5

Creating Layout Objects Using ROD

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 by using the *Create Rectangle* command. In this section you

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

You create the rectangle in a new cellview.

1. Choose *File – Open*.

   The Open File form appears.

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

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. Close the Create Rectangle form.

Note: If you had created the same rectangle using SKILL code, you would have typed the following in the CIW. You do not have to type this, this is just an example:

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

Examining the ROD Rectangle

To examine the data about the ROD rectangle,

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

   ```
   rect = rodGetObj("rect" geGetEditCellView())
   ```

```
rect~>??
```

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

```
("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

Another way to view the data of the rectangle is through the Edit Properties form. Any changes you make to the rectangle are reflected in this 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* in the Edit Properties banner.

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

4. Compare the values of *upperLeft* and *lowerRight* in the *System handle* field. They should be the same as the coordinates in the *Attribute* fields.

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

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

   The Stretch form appears.

3. Click and drag the right edge of the rectangle to approximately X=11, Y=10.
4. Press `Return` to end the stretch.
5. Click *Cancel* to close the Stretch form.
6. Click on the rectangle.

   The *Right* value should have changed from *9* to *11*.

Creating a ROD Polygon

In this section, you create a ROD polygon. After you create the polygon, you examine and then 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`.

5. Enter the following points in the cellview at these XY coordinates. When you enter the last point, X = 13, Y = 11, either double click or press *Return*:

   First click: X = 11, Y = 11

   Second click: X = 11, Y = 7

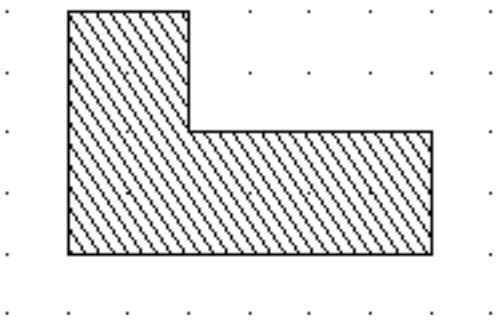
   Third click: X = 17, Y = 7

   Fourth click: X = 17, Y = 9

   Fifth click: X = 13, Y = 9

   Sixth click: X = 13, Y = 11

   The completed polygon.



Note: If you had created the same polygon using SKILL code, you would have typed the following in the CIW. You do not have to type this, this is just an example:

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

Examining the ROD Polygon

Now you can examine the data about the ROD polygon.

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

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

2. Examine the data 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"
```

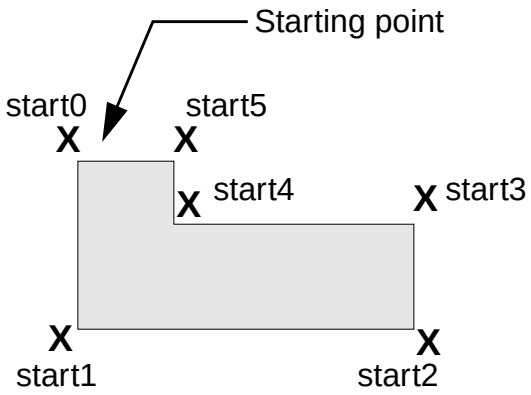
```
"length1" "start1" "mid1" "end1" "length2"
"start2" "mid2" "end2" "length3" "start3"
"mid3" "end3" "length4" "start4" "mid4"
"end4" "length5" "start5" "mid5" "end5"
"lengthLast" "startLast" "midLast" "endLast"
)
)
```

About ROD Handles

Now that you have created a ROD polygon and rectangle, you can examine and use their ROD attributes in your design. Important attributes of ROD objects are their handles. Handles are used to store points, calculations, and other information. In the Edit Properties form you can view the handle data in the ROD fields.

1. Select the polygon.
2. Click *ROD* in the Edit Properties banner.

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 updated *Points* values you set in *Attribute*.

