cadence

Introduction to Innovus and Block Implementation Flow Rapid Adoption Kit (RAK)

Innovus Version 18.1

June-2018

Note: RAK Testcase database, Scripts and references can be found at ‘Attachments’ and ‘Related Solutions’ sections below the PDF.

Cadence Design Systems, Inc., 2655 Seely Avenue, San Jose, CA 95134, USA

Cadence Trademarks

Trademarks and service marks of Cadence Design Systems, Inc. (Cadence) contained in this document are attributed to Cadence with the appropriate symbol. For queries regarding Cadence trademarks, contact the corporate legal department at the address above or call 800.862.4522.

Allegro®	Incisive®	Silicon Express™
Accelerating Mixed Signal Design®	InstallScape™	SKILL®
Assura®	IP Gallery™	SoC Encounter™
BuildGates®	NanoRoute®	SourceLink® online customer support
Cadence®	NC-Verilog®	Specman®
CeltIC®	NeoCell®	Spectre®
Conformal®	NeoCircuit®	Speed Bridge®
Connections®	OpenBook® online documentation library	UltraSim®
Diva®	OrCAD®	Verifault-XL®
Dracula®	Palladium®	Verification Advisor®
ElectronStorm®	Pearl®	Verilog®
Encounter®	PowerSuite®	Virtuoso®
EU CAD®	PSpice®	VoltageStorm®
Fire & Ice®	SignalStorm®	Xtreme®
First Encounter®	Silicon Design Chain™	
HDL-ICE®	Silicon Ensemble®	

Other Trademarks

Open SystemC, Open SystemC Initiative, OSCI, SystemC, and SystemC Initiative are trademarks or registered trademarks of Open SystemC Initiative, Inc. in the United States and other countries and are used with permission.

All other trademarks are the property of their respective holders.

Confidentiality Notice

Contents

Module 1: Introduction and Setup

1-1 Introduction

The block implementation Rapid Adoption Kit (RAK) introduces you to Innovus and walks you through the basic steps in the implementation flow. It is recommended for users who are new to Innovus or the implementation flow. This is a flat implementation flow which can be applied to chip level designs as well as blocks.

The goal of this tutorial is to provide you a small example of using the Innovus software. It is very basic by design so we highly recommend users attend one of several Innovus training classes provided by Cadence Educational Services. For more information on available training please visit www.cadence.com and click *Services->Training*.

In addition to this tutorial the following resources are recommended to continue your Innovus learning.
Online documentation is available via https://support.cadence.com
Resources ➔ Product manuals ➔ Innovus18.1

- Innovus 18.1 User Guide
- Innovus 18.1 Text Command Reference
- Innovus 18.1 Menu Reference

Testcase database, Scripts and references can be found at 'Attachments' and 'Related Solutions' sections below the PDF.
This pdf can be searched with the document 'Title:
Introduction to Innovus Implementation System 18.1 & Block Implementation Flow
on https://support.cadence.com

1-2 Leon Design Information

The design in this workshop is a Leon processor. The Leon design is a block level design with 35K instances, 4 memories, and 1200 IO pins. The library used is a Cadence Generic 45nm library using 9 routing layers.

1-3 Setting up Innovus and the Lab Directory

1. Download and install the Innovus 18.1 software from https://downloads.cadence.com. The recommended version for this lab is Innovus 18.10-p002_1.

2. Extract the RAK database and change directory to the work directory:

```
linux% tar xfz RAK_18.1_blockImplementation.tar.gz
linux% cd RAK_18.1_blockImplementation
```

3. Verify the `innovus` executable is in your path by typing:

```
linux% which innovus
```

4. Start Innovus:

```
linux% innovus
```

Innovus creates three files for storing commands and their output. Each of these files will have a number at the end of the name which is incremented with each session.

- **innovus.cmd** – Contains list of commands executed during the session. This file can be used to create scripts to automate the execution of the commands and learn what text command correspond to commands executed through the GUI.
- **innovus.log** – Contains basic information output from the executed commands. The commands in the file are preceded with `<CMD>` in the file.
- **innovus.logv** – Similar to innovus.log but contains more verbose amount of output. Useful for debugging.

Tips:

- Overwrite the default names by using the `-cmd` and `-log` options when executing `innovus`.
- Use `innovus -init scriptName` to execute a script when invoking innovus. Type `win` to open the GUI when the script completes.
- Use `innovus -nowin` to invoke Innovus in non-GUI mode.

The Linux shell Innovus is invoked from becomes the console where standard output is printed, and is also where you enter text commands:

```
Cadence Innovus(TM) Implementation System.
Copyright 2018 Cadence Design Systems, Inc. All rights reserved worldwide.

Version:        v18.10-p002_1, built Tue May 29 19:19:55 PDT 2018
Options:        -log logfiles/final.log
Date:           Thu Jun 21 16:30:29 2018
Host:           ccslinux31 (x86_64 w/Linux 2.6.18-194.el5) (6cores*12cpus*Intel(R) Xeon(R) CPU X5680 @ 3.33GHz 12288KB)
OS:             Red Hat Enterprise Linux Client release 5.5 (Tikanga)

License:
                invs    Innovus Implementation System   18.1    checkout succeeded
                8 CPU jobs allowed with the current license(s). Use setMultiCpuUsage to set your required CPU count.
Create and set the environment variable TMPDIR to /tmp/innovus_temp_8897_ccslinux31_msai_CGhhW7.

The soft stacksize limit is either up to the hard limit or larger than 0.2%RAM. No change is needed.
Sourcing startup file /home/msai/.cadence/innovus/gui.color.tcl
Sourcing startup file ./enc.tcl

**INFO:  MMMC transition support version v31-84
```

Tip: When entering text commands you can utilize up/down arrows to cycle through the command history as well as tab completion to complete commands and their options to save on typing.

Module 2: Getting to Know Innovus

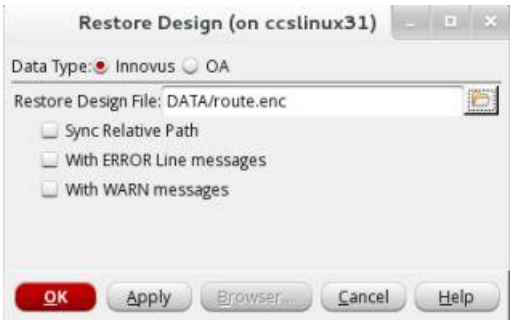
2-1 Opening a Design

Start by opening an existing design which is used to introduce you to Innovus.

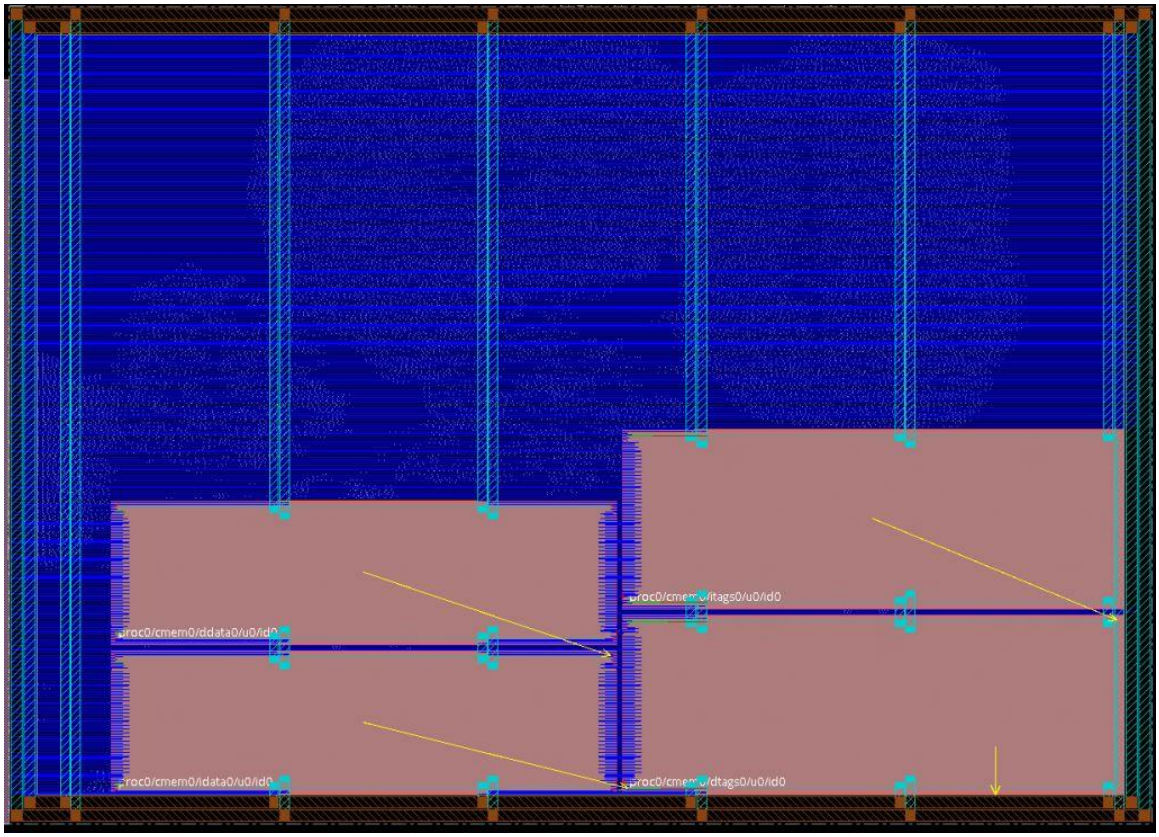
1. Select **File – Restore Design**.

