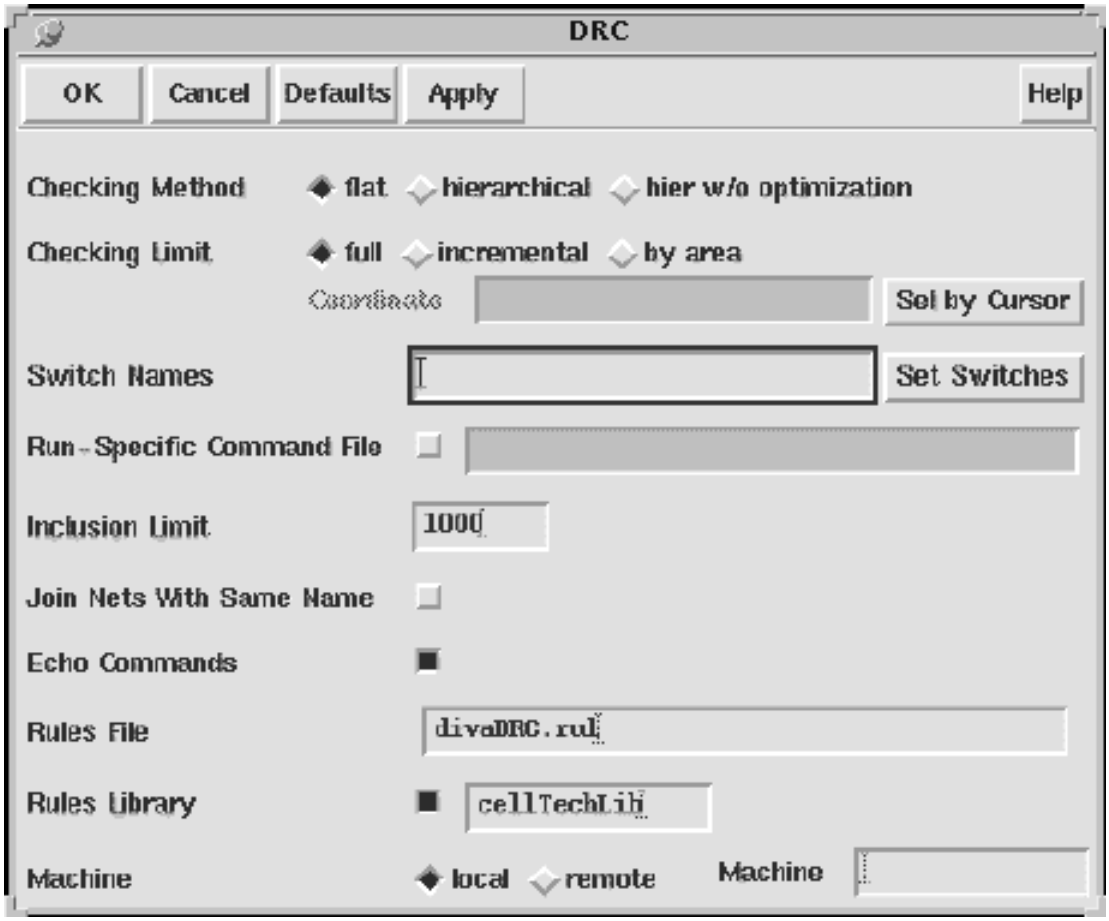
DRC checks your layout against physical design rules defined in the `divaDRC.rul` file located in the `cellTechLib` directory. This section shows you how to

- Run DRC to search for errors
- Display information about any errors

Running DRC

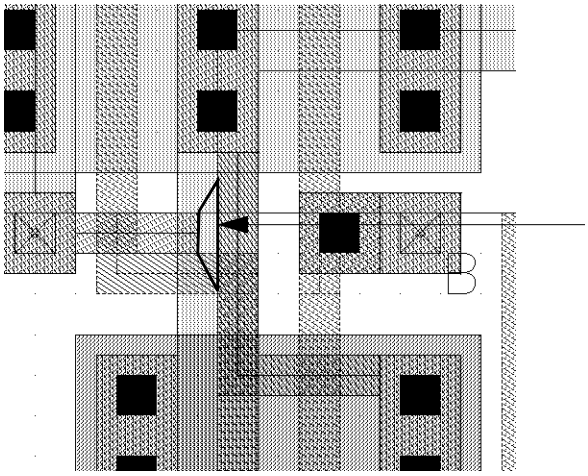
1. Choose *Verify – DRC*.

The DRC form appears.



2. Click *OK* to run DRC.

The CIW reports one error. A blinking polygon, called an error flag, appears at the location of the error.



Important

Do not correct this error yet. You will run LVS later in this chapter to see how LVS reports this same error.

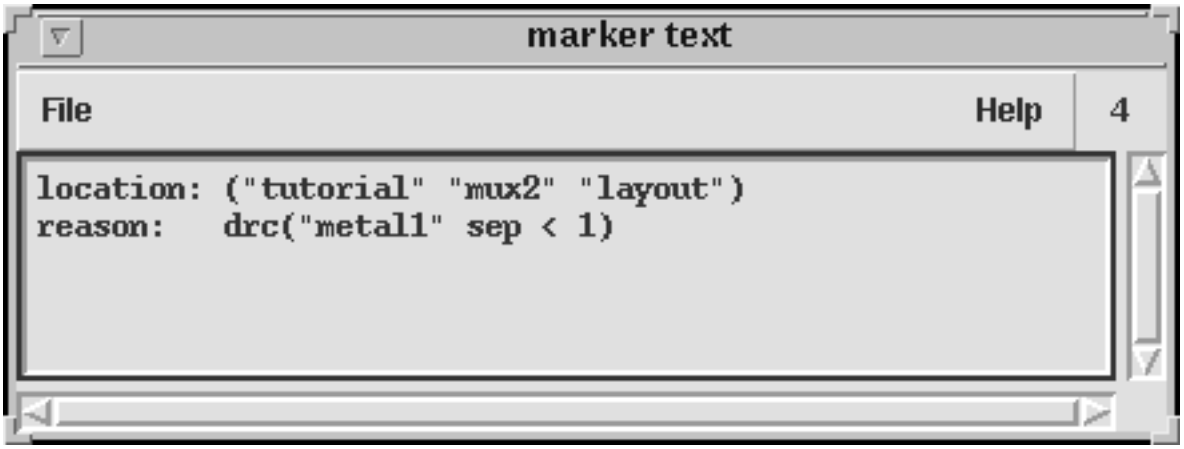
3. Press the `f` key to fit the entire design in your window, and look for any other errors you might have made.
4. If you have any other errors, correct them by redrawing the flagged objects using instructions in the previous chapters. Run DRC again before proceeding.

Viewing Errors

Use the *Markers – Explain* command to display more information about the error flagged by DRC.

1. In the *mux2* cellview window, choose *Verify – Markers – Explain*.
2. Click the error flag.

The error flag is highlighted in yellow to show that you selected it. A window appears at the top left of the screen. It lists the cellview containing the error and the rule that was violated.



In the CIW, DRC reports a spacing violation:

```
"metal1" sep < 1 (metal1 separation is less than 1 micron).
```

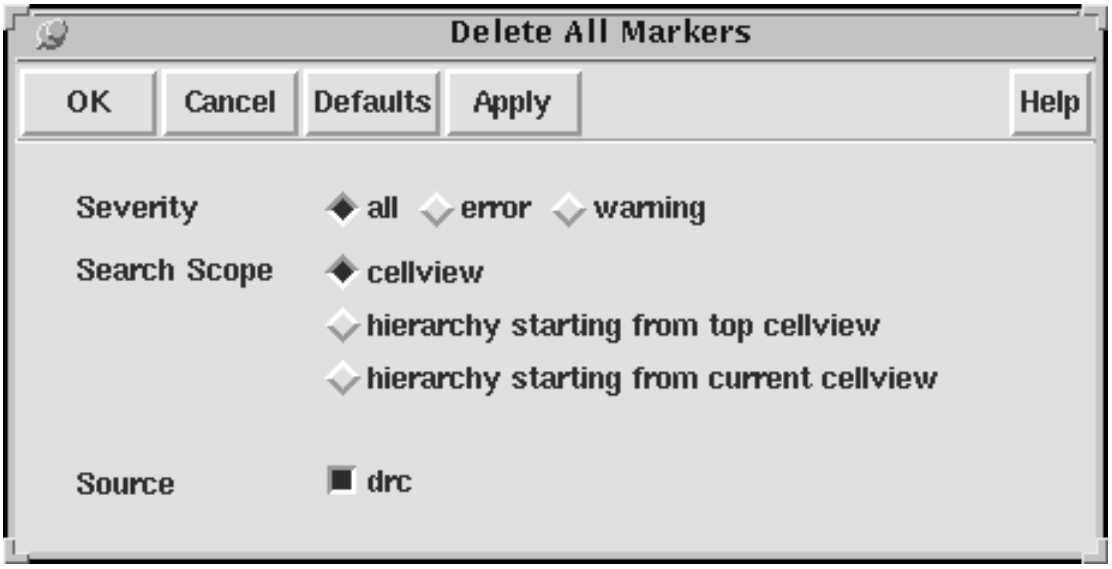
Even though the two paths should be connected, DRC reports a spacing violation because the spacing between objects on the *metal1* layer should be 1.0 microns and the space between the two paths on *metal1* is 0.5 microns.

3. Press the `Escape` key to cancel the *Explain* command.

If there were more error flags, you could continue to use *Explain* to explain the other errors.

4. Choose *Verify – Markers – Delete All* to remove the error marker.

The Delete All Markers form appears.



5. Click *OK*.

   The error marker is removed.

Extracting Connectivity from the Layout

You must extract the connectivity from the layout cellview to compare the layout and schematic cellviews. To extract connectivity, you run the Extract program. The Extract program uses rules defined in the technology file to recognize devices and establish electrical connections (nets).

The Extract program creates a temporary cellview, called the extracted view, that shows the nets. You will use both the extracted cellview and the layout cellview in this section.

Important

As you follow the steps in the rest of this chapter, be careful to use the correct cellview. Check the title banner for the view name *layout* or *extracted*.

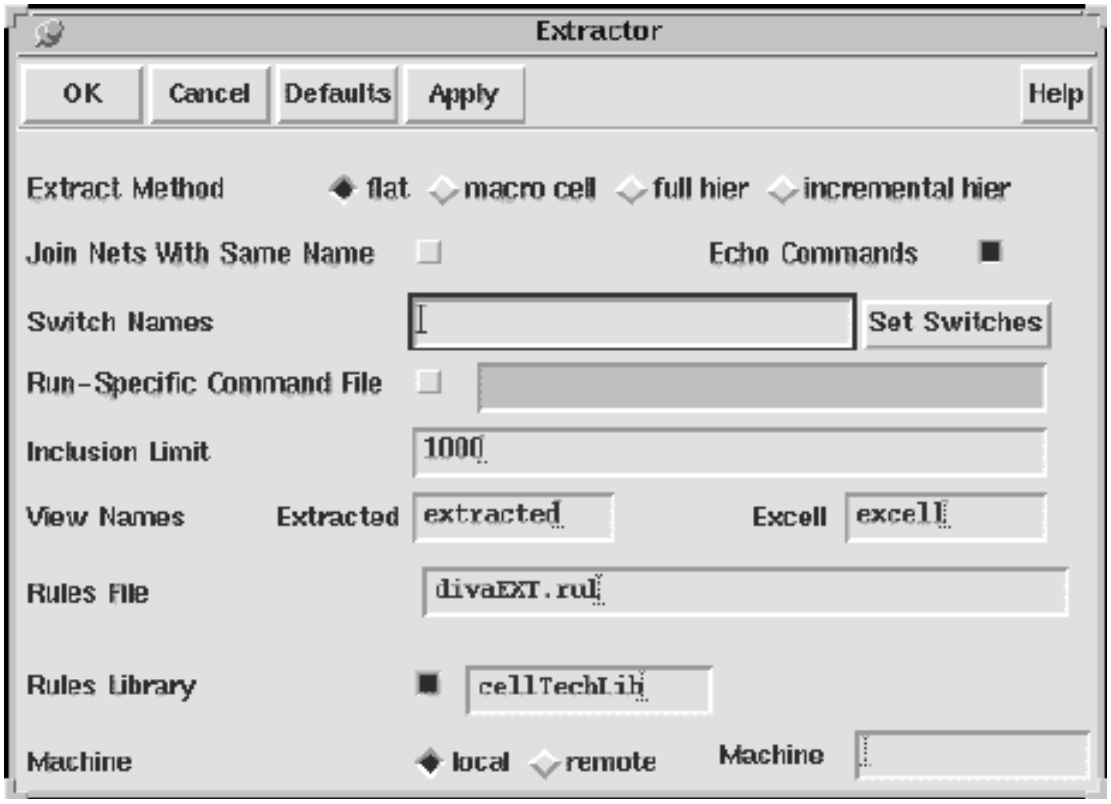
In this section, you learn to

- Use the *Extract* command to create an extracted view of *mux2*
- View the extracted data

Extracting the Layout

1. Choose *Verify – Extract*.

   The Extractor form appears.



   Note: If you are running the Affirma® anolog circuit design software, the Extractor form is different than the form that appears here. You can continue this tutorial despite the different form. If you want detailed descriptions of options that appear in the analog circuit design forms, refer to the *Assura Diva Verification Reference* manual.

2. Turn on *Join Nets With Same Name*. This will merge nets with the same names while suppressing warning messages about different nets that have the same name.

3. Click *OK* to run the Extract program.

   The extraction rules appear in the CIW as the extract program runs. When extraction is complete, you see

   ```
   saving rep tutorial/mux2/extracted
   ```

   This means the extracted cellview was created.

Viewing Extracted Data

The extracted view of *mux2* is similar to but not identical to the layout cellview. In this section, you look at the extracted cellview so you understand the differences between the extracted and the layout cellviews.

1. In the Command Interperter Window (CIW), choose *File – Open* to view the extracted *mux2* view.

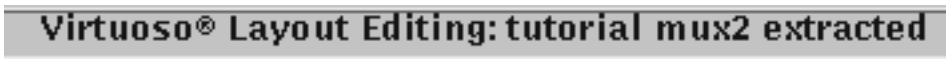
   The Open File form appears.

2. Type the library, cell, and view names as follows:

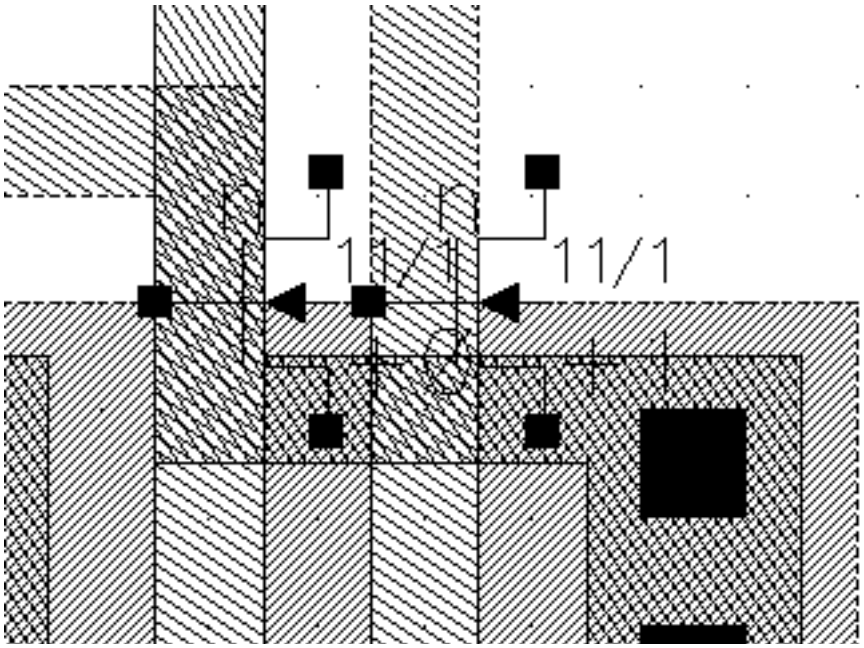
Library Name	`tutorial`
Cell Name	`mux2`
View Name	`extracted`

3. Click *OK*.

   The extracted cellview appears on top of the layout cellview. The banner contains the following:

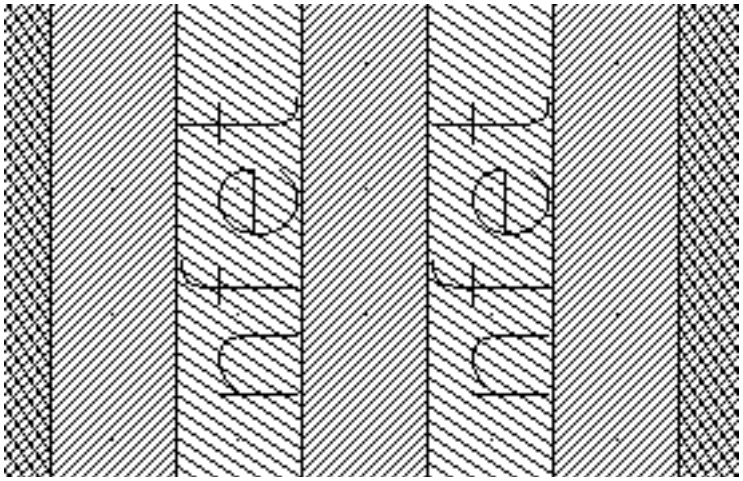


The extracted cellview is similar to the layout, but the gates now have symbols at one end.



4. Press `Control-f` to display only level 0 data.

   The gate symbols disappear, and you see a name inside each gate region. Each gate has been mapped to either an *nfet* or *pfet* device, identified by an instance of an *ivpcell*.

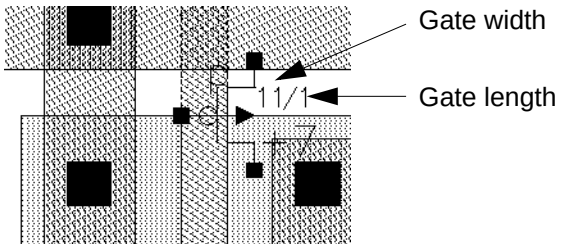


   An *ivpcell* is a special parameterized cell used by the verification program to display devices.

5. Press `Shift-f` to display all levels again.

6. Zoom in on one of the symbols.

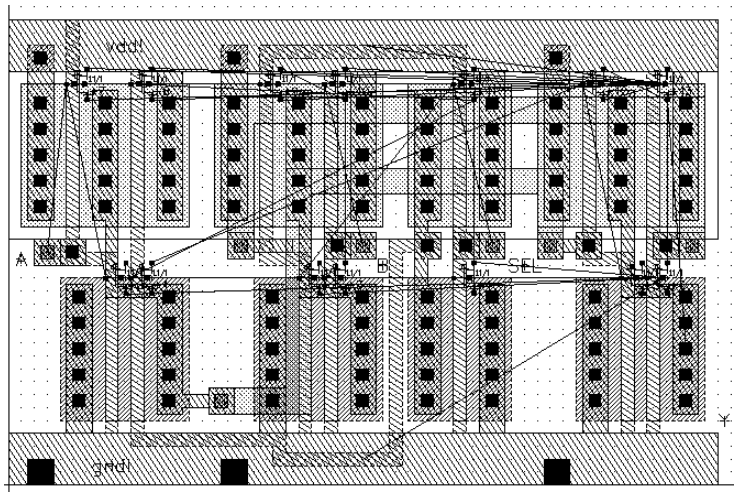
   You see the gate width and length displayed next to the symbol.



   Display the electrical connections by setting *Nets* on in the Display Options form.

7. Press the e key to open the Display Options form.
8. Select *Nets*.
9. Click *Apply*.
10. Move the cursor into the extracted view and press the f key to fit the design in the window again.

    You see the electrical connections in the extracted cellview.



11. In the Display Options form, turn *Nets* off.
12. Click *OK* to close the Display Options form.

Comparing the Layout to the Schematic

The LVS program lets you compare the schematic to the physical layout so you can check for connectivity errors. LVS uses both the extracted cellview you created in the previous section and the schematic view of the multiplexer. This tutorial provides a schematic cellview for you.

In this section, you learn to

- Display the schematic cellview
- Run LVS

Displaying the Schematic View

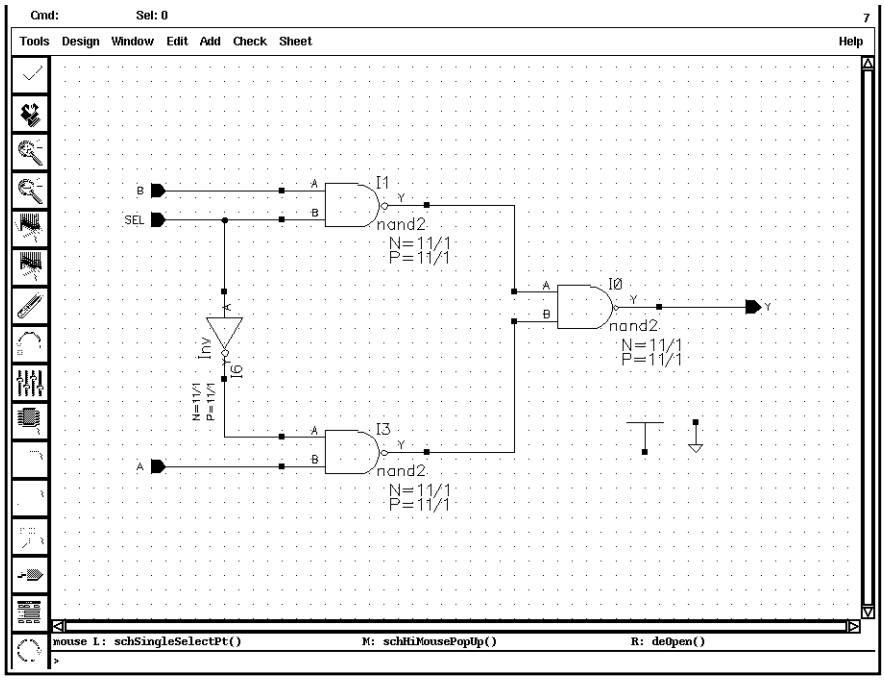
You will use the schematic for checking details between the schematic and layout after you run LVS. For now, you just need to be sure the schematic exists. You display the schematic to remind you of what LVS is using to check the design.