Creating User-Defined Handles

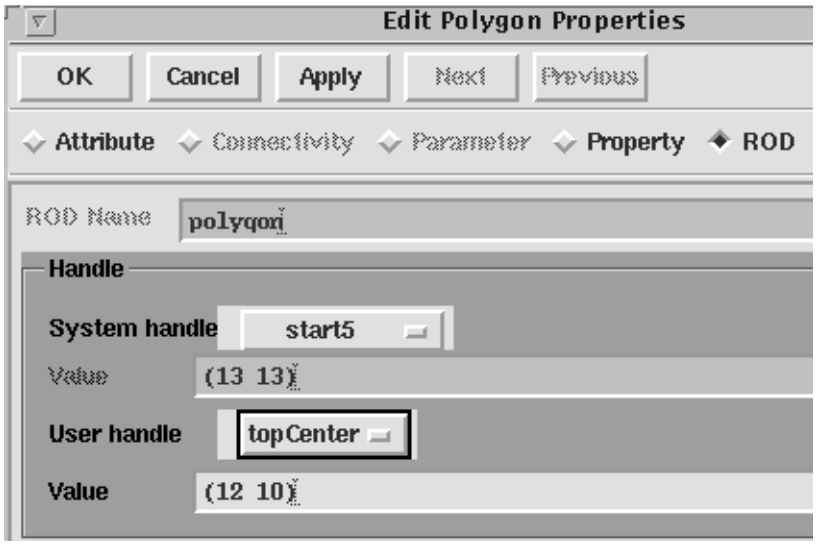
In this section, you create a user-defined handle for the polygon. When you define a user-defined handle, you specify a name and assign a value to it. The values of user-defined handles are stored in the database.

To create a user-defined handle,

1. In the CIW, type

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

2. Note the change in the *User handle* field. You may have to deselect and then reselect the polygon to refresh the form. The value should reflect what you set to create the user handle.



Aligning the ROD Polygon and Rectangle

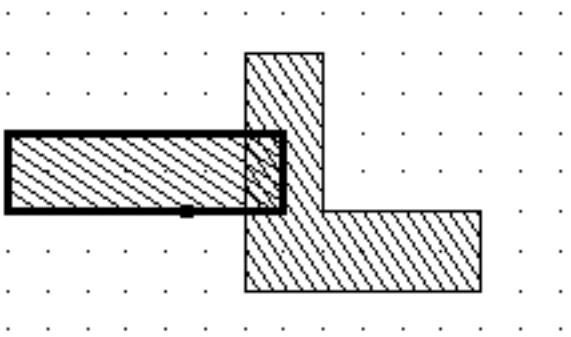
An important feature of ROD is the ability to specify the position of one named object in relation to another named object. This is called *relative alignment*. Usually, you align objects by specifying a point handle on each object. You can also specify the distance between the two objects in the direction 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. The reference object is the rectangle and the reference handle is *centerRIght*. The align object is the

polygon and the align handle is *topCenter*. Remember, *topCenter* is the user handle you just created.

1. In the CIW, type

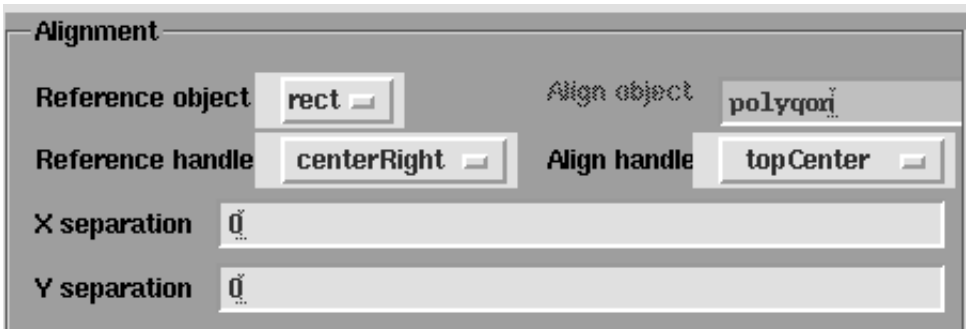
```
rodAlign(
?alignObj polygon
?alignHandle "topCenter"
?refObj rect
?refHandle "centerRight"
)
```