2. On the **Restore Design** form:

    - Select **Data Type:** *Innovus*
    - Specify **Design Restore File:** *DATA/route.enc*. You can type it in or use the file browser button next to the field to navigate to the file.

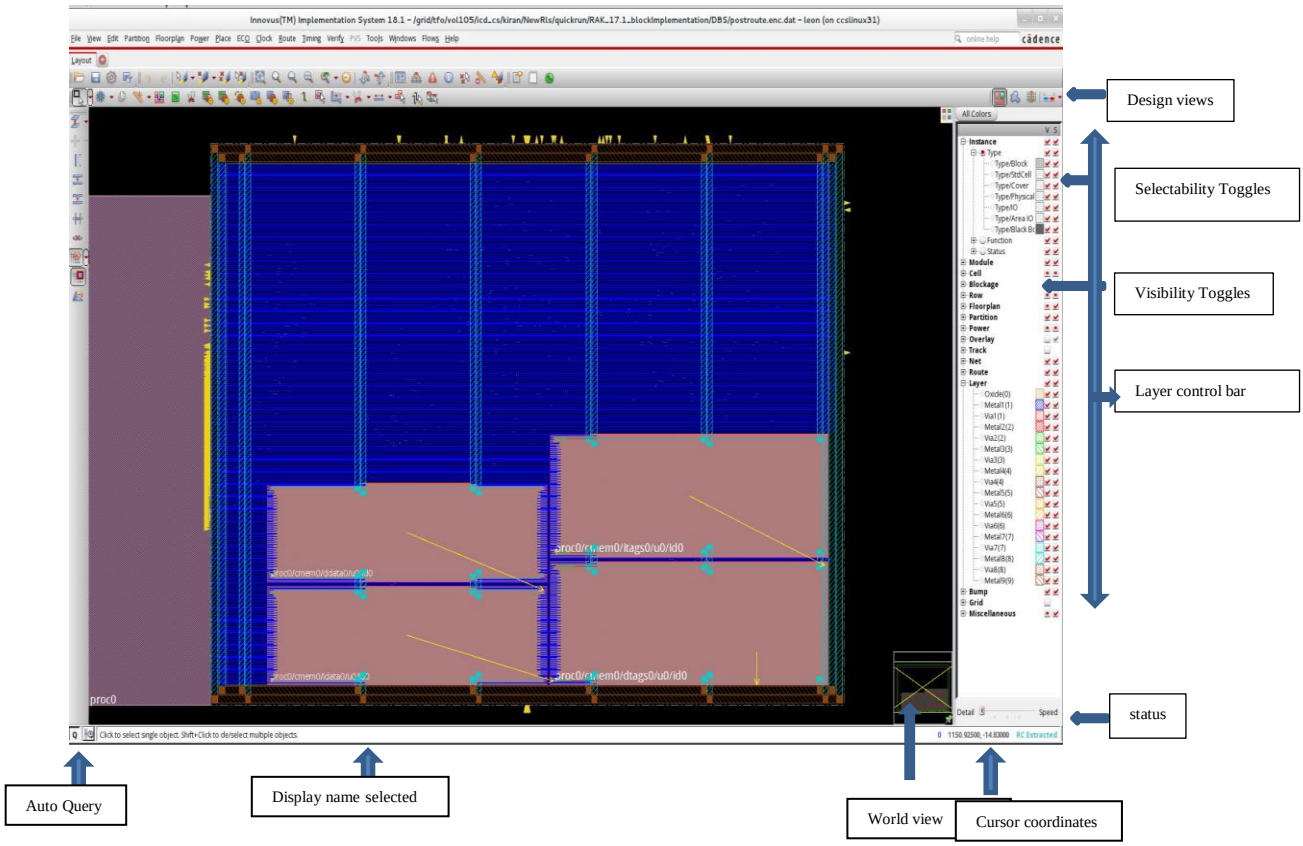


3. Click **OK** to restore the design. You should see the following:


proc0/cmem0/ddata0/u0/id0
proc0/cmem0/idata0/u0/id0
proc0/cmem0/itags0/u0/id0
proc0/cmem0/dtags0/u0/id0

2-2 Innovus GUI

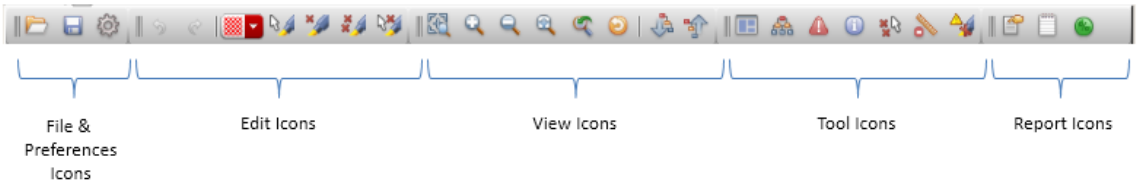
The Innovus GUI is shown below.



Tip: You can toggle the GUI on and off using the commands `win` and `win off`. These commands do not work when `innovus -nowin` is used.

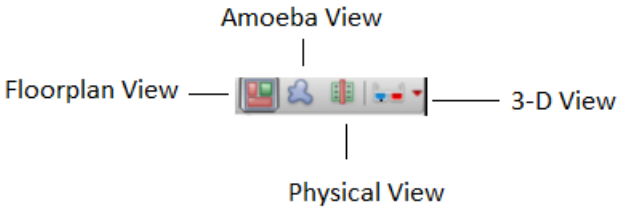
2-3 Toolbar Widgets

The **Toolbar Widgets** is the row of widgets directly under the Menus. To see what each widget is, hover the cursor over each widget to display their labels.