1. Choose *File – Open* to view the *mux2* schematic.
2. Type the library, cell, and view names as follows:

Library Name	`master`
Cell Name	`mux2`
View Name	`schematic`

3. Click *OK*.

The schematic cellview appears.

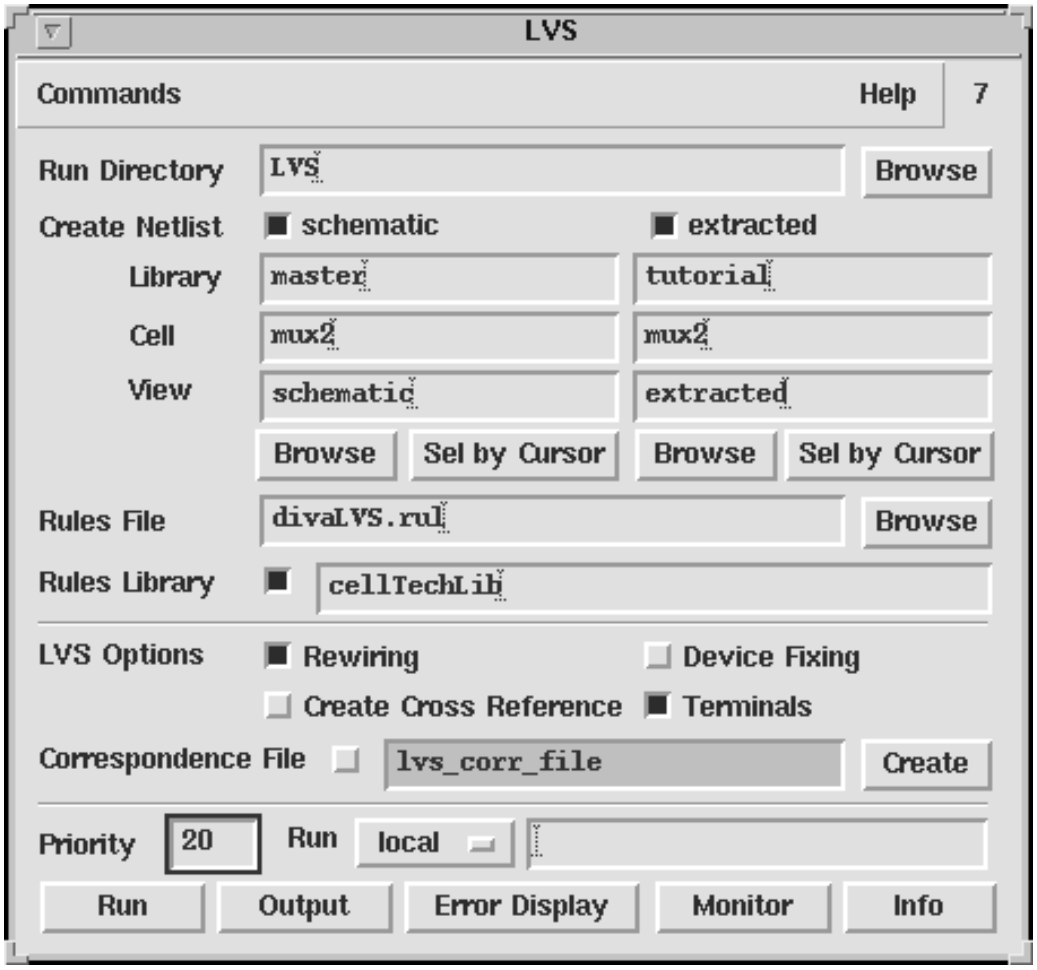


Note: To fit your windows on your screen, click and hold on any corner of the schematic window and drag the mouse until the window is a smaller size. Then press the f key in the schematic window to fit the schematic drawing within the resized window.

Running LVS

1. In the *mux2* extracted cellview, choose *Verify – LVS*.

   The LVS form appears.



   Note: If you are running the Affirma analog circuit design software, the LVS form is slightly different than the form that appears here. You can continue this tutorial despite the different form. If you want detailed descriptions of options that appear in analog circuit design forms, refer to the *Assura Diva Verification Reference* manual.

2. Fill in the schematic fields in the LVS form by clicking the *Sel by Cursor* button under the schematic fields, then click left in any area of the schematic cellview window.

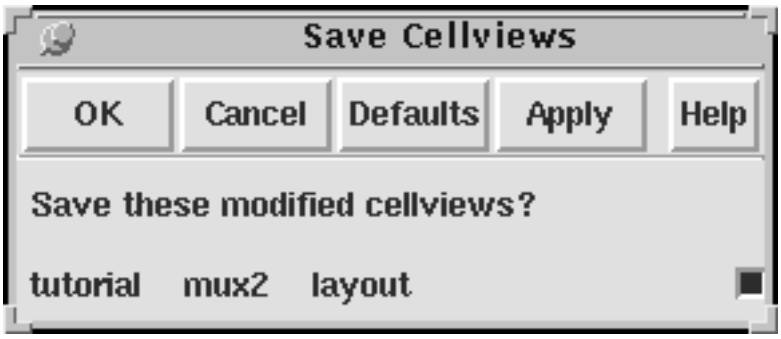
   The schematic fields are filled in with `master`, `mux2`, and `schematic`.

3. Set the *Priority* field to `20`.

   The default is *0* but that setting slows down other actions on the system.

4. Click *Run* to start the LVS job.

   The Save Cellviews form appears, asking if you want to save the *mux2* layout cellview.



5. Click *OK* to save the *mux2* layout.

   The LVS job runs in the background and might take a couple of minutes to complete. When the job is finished, you see a dialog box, entitled Analysis Job Succeeded, telling you the job succeeded.

6. In the dialog box, click *OK*.

Note: If the dialog box says your job did not get completed, click *Info* in the LVS form and look at the log. The log tells you what caused the job to be terminated and when.

Analyzing LVS Errors

Now that you have run LVS, you can display information about the comparison between the schematic and the layout. Because you deliberately added a small error to the layout, LVS will report the discrepancy.

You can use the probe commands on the *Verify* menu to highlight any nets, including nets that LVS lists as having errors. You can perform either a single probe to highlight a net in the extracted cellview or a cross-probe to highlight a net in both the extracted and the schematic cellviews.

In this section, you learn to

- Display the LVS report
- Display the errors LVS found
- Probe and cross-probe between the schematic and extracted cellviews

Displaying an LVS Report

1. In the LVS form, click *Output*.

   A text window listing the output from the LVS run appears.

2. Scroll until you see the section that compares the layout and schematic.

   LVS reports this information:

   ```
   The net-lists failed to match.
   ```

   You see LVS found 13 nets in the layout but only 12 in the schematic. It reports a net in the layout should be merged because the layout and schematic would match if two separate nets in the layout were connected (merged). Because the error you created was a disconnection within a net, this suggestion makes sense.

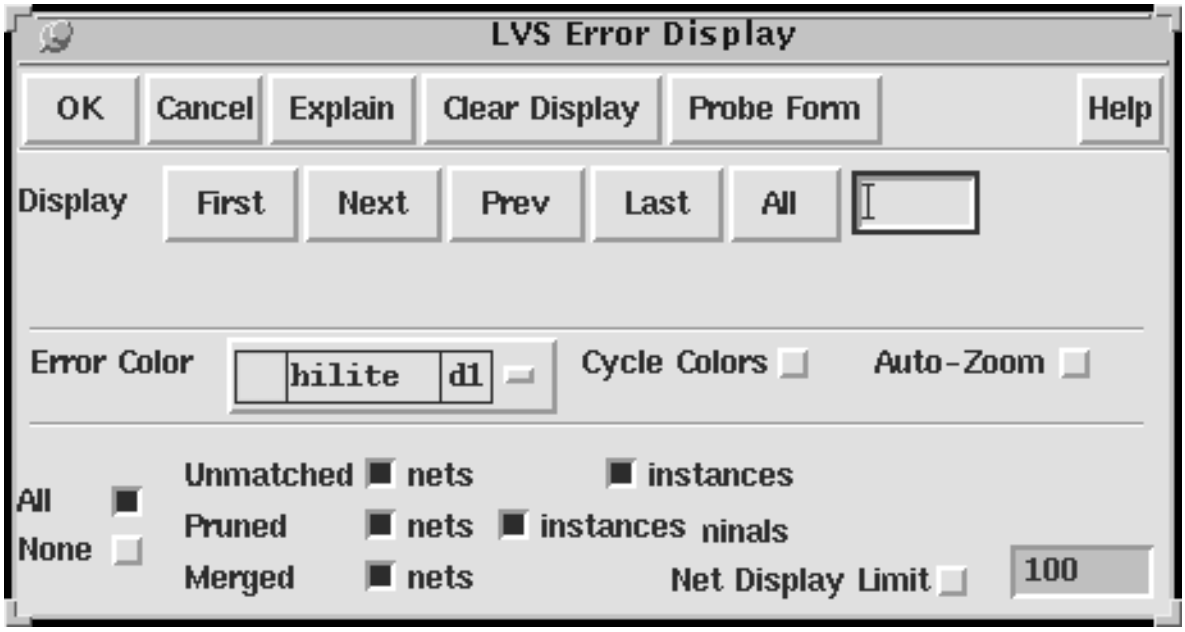
```
The net-lists failed to match.

                          layout  schematic
                            instances
        un-matched            0       0
        rewired               0       0
        size errors           0       0
        pruned                0       0
        active               14      14
        total                14      14

                              nets
        un-matched            0       0
        merged                1       0
        pruned                0       0
        active               13      12
        total                13      12
```

3. In the report window, choose *File – Close Window.*

Displaying the Errors

1. At the bottom of the LVS form, click *Error Display.*

The LVS Error Display form appears.

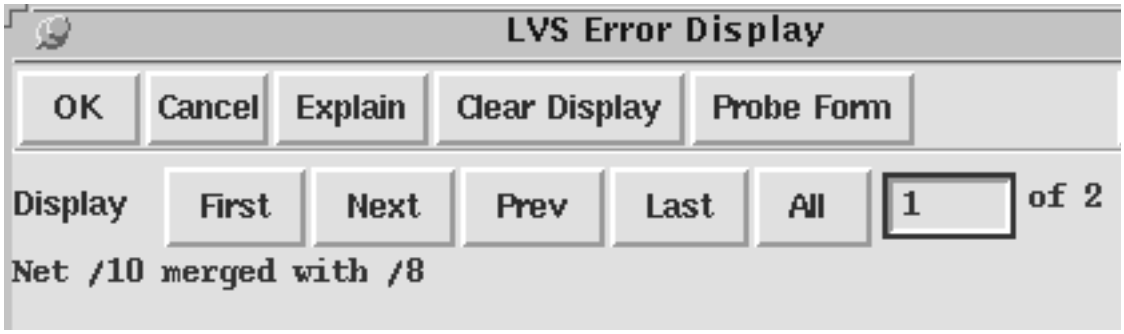


2. Move the cursor into the extracted window and press the `Escape` key.

This makes the extracted window the current window. LVS displays the errors in the current window.

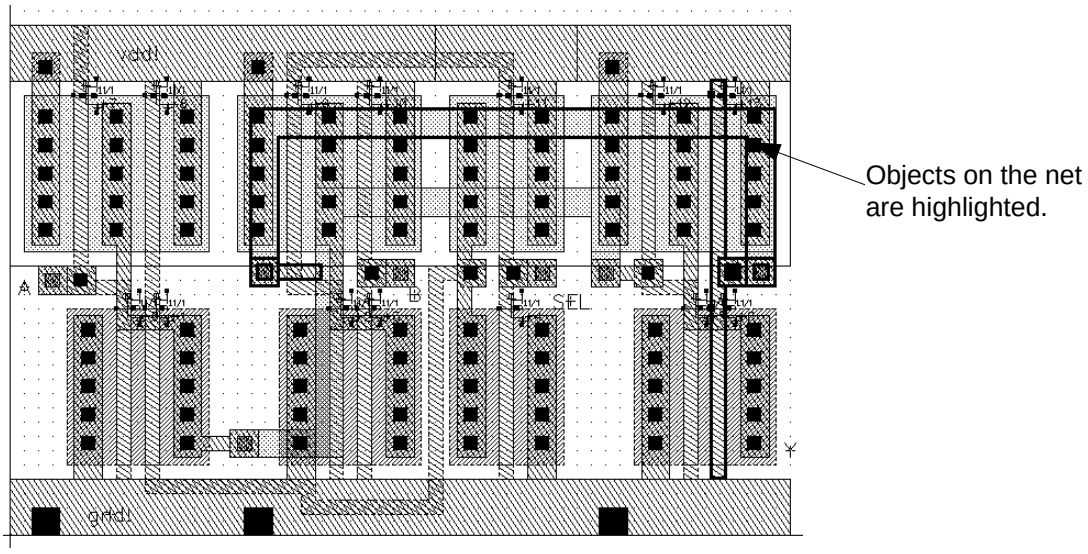
3. In the LVS Error Display form, click *First* in the *Display* field.

The LVS Error Display form displays a message indicating that two of the nets should be merged.



Note: LVS assigns numbers to the unlabeled nets. The numbers it assigns to your nets might not be identical to the net numbers shown above. Substitute the numbers on your LVS Error Display form in the following instructions.

In addition to the LVS Error Display form showing the nets to be merged, the geometries in the extracted layout that do not match anything in the schematic are highlighted in yellow. In this case, LVS highlights the objects on the part of the net you disconnected.



4. In the LVS Error Display form, click *Clear Display*.

Probing the Schematic and Layout

To look at the nets LVS suggests you merge, you probe the schematic and extracted cellviews to highlight the nets. This section shows you how to

- Perform a single probe to highlight a net in the extracted cellview

  You probe only the extracted view for net 10, which LVS found in the layout but not in the schematic.

- Perform a cross-probe to highlight a net in both the schematic and extracted cellviews

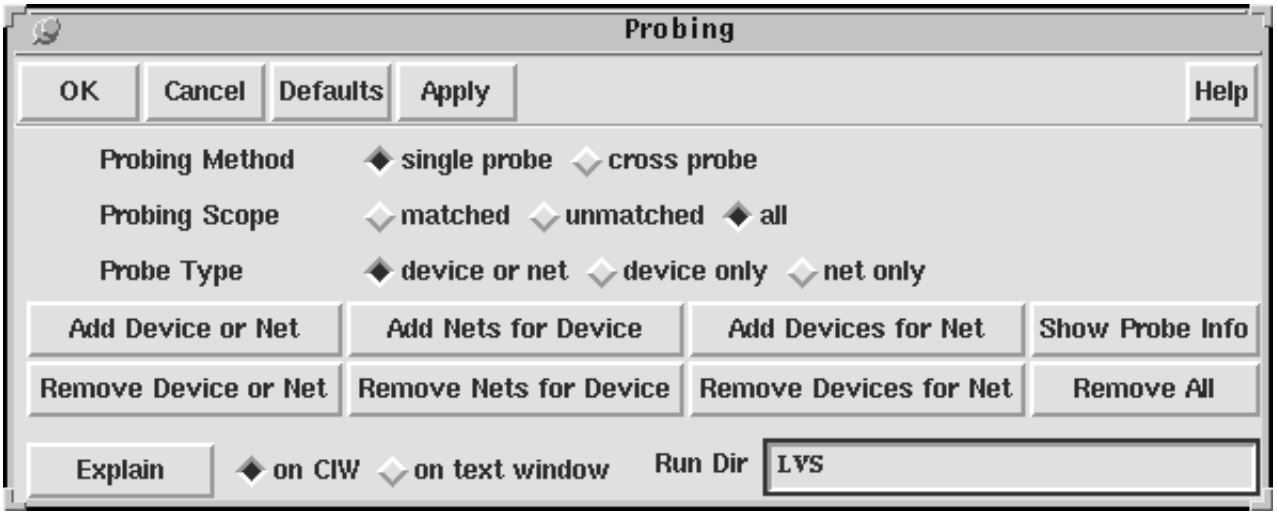
  You probe both views for net 8, which LVS found in both the layout and the schematic.

1. In the LVS Error Display form, click *Probe Form*.

   The Probing form appears.

   Note: If you are running analog circuit design software, the Probing form is different than the form that appears here. You can continue this tutorial despite the different form. If you want detailed descriptions of options that appear in analog circuit design software forms, refer to the *Assura Diva Verification Reference* manual.

2. Click *Add Device or Net*. If you are running analog circuit design software, click *Add Net*.



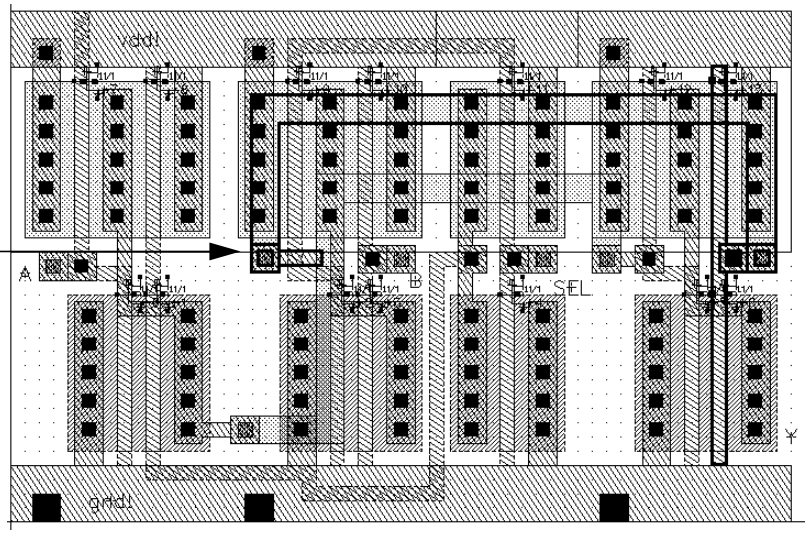
   By default, the form is set to perform a single probe. You first probe the extracted view for net 10.

3. In the CIW, type the net name, `"10"`, (type the quotation marks) and press `Return`.

Note: Several warning messages appear in the CIW, these do not have any effect on your probe.

   The shapes in net 10 are highlighted in yellow.

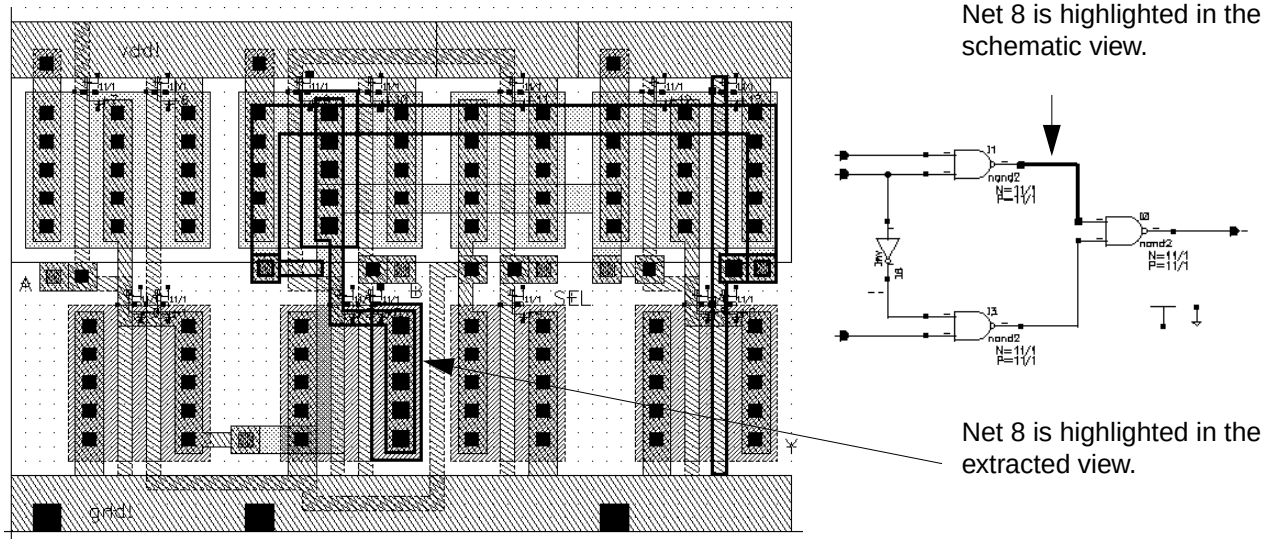
Net 10 is highlighted in the extracted view.



4. In the Probing form, change *Probing Method* to *cross probe*. If you are running Analog Artist, change the Probing Method to *cross probe matched*.

5. Click *Add Device or Net*.
6. Move the schematic cellview to the front of your screen so you can see its contents.
7. In the CIW, type the net name, `"8"`, (type the quotation marks) and press `Return`.

   The shapes in net 8 are highlighted in yellow in both the extracted and schematic views.



8. To remove the probe highlights, in the Probing form, click *Remove All*.

   Note: You can also probe from the schematic to the layout. After you open the Probing form, click a net in the schematic.
9. In the Probing form, click *Cancel*.
10. In the LVS Error Display form, click *Cancel*.
11. In the LVS form, choose *Commands – Close Window*.

    Now that you have determined where the error is, you do not need to see the schematic view anymore.
12. In the schematic cellview, choose *Window – Close*.

