Custom Compiler Training -Layout

1 Design Entry

Learning Objectives

During this lab, you will create a mask layout design of a CMOS logic gate using design rule-guided data creation and editing functions.

After completing this lab, you should be able to:

- Set and use grids for drawing shapes.
- Create layout using design rule-guided data creation functions.
- Use different usage models of data creation functions.
- Edit using design rule-guided editing functions.
- Use Object Layer Panel to select LPP for shapes creation.
- Create Pins.
- Create Labels.
- Create Boundaries.

Lab Duration:
30 minutes

Introduction

The purpose of this lab is to help you understand the fundamental data creation and editing functions of Custom Compiler. This lab uses generic process design rules shipped with the generic PDK kit, to design the simple inverter mask layout.

Before preceding to layout the inverter mask design, you need to apply a set of design rules during the design creation.

A. Generic Process Design Rules

This section lists the process design rules to be followed. Common design rules which are applicable for the PMOS, NMOS transistors and other general rules are defined under the General Design Rules section. Rules specific to PMOS and NMOS transistors are defined under their respective sections.

General Design Rules

Rule	Design Rule Description		Design RuleValue
1	Minimum Enclosure of diff by NWELL	≥	0.18µm
2	Minimum Enclosure of diff within Poly	≥	0.1 µm
3	Minimum Width for Contact	≥	0.06 µm
4	Minimum Spacing for Contact	≥	0.08 µm
5	Minimum Enclosure of Contact within Diffusion	≥	0.01/0.03 µm
6	Minimum Spacing from Poly to Contact	≥	0.045 µm
7	Minimum Enclosure of Contact within M1	≥	0
8	Minimum End of Line Enclosure of Contact within M1	≥	0.03 µm
9	Minimum width for M1	≥	0.07 µm
10	PPLUS should not overlap NPLUS (can abut)		

PMOS Design Rules

Rule	Design Rule Description		Design Rule Value
1	Minimum Enclosure of PPLUS within NWELL	≥	0.09 µm
2	Minimum Enclosure of Diffusion within NWELL	≥	0.18 µm
3	Minimum Enclosure of Diffusion within PPLUS	≥	0.09µm
4	Minimum Enclosure of Poly within PPLUS	≥	0.09µm
5	Minimum Enclosure of Contact within PPLUS	≥	0.01/0.03 µm
6	Minimum Spacing of PPLUS to Diffusion	≥	0.06µm

NMOS Design Rules

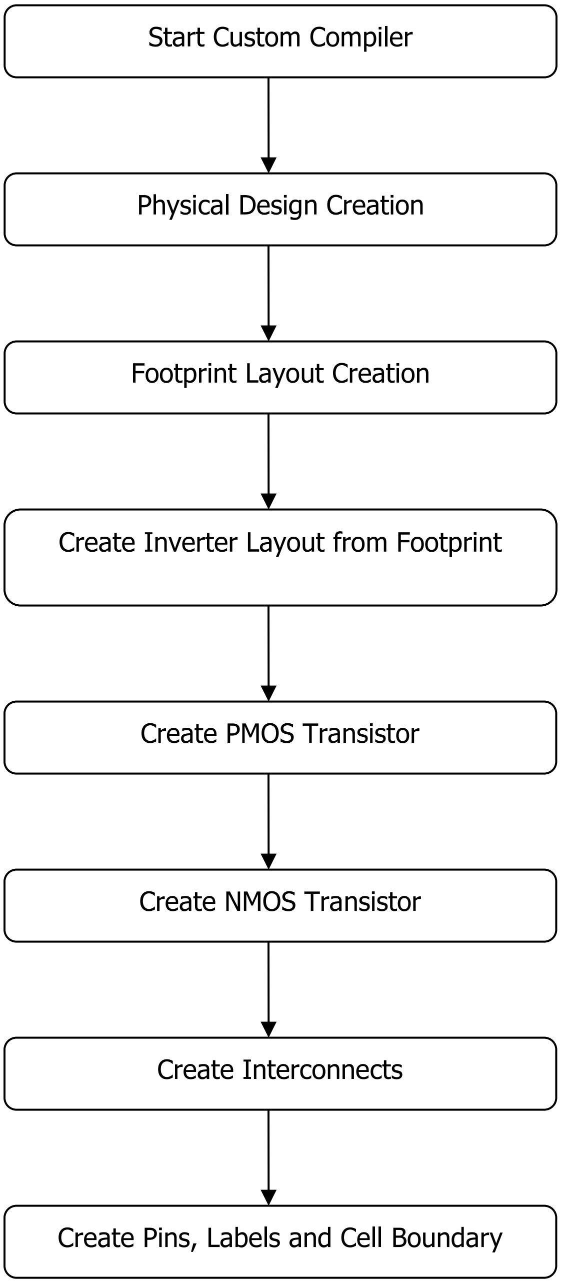
Rule	Design Rule Description		Design Rule Value
1	Minimum Enclosure of Diffusion within NPLUS	≥	0.09 µm
2	Minimum Enclosure of Contact within NPLUS	≥	0.01/0.03 µm

Tap Design Rules

Rule	Design Rule Description		Design Rule Value
1	Source/Drain contact to tap diffusion spacing	≥	0.15µm
2	Tap contact to MOS diffusion spacing	≥	0.16 µm
3	PPLUS enclosure of DIFF in a PWELL strap	≥	0.03 µm
4	NPLUS enclosure of DIFF in an NWELL strap	≥	0.03 µm

Flow Overview

Lab Tasks



Start Custom Compiler

Physical Design Creation

Footprint Layout Creation

Create Inverter Layout from Footprint

Create PMOS Transistor

Create NMOS Transistor

Create Interconnects

Create Pins, Labels and Cell Boundary

File Locations

All files for this lab are located in the directory **LE_Design_Entry_Lab1**.

Directory Structure

`LE_Design_Entry_Lab1`	Current working directory
`CellLibrary`	Cell library
`display.tcl`	Display attributes file
`lib.defs`	Library definition file
`.synopsys_cutom.tcl`	Startup file with tool configuration

Relevant Files

`../PDK`	
`techfiles/`	
`reference40nm_10lm.tf`	ASCII technology file

Answers & Solutions

There is an *ANSWERS / SOLUTIONS* section at the back of each lab. You are encouraged to refer often to this section to verify your answers, or to obtain help with the execution of some steps.

Instructions

Your goal is to create a mask layout of a CMOS inverter using the data creation and editing function.

Task 1. Starting Custom Compiler

1. In the UNIX window, change your working directory to **LE_Design_Entry_Lab1** and invoke Custom Compiler.

```
Unix% cd LE_Design_Entry_Lab1
Unix% custom_compiler &
```

Task 2. Creating Footprint Layout

Footprint cell is the base representation of the minimum physical area of the logic gates to be designed. It also has appropriate locations defined for power and ground rails. Task 4 explains how to create power and ground rails.

In this task, you will create footprint cell.

1. Create cell *Footprint* of view *layout* under library *CellLibrary*.

Task 3. Grid and Gravity Settings

Grids are used to make the data creation faster and it helps to keep all the device dimensions on grid. The Gravity option is used to snap the cursor to the objects.

In this task, you will learn to customize the grid settings and display the grid on the layout. To set the grid preferences, follow the steps mentioned below.

1. From the layout cell view *Footprint*, select **Options → Design** to bring up the **Layout Design Options** window.
2. Click on the tab **Snapping & Grids** to access the preferences related to grids.
3. Select the **Gravity** option. This causes the cursor to snap the objects to the grid.
4. Click **OK** to save the settings.

Task 4. Creating Power and Ground Rails

In this task, you will create power and ground rails with the **Create → Rectangle** command using **m1** layer of purpose **drawing (drw)** from the **Object/Layer Panel**.

Note: Object Layer Panel will be referred as OLP in future references. Use the layers of purpose type *drawing* to create the objects unless it is specified.

```
Object Layer Panel assistant configures the
selectability and visibility of valid objects and LPPs
in a design.
```

1. To select layer **m1** as the active layer, click on layer **m1** under **LPPs** from OLP.

```
LPP – Layer Purpose Pair. All shapes must be created
with a specific layer number and purpose number.
```

2. To enable the Dynamic Measurement of relative coordinates (dx,dy) for the object shape, as the shape is drawn, follow the below steps:

   a. Choose **Options → General**.

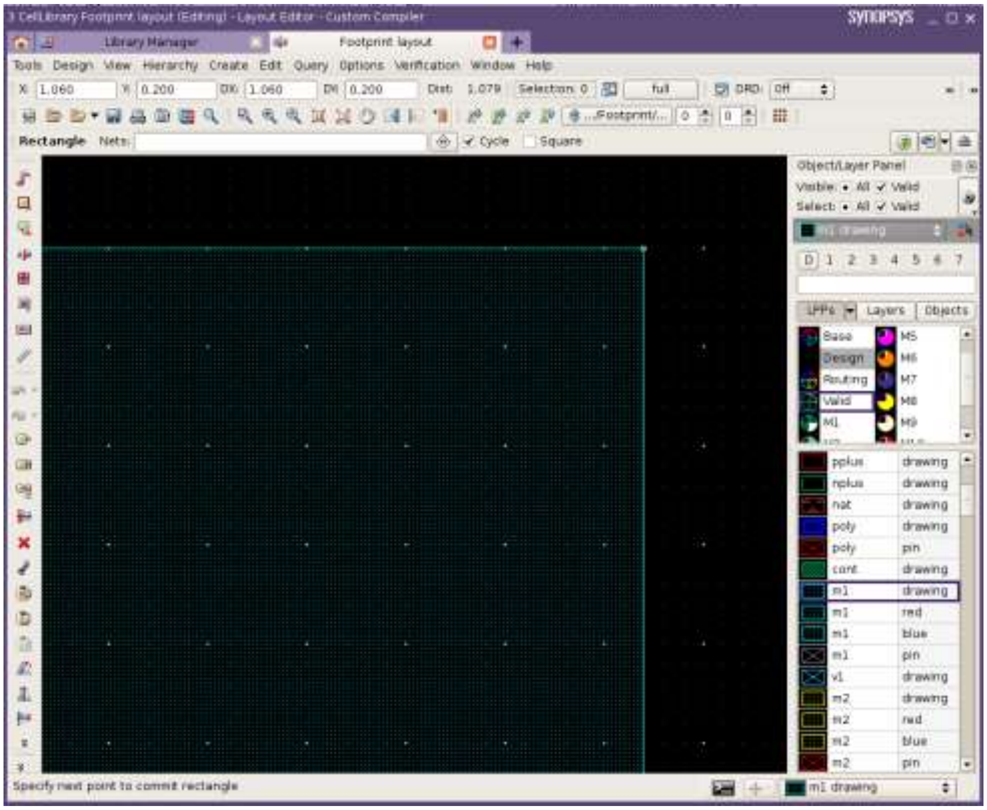
   b. Click on the **Cursor** tab under **Layout**.

   c. Enable **Show Dynamic Measurement**.

   d. Enable **Dynamic** and **DX/DY.**

   e. Click **OK** to apply the changes.

3. Create the ground rail of dimension [X*Y = 1.06µm * 0.2µm] on layer **m1**, using the **Create → Rectangle** command. Use origin (0, 0) as the start point.



3 CellLibrary Footprint layout (Editing) - Layout Editor - Custom Compiler

X 1.060 Y 0.200 DX 1.060 DY 0.200 Dist 1.079 Selection 0 full DRD: Off

Rectangle Nets: ☑ Cycle □ Square

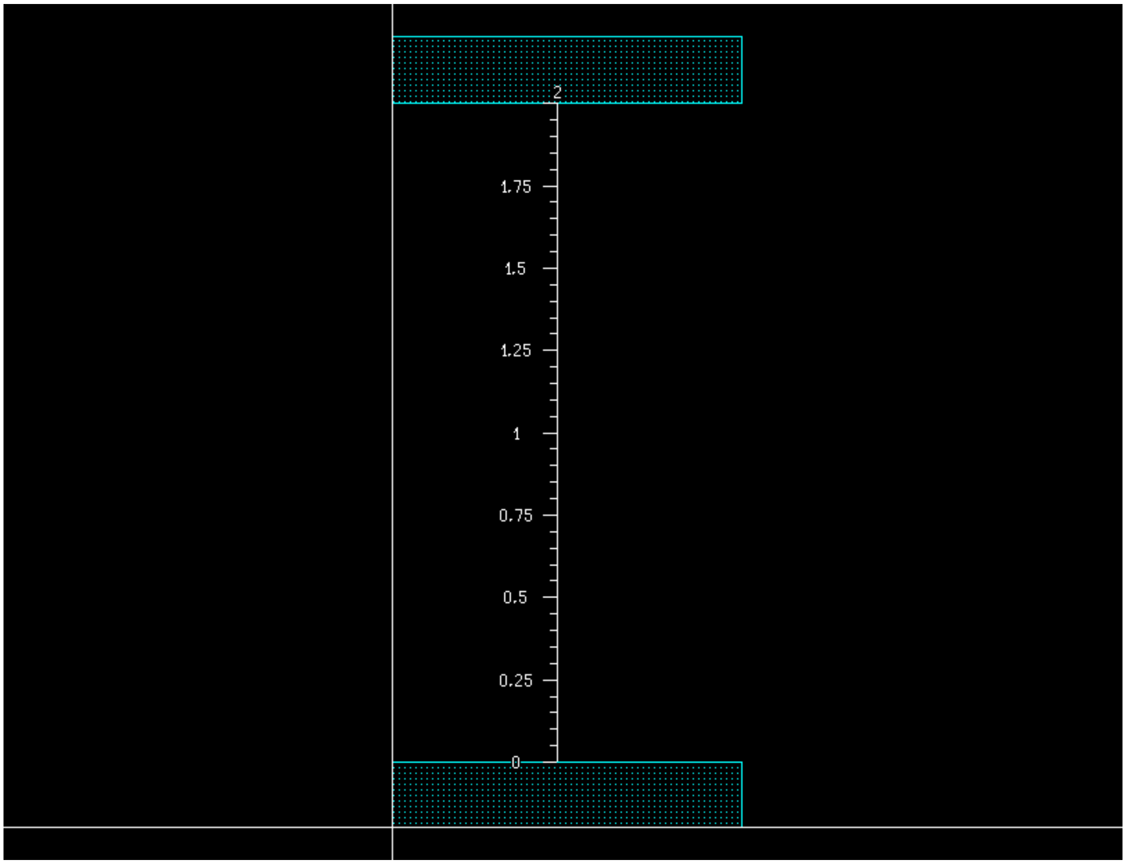
Specify next point to commit rectangle

Note: As the rectangle is drawn, observe that the lower left displays the next action for the current active command. This is applicable for data editing functions also.

4. Create the power rail polygon of the same dimension as the ground rail, with the separation of *2μm* from the ground rail. Use the ruler to measure the spacing.

   Note: Use the **Copy** command instead of drawing the shape again.

   Question 1. How can you create the power and ground rails of nets *VDD* and *VSS* in the same active create polygon command?



5. Check that power and ground rails are aligned. If not aligned, move the power rail using the **Edit → Move** command, in post selection mode.

Hint: To use the post selection mode, invoke the command and select the object.

   Note: It can be observed that, in post selection mode, the selection set remains as it is and the command is completed.

6. Delete the ruler objects using the bind key **Shift-k**.
7. This completes the creation of *Footprint* cell.

Task 5. Designing CMOS inverter Layout

The inverter circuit consists of one PMOS and one NMOS transistor. You will create an inverter cell, followed by individual transistors adhering to the design rules, and then complete the input, output, power, and ground connections. The following sections provides guidelines to accomplish this task.

With the footprint cell as the reference, the tool creates the inverter design.

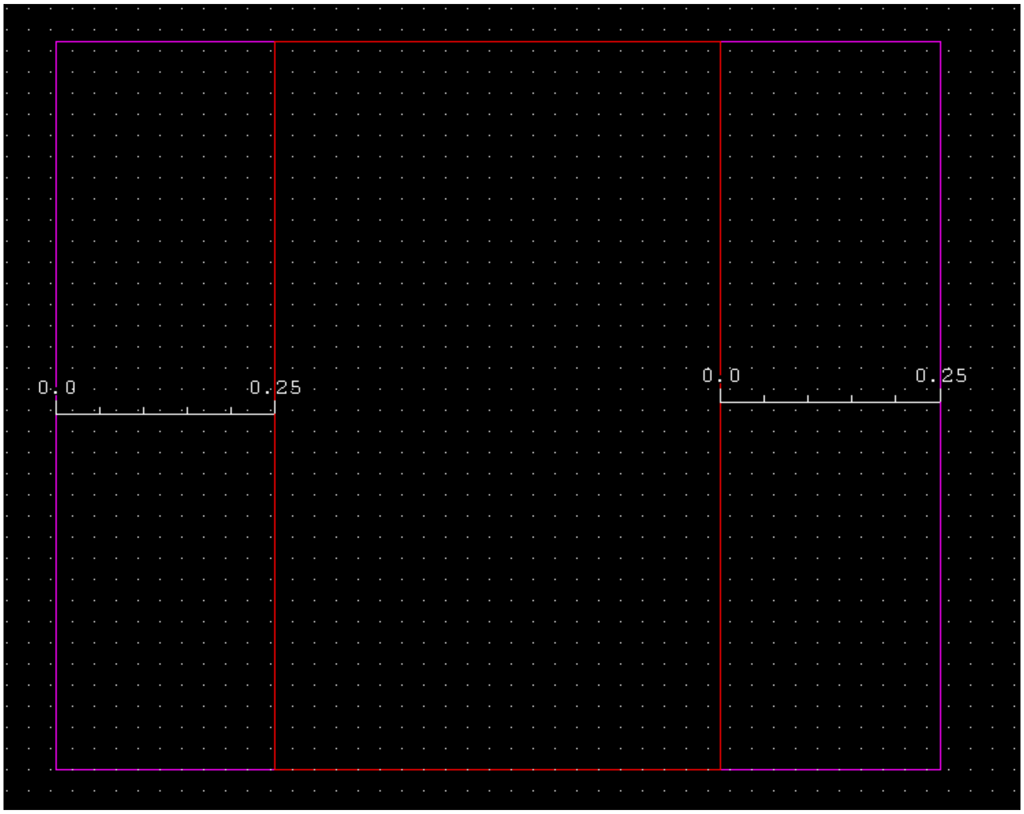
1. Create cell ***inverter*** from ***Footprint*** using **Design →Save As.**
2. Save and close cell *Footprint*.
3. Open cell *inverter* layout.

Task 6. Creating PMOS Transistor Layout

In this task, you will learn to create a PMOS transistor of size w/l = 0.48 µm /0.1 µm, in the inverter cell. Refer to the design rules listed at the beginning of this lab for geometrical constraints on mask layers.

1. The technology data provided is of nwell process. Create the nwell in which the pmos transistor will be formed.
    a. Select **nwell** layer of purpose drawing from **OLP**.
    b. Create nwell rectangle of size X*Y=1.01 µm * 0.86 µm with a spacing of 0.1µm from power rail.
2. Add ptype doping area, which defines the p-channel area within nwell region.
    a. Create rectangle of size X*Y=0.51 µm *0.86 µm, on **pplus** layer of purpose drawing.
3. Place the **pplus** into the **nwell**, using the move command, such that the enclosure value on left and right sides is 0.25 µm, as shown in the figure below.

    Note: Use Zoom commands to zoom to the location to place the pplus.

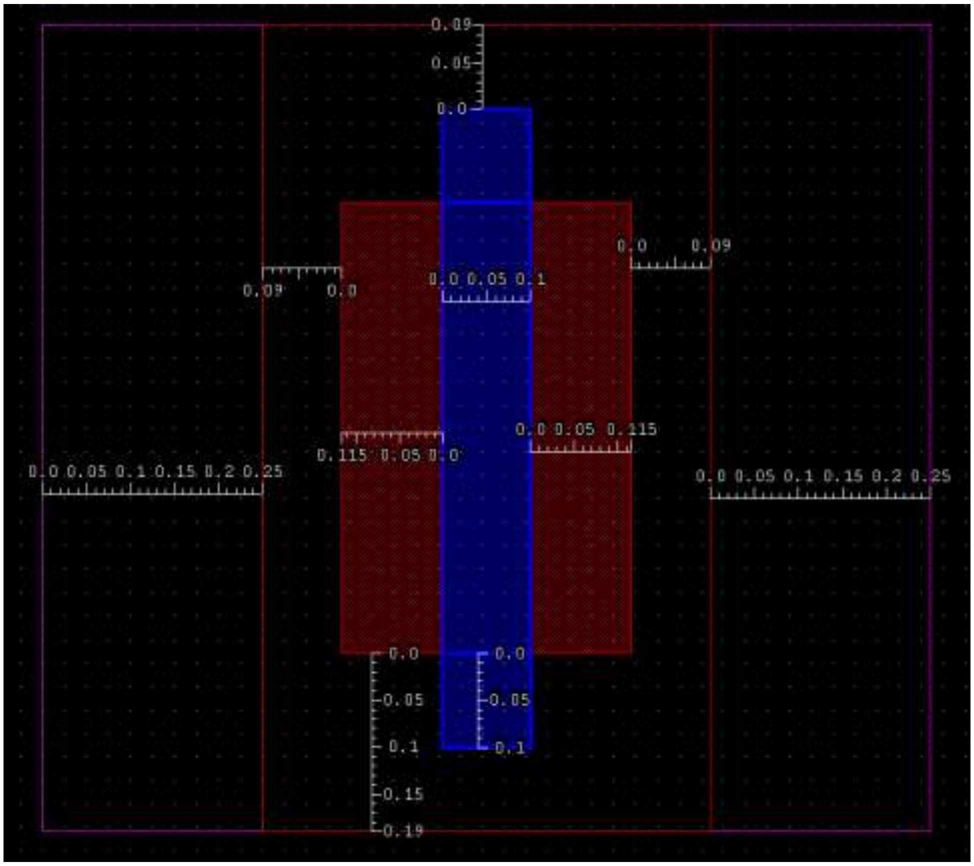


4. The next step is to define the active area, which defines the effective PMOS transistor area, within p-type doping region. Layer **diff** should be enclosed by **pplus** by 0.09 µm.

  a. Create rectangle of size X*Y=0.33 µm * 0.48 µm on layer **diff**.

5. Add polysilicon on the active area, which defines the p-transistor gate. Create the **poly** such that the distance between **poly** and **diff** is 0.115 µm, as shown in the figure below.

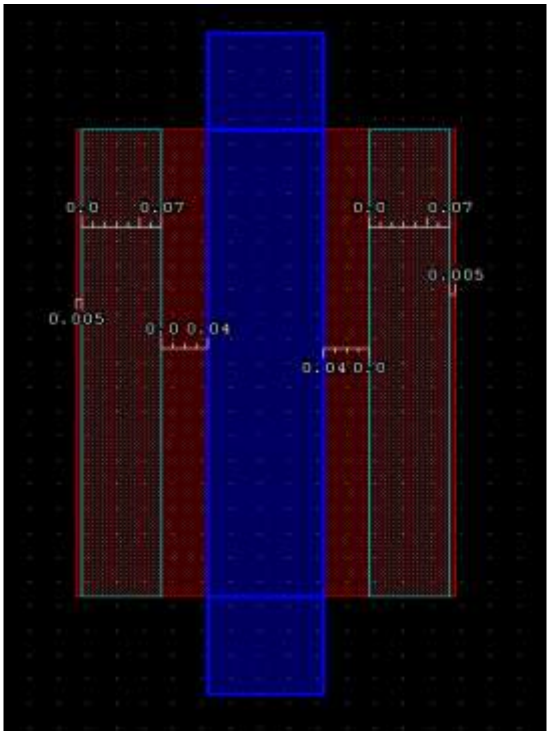
  a. Create rectangle of size X= 0.1 µm * Y= 0.68 µm , on layer **poly**.



6. Turn *off* the visibility of the **nwell** and **pplus** lpp, using middle-click on the lpp in the **OLP.**

7. Add metallization on the drain and source area.

- **a.** Create rectangle of size 0.07 µm * 0.48 µm on **m1** layer.
- **b.** Spacing outside m1 layer, top and bottom is 0.0 µm and left and right is 0.005 µm. The completed figure should resemble the figure given below.

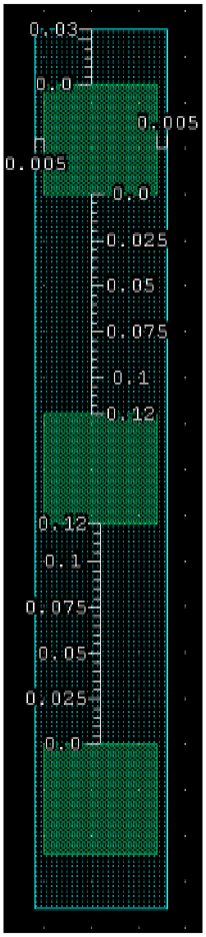


8. Add contacts on the active area to create the connection between active and metal1, as shown in the figure below. Based on the source/drain active area dimension, number of contacts to be added is 3. [Minimum Enclosure of **cont** within **m1** ≥ 0 µm and end of line enclosure value is 0.03 µm]

a. Create rectangle of size X*Y= 0.06 µm * 0.06 µm on layer **cont**.

b. Spacing between the contacts is 0.12 µm and the enclosure value of **cont** within **diff** is 0.01 µm.

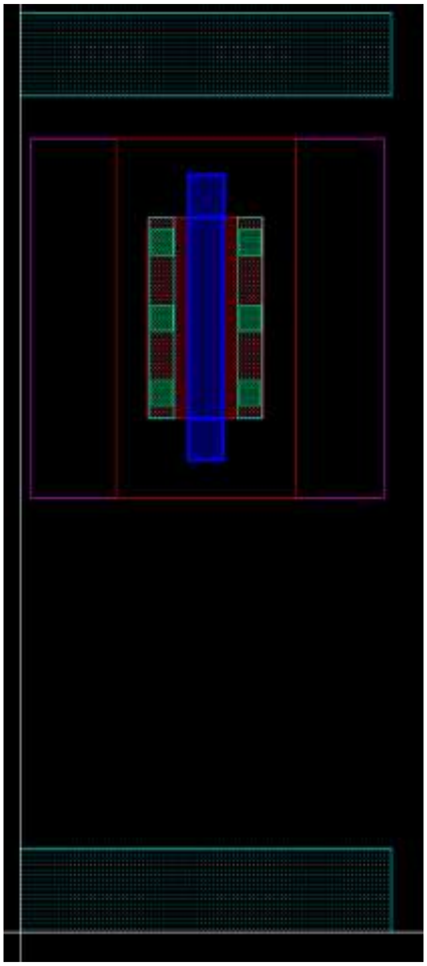
c. After creating a contact, the rest of the contacts can be created by using the copy command.



9. Turn on the visibility of the **nwell** and **pplus** lpp in the **OLP** using right-click**.**

10. This completes the creation of PMOS transistor.

11. At this point, the layout should look like in the figure below.



12. Save and close the cell.

Task 7. Creating NMOS Transistor Layout

In this task, you will create an NMOS transistor of size w/l = 0.33 μm /0.1 μm.

1. NMOS transistor is created by

- **a.** Adding an n-type doping area, which defines the n-channel area.
- **b.** Defining the active area, which defines the effective NMOS transistor area.
- **c.** Adding polysilicon on the active area, which defines the n-transistor gate.
- **d.** Adding metallization on the active area followed by adding contacts on the active area to create the connection between active and metal1.

2. NMOS transistor has been already created for you. Open the cellview *inverter_1*. This cell will be used as the reference for the next tasks.

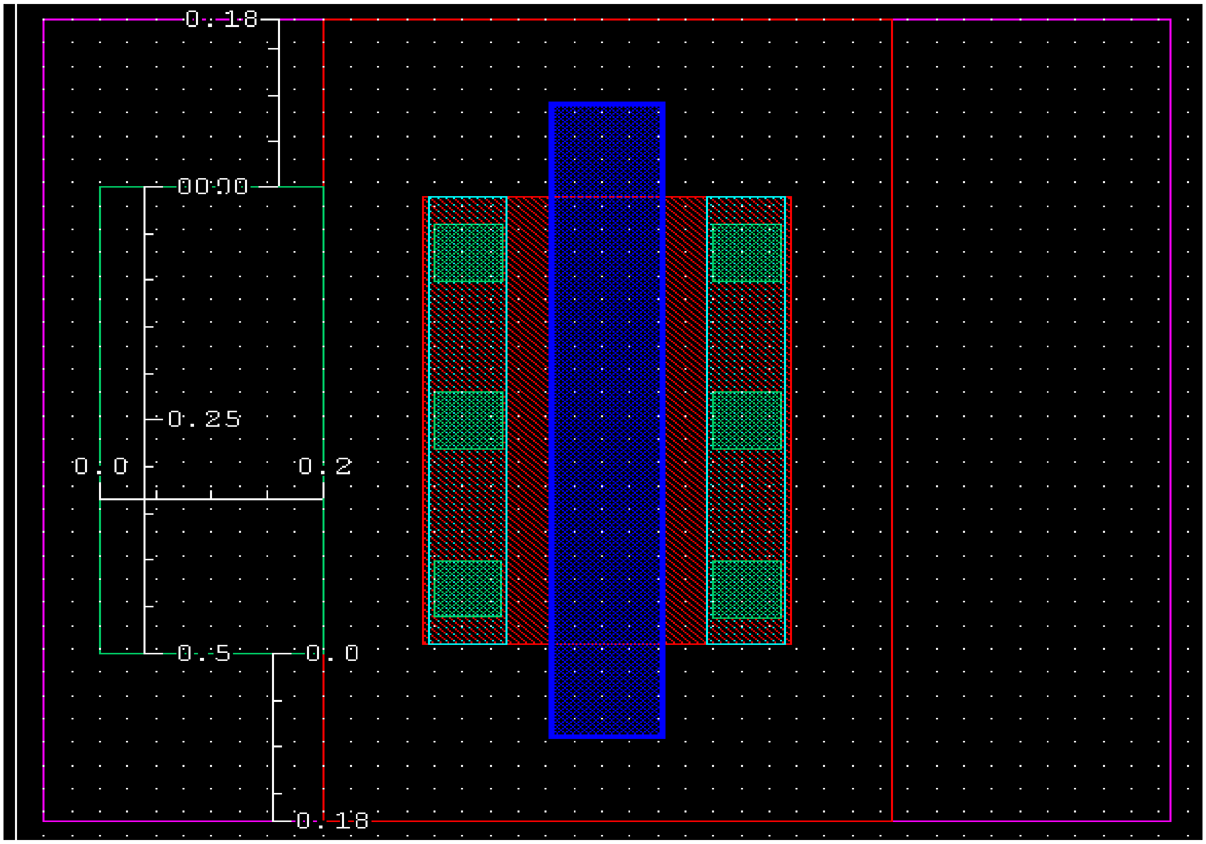
Task 8. Creating Well Contacts

PMOS transistor is placed in nwell, which needs to be biased properly. To properly bias the well, well contacts has to be added.

In this task, you will learn to create well contacts.

1. Add an n-type doping area, adjacent to the p-type doping area.

a. Create rectangle of size X*Y= 0.2µm*0.5 µm on layer **nplus**, as shown in the figure below.



2. Add an active area within the n-type doping area.

a. Create rectangle of size X*Y=0.18µm*0.44 µm on layer **diff**.

Note: For the next two steps (5 and 6), instead of using rectangle command to create contact and metal, **Copy** command can be used to copy previously created metal/contact objects of same LPP in the source/drain areas of PMOS transistor.

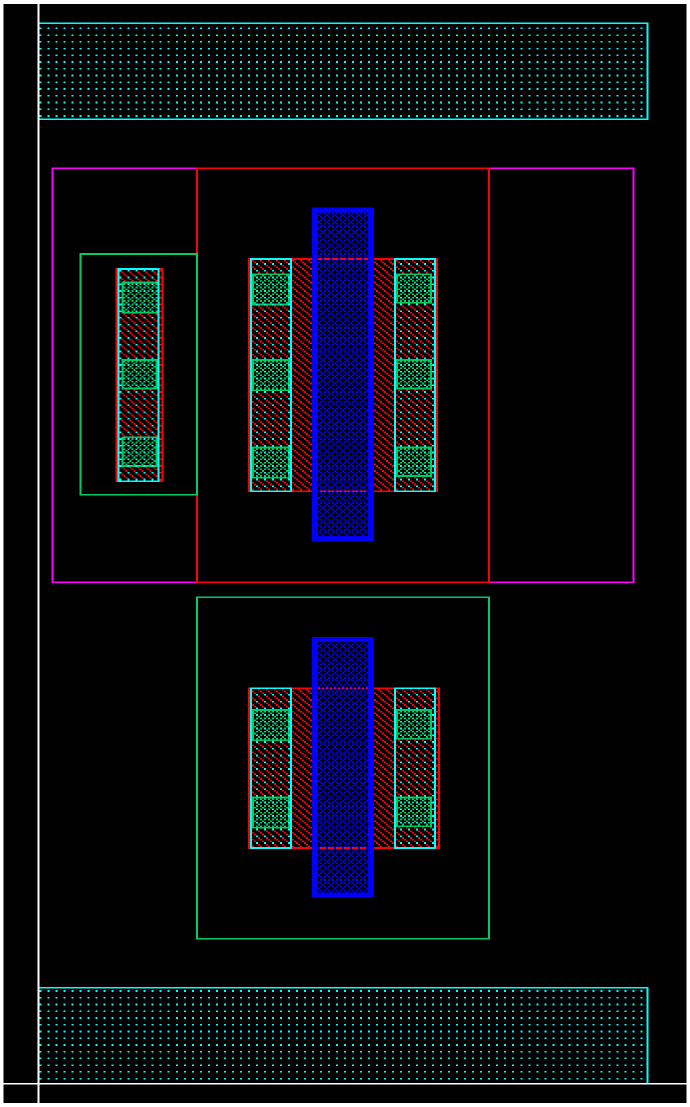
3. Add metallization on the active area.

a. Create rectangle of size X*Y=0.07 µm * 0.44 µm on layer **m1**.

b. Spacing outside **m1** layer, top and bottom is 0.0 µm and left and right is 0.005 µm.

4. Add contacts on the active area to create the connection between active and metal1 on layer **cont**.

5. This completes the creation of well contacts, which looks like the figure below.



6. Save and close the cell *inverter_1*.

Task 9. Creating Substrate Contacts

NMOS transistor is placed in P-substrate, which needs to be biased properly.

In this task, you will create p-type substrate contacts.

1. p-sub type contacts are created by
   - **a.** Adding a p-type doping area, adjacent to the n-type doping area.
   - **b.** Adding an active area within the p-type doping area.
   - **c.** Adding metallization on the active area.
   - **d.** Adding contacts on the active area to create the connection between active and metal1.

2. Substrate contacts for NMOS transistor has been already created for you. Open the cellview *inverter_2*. This cell will be used as the reference for next tasks.

Task 10. Modifying Power and Ground Rails

With the addition of well and substrate contacts, the cell size is already defined. Therefore, power and ground rails must be resized to match the current cell size.

1. Stretch both power and ground rails by *0.025* µm on right side using the **Stretch** command in pre-selection mode.

Hint: Pre-Selection: Select the objects and activate the command.

Question 2. How do you stretch both power and ground rails simultaneously?

....

Task 11. Creating Interconnects

Interconnects must be created for all the input, output, power and ground connections.

In this task, you will learn to create interconnects using paths.

1. Connect the gates of transistors, which will form the input, on layer **poly**. Use **Create → Interconnect**, to create a path of **Width** : *0.1*µm and with S**tyle :** *extend***.**

2. Using **Create → Via** command, create *polyCont* viadef on the input gates.

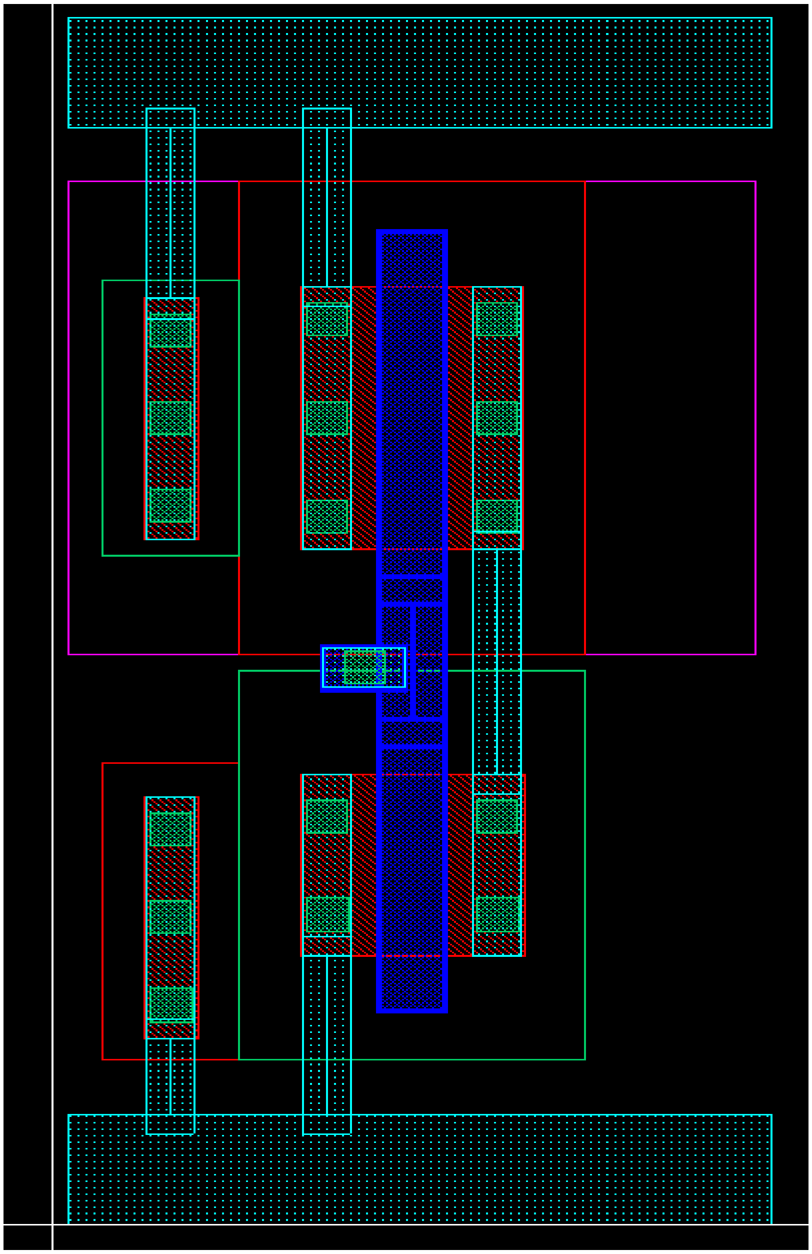
3. Create path to connect the drains of transistors, which will form the output, on layer **m1.** Use **Width** : *0.07*µm and **Style :** *extend*.

4. Complete the following connections using path of **Width :** *0.07* µm, **Style :** *extend* on layer **m1**.

a. Connect the source of PMOS transistor and well contact to power rail.

b. Connect the source of NMOS transistor and substrate contact to ground rail.

5. This completes all interconnections in the inverter design and the layout should look like the figure below.



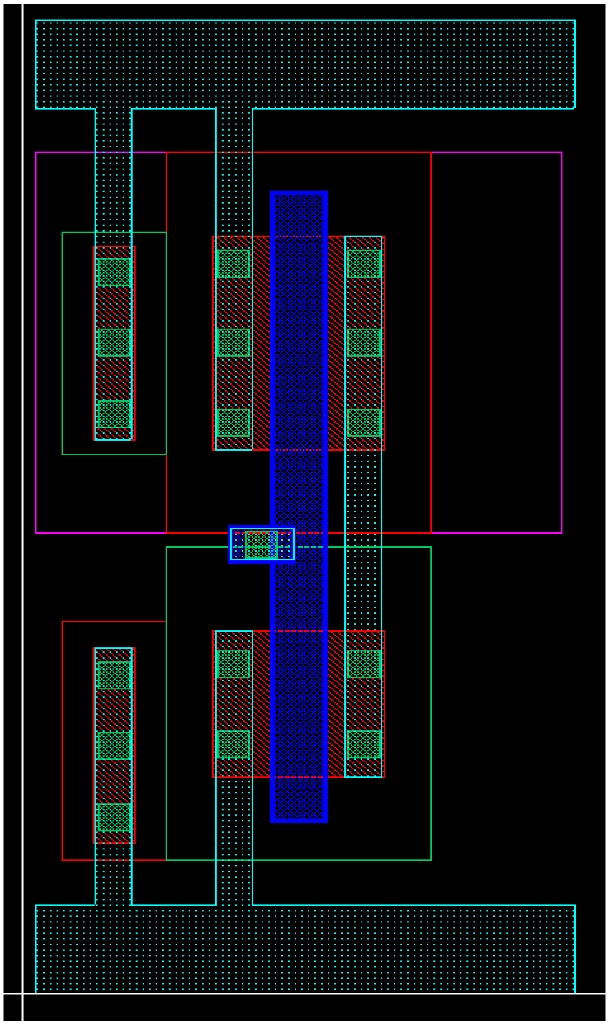
Task 12. Merging Power, Ground Interconnects, and Rail

In this task, you will learn to use the **Merge** command to merge the power and ground interconnects to their respective rails.

1. Use **Edit →Merge** command to merge all power connections with power rail.

   Note: Select the first object, use shift and left-click to select the other objects to be merged and then execute the command.

2. Similarly merge all ground connections with ground rail. The inverter should resemble the figure below.



Task 13. Creating Pins

In this task, you will learn to create pins using **Create→ Pin → By Shape** for all the nets in the layout.

1. Create pins *A*, *Z*, *VDD*, *VSS* of size X*Y=0.07µm*0.07µm on the respective objects. Use layer m1 of purpose pin to draw the pin shapes.

   Note: Command will not allow users to create pins until a valid name is entered.

2. Try to create all the four pins *A*, *Z*, *VDD*, *VSS* in single command activation.

   Question 3. How are the pins created if

   a. **Cycle** option is selected?

   b. **Cycle** option is deselected?

   ....

3. Select appropriate **Type**, **Access Direction, Signal Type** attribute before each pin is created.

Task 14. Creating Labels

Labels are used for displaying net names in a design. They are also used by the physical verification tool Hercules to identify the ports in the design.

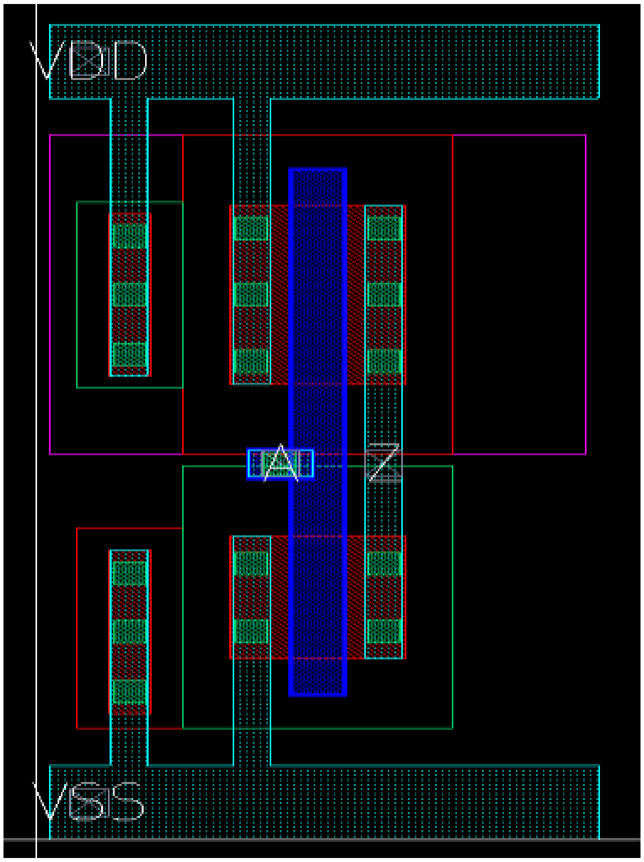
In this task, you will learn to create labels for all the nets in the layout using **Create → Text →Label** on lpp **m1text:drw**. The completed layout should look like the figure below.

Note: Use the **Filter** field in the **OLP**, to locate the **m1text:drw** lpp.

1. Create *A*,*Z*, *VDD* and *VSS* texts of **height** 0.1 on the respective net objects.

Question 4. What happens, when text objects are placed with **Attach** option selected?

....



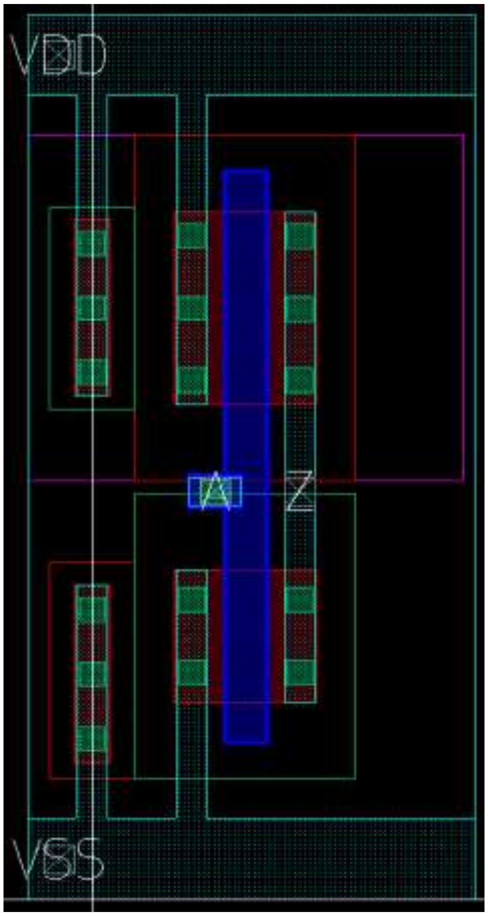
Task 15. Moving Origin and Create Boundary

Cell boundary is required to define the data extents of the cell. In this task, you will learn to create a cell boundary.

1. Currently the origin of the cell is not correct. Using the command **Edit → Other → Move Origin,** change the origin by clicking on the lower left corner of the ground rail.

2. Use **Create → Boundary to** create a cell boundary of type **P&R**. The completed layout should look like the figure below.

   Question 5. How can you create the cell boundary automatically?

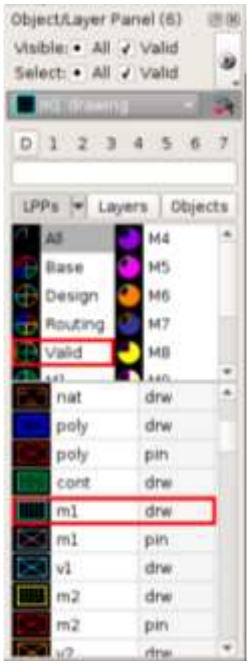


3. Save the cell.

Task 16. Filtering LPPs in OLP Used in the Design

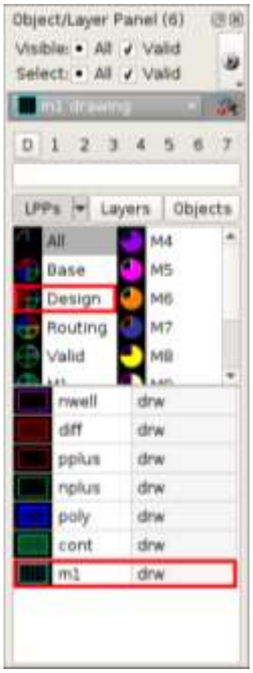
You can filter LPPs used in the design in the OLP panel.

1. From the **OLP** panel, click the **LPPs** tab.



nat	drw
poly	drw
poly	pin
cont	drw
m1	drw
m1	pin
v1	drw
m2	drw
m2	pin

2. Select the **Design** set. Observe the changed number of LPPs in the OLP panel.



nwell	drw
diff	drw
pplus	drw
nplus	drw
poly	drw
cont	drw
m1	drw

3. Close the design.

4. Exit Custom Compiler.

Answers / Solutions

Task 4. Creating Power and Ground Rails

Question 1. How can you create the power and ground rails of *VDD, VSS* nets in the same active create polygon command?

a. Bring up the **Create Rectangle** tool bar.

b. Enter *VDD VSS* in the **Net name(s)** field.

c. Check **Cycle** option.

d. Complete power rail creation.

Note: It can be seen that **Net name(s)** field is left with ***VSS***. **Cycle** option removes the net name once object with net name has been committed.

Task 10. Modifying Power and Ground Rails

Question 2. How do you stretch both power and ground simultaneously?

Select one of the rail edges. To select additional edges, use shift and left-click and execute the **Stretch** command.

Task 13. Creating Pins

Question 3. How are the pins created if:

a. **Cycle** option is selected?

b. **Cycle** option is deselected?

Solution:

c. **Cycle** → enabled- pin names are removed from the **Names** field after the pin creation.

d. **Cycle** → disabled- pin names are NOT removed from the **Names** field after pin creation.

Task 14. Creating labels

Question 4. What happens when text objects are placed with the **Attach** option selected?

The **Attach** option creates parent-child relationship between object and label.

When the **Attach** option is selected, after placing the label, the user is prompted to select the parent object to which the text will be attached.

Task 15. Moving Origin and Create Boundary

Question 5. How can you create the cell boundary automatically?

The **Auto Create** option automatically creates a cell boundary around the objects in the current design.

2

Interconnects Creation and Edit

Learning Objectives

During this lab, you will create a mask layout design of a CMOS logic gate using design rule-guided data creation and editing functions.

After completing this lab, you should be able to:

- Create interconnects for amplifier design
- Stretch interconnect

Lab Duration:
30 minutes

Introduction

The purpose of this lab is to help you understand the interconnect creation and editing in Custom Design Layout editor. This lab uses generic process design rules shipped with the generic PDK kit, to design the simple amplifier layout.

File Locations

All files for this lab are located in the directory **LE_Design_Entry_Lab2**.

Directory Structure

`LE_Design_Entry_Lab2`	Current working directory
`CellLibrary`	Cell library
`display.tcl`	display attributes file
`lib.defs`	Library definition file
`.synopsys_cutom.tcl`	Startup file with tool configuration

Relevant Files

`../PDK`	
`techfiles/`	
`reference40nm_10lm.tf`	ASCII technology file

Instructions

Your goal is to create and edit routing of layout of a CMOS amplifier using the data creation and editing function.