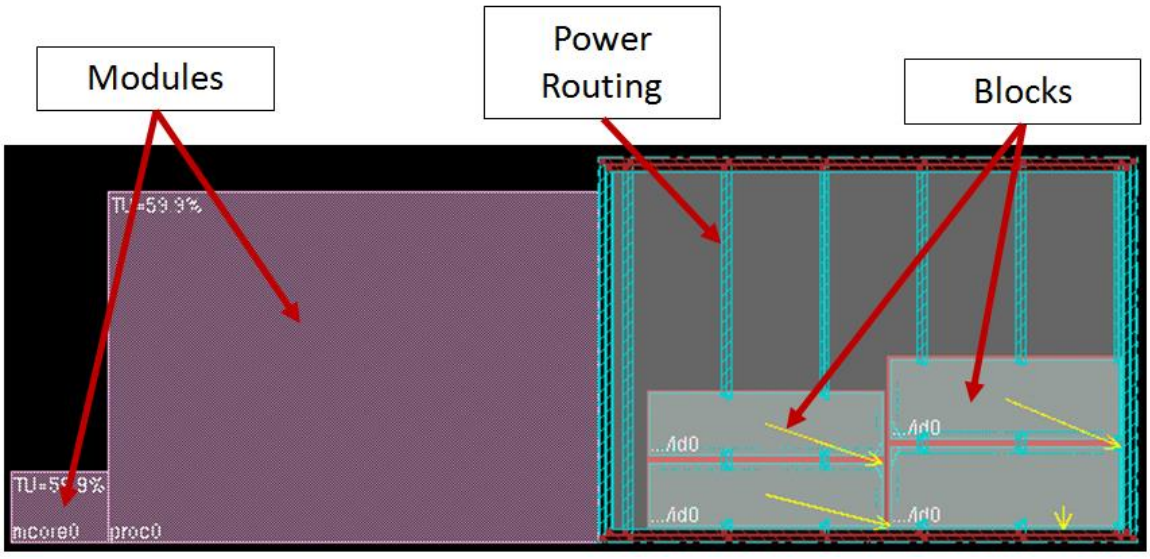


Design Views

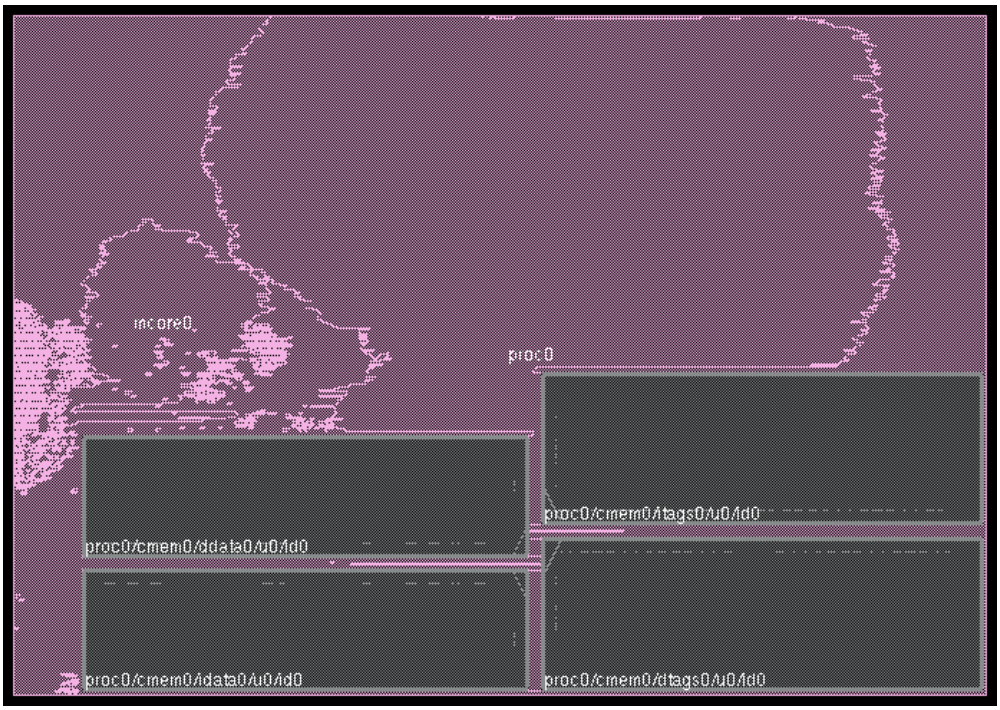
The three design views in Innovus allow certain objects to be viewable and selectable. You switch between views using the following widgets:



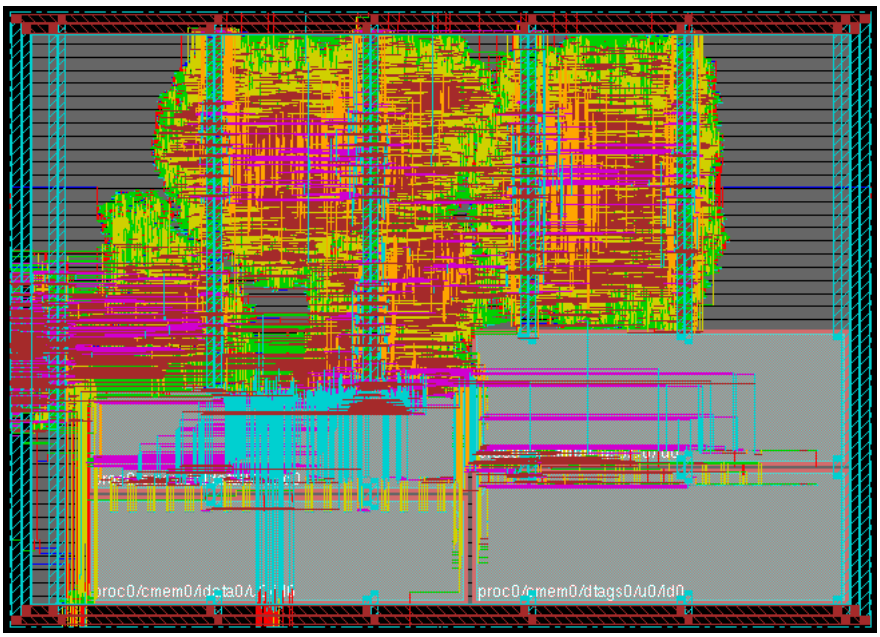
Floorplan View – Displays floorplan objects including modules, floorplan constraints, hard macros, power domains and power routing. Standard cells (unless FIXED) and signal routes are not displayed in the Floorplan View.



Amoeba View – Shows module shapes based on the placement of the standard cells and hard macros in each module. Useful for evaluating placement quality.



Physical View – Displays objects important to physical implementation including instances, power routing and signal routing. Note you can display floorplan objects in the Physical view using the Display tab of the Preferences window (**View - Set Preference**).



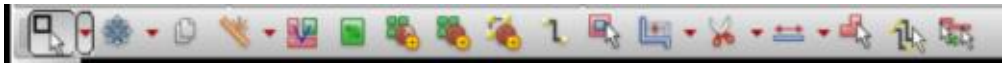
2-4 Tool Widgets

Below the Toolbar Widgets are the **Tool Widgets**. The Tool Widgets will change depending on which view is selected.

FloorplanView Widgets:

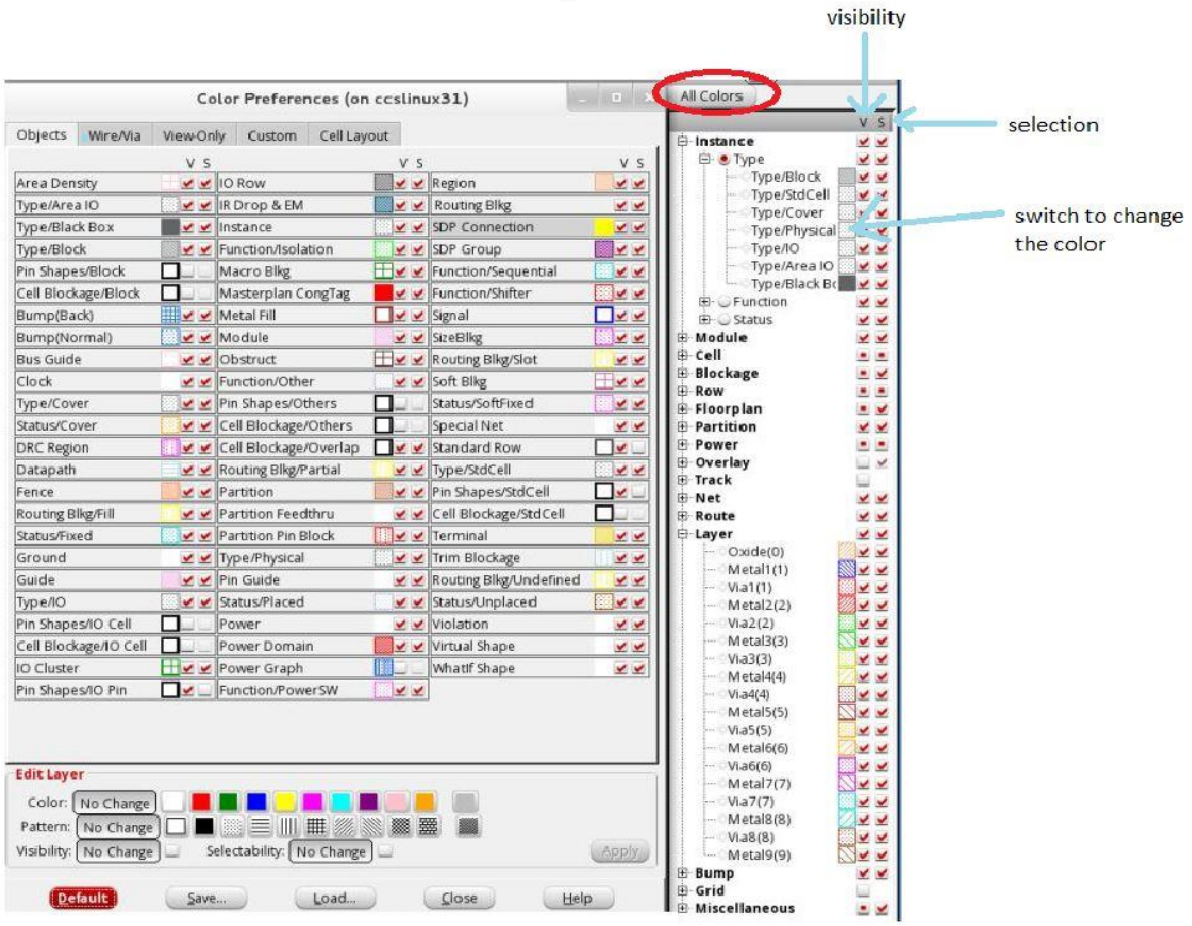
Amoeba View Widgets:

Physical View Widgets:



Layer Control Bar:

Layer Control Bar controls if an object is visible or can be selected.
Click **All colors** button to open form for controlling visibility, selection and display of all objects.