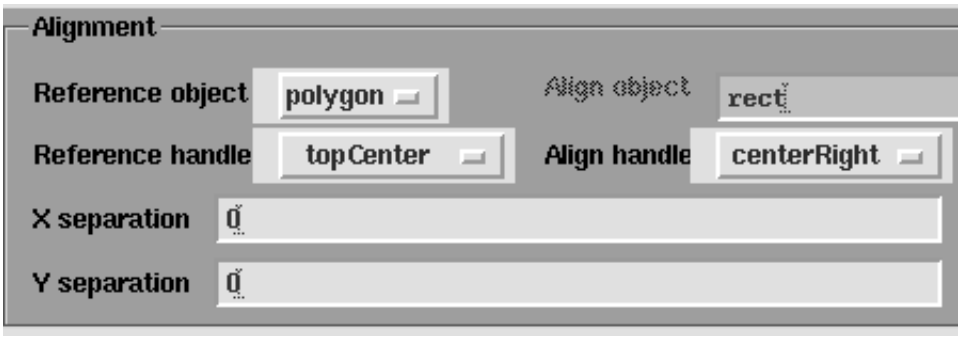
The rectangle's *centerRight* handle and the polygon's *topCenter* user handle are aligned.

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 data changes. Now the rectangle is the align object and the polygon is the reference object because the rectangle is the selected object.

Editing the Aligned Objects

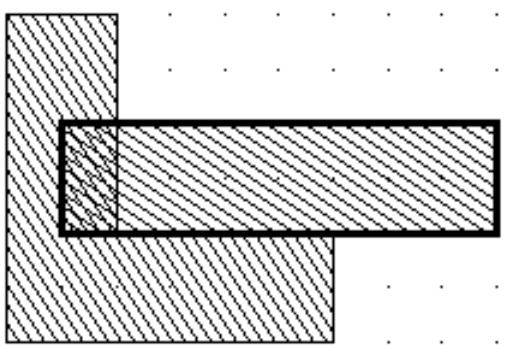
To demonstrate how aligned objects stay *relatively aligned*, in the next steps you

- Change the *Align handle* position of the rectangle

- Move the polygon
- Stretch the rectangle

To change the *Align handle*,

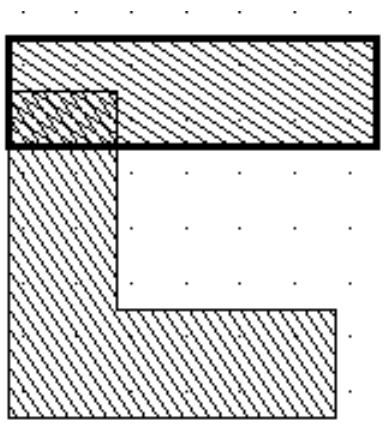
1. Select the rectangle.
2. Change the *Align handle* field from *centerRight* to *centerLeft*.
3. Click *Apply.*



The rectangle's *centerLeft* handle and the polygon's *topCenter* user handle 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.

To move the polygon and then stretch the rectangle,

1. Choose *Edit – Move.*

   The Move form appears.

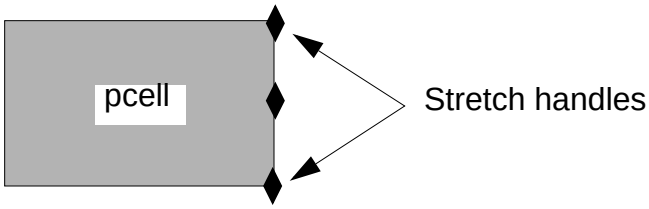
2. Select the polygon.
3. Move it anywhere in the window. The rectangle should stay aligned to the polygon.
4. Click *Cancel* to close the Move form.
5. Deselect all objects by clicking in an empty area of the window.
6. Choose *Edit – Stretch*.

7. Click on the right side of the rectangle.
8. Move the cursor two grid spaces to the right.
9. Press `Return`.

   The rectangle stretches to the right and retains the alignment with the polygon.
10. Close the window.