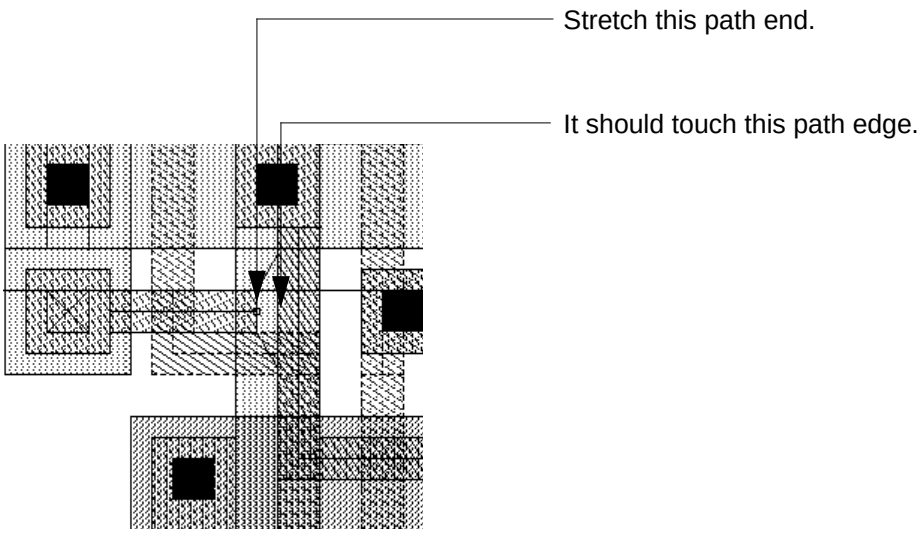
Correcting the Error

In the previous sections, you saw that the error both DRC and LVS discovered was caused by a break in one net in the layout. The break made it appear as if there were two nets. In this section, you correct the error and reconnect the net.

1. In the extracted cellview, choose *Window – Close*.

   The extracted cellview closes and you see the layout cellview. You always edit in the layout cellview and then reextract.

2. To correct the error in the layout cellview, press the s key and stretch the *metal1* path so it joins the two nets at point X = 33, Y = 28.



3. To stop the *Stretch* command, press the Escape key.

4. To save the layout cellview, in the icon menu, click the *Save* command icon.

   The layout cellview is written to disk.

Rerunning Verification

After correcting the errors in the layout, you run verification again. The steps are nearly identical to those you followed earlier in this chapter, except this time you run an incremental DRC. This means you check only the changed portion of the design. The verification programs should not find any errors.

This section tells you how to

- Run an incremental DRC

- Run an extraction on a layout
- Run LVS from the extracted cellview

The instructions in this section are brief because you have already done the steps before. If you want more details, you can go back through the previous sections.

Running an Incremental DRC

The system keeps track of any changes you made since the last DRC. You can run an incremental DRC to check only your changes to the design. This makes the DRC go faster.

1. In the *mux2* layout window, choose *Verify – DRC* to display the DRC form.
2. Set *Checking Limit* to *incremental*.
3. Click *OK* to run DRC.

   When DRC has been completed, you see output in the CIW that there are 0 errors.

Reextracting the Layout

You must extract the layout again so the extracted view includes your correction.

1. In the *mux2* layout window, choose *Verify – Extract* to display the Extractor form.
2. In the Extractor form, click *OK*.

   Extraction is complete when you see this message in the CIW:

```
saving rep tutorial/mux2/extracted
******  Summary of rule violation for cell "mux2 layout" *******
    Total errors found: 0
```

3. In the CIW, choose *File – Open.*

   The Open File form appears.

4. Type the library, cell, and view names as follows:

Library Name	`tutorial`
Cell Name	`mux2`
View Name	`extracted`

5. Click *OK*.

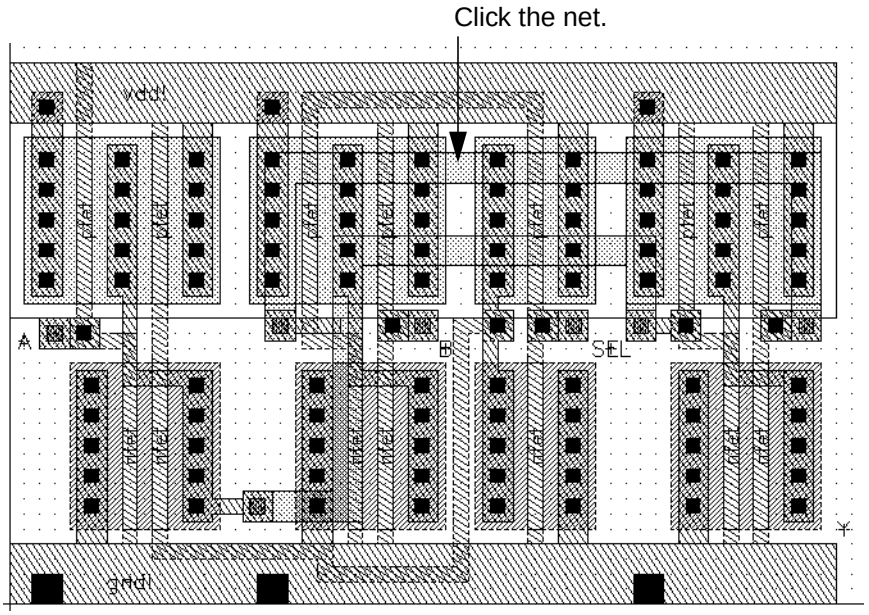
   The extracted cellview window opens. The two nets are now joined.

6. In the extracted cellview window, choose *Verify – Probe*.

   The Probing form opens.

7. In the Probing form, click *Add Device or Net,* and set *Probe Type* to *net only.*

8. Click left at X = 37, Y = 43 To select the net.

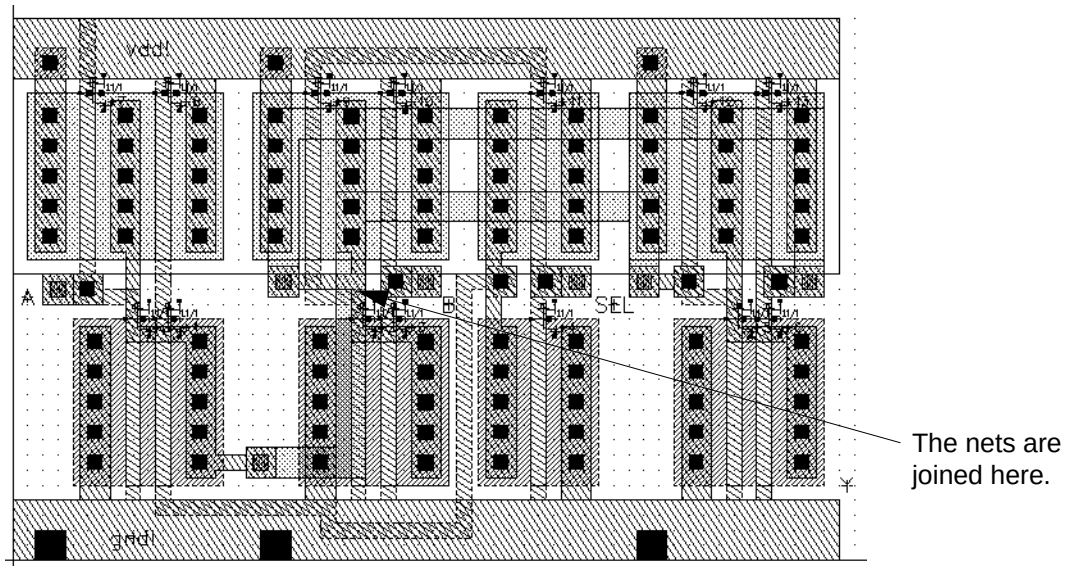


   Because there are two nets at this point, a text window appears so you can choose your net from a list.

9. Click 8 in the text window to choose your net.

10. In the Probing form, click *OK*.

You can see the nets are now joined.



11. In the Probing form, click *Remove All*.
12. Click *Cancel*.

Rerunning LVS

Now you can run LVS again on the new extracted cellview.

1. In the extracted cellview window, choose *Verify – LVS* to open the LVS form.
2. In the LVS form, click *Run*.

   A form asks if you want to save the cellview.
3. Click *OK* to save the *mux2* layout.

   The LVS job proceeds, then a dialog box appears, confirming the job has been completed. This might take a few minutes.
4. Click *OK* to close the dialog box.
5. In the LVS form, click *Output*.

   A text window containing the LVS report appears. The message in the text window should read

   ```
   The net-lists match
   ```

6. In the text window, choose *File – Close Window*.
7. In the LVS form, choose *Commands – Close Window*.
8. In the extracted cellview window, choose *Window – Close*.
9. In the layout cellview window, choose *Window – Close*.

   You have completed verifying the *mux2* layout.

Summary

In this chapter, you learned how to verify layout designs using interactive verification. Specifically, you

- Ran a DRC (Design Rule Checker)
- Viewed DRC errors
- Extracted a layout view
- Viewed extracted data
- Viewed a schematic
- Ran LVS
- Viewed LVS errors
- Cross-probed between the extracted layout and the schematic
- Ran verification programs again
- Used bindkeys:
  - Redraw [`Control-r`]
  - Zoom In [z]
  - Display Options [e]
  - Stretch [s]
  - Display Levels 0–20 [`Shift-f`]
  - Display Levels 0–0 [`Control-f`]
  - Fit All [f]
- Used the icon menu for *Save*

5

Creating and Editing ROD Objects with the Layout Editor

In this chapter you use Virtuoso relative object design (ROD) functionality in the Virtuoso layout editor to create simple layout objects and then examine their relationships to each other.

For complete information about ROD, see the *Virtuoso Relative Object Design User Guide*.

Note: You may proceed with this chapter even if you have not completed any of the previous chapters.

You use ROD to perform the following tasks:

When you finish this chapter, you will be able to

- Create simple objects using ROD
- Access ROD attributes through the Edit Properties form
- Create ROD user-defined handles
- Align one ROD object to another ROD object
- Create a path through a chop hole in a multipart path

About ROD

ROD lets you create objects and define their relationships at a high level of abstraction, so you can concentrate on your design objectives. ROD automatically handles the intricacies of traversing the design hierarchy and simplifies the calculations required to create and align geometries.

Every named database object, such as an instance, layout cellview, or named shape, automatically has relative object design information associated with it. This information is stored in a *ROD object*. A ROD object is also a database object, but it exists in relation to its associated named database object. A ROD object is identified by a unique *ROD object ID*.

A ROD object for a named shape, instance, or cellview contains the following information:

hierarchical name
cellview ID
database ID
transformation information (rotation, magnification, and offset)
alignment information, if any
number of segments (for shapes)
names and values of user-defined handles, if any
names of system-defined handles

Creating a ROD Rectangle

You can create a ROD rectangle by either typing commands in the Command Interpreter Window (CIW) or using the *Create Rectangle* command. In this section you

- Create a ROD rectangle using the *Create Rectangle* command
- Examine the code for the rectangle in the CIW
- Edit the rectangle and note the results in the Edit Properties form

You create the rectangle in the ROD library in a new cellview.

1. Choose *File – Open*.

   The Open File form appears.

2. Type the library, cell, and view names as follows and press *OK*:

Library Name	ROD

Cell Name	`examples`
View Name	`layout`

3. Obtain the ID for the current cellview by typing in the CIW

   ```
   cv = geGetEditCellView()
   ```

4. Choose *poly1* in the Layer Selection Window (LSW) for the entry layer.
5. Choose *Create – Rectangle.*

   The Create Rectangle form appears.

6. Set *As ROD Object* on.

   The *ROD Name* field becomes editable.

7. Type `rect` in the *ROD Name* field.
8. Click X= 3, Y = 11 and X= 9, Y= 9 to create the rectangle.
9. Press `Escape` to close the Create Rectangle form.

Note: To create the same rectangle using Cadence® SKILL language, you would type the following in the CIW. (You do not have to type this; it is just an example.)

```
rect = rodCreateRect(
    ?name "rect"
    ?cvId geGetEditCellView()
    ?layer "poly1"
    ?bBox list(3:11 9:9)
)
```

Examining the ROD Rectangle

You can look at information about a ROD object in the Edit Properties form or by typing commands in the CIW.

Looking at ROD Information in the Edit Properties Form

Changes you make to the rectangle are reflected in the Edit Properties form.

1. Select the rectangle.
2. Choose *Edit – Properties.*

The Edit Properties form appears. You should see the ROD name and XY coordinates you set in the Create Rectangle form.

3. Click *ROD* at the top of the Edit Properties form.

   The ROD fields appear. Examine the ROD information about the rectangle.

4. Compare the values of *upperLeft* and *lowerRight* in the *System handle* field.

The values should be the same as the coordinates in the *Attribute* fields.

Looking at ROD Information in the CIW

To examine the attributes of the ROD rectangle in the CIW,

1. Obtain the ROD Object ID by typing in the CIW

```
rect = rodGetObj("rect" geGetEditCellView())
rect~>??
```

   The system displays a list of the attribute names and values for the rectangle.

2. Examine the information displayed in the CIW. It should look like this, except that the ROD object D, cellview, and database IDs will be different

```
("rodObj:38395928" name "rect" cvId db:36579372
dbId db:36579580 transform
((0.0 0.0) "R0" 1.0) align
nil numSegments 4 userHandleNames nil
systemHandleNames
("width" "length" "lowerLeft" "lowerCenter" "lowerRight"
"centerLeft" "centerCenter" "centerRight" "upperLeft"
"upperCenter"
"upperRight" "length0" "start0" "mid0" "end0"
"length1" "start1" "mid1" "end1" "length2"
"start2" "mid2" "end2" "length3" "start3"
"mid3" "end3" "lengthLast" "startLast" "midLast"
"endLast"
)
```

Editing the ROD Rectangle

You are going to change the dimensions of the rectangle and view the results in the Edit Properties form.

1. In the layout window, deselect the rectangle by clicking an empty area.
2. Choose *Edit – Stretch*.

   The Stretch form appears.
3. Click on the right edge of the rectangle to start the stretch.

   As you move the cursor, an outline shows how the rectangle is changing.
4. Move the cursor to approximately X=11, Y=10 and press `Return` to end the stretch.
5. Click *Cancel* to close the Stretch form.
6. Click on the rectangle.

   In the Edit Properties form, the *Right* value should have changed from 9 to 11.
7. Deselect the rectangle by clicking in an empty area.

Creating a ROD Polygon

In this section, you create a ROD polygon. After you create the polygon, you examine and change the attributes using the Edit Properties form.

1. In the LSW, choose *metal1* for the entry layer.
2. Choose *Create – Polygon.*

   The Create Polygon form appears.
3. Set *As ROD Object* on.

   The *ROD Name* field becomes editable.
4. In the *ROD Name* field, type `polygon`.

   You can enter the points for a shape in the CIW or by clicking in the cellview.
5. To enter points for the polygon, do one of the following:
   - To enter points in the CIW, move the cursor to the command input area of the CIW and click the left mouse button, then type the XY coordinates as shown below, pressing `Return` after each set of coordinates:

```
11:11
11:7
17:7
17:9
13:9
13:11
```

- To enter points by clicking in the cellview, click once at each of the XY coordinates listed below. When you enter the last point (X = 13, Y = 11), either double-click or press `Return` to complete the polygon.

  First click: X = 11, Y = 11

  Second click: X = 11, Y = 7

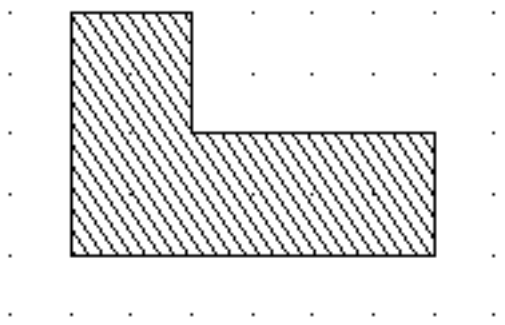
  Third click: X = 17, Y = 7

  Fourth click: X = 17, Y = 9

  Fifth click: X = 13, Y = 9

  Sixth click: X = 13, Y = 11

The completed polygon looks like this:



6. Click *Cancel* to close the Create Polygon form.

Note: To create the same polygon using SKILL code, you type the following in the CIW. (You do not have to type this; it is just an example.)

```
polygon = rodCreatePolygon(
            ?name "polygon"
            ?cvId geGetEditCellView()
            ?layer "metal1"
            ?pts list(11:11 11:7 17:7 17:9 13:9 13:11)
)
```

Examining the ROD Polygon

Now you can examine the attributes of the ROD polygon.

1. Obtain the ROD object ID by typing in the CIW

```
polygon = rodGetObj("polygon" geGetEditCellView())
polygon~>??
```

   The system displays a list of the attribute names and values for the polygon.

2. Examine the information displayed in the CIW. It should look like this:

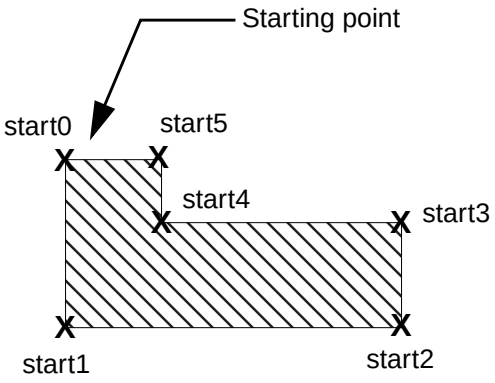
```
("rodObj:38395952" name "polygon" cvId db:36579372
dbId db:36579648 transform
((0.0 0.0) "R0" 1.0) align
nil numSegments 6 userHandleNames nil
systemHandleNames
("width" "length" "lowerLeft" "lowerCenter" "lowerRight"
"centerLeft" "centerCenter" "centerRight" "upperLeft"
"upperCenter"
"upperRight" "length0" "start0" "mid0" "end0"
"length1" "start1" "mid1" "end1" "length2"
"start2" "mid2" "end2" "length3" "start3"
"mid3" "end3" "length4" "start4" "mid4"
"end4" "length5" "start5" "mid5" "end5"
"lengthLast" "startLast" "midLast" "endLast"
)
)
```

Looking at Handles on ROD Objects

Now that you have created a ROD rectangle and polygon, you can examine and change their attributes. One important attribute of a ROD object is its handles. Handles are used to store points, calculations, and other information. In the Edit Properties form, you can view the names and values of handles in the ROD fields.

1. Select the polygon.
2. Click *ROD* at the top of the Edit Properties form.

3. Using this diagram as a reference, determine which points match the system handle values for *start0*, *start3*, and *start5*.



Editing the ROD Polygon

You can change the shape of the polygon by editing the points in the Edit Properties form.

1. Click *Attribute* in the Edit Properties banner.
2. In the *Points* field, change

11:11	to	11:13
13:11	to	13:13

3. Click *Apply*.

   The polygon changes to reflect the new points.
4. Click *ROD* in the Edit Properties banner.

   The values for *start0* and *start5* should be the new *Points* values you set for *Attribute*.
5. Deselect the polygon by clicking in an empty area.

Creating User-Defined Handles

In this section, you create a handle for the polygon you created earlier. When you define a handle, you specify a name and assign a value to it. The values of user-defined handles are stored in the database.

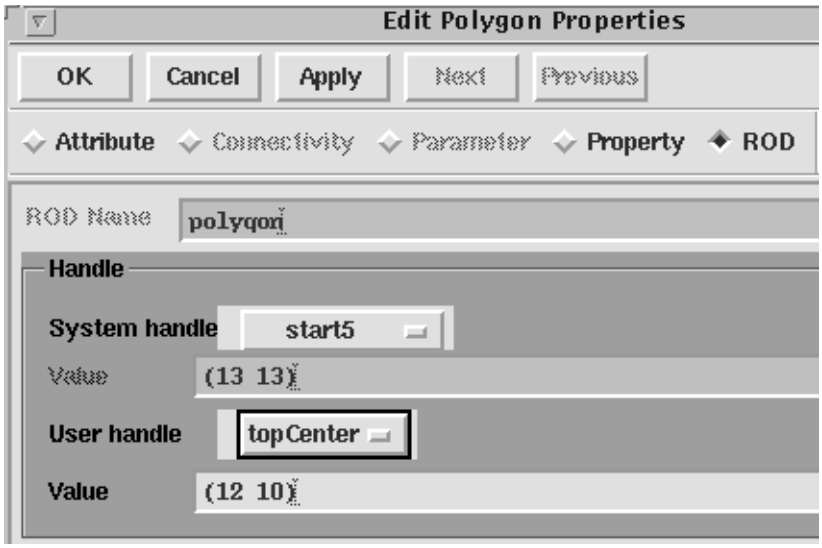
To create a user-defined handle,

1. In the CIW, type

```
rodCreateHandle(
?name "topCenter"
?type "point"
?value 12:10
?rodObj polygon
)
```

2. Reselect the polygon and notice the change in the *User handle* field.

   The value should reflect what you set to create the user-defined handle.



3. Deselect the polygon by clicking in an empty area.

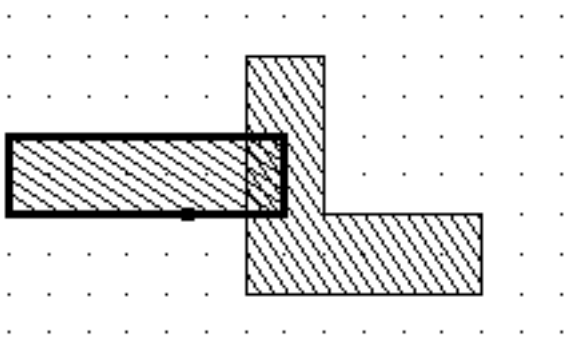
Aligning the ROD Polygon and Rectangle

An important feature of ROD is the ability to specify the position of one named object in relation to another named object. This is called *relative alignment*. Usually, you align objects by specifying a point handle on each object. You can also specify the distance between the two objects in the direction of the X axis, the Y axis, or both. The alignment between two objects is preserved when you manipulate either object and when you save and close the layout cellview.

In this section, you align the polygon and rectangle that you created earlier. The reference object is the rectangle and the reference handle is *centerRIght*. The object to be aligned is the polygon using its *topCenter* handle. Remember, *topCenter* is the handle you just created.