2-5 General Mouse Usage

Use the Left Mouse Button (LMB) to perform the action of the active icon. The **Select Object** icon  is active by default.

- Use the **Shift** key for multiple object selections, and to move multiple objects. You can also left-click and drag the mouse to select objects, such as partition pins and block pins.
- Use the **Space Bar** to change the highlighting focus on an object.
- Double-click the left mouse button on an object to view or change object attributes.

Tip: Use the **Selection** tab on the **Preferences** window (**View - Set Preference**) to change how selection behaves. For example, you can change selection to include objects intersecting the box or select by line.

Right-click and drag the mouse to specify an area that you want to see in greater detail. When you release the mouse button, the display zooms in to the selected area.

Click the middle button of your mouse to pan the viewable window to the center point. This is equivalent to using the **panCenter** command.

Move the scroll wheel of your mouse to pan and zoom the design:

- To zoom in or out, simply move the wheel forward or backward.
- To pan up or down, press **Ctrl-key** and move the wheel forward or backward.
- To pan left or right, press **Shift-key** and move the wheel forward or backward.

2-6 Binding Keys

Keyboard shortcuts, called binding keys, access commands in the design display area. A command's binding key is shown in the pull-down menus next to the command and also when hovering over a widget.

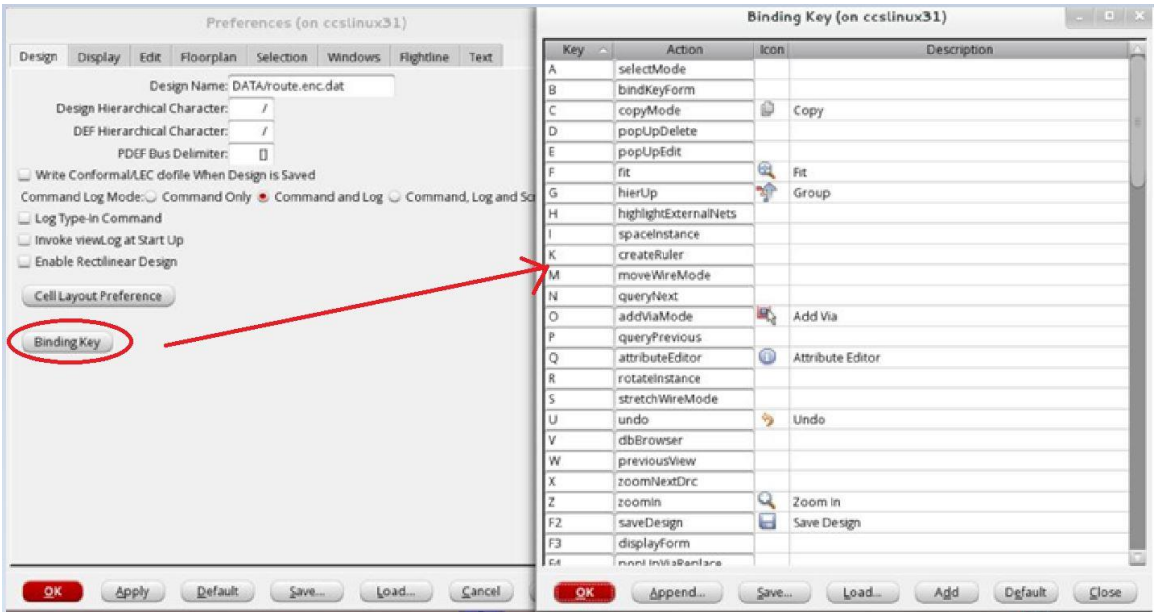
1. Zoom in and out using the **Z** and **Shift+Z** keys.

2. Fit the design by pressing the **F** binding key.

To see the full list of predefined binding keys:

3. Select **View - Set Preference**

4. On the **Design** tab select the **Binding Key** button. Here you can view, edit and add binding keys.



5. Close the **Binding Key** and **Preferences** forms.

2-7 Viewing and Editing Attributes

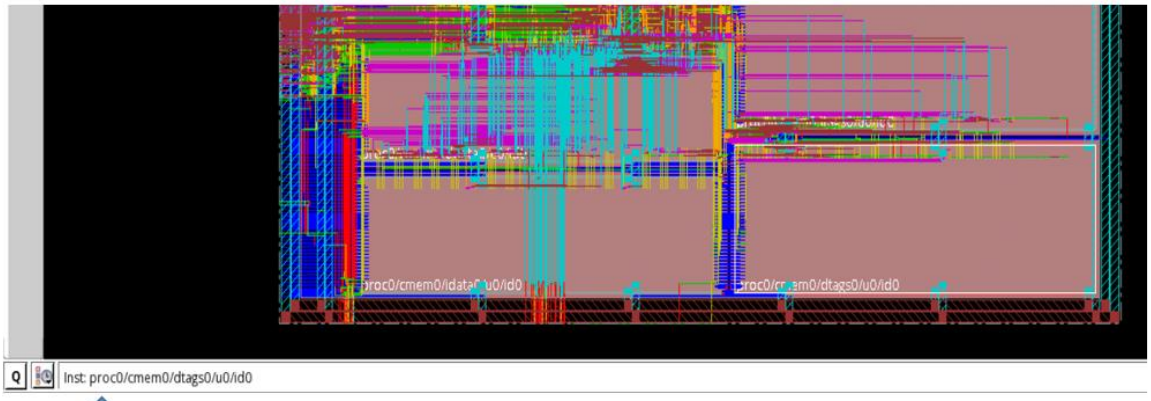
View and Edit object attributes using the Attribute Editor. It can be opened several ways:

- Double-click an object with the LMB or
- Select the object and enter the **Q** key or
- Place the mouse cursor over the object and click the RMB. Then select **Attribute Editor** on the menu which appears.

Tip: To open the Attribute Editor on overlapping objects, first click the LMB directly over the object, then use the **Space Bar** key to cycle through the objects. Once the desired object is selected, enter the **Q** key.

1. Display the Physical View by selecting the Physical View widget  .

2. Select one of the blocks in the floorplan. Observe the instance name is shown in the bottom left corner of the GUI.



3. Press the **Q** key to open the Attribute Editor for the selected object.

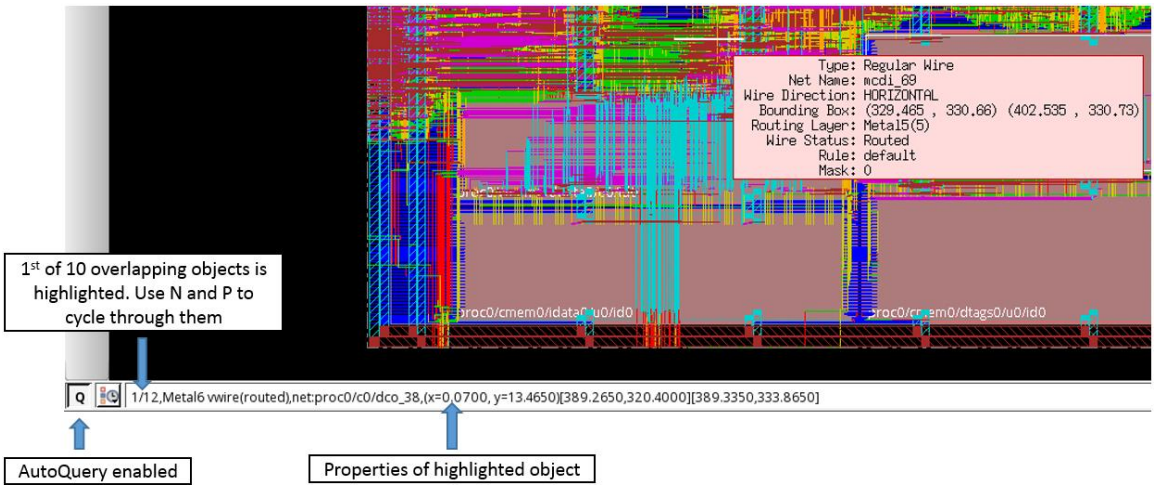
4. At the top the Attribute Editor it shows the Object Type followed by the attributes and their values. Notice the nets connected to the block are selected.