Task 17. Starting Custom Compiler

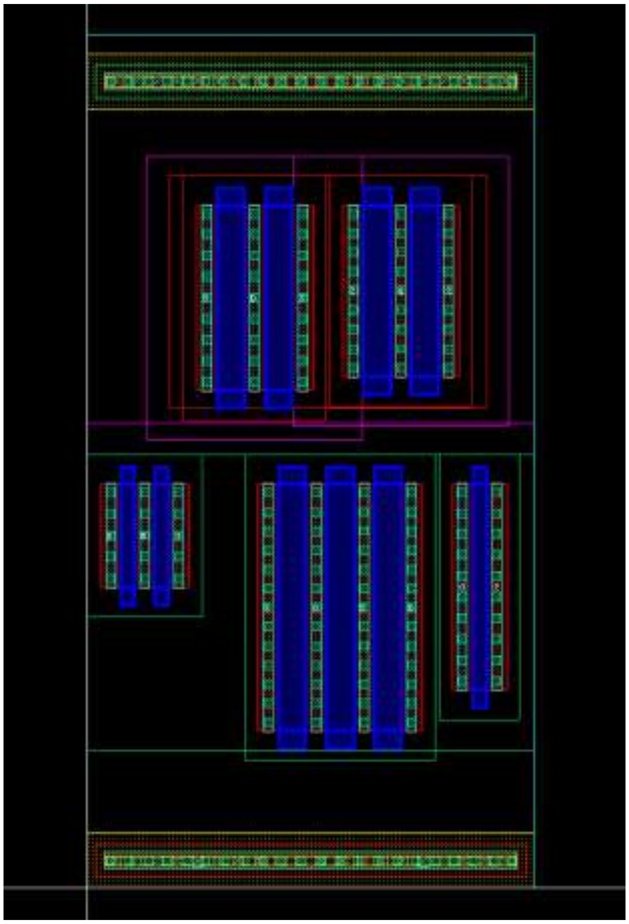
5. In the UNIX window, change your working directory to **LE_Design_Entry_Lab2** and invoke Custom Compiler.

```
Unix% cd LE_Design_Entry_Lab2
Unix% custom_compiler &
```

Task 18. Creating Interconnects of Amplifier Layout

In this task, you will route the amplified design by creating interconnects between terminals.

1. From the Library manager open **library/cell/vuiew - CellLibrary/amp/layout** design. Observe the amplifier design with missing routing.



2. Choose **Options → Design** to open the **Layout Design Options** dialog.

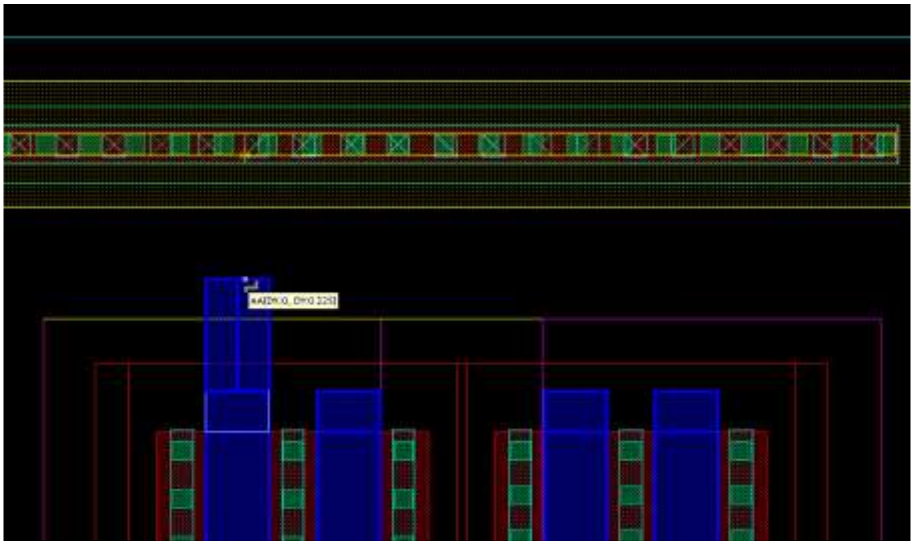
3. Go to the **Snapping & Grids** tab and then enable the **Gravity** option and set **Align Assist** to **Implicit**.

4. Click OK to close the **Layout Design Options** dialog.

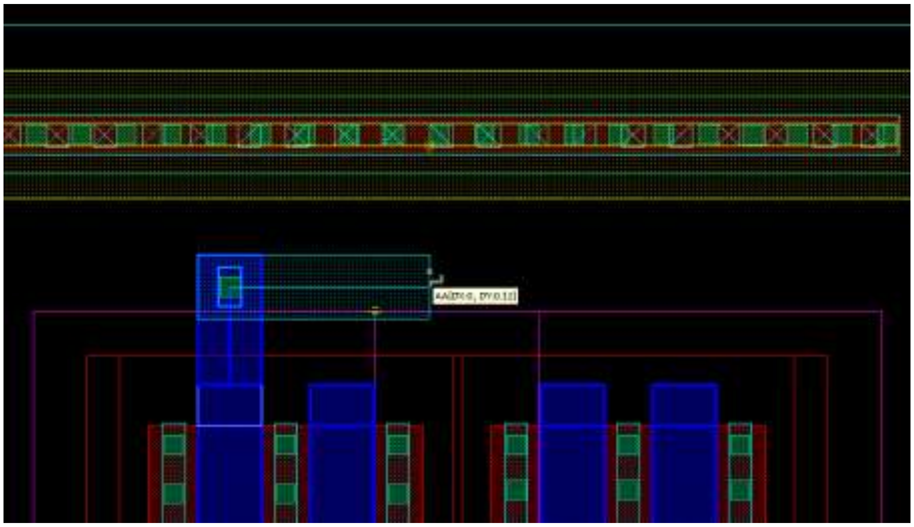
5. Choose **Create → Interconnect** to enable the **Create Interconnect** command.

6. From the **Interconnect** toolbar, choose **Taper** from the **Auto Options** option menu.

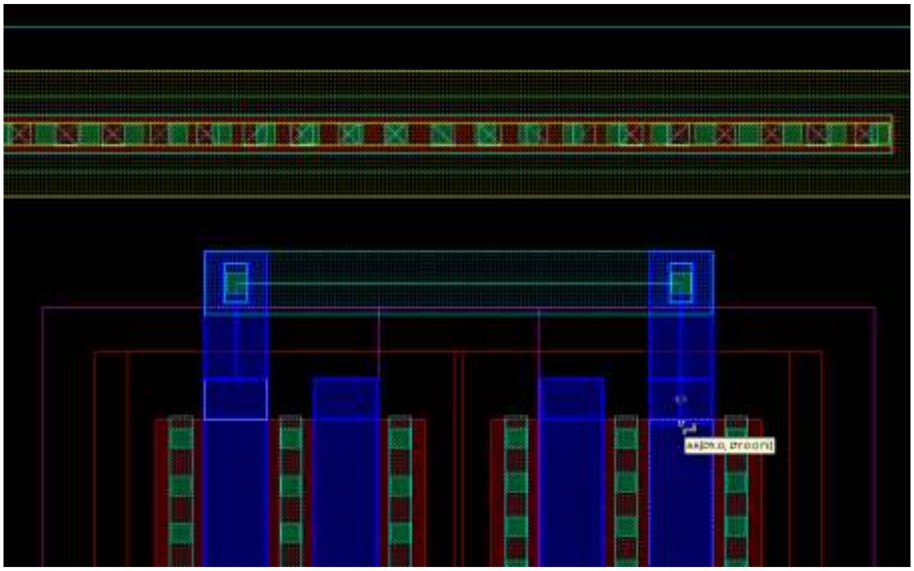
7. Start the interconnect creation from the left PMOS instance top poly terminal as on the figure below:



8. Use the shortcut key **Shift-V** to change the routing layer to metal1. This will also place a via between poly and metal1 layers.

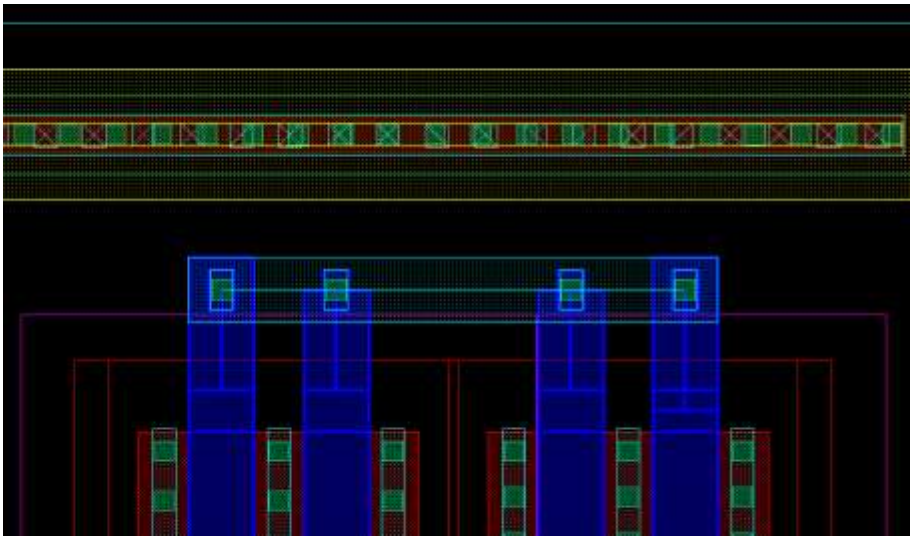


9. Route interconnect to the right PMOS upper gate terminal. Use the shortcut key **Ctrl-V** to change the routing layer to poly. Observe placed via between metal1 and poly.



10. Press **Enter** to commit the routing.

11. Connect all other PMOS terminals in the same way.



12. Enable the **Visual** mode from the **DRD** option menu on the **DRD** toolbar.

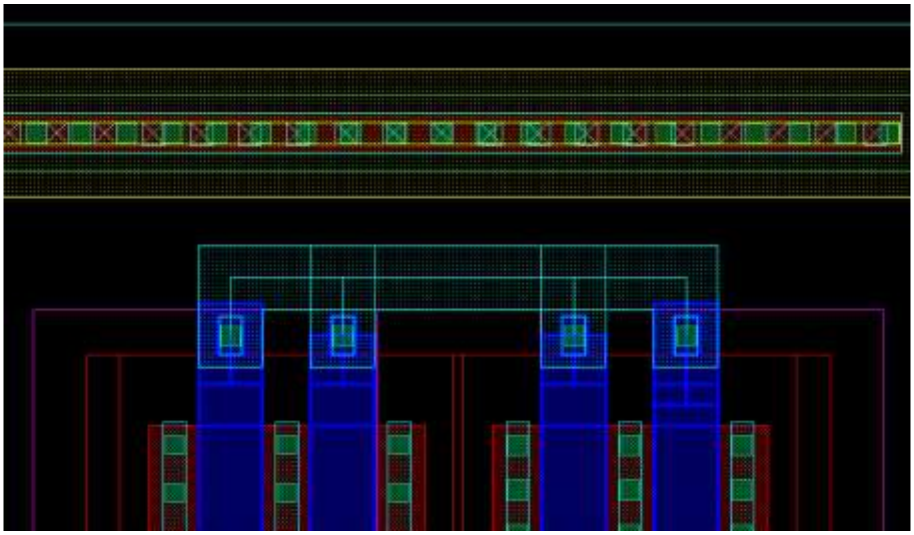
    Note: To enable DRD engine, select Advanced license from **Home → License** menu.

13. Choose **Edit → Stretch** to invoke the **Stretch** command.

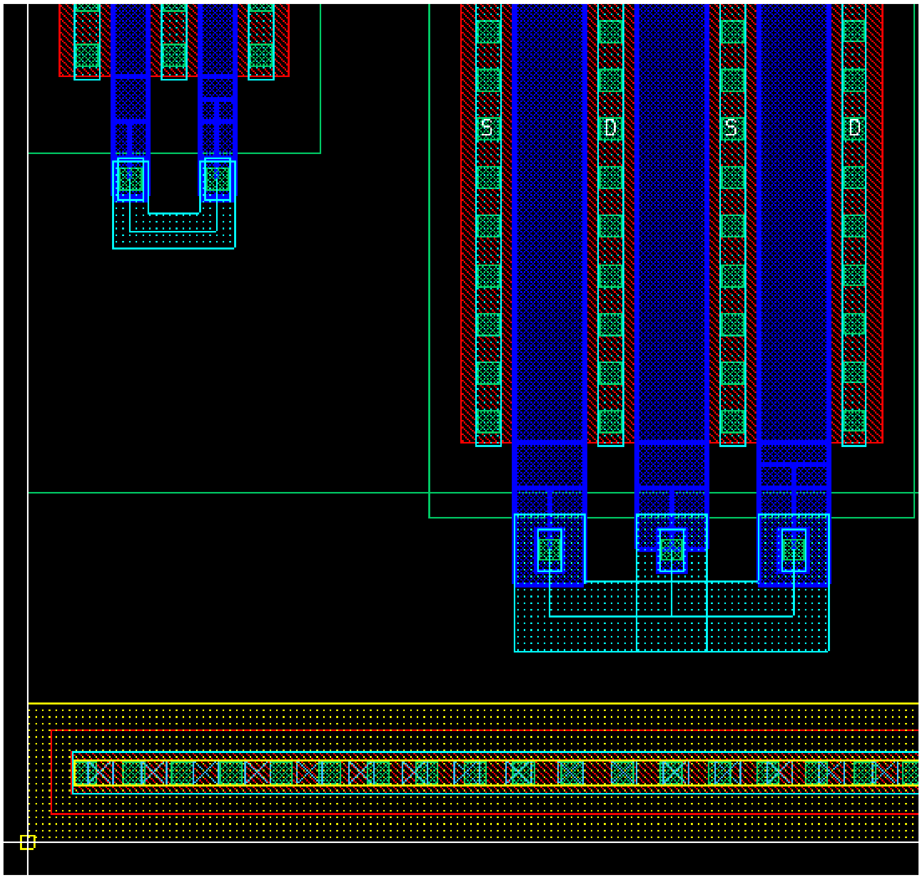
14. Enable the **Keep Connected** option from the **Stretch** command toolbar. Choose **Backbone** from the **Mode** option menu.

15. Select the **metal1 drawing** path showed on the figure above and then stretch down until the minimum DRC rule is reached.

16. Change the **Mode** to **Fixed Via** and then stretch the same metal path up again until the minimum DRC rule is reached. You will get the interconnect as show on the figure below:

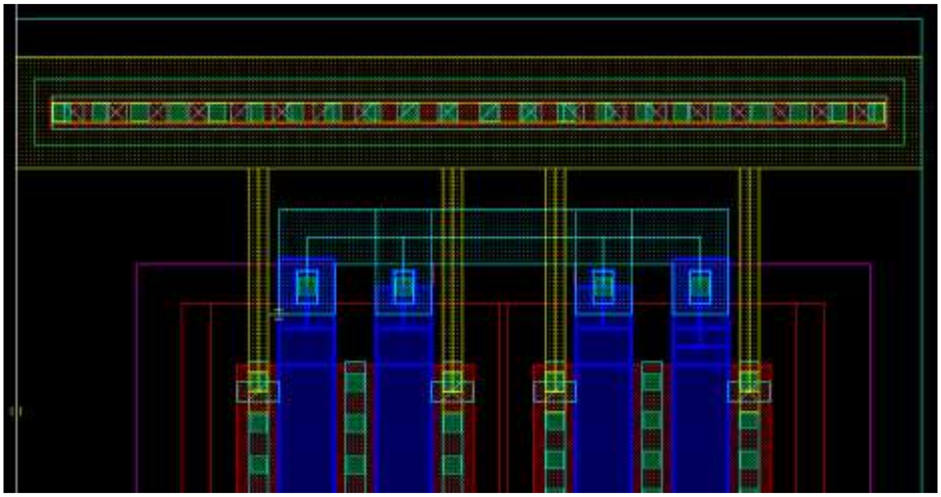


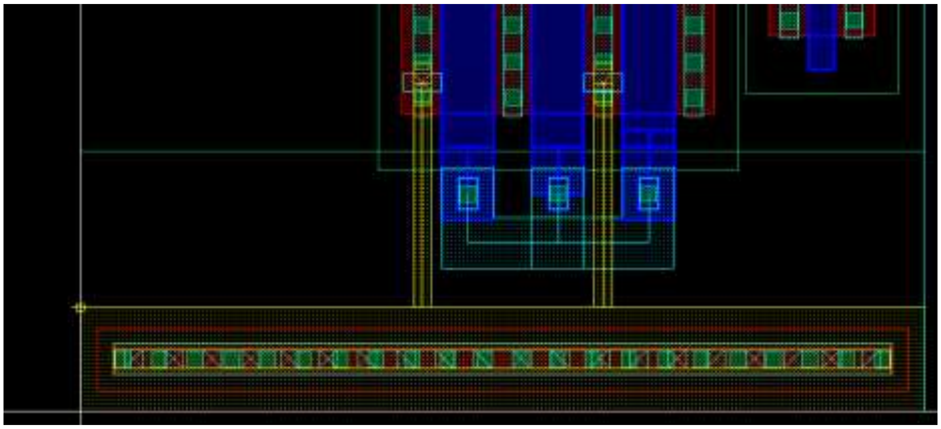
17. In the same way, route NMOS transistors gate terminals.



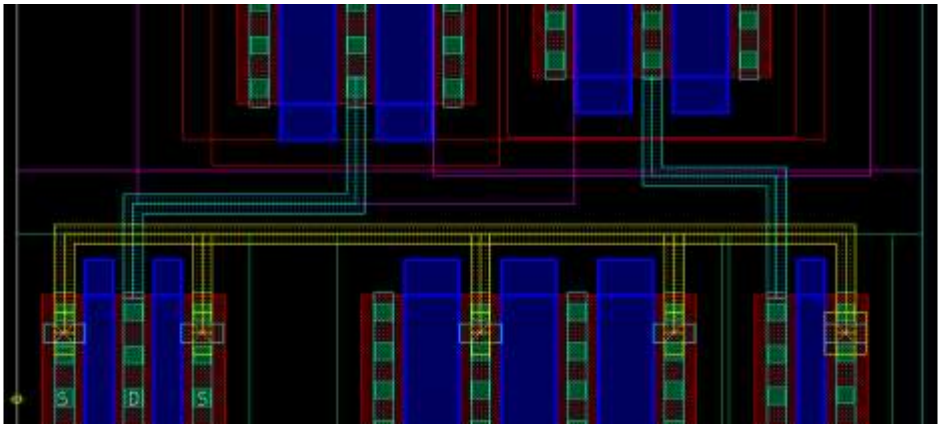
18. Connect PMOS and NMOS VDD and VSS terminals to power rails as shown in the figures below:

Note: You can adjust the width of interconnect by using the shortcut key **Ctrl-4** in case if it automatically changed what **Taper** option enabled.

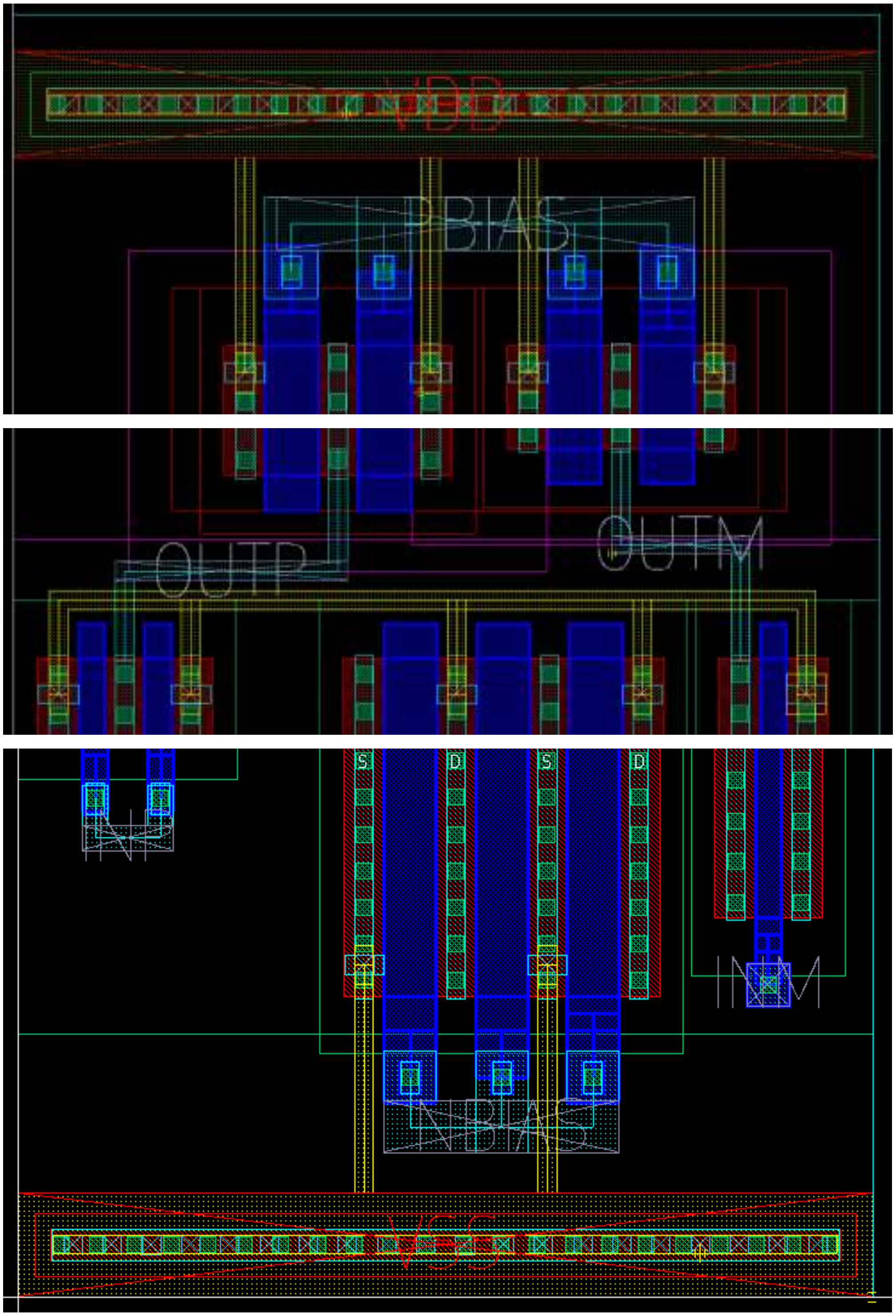




19. Create connections between PMOS and NMOS devices to complete the amplifier routing as shown in the figure below:



20. Create pins for the amplifier as shown in the figures below:



21. Save and close the design.

22. Exit Custom Compiler.

Copyright and Proprietary Information Notice

Destination Control Statement

All technical data contained in this publication is subject to the export control laws of the United States of America. Disclosure to nationals of other countries contrary to United States law is prohibited. It is the reader's responsibility to determine the applicable regulations and to comply with them.

Disclaimer

SYNOPSYS, INC., AND ITS LICENSORS MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Trademarks

Synopsys, Inc.
690 E. Middlefield Road
Mountain View, CA 94043
www.synopsys.com

3

3. Physical Verification Environment

Learning Objectives

In this lab you will use the **Physical Verification Environment (PVE)** and the **Custom Compiler Layout Editor (LE)** to:

- Set up and customize the options for **IC Validator (ICV)** DRC and LVS runs
- Execute **ICV** DRC and LVS from the **Custom Compiler PVE** environment
- View and debug violations using the **ICV VUE** interactive debugger
- Fix DRC and LVS violations using **Custom Compiler LE**

⏰ **Lab Duration: 30 minutes**

Introduction

The ***Physical Verification Environment (PVE)*** in ***Custom Compiler*** provides easy to use and interactive integration with the ***IC Validator (ICV)*** physical verification tool.

The ***PVE*** environment allows ***ICV*** setup information to be pre-configured for a process or PDK, allows ***ICV*** tool options and control variables to be imported from the rule deck, and stores run settings for ease of re-use for future runs. These features allow layout designers to conveniently and quickly run DRC and LVS verification from the ***Custom Compiler Layout Editor (LE)***.

We will set up and customize DRC and LVS options, and run the jobs with ***ICV***. We will then use the ***ICV VUE*** interactive debug with the ***Custom Compiler Layout Editor (LE)*** to debug fix the violations.

File Locations

All files for this lab are located in the directory **`LE_Physical_Verification_Lab1`**

Directory Structure

`LE_Physical_Verification_Lab1`	Current working directory
`DesignLibrary`	OpenAccess Design library
`../PDK/icv/drc/reference40nm_icv_drc.rs`	ICV DRC Rule Deck
`../PDK/icv/lvs/reference10lm_lvs.rs`	ICV LVS Rule Deck
`../PDK/techfiles/reference40nm_layer.map`	OA Layer Mapping File
`../PDK/pdk.xml`	PDK Configuration File

Tools Versions

The following tool versions should be setup before running the lab:

IC Validator	M-2018.06

Instructions

Task 19. Start Custom Compiler

In this task we will invoke ***Custom Compiler***.

23. In the Unix window, change your working directory to **LE_Physical_Verification_Lab1** and invoke ***Custom Compiler***.

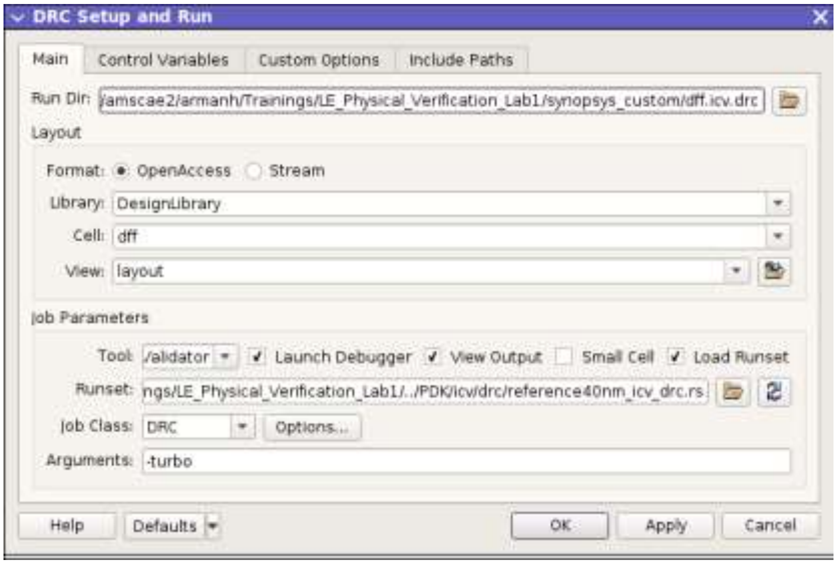
```
Unix% cd LE_Physical_Verification_Lab1
Unix% custom_compiler &
```

Task 20. Set Up *ICV* DRC Options

To run DRC in ***ICV***, a set of input files and options are required. In this task we will set up the required inputs and run DRC.

24. Open the cellview ***DesignLibrary/dff/layout*** from the ***Library Manager Window***. When the layout opens, hit the ***Shift-F*** bindkey to display all hierarchy levels.

25. Invoke the ***DRC Setup Dialog*** from the menu ***Verification > DRC > Setup and Run...***



Run Dir: /amscae2/armanh/Trainings/LE_Physical_Verification_Lab1/synopsys_custom/dff.icv.drc

Layout

Format: ● OpenAccess ○ Stream

Library: DesignLibrary

Cell: dff

View: layout

Job Parameters

Tool: Validator ☑ Launch Debugger ☑ View Output □ Small Cell ☑ Load Runset

Runset: ngs/LE_Physical_Verification_Lab1/../PDK/icv/drc/reference40nm_icv_drc.rs

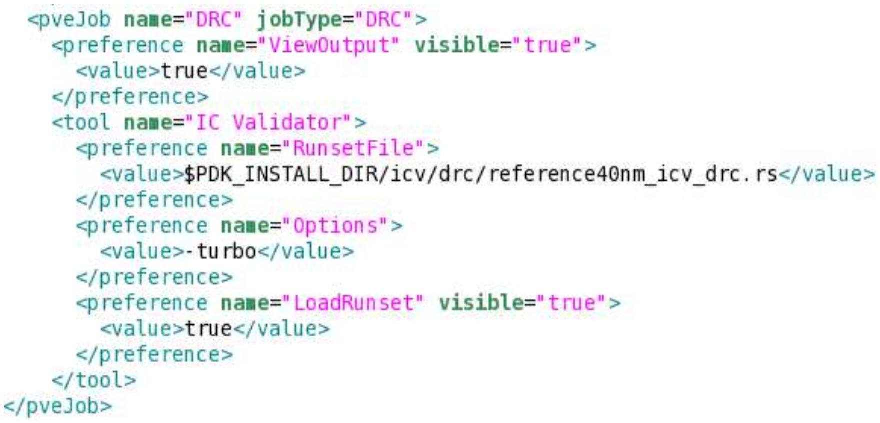
Job Class: DRC Options...

Arguments: -turbo

Note: Default values for the dialog fields are prefilled. Observe that the ***Run Dir*** field is filled by the value ***<cwd>/synopsys_custom/< cellname>.icv .drc***. The fields ***Library***, ***Cell***, ***View*** in the layout section are filled in by the active cellview open in ***LE***.

Note: Observe that PDK specific information is prefilled, such as the ***Runset*** field in the ***Main Tab***, and tool options in the ***Custom Options Tab***. These are set in the pdk configuration file.

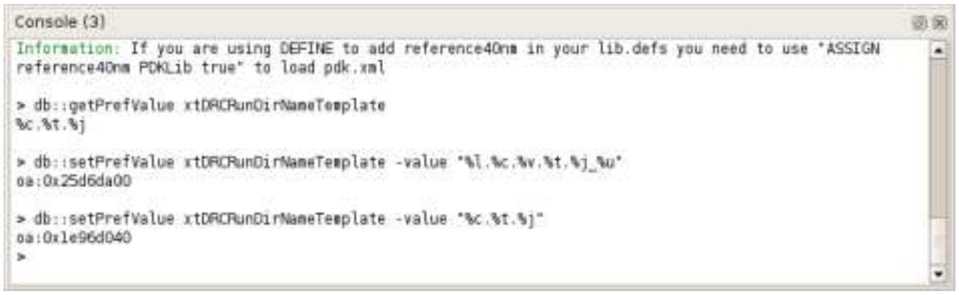
26. Open the pdk configuration file from ***../PDK/pdk.xml*** and check the ***ICV*** DRC job section to understand how these default settings were populated for this PDK. Make sure that the files in the ***Runset*** field in the ***Main Tab*** and the ***Layer Map*** field in the ***Custom Options Tab*** exist and are set to the values specified in the pdk configuration file.



```
    <pveJob name="DRC" jobType="DRC">
      <preference name="ViewOutput" visible="true">
        <value>true</value>
      </preference>
      <tool name="IC Validator">
        <preference name="RunsetFile">
          <value>$PDK_INSTALL_DIR/icv/drc/reference40nm_icv_drc.rs</value>
        </preference>
        <preference name="Options">
          <value>-turbo</value>
        </preference>
        <preference name="LoadRunset" visible="true">
          <value>true</value>
        </preference>
      </tool>
  </pveJob>
```

27. Click left-mouse-button ***(LMB)*** the console icon in the bottom right cotner of the ***LE Window*** to open the ***Console Assistant***, and enter the following command in the console: ***db::getPrefValue xtDRCRunDirNameTemplate*** and confirm that the preference value is set to ***"%c.%t.%j"***. The preference setting for the runDirName supports token substitution using the following tokens: ***%l=libName***, ***%c=cellName***, ***%v=viewName***, ***%t=toolName***, ***%j=jobType***, ***%u=userName***.

28. Enter the following command to customize the runDir name: ***db::setPrefValue xtDRCRunDirNameTemplate -value "%l.%c.%v.%t.%j_%u"***. Then close the ***DRC Setup Dialog*** and re-open it. Observe that the runDir name has been updated to the new evaluated string.



Console (3)

```
Information: If you are using DEFINE to add reference40nm in your lib.defs you need to use "ASSIGN
reference40nm PDKLib true" to load pdk.xml

> db::getPrefValue xtDRCRunDirNameTemplate
%c.%t.%j

> db::setPrefValue xtDRCRunDirNameTemplate -value "%l.%c.%v.%t.%j_%u"
oa:0x25d6da00

> db::setPrefValue xtDRCRunDirNameTemplate -value "%c.%t.%j"
oa:0x1e96d040
>
```

29. Reset the runDirName back to normal using the command: ***db::setPrefValue xtDRCRunDirNameTemplate -value "%c.%t.%j"*** then close and re-open the setup dialog again.

30. Leave the rest of the fields as is. Make sure that the ***Launch Debugger*** and ***View Output*** fields are enabled. This will automatically launch the ***ICV VUE*** violation debug environment after the DRC run completes. The ***View Output*** field will make the ***Text Viewer Window*** open and display the standard output and status of the job as it runs.

31. Click ***OK/Apply*** to start the run.

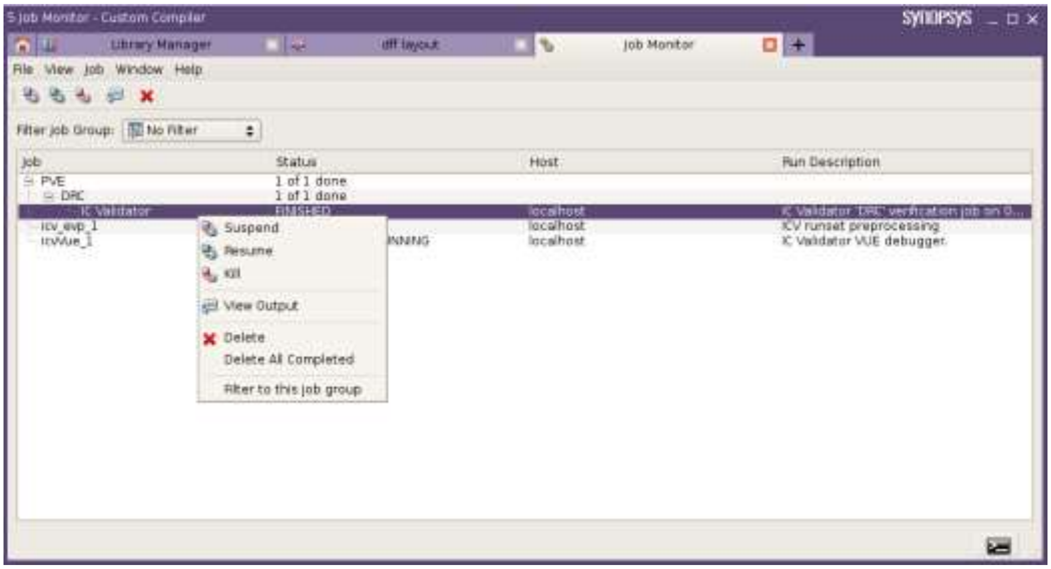
Task 21. *ICV* DRC Run Observations

In this task we will observe the output of the ***ICV*** DRC run during and after it completes.

32. As the options were set up in the previous task for the first time, pressing ***OK/Apply*** starts the ***ICV*** DRC job and stores the job setup in the run directory.

Note: The default run directory name is ***<cwd>/synopsys_custom/<cellName>.icv.drc***. If the run directory does not exist, it will be created and the job setup information will be stored in ***<drcRunDir>/prefs.xml*** when the job is executed. This allows the run settings to be stored and re-used on future runs for the same cellview.

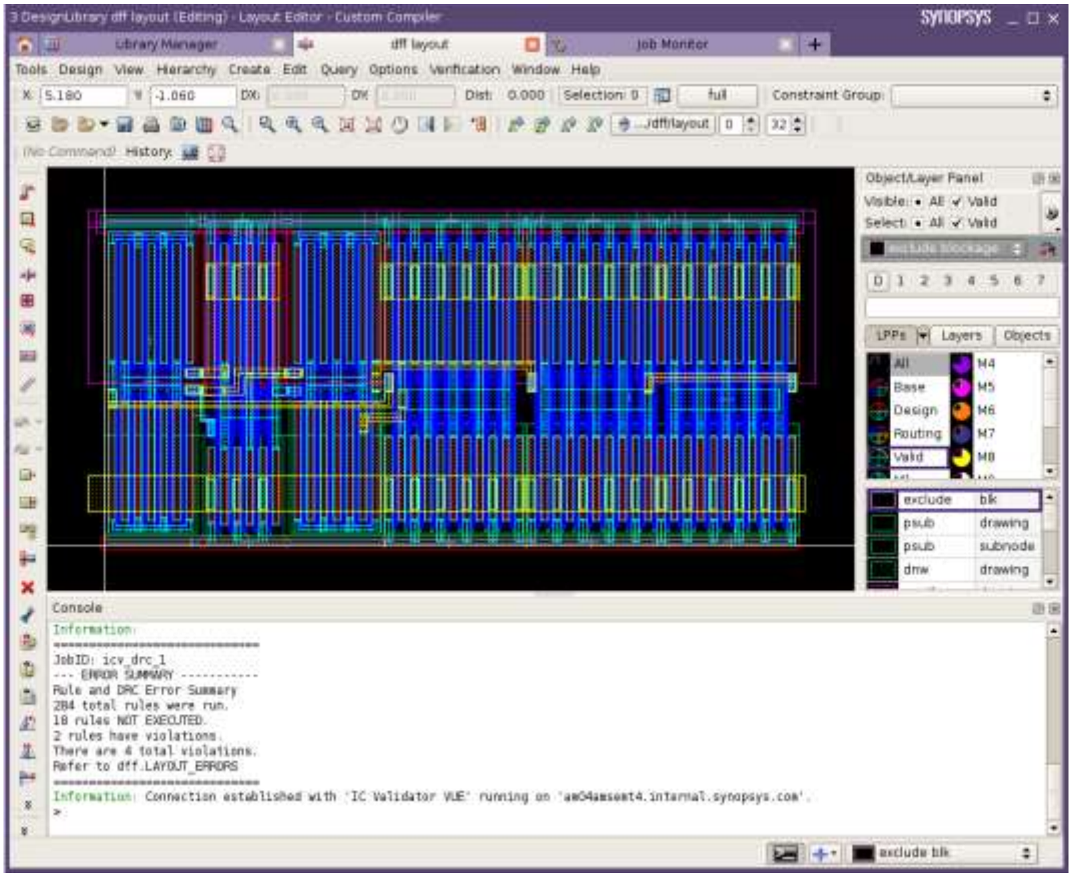
33. Invoke the job monitor using ***Home Page>Tools>Job Monitor***. Find this job in the ***Job Monitor Window*** and right mouse button click ***(RMB)*** to see the context sensitive menu ***(CSM).*** You can use the ***CSM*** to control the job (suspend, kill, view output, etc). Select V***iew Output*** to display the job's logfiles in the ***Custom Compiler Text Viewer Window***.



Job	Status	Host	Run Description
PVE	1 of 1 done		
DRC	1 of 1 done		
IC Validator	FINISHED	localhost	IC Validator 'DRC' verification job on 0...
icv_evp_1		localhost	ICV runset preprocessing
icvVue_1	NNING	localhost	IC Validator VUE debugger

Note: A job object gets created when launching jobs from the ***PVE*** interface. The job can be monitored using the ***Job Monitor Window***. The ***Job Monitor*** displays all jobs (batch and interactive) that were launched using ***Custom Compiler***'s xtJob infrastructure. You can ***RMB*** on the job to execute other job control actions from the ***CSM***.

34. The ***Console Assistant*** at the bottom of the ***LE Window*** will print a job summary with violation counts upon job completion as in the following figure.



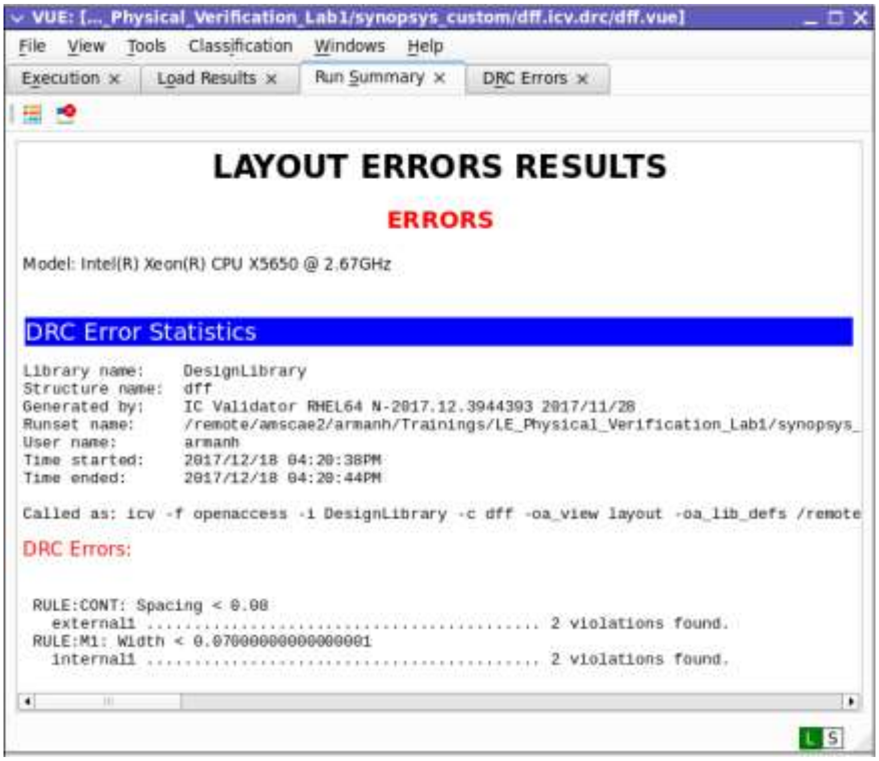
3 DesignLibrary dff layout (Editing) - Layout Editor - Custom Compiler
Library Manager
dff layout
Job Monitor
Tools Design View Hierarchy Create Edit Query Options Verification Window Help
X: 5.180 Y: -1.060 DX: DY: Dist: 0.000 Selection: 0 full Constraint Group:
(No Command) History:
Object/Layer Panel
Visible: All Valid
Select: All Valid
LPPs Layers Objects
All Base Design Routing Valid
M4 M5 M6 M7 M8
exclude blk
psub drawing
psub subnode
dnw drawing
Console
exclude blk

```
Information:
================================
JobID: icv_drc_1
--- ERROR SUMMARY -----------
Rule and DRC Error Summary
284 total rules were run.
18 rules NOT EXECUTED.
2 rules have violations.
There are 4 total violations.
Refer to dff.LAYOUT_ERRORS
================================
Information: Connection established with 'IC Validator VUE' running on 'am04amsemt4.internal.synopsys.com'.
>
```

35. Observe that the run directory ***synopsys_custom/dff.icv.drc*** was created in ***cwd*** upon completion of the DRC run.

36. Upon completion of the run, the ***ICV VUE*** violation debug environment will be automatically launched if any violations were found, as shown in the following figure.

Note: Automatic launch of the debugger is controlled by the ***Launch Debugger*** field in the setup dialog.



LAYOUT ERRORS RESULTS

ERRORS

Model: Intel(R) Xeon(R) CPU X5650 @ 2.67GHz

DRC Error Statistics

```
Library name:    DesignLibrary
Structure name:  dff
Generated by:    IC Validator RHEL64 N-2017.12.3944393 2017/11/28
Runset name:     /remote/amscae2/armanh/Trainings/LE_Physical_Verification_Lab1/synopsys_
User name:       armanh
Time started:    2017/12/18 04:20:38PM
Time ended:      2017/12/18 04:20:44PM

Called as: icv -f openaccess -i DesignLibrary -c dff -oa_view layout -oa_lib_defs /remote
```

DRC Errors:

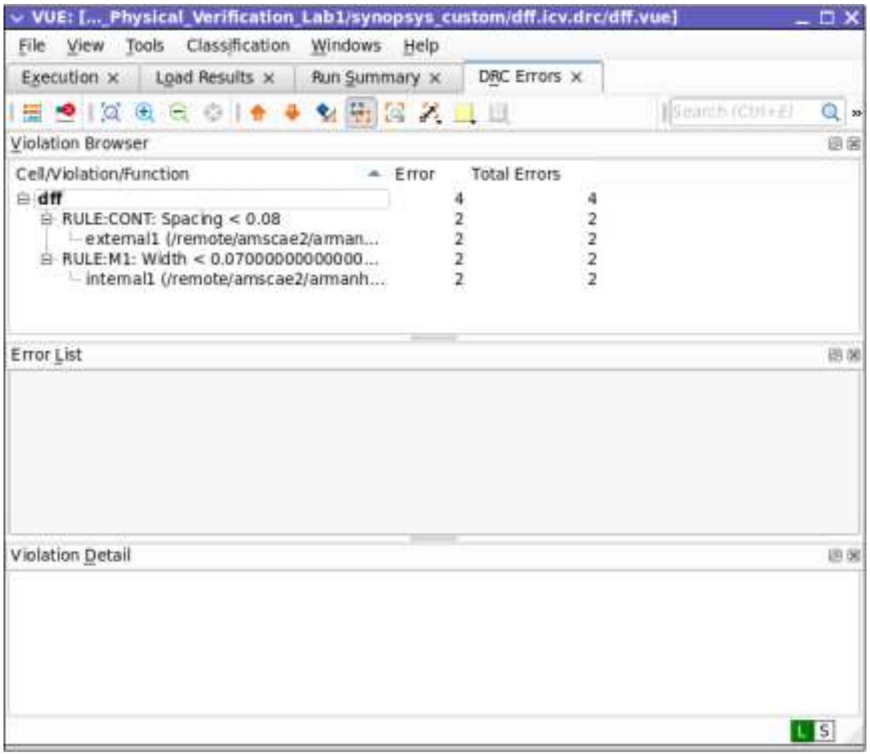
```
 RULE:CONT: Spacing < 0.08
   external1 ......................................... 2 violations found.
 RULE:M1: Width < 0.07000000000000001
   internal1 ......................................... 2 violations found.
```

Task 22. View and Debug DRC Violations Using ICV VUE

In this task we will view and debug DRC violations using ***ICV VUE***.

37. In the ***ICV VUE Window***, click on the ***DRC Errors Tab*** to view the violations.

38. Under the ***Cell/Violation/Function*** column, click ***LMB*** the “+” symbol next to the cell ***dff*** to expand the tree and view the violations in this cellview as in the following figure.

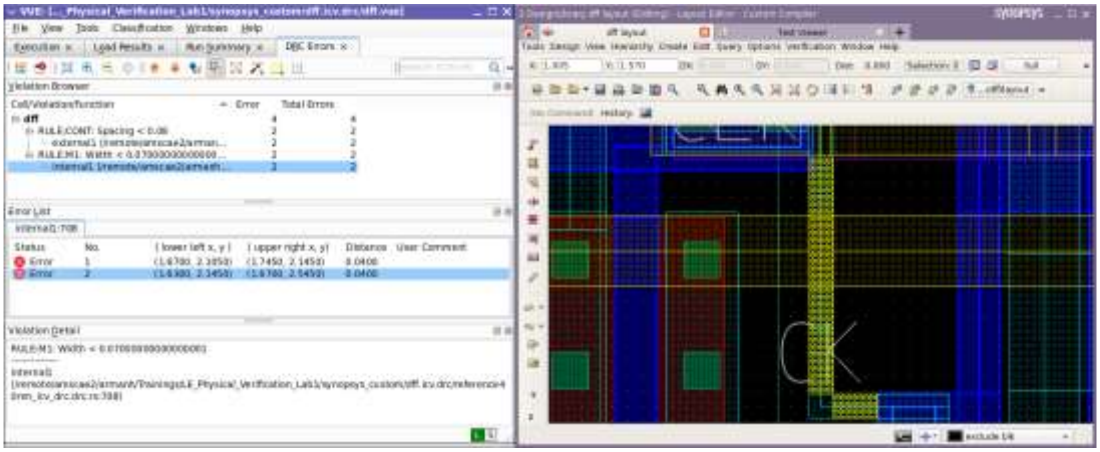

VUE: [..._Physical_Verification_Lab1/synopsys_custom/dff.icv.drc/dff.vue]

File View Tools Classification Windows Help
Execution × Load Results × Run Summary × DRC Errors ×
Search (Ctrl+E)
Violation Browser
Error List
Violation Detail
L S

Cell/Violation/Function	Error	Total Errors
⊟ **dff**	4	4
⊟ RULE:CONT: Spacing < 0.08	2	2
external1 (/remote/amscae2/arman...	2	2
⊟ RULE:M1: Width < 0.07000000000000...	2	2
internal1 (/remote/amscae2/armanh...	2	2

Note: The ***Cell/Violation/Function*** column in the ***Violation Browser Table***, lists violations in a tree format by cell→rule→function. The ***Error*** column represents the number of violation occurrences for each rule violation. When one violation row is selected, the ***Error List*** section at the bottom of the ***ICV VUE Window*** is populated with each violation occurrence. These occurrences can be selected to highlight in the ***LE*** canvas and display the violation details in the ***Violation Detail*** section of the ***ICV VUE Window***.

39. Select the rule ***M1: Width < 0.07*** and double click ***LMB*** to select the first violation occurrence of this rule in the ***Error List*** section to highlight in the ***LE*** canvas. Now highlight the second violation occurrence of the same rule. At this point, your ***LE Window*** should look like the following figure.


VUE: [..._Physical_Verification_Lab1/synopsys_custom/dff.icv.drc/dff.vue]

File View Tools Classification Windows Help
Execution × Load Results × Run Summary × DRC Errors ×
Violation Browser
Error List
Violation Detail
Tools Design View Hierarchy Create Edit Query Options Verification Window Help
dff layout
Text Viewer
synopsys
CK

Cell/Violation/Function	Error	Total Errors
⊟ **dff**	4	4
⊟ RULE:CONT: Spacing < 0.08	2	2
external1 (/remote/amscae2/arman...	2	2
⊟ RULE:M1: Width < 0.07000000000000...	2	2
internal1 (/remote/amscae2/armanh...	2	2

internal1:708

Status	No.	(lower left x, y)	(upper right x, y)	Distance	User Comment
Error	1	(1.6700, 2.1050)	(1.7450, 2.1450)	0.0400	
Error	2	(1.6300, 2.1450)	(1.6700, 2.5450)	0.0400	

RULE:M1: Width < 0.070000000000001

internal1
(/remote/amscae2/armanh/Trainings/LE_Physical_Verification_Lab1/synopsys_custom/dff.icv.drc/reference4
0nm_icv_drc.drc.rs:708)

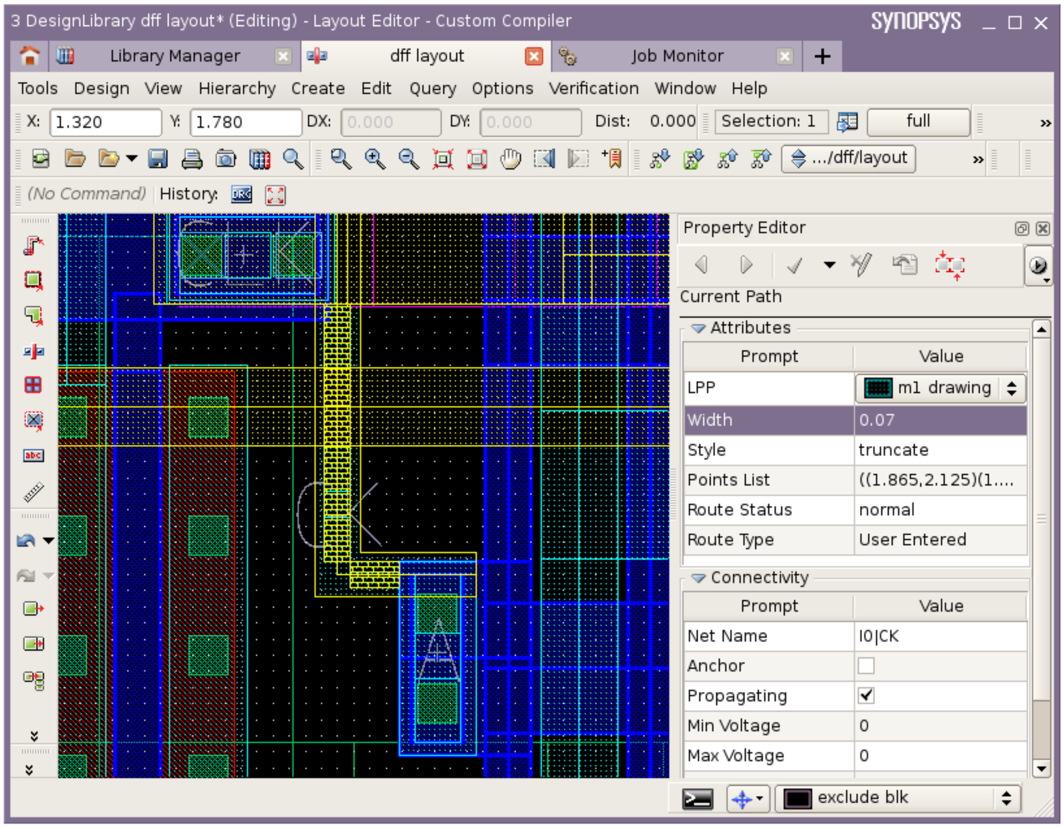
Task 23. Fix DRC Violations Using Custom Compiler LE

In this task we will fix DRC violations using ***Custom Compiler LE*** editing functions.