Stretching a Pcell

This section introduces you to editing a stretchable pcell. A *stretchable pcell* is a Cadence® SKILL-based pcell created with one or more stretch handles assigned to one or more of its parameters. A *stretch handle* is a named set of coordinates assigned to a specific parameter of the pcell. Stretch handles look like small diamonds.



You can change the value of pcell parameters that are associated with stretch handles by selecting the handles and dragging them. A handle can stretch 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.

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 cellview 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 is saved to the `tutorial` library.

7. Choose *Design – Open.*

The Open File form appears.

8. Type the following to close the `master` mux2gs and open the `tutorial` mux2gs:

Library Name	`tutorial`
Cell Name	`mux2gs`

Confirm in the layout window banner which cellview is open: it should display *tutorial mux2gs layout*.

The mux2gs contains several design rule errors. You confirm this by running the Design Rule Checker (DRC).

1. Choose *Verify – DRC*.

The DRC form appears.

2. Click *OK*.

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
```

To fix these errors you need to

- Stretch the inverter's height (a stretchable pcell) so it aligns with the neighboring cells
- Add a third column of contacts
- Rerun DRC to check your work

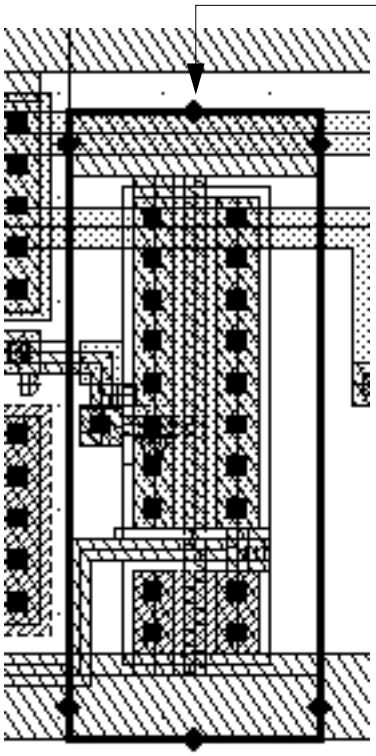
Before you fix the errors, delete the markers. Use *Verify – Markers– Delete All*.

Stretch the Inverter

1. Choose *Edit – Stretch.*

2. Click on the top center stretch handle.

.



The top center stretch handle.

3. Move the cursor up until the text display to the right of the inverter shows *Cell Height = 36u*.

   You may have to move the cursor slowly during the stretch because it takes a few seconds for the text display calculations to display. Also, it may take you a couple of stretch clicks to get the cell height to exactly 36 microns.

4. Click to end the stretch.

5. Cancel the *Stretch* command.

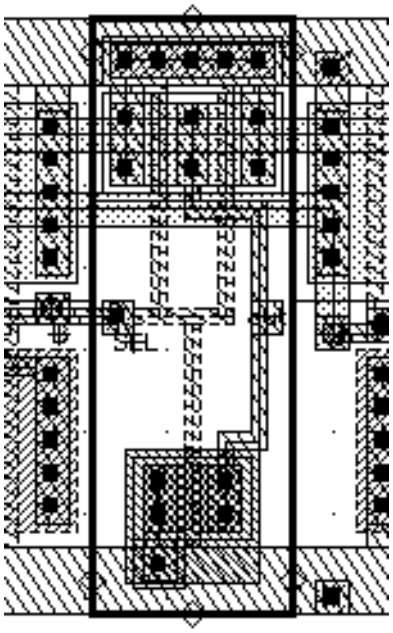
Add a Third Finger of Contacts

You add the contacts using the Edit Properties form.

1. Select the inverter.
2. Choose *Edit* – Properties.
3. Click *Parameter* in the Edit Properties banner.
4. Change *pMos Gate Width* to `10.0u`.
5. Change *pMos fingers* to 2.
6. Change *Supply Width* to 4u.
7. Click *Apply.*

The Edit Properties fields should look like this:

The stretched inverter should look like this:



Run DRC to Check Your Work

1. Choose *Verify – DRC*.

   The DRC form appears.

2. Click *OK*.

   The DRC summary in the CIW should report zero errors found. If any errors are reported, repeat the steps in "Stretching a Pcell" on page 5-17.

Creating a Path through a Multipart Path Chop Hole

The design is missing an output path. You create the output path and connect it to the Y pin to the right of the multipart path guard ring. To do this, you must chop a hole in the guard ring.

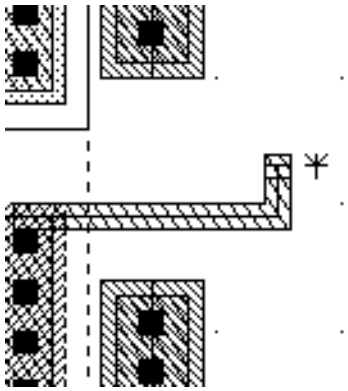
1. Select the entire guard ring.
2. Choose *Edit – Other – Chop.*
3. Click approximately X= 64, Y= 30 and X=71, Y=21.

   The chop hole is complete.
4. Deselect all objects.
5. Make sure *Gravity* is off in the Layout Editor Options form.

Create the path.

1. Click on *metal1* in the LSW.
2. Choose *Create – Path*.

3. Draw the path as shown below.



The completed path.

4. Cancel the *Path* command.
5. Save and close your design.

Aligning Hierarchical ROD Objects

In this section, you create two instances of a ROD object and align the contact in one instance to a contact in the other instance.

Hierarchical ROD alignment is an important aspect of ROD functionality. This functionality aligns a named object by a point handle on an object to a specific point or to a point handle on a reference object. You can align objects that are at different levels of hierarchy as 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.

For this exercise, you create the instances in a new cellview.

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. Choose *Create – Instance.*

   The Create Instance form appears.

5. Type the following:

Library	`ROD`
Cell	`ptran`
View	`layout`
Names	I1

6. Click on X=2.5, Y=0 to place the I1 instance.

7. Type the following to create the second instance.

Library	ROD
Cell	ptran
View	layout
Names	I2
Width	3

8. Click on X=10, Y=0 to place the I2 instance.
9. Cancel the *Create Instance* command.

Before you align the instances, you type several commands in the CIW to obtain the cellview and ROD object IDs.

1. Type in the CIW to obtain IDs:

```
cv=geGetEditCellView()
cont1Id=rodGetObj("I1/rightcont" cv)
cont2Id=rodGetObj("I2/leftcont" cv)
```

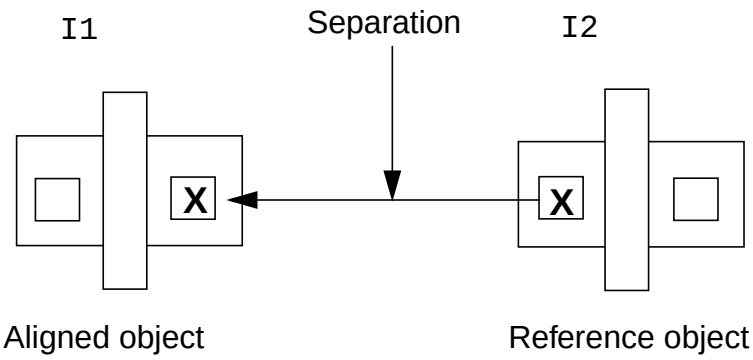
   The database IDs for the cellview and ROD objects display in the CIW.

2. Type in the CIW to align the instances:

```
rodAlign(
?alignObj cont1Id
?alignHandle "centerRight"
?refObj cont2Id
?refHandle "centerLeft"
```

```
?xSep -2.0
)
```

The *centerRight* contact in instance I1 is aligned to the *centerLeft* contact in I2 at a separation of -2. The negative number means the reference object is right of the aligned object.



3. 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. This 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 the ROD handle information 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 when you
  - Ran DRC to determine errors
  - Corrected DRC errors by
    - Stretching the ROD pcell inverter
    - Editing the parameters of the ROD pcell inverter
  - Created a chop hole for the output path
  - Created a *metal1* output path
  - Ran DRC to check the changes you made
- Aligned two ROD object instances