5. Change the **Status** to **Cover** and click **OK** to apply the change and close the Attribute Editor. A Status of Cover prevents the block from being moved manually or by any command.
6. Deselect everything by clicking LMB outside the design area.

2-8 Auto Query

The Auto Query (Q) button located at the bottom of the design display area enables and disables the auto query. When enabled, information on the object the cursor hovers over is shown in the lower left corner of the GUI. Auto Query is useful to quickly find out object information without having to select the object and open the Attribute Editor.

If there are overlapping objects under the cursor, text information displays for the object that is on top. Use the **N** key to get information on the next object, and the **P** key to get information on the previous object.

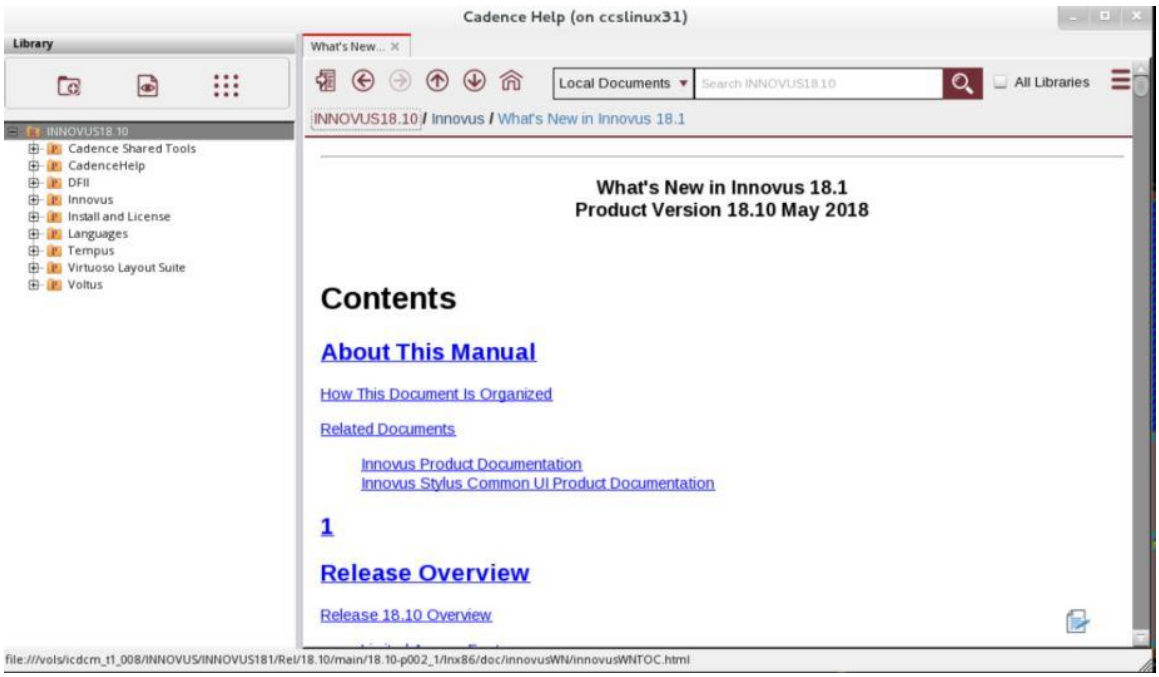


Tip: Press the **F8** when hovering over an object to print the property information to the console window. Note if a menu appears when you press F8, select **Send F8**.

2-9 Using Help

There are many ways to get help within Innovus:

- **Cadence Help** provides access to the documentation including the User Guide, Text Command Reference, Menu Reference and LEF/DEF Language Reference. Open Cadence Help by selecting **Help - Documentation Library** or executing `cdnshelp` from the Linux shell. Clicking the **Help** button on any of the Innovus forms will also open it. Select **Help - Launch Demo** from within Cadence Help to see a demonstration of using it.



- **Cadence Online Support** (http://support.cadence.com) is a 24x7 solution portal providing access to a variety of content (documentation, application notes, solution articles, Rapid Adoption Kits (RAKs), videos, etc.), the ability to create and manage support Cases, and you can customize it based on your preferences for products and notifications.

- Command line help is available from the Innovus console using the `man` or `help` commands.

7. Use the `man` command to display the man page for the `optDesign` command:

```
man optDesign
```

Press the Spacebar or Enter keys to scroll down and **Q** to quit the man page.

8. Use the `help` command to display the options available for a specific command. Use wildcards to report commands matching a certain string. Enter the following to see the commands for setting different command modes:

```
help set*Mode
```

   Enter the following to list the options for `setNanoRouteMode`:

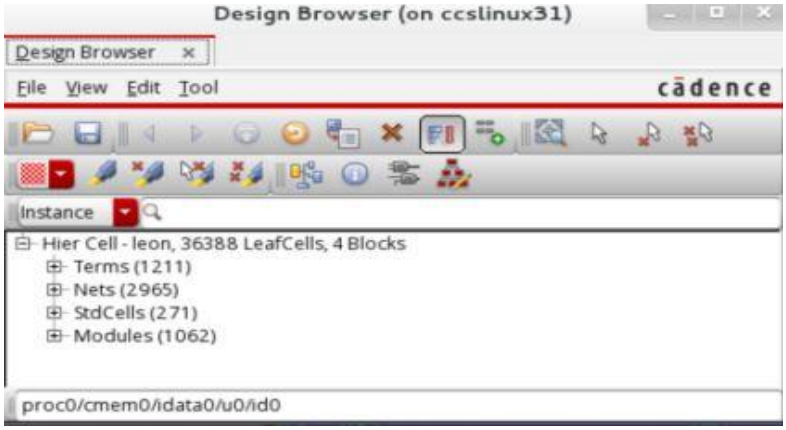
```
help setNanoRouteMode
```

2-10 Design Browser

Use the **Design Browser** to navigate through the chip's design hierarchy. You can use the Design Browser to view the design hierarchy tree at any time after the design is imported. The Design Browser also makes it easier to highlight specific modules, instances, or nets in the design display area.

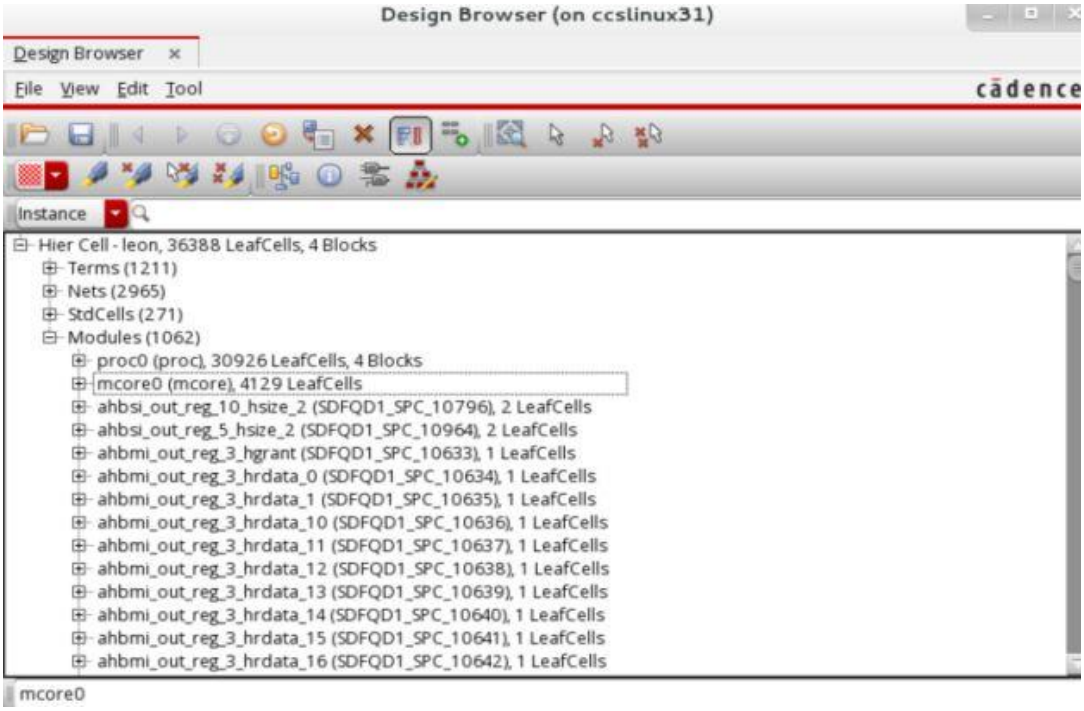
You can use the widgets in the Design Browser to open forms to navigate through displays, and perform actions. From the Design Browser, you can access the Connectivity Browser to display the number of nets between instances, and the Attribute Editor to display an object's type, name, and attributes.