40. Make sure that the layout window of the cell ***dff*** is active.

41. Select the violating Path object, press ***Q*** to open the ***Property Editor (PE)***. In the ***PE***, change the width attribute of the Path from ***0.04*** to ***0.07*** and press ***Enter*** or click the green arrow at the top of the ***PE***.

42. Save the design. At this point, your ***LE Window*** should look like the following figure.


3 DesignLibrary dff layout* (Editing) - Layout Editor - Custom Compiler
SYNOPSYS
Library Manager
dff layout
Job Monitor
Tools Design View Hierarchy Create Edit Query Options Verification Window Help
full
.../dff/layout
History:
Property Editor
Current Path
Attributes
Connectivity
exclude blk

X: 1.320 Y: 1.780 DX: 0.000 DY: 0.000 Dist: 0.000 Selection: 1

(No Command)

Prompt	Value
LPP	m1 drawing
Width	0.07
Style	truncate
Points List	((1.865,2.125)(1....
Route Status	normal
Route Type	User Entered

Prompt	Value
Net Name	I0|CK
Anchor	□
Propagating	☑
Min Voltage	0
Max Voltage	0

43. Close the ***ICV VUE Window***.

Task 24. Analyze the Run Directory and Rerun DRC

In this task we will analyze the run directory and rerun ***ICV*** DRC on the modified ***dff*** layout.

44. In the Unix terminal change the working directory to ***./synopsys_custom/dff.icv.drc***.

45. Use the unix command ***ls*** to list the directory contents.

```
Unix% cd synopsys_custom/dff.icv.drc/
```

```
Unix% ls
Unix% prefs.xml dff.LAYOUT_ERRORS  run_details   ..
List of output files
```

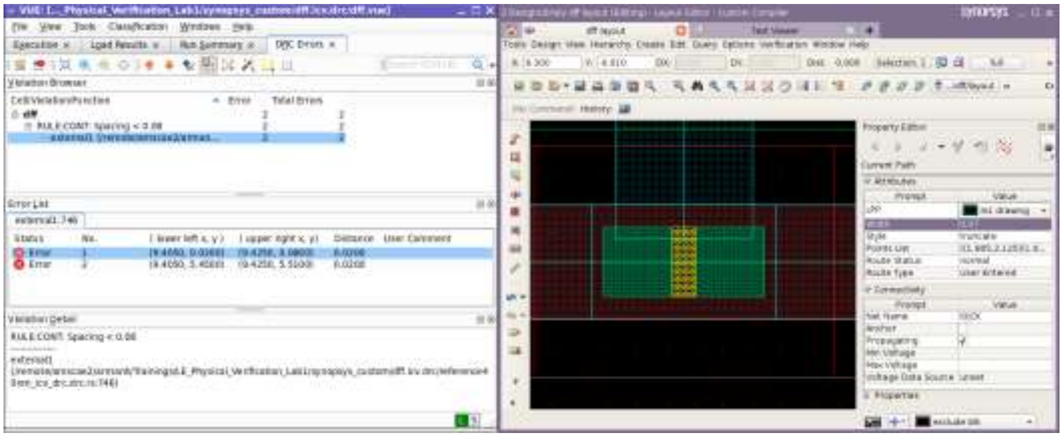
46. Open the ***prefs.xml*** file and note the contents of the file.

Note: The settings from the ***ICV*** DRC run in Task 2 are stored as preferences. In future runs, these preferences will re-populate the dialog fields when it is re-opened from the preferences stored in the dialog's ***RunDir*** field. Observe that all other files related to the job including ***ICV*** output files and logfiles are saved to the same run directory.

47. Rerun ***ICV*** DRC using the command ***Verification > DRC > Run***. When prompted to save changes to the design before running, choose ***OK*** to save. The same setup as before has been saved to the run directory and will be used for this run.

Note: You can rerun DRC by simply clicking DRC button on the left toolbar.

48. At this point, you should have 2 violations left of the rule ***CONT: Spacing < 0.08*** as in the following figure.



Cell/Violation/Function | Error | Total Errors
dff | 2 | 2
RULE CONT: Spacing < 0.08 | 2 | 2

external1.746

Status	No.	(lower left x, y)	(upper right x, y)	Distance	User Comment
Error	1	(9.4050, 0.0300)	(9.4250, 0.0800)	0.0200	
Error	2	(9.4050, 5.4500)	(9.4250, 5.5100)	0.0200	

RULE CONT: Spacing < 0.08

49. To fix the remaining contact spacing violations, you will need to open the sub block layout for ***inv4***, and adjust the parameters of the nwell guardring and the p-substrate guardring, until you get rid of the 2 contact spacing violations. After making the edits, use your knowledge from ***Tasks 2 > 6*** to edit the design, rerun ***ICV*** DRC, debug using ***ICV VUE***, until no violations remain.

50. If the ***ICV*** DRC run is clean, ***ICV VUE*** will not be launched at the end of the run and you will see the message in the Console Assistant "Completed with no errors".

Congratulations!

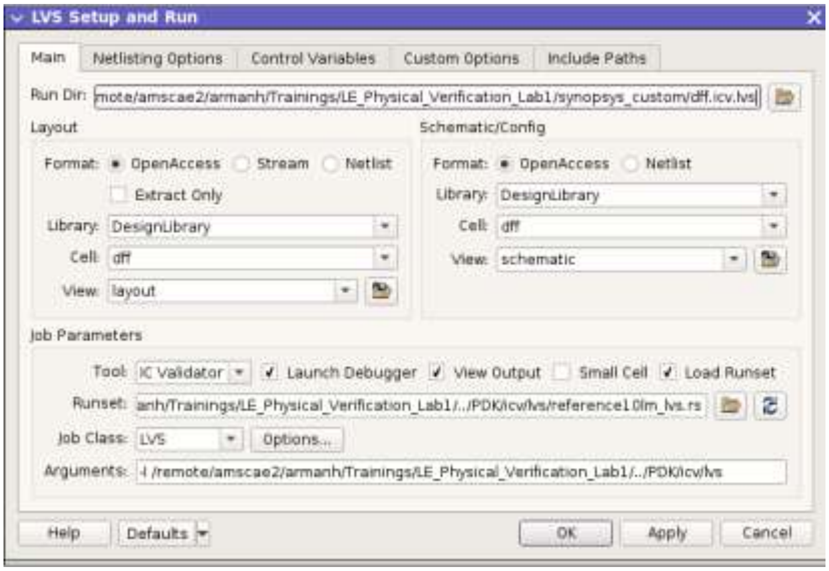
You have successfully completed DRC on the ***dff*** cell!

Task 25. Set Up ICV LVS Options

To run LVE in ***ICV***, a set of input files and options are required. In this task we will set up the required inputs and run LVS.

51. Open the cellview ***DesignLibrary/dff/layout*** if not already open in ***LE***. If any other windows are open, close them. Hit the ***ctrl-F*** bindkey to display all hierarchy levels.

52. Invoke the ***LVS Setup Dialog*** using the command ***Verification > LVS > Setup and Run...*** as in the following figure.



Run Dir: mote/amscae2/armanh/Trainings/LE_Physical_Verification_Lab1/synopsys_custom/dff.icv.lvs

Layout

Format: ● OpenAccess ○ Stream ○ Netlist

□ Extract Only

Library: DesignLibrary

Cell: dff

View: layout

Schematic/Config

Format: ● OpenAccess ○ Netlist

Library: DesignLibrary

Cell: dff

View: schematic

Job Parameters

Tool: IC Validator ☑ Launch Debugger ☑ View Output □ Small Cell ☑ Load Runset

Runset: anh/Trainings/LE_Physical_Verification_Lab1/../PDK/icv/lvs/reference10lm_lvs.rs

Job Class: LVS Options...

Arguments: -I /remote/amscae2/armanh/Trainings/LE_Physical_Verification_Lab1/../PDK/icv/lvs

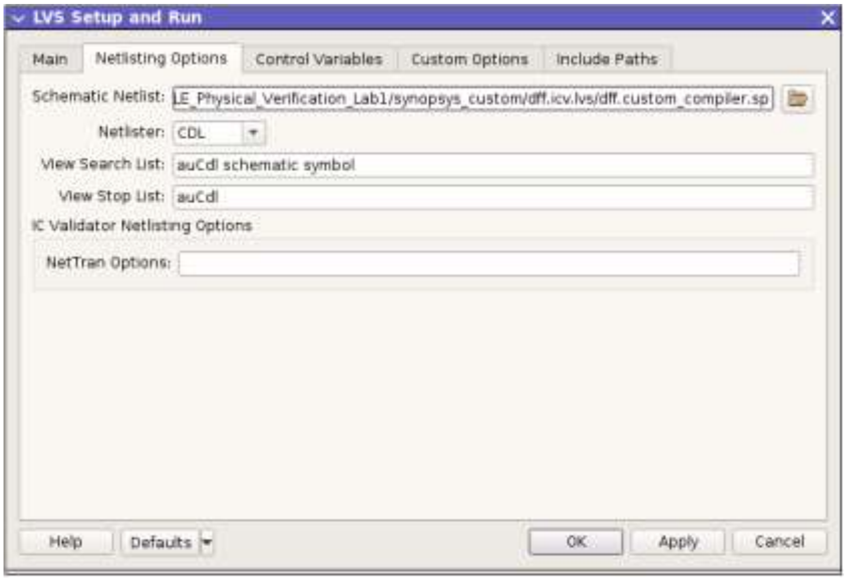
Note: Default values for some of the fields in the setup dialog are already prefilled. Observe that the ***Run Dir*** field is filled by the value ***./synopsys_custom<current working directory>/<cellname>.<pvtool>.lvs.*** The fields ***Library***, ***Cell***, ***View*** in the ***layout*** section are filled by the active cellview.

Note: Observe that process design kit (PDK) specific information such as the ***Runset*** field in the ***Main*** tab, and the ***Layer Map*** field in the ***Custom Options Tab***. These are set in the PDK configuration file.

53. Open the pdk configuration file from ***./../PDK/pdk.xml*** and check the ICV LVS job section to see how the default settings were populated for this PDK. Make sure that the files in the ***Runset*** field in the ***Main Tab*** and the ***Layer Map*** field in the ***Custom Options Tab*** exist and are set to the values specified in the PDK configuration file.

54. Observe that the ***ICV*** LVS job section of the PDK configuration file has ***Load Runset*** set to true which also shows in the setup dialog. This means that the ***PVE*** interface will parse the runset, to extract information about ***Control Variables*** and ***Custom Options***. These can be seen and set by the user in the ***Control Variables Tab*** and the ***Custom Options Tab*** of the setup dialog. These variables and values came from the provided runset and can be overridden by the user if desired.

55. Click ***LMB*** to select the ***Netlisting Options Tab*** and observe the netlisting settings needed to export the LVS netlist.



Schematic Netlist: LE_Physical_Verification_Lab1/synopsys_custom/dff.icv.lvs/dff.custom_compiler.sp

Netlister: CDL

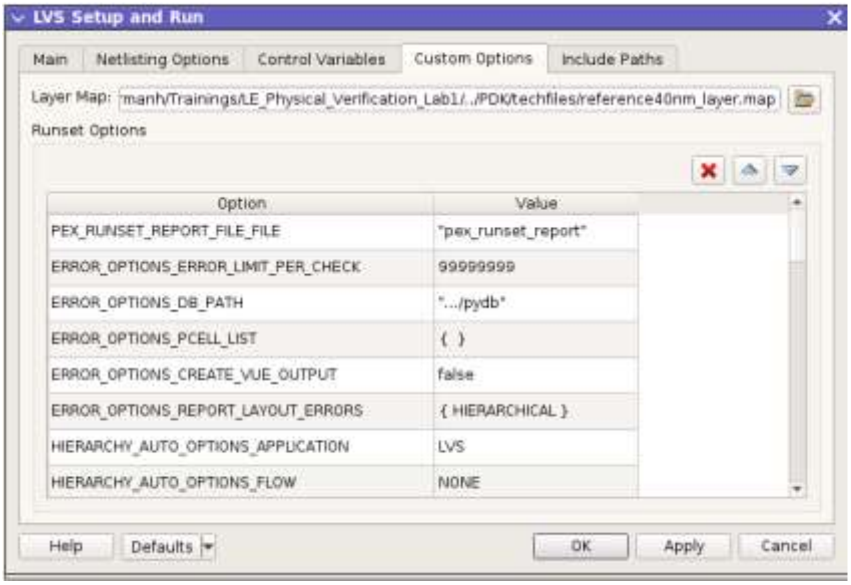
View Search List: auCdl schematic symbol

View Stop List: auCdl

IC Validator Netlisting Options

NetTran Options:

56. Click ***LMB*** to select the ***Custom Options Tab*** and observe that the form is filled in with the options defined in the runset.



Layer Map: manh/Trainings/LE_Physical_Verification_Lab1/../PDK/techfiles/reference40nm_layer.map

Runset Options

Option	Value
PEX_RUNSET_REPORT_FILE_FILE	"pex_runset_report"
ERROR_OPTIONS_ERROR_LIMIT_PER_CHECK	99999999
ERROR_OPTIONS_DB_PATH	".../pydb"
ERROR_OPTIONS_PCELL_LIST	{ }
ERROR_OPTIONS_CREATE_VUE_OUTPUT	false
ERROR_OPTIONS_REPORT_LAYOUT_ERRORS	{ HIERARCHICAL }
HIERARCHY_AUTO_OPTIONS_APPLICATION	LVS
HIERARCHY_AUTO_OPTIONS_FLOW	NONE

57. Leave the rest of the fields as is and make sure that the ***Launch Debugger*** field is enabled. This will automatically launch the ***ICV VUE*** debug environment upon completion of the run.

58. Click ***OK/Apply*** to start the run.

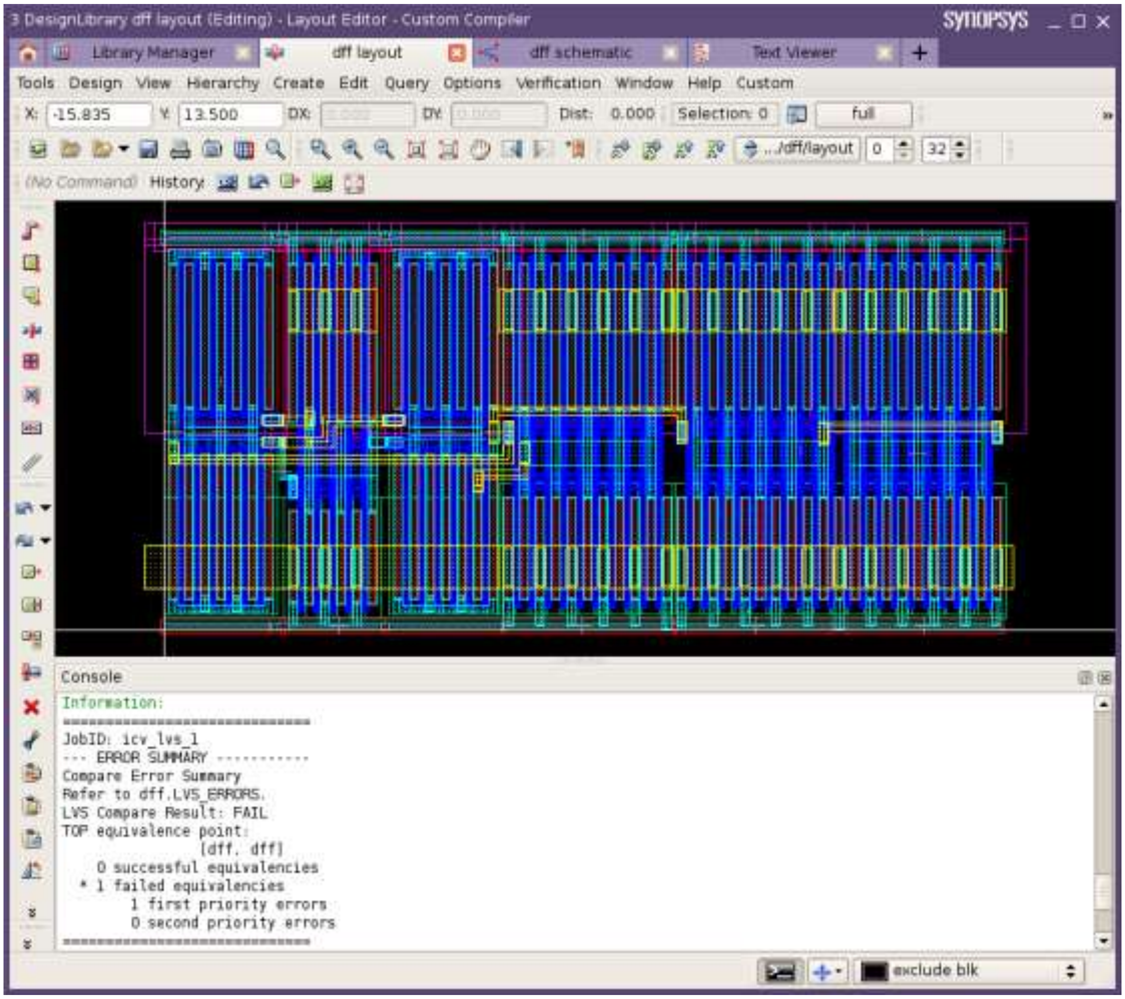
Task 26. ICV LVS Run Observations

In this task, we will observe the output of the ***ICV*** LVS run during and after it completes.

59. As the options were set in the previous task for the first time, pressing OK starts the ***ICV*** LVS job and stores the job setup in the run directory.

60. Open the ***Job Monitor Window*** view the progress and outputs of the job, similar to how you did in ***Task 3***, ***step 2***.

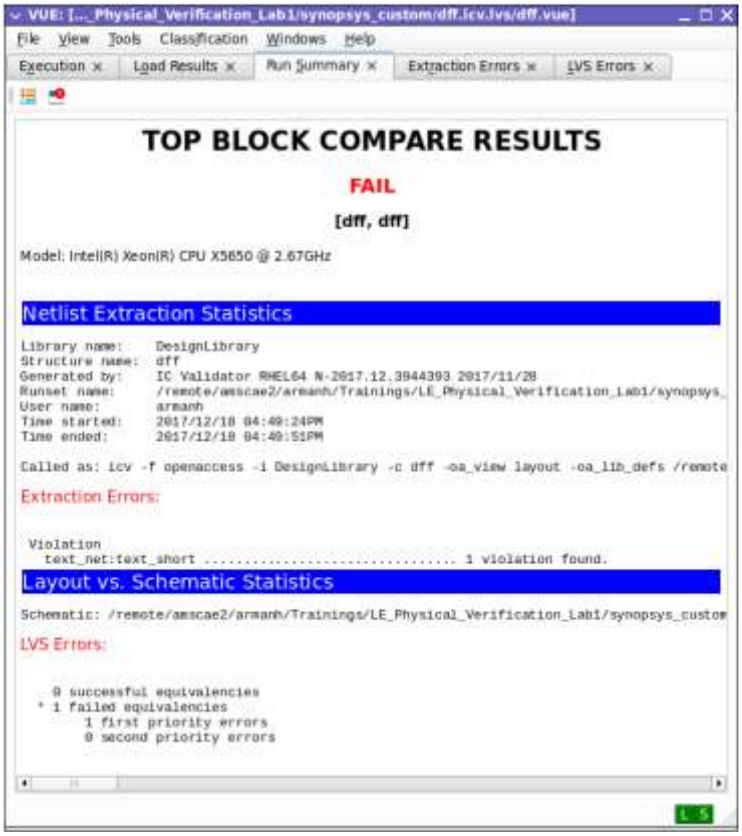
61. Upon job completion, the ***Console Assistant*** displays the status of the ***ICV*** run and violations as shown in the following figure. You can see that a short was found.


3 DesignLibrary dff layout (Editing) - Layout Editor - Custom Compiler
SYNOPSYS
Library Manager
dff layout
dff schematic
Text Viewer
Tools Design View Hierarchy Create Edit Query Options Verification Window Help Custom
X: -15.835 Y: 13.500 DX: DY: Dist: 0.000 Selection: 0 full
../dff/layout 0 32
(No Command) History
Console
exclude blk

```
Information:
==============================
JobID: icv_lvs_1
--- ERROR SUMMARY -----------
Compare Error Summary
Refer to dff.LVS_ERRORS.
LVS Compare Result: FAIL
TOP equivalence point:
                 [dff, dff]
    0 successful equivalencies
  * 1 failed equivalencies
        1 first priority errors
        0 second priority errors
==============================
```

62. Observe that the run directory ***synopsys_custom/dff.icv.lvs*** was created in ***cwd*** upon completion of the LVS run. Observe that the netlisting was successful and the ***ICV*** LVS comparison flagged one violation, a short.

63. ***ICV VUE*** is launched automatically if any violations were flagged, as in the following figure.


VUE: [..._Physical_Verification_Lab1/synopsys_custom/dff.icv.lvs/dff.vue]
File View Tools Classification Windows Help
Execution × Load Results × Run Summary × Extraction Errors × LVS Errors ×
L S

TOP BLOCK COMPARE RESULTS

FAIL

[dff, dff]

Model: Intel(R) Xeon(R) CPU X5650 @ 2.67GHz

Netlist Extraction Statistics

```
Library name:    DesignLibrary
Structure name:  dff
Generated by:    IC Validator RHEL64 N-2017.12.3944393 2017/11/28
Runset name:     /remote/amscae2/armanh/Trainings/LE_Physical_Verification_Lab1/synopsys_
User name:       armanh
Time started:    2017/12/18 04:49:24PM
Time ended:      2017/12/18 04:49:51PM

Called as: icv -f openaccess -i DesignLibrary -c dff -oa_view layout -oa_lib_defs /remote
```

Extraction Errors:

```
 Violation
   text_net:text_short ................................ 1 violation found.
```

Layout vs. Schematic Statistics

```
Schematic: /remote/amscae2/armanh/Trainings/LE_Physical_Verification_Lab1/synopsys_custom
```

LVS Errors:

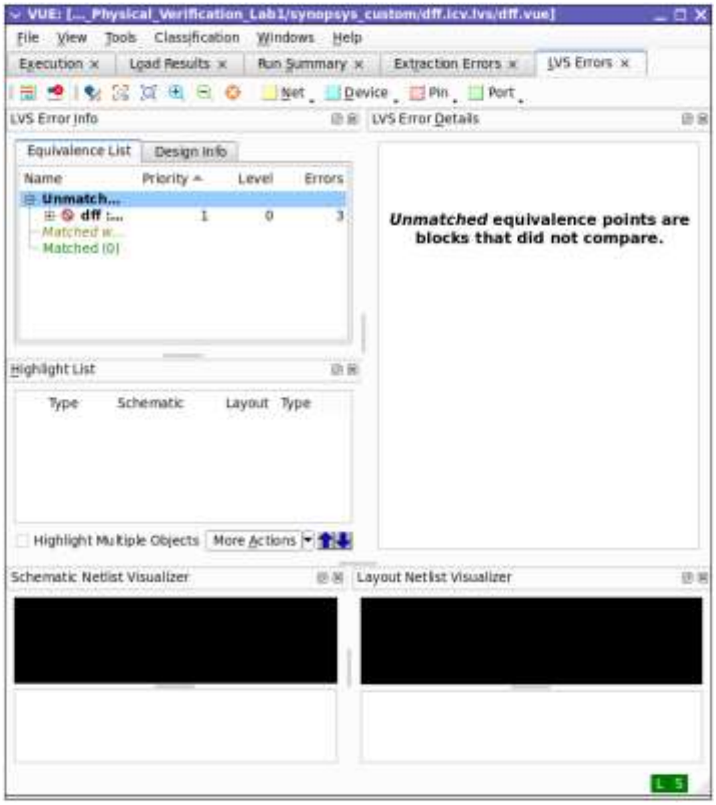
```
    0 successful equivalencies
  * 1 failed equivalencies
        1 first priority errors
        0 second priority errors
```

Task 27. View and Debug LVS Violations Using ICV VUE

In this task we will view and debug LVS violations using ***ICV VUE***.

64. In the ***ICV VUE Window***, click ***LMB*** the ***LVS Errors Tab***.

65. Blocks that failed LVS comparison are listed in the ***Equivalence List Tab*** in the ***LVS Error Info Section*** as in the following figure.


VUE: [..._Physical_Verification_Lab1/synopsys_custom/dff.icv.lvs/dff.vue]

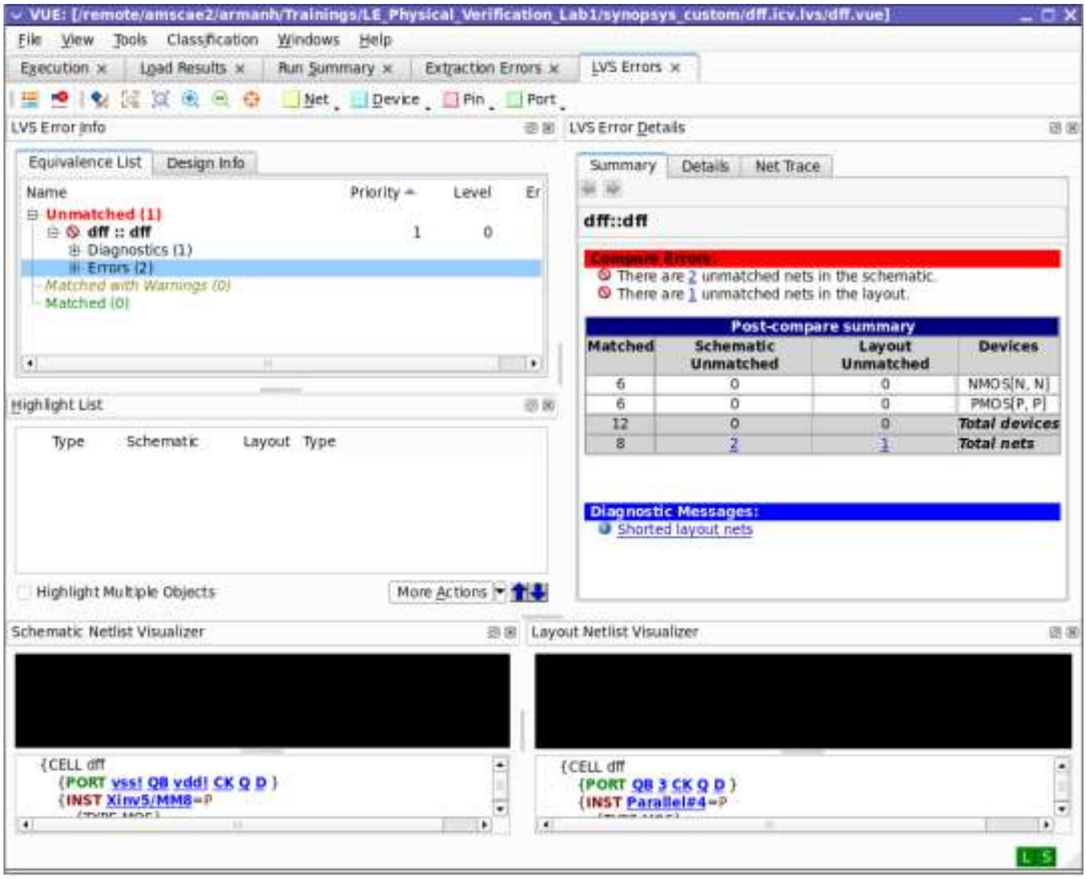
File View Tools Classification Windows Help
Execution × Load Results × Run Summary × Extraction Errors × LVS Errors ×
Net Device Pin Port
LVS Error Info
LVS Error Details
Equivalence List Design Info
Highlight List
Highlight Multiple Objects
More Actions
Schematic Netlist Visualizer
Layout Netlist Visualizer
L 5

Name	Priority	Level	Errors
Unmatch...			
dff :...	1	0	3
Matched w...			
Matched (0)			

Type	Schematic	Layout	Type

Unmatched **equivalence points are blocks that did not compare.**

66. In the ***Equivalence List Tab***, click ***LMB*** the "+" icon next to ***dff::dff*** to expand the tree, then click ***LMB*** to select ***Errors*** to see the LVS violations flagged for this block. At this point, the ***ICV VUE*** window should look like the following figure.


VUE: [/remote/amscae2/armanh/Trainings/LE_Physical_Verification_Lab1/synopsys_custom/dff.icv.lvs/dff.vue]
File View Tools Classification Windows Help
Execution × Load Results × Run Summary × Extraction Errors × LVS Errors ×
Net Device Pin Port
LVS Error Info
Equivalence List Design Info
Name Priority Level Er
Unmatched (1)
dff :: dff 1 0
Diagnostics (1)
Errors (2)
Matched with Warnings (0)
Matched (0)
Highlight List
Type Schematic Layout Type
Highlight Multiple Objects
More Actions
LVS Error Details
Summary Details Net Trace
dff::dff
Schematic Netlist Visualizer
Layout Netlist Visualizer
L S

Compare Errors:

There are 2 unmatched nets in the schematic.

There are 1 unmatched nets in the layout.

Post-compare summary			
Matched	Schematic Unmatched	Layout Unmatched	Devices
6	0	0	NMOS[N, N]
6	0	0	PMOS[P, P]
12	0	0	*Total devices*
8	2	1	*Total nets*

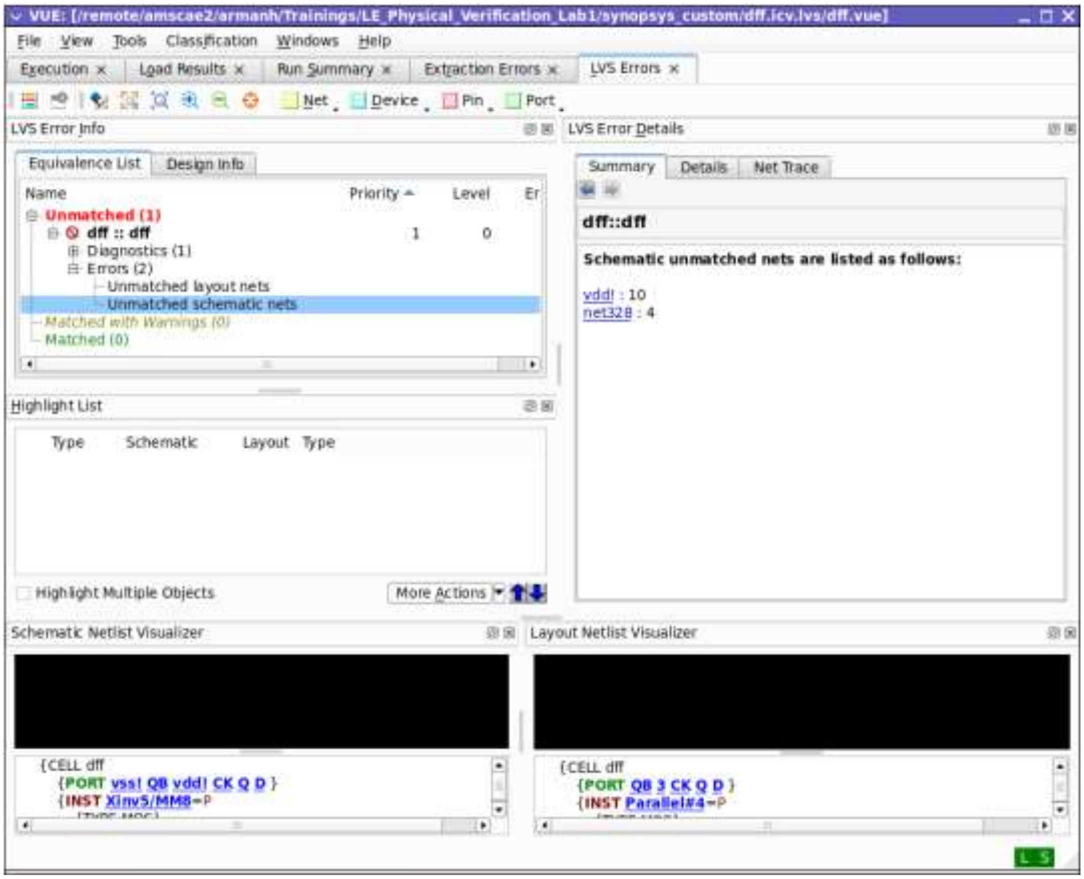
Diagnostic Messages:

Shorted layout nets

```
{CELL dff
  {PORT vss! QB vdd! CK Q D }
  {INST Xinv5/MM8=P
```

```
{CELL dff
  {PORT QB 3 CK Q D }
  {INST Parallel#4=P
```

67. In the ***ICV VUE Window***, expand the ***Errors*** item in the ***Equivalence List Section***, and select ***Unmatched schematic nets***. Then in the ***Summary Tab*** on the right, you should see 2 unmatched schematic nets: ***vdd!*** and ***net328*** as in the following figure.


VUE: [/remote/amscae2/armanh/Trainings/LE_Physical_Verification_Lab1/synopsys_custom/dff.icv.lvs/dff.vue]
File View Tools Classification Windows Help
Execution Load Results Run Summary Extraction Errors LVS Errors
Net Device Pin Port
LVS Error Info
Equivalence List Design Info
Name Priority Level Er
Unmatched (1)
dff :: dff 1 0
Diagnostics (1)
Errors (2)
Unmatched layout nets
Unmatched schematic nets
Matched with Warnings (0)
Matched (0)
Highlight List
Type Schematic Layout Type
Highlight Multiple Objects
More Actions
LVS Error Details
Summary Details Net Trace
Schematic Netlist Visualizer
Layout Netlist Visualizer

dff::dff

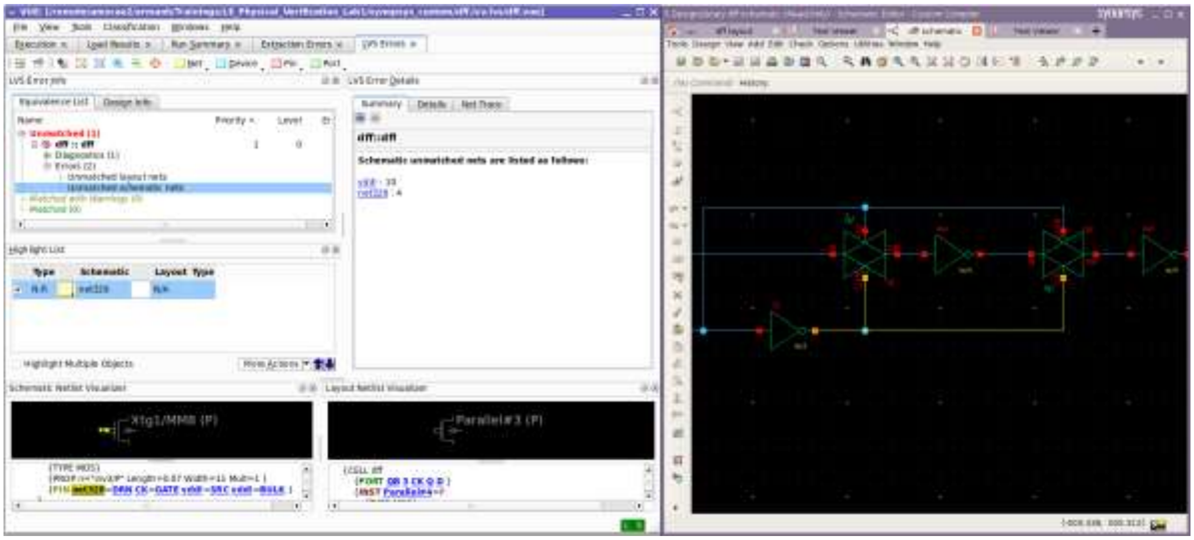
Schematic unmatched nets are listed as follows:

vdd! : 10
net328 : 4

```
{CELL dff
  {PORT vss! QB vdd! CK Q D }
  {INST Xinv5/MM8=P
```

```
{CELL dff
  {PORT QB 3 CK Q D }
  {INST Parallel#4=P
```

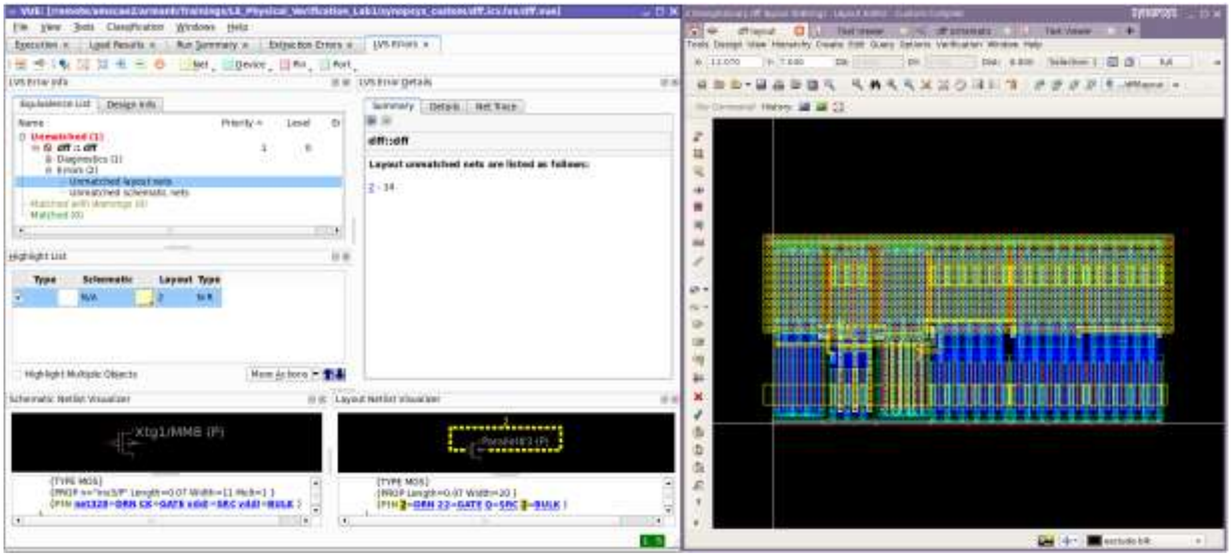
68. Click ***LMB*** on one of the nets. This highlights the selected nets in the ***Custom Compiler Schematic Editor (SE)*** and in ***ICV VUE***’s the ***Schematic Netlist Visualizer***, as shown in the figure below. Note that in this case, the net is not highlighted ***ICV VUE***’s ***Layout Netlist Visualizer*** or in ***Custom Compiler LE***, because it was not matched to any equivalent layout net.


Equivalence List Design Info
Unmatched (1)
dff :: dff
Unmatched layout nets
Unmatched schematic nets
Summary Details Net Trace
dff::dff
Schematic unmatched nets are listed as follows:
vdd! : 10
net328 : 4
Highlight List
Type Schematic Layout Type
N/A net328 N/A
Highlight Multiple Objects
More Actions
Schematic Netlist Visualizer
Layout Netlist Visualizer
Xig1/MM8 (P)
Parallel#3 (P)

```
{CELL dff
  {PORT QB 3 CK Q D }
  {INST Parallel#4=P
```

Net *net328* highlighted in schematic

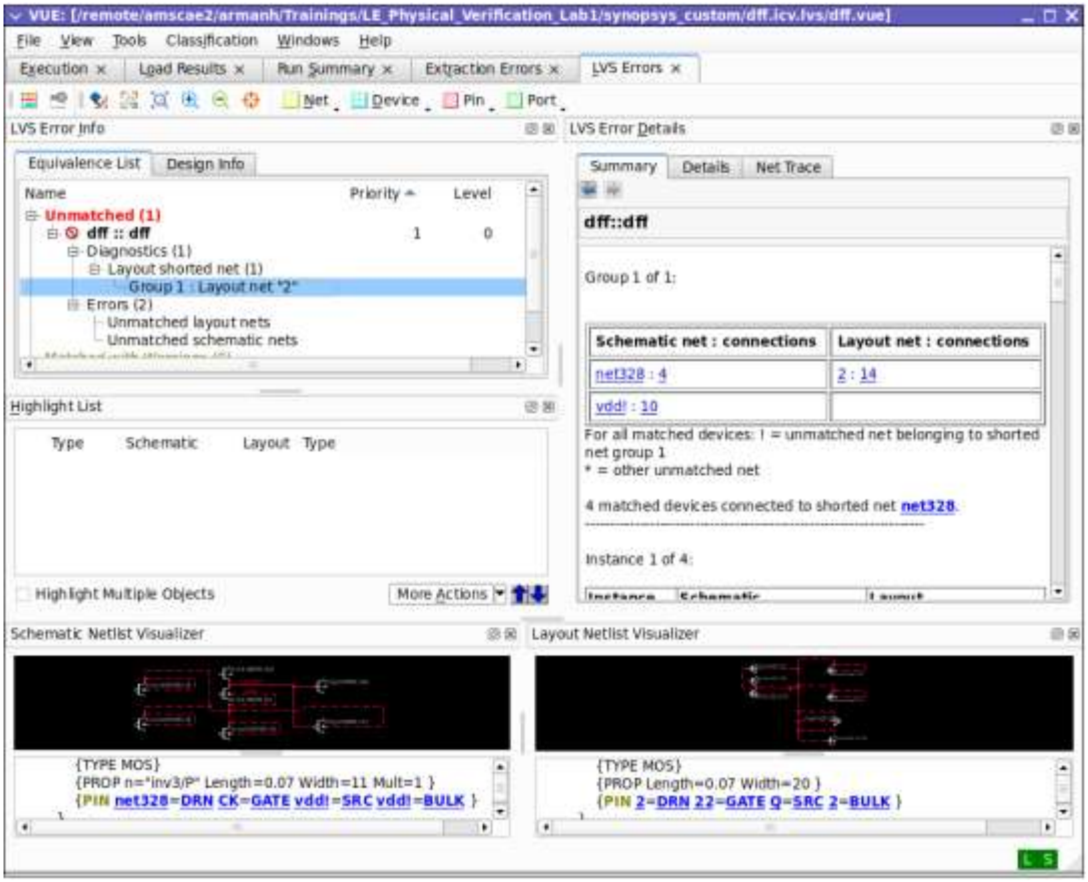
69. Similarly select ***Unmatched layout nets*** in the ***Equivalence List Section***, then select the net ***2*** listed in the ***Summary Tab***. This highlights the net in ***ICV VUE***’s ***Layout Netlist Visualizer*** and in ***Custom Compiler LE*** as in the following figure. Note that in this case, the net is not highlighted ***ICV VUE***’s ***Schematic Netlist Visualizer*** or in ***Custom Compiler SE***, because it was not matched to any equivalent schematic net.



Net *2* highlighted in nand layout

70. Clear the violation highlights from the layout canvas using the ***Clear Highlights Button*** near the top left part of the ***ICV VUE*** toolbar.

71. In the ***Equivalence List Tab*** of the ***LVS Errors Info Section*** in the ***ICV VUE Window***, expand and select ***Diagnostics*** > ***Layout Shorted Net*** > ***Group 1***. This displays the net short in the tabular form in the ***Summary Tab***, as in the following figure.


VUE: [/remote/amscae2/armanh/Trainings/LE_Physical_Verification_Lab1/synopsys_custom/dff.icv.lvs/dff.vue]
File View Tools Classification Windows Help
Execution × Load Results × Run Summary × Extraction Errors × LVS Errors ×
Net Device Pin Port
LVS Error Info
Equivalence List Design Info
LVS Error Details
Summary Details Net Trace
Highlight List
Highlight Multiple Objects
More Actions
Schematic Netlist Visualizer
Layout Netlist Visualizer
L S

Name	Priority	Level
⊟ **Unmatched (1)**		
⊟ **dff :: dff**	1	0
⊟ Diagnostics (1)		
⊟ Layout shorted net (1)		
Group 1 : Layout net "2"		
⊞ Errors (2)		
Unmatched layout nets		
Unmatched schematic nets		

Type Schematic Layout Type

dff::dff

Group 1 of 1:

Schematic net : connections	Layout net : connections
net328 : 4	2 : 14
vdd! : 10	

For all matched devices: ! = unmatched net belonging to shorted net group 1
* = other unmatched net

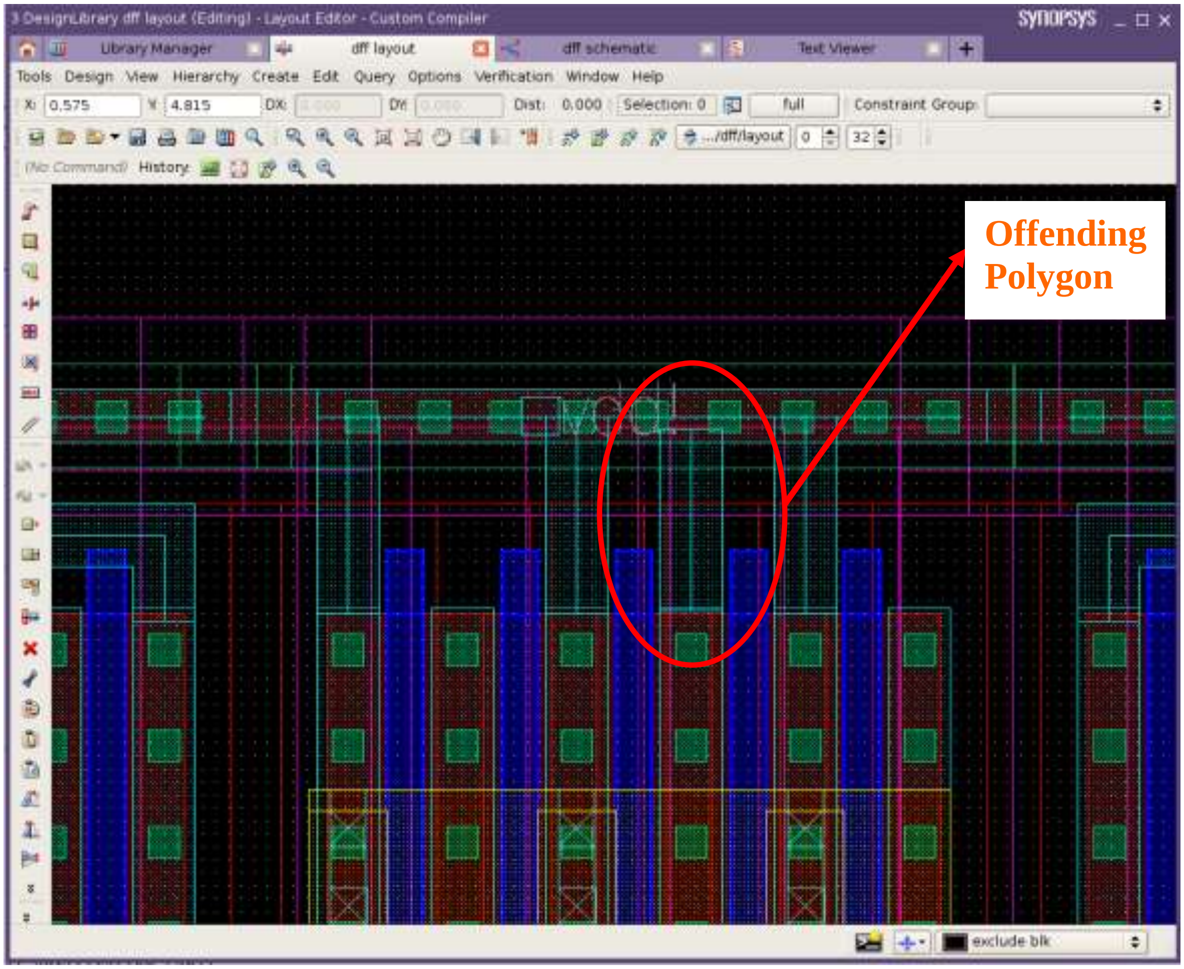
4 matched devices connected to shorted net **net328**.

Instance 1 of 4:

```
{TYPE MOS}
{PROP n="inv3/P" Length=0.07 Width=11 Mult=1 }
{PIN net328=DRN CK=GATE vdd!=SRC vdd!=BULK }
```

```
{TYPE MOS}
{PROP Length=0.07 Width=20 }
{PIN 2=DRN 22=GATE Q=SRC 2=BULK }
```

72. As you probably figured out, the schematic nets ***net328*** and ***vdd!*** Are shorted in the layout. The short is caused by an offending polygon as in the following figure.



73. Close the ***ICV VUE Window***.

Task 28. Fix LVS Violations Using Custom Compiler LE

In this task you will fix LVS violations using ***Custom Compiler LE*** editing functions.

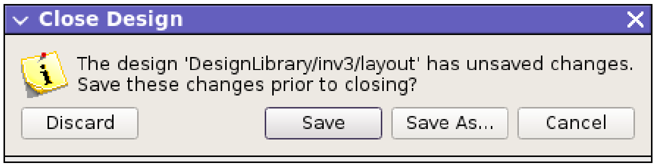
74. Make the edit to the ***inv3*** cellview instance using the command ***Hierarchy*** > ***Edit In Place***, then select the instance named ***I0|inv3***.

    Note: You will know that you are in Edit In Place mode inside of the ***inv3*** cellview by looking at the top window banner of the layout editor which has the name of the cellview being edited.

75. Fix the ***m1*** offending polygon by deleting the offending metal1 path.

76. Return to the top cell ***dff*** using the command ***Hierarchy*** > ***Return to Top***

77. When prompted, save the ***inv3*** layout.



The design 'DesignLibrary/inv3/layout' has unsaved changes.
Save these changes prior to closing?

78. Save the ***dff*** cellview.

Task 29. Analyze the Run Directory and Rerun LVS

In this task we will analyze the run directory and rerun ***ICV*** LVS on the modified ***dff*** layout.