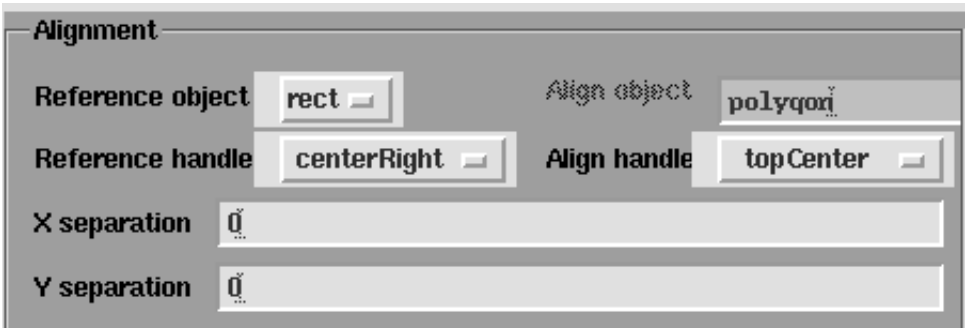
1. In the CIW, type

```
rodAlign(
?alignObj polygon
?alignHandle "topCenter"
?refObj rect
?refHandle "centerRight"
)
```



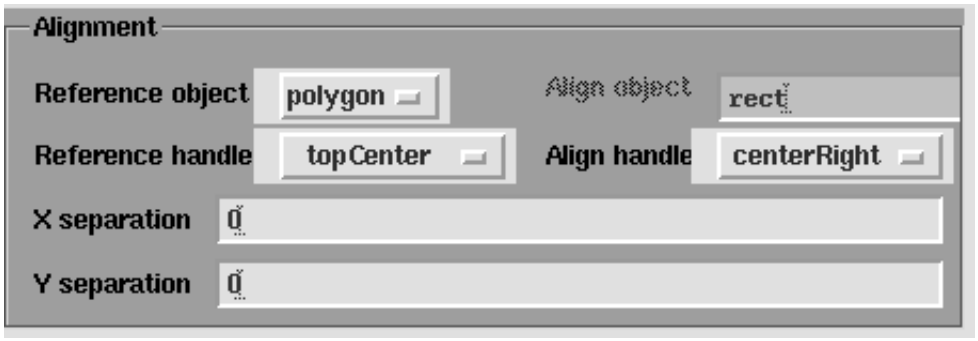
Now the *centerRight* handle of the rectangle is aligned to the *topCenter* user-defined handle of the polygon.

2. Select the polygon.
3. Look at the *Alignment* fields in the Edit Properties form. You should see the information you set in the CIW.



4. Select the rectangle.

The information in the Edit Properties form changes.



With the rectangle as the selected object, the polygon is the reference object and the rectangle is the object that is aligned.

Editing the Aligned Objects

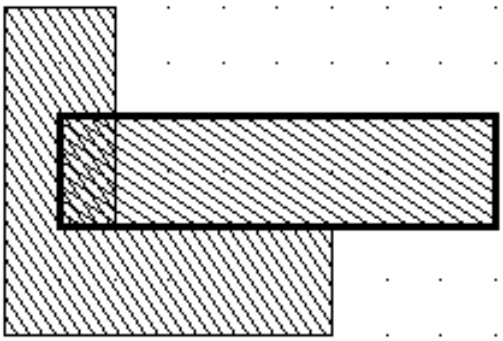
To demonstrate how aligned objects stay *relatively aligned*, in the next steps you change the value of the handle by which the rectangle is aligned, move the polygon, and stretch the rectangle.

Changing the Align Handle for the Rectangle

To change the value of the handle by which the rectangle is aligned, you update the *Align handle* field.

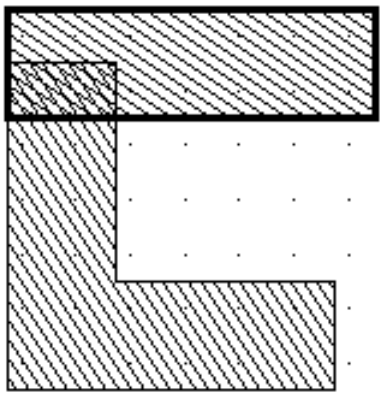
1. Select the rectangle.
2. In the *Alignment* section of the Edit Properties form, change the *Align handle* field from *centerRight* to *centerLeft*.

3. Click *Apply*.



The Edit Properties form shows that the *centerLeft* handle of the rectangle and the *topCenter* user-defined handle of the polygon are aligned.

4. Change the *Reference handle* field from *topCenter* to *start0*.
5. Click *Apply*.



The rectangle's *centerLeft* handle and the polygon's *start0* handle are aligned.

6. Deselect the rectangle by clicking in an empty area of the window.

Moving an Aligned Object

To move the polygon,

1. Choose *Edit – Move*.

   The Move form appears.

2. Select the polygon.
3. Move the polygon anywhere in the window.

   The rectangle stays aligned to the polygon.
4. Click *Cancel* to close the Move form.
5. Deselect all objects by clicking in an empty area of the window.

Stretching an Aligned Object

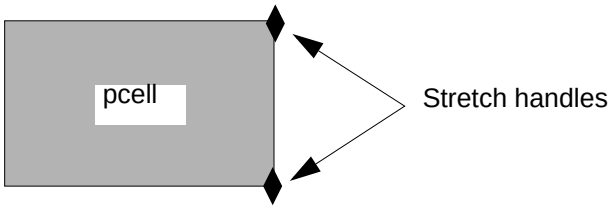
To stretch the rectangle,

1. Choose *Edit – Stretch*.
2. Click on the left side of the rectangle.
3. Move the cursor two grid spaces to the left.
4. Press `Return`.

   The rectangle stretches to the left. The polygon moves to stay aligned with the *centerLeft* handle on the rectangle.
5. Close the window.

Stretching a Parameterized Cell

This section introduces you to editing a stretchable parameterized (pcell). A *stretchable pcell* is a Cadence® SKILL-based pcell created with a handle assigned to one or more of its parameters for the purpose of changing the value of the parameter graphically. This kind of handle is called a *stretch handle*.



After you place an instance of the pcell, stretch handles let you graphically change the value of the associated parameters by selecting them and using the *Stretch* command. You can stretch a handle in the direction of the X or Y axis, depending on how it is defined in the pcell.

You are not actually stretching objects within the pcell or the pcell instance itself. Instead, you are graphically updating the value of the parameters associated with the selected handles. Graphically stretching a pcell instance has the same result as editing its parameters using the Edit Properties form.

In the following tasks, you open the `mux2gs layout` design from the `master` directory and save it in the `tutorial` directory, check for design rule violations, fix the violations you find by stretching the inverter and adding more contacts, and check to make sure the violations are gone.

Saving mux2gs in the tutorial Directory

Make a copy of the `mux2gs` design in the tutorial directory to use in the next several tasks.

1. Choose *File – Open.*

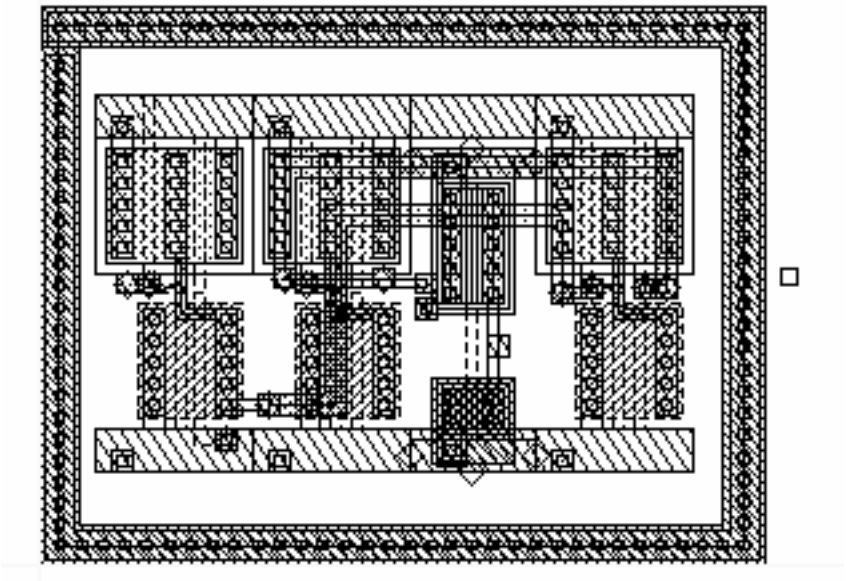
   The Open File form appears.

2. Type the library, cell, and view names as follows:

Library Name	`master`
Cell Name	`mux2gs`
View Name	`layout`

3. Click *OK*.

   The `mux2gs` cell from the `master` library opens.



4. In the layout window, choose *Design – Save As*.

   The Save As form appears.

5. Type the following:

Library Name	`tutorial`
Cell Name	`mux2gs`

6. Click *OK*.

   The `mux2gs` cell is saved to the `tutorial` library.

7. Close the *master mux2gs* layout window.

8. Choose *File – Open*.

   The Open File form appears.

9. Type the following to open the `tutorial` `mux2gs` `layout` design:

Library Name	`tutorial`
Cell Name	`mux2gs`

The layout window banner should say *tutorial mux2gs layout*.

Check for Design Rule Violations

The `mux2gs` cell contains several design rule errors. You'll discover this by running the Design Rule Checker (DRC).

1. Choose *Verify – DRC*.

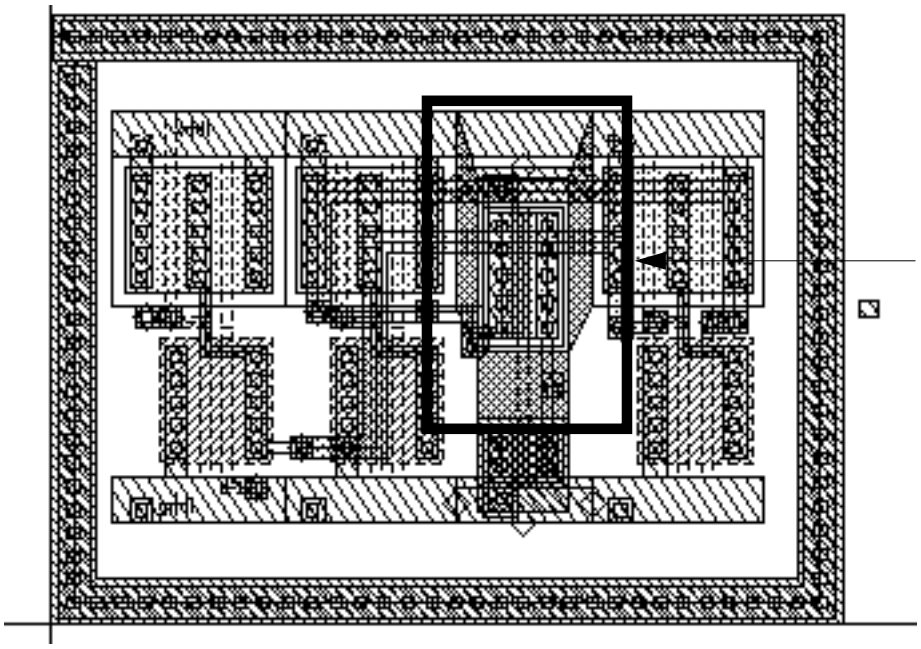
   The DRC form appears.

2. Click *OK* to accept the defaults.

   The following design rule errors appear in the CIW:

```
\o ** Summary of rule violation for cell "mux2gs layout" **
\o  # errors  Violated Rules
\o         1  drc("poly1" "pdiff" sep < 0.5)
\o         2  drc("nwell" sep < 8.5)
\o         1  drc("nwell" "pwell" sep < 8.5)
\o         1  drc("cont" "poly1" sep < 1)
\o         2  drc("metal1" sep < 1)
\o         7  Total errors found
```

DRC displays flashing markers to show the errors.



The pmos device is too close to the surrounding devices and interconnect; the nwell spacing and nwell-to-pwell spacing is incorrect. The pmos device overlaps the interconnect between the devices.

After you delete the markers, you will edit the inverter to fix the errors.

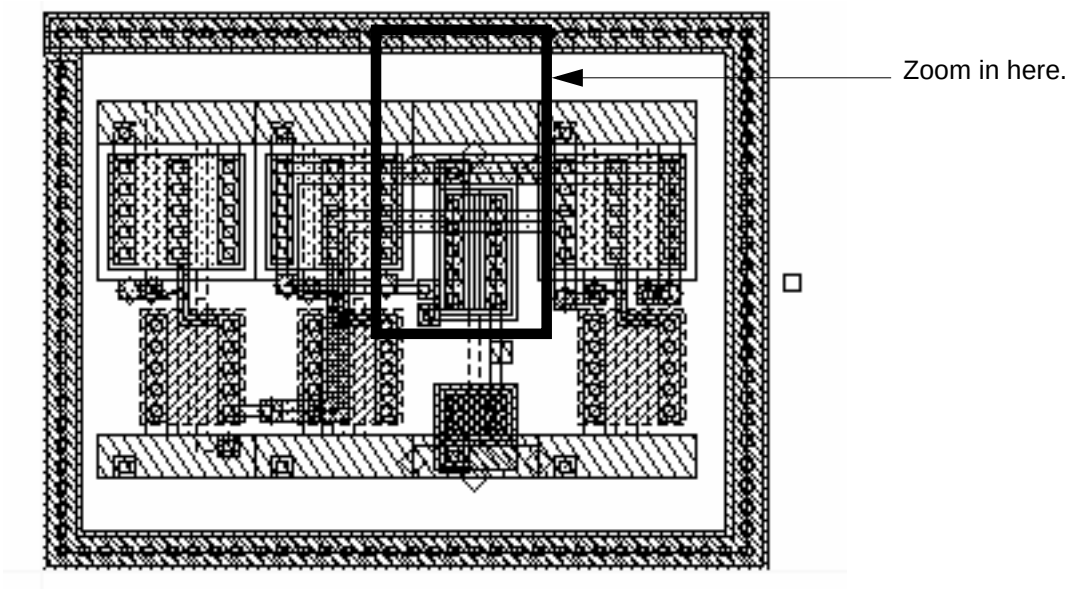
3. To delete the markers,

- Choose *Verify – Markers – Delete All*.
- Click OK to accept the defaults in the Delete All Markers form.

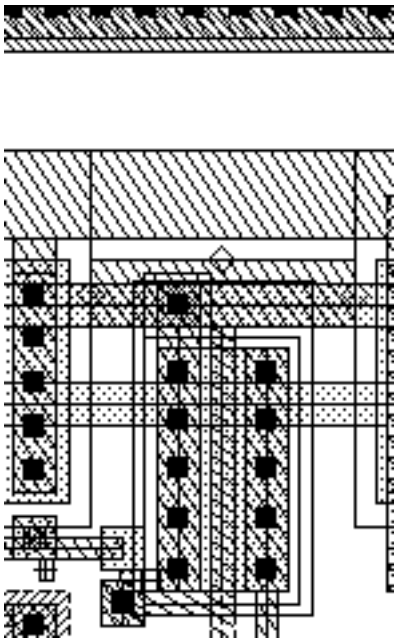
Fixing Design Rule Violations

The inverter with the violations is a stretchable pcell. You can fix the violations by stretching the inverter to increase its height so that it aligns with the neighboring cells. You also need to add a third column of contacts.

1. Zoom in on the upper half of the inverter, including some space above it.



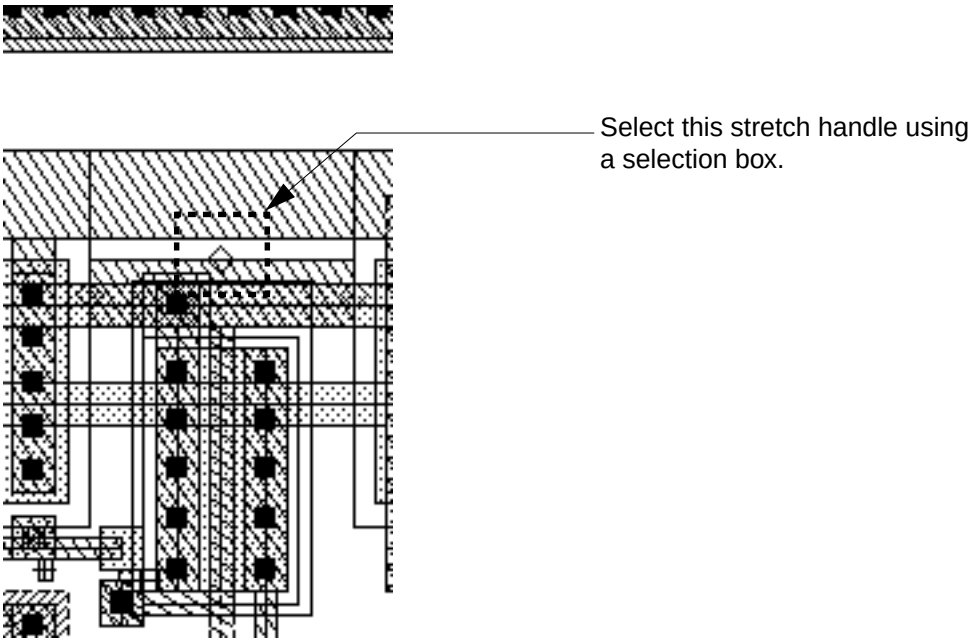
You should now see something like this:



2. Choose *Edit – Stretch*.

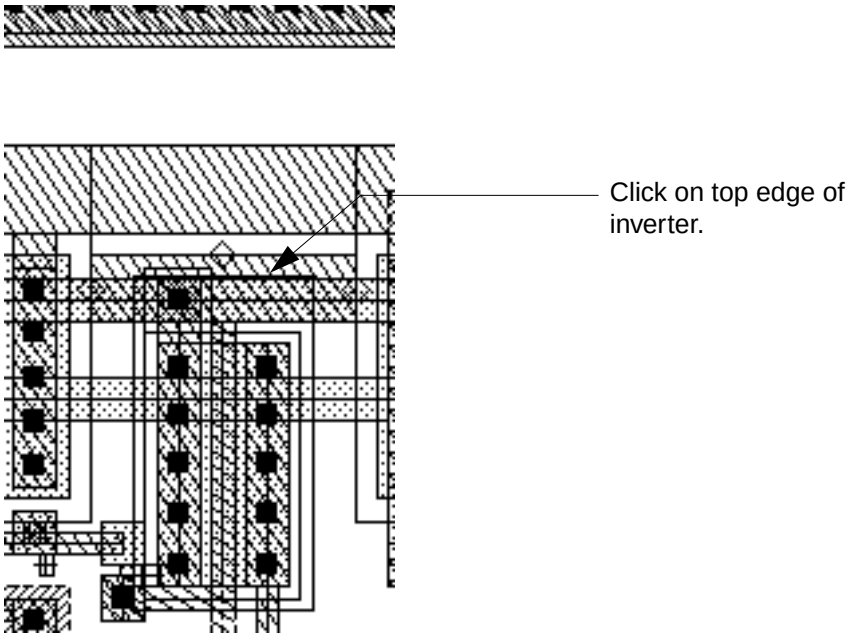
The system prompts you to select the figure to be stretched.

3. Select the stretch handle at the center of the top of the inverter using a selection box (click and hold the left mouse button).



When you release the mouse, the inverter instance is highlighted. You are prompted for a reference point for the stretch.

4. Click on the top edge of the inverter for the reference point.



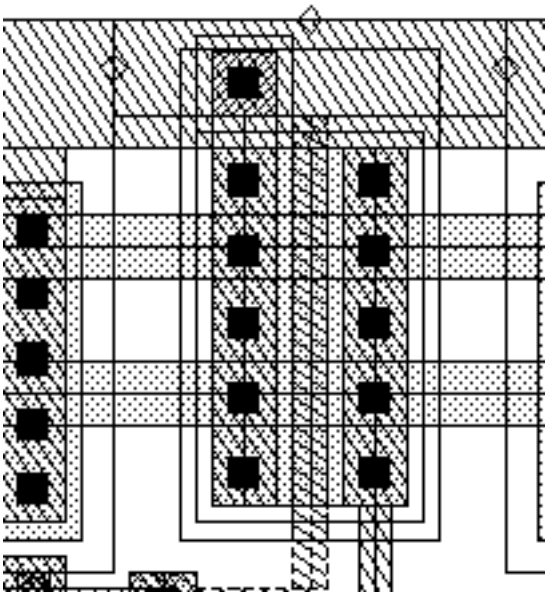
The system prompts you for the new location and displays text next to the inverter to show its current height:

```
Cell Height = 31
```

5. Slowly move the cursor upward until the text shows 36 units, then click.

```
Cell Height = 36
```

Zooming in, the upper part of the inverter now looks like this:



6. Cancel the *Stretch* command.

Adding More Contacts

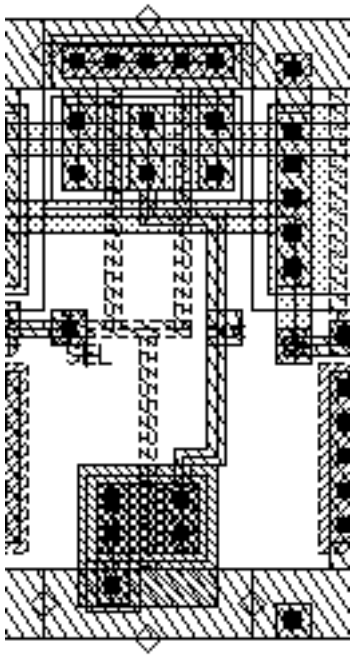
The inverter needs more contacts. You can add a third finger of contacts with the Edit Properties form.

1. Select the inverter.
2. Choose *Edit – Properties*.
3. Click *Parameter* at the top of the Edit Properties form.
4. Change *pMos Gate Width* to `10u`.
5. Change *pMos fingers* to `2`.
6. Change *Supply Width* to `4u`.

The Edit Properties form should look like this:

7. Click *Apply*.

After your changes and zooming out, the inverter should look like this:



8. Deselect the inverter by clicking in an empty space.
9. Run DRC again to verify that there are no more violations.

The DRC summary in the CIW should report zero errors found.