1. Make sure nothing is selected and then open the Design Browser by selecting **Tools - Design Browser**. The window below appears.

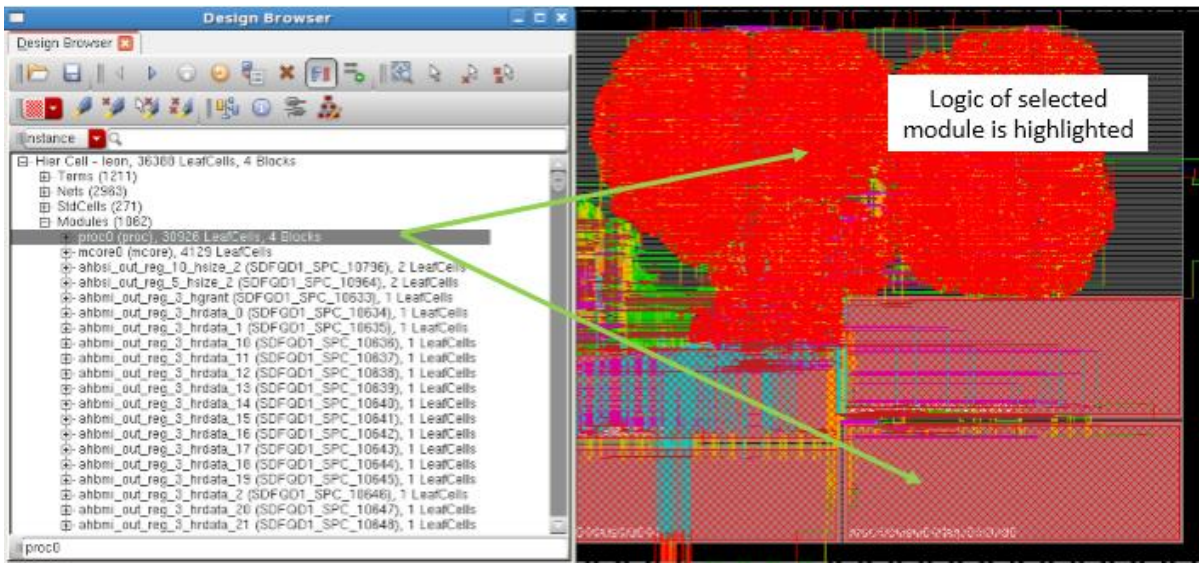


2. The root of the Design Browser is the top cell of the design, specified as the Hier Cell. Note if objects were selected, they would be listed in the Design Browser rather than the top cell. List the Modules at the next level of hierarchy by click the plus sign '+' next to **Modules**.

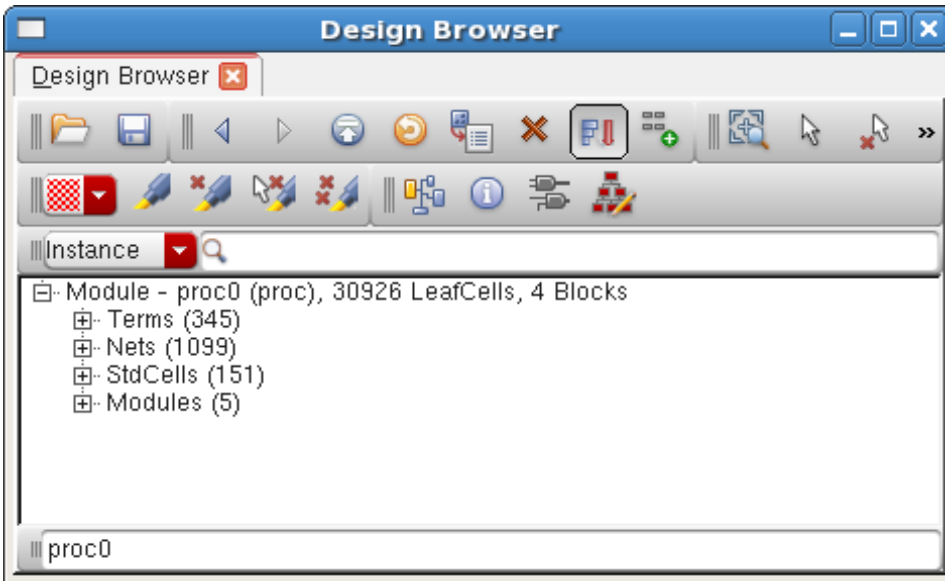
   For each module it lists the instance name (i.e. *proc0*) and the module name in parenthesis (i.e. *proc*) following by the number of leaf cells and blocks in the module.



3. Single-click the instance *proc0* in the Design Browser and observe the objects belonging to this instance are highlighted in the GUI:



4. Double-click on the instance *proc0* in the Design Browser to drill down and list the objects belonging to it. Notice *proc0* is listed as the top module now.



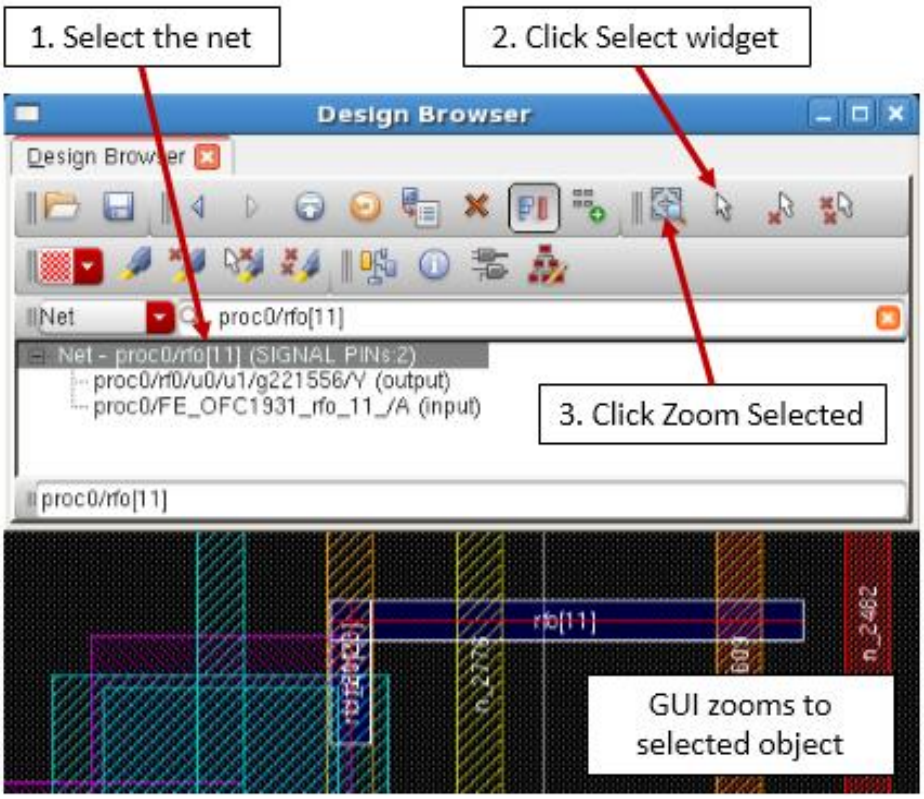
Tip: These navigation widgets  take you to the previous, next and top.