79. In the unix terminal, change the working directory to ***./synopsys_custom/dff.icv.lvs***.

80. Use the unix command *ls* to list the contents of the directory.

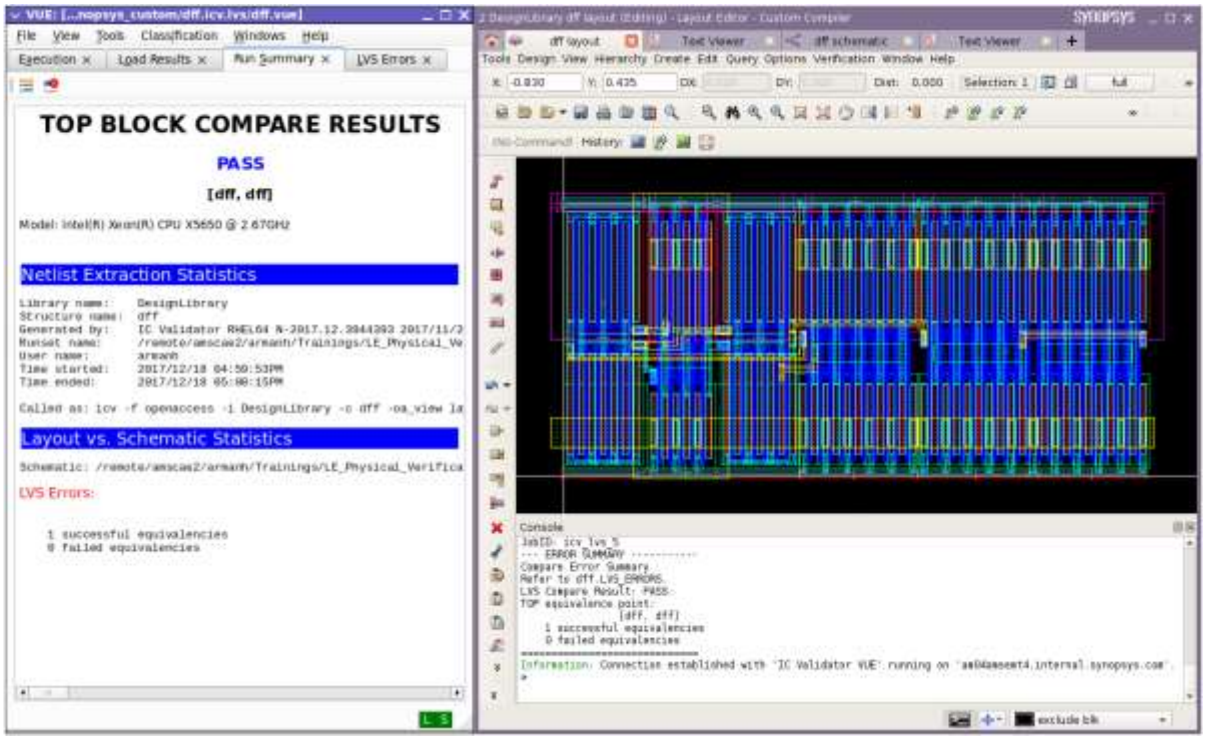
```
Unix% cd ./synopsys_custom/dff.icv.lvs
Unix% ls
Unix% prefs.xml MILKYWAY_OUTPUT
…. List of output files of ICV LVS
```

81. Open ***prefs.xml*** file and note that the setup for your previous run are stored in it. The same settings will be recalled and used next time you perform ***Verification > LVS > Run*** or ***Setup and Run…***

82. Rerun ***ICV*** LVS using the menu action ***Verification > LVS > Run***.

Note: You can also rerun ***ICV*** LVS with the previously used options and preferences simply by clicking LVS button on the left toolbar in the ***Custom Compiler LE Window***.

83. If the LVS comparison is clean, ***ICV VUE*** will be launched with ***PASS*** notification. Check the console for pass/fail messages to confirm that the there are no remaining LVS violations.



VUE: [...nopsys_custom/dff.icv.lvs/dff.vue]
File View Tools Classification Windows Help
Execution × Load Results × Run Summary × LVS Errors ×
Console
exclude blk

TOP BLOCK COMPARE RESULTS

PASS

[dff, dff]

Model: Intel(R) Xeon(R) CPU X5650 @ 2.67GHz

Netlist Extraction Statistics

```
Library name:    DesignLibrary
Structure name:  dff
Generated by:    IC Validator RHEL64 N-2017.12.3844393 2017/11/2
Runset name:     /remote/amscae2/armanh/Trainings/LE_Physical_Ve
User name:       armanh
Time started:    2017/12/18 04:59:53PM
Time ended:      2017/12/18 05:00:15PM

Called as: icv -f openaccess -i DesignLibrary -c dff -oa_view la
```

Layout vs. Schematic Statistics

```
Schematic: /remote/amscae2/armanh/Trainings/LE_Physical_Verifica
```

LVS Errors:

```
    1 successful equivalencies
    0 failed equivalencies
```

```
JobID: icv_lvs_5
--- ERROR SUMMARY -----------
Compare Error Summary
Refer to dff.LVS_ERRORS.
LVS Compare Result: PASS
TOP equivalence point:
                [dff, dff]
    1 successful equivalencies
    0 failed equivalencies
==============================
Information: Connection established with 'IC Validator VUE' running on 'ae04amsemt4.internal.synopsys.com'.
>
```

84. Close the ***dff*** cellview.

85. Exit ***Custom Compiler***.

Congratulations!

You have successfully performed LVS verification for the ***dff*** lay

4

4. Parasitic Extraction Interface

Learning Objectives

In this lab we will use the ***Physical Verification Environment (PVE)*** and the ***Custom Compiler Layout Editor (LE)*** to:

- Set up and customize the options for ***StarRC*** Layout Parasitic Extraction (LPE) runs
- Execute ***StarRC LPE*** from the ***Custom Compiler PVE*** environment
- View and analyze the extracted interconnect parasitics in the extracted cellview using ***Custom Compiler***'s ***Parasitics*** features.

⏰ **Lab Duration:**
30 minutes

Introduction

The ***Physical Verification Environment (PVE)*** in ***Custom Compiler*** provides easy to use and interactive integration with the ***StarRC*** LPE tool.

The ***PVE*** environment allows ***StarRC*** setup information to be pre-configured for a process or PDK, allows ***StarRC*** tool options to be imported from the command file, and stores run settings for ease of re-use for future runs. These features allow circuit and layout designers to conveniently and quickly run LPE from the ***Custom Compiler Layout Editor (LE)***.

We will set up and customize LPE options, and run the jobs with ***StarRC***. We will then use the ***Parasitics*** interactive debug and analysis features in the ***Custom Compiler Layout Editor (LE)*** to analyze the parasitics.

File Locations

All files for this lab are located in the directory **`LE_Parasitic_Extraction_Lab1`**

Directory Structure

`LE_Parasitic_Extraction_Lab1`	Current working directory
`sourceMe`	PDK environment setup file
`DesignLibrary`	OpenAccess Design library
`../PDK/icv/drc/reference40nm_icv_drc.rs`	ICV DRC Rule Deck
`../PDK/icv/lvs/reference10lm_lvs.rs`	ICV LVS Rule Deck
`../PDK/techfiles/reference40nm_layer.map`	OA Layer Mapping File
`../PDK/starrc/reference40_1P10M_typical.nxtgrd`	Interconnect Technology File
`../PDK/starrc/star_icv_cmd`	Star Command File
`../PDK/pdk.xml`	PDK Configuration File

Tools Versions

The following tool versions should be setup before running the lab:

IC Validator	M-2018.06
StarRC	M-2018.06

Instructions

Task 30. Start *Custom Compiler*

In this task, we will invoke ***Custom Compiler***. Before running Custom Compiler use sourceMe file to setup path to the PDK.

86. In the Unix window, change your working directory to **LE_Parasitic_Extraction_Interface_Lab1** and invoke ***Custom Compiler***.

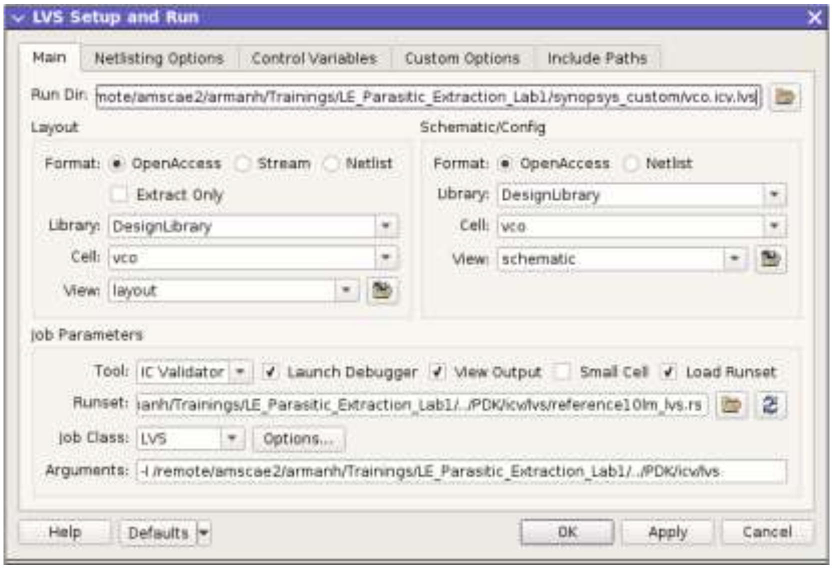
```
Unix% cd LE_Parasitic_Extraction_Lab1
Unix% custom_compiler &
```

Task 31. Run ICV LVS

In this task, we will run ***ICV*** LVS. The Milkway database generated by ***ICV*** in this step is a required input for ***StarRC***.

87. Open the cellview ***DesignLibrary/vco/layout*** from the ***Library Manager Window***. When the layout opens, hit the ***Shift-F*** bindkey to display all hierarchy levels.

88. Invoke the ***LVS Setup Dialog*** from the menu ***Verification > LVS > Setup and Run...***



Run Dir: mote/amscae2/armanh/Trainings/LE_Parasitic_Extraction_Lab1/synopsys_custom/vco.icv.lvs

Layout

Format: ● OpenAccess ○ Stream ○ Netlist

□ Extract Only

Library: DesignLibrary

Cell: vco

View: layout

Schematic/Config

Format: ● OpenAccess ○ Netlist

Library: DesignLibrary

Cell: vco

View: schematic

Job Parameters

Tool: IC Validator ☑ Launch Debugger ☑ View Output □ Small Cell ☑ Load Runset

Runset: ıanh/Trainings/LE_Parasitic_Extraction_Lab1/../PDK/icv/lvs/reference10lm_lvs.rs

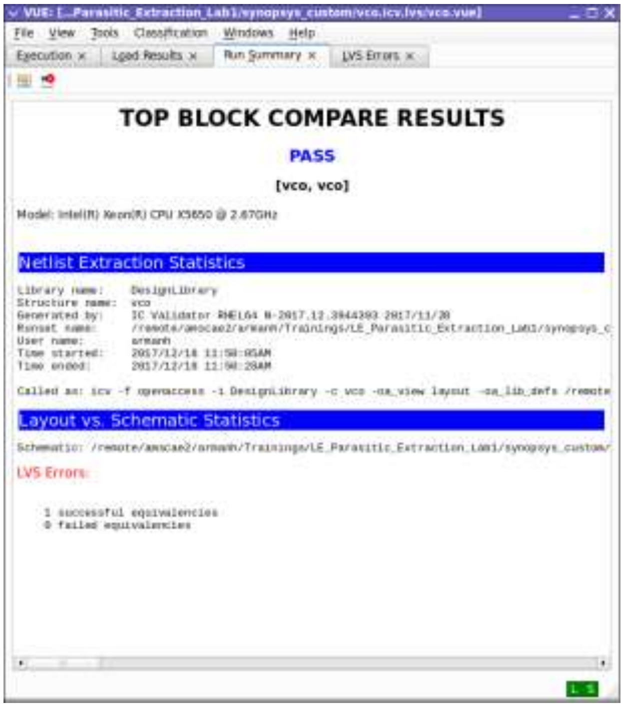
Job Class: LVS

Arguments: -I /remote/amscae2/armanh/Trainings/LE_Parasitic_Extraction_Lab1/../PDK/icv/lvs

89. All settings should already be pre-set and ready to run. You can double check them if desired

90. Click ***OK/Apply*** to start the run.

91. When the run completes, an ***ICV VUE Window*** will be automatically opened. Check the ***ICV VUE Window*** and the ***Custom Compiler Console Assistant*** to make sure there are no LVS violations. The design should be LVS clean.


VUE: [...Parasitic_Extraction_Lab1/synopsys_custom/vco.icv.lvs/vco.vue]
File View Tools Classification Windows Help
Execution × Load Results × Run Summary × LVS Errors ×

TOP BLOCK COMPARE RESULTS

PASS

[vco, vco]

Model: Intel(R) Xeon(R) CPU X5650 @ 2.67GHz

Netlist Extraction Statistics

```
Library name:    DesignLibrary
Structure name:  vco
Generated by:    IC Validator RHEL64 N-2017.12.3844393 2017/11/28
Runset name:     /remote/amscae2/armanh/Trainings/LE_Parasitic_Extraction_Lab1/synopsys_c
User name:       armanh
Time started:    2017/12/18 11:58:05AM
Time ended:      2017/12/18 11:58:28AM

Called as: icv -f openaccess -i DesignLibrary -c vco -oa_view layout -oa_lib_defs /remote
```

Layout vs. Schematic Statistics

```
Schematic: /remote/amscae2/armanh/Trainings/LE_Parasitic_Extraction_Lab1/synopsys_custom/
```

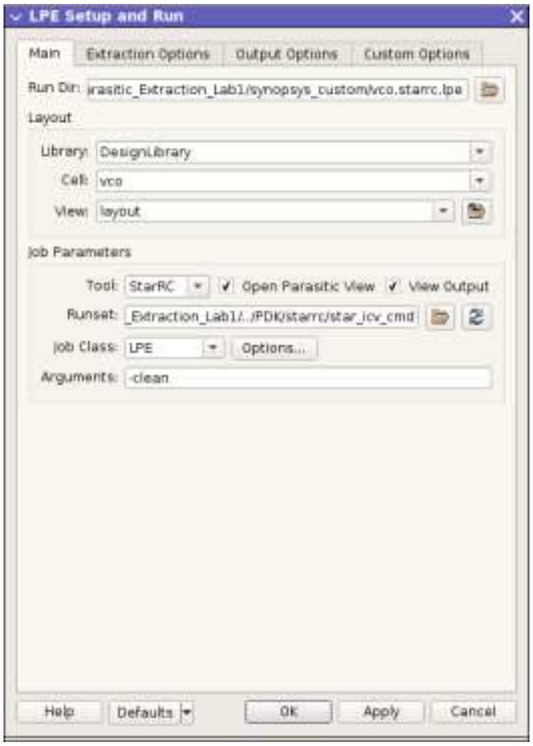
LVS Errors:

```
    1 successful equivalencies
    0 failed equivalencies
```

Task 32. Set Up *StarRC* LPE Options

To run LPE in ***StarRC***, a set of input files and options are required. In this task we will set up the required inputs and run LPE.

92. Invoke the ***LPE Setup and Run Dialog*** from the menu ***Verification > LPE > Setup and Run..***.



Run Dir: rasitic_Extraction_Lab1/synopsys_custom/vco.starrc.lpe

Layout

Library: DesignLibrary

Cell: vco

View: layout

Job Parameters

Tool: StarRC ☑ Open Parasitic View ☑ View Output

Runset: _Extraction_Lab1/../PDK/starrc/star_icv_cmd

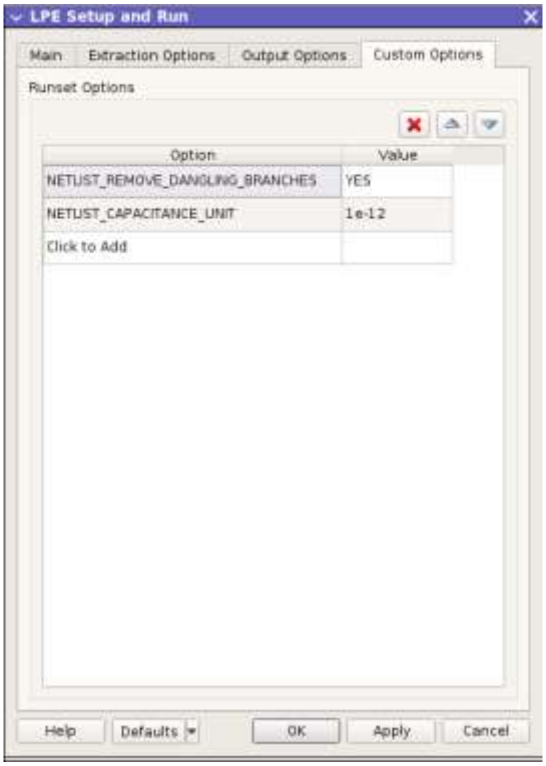
Job Class: LPE Options...

Arguments: -clean

Note: Default values for the dialog fields are prefilled. Observe that the ***Run Dir*** field is filled by the value ***<cwd>/synopsys_custom/<cellname>.starrc.lpe***. The fields ***Library***, ***Cell***, ***View*** in the layout section are filled in by the active cellview open in ***LE***.

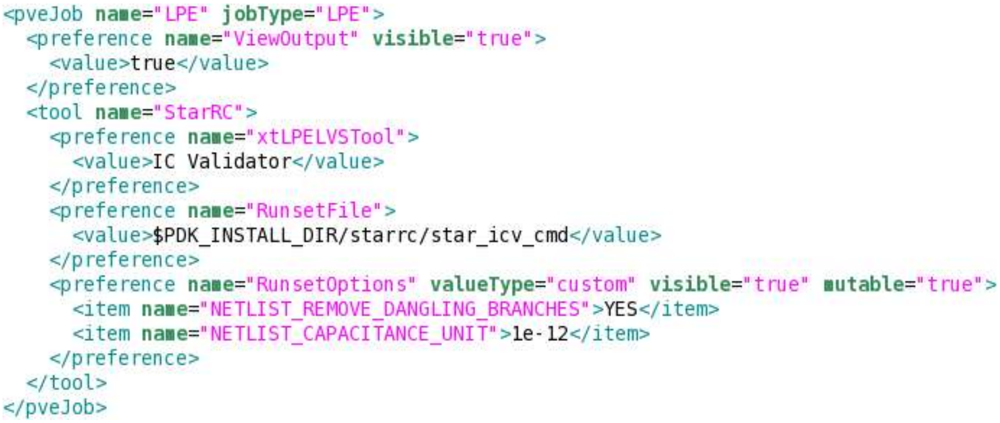
Note: Observe that PDK specific information is prefilled, such as the ***Runset*** field in the ***Main Tab***, and some tool options in the ***Custom Options Tab***. These were preset in the pdk configuration file.

93. Click left mouse button ***(LMB)*** to select the ***Custom Options Tab***. Open the pdk configuration file from ***../PDK/pdk.xml***. Look at the ***StarRC*** LPE job section to understand how these default settings were seeded into the ***Main and Custom Options Tabs*** for this PDK as in the following figures.



Runset Options

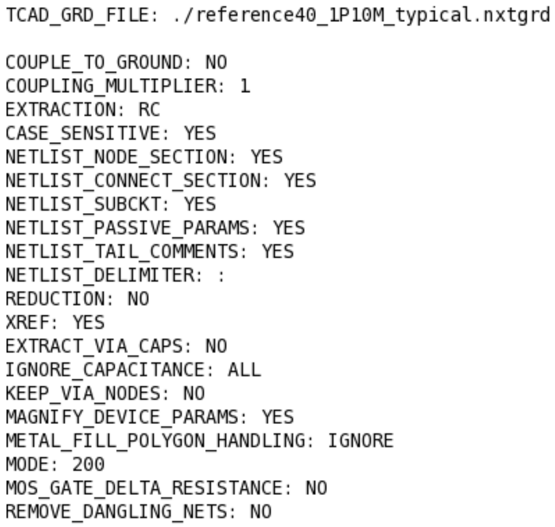
Option	Value
NETLIST_REMOVE_DANGLING_BRANCHES	YES
NETLIST_CAPACITANCE_UNIT	1e-12
Click to Add	



```
<pveJob name="LPE" jobType="LPE">
  <preference name="ViewOutput" visible="true">
    <value>true</value>
  </preference>
  <tool name="StarRC">
    <preference name="xtLPELVSTool">
      <value>IC Validator</value>
    </preference>
    <preference name="RunsetFile">
      <value>$PDK_INSTALL_DIR/starrc/star_icv_cmd</value>
    </preference>
    <preference name="RunsetOptions" valueType="custom" visible="true" mutable="true">
      <item name="NETLIST_REMOVE_DANGLING_BRANCHES">YES</item>
      <item name="NETLIST_CAPACITANCE_UNIT">1e-12</item>
    </preference>
  </tool>
</pveJob>
```

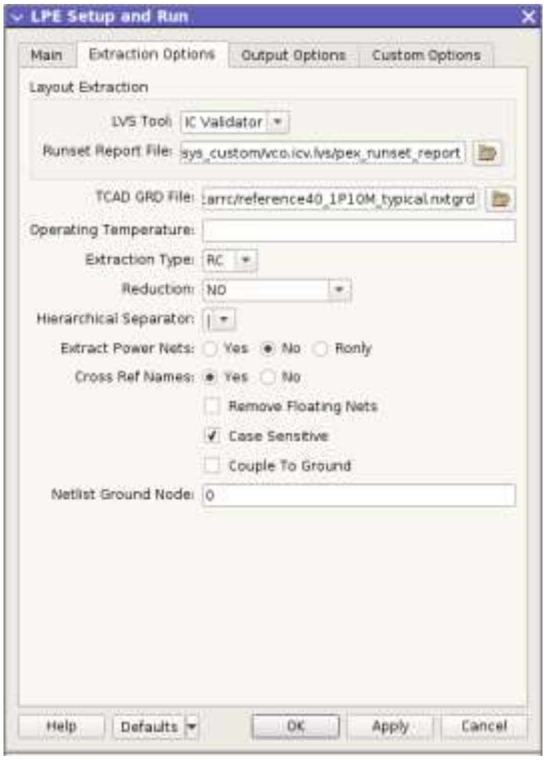
94. Notice that the ***Library***, ***Cell***, and ***View*** fields in the ***Layout Section*** of the ***Main Tab*** are prefilled with the active cellview.

95. Click ***LMB*** to select the ***Extraction Options Tab***. Open the star command file from ***../PDK/starrc/star_icv_cmd***, and notice that the options from the command file were parsed by the PVE environment and set in the dialog as in the following figures.



```
TCAD_GRD_FILE: ./reference40_1P10M_typical.nxtgrd

COUPLE_TO_GROUND: NO
COUPLING_MULTIPLIER: 1
EXTRACTION: RC
CASE_SENSITIVE: YES
NETLIST_NODE_SECTION: YES
NETLIST_CONNECT_SECTION: YES
NETLIST_SUBCKT: YES
NETLIST_PASSIVE_PARAMS: YES
NETLIST_TAIL_COMMENTS: YES
NETLIST_DELIMITER: :
REDUCTION: NO
XREF: YES
EXTRACT_VIA_CAPS: NO
IGNORE_CAPACITANCE: ALL
KEEP_VIA_NODES: NO
MAGNIFY_DEVICE_PARAMS: YES
METAL_FILL_POLYGON_HANDLING: IGNORE
MODE: 200
MOS_GATE_DELTA_RESISTANCE: NO
REMOVE_DANGLING_NETS: NO
```



LPE Setup and Run

Main | Extraction Options | Output Options | Custom Options

Layout Extraction

LVS Tool: IC Validator

Runset Report File: sys_custom/vco.icv.lvs/pex_runset_report

TCAD GRD File: :arrc/reference40_1P10M_typical.nxtgrd

Operating Temperature:

Extraction Type: RC

Reduction: NO

Hierarchical Separator: |

Extract Power Nets: ○ Yes ● No ○ Ronly

Cross Ref Names: ● Yes ○ No

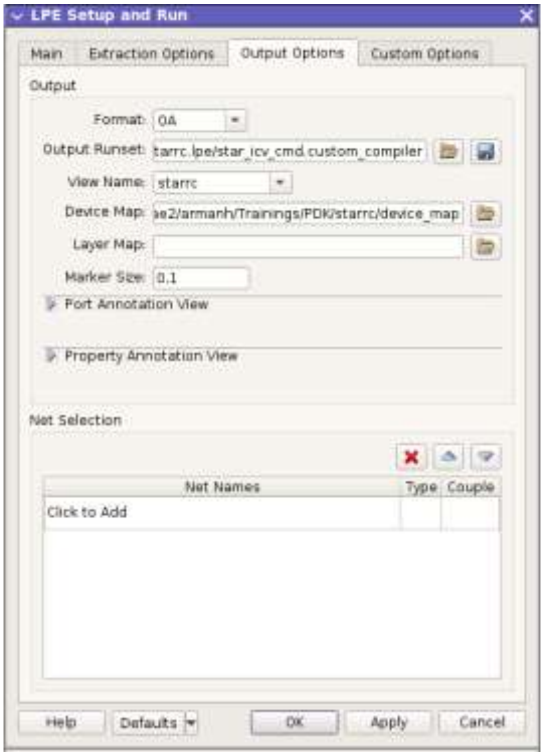
□ Remove Floating Nets

☑ Case Sensitive

□ Couple To Ground

Netlist Ground Node: 0

96. Click ***LMB*** to select the ***Output Options Tab*** and set the as in the following figure. Notice that these dialog fields are also prefilled from the ***Star Command File***, similar to the ***Extraction Options***.



Format: OA

Output Runset: tarrc.lpe/star_icv_cmd.custom_compiler

View Name: starrc

Device Map: ie2/armanh/Trainings/PDK/starrc/device_map

Layer Map:

Marker Size: 0.1

Net Names	Type	Couple
Click to Add		

97. The ***Net Selection*** section can be used to tell Star RC to only consider certain nets in the extraction output. We will leave this empty for the time being.

98. Click ***OK/Apply*** to start the run.

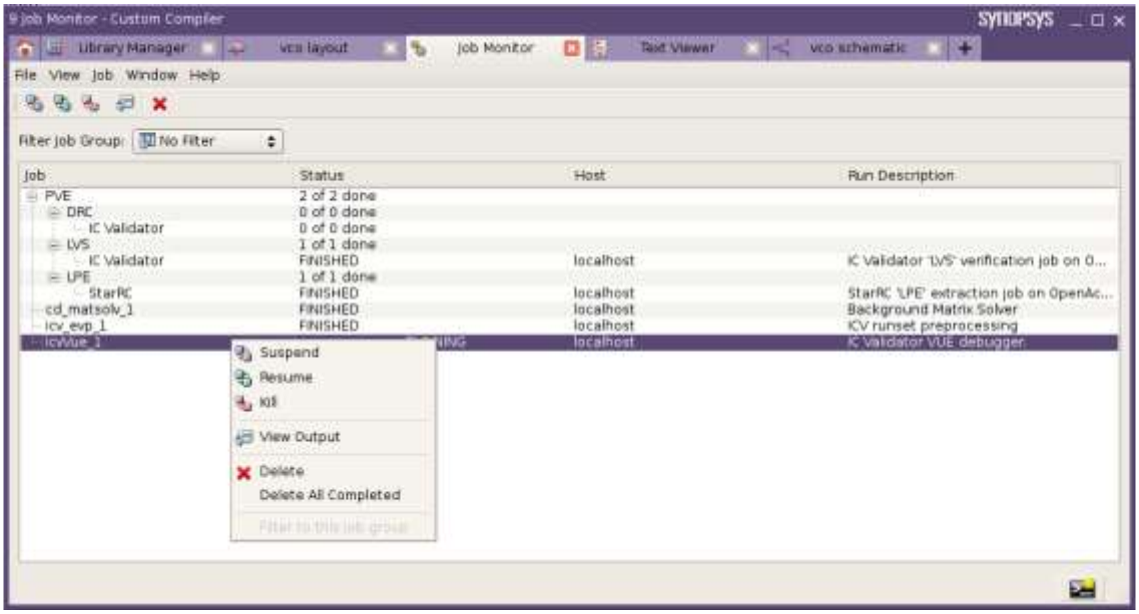
Task 33. *StarRC* LPE Run Observations

In this task we will observe the output of the ***StarRC*** LPE run during and after it completes.

99. As the options were set up in the previous task for the first time, pressing ***OK/Apply*** starts the ***StarRC*** LPE job and stores the job setup in the run directory.

Note: The default run directory name is ***<cwd>/synopsys_custom/<cellName>.starrc.lpe***. If the run directory does not exist, it will be created and the job setup information will be stored in ***<lpeRunDir>/prefs.xml*** when the job is executed. This allows the run settings to be stored and re-used on future runs for the same cellview.

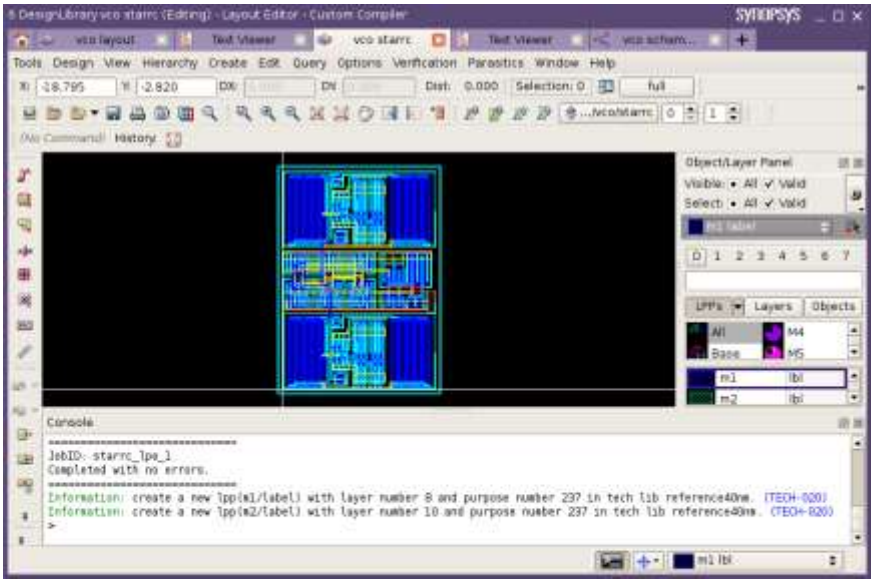
100. Invoke the job monitor using ***Console >Tools > Job Monitor***. Find this job in the ***Job Monitor Window*** and right mouse button click ***(RMB)*** to see the context sensitive menu ***(CSM).*** You can use the ***CSM*** to control the job (suspend, kill, view output, etc). Select V***iew Output*** to display the job's log files in the ***Custom Compiler Text Viewer Window***.



Job	Status	Host	Run Description
PVE	2 of 2 done		
DRC	0 of 0 done		
IC Validator	0 of 0 done		
LVS	1 of 1 done		
IC Validator	FINISHED	localhost	IC Validator 'LVS' verification job on O...
LPE	1 of 1 done		
StarRC	FINISHED	localhost	StarRC 'LPE' extraction job on OpenAc...
cd_matsolv_1	FINISHED	localhost	Background Matrix Solver
icv_evp_1	FINISHED	localhost	ICV runset preprocessing
icvVue_1	...NING	localhost	IC Validator VUE debugger

Note: A job object gets created when launching jobs from the ***PVE*** interface. The job can be monitored using the ***Job Monitor Window***. The ***Job Monitor*** displays all jobs (batch and interactive) that were launched using ***Custom Compiler***'s xtJob infrastructure. You can ***RMB*** on the job to execute other job control actions from the ***CSM***.

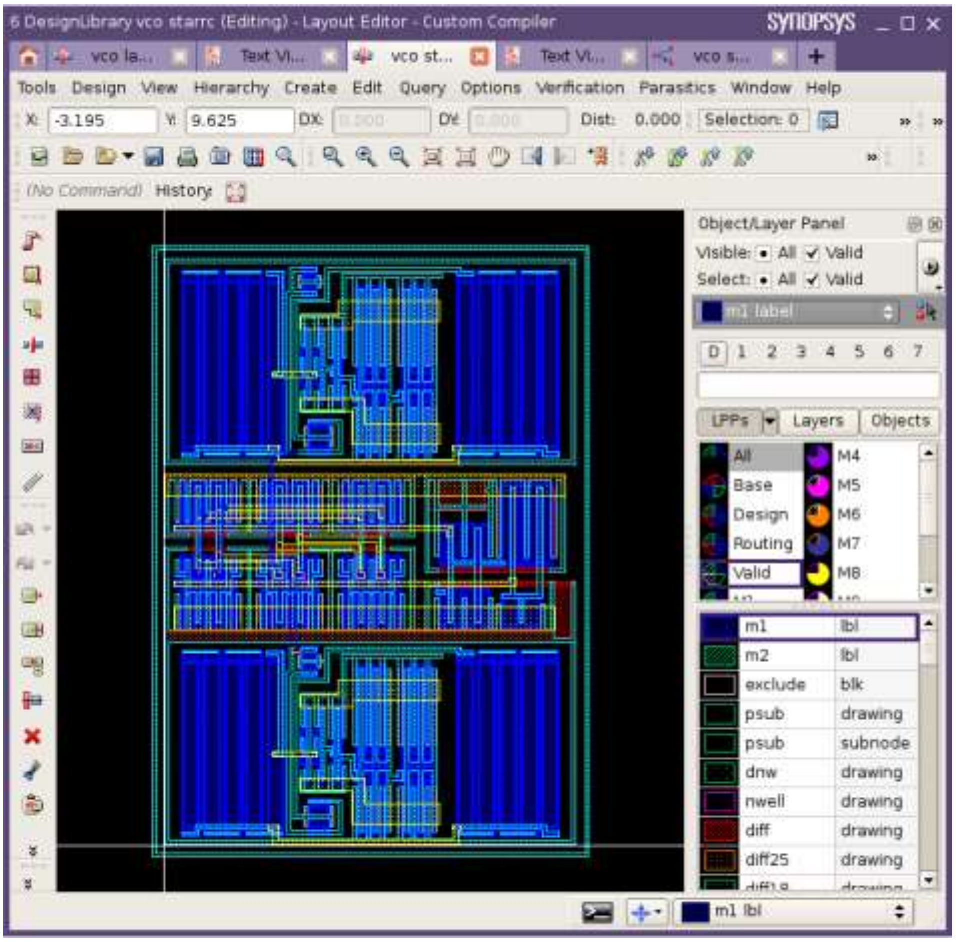
101. The ***Console Assistant*** at the bottom of the ***LE Window*** will print a job summary and status upon job completion as in the following figure.



```
==============================
JobID: starrc_lpe_1
Completed with no errors.
==============================
Information: create a new lpp(m1/label) with layer number 8 and purpose number 237 in tech lib reference40nm. (TECH-020)
Information: create a new lpp(m2/label) with layer number 10 and purpose number 237 in tech lib reference40nm. (TECH-020)
>
```

102. Observe that the run directory ***./synopsys_custom/vco.starrc.lpe*** was created in ***synopsys_custom*** upon completion of the LPE run.

103. Upon completion of the run, the extracted cellview ***DesignLibrary/vco/starrc*** is automatically opened in the ***LE Window*** as in the following figure.



6 DesignLibrary vco starrc (Editing) - Layout Editor - Custom Compiler

X: -3.195 Y: 9.625 DX: 0.000 DY: 0.000 Dist: 0.000 Selection: 0

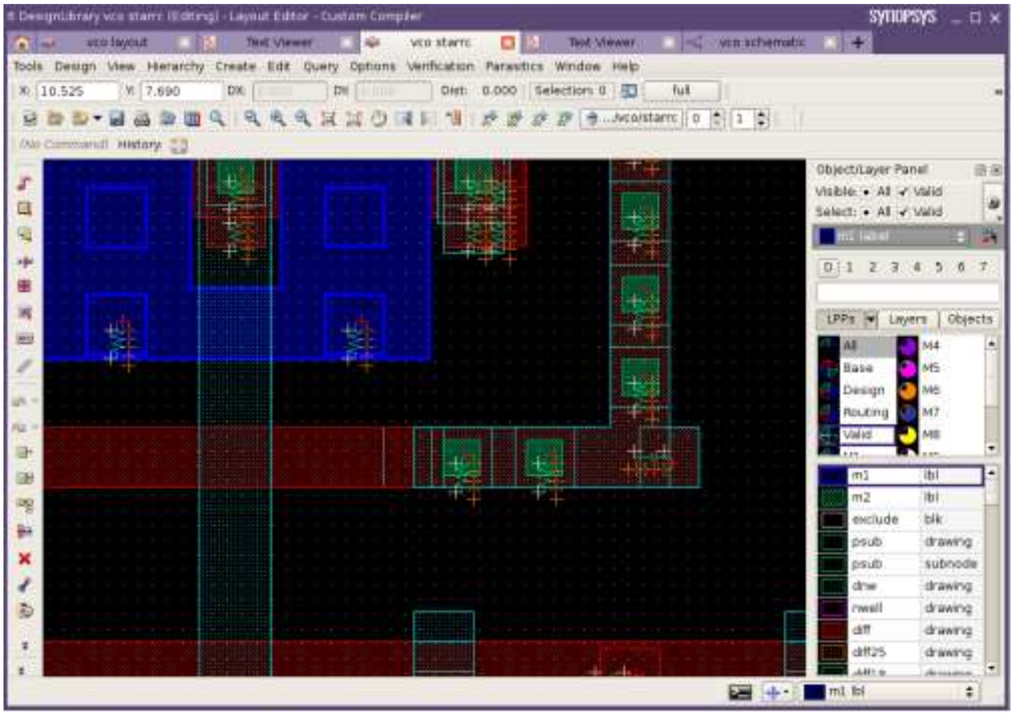
(No Command) History

Note: The extracted view being automatically opened after the run is controlled by the ***Open Parasitic View*** field on the ***Main Tab*** of the ***LPE Setup Dialog***.

Task 34. View the Extracted Cellview

In this task, we will explore the parasitic view generated in ***Task 4***.

104. Make sure that the starrc view is the active ***LE Window*** in your ***Custom Compiler*** session. Click and drag ***RMB*** more and more until you can see individual parasitic R and C elements as in the following figure.



105. Observe that there are symbols of ideal resistors and capacitors representing the parasitic elements.

Note: The values of extracted parasitic instances are displayed on the canvas. You can also query them (**Q bindkey)** to get information in the ***Property Editor Assistant (PE)***.

106. Zoom and pan until you find a transistor instance. Hit the ***Q*** bindkey to query device parameters in the ***PE***.

Note: The parasitic and device instances are references of the master cellviews ***ivpcell***. The ivpcell views are pcells that can be scaled/sized using the ***mag*** parameter.

Note: The device masters used in the ***starrc*** view are controlled by the device_mapping file ***../PDK/starrc/device_map***, which can be customized by the user.

Task 35. Analyze the *StarRC* Run Directory

In this task, we will analyze the run directory ***vco.starrc.lpe*** that was created by ***StarRC*** in the previous tasks.

107. In the Unix terminal, change your working directory to ***./synopsys_custom/vco.starrc.lpe***.

108. Use the Unix command **ls** to list the contents of the directory.

```
Unix% cd ./synopsys_custom/vco.starrc.lpe
Unix% ls
```

```
Unix% lib.defs run_starrc.sh
star_icv_cmd.custom_compiler

vco.star_sum  prefs.xml  star  stdout.lpe.log
```

109. Open the ***prefs.xml*** file and note its contents.

Note: The ***prefs.xml*** file stores the setup information of the last ***StarRC*** run.

110. Open the file ***star_icv_cmd.cdesigner***. This file was generated by the ***Custom Compiler PVE*** environment based on the original ***star_icv_cmd*** file, and the custom options that the user set in the dialog. This is the file that ***StarRC*** will read in.

Note: For future runs, these settings will be preset when the ***LPE Setup Dialog*** is opened or when***Verification→LPE→Run*** is invoked.

111. Observe that, output files containing run details of the ***StarRC*** LPE run are stored in the ***star*** directory.

Task 36. Query Parasitics (Net)

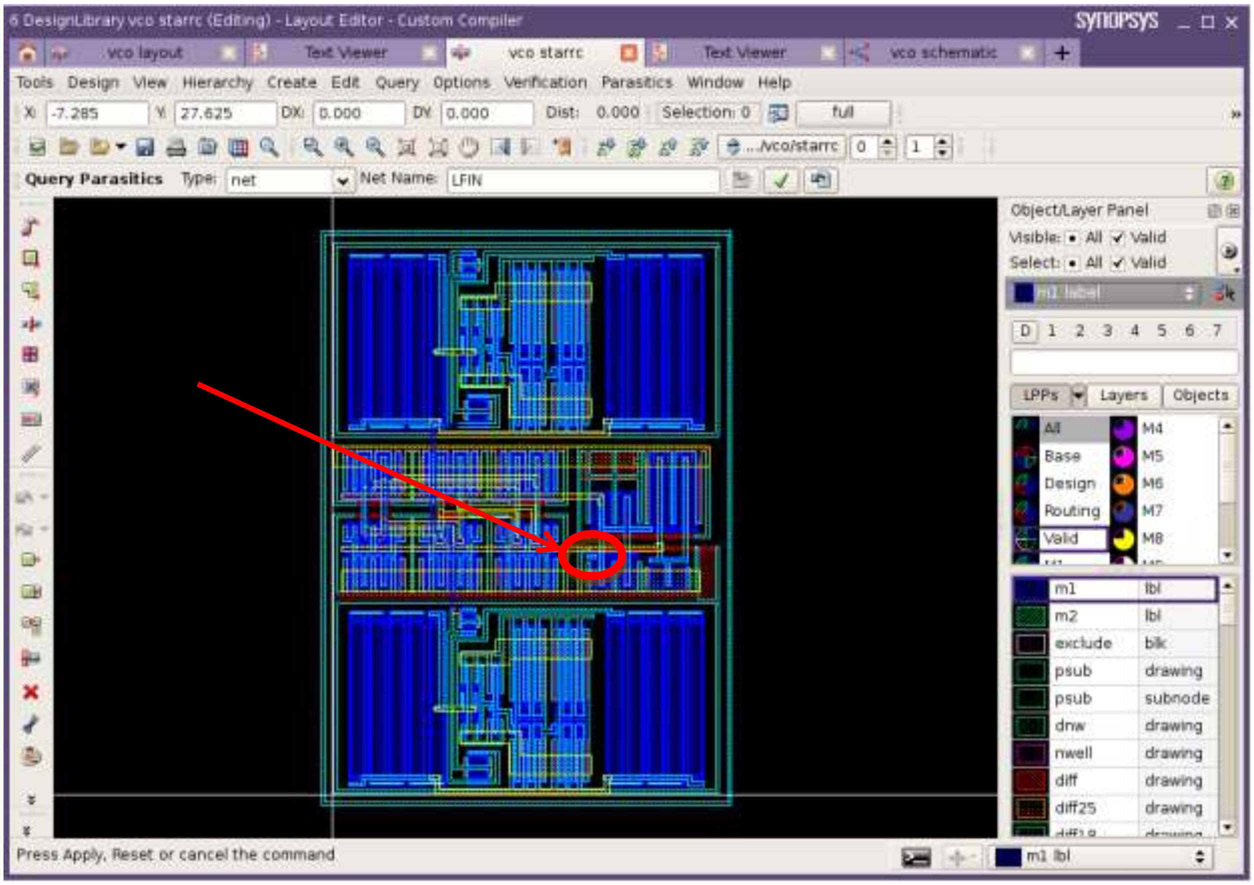
In this task, we will use the query parasitics feature to query a net.

112. Make ***DesignLibrary/vco/starrc*** the active cellview.

113. Fit the design (***F*** bindkey). If the ***Parasitics*** menu is not shown in the ***LE Toolbar***, then enable it by invoking ***Tools*** > ***Parasitics***.

114. Invoke ***Parasitics*** > ***Query.***

115. In the ***Command Options Toolbar (COT)***, leave the ***Type*** field set to ***Net***. Click ***LMB*** to select the net ***LFIN*** in the ***starrc*** view as in the following figure. Alternatively, you can type ***LFIN*** into the ***netName*** field.

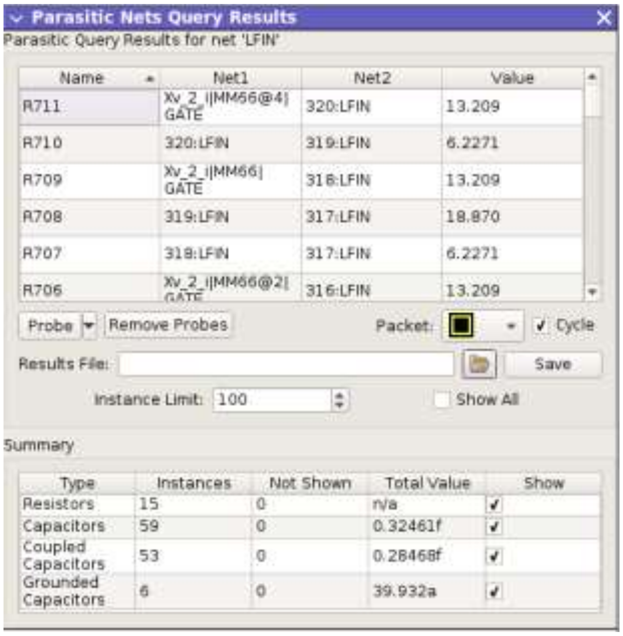

6 DesignLibrary vco starrc (Editing) - Layout Editor - Custom Compiler

X: -7.285 Y: 27.625 DX: 0.000 DY: 0.000 Dist: 0.000 Selection: 0 full

Query Parasitics Type: net Net Name: LFIN

Press Apply, Reset or cancel the command

116. Click the green arrow icon in the ***COT*** to perform the query of net ***LFIN***. The ***Query Results Dialog*** should open as in the following figure.


Parasitic Nets Query Results

Parasitic Query Results for net 'LFIN'

Name	Net1	Net2	Value
R711	Xv_2_i|MM66@4|GATE	320:LFIN	13.209
R710	320:LFIN	319:LFIN	6.2271
R709	Xv_2_i|MM66|GATE	318:LFIN	13.209
R708	319:LFIN	317:LFIN	18.870
R707	318:LFIN	317:LFIN	6.2271
R706	Xv_2_i|MM66@2|GATE	316:LFIN	13.209

Probe ▾ Remove Probes Packet: ■ ▾ ☑ Cycle

Results File: [ ] Save

Instance Limit: 100 □ Show All

Summary

Type	Instances	Not Shown	Total Value	Show
Resistors	15	0	n/a	☑
Capacitors	59	0	0.32461f	☑
Coupled Capacitors	53	0	0.28468f	☑
Grounded Capacitors	6	0	39.932a	☑

117. The following observations can be made from the query dialog:

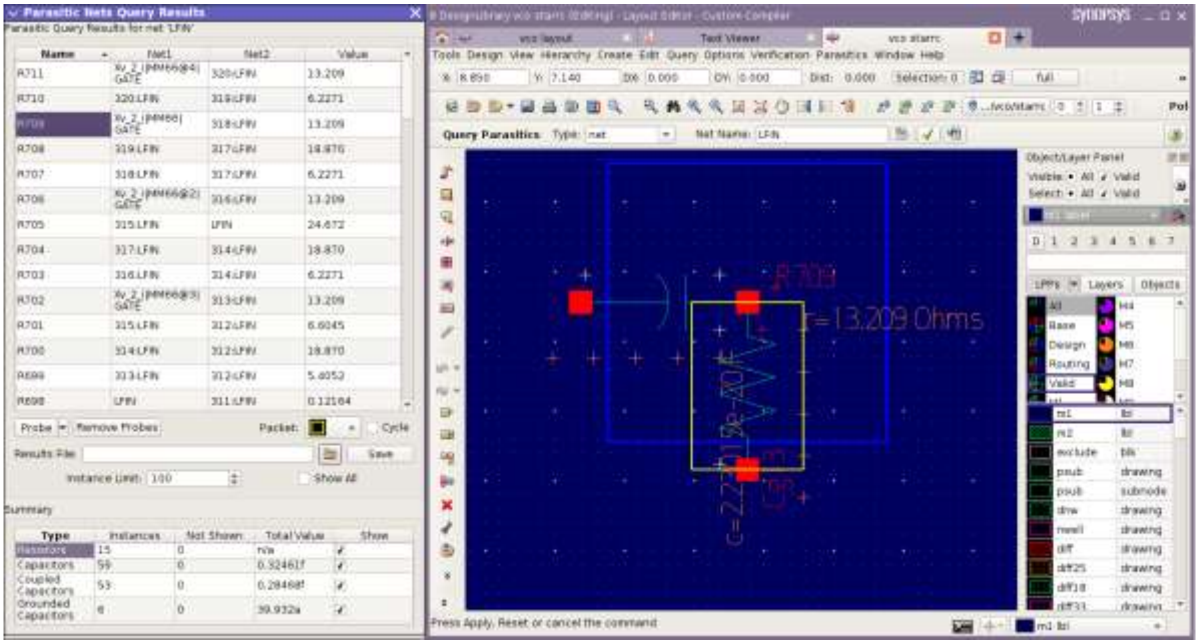
a. The ***Query Results Table*** displays all parasitics on the net being queried.

b. The ***Summary Table*** contains summaries and a total value for each type of parasitic device in the results table.

118. Probe parasitic elements from the ***Query Results Table*** dialog:

a. Select a device listed under ***Name*** column. Click ***RMB,*** to invoke the ***CSM***, and select ***Probe Selected Names*** from the menu.

b. Observe ***LE*** zooms to the selected parasitic instance in the ***LE Window*** and highlights it.



Parasitic Nets Query Results

Parasitic Query Results for net 'LFIN'

Name	Net1	Net2	Value
R711	Xv_2_(MM66@4)|GATE	320:LFIN	13.209
R710	320:LFIN	319:LFIN	6.2271
R709	Xv_2_(MM66)|GATE	318:LFIN	13.209
R708	319:LFIN	317:LFIN	18.870
R707	318:LFIN	317:LFIN	6.2271
R706	Xv_2_(MM66@2)|GATE	316:LFIN	13.209
R705	315:LFIN	LFIN	24.672
R704	317:LFIN	314:LFIN	18.870
R703	316:LFIN	314:LFIN	6.2271
R702	Xv_2_(MM66@3)|GATE	313:LFIN	13.209
R701	315:LFIN	312:LFIN	6.6045
R700	314:LFIN	312:LFIN	18.870
R699	313:LFIN	312:LFIN	5.4052
R698	LFIN	311:LFIN	0.12164

Instance Limit: 100

Summary

Type	Instances	Not Shown	Total Value	Show
Resistors	15	0	n/a	✓
Capacitors	59	0	0.32461f	✓
Coupled Capacitors	53	0	0.28468f	✓
Grounded Capacitors	6	0	39.932a	✓

Query Parasitics Type: net Net Name: LFIN

Press Apply, Reset or cancel the command

Note: Using ***Probes*** can make it easier to visualize and locate parasitic elements in the parasitic view.

c. Clear the probes by clicking the ***Remove Probes*** button in the ***Query Results Dialog***.