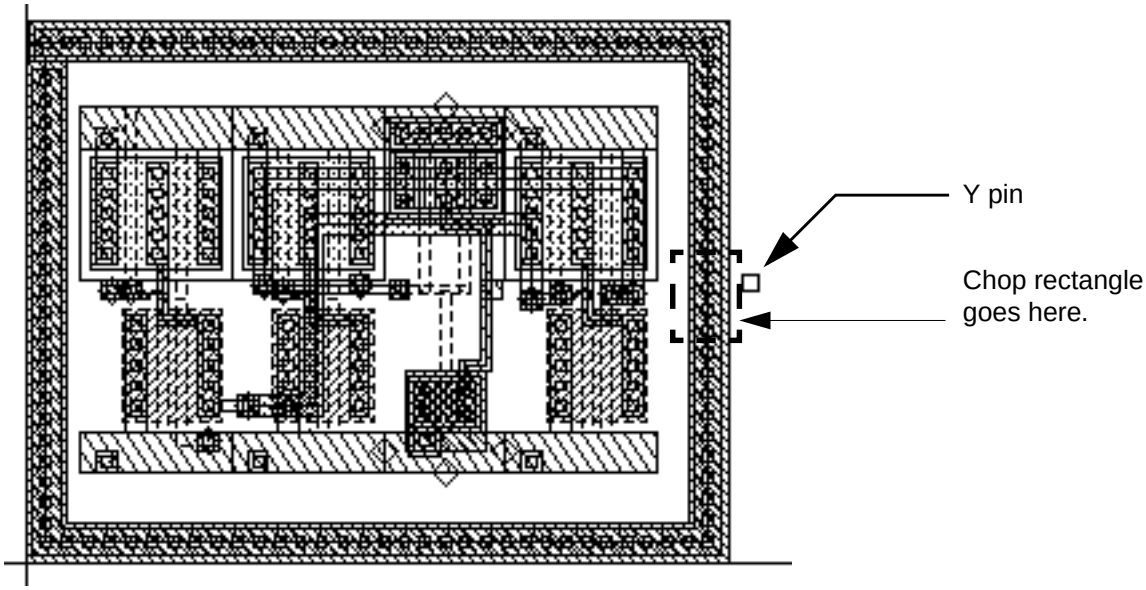
Creating a Path through a Multipart Path

The `tutorial mux2gs layout` design is missing an output path. You'll create the output path and connect it to the Y pin outside the multipart path guard ring, on the right side. To do this, you must chop a hole in the guard ring.

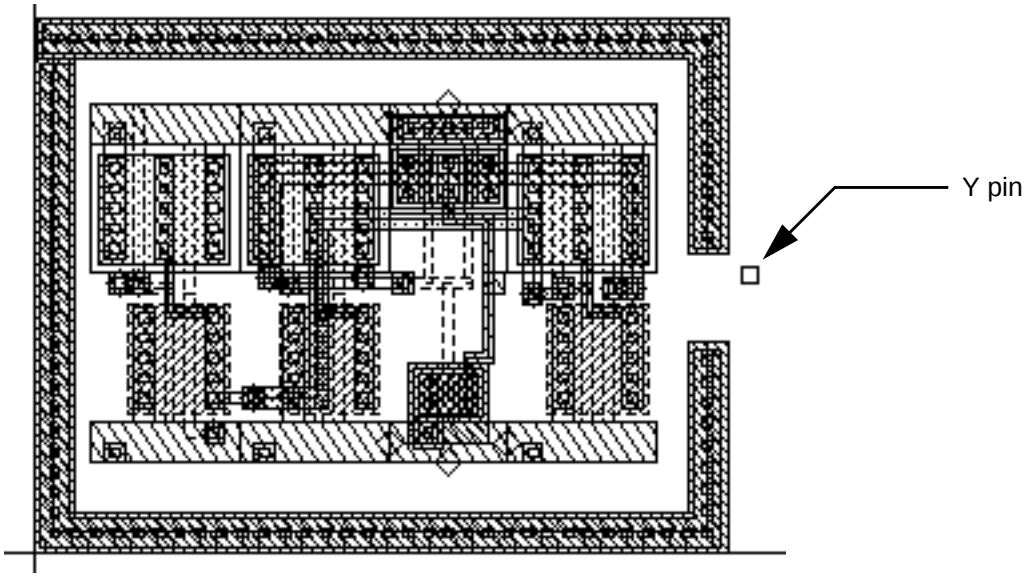
1. Select the entire guard ring.
2. Choose *Edit – Other – Chop*.

   The system prompts you to point at the first corner of the chop rectangle.

3. To create the chop rectangle, click on or near X= 64, Y= 30, then on or near X=71, Y=21.



The `mux2gs` cell looks like this with the chop hole:

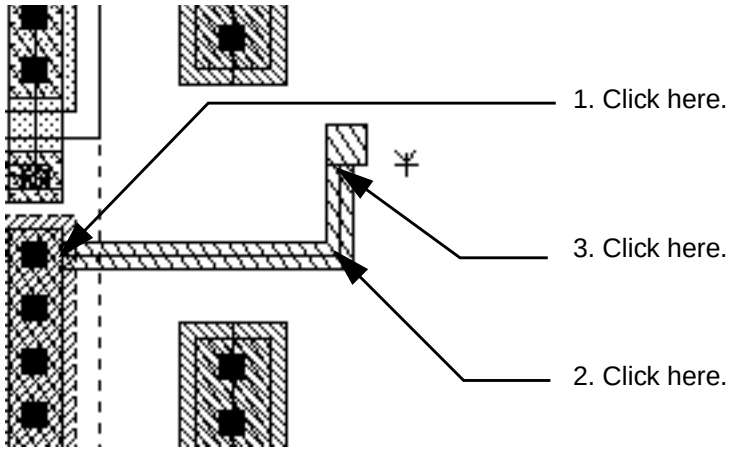


4. Deselect all objects.

5. Create the path by doing the following:

- In the Layout Editor Options form, make sure *Gravity* is off.
- Click on *metal1* in the LSW.

- Choose *Create – Path*.
- Zoom in.
- Draw the path as shown below.



6. Cancel the *Path* command.
7. Save and close your design.

Aligning ROD Objects

You can align a ROD object to a specific point or to a point handle on another ROD object (reference object). When aligning two ROD objects, the objects can be at the same level in the hierarchy or at different levels long as both objects are in the same top-level layout cellview. Also, you can specify positive or negative separation between alignment points in the direction of both the X and Y axes.

All instances have names and are therefore ROD objects. In this section, you create two instances of a transistor in a new cellview and align the contact in the first instance to a contact in the second instance.

1. In the CIW, choose *File – New – Cellview*.

   The Create New File form appears.

2. Type the following:

Library Name	`tutorial`
Cell Name	`align`

View Name	`layout`
Tool	*Virtuoso*

3. Click *OK*.

   A layout window opens.

4. To create the first instance, do the following:

   - Choose *Create – Instance*.

     The Create Instance form appears.

   - Type the following:

Library	`ROD`
Cell	`ptran`
View	`layout`
Names	`I1`

   - Click on X=2.5, Y=0.

5. To create the second instance, do the following:

   - In the Create Instance form, type the following:

Library	`ROD`
Cell	`ptran`
View	`layout`
Names	`I2`
Width	`3`

   - Click on X=10, Y=0.

6. Cancel the *Create Instance* command.

   Before you align the instances, you need to get the cellview ID and the ROD object IDs for the instances.

7. To get the cellview and ROD object IDs, type the following in the CIW:

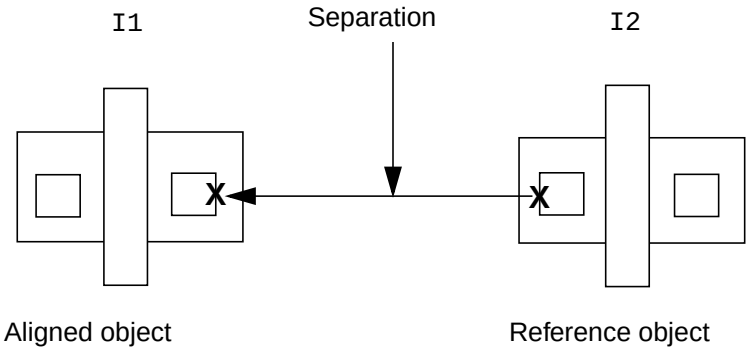
```
cv=geGetEditCellView()
cont1Id=rodGetObj("I1/rightcont" cv)
cont2Id=rodGetObj("I2/leftcont" cv)
```

As you type each statement, the database ID displays in the CIW.

8. To align the two instances, type the following in the CIW:

```
rodAlign(
?alignObj cont1Id
?alignHandle "centerRight"
?refObj cont2Id
?refHandle "centerLeft"
?xSep -2.0
)
```

The *centerRight* point handle on the right contact in instance `I1` is aligned to the *centerLeft* point handle on the left contact in `I2` with a separation of `-2`. The separation is a negative number because the reference object is on the right side of the aligned object.



9. Move either of the instances.

The instances should retain their alignment when you move one of them.

You can study the SKILL code that created the instances. The code is located in your `cell_design` directory:

`cell_design/skill/pcell.il`

Summary

In this chapter, you learned how to work with ROD objects. Specifically, you

- Created simple ROD objects
- Examined ROD object database information in the CIW
- Edited ROD objects using the Edit Properties form
- Viewed information for handles on ROD objects in the Edit Properties form
- Created a ROD user-defined handle
- Aligned two ROD objects using the user-defined handle you created
- Edited the aligned ROD objects
- Finished a layout design by
  - Running DRC to determine errors
  - Correcting DRC errors by
    - Stretching the ROD pcell inverter
    - Editing the parameters of the ROD pcell inverter
  - Creating a chop hole for the output path
  - Creating a *metal1* output path
  - Running DRC to check the changes you made
- Aligned two instances across hierarchy

6

Creating a Graphical Parameterized Cell

This chapter covers the following topics:

When you finish this chapter, you will be able to

- Understand parameter definitions
- Define stretch lines
- Define repeat groups
- Define dependent stretch lines
- Examine the Parameter Summary
- Compile pcells
- Test and modify pcell parameters

For more information about graphical pcells, see the *Virtuoso Parameterized Cell Reference* manual. For information about ROD pcells, see the *Virtuoso Relative Object Design User Guide*.

Introduction

This chapter shows you how to create graphical parameterized cells (pcells) using the Virtuoso layout editor environment. A parameter is a variable that controls the size, shape, or contents of a cell instance. When you place an instance of a pcell, the Create Instance form displays a list of its parameters. You enter values for the parameters to change the characteristics of the instance.

Note: Graphical pcells differ from Virtuoso relative object design (ROD) pcells; they do not have ROD information stored with them.

Creating and using pcells allows you to add many different versions of the same cell to a layout by assigning different values to their parameters. For example, earlier in this tutorial, you used the `ptran` and `ntran` cells to create an inverter. They are both pcells with parameters for setting the width, length, and number of gates for each instance.

Without parameterized cells, your library might need to contain a large number of similar cells. Instead, you can create one representative pcell, such as an N-transistor, and define different values for its parameters as you create instances.

Defining parameters for a pcell is an iterative process. A good process to follow, and one this chapter follows, is to define one parameter, compile the pcell, test it, adjust the parameter if necessary, and then define the next parameter. See "Flowchart for Defining Pcell Parameters" on page 162.

In this chapter, you will learn how to create a pcell by defining parameters that let you change shapes in your designs. You will compile your cell into a pcell each time you define a parameter, then test the change by placing an instance.

The Pcell compiler evaluates all parameters when you compile. In this tutorial, you will define one new parameter, compile the pcell, and then check the results. After you have learned how to define parameters, you can define several new parameters before compiling. Also, you can define an expression that refers to a parameter before you define the parameter, as long as you define the parameter before you compile. The Pcell compiler evaluates all parameters when you compile, not before.

When you compile, the system creates or updates the original cell in your library. This cell is called a *supermaster*. When you place instances in a layout cellview, each unique variation is temporarily stored in virtual memory and referred to as a *submaster*.

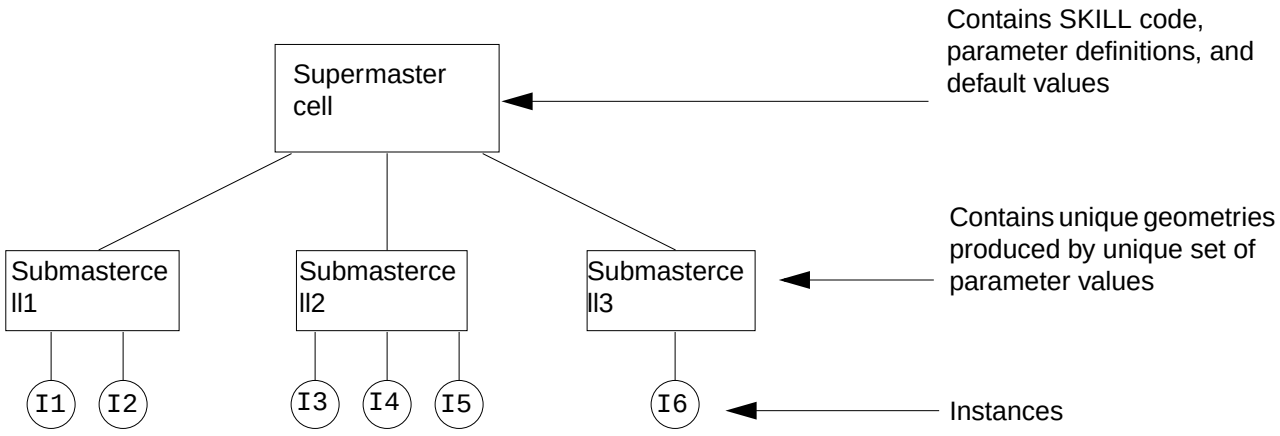
More About Pcell Supermaster and Submaster Cells

A pcell exists at three levels:

- Supermaster
- Submaster
- Instance

The original pcell supermaster resides in the database and contains all parameter definitions and their default values. When you want to add, change, or delete a parameter, you need to open the supermaster.

The Parameterized Cell software creates one submaster in virtual memory for each unique set of parameter values assigned to an instance of the supermaster in any open cellview.



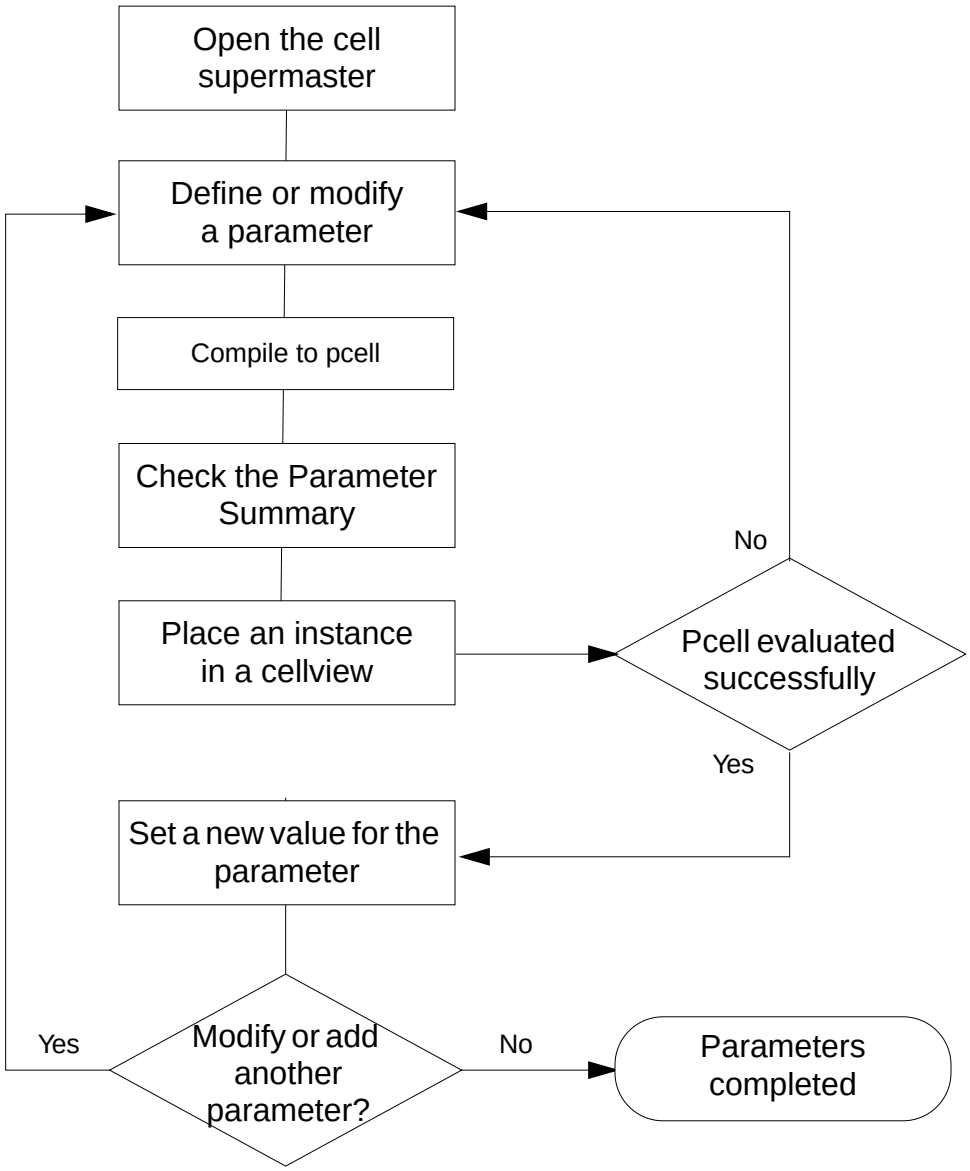
Submaster cells reside in virtual memory for the duration of your editing session and are accessible to and shared by all open cellviews. When you create a new instance, and a submaster with the same parameter values already exists in virtual memory, the new instance references the existing submaster cell; the software does not generate a new submaster.

How Submasters Are Deleted

The database software automatically purges (removes) referenced submaster cells from virtual memory (but does not remove the supermaster pcell from your disk) when you:

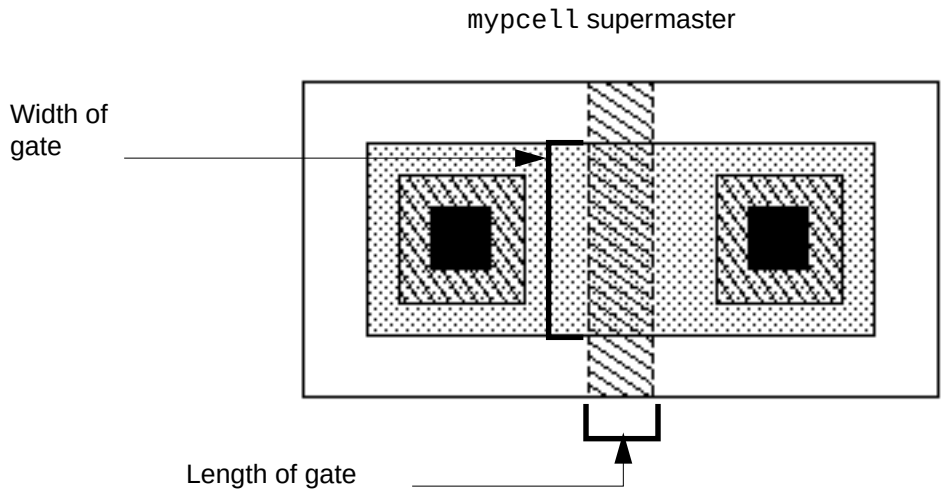
- Close all cellviews that reference the particular submaster cells
- End an editing session by exiting the Cadence software

Flowchart for Defining Pcell Parameters



About the Tutorial Transistor

You will use the following CMOS transistor in this chapter: `pCells mypcell layout`. It looks like this:

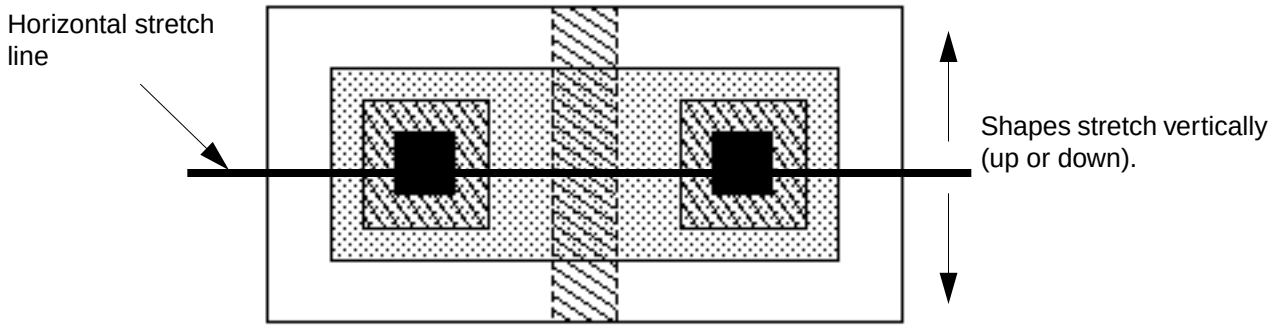


About Stretch Lines

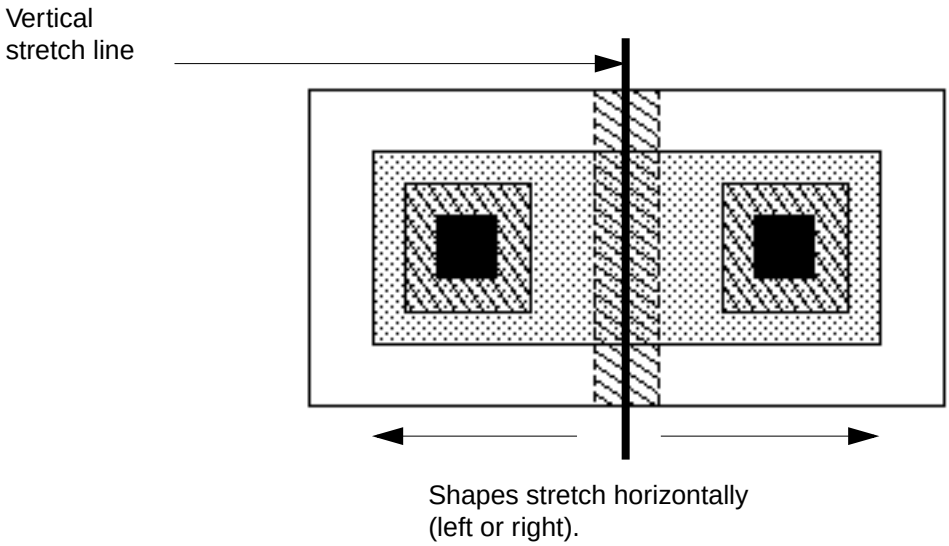
A *stretch line* is a graphical pcell parameter that allows the width or length of one or more shapes to change when you place an instance. Defining a stretch line is a graphical way of specifying that certain shapes can stretch and the direction in which they can stretch. You can specify a stretch line parameter as either a variable (named parameter) or as an expression.