The search box is used to search for instances, nets, groups or cells and accepts wildcards.

5. Set the object type to **Net** and enter the net name `proc0/rfo[11]` followed by the **Enter** key. The net matching that name is listed in the Design Browser. You can expand it to see what pins it connects to.

6. To select and zoom to this net:

   a. Select the object in the Design Browser
   b. Click the **Select** widget
   c. Click the **Zoom Selected** widget

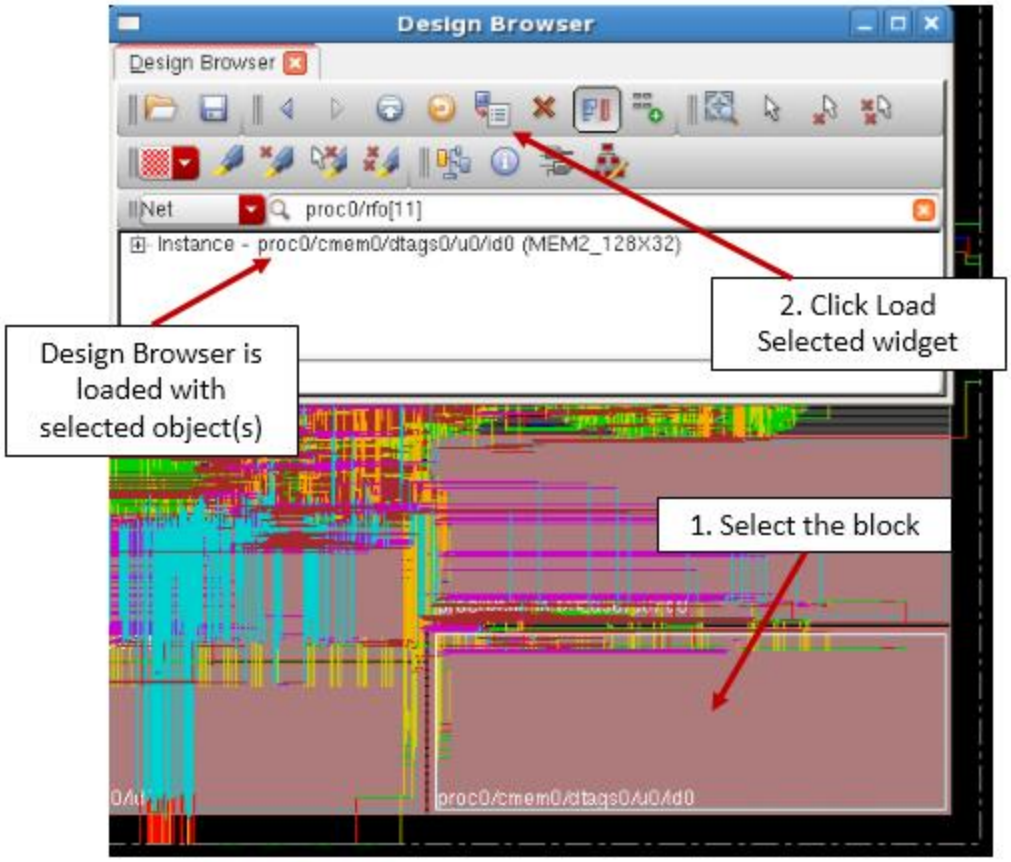


Tip: When a net is selected each logical terminal (represented by yellow box) is marked with an ‘X’ meaning input or an ‘O’ meaning an output pin.

You can also load selected objects into the Design Browser.

7. Fit the design and select the block in the lower right of the floorplan.

8. Click the **Get Selected** widget in the Design Browser and observe the selected object(s) is listed.



9. Close the Design Browser.

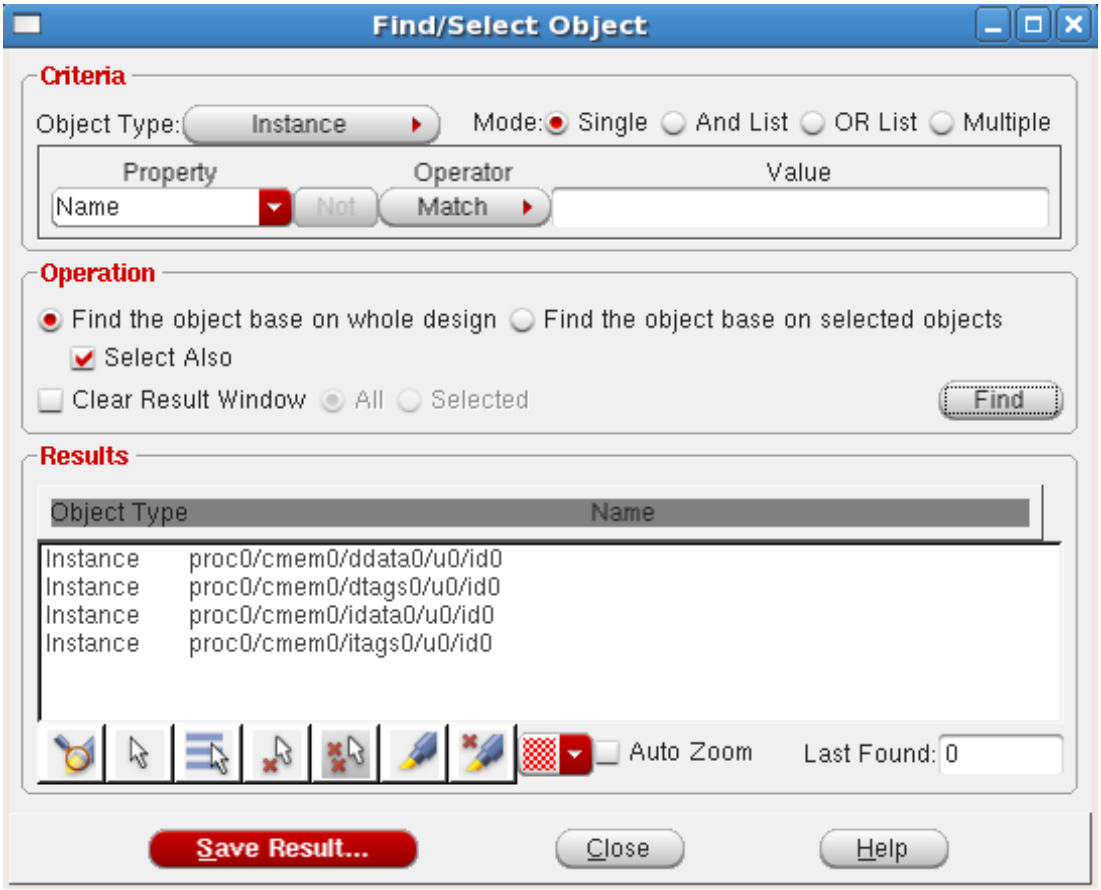
2-11 Find/Select Object

Use the **Find/Select Object** form to find and select specific object types in the design display area. Use the Find/Select Object form to select all the hard macros.

1. Select **View - Find/Select Object…**

2. Enter the following:

   Object Type: *Instance*
   Property: *Type*
   Value: *Block*
   Select *Find the object base on whole design*
3. Click **Find** and list the instance names of all the hard macros. Click the **Select All** widget  to select all the items in the GUI that are in the list.



Tip: If you are running a new search, make sure you check the **Clear Result Window** checkbox to clear the preexisting results.

4. Close the **Find/Select Object** form.

2-12 Stylus

Stylus is an infrastructure that offers three significant features:

- Stylus Common User Interface offering consistent commands across the whole digital flow
- Stylus Unified Metrics for capturing and reporting
- Stylus Flow Kit for design flow capture and deployment

The Stylus Common User Interface (Common UI from this point onwards) has been designed to be used across Genus, Joules, Modus, Innovus, Tempus, and Voltus tools. By providing a common interface from RTL to signoff, the Common UI enhances user experience by making it easier to work with multiple Cadence products.

Common UI:

The Common UI provides an improved interface with reduced number of commands and attribute cleanup. It also enhances usability with cleaned up log files and improved messaging.

You can now use a single command, `get_db` , to query and filter all database attributes. `get_db` replaces `dbGet, get_ccopt_property`, and various other get methods. It includes `get_property` timing attributes, although `get_property` is still retained for SDC usage.