119. Close the ***Query Results Dialog***.

120. Hit the ***Esc*** key to exit the ***Query Parasitics Command***.

Task 37. Query Parasitics (Net to Net)

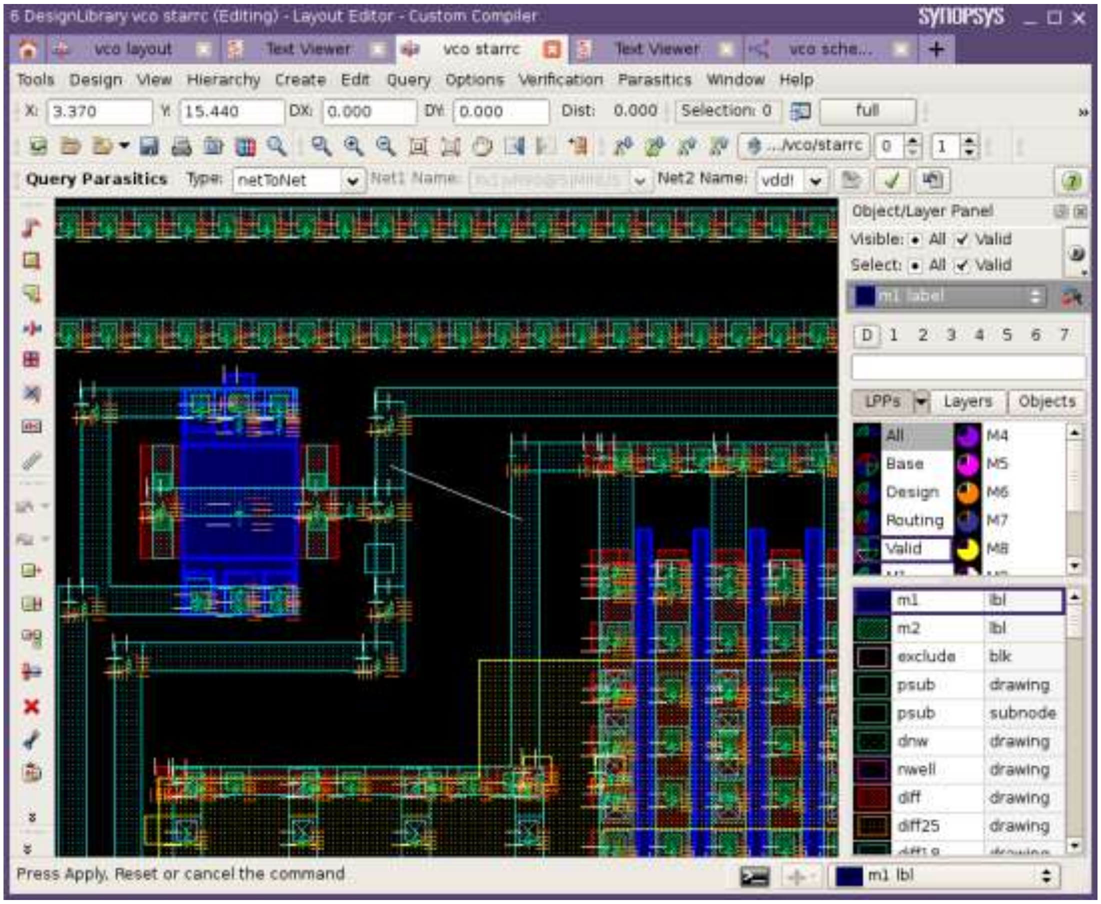
In this task, we will use the query parasitics feature to query coupling capacitance between nets.

121. Make ***DesignLibrary/vco/starrc*** the active cellview and fit the design (***F*** bindkey).

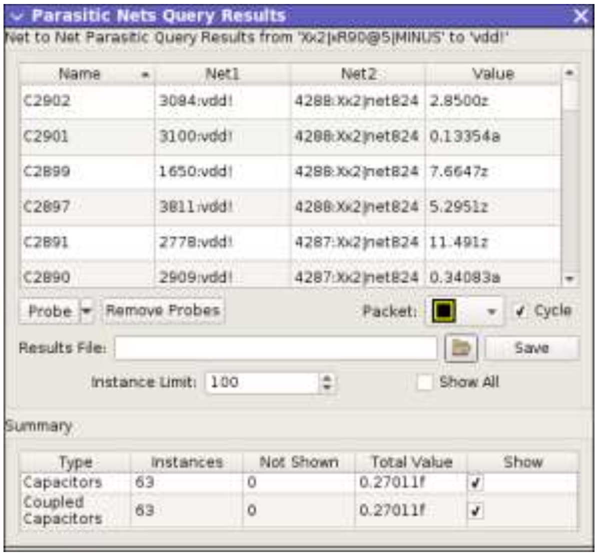
122. Invoke ***View > Pan*** (or ***3*** bindkey). In the ***Pan COT***, type in the coordinates {***4.8 16.4***}, enable ***Zoom*** and set the zoom factor to ***7.0***, and hit the ***Enter*** key to zoom around the area as in the following figure.

123. Invoke ***Parasitics > Query*** and set the ***Type*** field in the ***COT*** to ***netToNet***.

124. Select the 2 nets to query by clicking ***LMB*** anywhere on the 2 nets as in the 2 points shown in the following figure.



125. Click the green arrow icon in the ***COT*** to perform the net to net query. The ***Query Results Dialog*** should open as in the following figure.


Parasitic Nets Query Results

Net to Net Parasitic Query Results from 'Xx2|xR90@5|MINUS' to 'vdd!'

Name	Net1	Net2	Value
C2902	3084:vdd!	4288:Xx2|net824	2.8500z
C2901	3100:vdd!	4288:Xx2|net824	0.13354a
C2899	1650:vdd!	4288:Xx2|net824	7.6647z
C2897	3811:vdd!	4288:Xx2|net824	5.2951z
C2891	2778:vdd!	4287:Xx2|net824	11.491z
C2890	2909:vdd!	4287:Xx2|net824	0.34083a

Probe | Remove Probes | Packet: | ✓ Cycle

Results File: | Save

Instance Limit: 100 | □ Show All

Summary

Type	Instances	Not Shown	Total Value	Show
Capacitors	63	0	0.27011f	✓
Coupled Capacitors	63	0	0.27011f	✓

126. The following observations can be made from the query dialog:

- **a.** The ***Query Results Table*** lists all coupling capacitors between the 2 nets.
- **b.** The ***Summary Table*** contains summaries and a total value for each type of parasitic device in the results table.

c. Elements from the ***Query Results Table*** can be cross probed to the extracted view in ***LE***, similar to the query net case.

127. Close the ***Query Results Dialog***.

128. Hit the ***Esc*** key to exit the ***Query Parasitics Command***.

Task 38. Query Parasitics (Point to Point)

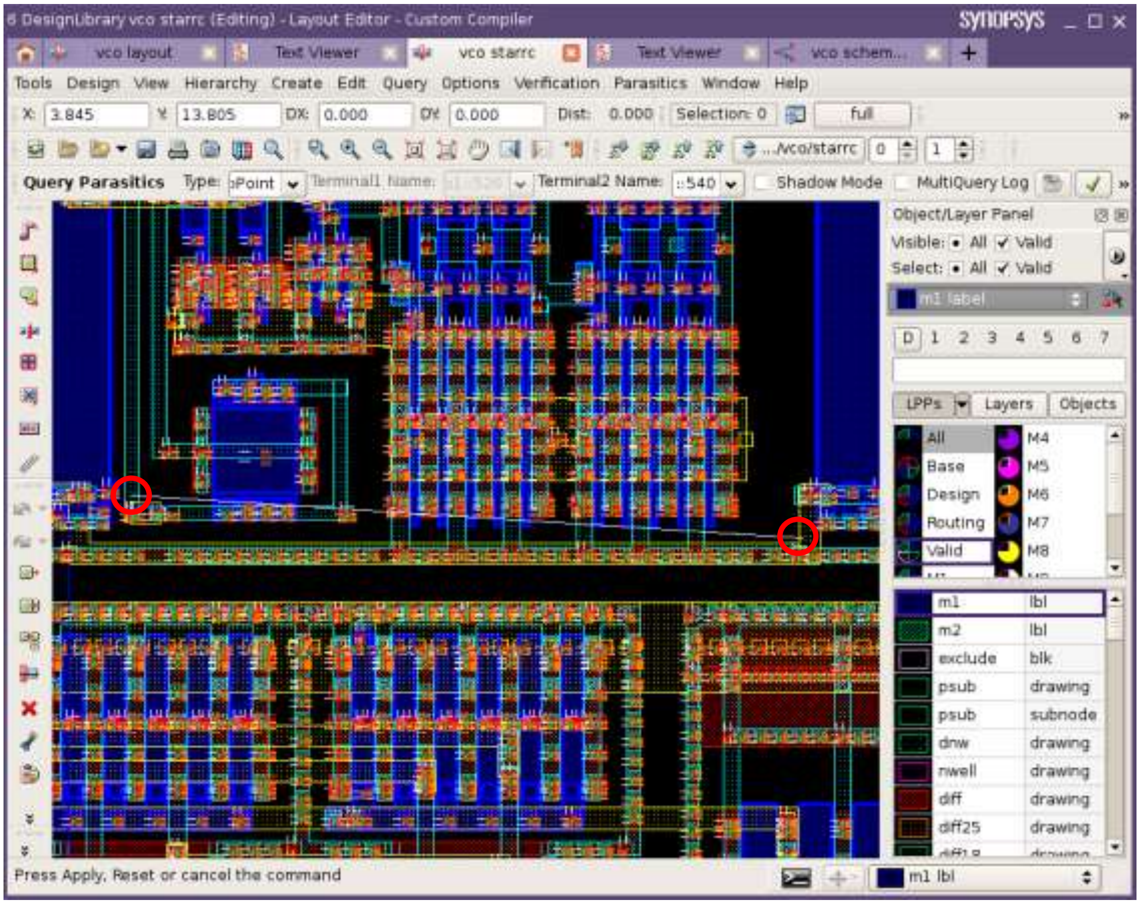
In this task, we will use the query parasitics feature to query the resistance between 2 points on the same net.

129. Make ***DesignLibrary/vco/starrc*** the active cellview and fit the design (***F*** bindkey).

130. Invoke ***View*** > ***Pan*** (or ***3*** bindkey). In the ***Pan COT***, type in the coordinates {***6.0 11.3***}, enable ***Zoom*** and set the zoom factor to ***3.5***, and hit the ***Enter*** key to zoom around the area as in the following figure.

131. Invoke ***Parasitics→Query*** and set the ***Type*** field in the ***COT*** to ***pointToPoint***.

132. Select the 2 points to query by clicking ***LMB*** near the 2 points as in the following figure.



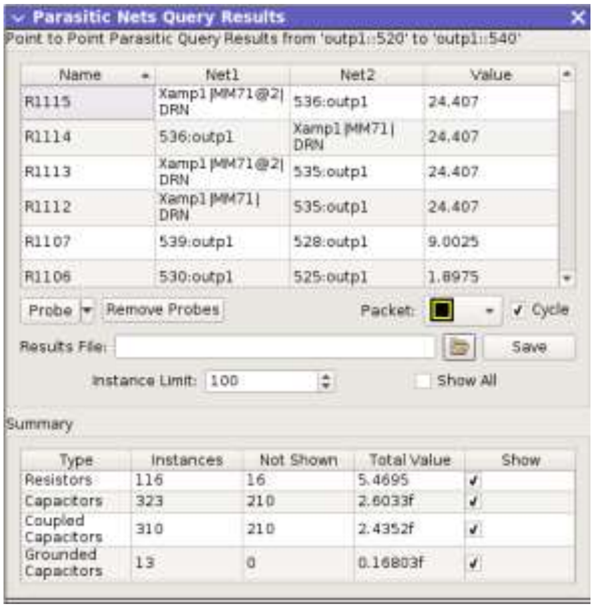
6 DesignLibrary vco starrc (Editing) - Layout Editor - Custom Compiler

X: 3.845 Y: 13.805 DX: 0.000 DY: 0.000 Dist: 0.000 Selection: 0 full

Query Parasitics Type: pPoint Terminal1 Name: [illegible] Terminal2 Name: ::540 □ Shadow Mode □ MultiQuery Log

Press Apply, Reset or cancel the command

133. Click the green arrow icon in the ***COT*** to perform the point to point query. The ***Query Results Dialog*** should open as in the following figure.



Parasitic Nets Query Results

Point to Point Parasitic Query Results from 'outp1::520' to 'outp1::540'

Name	Net1	Net2	Value
R1115	Xamp1 |MM71@2| DRN	536:outp1	24.407
R1114	536:outp1	Xamp1 |MM71| DRN	24.407
R1113	Xamp1 |MM71@2| DRN	535:outp1	24.407
R1112	Xamp1 |MM71| DRN	535:outp1	24.407
R1107	539:outp1	528:outp1	9.0025
R1106	530:outp1	525:outp1	1.8975

Packet: ■ ✓ Cycle

Results File:

Instance Limit: 100 □ Show All

Summary

Type	Instances	Not Shown	Total Value	Show
Resistors	116	16	5.4695	✓
Capacitors	323	210	2.6033f	✓
Coupled Capacitors	310	210	2.4352f	✓
Grounded Capacitors	13	0	0.16803f	✓

134. The following observations can be made from the query dialog:

a. The ***Query Results Table*** lists all resistors that occur between the 2 points on the net and all capacitors that couple to other nets from points between the 2 probed points.

b. The ***Summary Table*** contains summaries and a total value for each type of parasitic device in the results table. The total resistance is not simply a sum of all resistors, it is in fact a highly accurate total R between the 2 points computed by a numerical matrix solver.

c. Elements from the ***Query Results Table*** can be cross probed to the extracted view in ***LE***, similar to the query net case.

135. Close the ***Query Results Dialog***.

136. Hit the ***Esc*** key to exit the ***Query Parasitics Command***.

Task 39. Cross Probing Between Views

In this task, we will cross probe nets and instances from the schematic view to their equivalents in the extracted view.

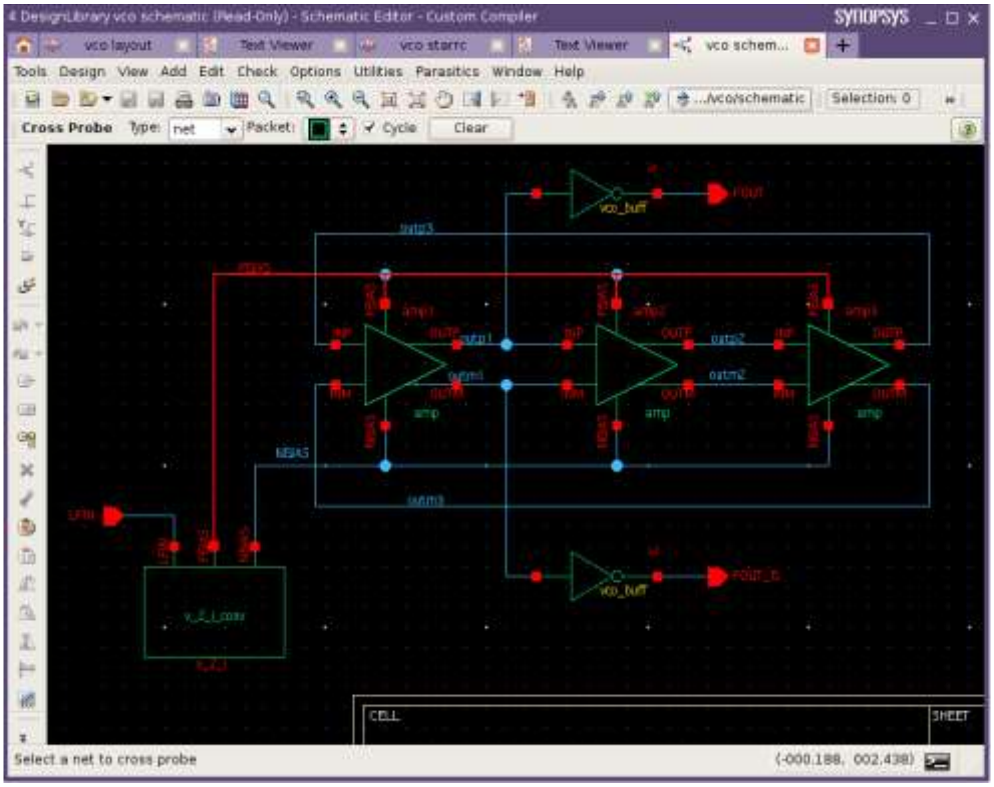
137. Make ***DesignLibrary/vco/starrc*** the active cellview and fit the design (***F*** bindkey).

138. Invoke **Parasitics > Initialize**. Click LMB ***OK*** to associate the ***starrc*** view to the ***schematic*** view and open the schematic view in the ***Custom Compiler Schematic Editor (SE)***.

139. In the ***SE Window***, invoke ***Tools > Parasitics***, then invoke ***Parasitics > CrossProbe***.

140. In the ***Cross Probe COT***, leave the ***Type*** field as ***Net***, disable ***Cycle*** option, and set the probe color ***Packet*** to ***red*** as in the following figure.

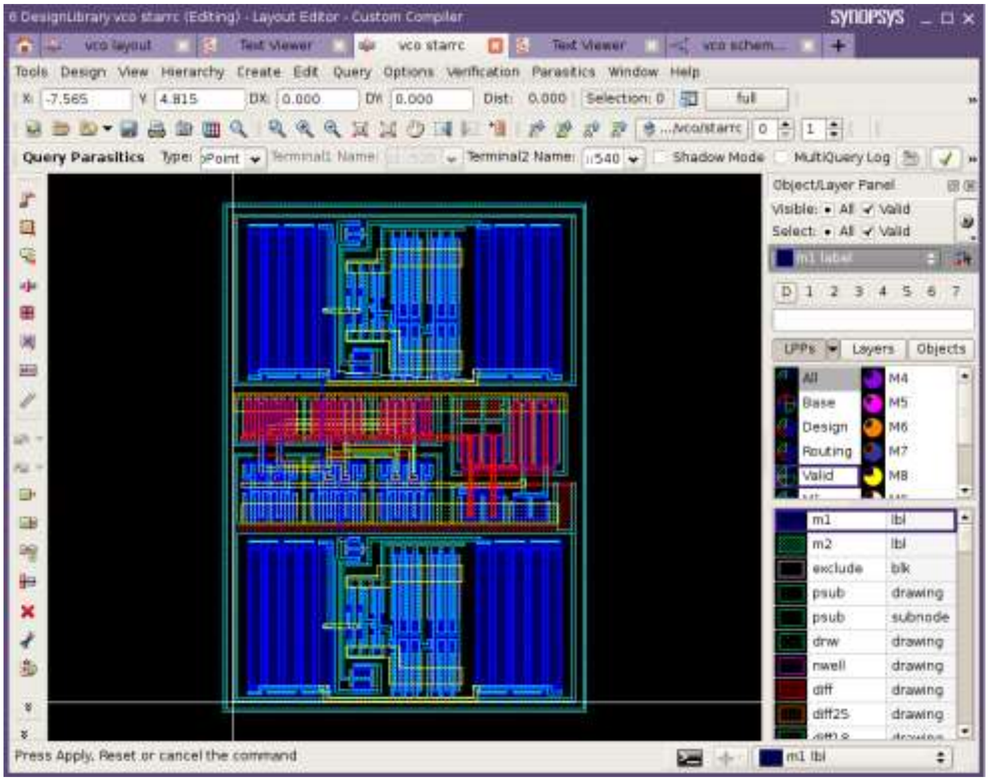
141. Click ***LMB*** in the ***SE Canvas*** to select a wire on the net ***PBIAS*** and notice that the net is probed in SE and colored red, as in the following figure.



Cross Probe Type: net Packet: ☑ Cycle

Select a net to cross probe

142. Click ***LMB*** to open the ***LE Window*** tab in the ***Custom Compiler*** tabbed windows, to raise the ***starrc*** view in the ***LE Window***. Notice that the net probed from the schematic has been cross probed to the starrc view. All subnodes, devices, parasitic elements, and layer objects are probed in red as in the following figure.


6 DesignLibrary vco starrc (Editing) - Layout Editor - Custom Compiler
SYNOPSYS
vco layout
Text Viewer
vco starrc
Text Viewer
vco schem...
Tools Design View Hierarchy Create Edit Query Options Verification Parasitics Window Help
...\vco/starrc
Object/Layer Panel
LPPs
Layers
Objects
All
Base
Design
Routing
Valid
M4
M5
M6
M7
M8

X: -7.565 Y: 4.815 DX: 0.000 DY: 0.000 Dist: 0.000 Selection: 0 full

Query Parasitics Type: Point Terminal1 Name: Terminal2 Name: i1540 □ Shadow Mode □ MultiQuery Log

Visible: ● All ☑ Valid

Select: ● All ☑ Valid

m1 label

m1	lbl
m2	lbl
exclude	blk
psub	drawing
psub	subnode
drw	drawing
nwell	drawing
diff	drawing
diff25	drawing

Press Apply, Reset or cancel the command

m1 lbl

143. If time permits, you can perform more cross probes for nets or devices, from schematic to starrc view, or the other way around.

Task 40. Generate Parasitics in SPF Format

In this task, you will rerun parasitic extraction using Star RC to generate parasitics in SPF format.

144. Close all editor windows. Open the layout view: ***DesignLibrary/vco/layout*** Fit the screen (press the ***F*** bindkey).

145. We will rerun the extraction to generate parasitics in SPF format. The name of the output netlist file can be specified dynamically using the token substitution feature. Supported tokens are %u(userName), %j(jobName), %t(toolName), %l(libName), %c(cellName), %v(viewName).

146. In the ***Custom Compiler Console***, enter the following command to set the preference that controls dynamically naming the output file:

db::setPrefValue xtLPEDefaultOutputNetlistTemplate -value "%l_%c_%v_extracted"

147. In the layout editor menu, invoke ***Verification > LPE > Setup and Run …*** to open the LPE setup dialog again. Notice that the settings from your previous run are preserved. Leave them the same as before.

148. Click the Output Options tab, and select ***Format: SPF***. Notice that the ***Netlist File*** field has been populated by the name: ***DesignLibrary_vco_layout_extracted.SPF***. The tokens (%l, %c, %v) have been evaluated and filled in at runtime based on your template string.

149. Uncheck "**Use LVS Netlist Port Order**". Click ***OK/Apply*** to execute the extraction job.

150. At the end of the run, open the output SPF netlist file to verify that it was correctly created by Star RC.

Congratulations!

You have successfully performed parasitic extraction and explored the parasitic analysis features in ***Custom Compiler Layout Editor***.

5 Design Rule Aware Editing

Learning Objectives

During this lab, you will edit the design according to constraints and design rules.

After completing this lab, you will be able to do the following tasks:

- Constraint Aware Editing
- Use DRD Visual/Assist modes
- Use DRD Auto-Fix engine

Lab Duration:
30 minutes

File Locations

All files for this lab are located in the directory

`LE_Design_Rule_Aware_Editing_Lab1`

Directory Structure

`LE_Design_Rule_Aware_Editing_Lab1`	Current working directory
`DesignLibrary`	Design library
`technology.tf`	ASCII technology file
`.synopsys_cutom.tcl`	Startup file with tool configuration

Instructions

Task 41. Start Custom Compiler

In the UNIX window, change your working directory to `LE_Design_Rule_Aware_Editing_Lab1` and invoke Custom Compiler.

```
unix% cd LE_Design_Rule_Aware_Editing_Lab1
unix% custom_compiler &
```

Task 42. Constraint Driven Layout

In this step, constraint-driven layout will be explored.

151. Invoke **Library Manager** and open cell: **DesignLibrary/top/layout**

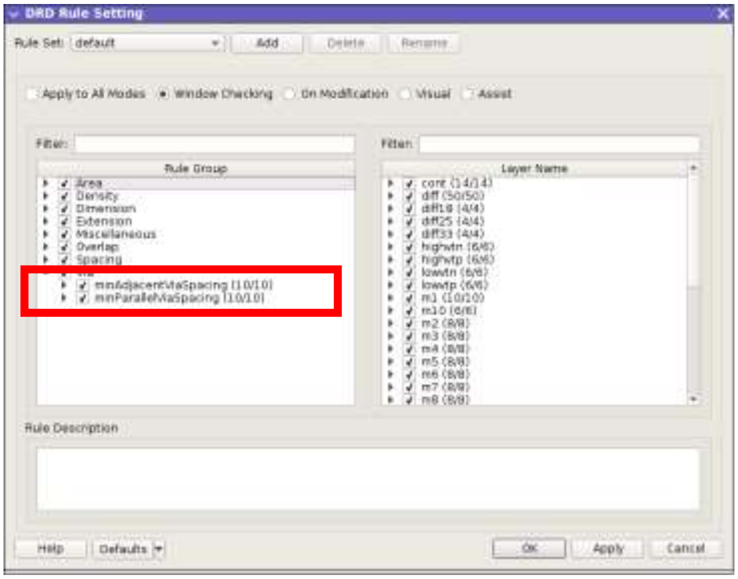
152. Enable **DRD Visual** with **On Modification** flag enabled. **DRD Visual** will mark any errors that occur during editing of the design.

    Note: To enable DRD Visual you have to enable Advanced license from Home → License.

153. Check if **MinAdjustmentViaSpacing** rule enabled in **DRD Rule Selection Dialog** .

154. Press check DRD button ☑ DRD toolbar menu.

155. You will see the power mesh and you would need to clear DRCs on vias.

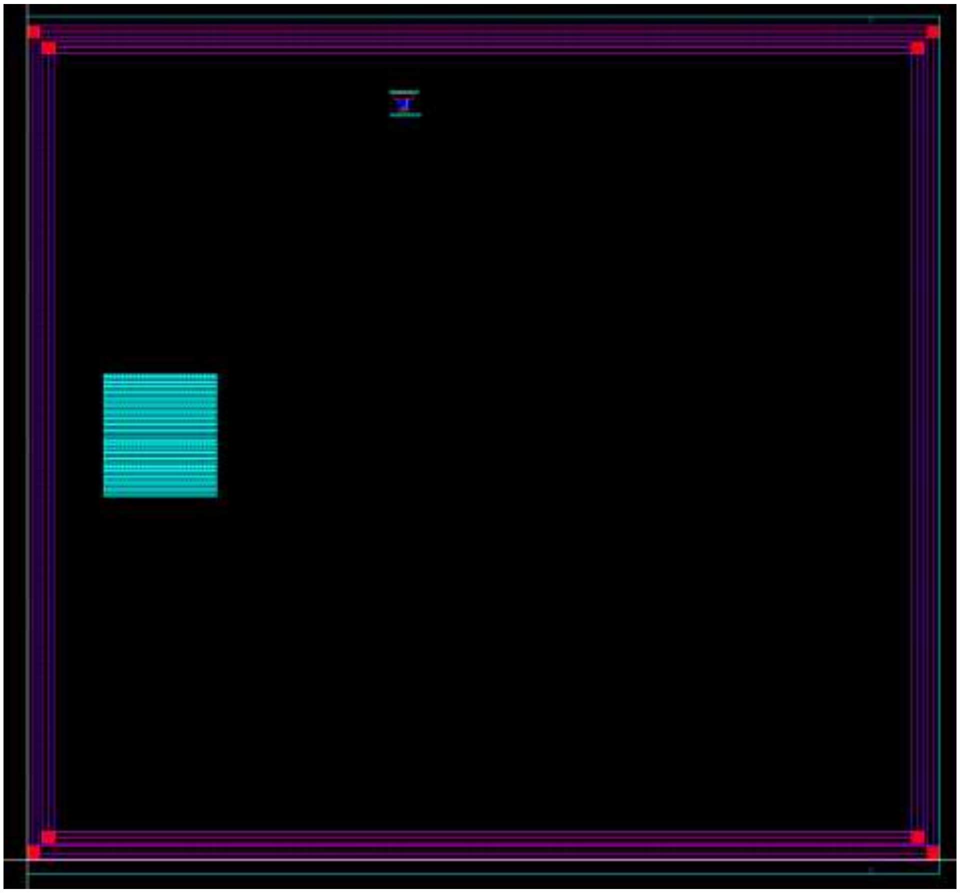


Rule Set: default

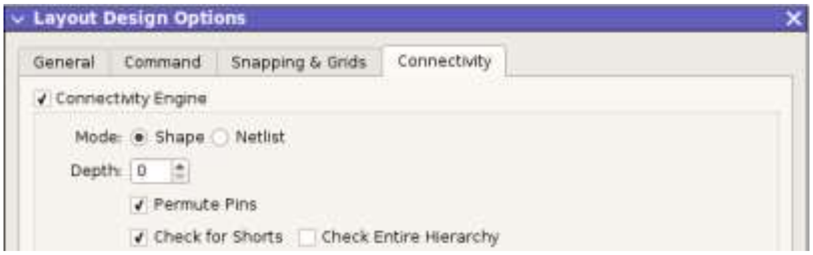
☐ Apply to All Modes ● Window Checking ○ On Modification ○ Visual ☐ Assist

Rule Group:
- ☑ Area
- ☑ Density
- ☑ Dimension
- ☑ Extension
- ☑ Miscellaneous
- ☑ Overlap
- ☑ Spacing
- Via
  - ☑ minAdjacentViaSpacing (10/10)
  - ☑ minParallelViaSpacing (10/10)

Layer Name:
- ☑ cont (14/14)
- ☑ diff (50/50)
- ☑ diff18 (4/4)
- ☑ diff25 (4/4)
- ☑ diff33 (4/4)
- ☑ highwtn (6/6)
- ☑ highwtp (6/6)
- ☑ lowwtn (6/6)
- ☑ lowwtp (6/6)
- ☑ m1 (10/10)
- ☑ m10 (6/6)
- ☑ m2 (8/8)
- ☑ m3 (8/8)
- ☑ m4 (8/8)
- ☑ m5 (8/8)
- ☑ m6 (8/8)
- ☑ m7 (8/8)
- ☑ m8 (8/8)



156. Turn on the **Connectivity Engine**



Layout Design Options

☑ Connectivity Engine

Mode: ● Shape ○ Netlist

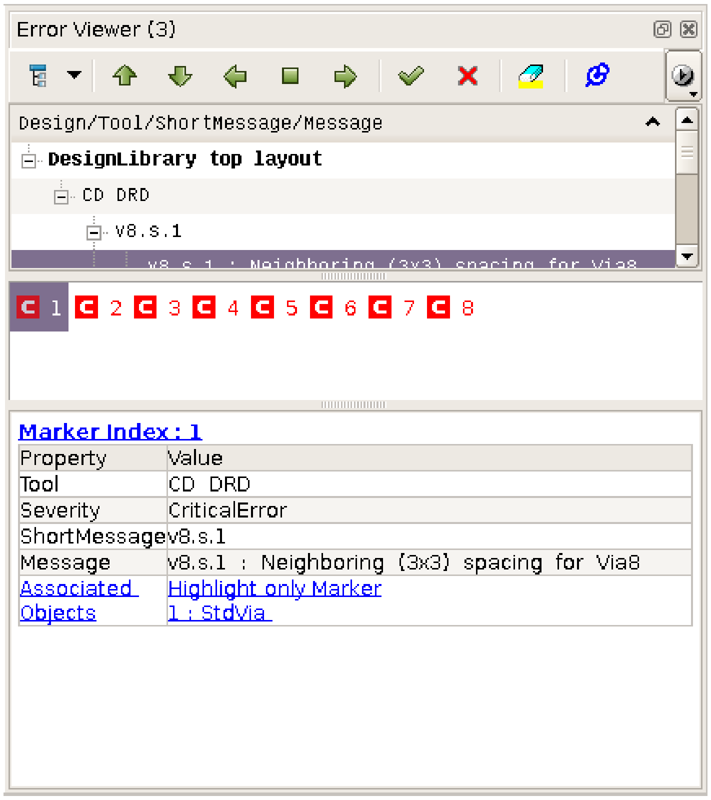
Depth: 0

☑ Permute Pins

☑ Check for Shorts □ Check Entire Hierarchy

157. There are lots of errors on placed vias because of the following rule: minAdjacentViaSpacing = 0.11 if three or more neighboring vias as present.

Hint: Open the **Error Viewer** to view the DRC errors

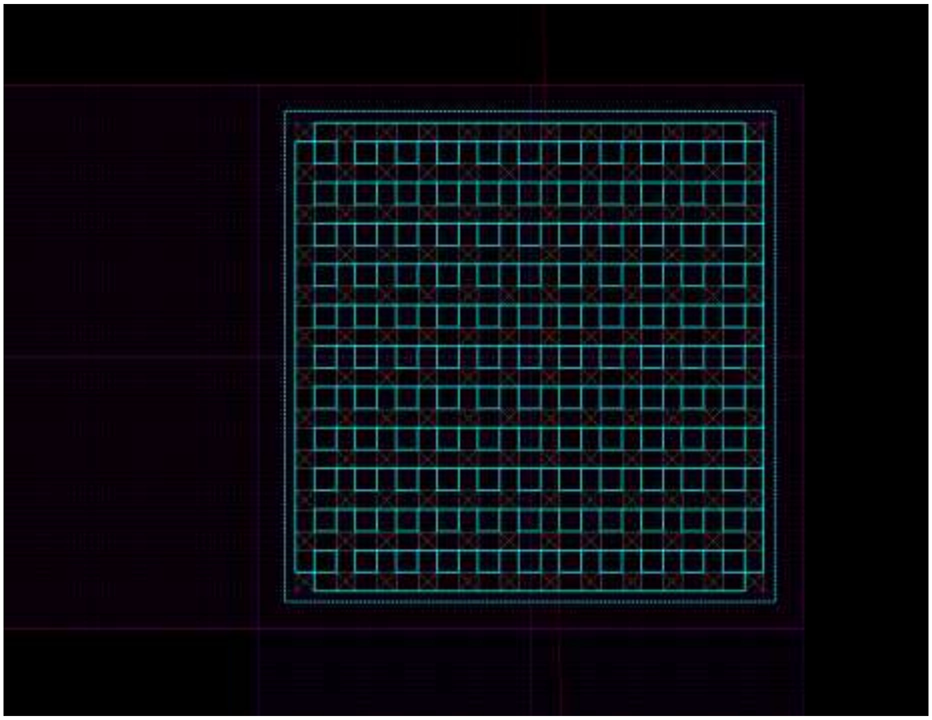


Error Viewer (3)

```
Design/Tool/ShortMessage/Message
DesignLibrary top layout
  CD DRD
    v8.s.1
      v8.s.1 : Neighboring (3x3) spacing for Via8
```

Marker Index : 1

Property	Value
Tool	CD DRD
Severity	CriticalError
ShortMessage	v8.s.1
Message	v8.s.1 : Neighboring (3x3) spacing for Via8
Associated Objects	Highlight only Marker 1 : StdVia

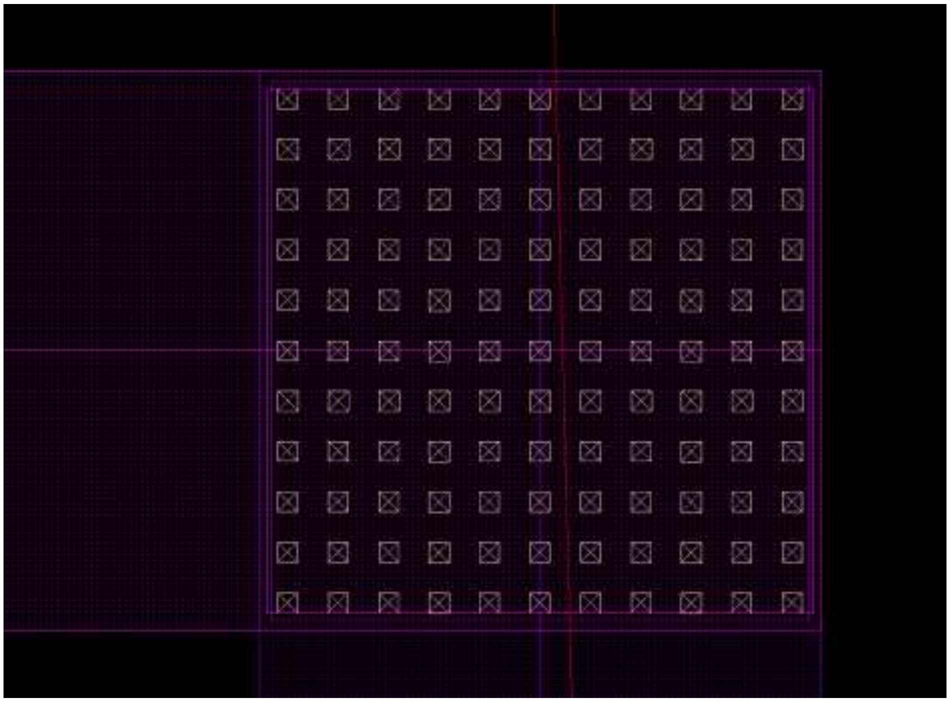


158. Two ways to correct these DRC:

- **a.** Invoke Property editor and change the VIA cut spacing. This operation is time consuming
- **b.** Use the Constraint Driven Layout concept that will use the Via Parameter Engine to create DRC clean Via Array.

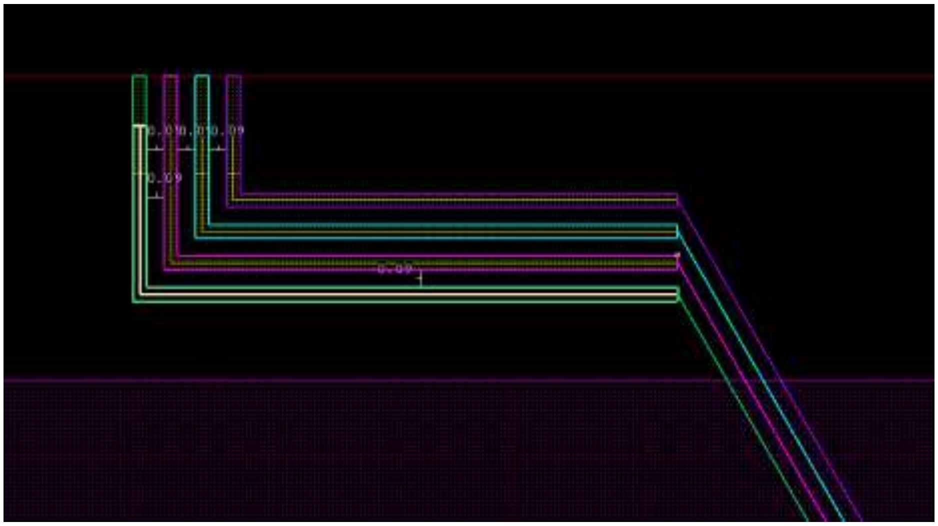
159. Clear highlighting from Error Viewer. Remove the placed vias

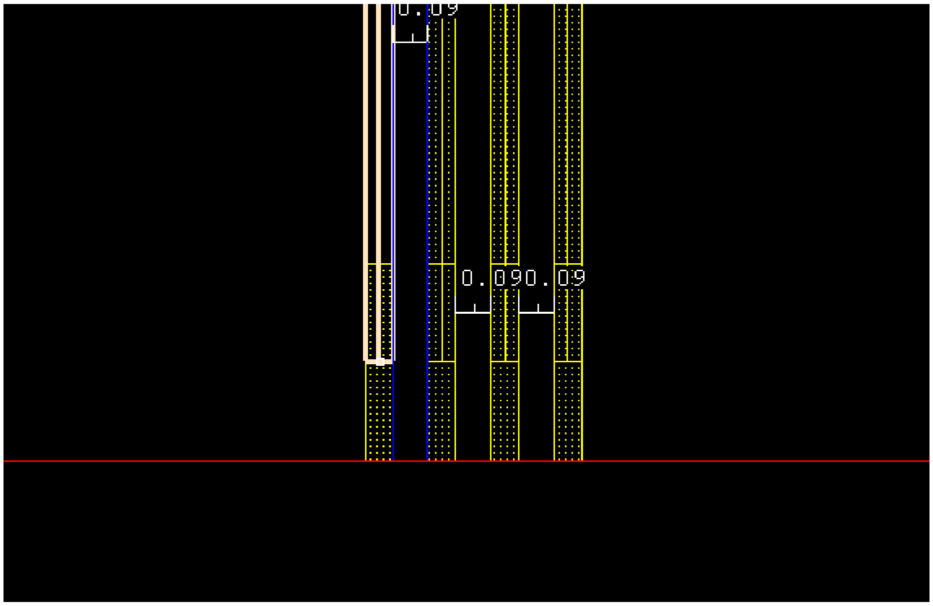
160. Now by the Create Auto Via command (**Create → Via**) you can create the new vias on these nets. Cut spacing is now set to 0.11 instead of the default value of 0.08. Do this for all corner vias.



161. **Create Bus** command will obey width-dependent spacing rules if the Constraint Aware flag is enabled.

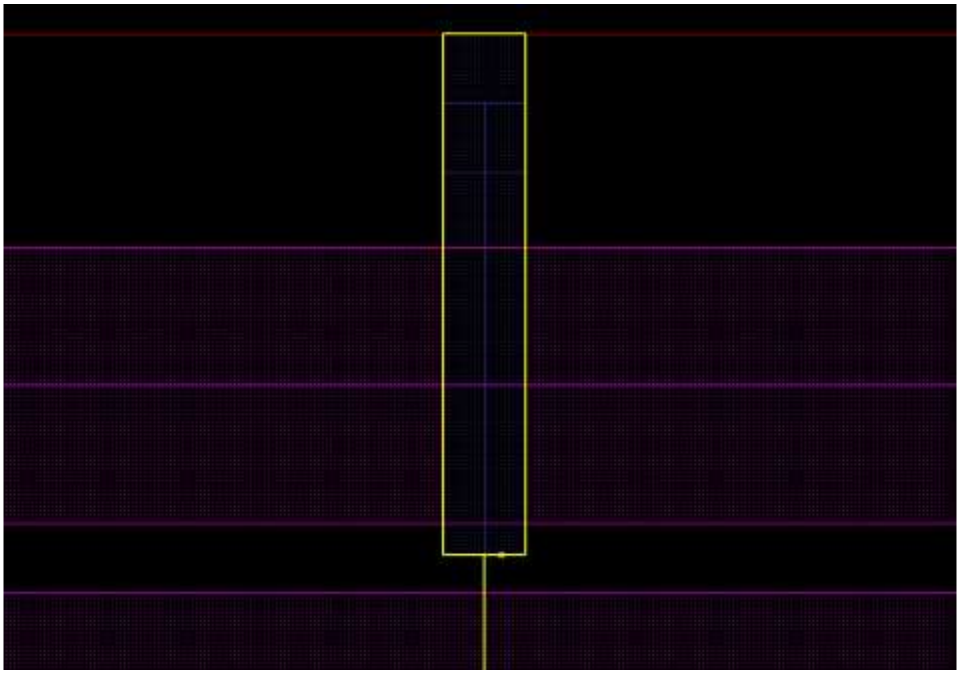
162. Connect the nets **A<0:3>** and **B<0:3>** in the layout editor by follow the flightlines using the **Create → Bus** command. You can see that with the Constraint Aware editing concept, the spacing between the bus lines is automatically adjusted to be DRC clean.





163. Connect the Net C in the design. Net C has a NDR defined for the width to be 0.6

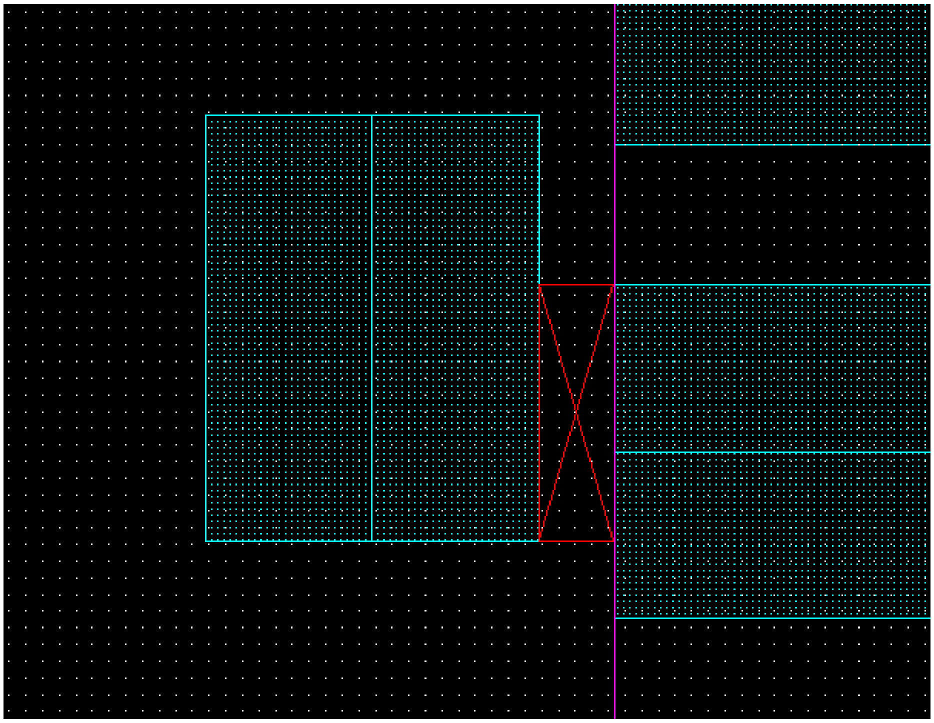
164. Invoke create→Interconnect command and route Net C. You can see that the width of the path is set to 0.6



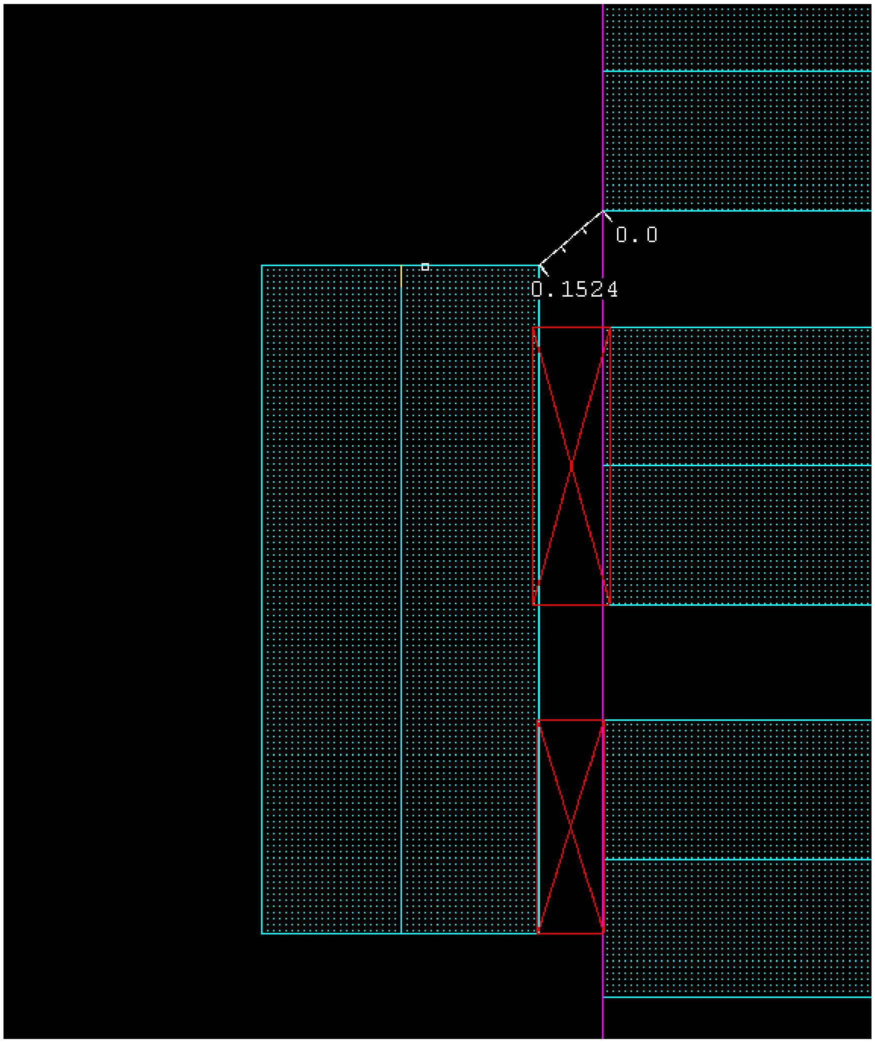
Task 43. DRD Visual/Assist modes and Auto-Fix engine

165. From **Options → DRD** you can observe the DRD settings for **Visual** and **Assist** modes. Check all checkable boxes next from **Constraint Types**

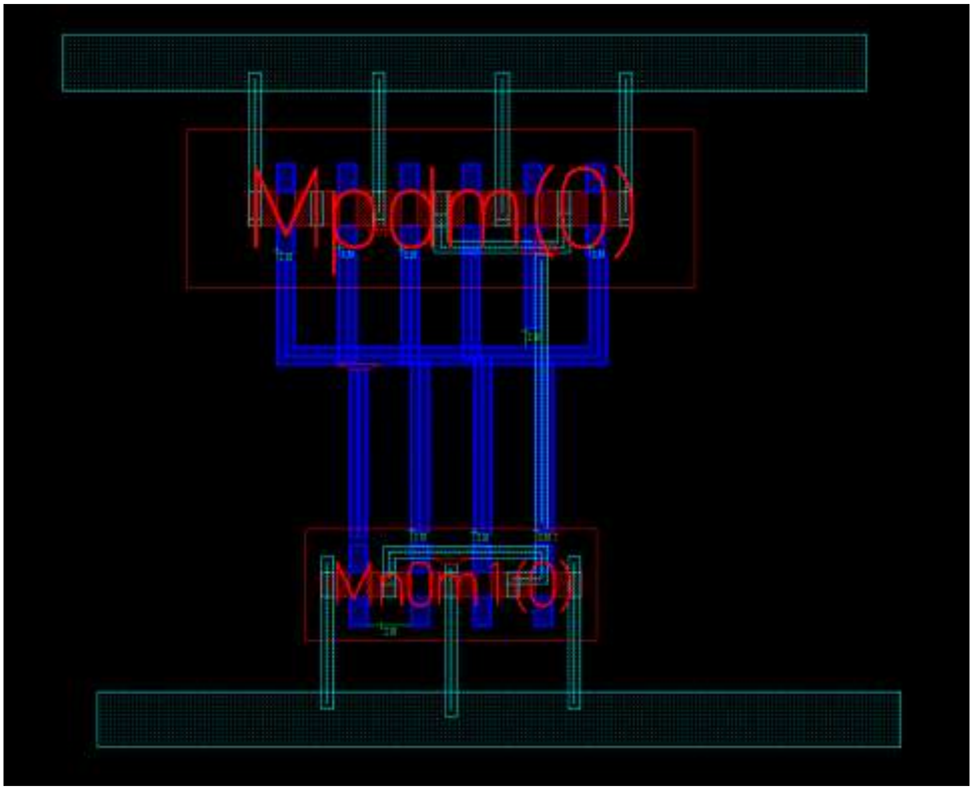
166. Zoom in to the location (12.0 60.0) in the design. You will see one short vertical M1 path and a few horizontal M1 paths



167. Stretch the vertical path towards the top of the layout. You will see that the dynamic DRD markers will only appear within a certain radius from the cursor location.