- If you define a stretch line parameter as a variable, you can enter a new value for the stretch line parameter in the Create Instance form when you place an instance.
- If you define a stretch line as an expression, the system calculates its value when you place an instance. An expression can refer to other parameters defined for the same pcell.

All shapes intersected by a stretch line can stretch in a perpendicular direction from the stretch line. For example, to make shapes stretch vertically, you define a horizontal stretch line.



To make shapes stretch horizontally, you define a vertical stretch line.



The Reference Dimension

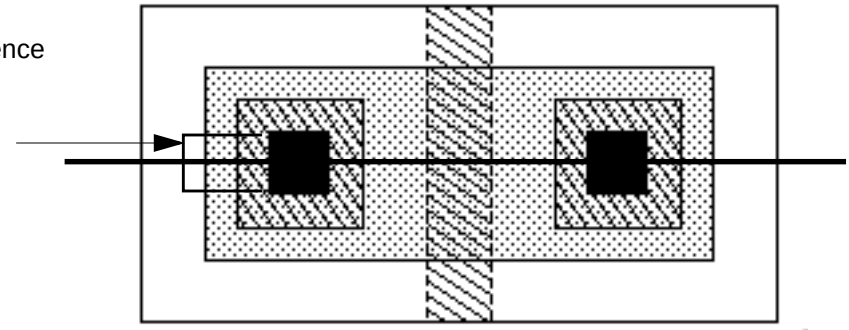
Stretch lines require a *reference dimension*. The reference dimension is a number that serves as a starting point for a stretch and is usually equal to the size of a significant shape intersected by the stretch line in the supermaster.

The system computes the results of a stretch by **subtracting** the reference dimension from the value you enter for the stretch parameter.

You can specify the reference dimension or use the system default, which is the size of the shortest edge of a shape crossed by the stretch line.

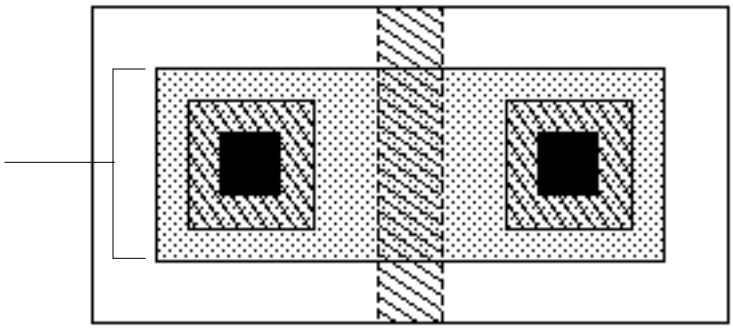
For example, one goal for the tutorial transistor is to stretch the width of the gate (and other intersected shapes) by defining a horizontal stretch line. Of the shapes intersected by the horizontal stretch line, the shortest belongs to the contacts, so the system would set the reference dimension equal to the width of the contacts.

System default for reference dimension is width of contacts.



However, the contacts are not significant in determining the width of the gate; the p-diffusion should control the width of the gate. The p-diffusion measures 3 microns, so in the tutorial, you need to specify the reference dimension as 3 rather than accept the default.

Reference dimension should be width of p-diffusion (3 microns).



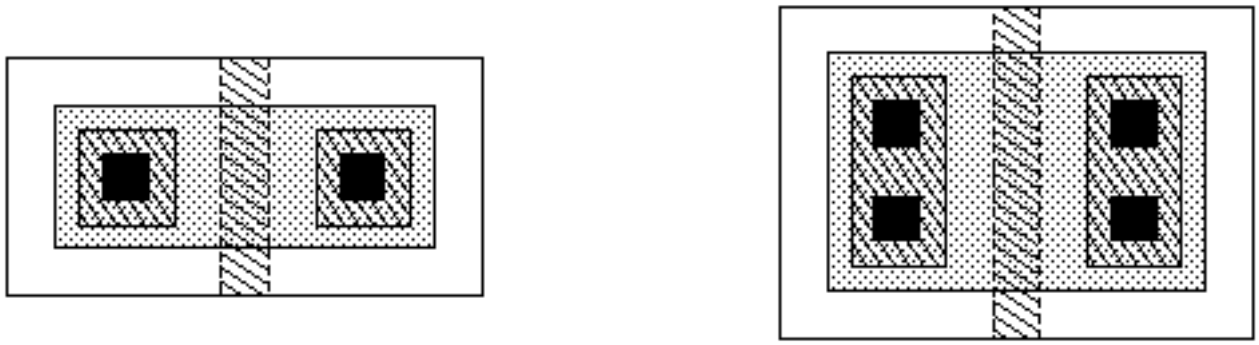
When you place an instance of the tutorial pcell and change the value of the stretch line parameter, shapes intersected by the stretch line change by the value you enter **minus** the 3-micron reference dimension.

About Repetition Parameters

A *repetition parameter* causes objects to repeat in the X direction, Y direction, or both. You create a repeat group by selecting shapes to be included in the group and specifying the number of repetitions (number of times the object occurs).

For example, for the contacts in the tutorial transistor, you will define a repetition parameter to control the number of repetitions (number of times the contact is repeated). Rather than defining the repetition parameter as a variable, you will define it as an expression based on the stretch line parameter you defined earlier for the width of the diffusion. This causes the system to compute the number of contacts needed for the current width of the diffusion.

In the illustration below, the width of the diffusion for the transistor on the left is three microns; the width of the diffusion for the transistor on the right is five microns, resulting in repeated contacts.

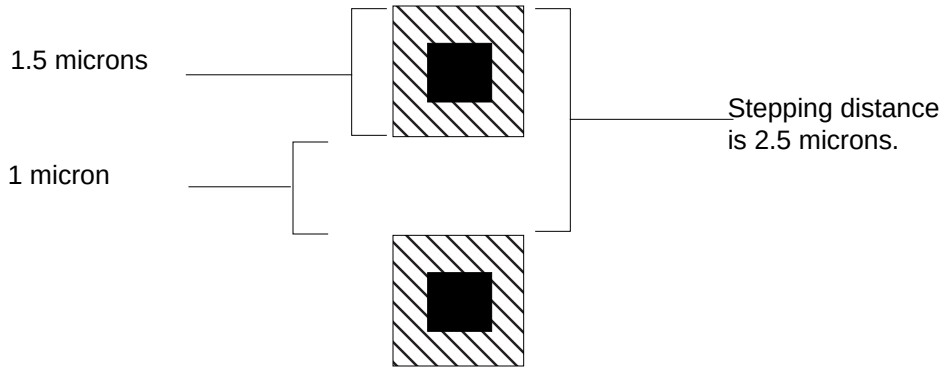


The Stepping Distance

You also must specify a *stepping distance*. The stepping distance controls the space between repeated objects or groups of objects. It is equal to the width of the object you want to repeat plus the space you want between repeated objects.

`Stepping Distance = object_width + intervening_space`

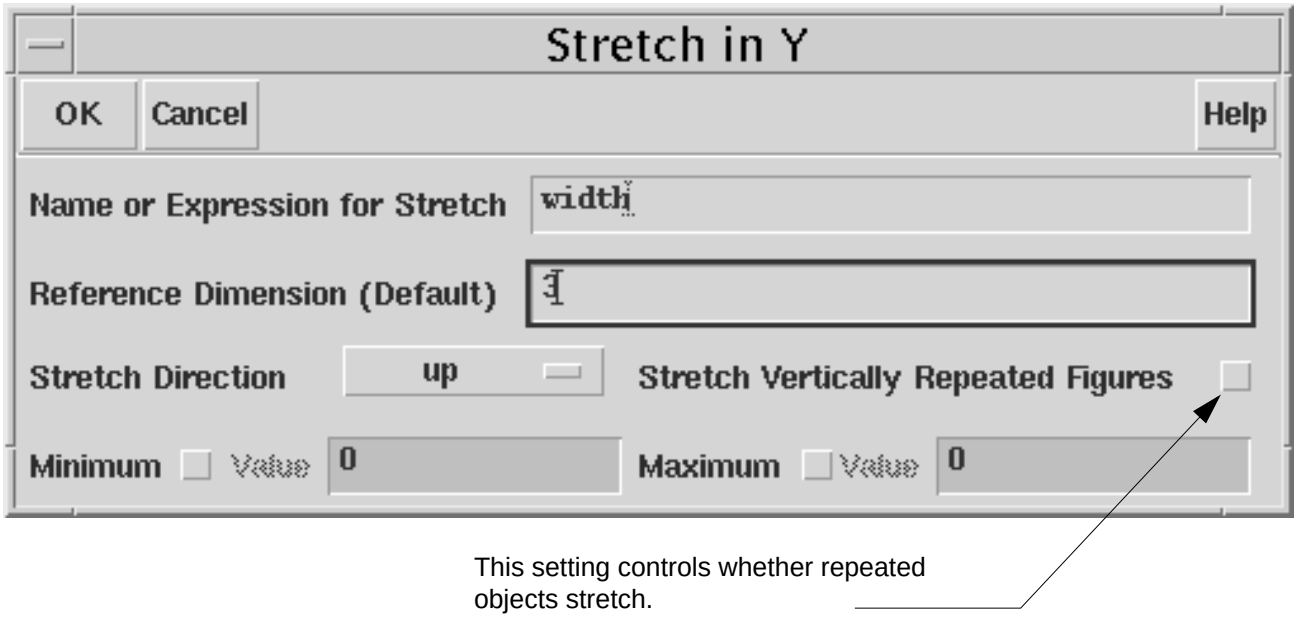
For example, when each contact is 1.5 microns wide, and you want a 1-micron space between contacts, the stepping distance is 2.5.



As is true for any pcell parameter, the stepping distance and number of repetitions can be defined as parameters (variables) or as expressions that change based on the values of other parameters.

Repeating and Stretching the Same Objects

You can both repeat and stretch the same objects. You control whether or not repeated objects stretch with the *Stretch Vertically Repeated Figures* option in the Stretch in Y form. By default, the system does not stretch objects crossed by a stretch line if those objects are also defined to repeat.



Starting the Layout Editor

You can follow the steps in this chapter with or without completing the previous chapters.

- Type the following in an xterm window to start the layout editor:

```
cd ~/cell_design
layout &
```

Opening the Tutorial Transistor

To open the tutorial design containing the CMOS transistor, do the following:

1. Choose *File – Open*.

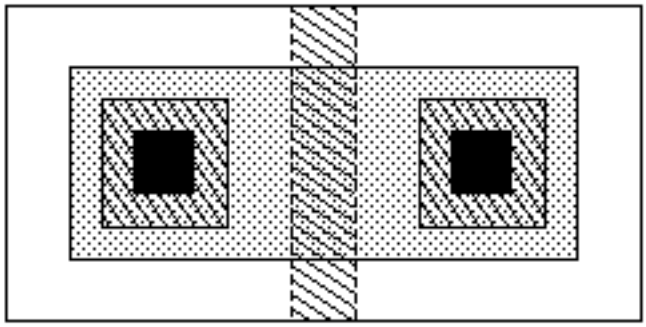
   The Open File form appears.

2. Set the library, cell, and view names as follows:

Library Name	`pCells`
Cell Name	`mypcell`
View Name	`layout`

3. Click *OK*.

A layout window containing the `mypcell` layout cellview appears.



Defining the First Parameter: Stretch Line for Gate Width

In this section, you learn to

- Turn off repeat mode, so you do not inadvertently repeat commands
- Check the pcell to see if it already has parameters defined
- Define a stretch line parameter as a named variable
- Define the width of the p-diffusion as the reference dimension

Turning Off Command Repeat Mode

Repeat mode eliminates steps when you use the same command many times in a row. For this chapter, you will turn off repeat mode off to avoid accidentally creating extra parameters.

1. In the `mypcell` layout window, press `Shift-e` to display the Layout Editor Options form.
2. Click the button next to *Repeat Commands* and click *OK*.

Checking for Existing Parameters

To check the cell for parameters, do the following:

1. Choose *Pcell – Parameters – Summarize*.

   There are no parameters defined for this cell. The system beeps twice and a dialog box appears saying there are no parameters.

2. Click *Close* in the dialog box.

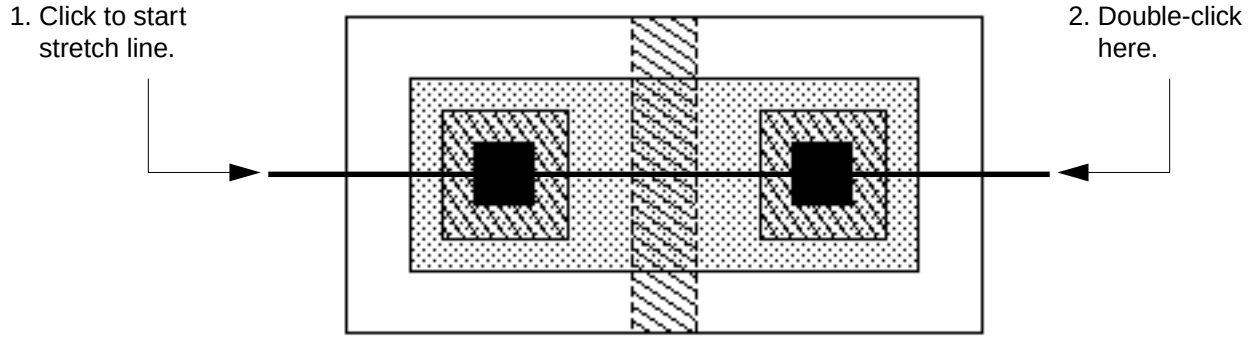
Defining a Stretch Line for the Width

To make objects stretch in the direction of the Y axis, you need to define a horizontal stretch line. You will define the line across the center of the whole cell so that all shapes within it stretch together.

1. Choose *Pcell – Stretch – Stretch in Y.*

   The system prompts you to draw a horizontal line to control vertical stretch.

2. Define the stretch line by doing the following:
   - Start the stretch line by clicking in the center of the left edge of the pcell.
   - End the stretch line by moving the cursor to the right edge and double-click.

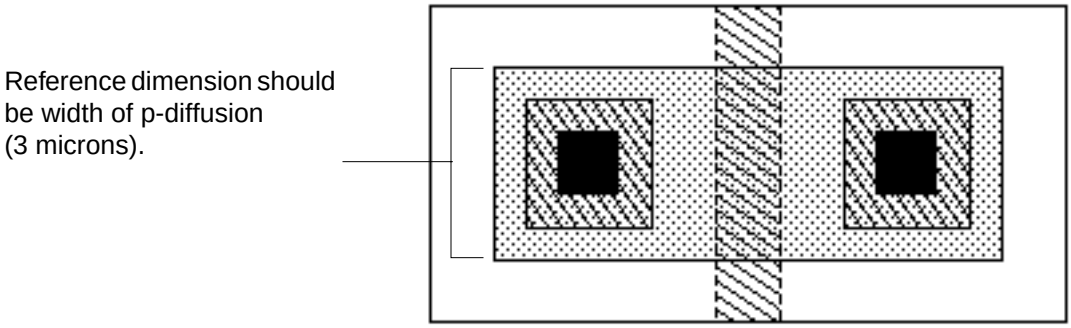


   The Stretch in Y form appears. You will define the stretch line as a parameter (variable) named `width`.

3. For *Name or Expression for Stretch*, type `width`.

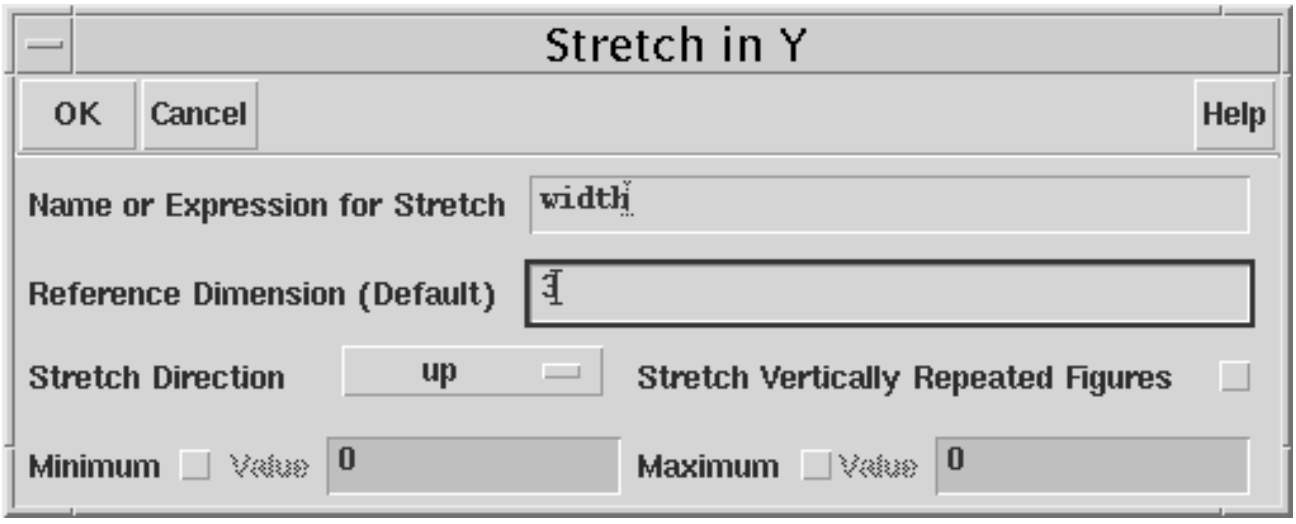
   When you create an instance of this pcell, the Create Instance form contains a field named *width* to let you enter a value for the stretch line parameter.

   You want *Reference Dimension* to equal the width of the p-diffusion.



4. For *Reference Dimension*, change the value to 3.

5. Make sure the *Stretch Direction* is set to *up* and click *OK*.



   You accepted the default for the *Stretch Direction*, so all objects intersected by the stretch line will stretch upward from the line.

You have completed defining the first parameter: the stretch line for the width of all objects in the cell. Next, you will compile the cell to create a new version of the supermaster and test it.

Compiling and Testing the Width Stretch Line Parameter

If you compile the pcell and test it after each change, it is easier to find out what is wrong, if anything, and go back to make corrections.

In this section, you learn to

- Compile the pcell
- Check the Parameter Summary
- Test your change by placing an instance of the pcell

Compiling the Pcell

To compile a pcell, do the following:

1. In the `mypcell` layout window, choose *Pcell – Compile – To Pcell*.

   The first time you compile a parameterized cell, the Compile To Pcell form appears so that you can classify the pcell as a *transistor*, *contact*, *substrate contact*, or *none* (the Virtuoso compactor needs this classification).

2. Click *OK* to accept the default, *transistor*.

   You see the following messages in the Command Interpreter Window (CIW).

```
Compiling Parameterized Cell ...
Compilation complete
```

Looking at the Parameter Summary

After compiling, it is a good idea to look at the Parameter Summary to verify your changes. The Parameter Summary shows information about all the parameters defined for the supermaster pcell.

1. In the `mypcell` layout window, choose *Pcell – Parameters – Summarize*.

   A window opens, showing the following information:

```
Parameters defined in this parameterized cell:
    width

Stretch
-------
Stretch Type: Vertical
Name or Expression for Stretch:  width
Stretch Direction:  up
Reference Dimension (Default):  3.000000
```

   The information above is correct for the tutorial pcell.

2. Close the window by choosing *File – Close Window*.

Testing the Stretch Line Parameter

You will test the stretch line parameter by opening a cellview, creating an instance, and changing the value of *width*.

1. Open a cellview for placing instances by doing the following:

   - In the CIW, choose *File – Open*.

     The Open File form appears.

   - In the Open File form, choose *pcelltest* from *Cell Names*.

   - Click *OK*.

   The `pcelltest` cellview window opens. It is empty except for a large rectangle. The rectangle makes the window large enough to place several instances.

2. Place an instance by doing the following:

   - In the `pcelltest` cellview window, open the Create Instance form by pressing `i`.

     The Create Instance form appears.

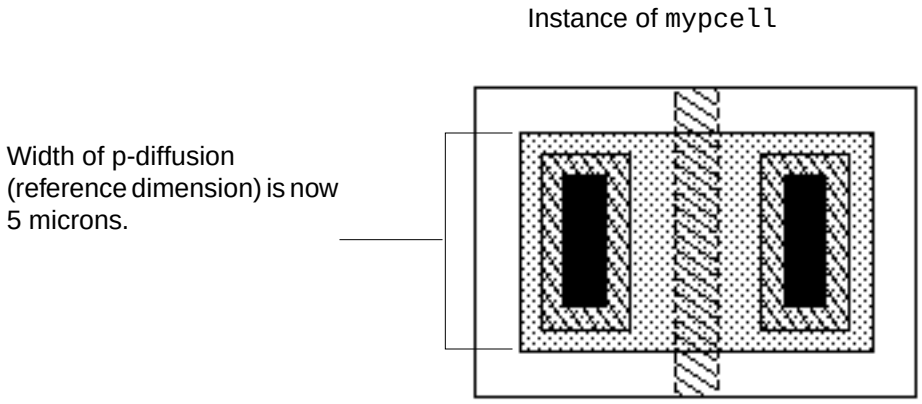
   - In the Create Instance form, type `mypcell` for *Cell*.

   - Display the width parameter field in the Create Instance form by pressing `Tab`.

     The *width* parameter for *mypcell* appears at the bottom of the Create Instance form. The value shown is the default. You set this default value when you typed 3 for *Reference Dimension* in the Stretch in Y form.

- Change *width* to 5.
- Click in the `pcelltest` layout window.

  The instance looks like this:



The width increased for all shapes intersected by the stretch line. The stretch line is working as it was defined. However, it would be better if the contacts repeat to fill the space available, rather than stretch. In the next section, you create a new parameter to make the contacts repeat.