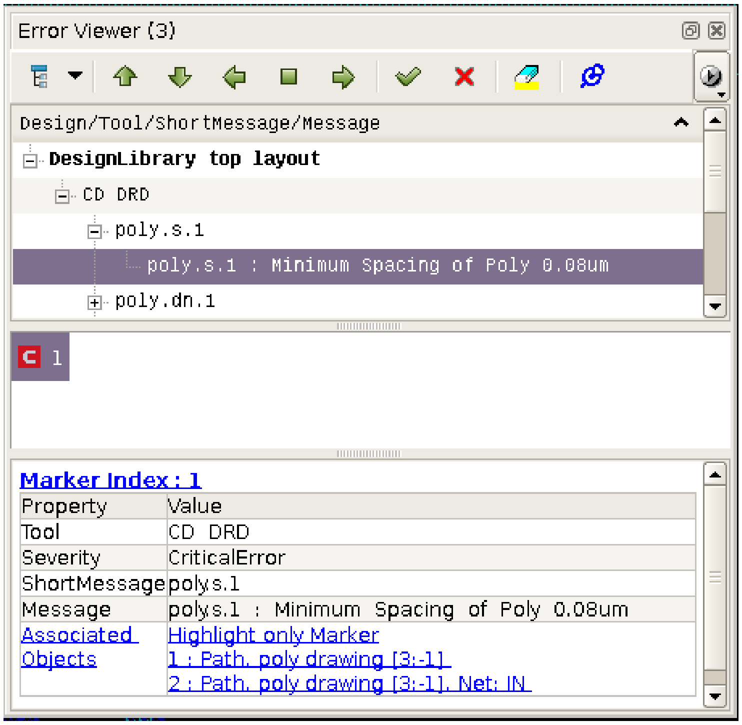


168. Zoom in to the location (59.0 120.0) in the design. You will see an inverter layout.



169. Set the **Stop Level** to **32** using **Shift+F** bindkey.

170. There are DRC violations in the inverter layout. Please open the **Error Viewer** and see the errors.


Error Viewer (3)

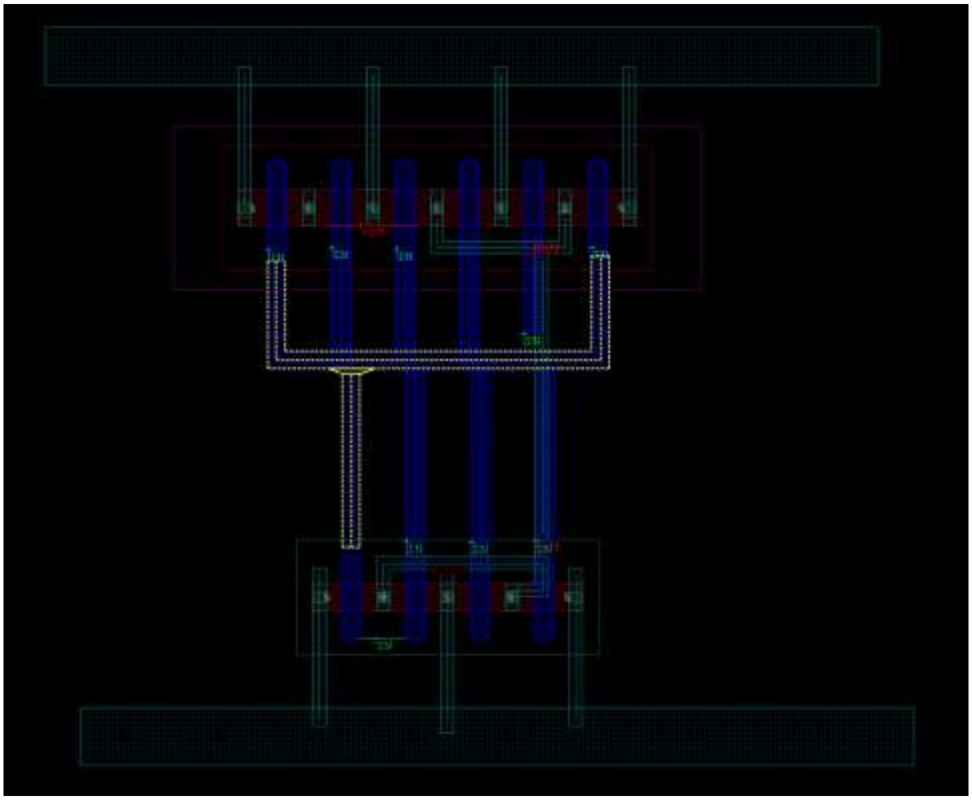
Design/Tool/ShortMessage/Message

```
DesignLibrary top layout
  CD DRD
    poly.s.1
      poly.s.1 : Minimum Spacing of Poly 0.08um
    poly.dn.1
```

1

Marker Index : 1

Property	Value
Tool	CD DRD
Severity	CriticalError
ShortMessage	polys.1
Message	polys.1 : Minimum Spacing of Poly 0.08um
Associated Objects	Highlight only Marker 1 : Path, poly drawing [3:-1] 2 : Path, poly drawing [3:-1], Net: IN



171. Use **Edit → DRD AutoFix** to resolve the errors in the inverter layout:

a. Lock the instances and vias in the design when fixing the DRC errors by enabling the lock in **DRD AutoFix** toolbar.

b. Fix POLY errors only first and then fix the M1 errors.

Hint: Please make M1 invisible and then turn on the lock on 'Invisible Shapes' in **DRD AutoFix** toolbar to fix the POLY violations only.

172. You can see that once the DRC violations are fixed, the POLY and the M1 shapes are moved apart, but the connections are lost.

a. How do you keep the connectivity between the shapes when trying to fix the DRC violations?

Hint: You would need to add **mergeSpaceAllowed** constraint in the technology to achieve this.

6 SDL Initialization and Elaboration

Learning Objectives

During this lab, you will understand the options in the SDL initialization menu and the elaboration rules in the SDL config file.

After completing this lab, you will be able to:

- Set up the SDL initialization menu and understand the options.
- Understand lxUseCell in SDL binding.
- Understand how lxRemoveDevice ignoring parasitic devices in SDL.
- Learn to interpret the fields in Hierarchy Editor.

Lab Duration:
30 minutes

Introduction

The purpose of this lab is to help you understand the SDL initialization and binding.

Below are the features and commands will be used:

Design Navigator schematic **–** a list of devices that can be placed (realized) in layout.

Schematic Assistant – a read-only schematic docked in LE window.

Library/Cell/View (LCV) – A list of library name, cell name, and view name of a design

File Locations

All files for this lab are located in the directory **`SDL_Initialization_and_Elaboration_Lab1`**

Directory Structure

`display.tcl`	Display attributes file
`lib.defs`	Library definition file.
`vcoLib_Init, vcoLib_Init_refLib`	Design libraries

Instructions

Task 44. Start Custom Compiler

173. In the Unix window, change your working directory to **SDL_Initialization_and_Elaboration_Lab1** and invoke Custom Compiler.

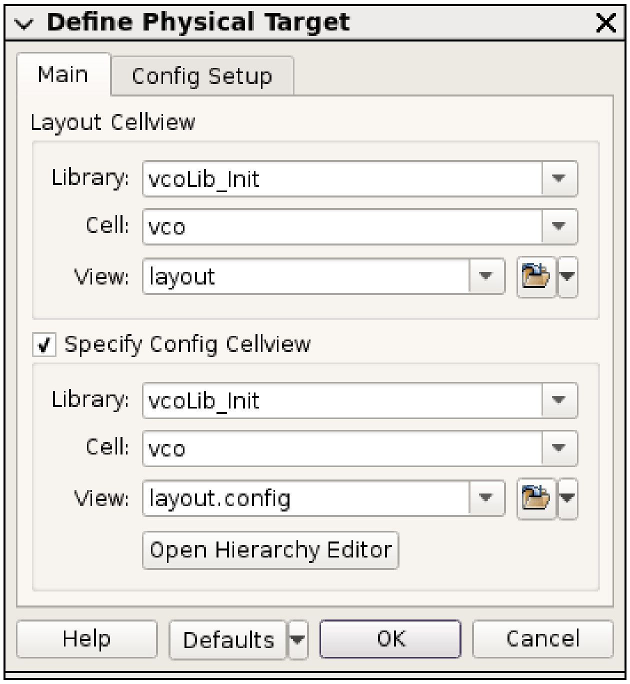
```
unix% cd SDL_Initialization_and_Elaboration_Lab1
unix% custom_compiler &
```

Task 45. SDL initialization and lxUseCell

In this task, you will enable SDL on a pre-existing layout that was created using SDL.

174. From **Library Manager**, open cellview *vcoLib_Init/vco/schematic*.

175. Invoke SDL using **Tools → SDL** and the initialization window **Define Physical Target** pops up as shown below.



Define Physical Target

Main | Config Setup

Layout Cellview

Library: vcoLib_Init

Cell: vco

View: layout

☑ Specify Config Cellview

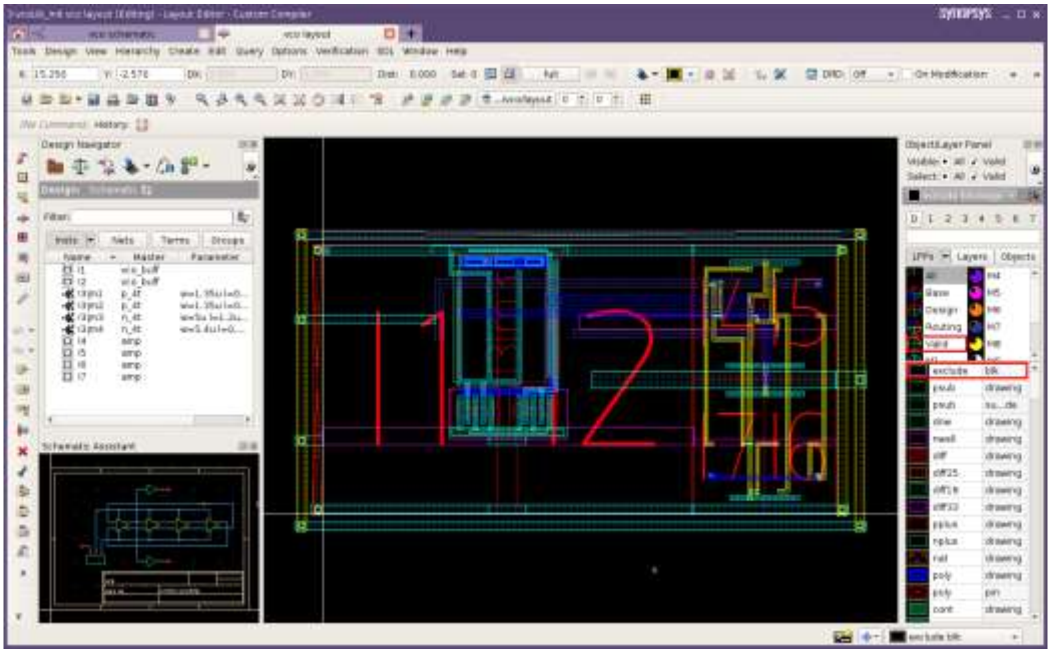
Library: vcoLib_Init

Cell: vco

View: layout.config

176. Click **OK** to initialize **SDL**.

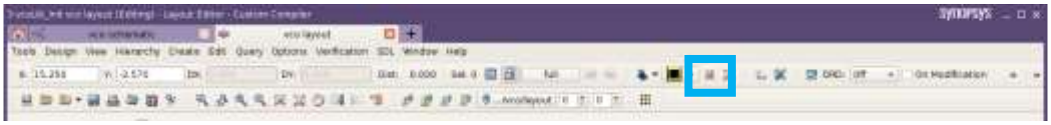
177. A layout view is opened with SDL environment that has a **Design Navigator** in schematic mode at upper left and a **Schematic Assistant** at lower left.



Task 46. SDL Layout Binding

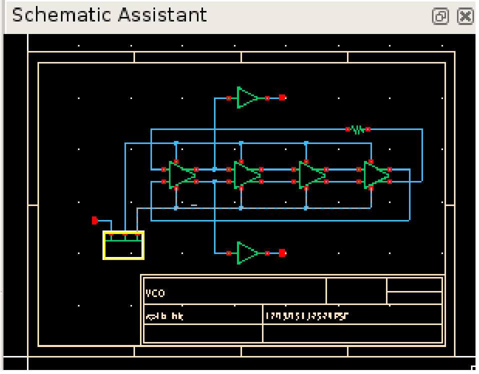
In this task, you will learn what information is used for SDL to bind layout.

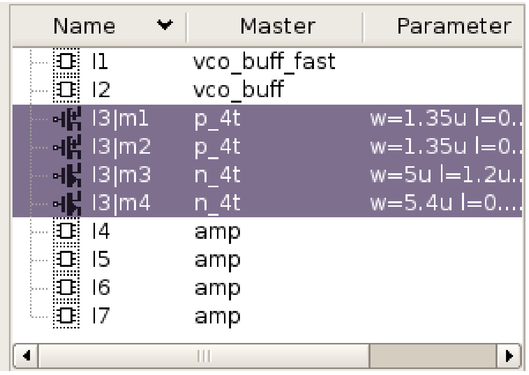
178. Enable cross select from the **Enable Cross Select** icon on LE canvas toolbar.


synopsys
vco schematic
vco layout
Tools Design View Hierarchy Create Edit Query Options Verification SDL Window Help

179. Make sure that no instance is selected in the **Schematic Assistant** or LE canvas.

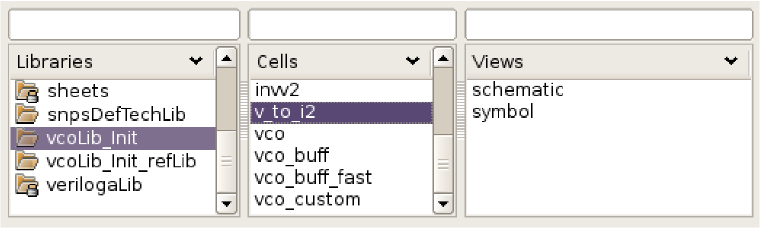
180. In the **Schematic Assistant**, select instance *I3* (vcoLib_Init/v_to_i2) at lower left corner. Observe that it maps to 4 transistors in the **Design Navigator**.





Name	Master	Parameter
I1	vco_buff_fast	
I2	vco_buff	
I3|m1	p_4t	w=1.35u l=0..
I3|m2	p_4t	w=1.35u l=0..
I3|m3	n_4t	w=5u l=1.2u..
I3|m4	n_4t	w=5.4u l=0....
I4	amp	
I5	amp	
I6	amp	
I7	amp	

181. From **Library Manager**, observe that the cell, *vcoLib_Init/v_to_i2*, has no layout view. Since there is no layout, *I3* is bound to *vcoLib_Init/v_to_i2/schematic* and flattened to transistors.



Libraries

- sheets
- snpsDefTechLib
- vcoLib_Init
- vcoLib_Init_refLib
- verilogaLib

Cells

- inw2
- v_to_i2
- vco
- vco_buff
- vco_buff_fast
- vco_custom

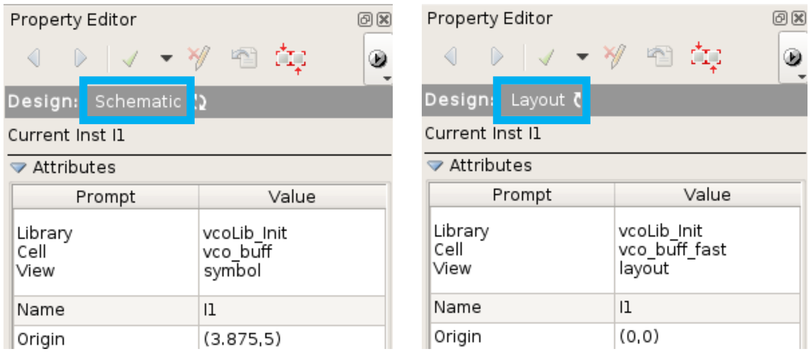
Views

- schematic
- symbol

182. Switch to the *vco layout* tab.

183. In the **Schematic Assistant** select instance *I1* (the *vco_buff* at top) and press binding key **Q** to bring up the **Property Editor** (make sure your cursor is inside **Schematic Assistant**). Observe that the **Design** status of the **Property Editor** is *Schematic*.

184. Move the cursor to layout canvas and press binding key **Q** again. Observe that the **Design** status of **Property Editor** changed to *Layout*. **Property Editor** automatically chooses design status depends on the curosor location (active window) when it is invoked.



Property Editor

Design: Schematic

Current Inst I1

Attributes

Prompt	Value
Library Cell View	vcoLib_Init vco_buff symbol
Name	I1
Origin	(3.875,5)

Property Editor

Design: Layout

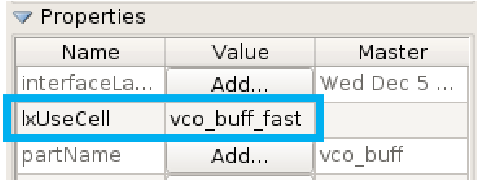
Current Inst I1

Attributes

Prompt	Value
Library Cell View	vcoLib_Init vco_buff_fast layout
Name	I1
Origin	(0,0)

185. Click on **Design: Layout** to switch to *Schematic* mode manually.

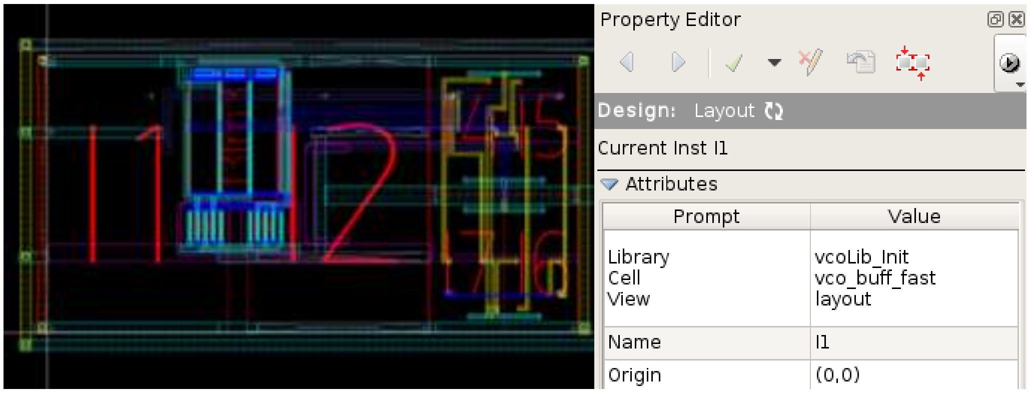
186. Expand the **Properties** section and observe the property *lxUseCell*. It is set to *vco_buff_fast*. This property instructs SDL to use *vco_buff_fast/layout* as the SDL binding for this schematic instance (instead of using *vco_buff/layout)*.



Properties

Name	Value	Master
interfaceLa...	Add...	Wed Dec 5 ...
lxUseCell	vco_buff_fast	
partName	Add...	vco_buff

187. Switch **Property Editor** to layout mode. Observe that the LCV is *vcoLib_Init/vco_buff_fast/layout*.



Property Editor

Design: Layout

Current Inst I1

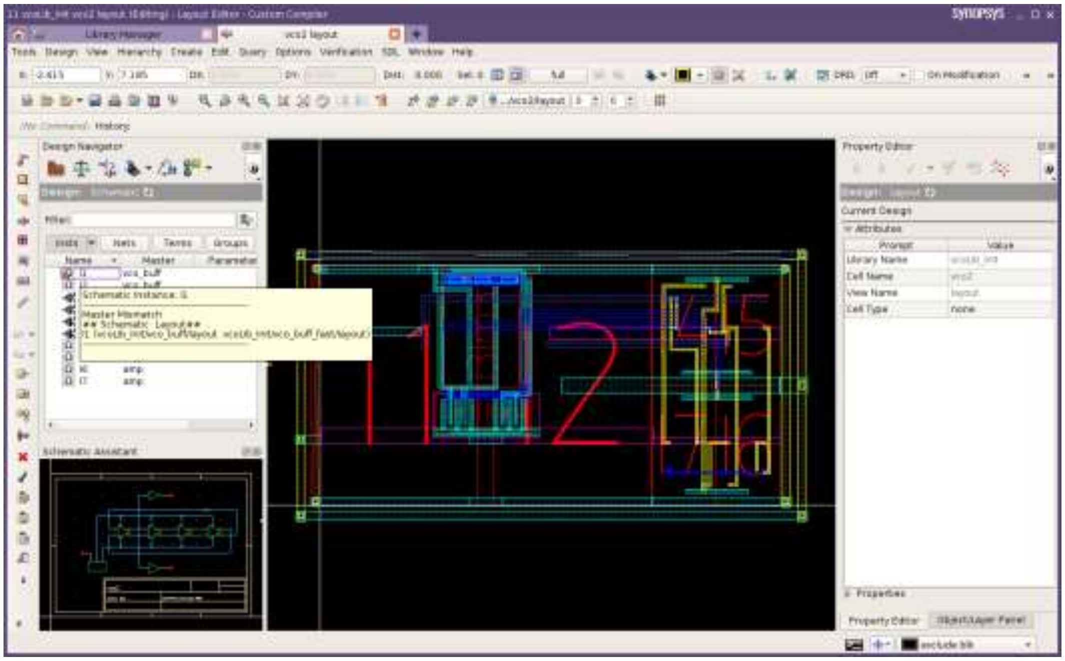
Attributes

Prompt	Value
Library Cell View	vcoLib_Init vco_buff_fast layout
Name	I1
Origin	(0,0)

Task 47. SDL Instance Mapping

In this task, you will learn how to set instance mapping.

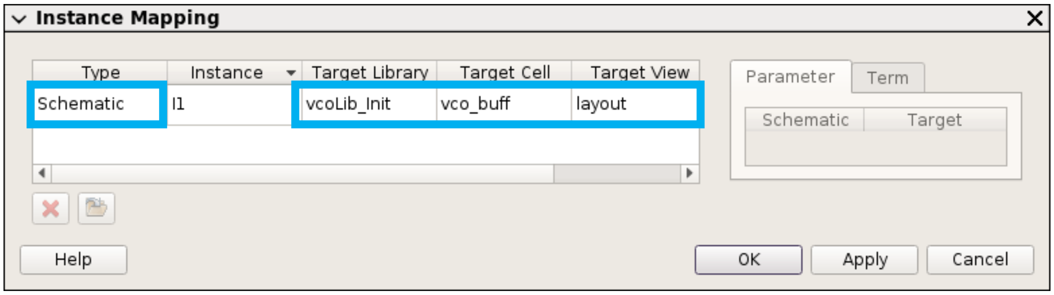
188. From **Library Manager**, open design *vcoLib_Init/vco2/layout* and you should be in SDL mode. Observe in **Design Navigator** that instance *I1* is flagged with error.



Schematic Instance: I1
Master Mismatch
** Schematic Layout**
I1 (vcoLib_Init/vco_buff/layout vcoLib_Init/vco_buff_fast/layout)

189. Move the cursor on top of the error (*I1*) in **Design Navigator** and stay for few seconds. A balloon popup and shows there is a master mismatch on this instance where *vco_buff* is expected but got *vco_buff_fast* in layout. This is exactly the same situation with the previous task but *lxUseCell* doesn't exist in this schematic.

190. Select *I1* in **Design Navigator** and select **Instance Mapping** from **CSM** (right-mouse click). An instance mapping dialog popup and indicates the current mapping is *vcoLib_Init/vco_buff/layout* which comes from schematic.



Type	Instance	Target Library	Target Cell	Target View
Schematic	I1	vcoLib_Init	vco_buff	layout

191. Change the target cell to *vco_buff_fast* by either the emthods:

a. Double click on the target cell *vco_buff* and change the value.

b. Select the row in the dialog and use the browse icon 📂 at bottom left to choose the new layout.

192. Click **OK** to apply the change and observe that the error in **Design Navigator** on *I1* is disappeared.

193. Close the design

Task 48. Usage of lxRemoveDevice

In this task, you will use **lxRemoveDevice** to ignore the instances.

194. Switch back to the *vco layout* tab and observe that there is a resistor, *R0*, in **Schematic Assistant** but not in **Design Navigator**. Select *R0* in **Schematic Assistant**.

195. Bring up the **Property Editor** (and switch to schematic mode if necessary) and examine the **Properties** section. Observe that:

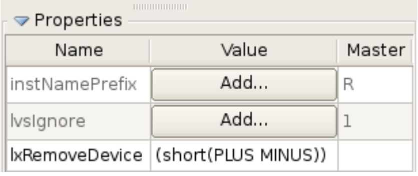
    a. *lvsIgnore* is set to *1*. This indicates that this device will not be placed for SDL in layout (that's the reason you don't see *R0* in **Design Navigator**)

    b. *lxRemoveDevice* is set to *(short(PLUS MINUS))*. When *R0* is removed from SDL, SDL chooses one of the nets on PLUS and the MINUS terms based on the precedence:

        i. Global net

        ii. Net connected to pin

        iii. Alphanumeric



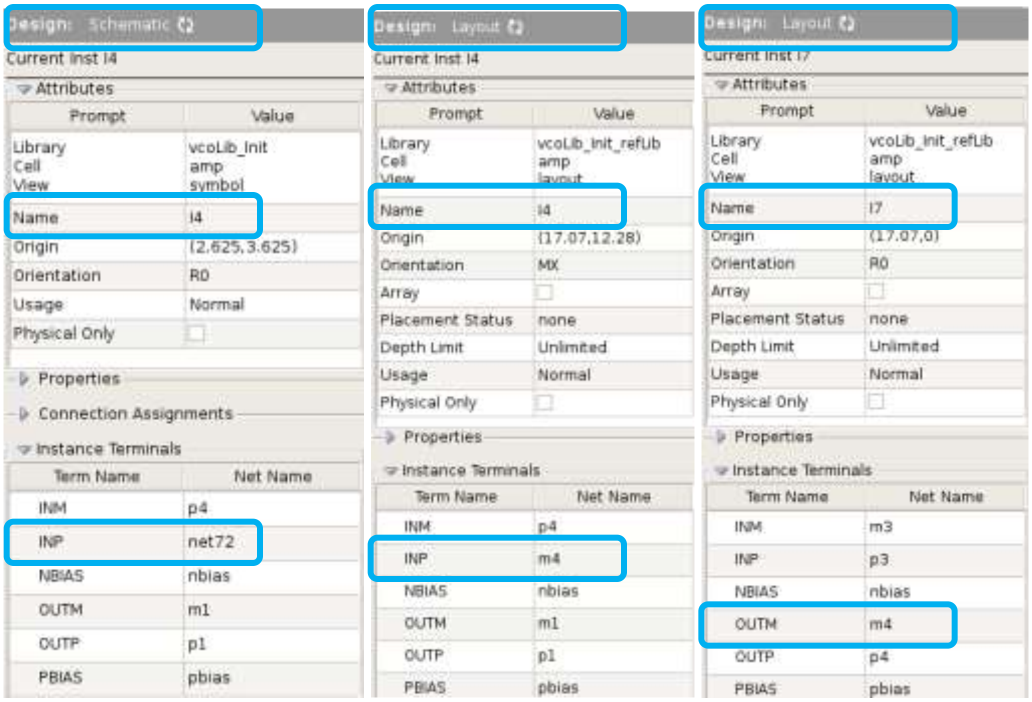
Name	Value	Master
instNamePrefix	Add...	R
lvsIgnore	Add...	1
lxRemoveDevice	(short(PLUS MINUS))	

196. You can verify that the nets by examining the **Instance Terminals** of instance *I4* and instance *I7* in the layout shown with the following steps.

197. Use bindkey **Ctrl-F** to set view stop level to 0. You will now be able to see the instance names in the layout.

198. Select instance *I4*. Bring up the **Property Editor** (make sure it's in layout mode) to see the properties. Expand the **Instance Terminals** and examine the net on term *INP*.

199. Switch to instance *I7* and examine the net on term *OUTM*. Notice in the layout view that the net connected to instance *I4* terminal *INP* is the same with the net on instance *I7* terminal *OUTM*. They are shorted since the resistor *R0* has been removed.



Design: Schematic

Current Inst I4

Attributes

Prompt	Value
Library Cell View	vcoLib_Init amp symbol
Name	I4
Origin	(2.625,3.625)
Orientation	R0
Usage	Normal
Physical Only	□

Properties

Connection Assignments

Instance Terminals

Term Name	Net Name
INM	p4
INP	net72
NBIAS	nbias
OUTM	m1
OUTP	p1
PBIAS	pbias

Design: Layout

Current Inst I4

Attributes

Prompt	Value
Library Cell View	vcoLib_Init_refLib amp layout
Name	I4
Origin	(17.07,12.28)
Orientation	MX
Array	□
Placement Status	none
Depth Limit	Unlimited
Usage	Normal
Physical Only	□

Properties

Instance Terminals

Term Name	Net Name
INM	p4
INP	m4
NBIAS	nbias
OUTM	m1
OUTP	p1
PBIAS	pbias

Design: Layout

Current Inst I7

Attributes

Prompt	Value
Library Cell View	vcoLib_Init_refLib amp layout
Name	I7
Origin	(17.07,0)
Orientation	R0
Array	□
Placement Status	none
Depth Limit	Unlimited
Usage	Normal
Physical Only	□

Properties

Instance Terminals

Term Name	Net Name
INM	m3
INP	p3
NBIAS	nbias
OUTM	m4
OUTP	p4
PBIAS	pbias

200. Close all designs.

7

Design Creation Using SDL Symbolic Editor

Learning Objectives

After completing this lab, you should be able to:

- Use Symbolic Editor chaining mode to edit the placement
- Create a cell template to match with a specific cell height

Lab Duration:
30 minutes

Introduction

The purpose of this lab is to help you understand the SDL **Symbolic Editor** functions.

Below are the features and commands will be used:

Symbolic Editor – a device placement tool that displays key information of the devices. It makes device placement editing easier.

Design Navigator Schematic – a list of devices that can be placed (realized) in layout.

Schematic Assistant – a read-only schematic docked in LE window.

Cell Template – Symbolic Editor function to customize layout template for custom digital layout style.

S/D – Source and drain of the transistors in the symbolic canvas

P/G – Power and ground

File Locations

All files for this lab are located in the directory **`SDL_Symbolic_Editor_Lab1`**

Directory Structure

`display.tcl`	Display attributes file
`lib.defs`	Library definition file
`snps_mpll`	Design library

Instructions

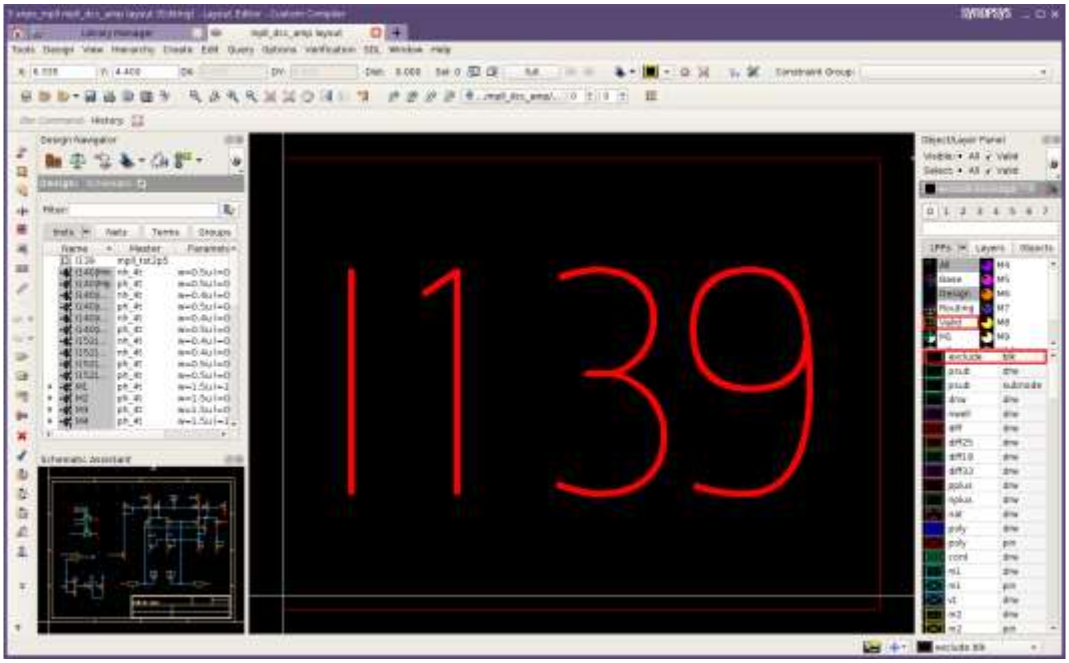
Your goal is to place some transistors of a diff amp by using the SDL **Symbolic Editor** chaining mode.

Task 49. Start Custom Compiler and SDL

201. In the Unix window, change your working directory to **SDL_Symbolic_Editor_Lab1** and invoke Custom Compiler.

```
Unix% cd SDL_Symbolic_Editor_Lab1
Unix% custom_compiler &
```

202. Open *snps_mpll/mpll_dcc_amp/layout* and you have the SDL environment.

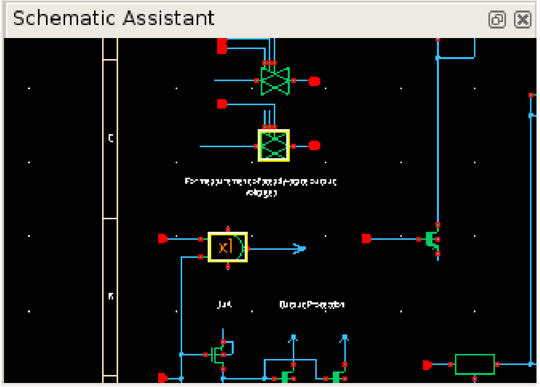


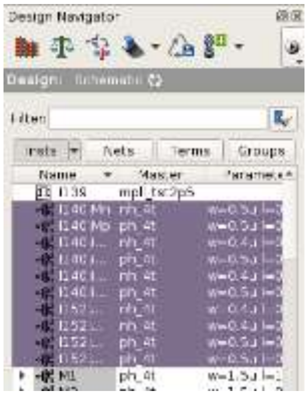
Observe that there is one transmission gate layout instance, *I139*, placed in the LE canvas. In **Design Navigator**, the other transmission gate, *I140*, is flattened to transistor level (with instance name *I140|**) and has not been placed yet (the background of these instances is in grey).

Task 50. Place Instances by Using Symbolic Editor Chaining Mode

In this task, you will learn to use **Symbolic Editor** chaining mode to place transistors.

203. Select the transmission gate, *I140*, and the nand gate, *I152*, from **Schematic Assistant** and observe that all transistors belong to *I140* and *I152* in **Design Navigator** are also selected.




Design Navigator

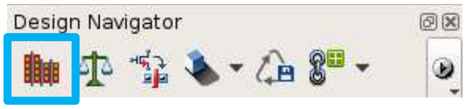
Design: Schematic

Filter

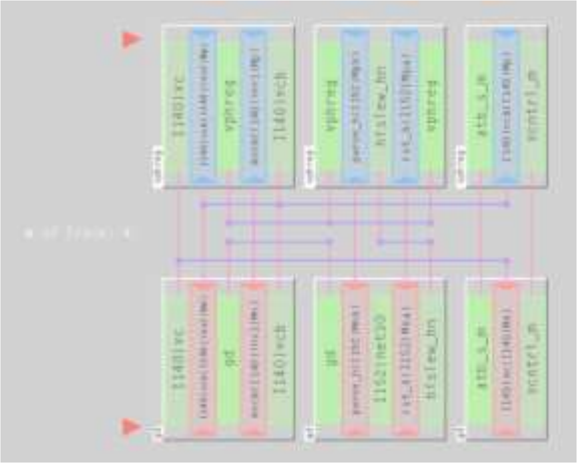
Insts | Nets | Terms | Groups

Name	Master	Parameters
I138	mpl_tsr2p5	
I140|Mn	nh_4t	w=0.5u l=
I140|Mp	ph_4t	w=0.5u l=
I140|...	nh_4t	w=0.4u l=
I140|...	ph_4t	w=0.5u l=
I140|...	nh_4t	w=0.4u l=
I140|...	ph_4t	w=0.5u l=
I152...	nh_4t	w=0.4u l=
I152...	nh_4t	w=0.4u l=
I152...	ph_4t	w=0.5u l=
I152...	ph_4t	w=0.5u l=
M1	ph_4t	w=1.5u l=

204. Launch **Symbolic Editor** chaining mode from **Design Navigator** toolbar.



205. **Symbolic Editor** gives a suggested placement with maximized S/D sharing and gate alignment. Observe that the transistors from the nand gate (*I152*|*) are placed in the middle.

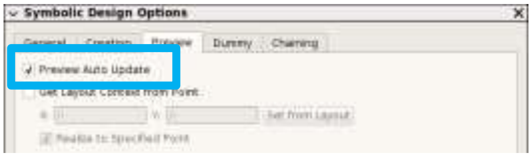


206. Select **Options → Design** and

a. Enable **Preview Auto Update** in **Preview** tab.

b. In **Creation** tab, observe that the **Same Row** spacing is 1 (um) and the **Different Row** spacing value is 1 (um).

c. Click on **OK** to apply the change.


Symbolic Design Options

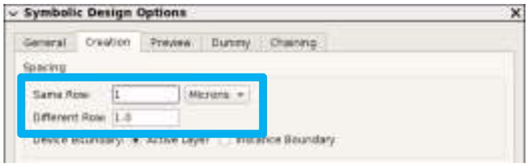
General | Creation | Preview | Dummy | Chaining

☑ Preview Auto Update

□ Get Layout Context from Point

X: 0 Y: 0 Set from Layout

☑ Realize to Specified Point


Symbolic Design Options

General | Creation | Preview | Dummy | Chaining

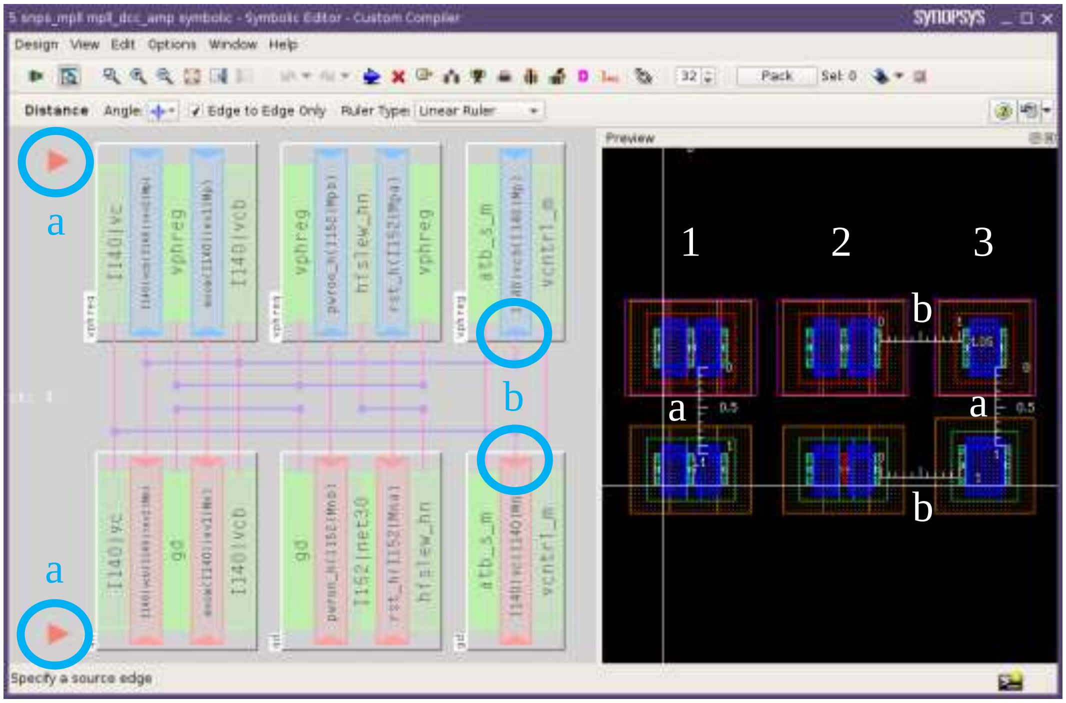
Spacing

Same Row: 1 Microns

Different Row: 1.0

Device Boundary: ● Active Layer ○ Instance Boundary

207. Select **Window → Assistants → Preview** to open the **Preview** assistant. You can drag the preview assistant to adjust its location as below.

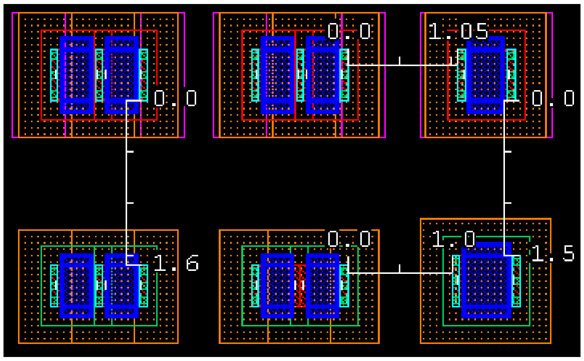

5 snps_mpll mpll_dcc_amp symbolic - Symbolic Editor - Custom Compiler
Design View Edit Options Window Help
Distance Angle Edge to Edge Only Ruler Type: Linear Ruler
Pack Sel: 0
Preview
a
b
b
a
1
2
3
b
a
a
b
Specify a source edge

a. Observe that the diffusion spacing between PMOS and NMOS (different row spacing) is 1.1 um in group 1 and 1.0 um in group 3. This is due to the row alignment of NMOS and PMOS. You can get the row alignment status from the red triangle at the left-hand side of each row in the symbolic canvas. In this case, the row alignment of NMOS is bottom and PMOS is top. Since gate width of NMOS is 0.4 um in group 1 and 0.5 um in group3, it causes a 0.1 um difference.

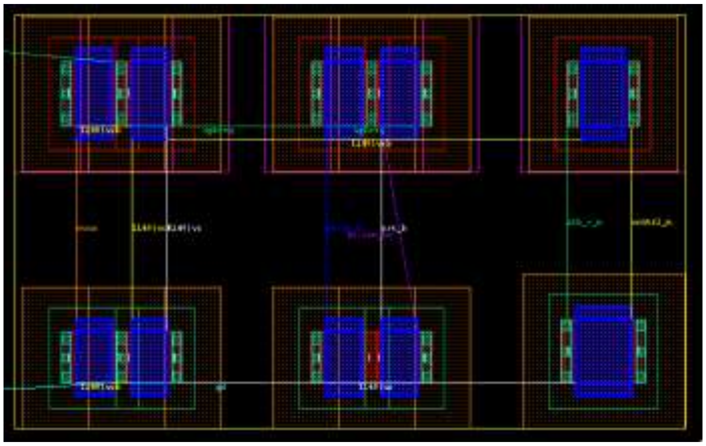
b. Observe that the diffusion spacing between group 2 and group 3 (same row spacing) is 1.05 um for PMOS and 1.0 um for NMOS. This is due to the gate alignment of NMOS and PMOS gates. You can get the gate alignment status from the triangle at top/bottom of the gates (red for NMOS and blue for PMOS) in the symbolic canvas. In this case, the gate length in group 3 is 0.33 um for PMOS and 0.43 um for NMOS so it causes a 0.05 um difference.

Hint: you can use **Edit → Distance** to measure the distance.

208. Change the **Different Row** spacing value to 1.5 in **Options → Design → Creation** and observe the layout difference from **Preview** assistant.



209. Select **Design → Realize** to place the devices in layout canvas. Observe that layout instances are grouped in a figure group (option **Group Realized Design** in **Symbolic Editor**).



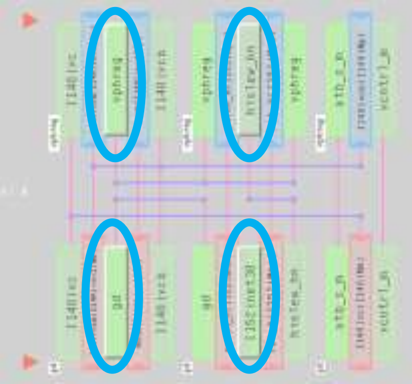
Task 51. Modify Placement of Placed Layout Instance by Using Symbolic Editor

In this task, you will learn to use **Symbolic Editor** chaining to modify placement.

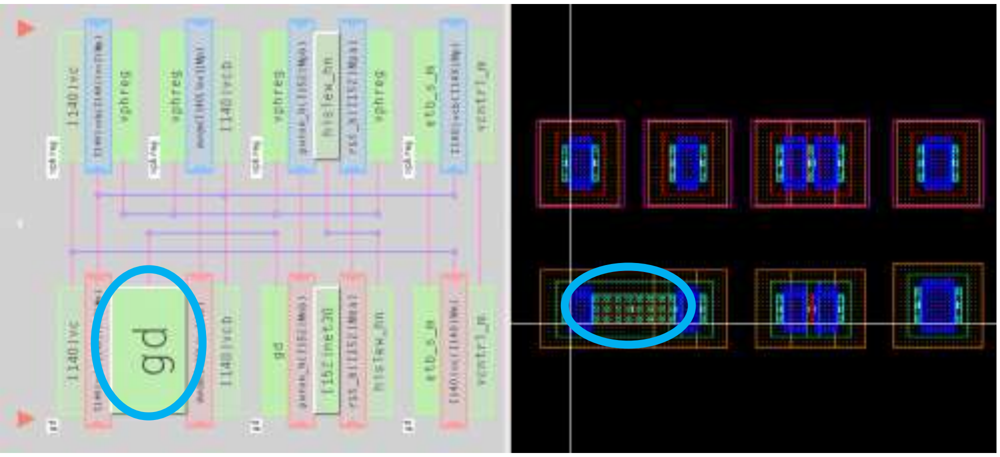
210. Select the layout figure group created from previous task and select **SDL → Symbolic Editor → Chaining** to invoke **Symbolic Editor** chaining mode from selected layout instances.

Note: When the selected set is from layout canvas, you can't use the **Symbolic Editor** from the **Design Navigator** toolbar.

211. Select **Edit → SplitSD**. Observe that **Symbolic Editor** pops up the S/D candidates that can be split.



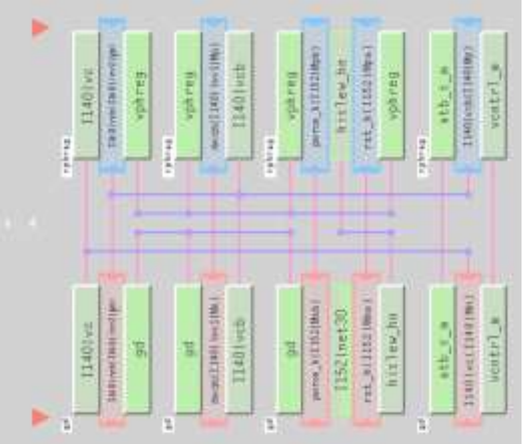
212. Click on the S/D with net name *vphreg* in PMOS (top) row to split the S/D. Observe that after the PMOS S/D is split, the NMOS (bottom) S/D with net name *gd* is still shared but enlarged. This is due to the two NMOS gates are still aligned to PMOS gates. **Symbolic Editor** has the capability to stretch S/D in layout to keep gate aligned and S/D abutted.



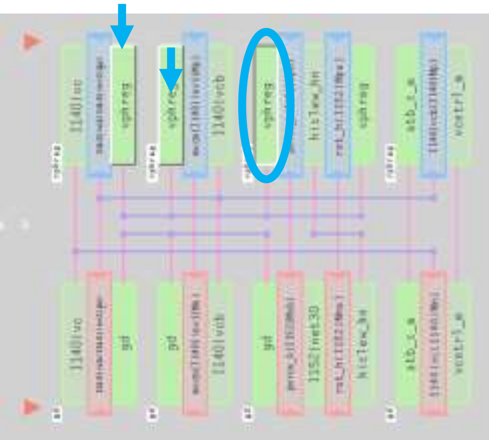
Note: If transistor layout does not have the capability to enlarge S/D, **Symbolic Editor** breaks abutment and keeps gate aligned.

213. Split the S/D with net name *gd* in NMOS row (the enlarged one) too.

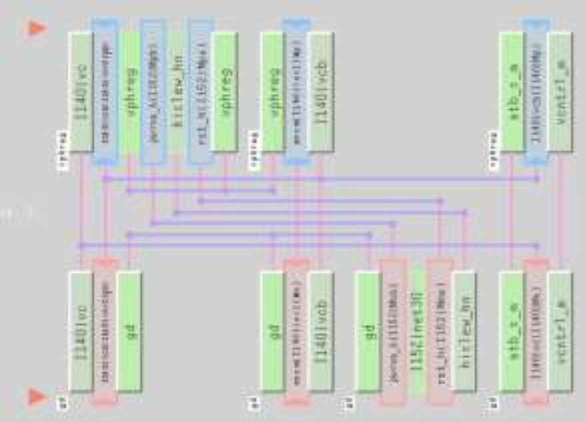
214. Select **Edit → MergeSD**. Observe that S/D candidates pop up.



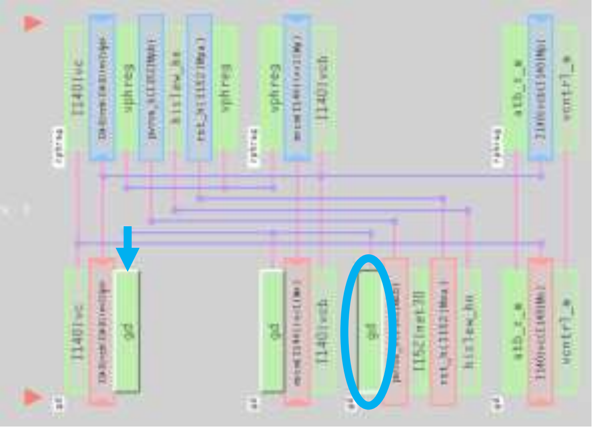
215. Select the S/D with net *vphreg* at left-hand side of gate *I152|Mpb*. Observe that once the S/D with net *vphreg* is selected, **Symbolic Editor** pops up only candidates with the same net name because only the S/D with the same net name can be merged.