You have completed testing first parameter: stretch line for width.

Defining the Second Parameter: Repeating the Contacts

In this section, you define a repetition parameter for the contacts in the tutorial transistor. To determine the number of repetitions, you can specify a new parameter name or enter an expression. If you specify a new parameter name, you have to type the number of repetitions when you place an instance (or accept the default). If you enter an expression, you can make the number of contacts dependent on another parameter.

You do not want the contacts to stretch; you would like them to repeat to fill the width of the p-diffusion. You will define the value of the repetition parameter as an expression so that the system computes the number of contacts based on the current value of the pcell *width* parameter. (You defined the *width* parameter earlier with a horizontal stretch line.)

In this section, you learn to

- Create a repeat group

- In the Command Interpreter Window (CIW), look at the expression for controlling the number of repetitions

Defining a Repeat Group for Contacts

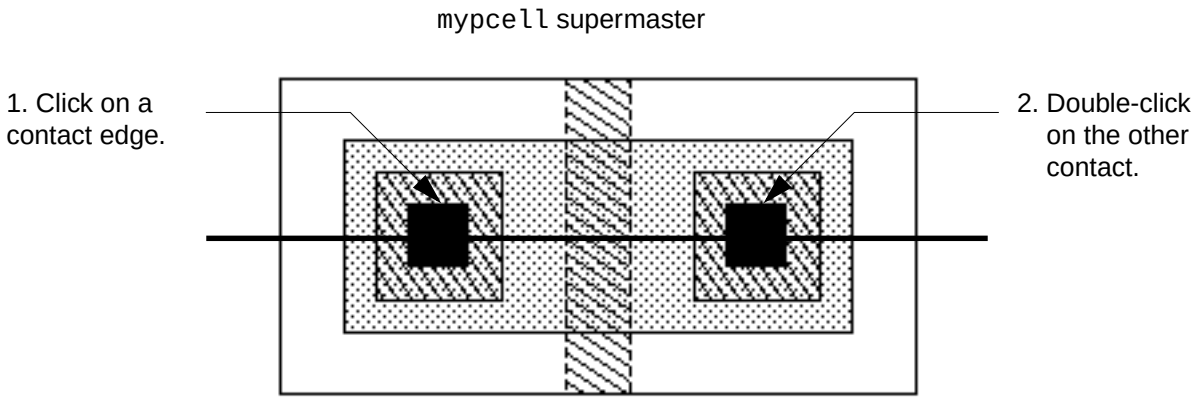
To make the contacts repeat along the Y axis to fill the space made available by the value of the *width* parameter, do the following:

1. Move the cursor back to the `mypcell` layout window.
2. Choose *Pcell – Repetition – Repeat in Y.*

   The system prompts you to choose objects for the repeat group.
3. Click on the edge of one of the contacts.

   The contact is highlighted. If you select the wrong shape, press `Control` and click the shape to deselect it.

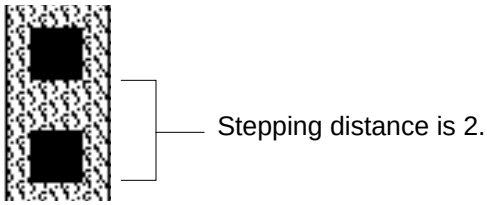
   Note: In the layout editor, you usually press `Shift` and click to select more than one object. The Pcell program assumes you want to select multiple shapes, so you do not need to press `Shift`.
4. Finish selecting by double-clicking on the other contact.



   Both contacts are highlighted and the Repeat in Y form appears. (If the form does not appear, press `Return`.)

   You need to enter a value for *Stepping Distance*. The stepping distance is equal to the width of the object you want to repeat plus the space you want between repeated objects.

Each contact is 1 micron wide and you want a 1 micron space between contacts, so the stepping distance is 2.



5. For *Stepping Distance*, type 2.

Caution

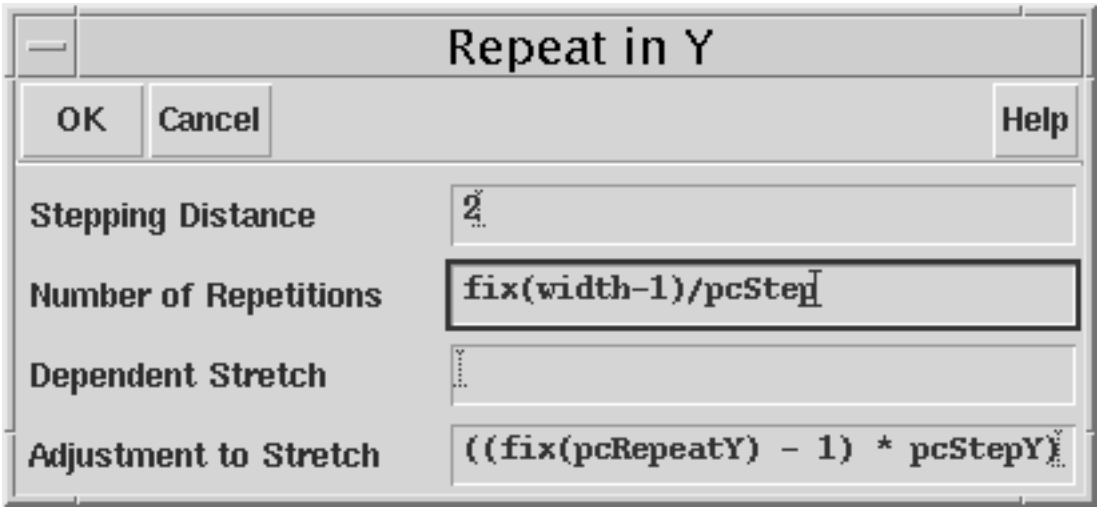
In the following step, be careful with spaces around the minus sign (–). If you enter a space on one side and not the other, the pcell will not be created correctly.

6. For *Number of Repetitions*, type the following expression, without spaces:

```
fix(width-1)/pcStep
```

This expression is explained in "Understanding the Number of Repetitions Expression" on page 177.

The Repeat in Y form looks like this:



7. Click *OK*.

You have completed the second parameter: the repetition parameter for contacts.

Compiling and Testing the Contact Repeat Group Parameter

Now you will compile again, check the Parameter Summary, and place an instance to test the repetition parameter for contacts.

1. In the `mypcell` layout window, recompile the pcell by choosing *Pcell – Compile – To Pcell*.

   This time, the Compile To Pcell form does not appear because you already set the pcell classification to *transistor*.

2. Check the CIW to make sure your pcell compiled without errors.

3. Check the Parameter Summary by choosing *Pcell – Parameters – Summarize*.

   A window opens, showing the following information:

```
Parameters defined in this parameterized cell:
    width

Stretch
-------
Stretch Type: Vertical
Name or Expression for Stretch:  width
Stretch Direction:  up
Reference Dimension (Default):  3.000000


Repetition
----------
Objects Repeated in:   Y
Number of Objects in this group:  2
Stepping Distance:  2
Number of Repetitions:  (fix((width - 1)) / pcStep)
Dependent Stretch:
Adjustment to Stretch:
```

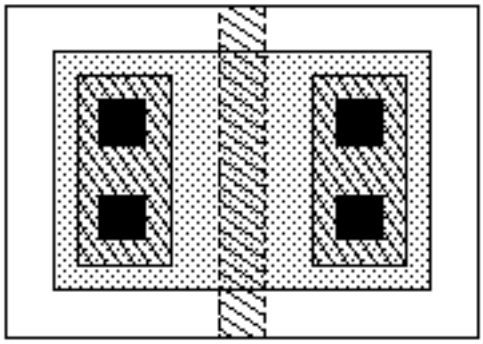
   The information shown above is correct for the tutorial pcell.

4. Close the window by choosing *File – Close Window*.

   You test the repetition parameter by redrawing the instance in the `mypcell` layout window. The width of the instance is already set to 5.

5. Redraw the instance by choosing *Window – Redraw*.

The instance now looks like this:



You have completed testing the second parameter: repetition for contacts.

Understanding the Number of Repetitions Expression

You used a Cadence® SKILL language expression for the number of repetitions of contacts.This expression is typical of expressions used in parameterized cells. The entire expression

```
fix(width-1)/pcStep
```

means that the number of repetitions is an integer equal to 1 micron less than the value of `width` (the stretch line parameter), divided by the stepping distance (2), where

`fix`	is a SKILL function that converts nonintegers to integers by rounding the number down. The number of repetitions must be an integer because you want to repeat whole objects.
`width-1`	lets you control the area within which you want to fit the repeat group. You created a stretch line parameter named `width` based on the width of the p-diffusion. As the diffusion stretches upward and the contacts repeat within it, you want the final contact to be spaced away from the edge of the diffusion by at least 1 micron. So the available space for the contacts is width minus 1.
`pcStep`	is a system variable known to the pcell compiler that represents the stepping distance for this repetition parameter. You defined the stepping distance as 2.

Testing the Expression in the CIW

You can use the CIW to simulate how the pcell program evaluates the expression. You are using the CIW only to test the expression; none of the pcell variables is affected.

1. In the CIW, type `width = 5.5` and press `Return`.

   You see `5.5` in the output region.

2. In the CIW, type `pcStep = 2` and press `Return`.

3. In the CIW, type `(width-1)/pcStep` and press `Return`.

   You see `2.25` in the output region. The number of repetitions must evaluate to an integer, however, so you need to add the `fix` function to round the value down to the nearest integer.

4. Round the number of repetitions down to an integer by doing the following:

   - In the output region of the CIW, click `(width-1)/pcStep`.

     The entire line is highlighted in black and appears in the input region of the CIW.

   - At the beginning of the line in the input region, click.

     The blinking text insertion bar appears in front of the expression.

   - Type `fix`.

     The CIW input region now contains the complete expression:

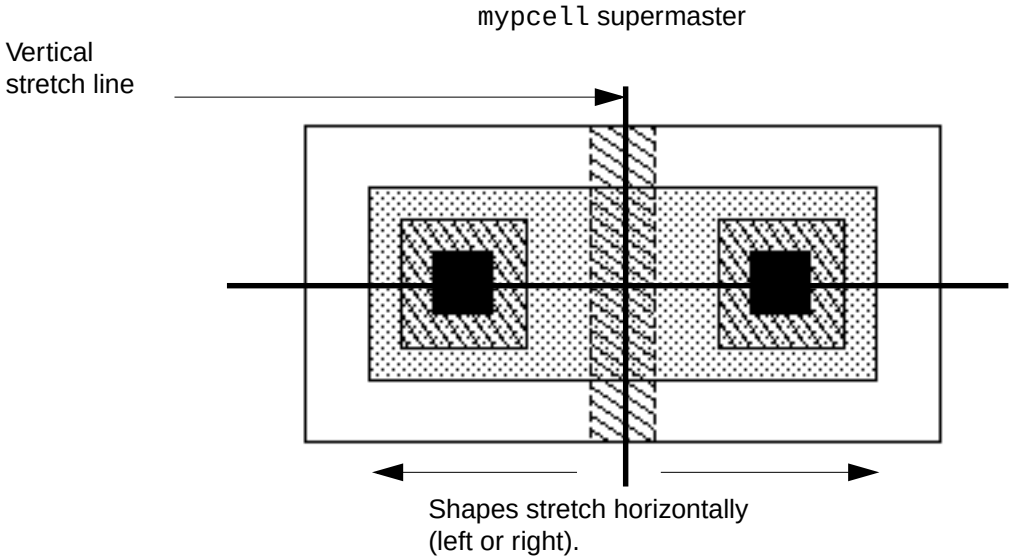
     ```
     fix(width-1)/pcStep
     ```

   - Press `Return`.

You see 2 in the output region. If the value of the stretch line parameter were set to `5.5`, the contacts would be repeated two times. There is no decimal point because the number is now an integer.

Defining the Third Parameter: Stretch Line for Gate Length

In this section, you create a vertical stretch line parameter to stretch the transistor horizontally along the X axis. The objects intersected by the stretch line will be the `poly1` gate, p-diffusion, and the nwell surrounding the transistor.



Unlike the repeating contacts you created previously, you want the `poly1` shape to both repeat and stretch. So this time you set an option in the Stretch form to allow all intersected objects that repeat along the X axis (the `poly1` gate) to stretch.

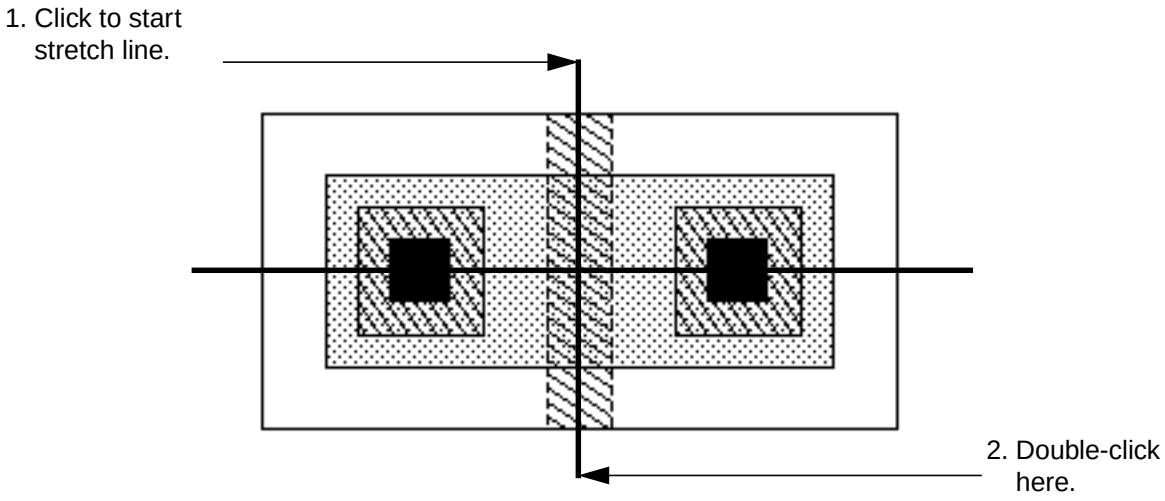
1. In the `mypcell` layout window, choose *Pcell – Stretch – Stretch in X*.

   The system prompts you to draw a vertical line to control horizontal stretch. You will create the stretch line down the middle of the `poly1` gate.

2. Define the vertical stretch line by doing the following:

   - Click the top edge of the `poly1` gate.

- Move the cursor to the bottom edge and double click or press `Return`.



The Stretch in X form appears.

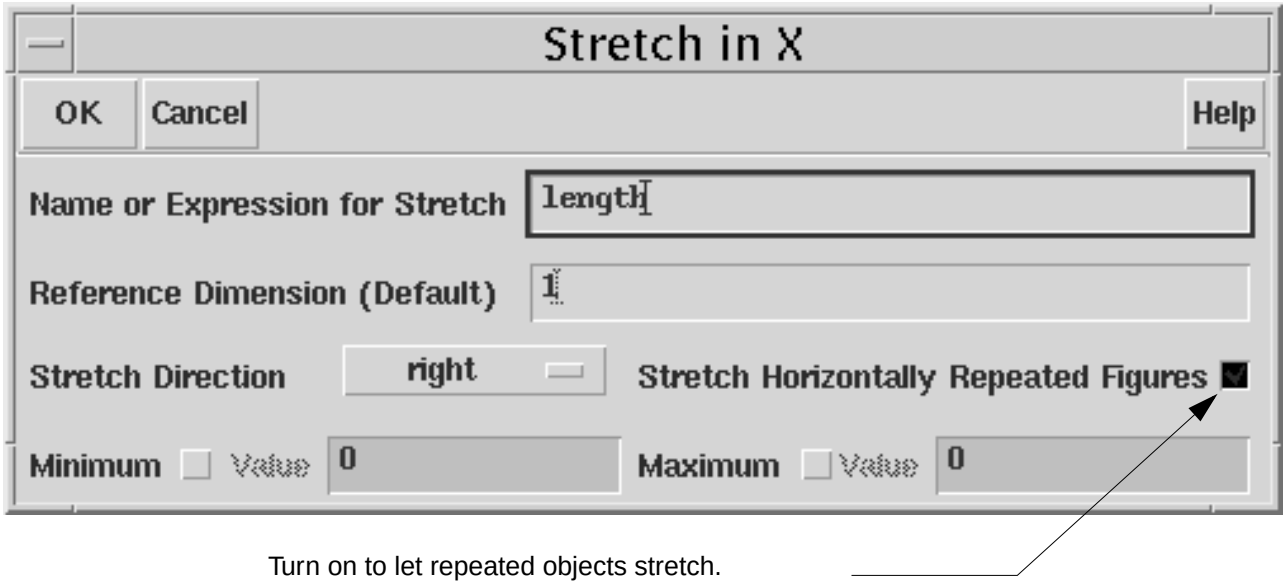
3. For *Name or Expression for Stretch*, type `length`.

The word `length` is now the parameter name for the stretch line through the gate. When you place an instance, the Create Instance form contains a field where you can type a value for `length`.

Reference Dimension is set to `1` by default. This value represents the smallest dimension crossed by the stretch line, in this case, the length of the `poly1` gate. You will use the default.

4. Turn on the *Stretch Horizontally Repeated Figures* option.

Turning this option on lets you both stretch and repeat the `poly1` gate.



5. Make sure *Stretch Direction* is set to *right* and click *OK*.

Objects intersected by the stretch line can stretch to the right. You have completed the third parameter: the stretch line for the `poly1` gate.

Compiling and Testing the Length Stretch Line Parameter

Now you will compile again, check the Parameter Summary, and place an instance to test the stretch line for the `poly1` gate.

1. In the `mypcell` layout window, recompile the pcell by choosing *Pcell – Compile – To Pcell*.
2. Check the CIW to make sure your pcell compiled without errors.
3. Check the Parameter Summary by choosing *Pcell – Parameters – Summarize*.

   A window opens, showing the following information:

```
Parameters defined in this parameterized cell:
    length width

Stretch
-------
Stretch Type: Vertical
Name or Expression for Stretch:  width
Stretch Direction:  up
```

```
Reference Dimension (Default):  3.000000

Stretch Type: Horizontal
Name or Expression for Stretch:  length
Stretch Direction:  right
Reference Dimension (Default): 1.000000


Repetition
----------
Objects Repeated in:   Y
Number of Objects in this group:  2
Stepping Distance:  2
Number of Repetitions:  (fix((width - 1)) / pcStep)
Dependent Stretch:
Adjustment to Stretch:
```

The information shown above is correct for the tutorial pcell.

4. Close the list window by choosing *File – Close Window*.

You'll test the new stretch parameter by creating an instance and changing the value of *length*.

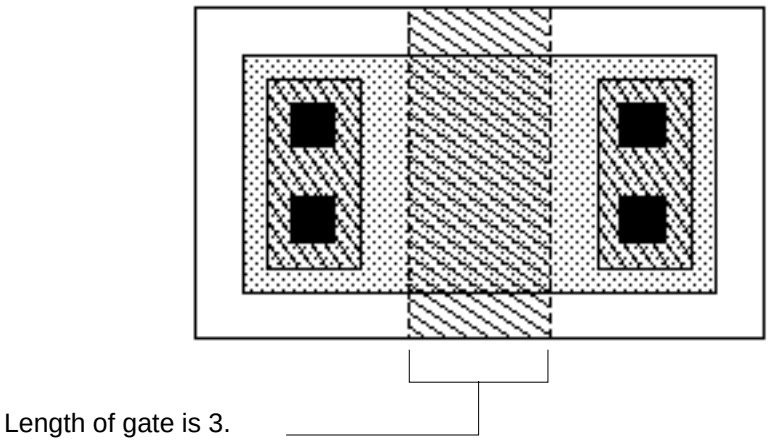
5. Create the instance by doing the following:

- With the cursor in the `pcelltest` layout window, open the Create Instance form by pressing `i`.

  The *Cell* field is still set to `mypcell`.

- Set *width* to 5.
- Set *length* to 3.
- Click in the `pcelltest` layout window.

The instance looks like this:



You have completed testing the third parameter: stretch line for gate length.

Defining the Fourth Parameter: Repeating the Gate

In this section, you create another repetition parameter to repeat the `poly1` gate along the X axis. For this repetition parameter, you will enter an expression that refers to the *length* parameter you created earlier (stretch line for gate length).

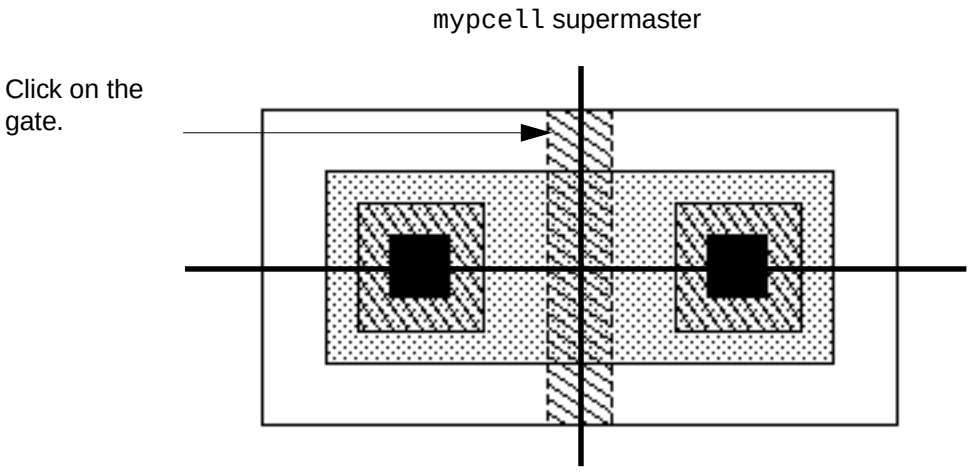
In this tutorial, you are defining parameters before using them in expressions. However, it is not necessary to do so. The Pcell compiler evaluates all parameters when you compile, not before, so you can define an expression that refers to a parameter before you define the parameter, as long as you define the parameter before you compile.

Note: If you define an expression containing a parameter that is not defined and then compile, the Pcell software treats the undefined parameter as a parameter and displays it on the Create Instance form.

1. In the `mypcell` layout window, choose *Pcell – Repetition – Repeat in X*.

   You are prompted to point at the shapes for the repeat group.

2. Select only the `poly1` gate and open the Repeat in X form by clicking on the gate and pressing `Return`.



The gate is highlighted and the Repeat in X form appears.

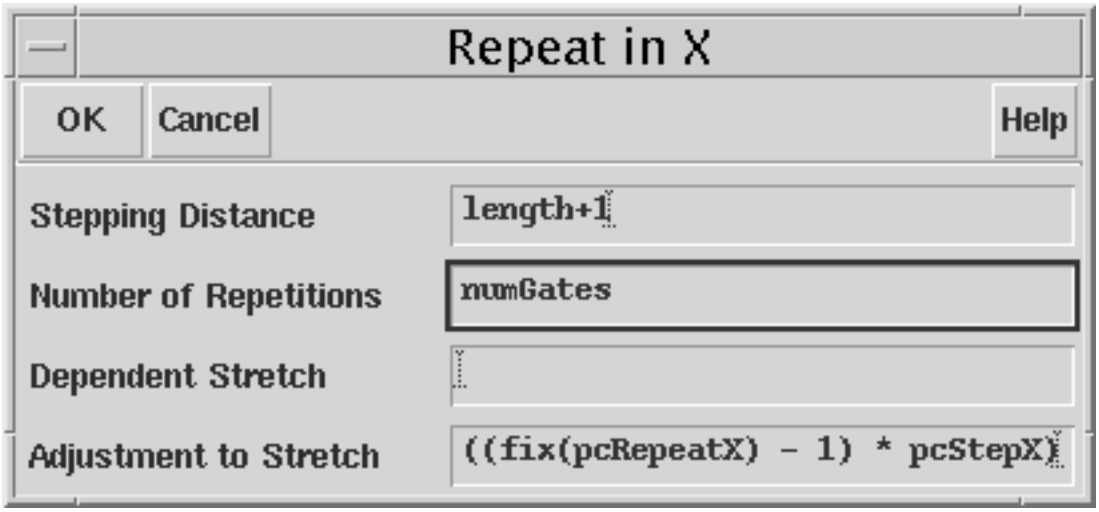
You will define the distance between gates (the stepping distance) as an expression based on the `length` parameter you defined earlier for the length of the gate, as follows: `length+1`

where `length` is also the reference dimension for stretching the gate horizontally. The repeated gates will be spaced 1 micron apart no matter how much you stretch the `poly1` gate with the `length` parameter.

3. For *Stepping Distance*, type `length+1`.

You'll define the number of gates as a parameter named `numGates` so you can type in a value when you place an instance.

4. For *Number of Repetitions,* type `numGates`.



5. Click *OK*.

After you compile and place an instance of the pcell, the Create Instance form contains a field named *numGates* to let you type the number of gates.

You have completed defining the fourth parameter: a repeat group for the `poly1` gate.

Compiling and Testing the Gate Repeat Group Parameter

Now you will compile again, check the Parameter Summary, and place an instance to test the repeat group for the `poly1` gate.

1. In the `mypcell` layout window, recompile the pcell by choosing *Pcell – Compile – To Pcell*.
2. Check the CIW to make sure your pcell compiled without errors.
3. Check the Parameter Summary by choosing *Pcell – Parameters – Summarize*.

   A window opens, showing the following information:

```
Parameters defined in this parameterized cell:
    length width numGates

Stretch
-------
Stretch Type: Vertical
Name or Expression for Stretch:  length
Stretch Direction:  right
Reference Dimension (Default): 1.000000
```

```
Stretch Type: Horizontal
Name or Expression for Stretch:  width
Stretch Direction:  up
Reference Dimension (Default):  3.000000


Repetition
----------
Objects Repeated in:   Y
Number of Objects in this group:  2
Stepping Distance:  2
Number of Repetitions:  (fix((width - 1)) / pcStep)
Dependent Stretch:
Adjustment to Stretch:

Objects Repeated in:   X
Number of Objects in this group: 1
Stepping Distance: (length + 1)
Number of Repetitions: numGates
Dependent Stretch:
Adjustment to Stretch:
```

The information shown above is correct for the tutorial pcell.

4. Close the window by choosing *File – Close Window*.

You'll test the new stretch parameter by creating an instance and changing the value of *length*.

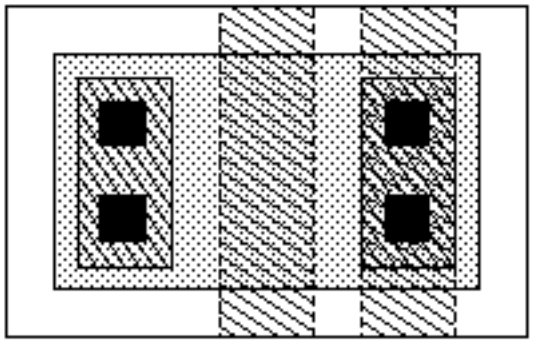
5. Create the instance by doing the following:

- Open the Create Instance form by pressing `i` with the cursor in the `pcelltest` layout window.

  The *Cell* field is still set to `mypcell`.

- For *length*, type 2.
- For *width*, type 5.
- For *numGates*, type 2.
- Click in the `pcelltest` layout window.

The instance looks like this:



You have completed testing the fourth parameter: repeat group for `poly1` gate.

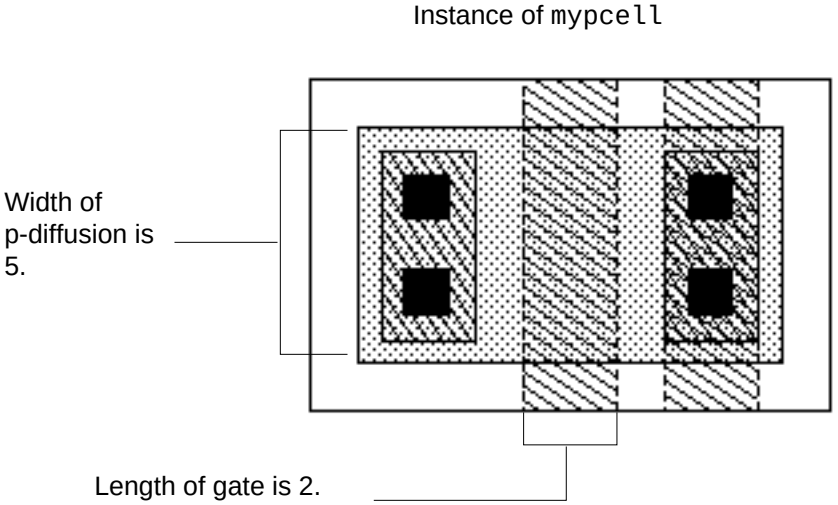
This instance is not correct yet because one of the gates is on top of the right-hand contacts. You need to fine-tune the parameters so that there is room for the gates to repeat: the diffusion needs to expand and the contacts and metal layer need to move to the right.

Reviewing Parameter Values for the Instance

Here is how the values of the parameters were set when you placed the instance:

- You entered 2 for the `length` parameter to stretch the gate length from 1 to 2 microns. The increase in the `length` parameter also stretched the p-diffusion and the nwell.
- You entered 5 for the `width` parameter to set the p-diffusion width to 5 microns.
- You entered 2 for the *numGates* parameter to create two gates 1 micron apart.

- The system computed the repetition parameter for the number of contacts using the expression you entered. The expression is based on the value of the `width` parameter and results in two contacts.



Although the `length` parameter stretched the `poly1` gate, p-diffusion, and nwell each by 1 micron, it did not stretch the p-diffusion and nwell enough to accommodate the second gate created by the `gate` repeat parameter. Now you need to find a way to

- Stretch the p-diffusion and nwell to make room for additional gates
- Move the contacts and metal layer to the right, out of the way

Defining a Dependent Stretch Line

You can correct the tutorial pcell by adding a parameter called a *dependent stretch line*. A dependent stretch line causes the shapes it affects to stretch based on a repeat group. In this case, you will define a dependent stretch line to make the size of the p-diffusion and nwell and the position of the right-hand contacts dependent on the number of gates.

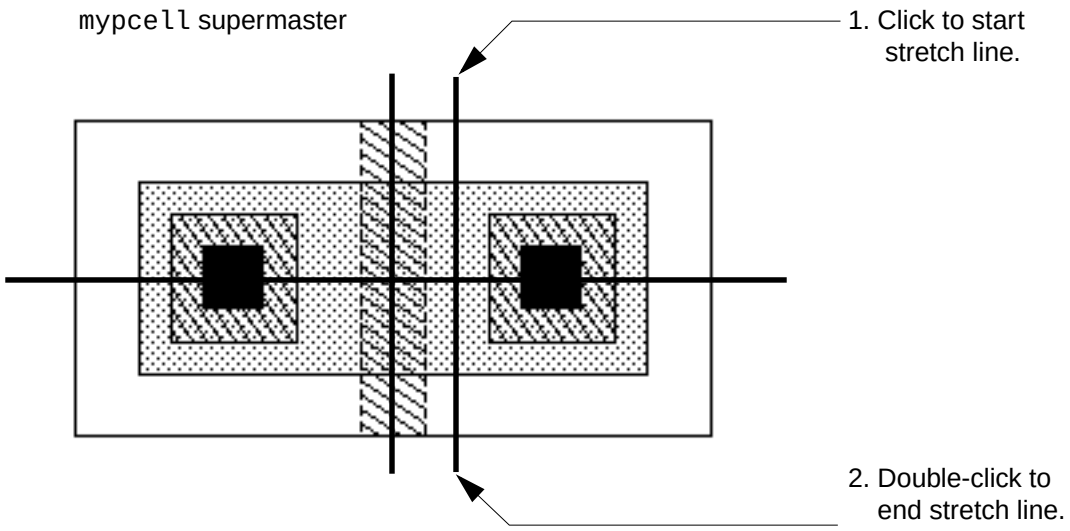
This is a two-part process:

- Define a new stretch line through the p-diffusion and nwell
- Modify the gate repeat group to assign the new stretch line as dependent

Defining the Dependent Stretch Line

Now you will add a dependent stretch line to the `mypcell` supermaster to stretch the p-diffusion and nwell rectangles.

1. In the `mypcell` window, choose *Pcell – Stretch – Stretch in X*.
2. Define a vertical stretch line through only the p-diffusion and the nwell by clicking as shown below.



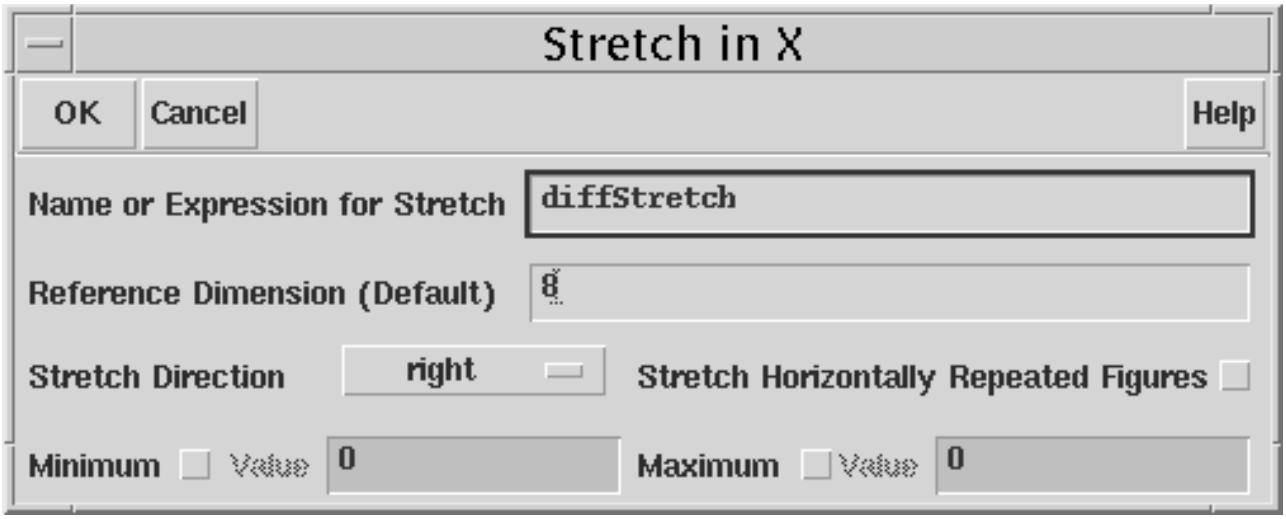
   The Stretch in X form appears.

   Because the line intersects only the p-diffusion and nwell, only these objects will stretch. This means the contact and metal1 shapes to the right of the stretch line do not stretch but move to the right instead.

3. For *Name or Expression for Stretch*, type `diffStretch`.

   Stretch Direction is set to *right*, so the objects will stretch to the right.

Reference Dimension is set to its default. The system will not use this value because, in the next section, you make this parameter a dependent stretch line to cause the layout editor to calculate the amount to stretch based on the size of the gate repeat group.



4. Click *OK*.

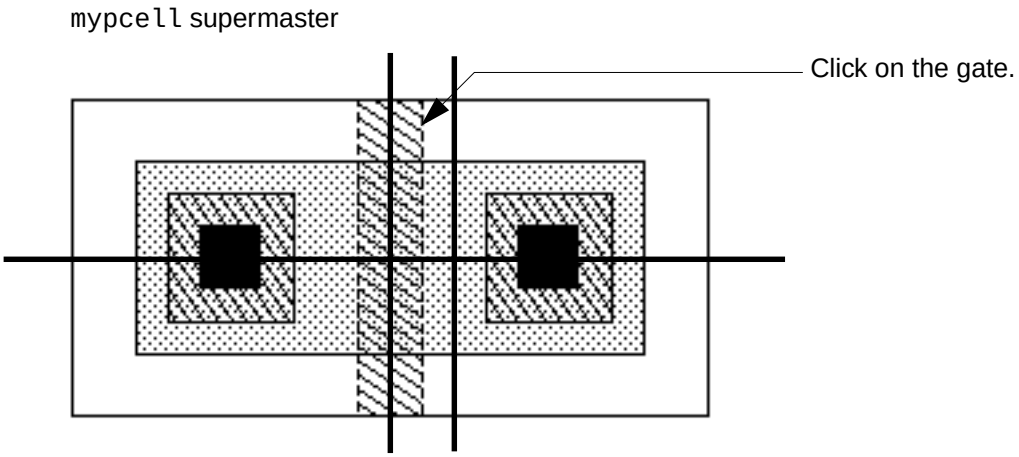
Modifying the Gate Repeat Group

You want the new stretch line to let the p-diffusion and nwell stretch based on how much space is needed to repeat gates, so you must make it depend on the gate repetition parameter.

1. In the `mypcell` window, choose *Pcell – Repetition – Modify*.

   You are prompted to point at a shape in the repeat group to modify.

2. Identify the repeat group by clicking on the `poly1` gate.

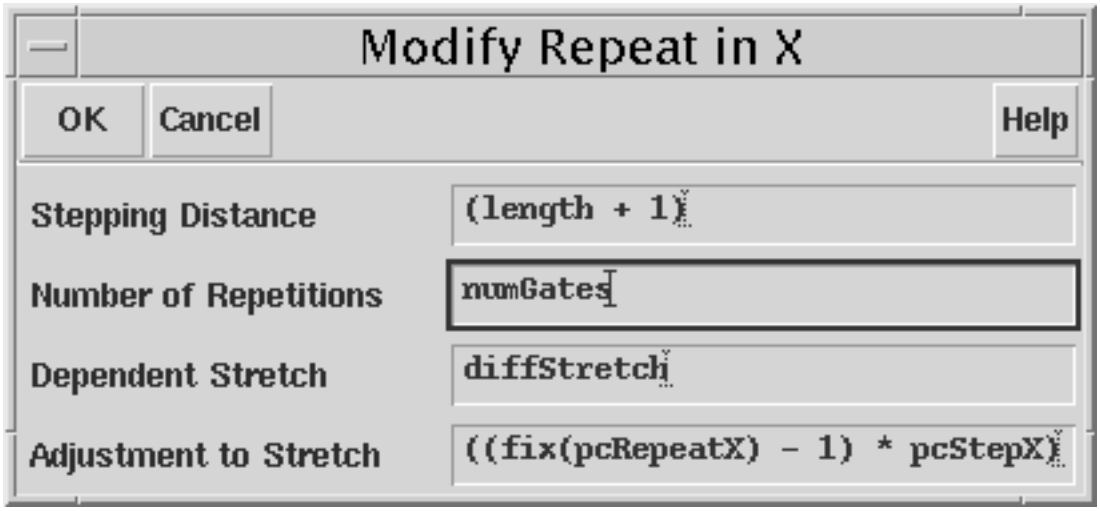


3. Indicate that you do not want to add more shapes to the group by double-clicking anywhere in the cellview window or by pressing `Return`.

   The Modify Repeat in X form appears.

4. For *Dependent Stretch*, type `diffStretch`.

   Now the `diffStretch` parameter is dependent on the gate repetition parameter.



5. Click *OK*.

You have completed defining the dependent stretch line.

Compiling and Testing the Dependent Stretch Line

Now you will compile again, check the Parameter Summary, and place an instance to test the dependent stretch line for the gate repeat group.

1. In the `mypcell` layout window, recompile the pcell by choosing *Pcell – Compile – To Pcell*.
2. Check the CIW to make sure your pcell compiled without errors.
3. Check the Parameter Summary by choosing *Pcell – Parameters – Summarize*.

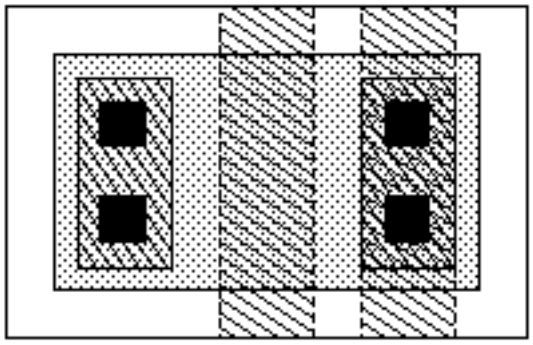
   A window opens, showing the following information:

```
Parameters defined in this parameterized cell:
    width length numGates

Stretch
-------
Stretch Type: Horizontal
Name or Expression for Stretch:  diffStretch
Stretch Direction:  right
Reference Dimension (Default): 8.000000

Stretch Type: Horizontal
Name or Expression for Stretch:  length
Stretch Direction:  right
Reference Dimension (Default): 1.000000

Stretch Type: Vertical
Name or Expression for Stretch:  width
Stretch Direction:  up
Reference Dimension (Default):  3.000000


Repetition
----------
Objects Repeated in:   Y
Number of Objects in this group:  2
Stepping Distance:  2
Number of Repetitions:  (fix((width - 1)) / pcStep)
Dependent Stretch:
Adjustment to Stretch:

Objects Repeated in:   X
Number of Objects in this group: 1
Stepping Distance: (length + 1)
Number of Repetitions: gate
Dependent Stretch: diffStretch
Adjustment to Stretch: ((fix(pcRepeatX) - 1) * pcStepX)
```

   The information shown above is correct for the tutorial pcell.

4. Close the window by choosing *File – Close Window*.

You will test to verify whether the dependent stretch line fixed the problem. Previously, the repeated gate was on top of the contacts, like this:



The dependent stretch line should make the contacts and metal layer move to the right and the p-diffusion and nwell stretch to accommodate the additional gate.

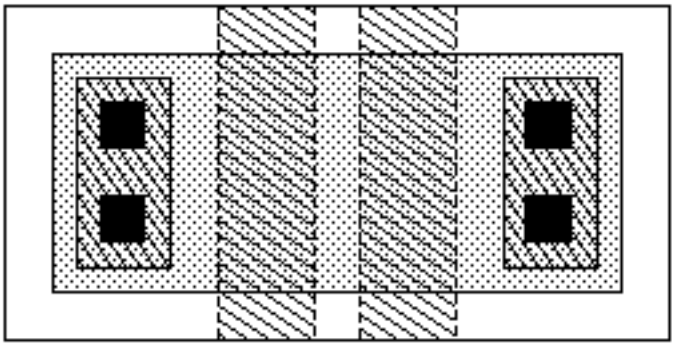
5. Create the instance by doing the following:

- With the cursor in the `pcelltest` layout window, open the Create Instance form by pressing `i`.

  The *Cell* field is still set to `mypcell`.

- For *length*, type 2.
- For *width*, type 5.
- For *numGates*, type 2.
- Click in the `pcelltest` layout window.

  The instance looks like this:



You have completed creating the parameterized cell.

Saving the Pcell and Exiting the Software

To complete the tutorial, you will close the `pcelltest` layout window, save your pcell parameters to a file, and save and close the `mypcell` layout window.

In this section, you learn to

- Save parameters to a file for future reference
- Save the pcell

1. Close the `pcelltest` layout window by doing the following:

- In the `pcelltest` layout window, choose *Window – Close*.

  The Save Changes dialog box appears. You do not need to save the results of your testing.

- In the Save Changes dialog box, click *No*.

2. Save your parameter definitions in a file so that you can use them again by doing the following:

- In the `mypcell` layout window, choose *Pcell – Parameters – Summarize*.

  A window opens, showing the all parameters for the pcell. This is a temporary file.

- Choose *File – Save As*.

  The Save As form appears.

- For *File Name*, type `mypcell.param`.

- Click *OK*.

The system creates a text file named `mypcell.param` in your current working directory (`cell_design`).

- In the window, choose *File – Close Window.*

  The window closes.

3. Save the supermaster pcell and close the window by doing the following:
   - In the `mypcell` layout window, choose *Design – Save.*
   - Choose *Window – Close*.

You have completed closing the `pcelltest` layout window, saving your pcell parameters to a file, and saving and closing the `mypcell` layout window.

Summary

In this chapter you learned to create and modify pcells. Specifically, you

- Defined stretch line parameters
- Defined repeat group parameters
- Defined a dependent stretch line
- Modified a repeat group parameter for the dependent stretch line
- Examined the Parameter Summary after each change
- Compiled and tested your parameters
- Saved your parameters to a file
- Saved the supermaster of the pcell