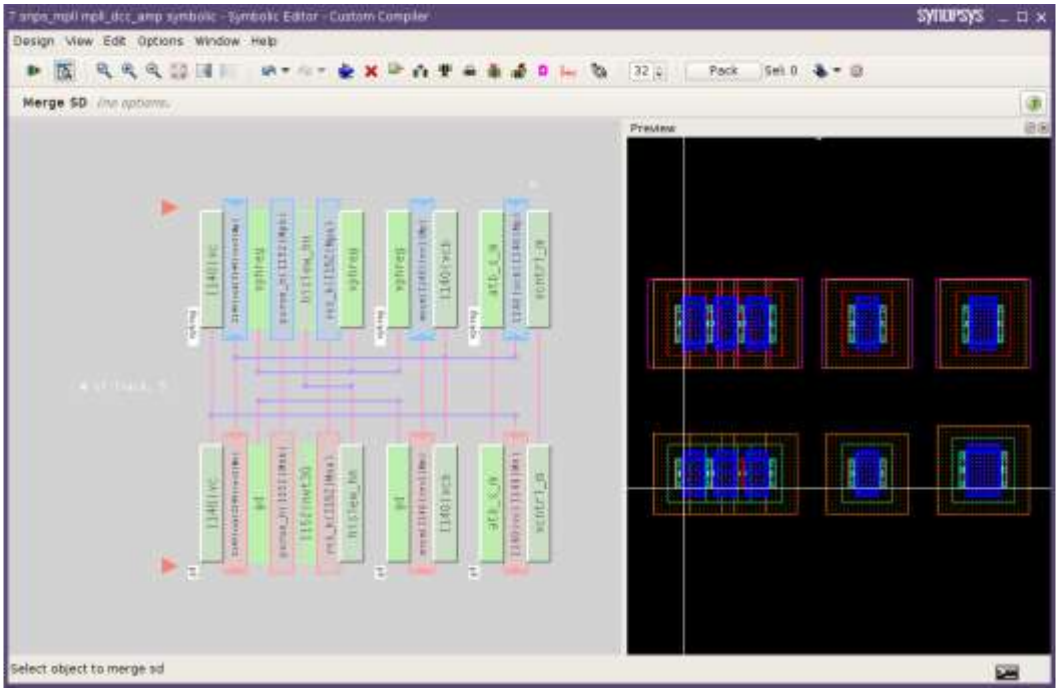
216. Select the left candidate to merge. Observe that the **MergeSD** command not only merges S/D but also move devices.



217. Merge the S/D (net name *gd*) at left-hand side of gate *I152|Mnb* to left candidate.



218. Now you have the placement.


* snps_mpll mpll_dcc_amp symbolic - Symbolic Editor - Custom Compiler
SYNOPSYS
Design View Edit Options Window Help
Pack
Sel: 0
Merge SD
line options.
Preview

Select object to merge sd

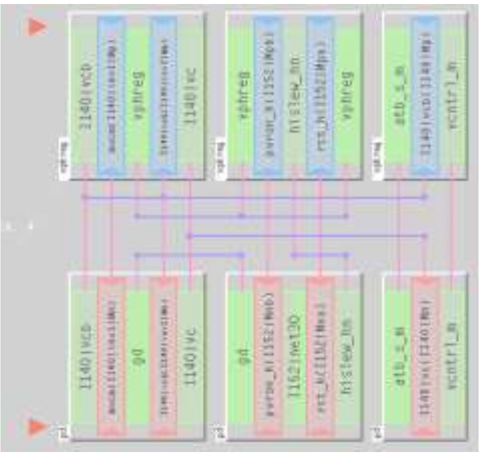
Task 52. Reset the Placement, Add Devices Incrementally,

and Add Dummies

In this task, you will learn to use **Rechain** command to reset the placement, add more devices for placement and add dummies.

219. If you don’t like the placement you edited, you can use **Rechain** command and **Symbolic Editor** gives you a suggested placement on selected devices.

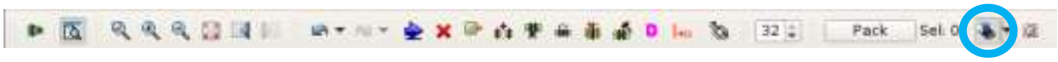
220. Select all devices and select **Edit → Rechain**. Observe that the command rearranges the placement to the initial placement (task 2, step 3)



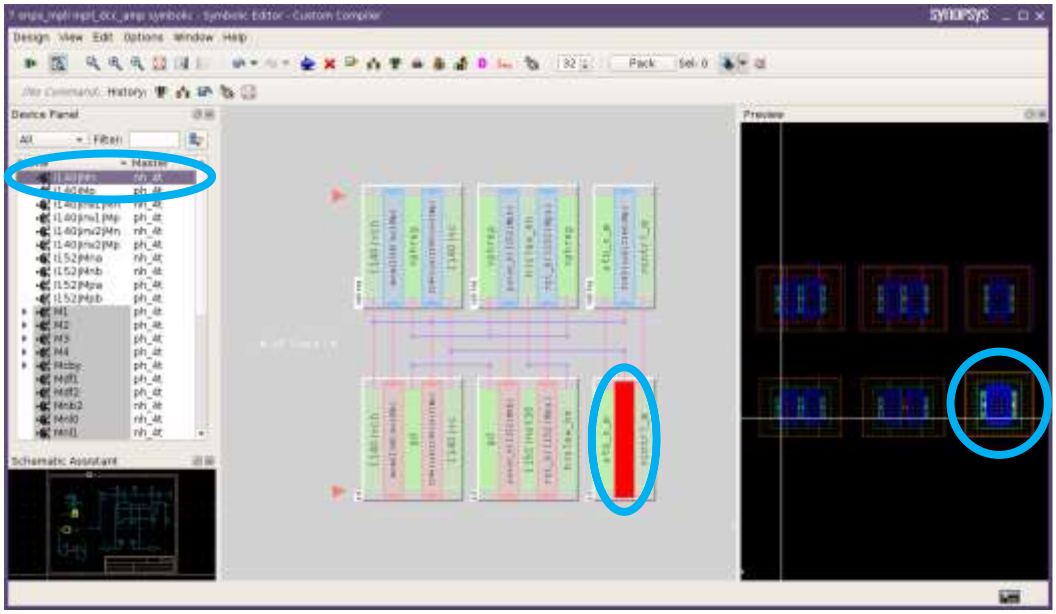
221. Select **Window → Assistants → Device Panel** and **Window → Assistants → Schematic Assistant**.

222. Observe in **Device Panel** that some devices are with white background and some are with grey background. The white background indicates the devices are already in the **Symbolic Editor** (symbolic canvas).

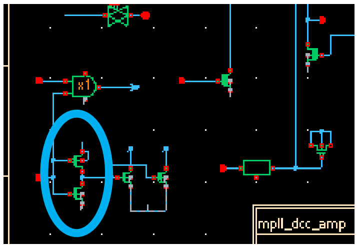
223. Enable the **Cross Shadow** from the toolbar.



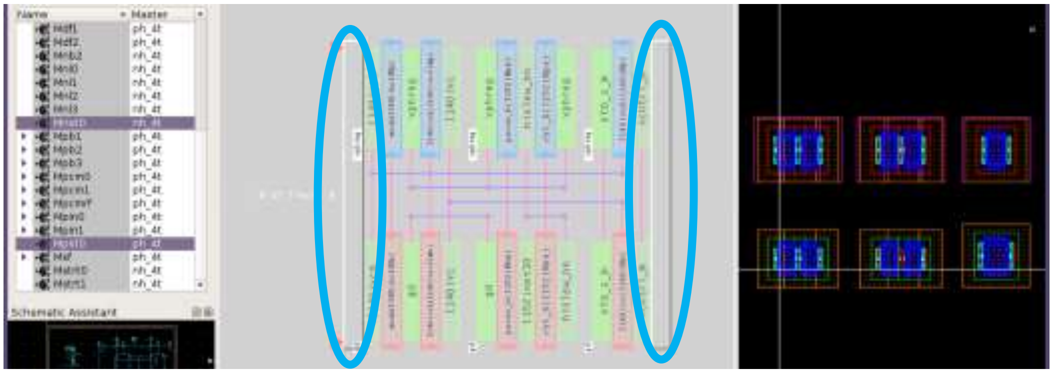
224. Select any devices with white background in **Device Panel**, observe that the corresponded devices in the symbolic canvas and **Preview** are highlighted.


Symbolic Editor - Custom Compiler
Design View Edit Options Window Help
Pack
Device Panel
All
Filter:
Master
Schematic Assistant
Preview

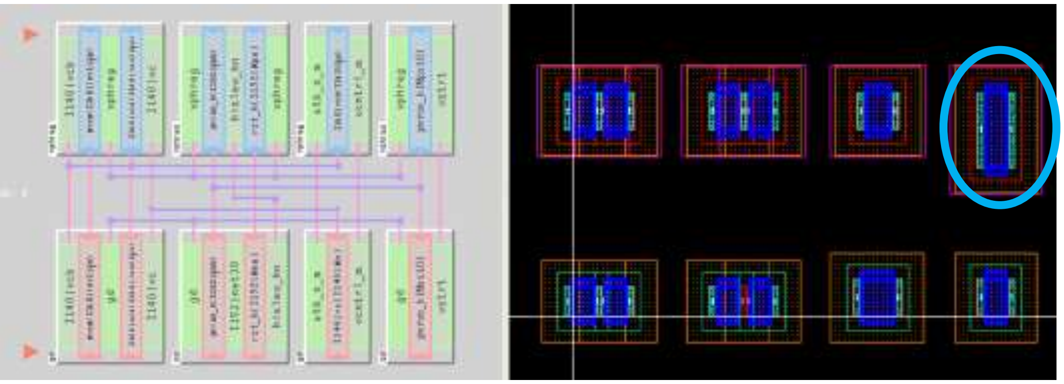
225. Select *Mpst0* and *Mnst0* from **Device Panel** or **Schematic Assistant** (lower-left corner).



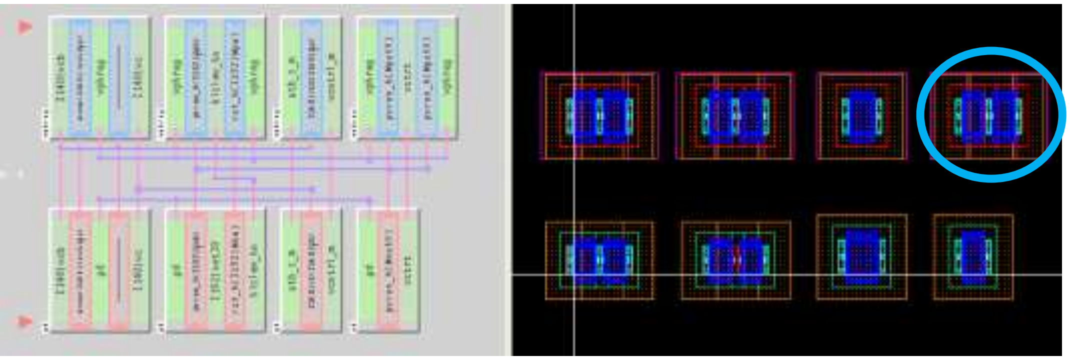
226. Select **Edit → Add** and observe two candidates are available for placing the new devices. Click on the right-hand side candidate to place.



227. In the **Preview**, observe that the gate width of *Mpst0* is larger than the other PMOSs.

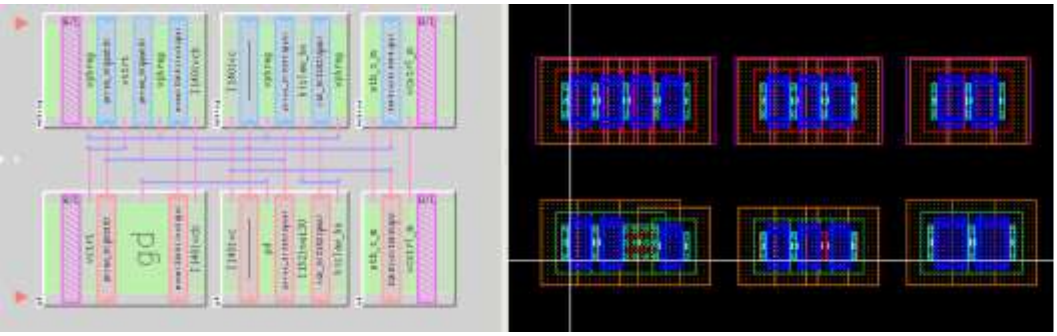


228. Select *Mpst0* in symbolic canvas and select **Edit → Fold Gates**. Observe that *Mpst0* has been folded into two gates with half of the original gate width.

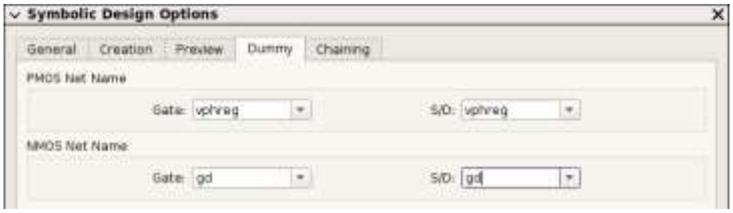


229. Select all devices and select **Edit → Rechain** to rearrange the placement.

230. Select **Edit → Add Dummy** and click on the left-most and right-most S/D in each row to add dummies. Observe that no net on dummies except the S/D that merged with devices.



231. Select **Options → Design → Dummy** tab. Set the net names for dummies.



Symbolic Design Options

General | Creation | Preview | Dummy | Chaining

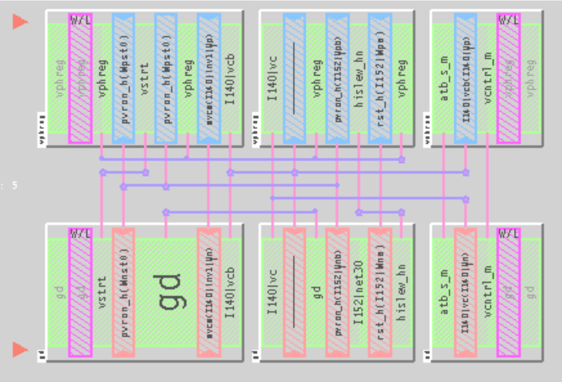
PMOS Net Name

Gate: vphreg S/D: vphreg

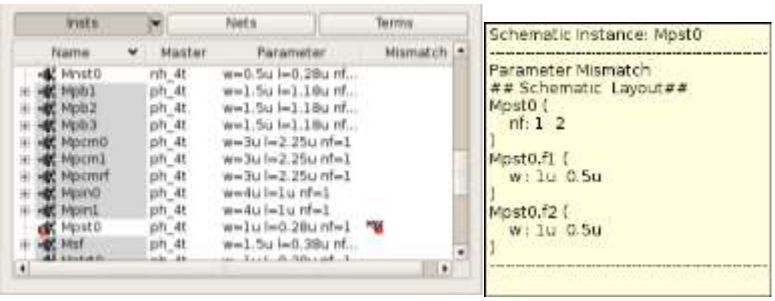
NMOS Net Name

Gate: gd S/D: gd

232. Dummy gates and S/D that are not shared with devices now have nets.



233. Select **Design → Realize** and observe that a mismatch is flagged on *Mpst0*. This is because the device was folded in **Symbolic Editor**. Move the cursor on the mismatch icon to see detail from the pop up tip.



Name	Master	Parameter	Mismatch
Mnst0	nh_4t	w=0.5u l=0.28u nf...	
Mpb1	ph_4t	w=1.5u l=1.18u nf...	
Mpb2	ph_4t	w=1.5u l=1.18u nf...	
Mpb3	ph_4t	w=1.5u l=1.18u nf...	
Mpcm0	ph_4t	w=3u l=2.25u nf=1	
Mpcm1	ph_4t	w=3u l=2.25u nf=1	
Mpcmrf	ph_4t	w=3u l=2.25u nf=1	
Mpin0	ph_4t	w=4u l=1u nf=1	
Mpin1	ph_4t	w=4u l=1u nf=1	
Mpst0	ph_4t	w=1u l=0.28u nf=1	✗
Msf	ph_4t	w=1.5u l=0.38u nf...	

```
Schematic Instance: Mpst0
------------------------------------------
Parameter Mismatch
## Schematic  Layout##
Mpst0 {
   nf: 1  2
}
Mpst0.f1 {
   w: 1u  0.5u
}
Mpst0.f2 {
   w: 1u  0.5u
}
------------------------------------------
```

234. From the mismatch report, finger number (*nf*) is 1 in schematic but 2 in layout. Finger width (*w*) is 1 um in schematic but 0.5 um in layout. Although total width values are still 1 um, parameter *nf* and *w* values are mismatched.

235. Select **SDL → Options** and enable option **Verify Total Width Only**. Select **Apply** and observe that the mismatch is gone. This option allows SDL to compare the product of finger width and finger number (total width) instead of individual values.

Task 53. Create Cell Template

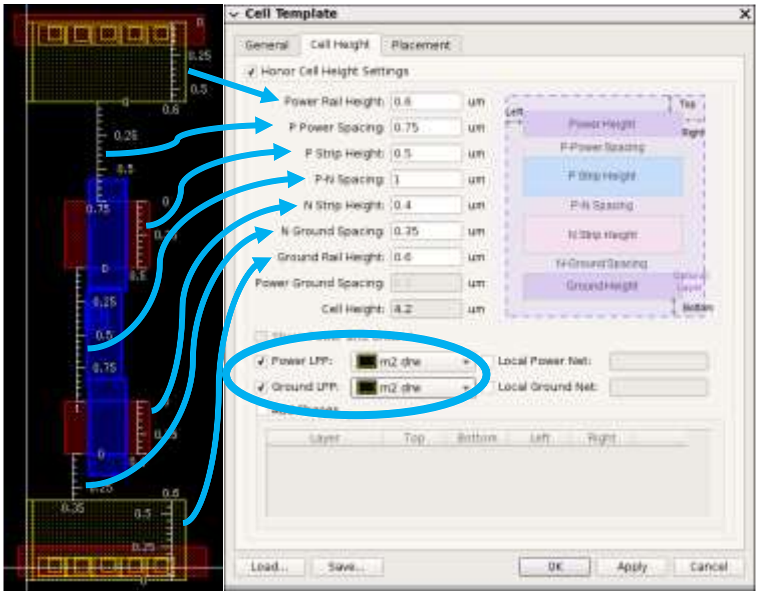
In this task, you will learn to create a cell template to customize the layout created from **Symbolic Editor** chaining to match the layout style (cell height, P/G rail, … etc.) of existing design.

236. Select the layout instances (or group) realized previously from **Symbolic Editor** chaining and select **SDL → Symbolic Editor → Chaining** to launch **Symbolic Editor** again.

237. Select **Options → Cell Template**. Open *snps_mpll/m2p5inv1x/layout* as reference for the **Cell Template**.

238. In the **Cell Height** tab

- **a.** Enable **Honor Cell Height Setting**.
- **b.** Follow the reference layout for all the height/spacing values.
- **c.** Enable **Power LPP** and **Ground LPP** and set the value to *{m2 drawing}* for P/G rails.



Cell Template

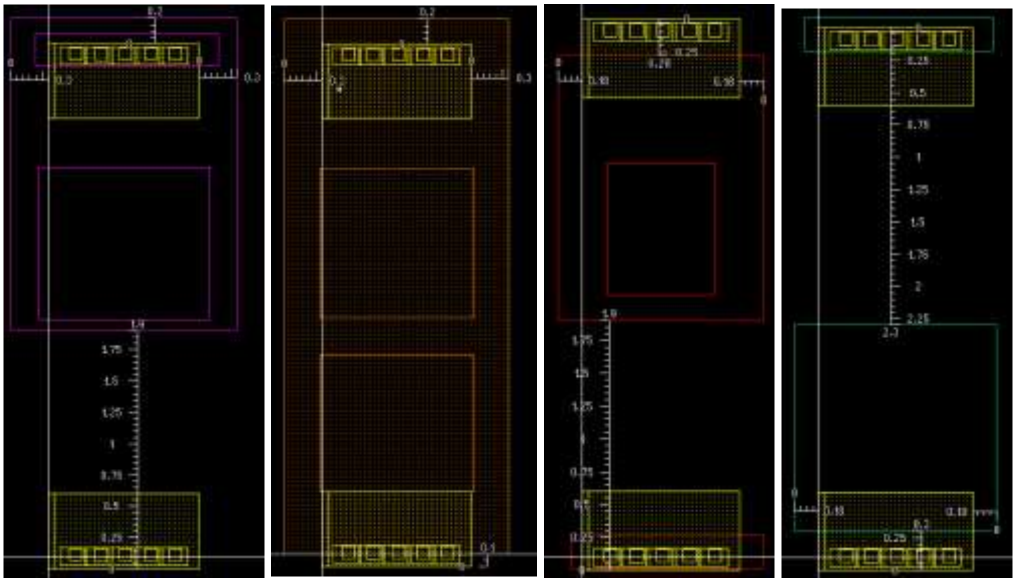
☑ Honor Cell Height Settings

Power Rail Height:	0.6	um
P Power Spacing:	0.75	um
P Strip Height:	0.5	um
P-N Spacing:	1	um
N Strip Height:	0.4	um
N Ground Spacing:	0.35	um
Ground Rail Height:	0.6	um
Power Ground Spacing:		um
Cell Height:	4.2	um

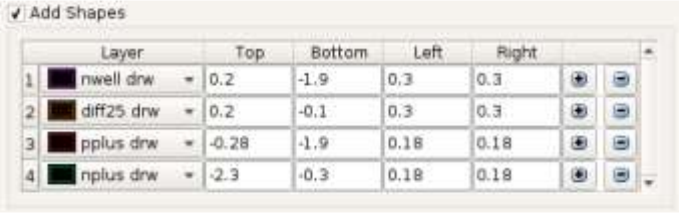
☑ Power LPP: m2 drw — Local Power Net:

☑ Ground LPP: m2 drw — Local Ground Net:

d. Enable **Add Shapes** and add *nwell*, *diff25*, *pplus*, and *nplus*. The reference edges for the enclosure values are the P/G rails.



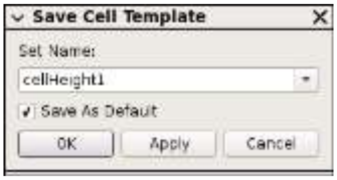
nwell *diff25* *pplus* *nplus*



☑ Add Shapes

	Layer	Top	Bottom	Left	Right
1	nwell drw	0.2	-1.9	0.3	0.3
2	diff25 drw	0.2	-0.1	0.3	0.3
3	pplus drw	-0.28	-1.9	0.18	0.18
4	nplus drw	-2.3	-0.3	0.18	0.18

239. Click **Save** and give the template a name. Enable **Save As Default** to set this template as a default for **Symbolic Editor** chaining. Click **OK** to save the cell template.

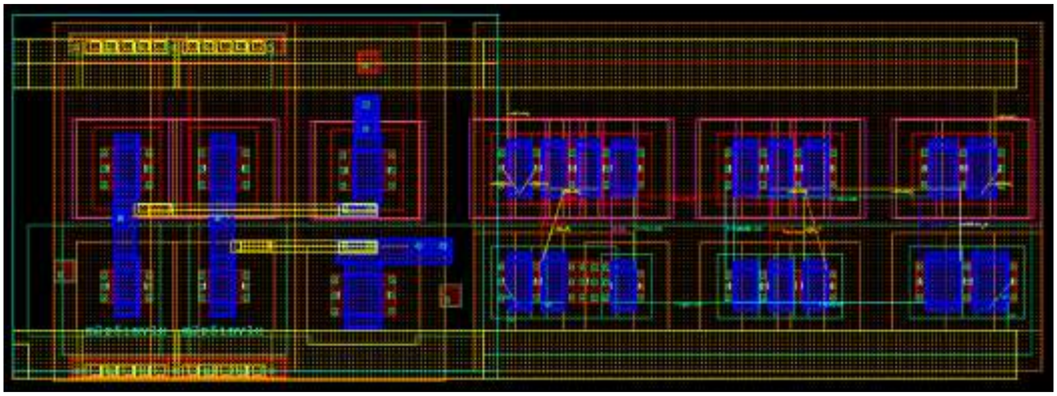


Save Cell Template

Set Name: cellHeight1

☑ Save As Default

240. Click **OK** on the **Cell Template** to apply the cell template and click **Design → Realize** to see the result in layout. Move the result to align the power/ground rail with *I139.*



Hint: you can use the **Align** command to move the whole group.

241. Save and close the design.

8

Design Creation Using Symbolic Editor

Learning Objectives

After completing this lab, you should be able to:

- Use Symbolic Editor matching mode for transistor placement

Lab Duration:
30 minutes

Introduction

The purpose of this lab is to help you understand the **Symbolic Editor** functions.

Below are the features and commands will be used:

Symbolic Editor – a device placement tool that displays key information of the devices. It makes device placement editing easier.

Design Navigator schematic mode **–** a list of devices that can be placed (realized) in layout.

Schematic Assistant – a read-only schematic docked in LE window.

Pattern Generator – Symbolic Editor function to place all devices in symbolic canvas with specific placement styles.

File Locations

All files for this lab are located in the directory **`SDL_Symbolic_Editor_Lab2`**

Directory Structure

`display.tcl`	Display attributes file
`lib.defs`	Library definition file
`snps_mpll`	Design library

Instructions

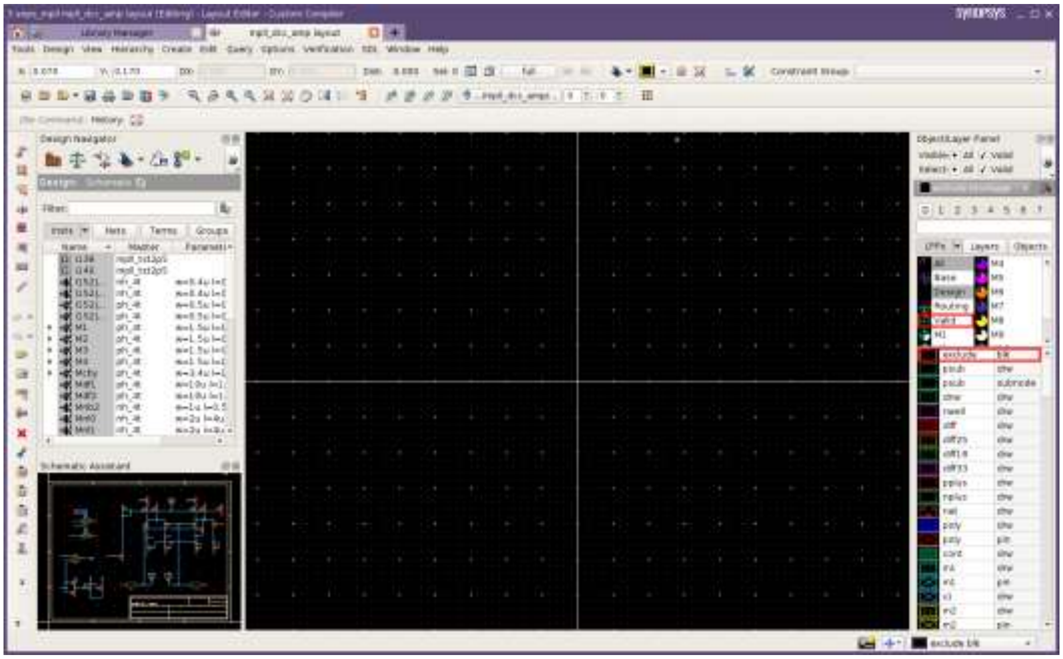
Your goal is to create an interdigitated layout using the **Symbolic Editor** matching mode.

Task 54. Start Custom Compiler and SDL

242. In the Unix window, change your working directory to **SDL_Symbolic_Editor_Lab2** and invoke Custom Compiler.

```
Unix% cd SDL_Symbolic_Editor_Lab2
Unix% custom_compiler &
```

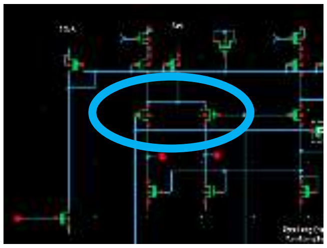
243. Open *snps_mpll/mpll_dcc_amp/layout* and you have the SDL environment.



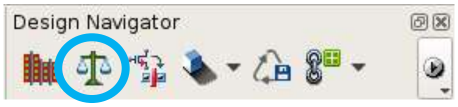
Task 55. Place Instances by Using Symbolic Editor Matching Mode

In this task, you will learn to use **Symbolic Editor** matching mode to place instances.

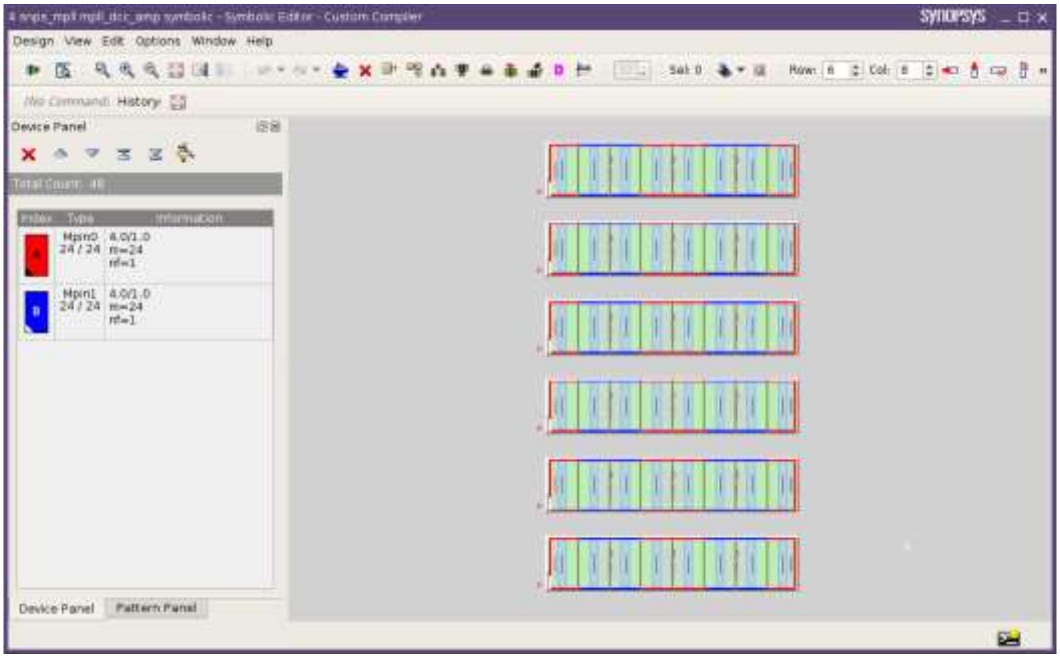
244. Select instances *Mpin0* and *Mpin1* from **Design Navigator** or **Schematic Assistant** (the differential pair in the middle).



245. Launch **Symbolic Editor** matching mode from **Design Navigator** toolbar.



246. You have **Device Panel** at left-hand side and a default placement of *Mpin0* and *Mpin1* in the symbolic canvas. Observe that for each device, **Symbolic Editor** assigns an index and a color in **Device Panel**. Here the index of *Mpin0* is *A* and the color is red. The index of *Mpin1* is *B* and the color is blue.



4 snps_mpll mpll_dcc_amp symbolic - Symbolic Editor - Custom Compiler
SYNOPSYS
Design View Edit Options Window Help
Sel: 0
Row: 6
Col: 8
(No Command) History
Device Panel
Device Panel
Pattern Panel

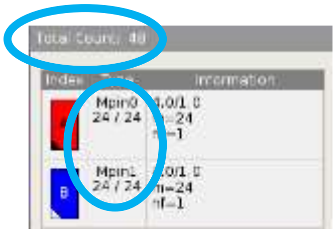
Total Count: 48

Index	Type	Information

A	Mpin0
	24 / 24
		4.0/1.0
		m=24
		nf=1

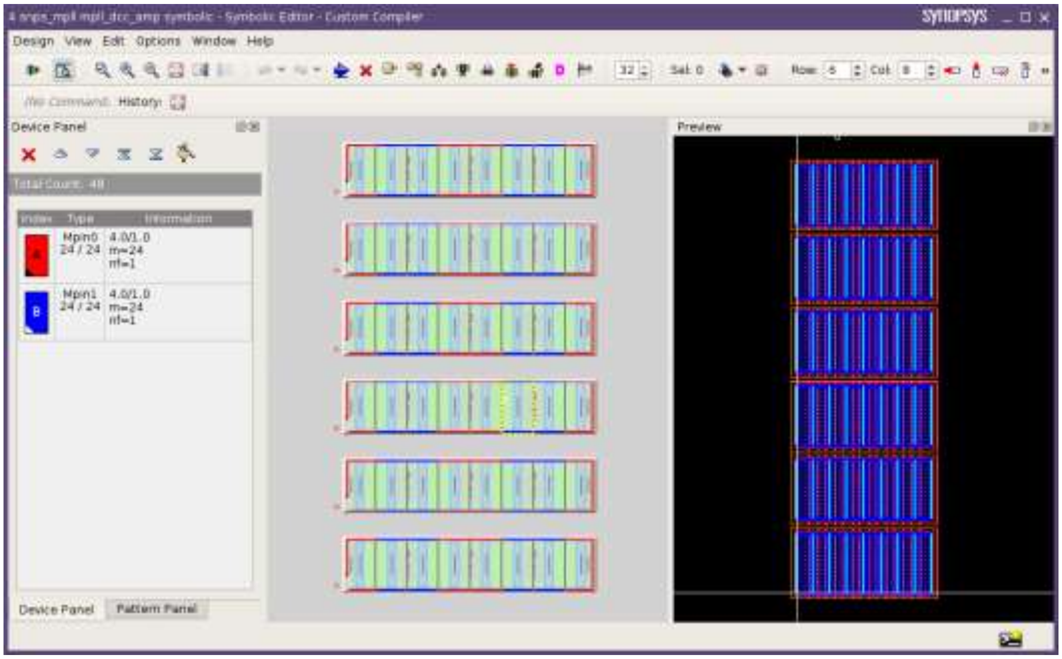
B	Mpin1
	24 / 24
		4.0/1.0
		m=24
		nf=1

247. In the **Device Panel**, observe that the total count is 48. This is because the multiplier value is 24 for both *A* and *B*.

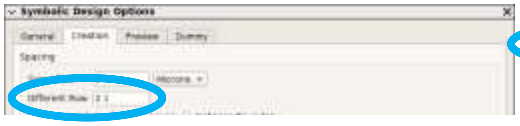


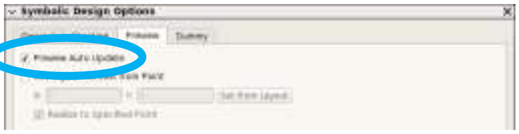
248. Also observe that in the **Type** column, 1st row shows the device name and 2nd row shows the *placed count (24) / total count (24)* for each device.

249. Select **Window → Assistants → Preview** to open the preview assistant. You can drag the preview assistant to adjust its location.

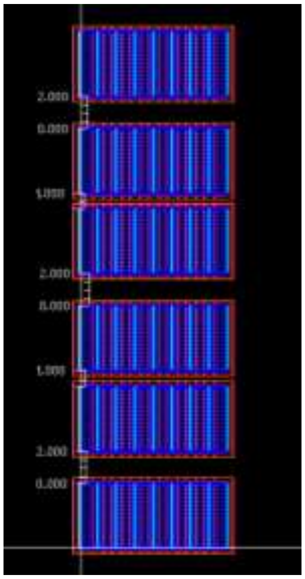

4 snps_mpl mpl_dcc_amp symbolic - Symbolic Editor - Custom Compiler
SYNOPSYS
Design View Edit Options Window Help
Row: 6 Col: 8
Device Panel
Total Count: 48
Index Type Information
A Mpin0 24 / 24 4.0/1.0 m=24 nf=1
B Mpin1 24 / 24 4.0/1.0 m=24 nf=1
Device Panel Pattern Panel
Preview

250. Select **Options → Design**. In **Creation** tab, set **Different Row** spacing to *2 1* (two values with a space in between). In Preview tab, enable **Preview Auto Update**. Click **OK**.





251. Observe that the row spacing in **Preview** is enlarged for certain tows. **Symbolic Editor** repeats the row spacing values and applies each spacing value to each row spacing. Since there are 5 row spacing, the spacing is now *2um 1um 2um 1um 2um.* You can use **Distance** command to verify that.

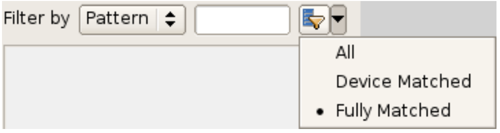


Task 56. Create a Placement – Manual Editing

In this task, you will learn to use **Pattern Panel** to create a placement.

252. Switch to **Pattern Panel** assistant (tab is at lower-left corner). Observe that no pattern is listed because **Symbolic Editor** can't find a matched build-in pattern based on the total device count (48).

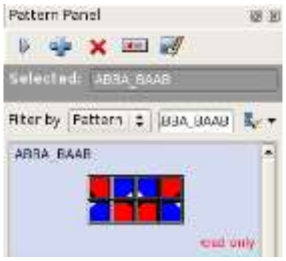
253. Expand the **Filter** pull-down and observe that current value is **Fully Matched**. **Fully Matched** means both the device count (*A* and *B*) and total count for each device (24) need to match the pattern.



Filter by Pattern

254. Change from **Fully Matched** to **Device Matched** and observe that patterns with A*B* are displayed. **Device Matched** means pattern are shown as long as device count matched (total count doesn't matter). In the **Filter** field, specify text *ABBA_BAAB* for pattern filtering.

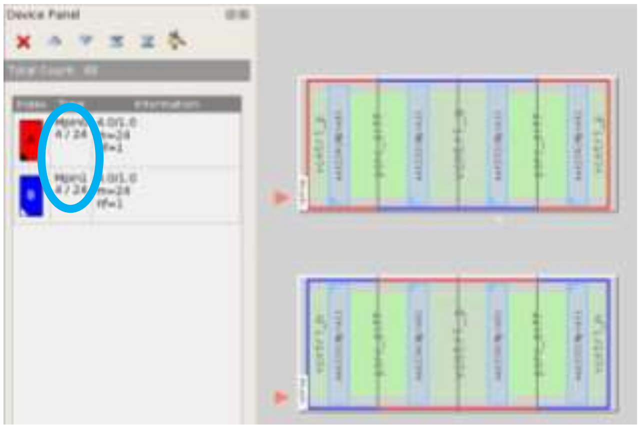
255. Double click on the 1st pattern from top, *ABBA_BAAB*, or select the pattern and click **Apply Selected Pattern** ▷ on the toolbar to apply the pattern.



Selected: ABBA_BAAB

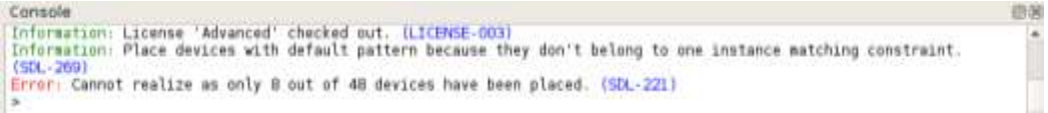
Filter by Pattern ABBA_BAAB

256. Switch the assistant to **Device Panel**. Observe that the *placed count / device count* is now *4/24* because there is only 4 *A* and 4 *B* placed in the symbolic canvas (pattern *ABBA_BAAB*).



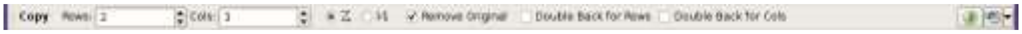
Total Count: 48

257. Select **Design → Realize** to realize the current placement to layout and observe that **Symbolic Editor** does not allow this action because **Symbolic Editor** does not support partial realization. A complete placement is required before realization.



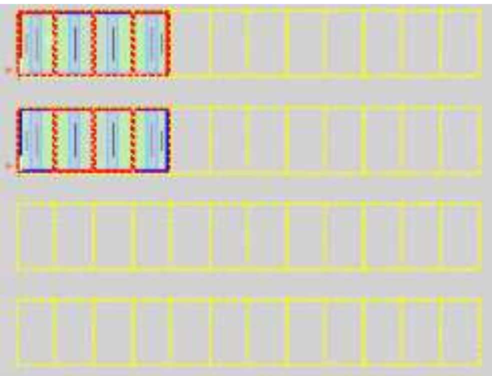
```
Console
Information: License 'Advanced' checked out. (LICENSE-003)
Information: Place devices with default pattern because they don't belong to one instance matching constraint.
(SDL-269)
Error: Cannot realize as only 8 out of 48 devices have been placed. (SDL-221)
>
```

258. Select all the devices from symbolic canvas and select **Edit → Copy**. Set **Rows** to 2, **Cols** to 3, and enable **Remove Original**.

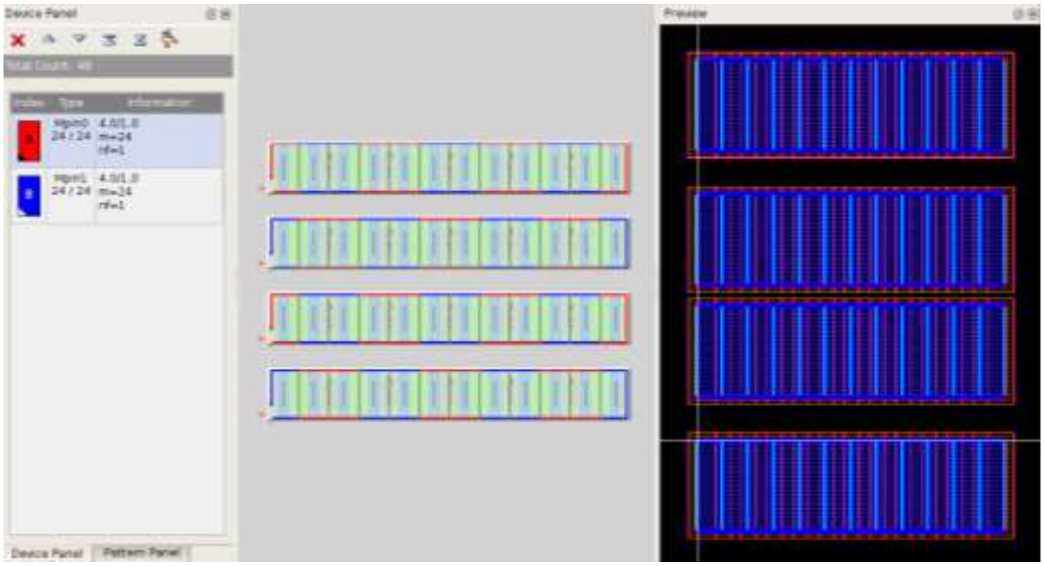


Copy Rows: 2 Cols: 3 ● Z ○ N ☑ Remove Original □ Double Back for Rows □ Double Back for Cols

259. Click in the symbolic canvas to set the anchor point for copy. Zoom out to observe expected copied result (displayed as yellow dashed lines). The red dashed lines indicate there are existing devices in the canvas and will be overwritten.



260. Click to commit the copy and overwrite all the existing devices.



Device Panel

Total Count: 48

Index	Type	Information

A	Mpm0
	24 / 24
		4.0/1.0
		m=24
		nf=1

B	Mpm1
	24 / 24
		4.0/1.0
		m=24
		nf=1

Task 57. Create a Placement – Pattern Generator

In this task, you will learn to use **Pattern Generator** to create a pattern.

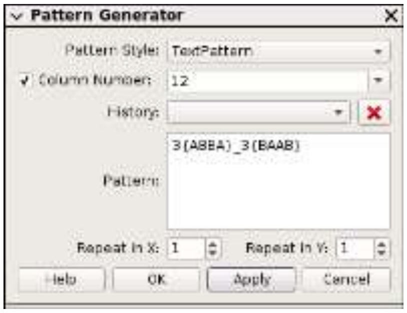
261. Delete all devices in the symbolic canvas.

262. Select **Edit → Generate Pattern**. In **Pattern Generator** dialog, set:

- **a.** **Pattern Style**: *TextPattern*
- **b.** Enable **Column Number** and set value to *12*

c. In **Pattern** field, input *3{ABBA}_3{BAAB}*

d. Click **OK**.



Pattern Style: TextPattern

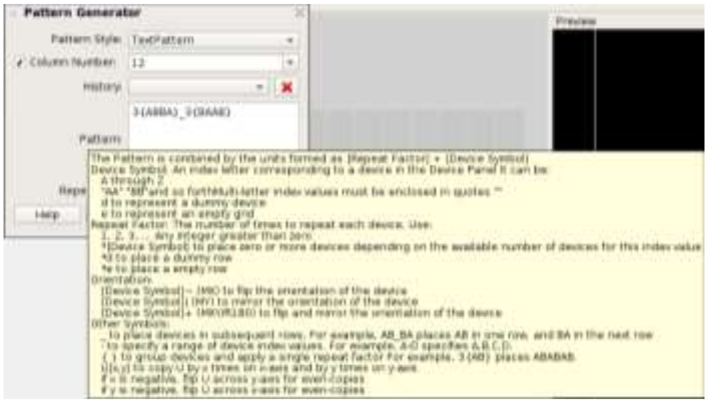
☑ Column Number: 12

History:

Pattern: 3{ABBA}_3{BAAB}

Repeat In X: 1   Repeat In Y: 1

Note: Moving the cursor on **Pattern** popups a tip of detail syntax.



Pattern Style: TextPattern

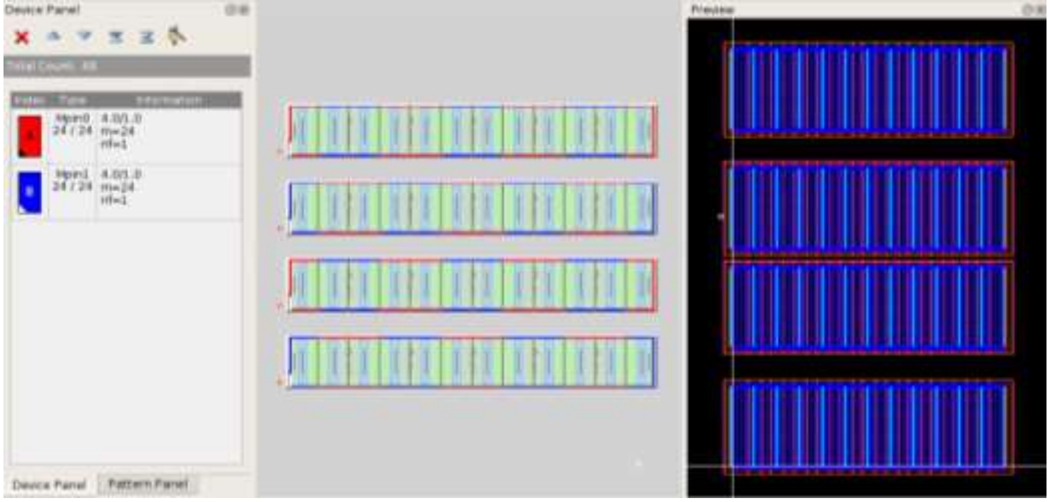
☑ Column Number: 12

History:

Pattern: 3{ABBA}_3{BAAB}

```
The Pattern is combined by the units formed as [Repeat Factor] + [Device Symbol]
Device Symbol: An index letter corresponding to a device in the Device Panel It can be:
  A through Z
  "AA" "BB"and so forth Multi-letter index values must be enclosed in quotes ""
  d to represent a dummy device
  e to represent an empty grid
Repeat Factor: The number of times to repeat each device. Use:
  1, 2, 3... Any integer greater than zero
  *[Device Symbol] to place zero or more devices depending on the available number of devices for this index value
  *d to place a dummy row
  *e to place a empty row
Orientation:
  [Device Symbol]~ (MX) to flip the orientation of the device
  [Device Symbol]| (MY) to mirror the orientation of the device
  [Device Symbol]+ (MXY/R180) to flip and mirror the orientation of the device
Other Symbols:
  _ to place devices in subsequent rows. For example, AB_BA places AB in one row, and BA in the next row
  - to specify a range of device index values. For example, A-D specifies A,B,C,D.
  { } to group devices and apply a single repeat factor For example, 3{AB} places ABABAB
  U[x,y] to copy U by x times on x-axis and by y times on y-axis
  if x is negative, flip U across y-axis for even-copies
  if y is negative, flip U across x-axis for even-copies
```

263. Observe that it gave you *ABBAABBAABBA_BAABBAABBAAB* pattern. This placement is the same with the one you manually created in previous task.



Total Count: 48

Index	Type	Information

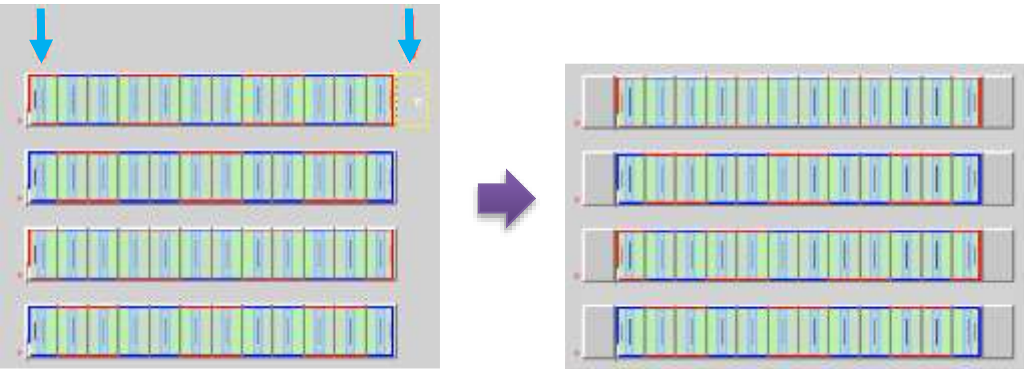
A	Mpm0
	24 / 24
		4.0/1.0
		m=24
		nf=1

B	Mpm1
	24 / 24
		4.0/1.0
		m=24
		nf=1

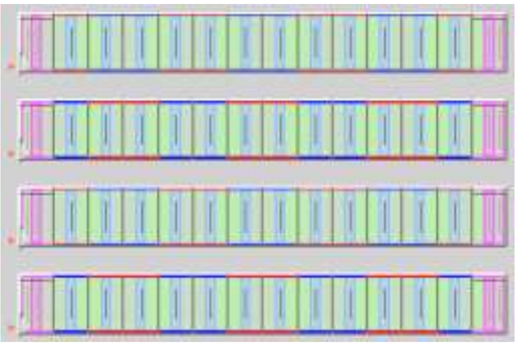
Task 58. Create Dummy and Save Pattern

In this task, you will learn to create dummies and save pattern

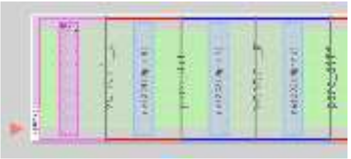
264. Select **Edit → Insert → Insert Column** and insert new columns to the last and 1st column.



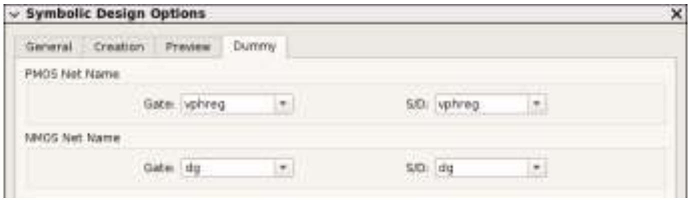
265. Select **Edit → Add Dummy** and area select the empty tiles to create dummies.



266. Observe that there is no net on the dummies except the S/D shared with devices.



267. In **Options → Design → Dummy** tab, set *gd* for **NMOS Net Name** and *vphreg* for **PMOS Net Name**. Select OK.



Symbolic Design Options

General | Creation | Preview | Dummy

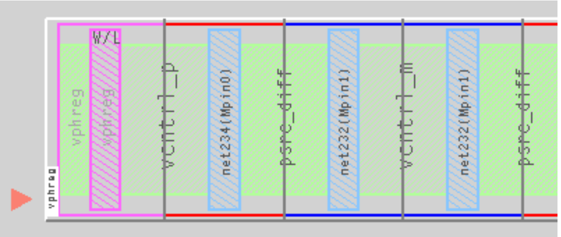
PMOS Net Name

Gate: vphreg S/D: vphreg

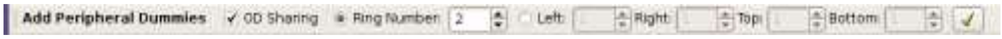
NMOS Net Name

Gate: dg S/D: dg

268. Dummies are now tied to *vphreg*.

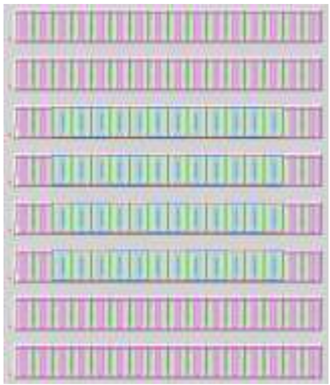


269. Select all devices without dummies and select **Edit → Add Peripheral Dummies**. Set **Ring Number** to 2 and click **Apply** ✓.

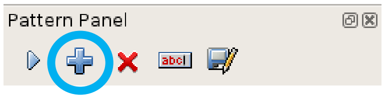


Add Peripheral Dummies ✓ OD Sharing ● Ring Number 2 ○ Left Right Top Bottom ✓

270. Observe that it creates 2 rings of dummies that surround the selected devices and previously created dummies are overwritten.

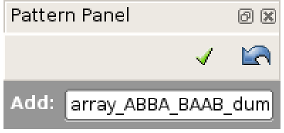


271. Switch to **Pattern Panel** and select **Save Placement as Pattern** on the **Pattern Panel** toolbar



Pattern Panel

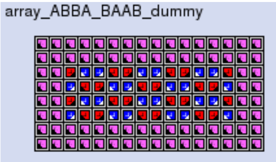
272. Give it a name (*array_ABBA_BAAB_dummy*) and select **Apply** ✓ to save the pattern.



Pattern Panel

Add: array_ABBA_BAAB_dum

273. The new pattern appears in the **Pattern Panel** and is saved in a file, *sed.pattern*, in the $SYNOPSYS_CUSTOM_LOCAL directory (default is *synopsys_custom*).



array_ABBA_BAAB_dummy

274. Select **Design → Realize** to place the device layout.

275. Close the design.

9 Block Placement Flow Tutorial

Learning Objectives

The purpose of this lab is to get you familiarized with how the placement assistant can be used to achieve block placement

After completing this lab, you should be able to perform the following tasks:

- Understand how the different placement constraints like rows,cluster and decorators like halo, symmetry, guardrings can be set
- Perform placement of multiple blocks using these block placement constraints

Lab Duration:
45 minutes

Introduction

In this lab, you are provided with block-level design with pin-only layout for most sub-blocks. You will use SDL flow start the layout and setup block placement constraints to finish a sub-block. Thereafter, for another sub-block SDL flow is used to create standard cell constraint and realized by Auto place. The top-level block will finally be placed by setting up a few block placement constraints.

Covered topics include:

- Placement Assistant

You will need to use Custom Compiler O-2018.09 version.

File Locations

All files for this lab are located in directory: **LE_Block_Placer_Lab**

Directory Structure

`/Labs/LE_Block_Placer_Lab/`	Working directory
`pllLib/`	OA design library
`stdCells/`	Sample standard cell library

Instructions

Task 59. Open the *CCO* design in Custom Compiler

In this task, you will first open a sub-block design. You will then create a new layout view for it using SDL.

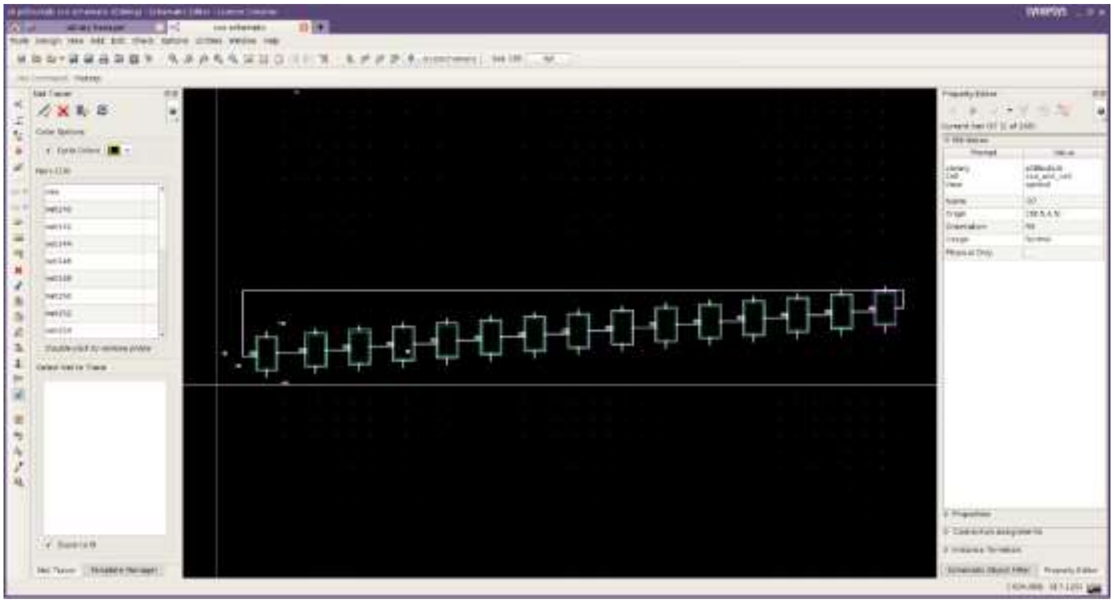
1. Start Custom Compiler from the UNIX prompt.

```
unix% cd ./Labs/LE_Block_Placer_Lab
unix% custom_compiler &    (O-2018.09 or newer)
```

2. In the home screen, click on **Library Manager** icon to bring it up

3. In **Library Manager** window, open the following cell view:

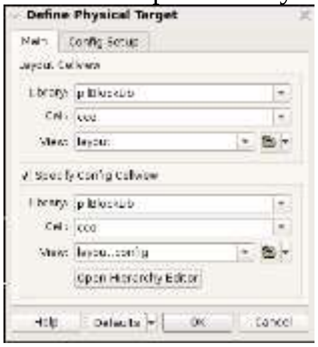
pllBlockLib /cco/schematic



4. Create new layout through SDL.

   a. **Tools → SDL**

   b. Create and open new layout by clicking on "**OK**"



Define Physical Target

Main | Config Setup

Layout Cellview

Library: pllBlockLib

Cell: vco

View: layout

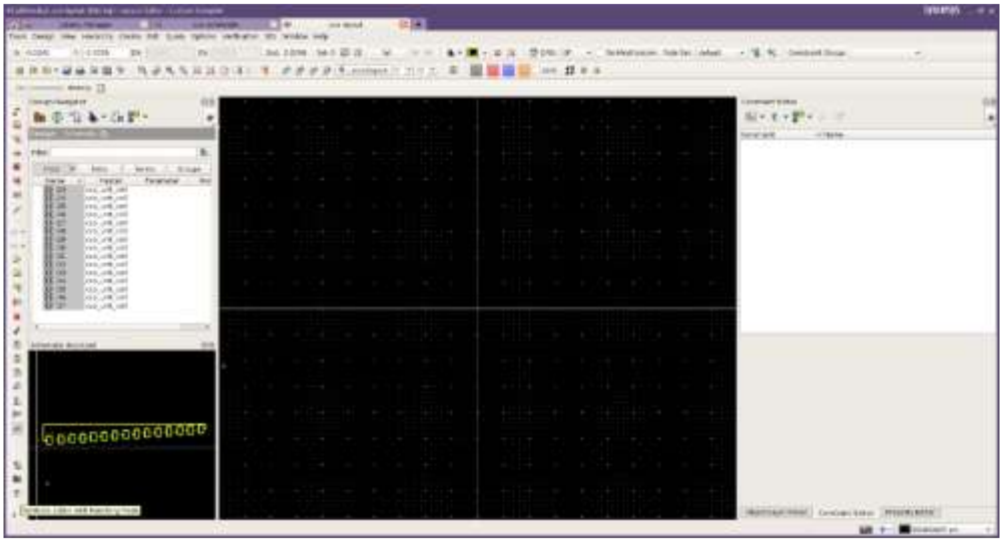
☑ Specify Config Cellview

Library: pllBlockLib

Cell: vco

View: layout_config

An empty layout shall be created and opened.

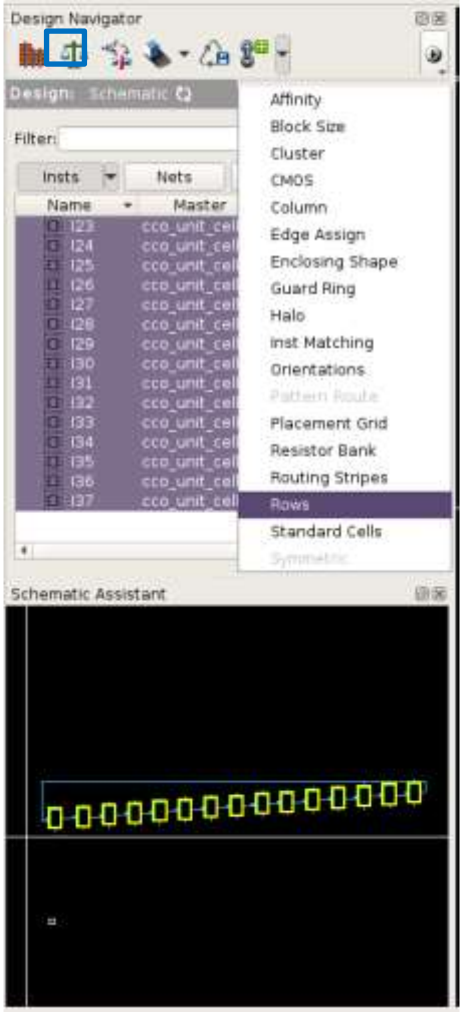


Task 60. Creating placement constraint

In this task, you will use **Design Navigator** to setup placement constraints to realize a multi-row placement with guard ring.

5. Select All instances in the design, from **Schematic Assistant** or **Design Navigator**

6. Click Constraint Menu to list the placement constraints.



Name	Master
I23	cco_unit_cel
I24	cco_unit_cel
I25	cco_unit_cel
I26	cco_unit_cel
I27	cco_unit_cel
I28	cco_unit_cel
I29	cco_unit_cel
I30	cco_unit_cel
I31	cco_unit_cel
I32	cco_unit_cel
I33	cco_unit_cel
I34	cco_unit_cel
I35	cco_unit_cel
I36	cco_unit_cel
I37	cco_unit_cel

The Constraint Menu greys out constraints that cannot be applied for selected set of instances.

Assuming a more balanced placement pattern is preferred.

7. Select the **Rows** constraint.

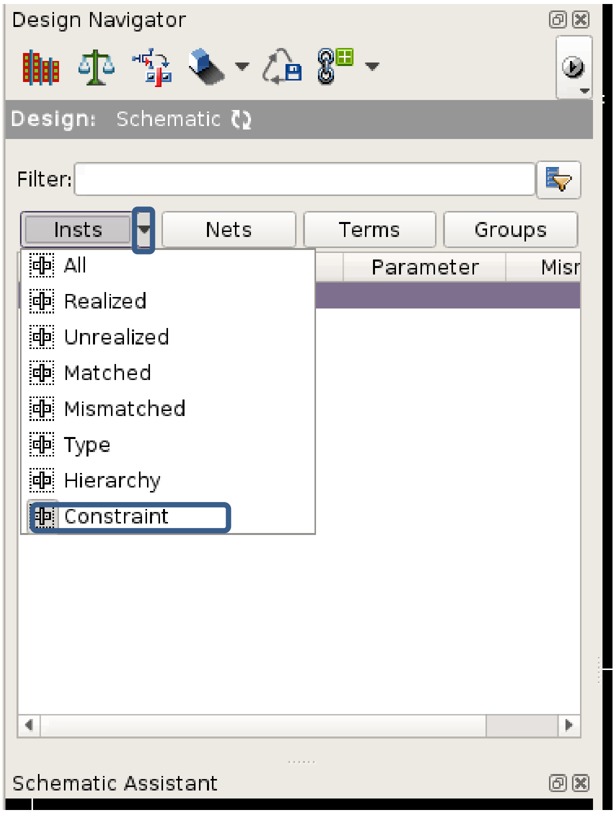
8. View the constraint in Constraint Editor

a. Open the Constraint Editor from **Window** → **Assistants** → **Constraint Editor**

b. Click the **Rows** constraint to review the supported options

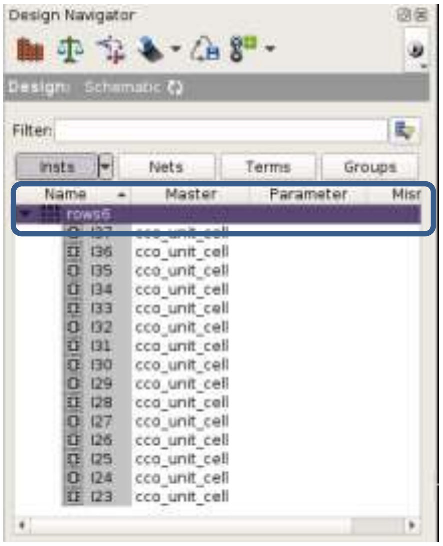
9. Now, we will set a Guard Ring constraint around the whole set.

a. Change Instance View in **Design Navigator** to **Constraint** by Clicking the Menu button besides **Insts** button and selecting **Constraint**



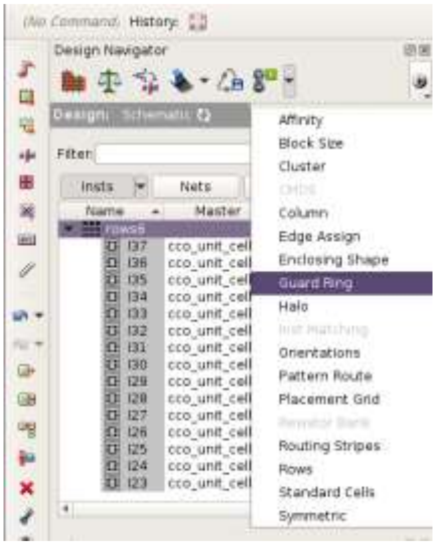
b. Expand the **rows<n>** constraint group.

c. Now, select **only** the **rows<n>** line.

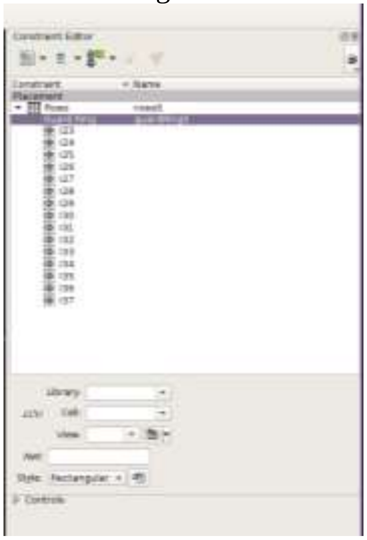


Name	Master	Parameter	Misr
rows6			
I37	cco_unit_cell		
I36	cco_unit_cell		
I35	cco_unit_cell		
I34	cco_unit_cell		
I33	cco_unit_cell		
I32	cco_unit_cell		
I31	cco_unit_cell		
I30	cco_unit_cell		
I29	cco_unit_cell		
I28	cco_unit_cell		
I27	cco_unit_cell		
I26	cco_unit_cell		
I25	cco_unit_cell		
I24	cco_unit_cell		
I23	cco_unit_cell		

d. Click the Constraint Menu on **Design Navigator** toolbar and select **Guard Ring**



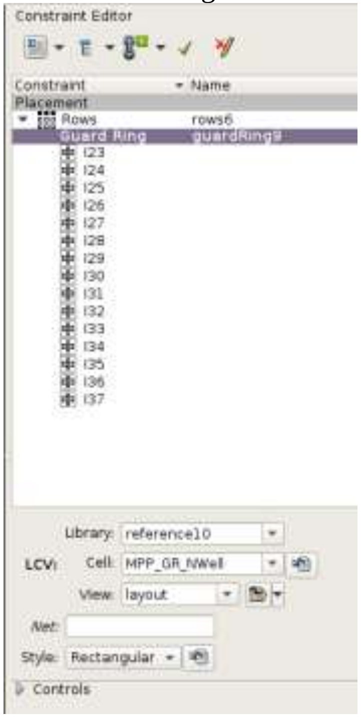
e. A Guard Ring Constraint is created in **Constraint Editor**



f. Select the desired Guard Ring by browsing the LCV button

g. Select Library – reference10; Cell – MPP_GR_NWell

h. Click Apply ✓ to save the changes

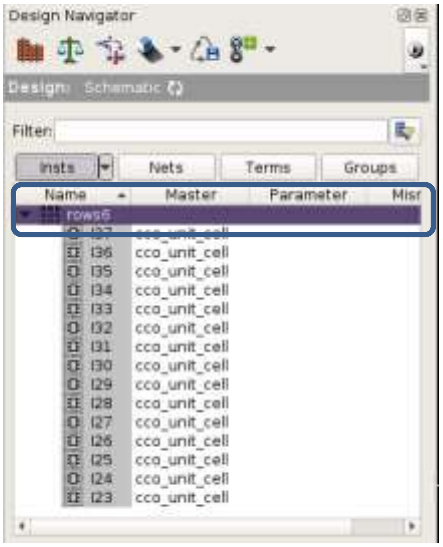


Constraint	Name
Placement	
Rows	rows6
Guard Ring	guardRing9
I23	
I24	
I25	
I26	
I27	
I28	
I29	
I30	
I31	
I32	
I33	
I34	
I35	
I36	
I37	

LCV: Library: reference10
Cell: MPP_GR_NWell
View: layout
Net:
Style: Rectangular

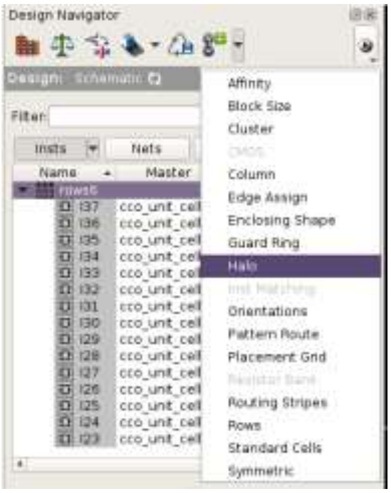
10. Next, we will add a halo to the complete group. This will ensure the pins are created at a pre-determined spacing from the guard ring protected group.

a. Again, select **only** the **rows<n>** line.

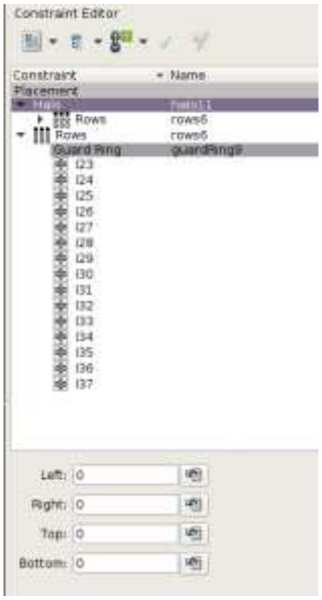


Name	Master	Parameter	Misr
rows6			
I37	cco_unit_cell		
I36	cco_unit_cell		
I35	cco_unit_cell		
I34	cco_unit_cell		
I33	cco_unit_cell		
I32	cco_unit_cell		
I31	cco_unit_cell		
I30	cco_unit_cell		
I29	cco_unit_cell		
I28	cco_unit_cell		
I27	cco_unit_cell		
I26	cco_unit_cell		
I25	cco_unit_cell		
I24	cco_unit_cell		
I23	cco_unit_cell		

b. Click the Constraint Menu on **Design Navigator** toolbar and select **Halo**



c. A Halo Constraint is created in **Constraint Editor**



Left: 0

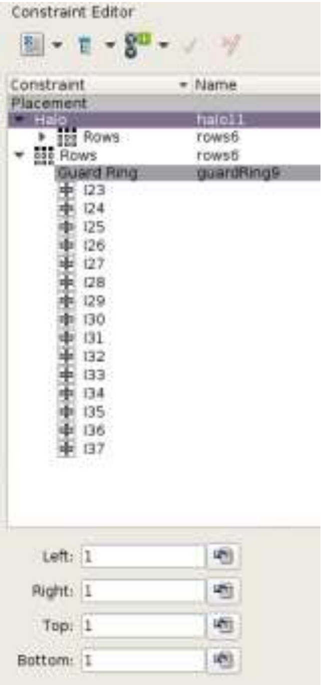
Right: 0

Top: 0

Bottom: 0

d. Edit the Halo Constraint in the **Constraint Editor.** Set the Left, Right, Top and Bottom values to 1

e. Click Apply ✓ to save the changes



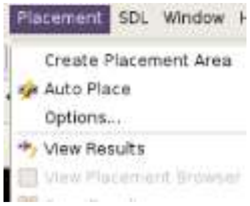
Constraint	Name
Placement	
Halo	halo11
Rows	rows6
Rows	rows6
Guard Ring	guardRing9
I23	
I24	
I25	
I26	
I27	
I28	
I29	
I30	
I31	
I32	
I33	
I34	
I35	
I36	
I37	

Left: 1
Right: 1
Top: 1
Bottom: 1

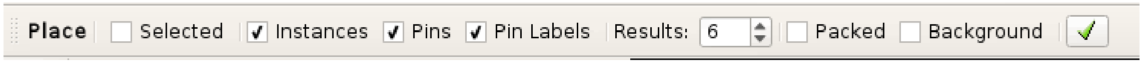
Task 61. Run Placement on CCO Design

In this task, you will complete the placement using **Placement Assistant**.

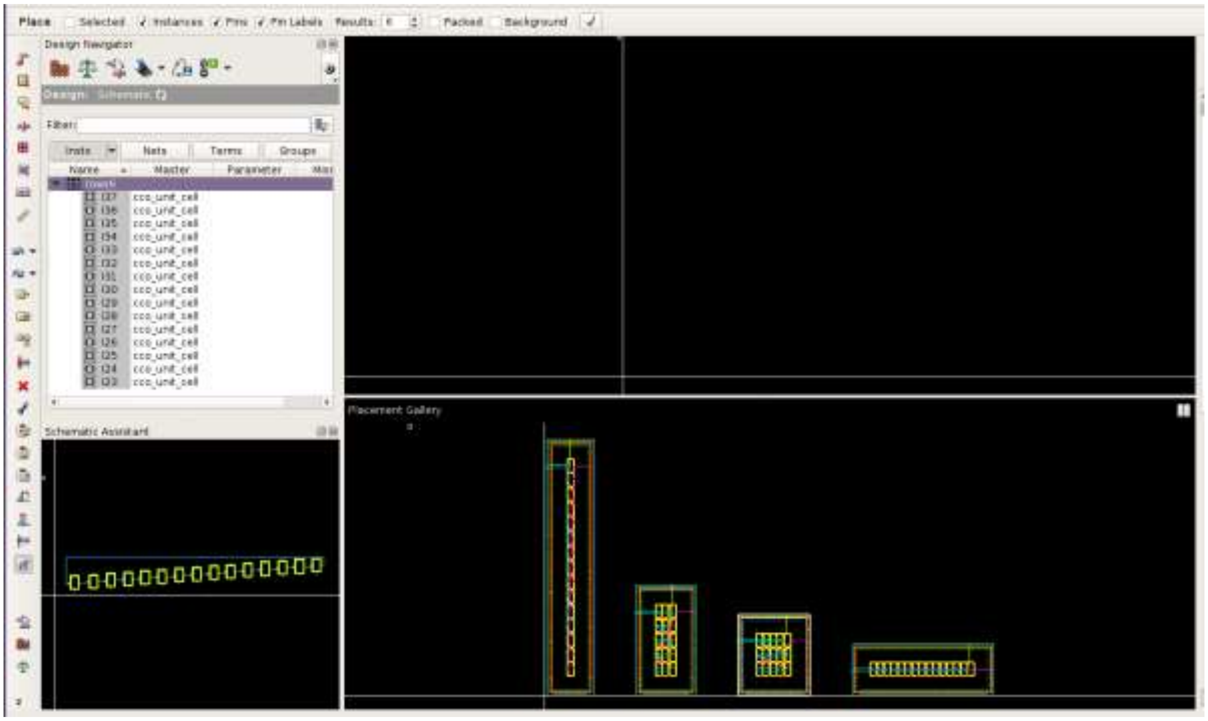
1. Enable the Placement Assistant Menu from **Tools → Placement**
2. Run placement using **Placement Assistant**.
   a. **Placement → Auto Place**



   b. Ensure that the **Selected** box is unchecked
   c. Check the Instances, Pins and Pin Labels boxes


Place □ Selected ☑ Instances ☑ Pins ☑ Pin Labels Results: 6 □ Packed □ Background ✓

   d. Click on ✓ in the **Place** toolbar proceed with the placement run.
3. Upon completion, the **Placement Gallery** will open below the layout canvas, which will contain **four** results.



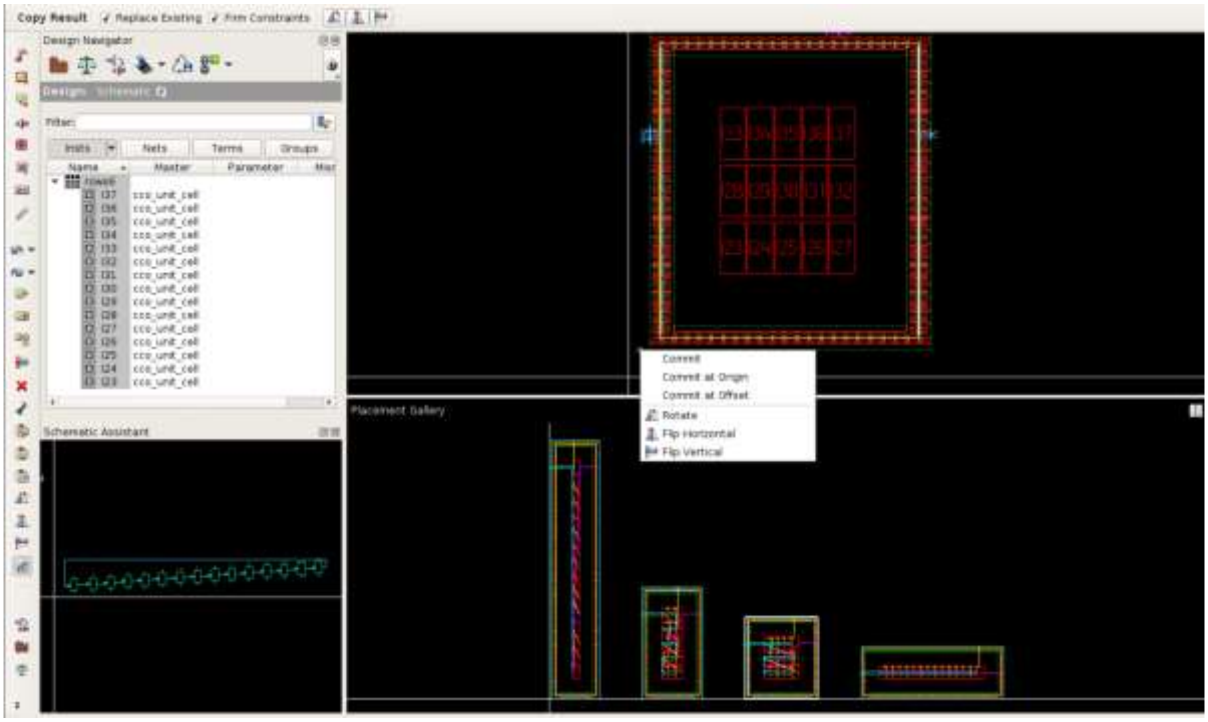
4. On observing the placement results, it can be observed that the solutions range from a 15-row to a single-row solution.

5. Place the result by picking from the gallery and copying them into the layout.

   a. Invoke **Copy Result** command.

      Placement → Copy Result (or use bindkey "**C**" when the cursor is in the gallery)

   b. Select the third result in Placement Gallery



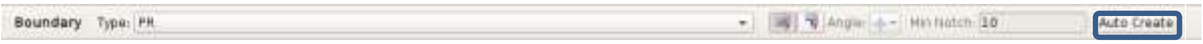
c. Move the cursor into the layout canvas and Right-click

d. Select Commit at Origin

6. Create a boundary for the cell to complete the sub-block

   a. Use **Create → Boundary** to bring up the **Boundary** toolbar

   b. Click **Auto Create** to create a boundary around the layout.



Boundary Type: PR

Angle

Min Notch: 10

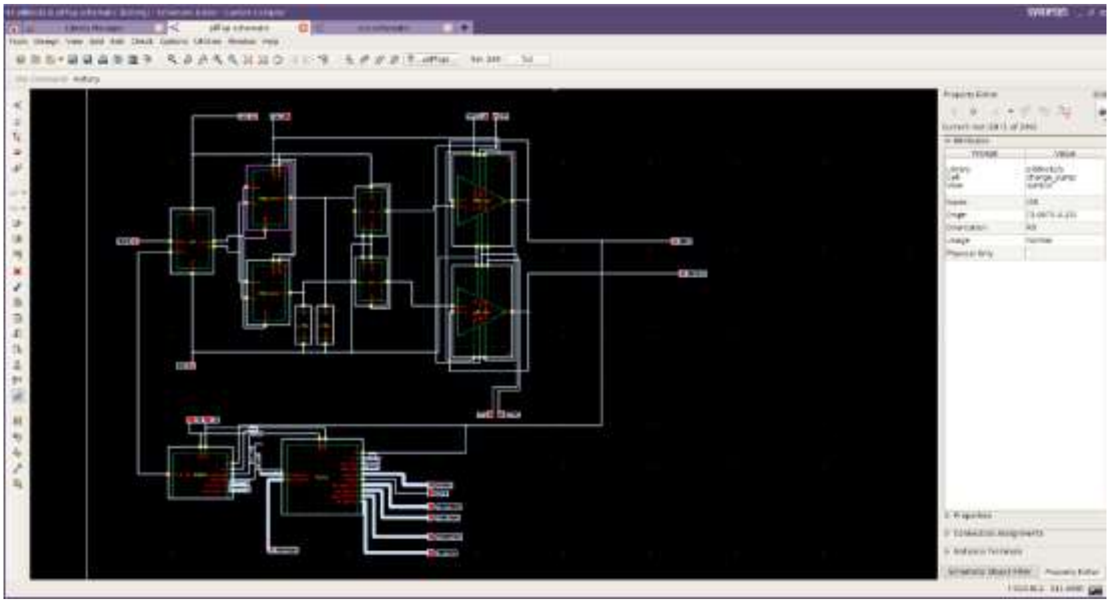
7. Save the design (**Design → Save)** and close the design (**Ctrl + W**)

Task 62. Open the *pllTop* design in Custom Compiler

In this task, you will first open top-level design. You will then create a new layout view for it using SDL.

8. In **Library Manager** window, open the following cell view:

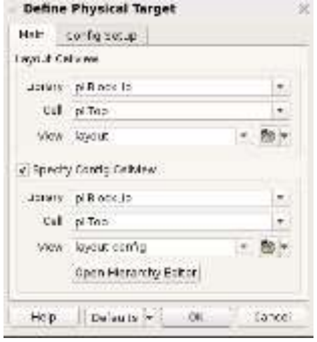
   pllBlockLib/pllTop/schematic



9. Create new layout through SDL.

   a. **Tools → SDL**

   b. Create and open new layout by clicking on “**OK**”



Define Physical Target

Main | config setup

Layout Cellview

Library: pll_Block_lb

Cell: pll_Top

View: layout

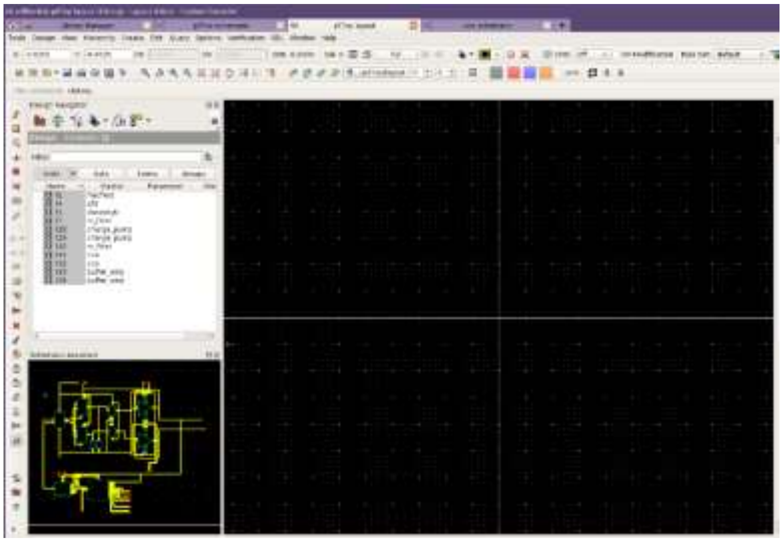
☑ Specify Config Cellview

Library: pll_Block_lb

Cell: pll_Top

View: layout_config

An empty layout shall be created and opened.

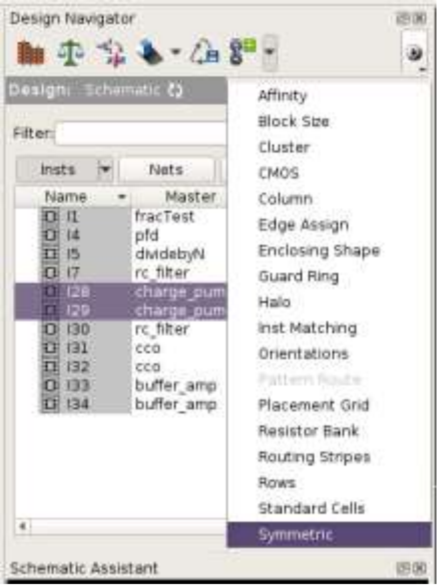


Task 63. Creating placement constraint

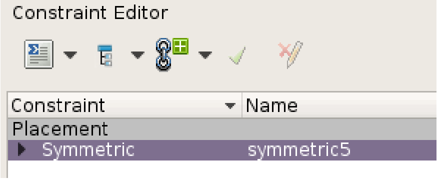
In this task, we look at the appropriate block constraints to get the desired layout. As can be observed there are four blocks **rc_filter (I7, I30), charge_pump (I28, I29), buffer_amp (I33, I34) and cco (I31, I32)** which are present in pairs. This design has differential signaling and so needs to have these blocks symmetric. The **pfd** block (**I4**) needs to feed these differential paths and so should be centre aligned with the symmetric blocks.

10. We will start with adding **Symmetric** decorator to each of the pairs listed above.

- **a.** Select the **charge_pump** instances (I28, I29) in **Design Navigator**
- **b.** Click Constraint Menu and Select **Symmetric**
- **c.** Repeat **a-b** steps for **rc_filter (I7, I30), buffer_amp (I33, I34) and cco (I31, I32)**

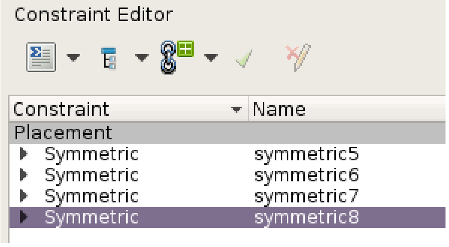


Name	Master
I1	fracTest
I4	pfd
I5	dividebyN
I7	rc_filter
I28	charge_pum
I29	charge_pum
I30	rc_filter
I31	cco
I32	cco
I33	buffer_amp
I34	buffer_amp



Constraint	Name
Placement	
▸ Symmetric	symmetric5

- **d.** Four **Symmetric** decorators are created in **Constraint Editor**

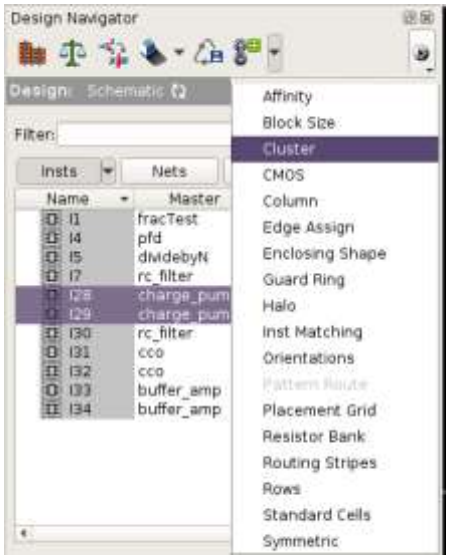


Constraint	Name
Placement	
▸ Symmetric	symmetric5
▸ Symmetric	symmetric6
▸ Symmetric	symmetric7
▸ Symmetric	symmetric8

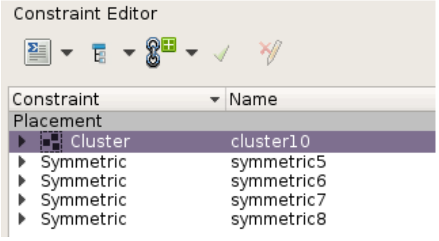
11. Next, we will add cluster constraint to enable the realization of the four symmetric sets in a group

a. Select the **charge_pump** instances (I28, I29) in **Design Navigator**

b. Click Constraint Menu and Select **Cluster.** A Cluster constraint will reflect in the Constraint Editor as well.



Name	Master
I1	fracTest
I4	pfd
I5	dividebyN
I7	rc_filter
I28	charge_pum
I29	charge_pum
I30	rc_filter
I31	cco
I32	cco
I33	buffer_amp
I34	buffer_amp

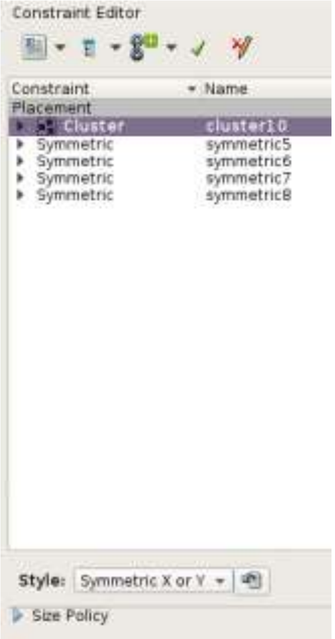


Constraint	Name
Placement	
Cluster	cluster10
Symmetric	symmetric5
Symmetric	symmetric6
Symmetric	symmetric7
Symmetric	symmetric8

c. Click the **Cluster** constraint in Constraint Editor.

d. Change the **Style** to **Symmetric X or Y**

e. Click Apply ✓ to save the changes to constraint

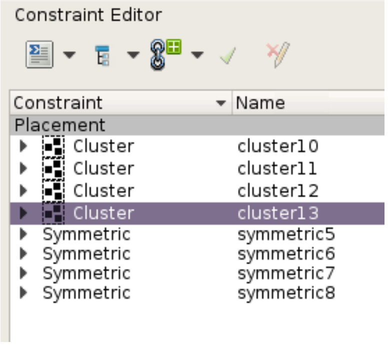


Constraint	Name
Placement	
Cluster	cluster10
Symmetric	symmetric5
Symmetric	symmetric6
Symmetric	symmetric7
Symmetric	symmetric8

Style: Symmetric X or Y

f. Repeat **a-e** steps for **rc_filter (I7, I30), buffer_amp (I33, I34) and cco (I31, I32)**

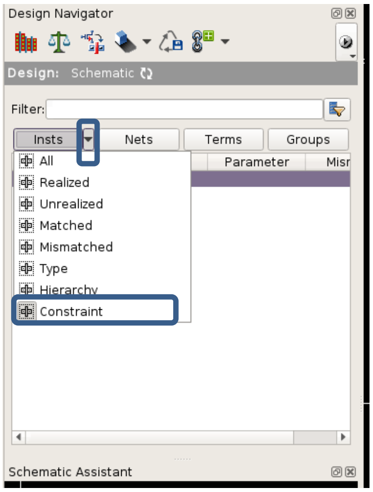
g. Four **Cluster** constraints are added in the Constraint Editor



Constraint	Name
Placement	
Cluster	cluster10
Cluster	cluster11
Cluster	cluster12
Cluster	cluster13
Symmetric	symmetric5
Symmetric	symmetric6
Symmetric	symmetric7
Symmetric	symmetric8

12. Next to get an aligned placement of these four pairs with the **pfd** block, we can set a **Rows** constraint. As seen in Task 3, The **Rows** constraint can provide column-wise and row-wise results in **Placement Gallery.**

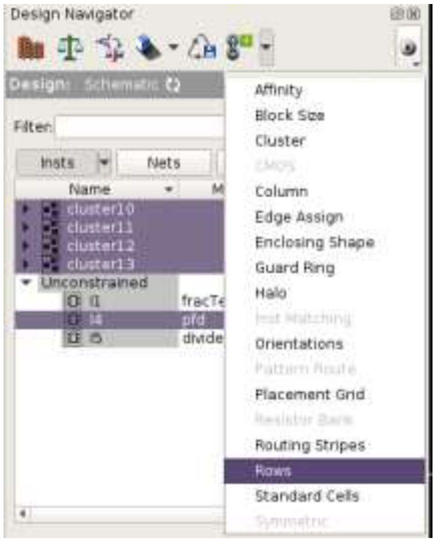
   a. Change Instance View in **Design Navigator** to **Constraint** by Clicking the Menu button besides **Insts** button and selecting **Constraint**

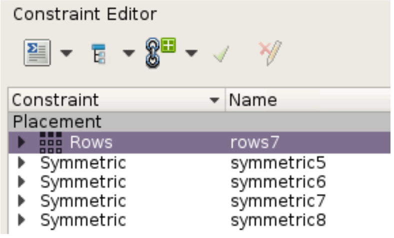


   b. Expand the **Unconstrained** row.

   c. Select **I4** and the four **Cluster<n>** groups (Press **ctrl** to allow multiple selection

   d. Click the Constraint Menu on **Design Navigator** toolbar and select **Rows**




Constraint Editor

Constraint	Name
Placement	
Rows	rows7
Symmetric	symmetric5
Symmetric	symmetric6
Symmetric	symmetric7
Symmetric	symmetric8

e. A **Rows** Constraint is created in **Constraint Editor**

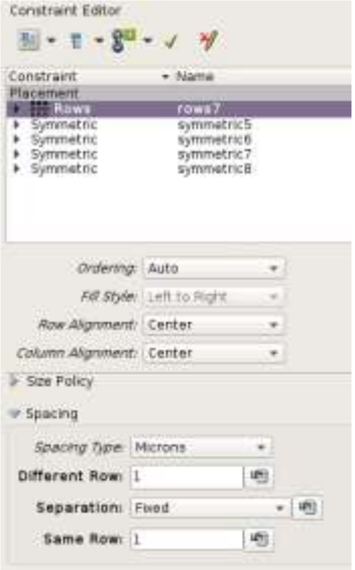
f. Click the Rows constraint in **Constraint Editor.**

g. Expand the **Spacing** Options of the constraint.

h. Set the **Different Row** spacing to 1

i. Set the **Separation** to **Fixed**

j. Set the **Same Row** spacing to 1


Constraint Editor

Constraint	Name
Placement	
Rows	rows7
Symmetric	symmetric5
Symmetric	symmetric6
Symmetric	symmetric7
Symmetric	symmetric8

Ordering: Auto
Fill Style: Left to Right
Row Alignment: Center
Column Alignment: Center
Size Policy
Spacing
Spacing Type: Microns
Different Row: 1
Separation: Fixed
Same Row: 1

k. Click ✓ to save the changes to the constraint in **Constraint Editor**.

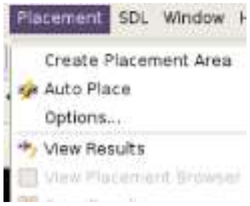
Task 64. Run Placement on Top Design

In this task, you will run placement on the top design and analyze the effect of the cosntraints set in the previous task.

1. Enable the Placement Assistant Menu from **Tools → Placement**

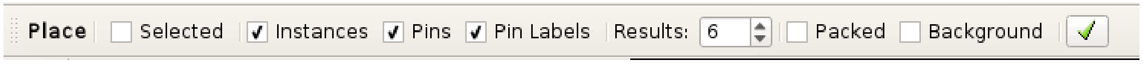
2. Run placement using **Placement Assistant**.

   a. **Placement → Auto Place**



   b. Ensure that the **Selected** box is unchecked

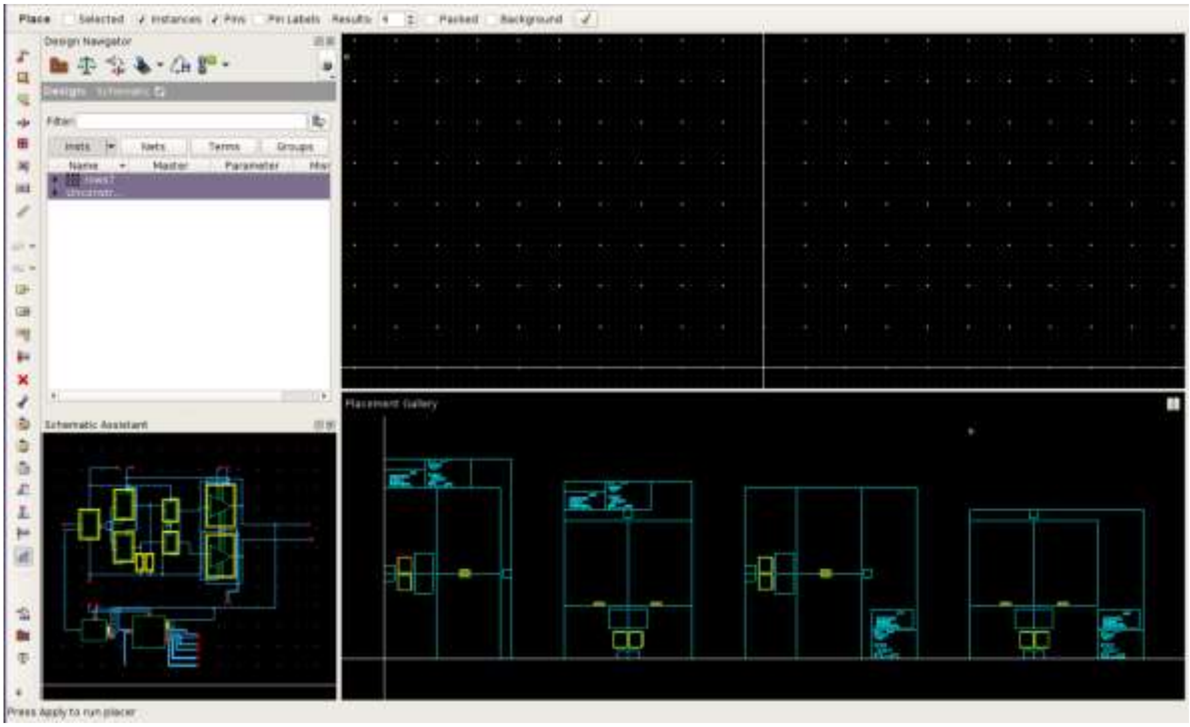
   c. Check the Instances, Pins and Pin Labels boxes



Place □ Selected ☑ Instances ☑ Pins ☑ Pin Labels Results: 6 □ Packed □ Background ✓

   d. Click on ✓ in the **Place** toolbar proceed with the placement run.

3. Upon completion, the **Placement Gallery** will open below the layout canvas, which will contain **four** results.



Press Apply to run placer

4. Click on the **rows<n>** in **Design Navigator.**

5. The realizations can be observed for the symmetric layout. Also, **Vertical and Horizontal** realizations are provided in Gallery results

6. Place the result by picking from the gallery and copying them into the layout.

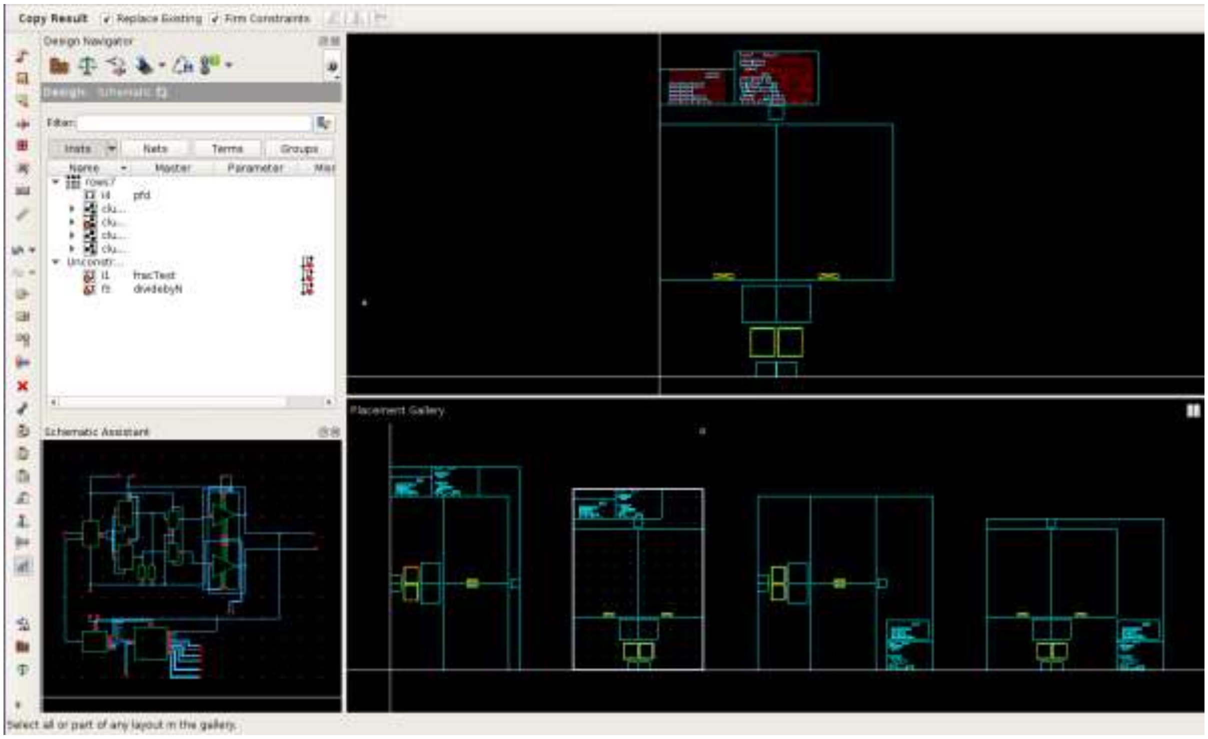
   a. Invoke **Copy Result** command.

Placement → Copy Result (or use bindkey "**C**" when the cursor is in the gallery)

b. Select the third result in Placement Gallery

c. Move the cursor into the layout canvas and Right-click

d. Select Commit at Origin



Select all or part of any layout in the gallery.