The `set_db` and `reset_db` commands are the companion commands to set values.

Unified Metrics:

Unified Metrics is a system integrated with Innovus that delivers data about the design and run to you. It displays the streamed data from the reports obtained during the flow in a structured format.

Follow the steps to record values using Unified Metrics,

- Enable Unified Metrics by the command `enable_metrics –on`
- Initiate a starting point by using command `push_snapshot_stack`
- Perform the step needed to be recorded like `place_opt_design, report_timing, ccopt_design .`
- End the stack using command `pop_snapshot_stack`
- Create the snapshot name using command `create_snapshot –name optDesign`
- Report the final values in a html file using command `report_qor -file metrics.html -format html`

The unified metrics are implemented in the `preCTS, postCTS` and `postROUTE` stages of the design in this RAK.

New Flow Kit

A new set of flow commands, including `create_flow` and `create_flow_step`, support user flows more easily. New flow and flow_step objects store the state of the flow with the database.

For more information on Stylus, please refer Innovus 18.1 User Guide.

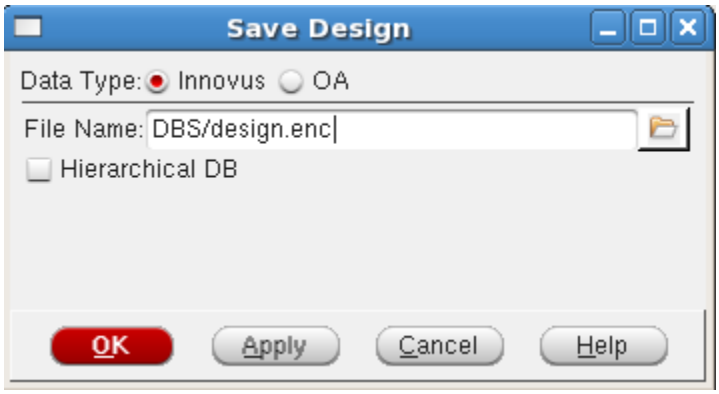
2-13 Saving and Restoring a Design Database

Designs are saved and restored in Innovus using **File - Save Design** and **File - Restore Design** menus. The respective text commands are `saveDesign` and `restoreDesign`.

1. Selecting **File - Save Design**.

2. On the Save Design form:

    - Select **Data Type:** *Innovus* to save it in the Innovus database format.
    - Specify **File Name:** *DBS/design.enc* or use the File Browser to navigate to the DBS directory and specify *design.enc*.



3. Click **OK** to save the design.

This will create a file called `DBS/design.enc` which contains the command to restore the database directory `design.enc.dat`.

In a new session you can then restore the design using one of the following methods:

- Using the menu by selecting **File - Restore Design** and select `DBS/design.enc`
- OR source the file at the Innovus prompt: `source DBS/design.enc`
- OR load the design when invoking Innovus: `innovus -init DBS/design.enc`

The Innovus design database is composed of several ascii files representing the different design data. The files may vary depending on the state of the design:

File/Directory	Description
.fp	Saves floorplan information; this file is not incremental unlike the DEF
.globals	Contains pointers to libraries and netlist, MMMC file and other design information
.mode	Saves mode options (like setPlaceMode, setOptMode, setNanoRouteMode)
.place	Saves standard cell placement
.route	Saves routing information
.pref.tcl	Saves user-specified design and display preferences
viewDefinition.tcl	MMMC setup file
mmmc directory	Includes the constraint modes

```
[DEV}innovus 2> ls DBS/design.enc.dat/
gui.pref.tcl       leon.fp.spr.gz  leon.place.gz   leon_power_constraints.tcl
inn.cmd.gz         leon.globals    leon.prop       libs
leon.ctstch        leon.init       leon.route.gz   mmmc
leon.dbglobals     leon.metric.gz  leon.symtbl.gz  set_inst_lib.tcl
leon.fp.gz         leon.mode       leon.v.bin      viewDefinition.tcl
leon.fp.relFPlan   leon.opconds    leon.v.bin_lib
```

Exit Innovus by entering:

```
exit
```

Module 3: Flat Implementation Flow

3-1 Importing the Design

In the module you implement a block level design using Innovus physical design flow. You are provided the floorplan and then execute the flow including placement, timing optimization and analysis, clock tree synthesis, and routing.

1. Invoke Innovus:

```
innovus
```

Use the **Design Import** form to import the Verilog netlist, physical libraries (LEF), process technology libraries, timing libraries, and timing constraints.

2. Select **File - Import Design**

3. Select the **Load** button.

4. Select the file *DATA/leon.globals* then click **Open**.

Observe the Design Import form is populated with the settings from *leon.globals*.

5. Click **OK** to import the design data.

Tips:

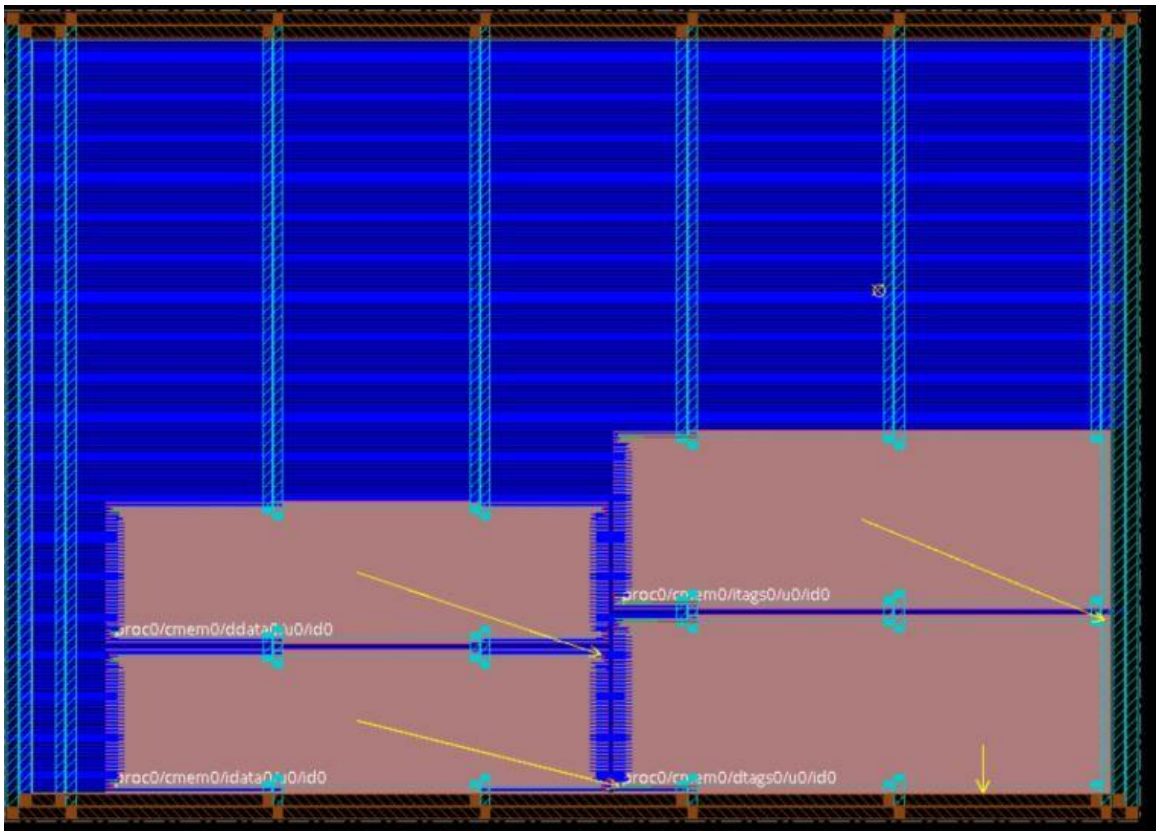
- When entering the values for the first time use the **Save** button to save the settings to a file. Then you can load the file in future sessions.
- The technology LEF file must always be listed first in the list of LEF files.
- Use the equivalent text commands to the Design Import form to quickly load an existing globals file:

```
source DATA/leon.globals
init_design
```

6. Select **File – Load – Floorplan** to load the floorplan file.

7. Select the file *DATA/leon.power.fp* and click **Open**.

8. Press the **F** key to fit the design and you should see the floorplan below. The floorplan has the blocks placed and power routing complete. You are now ready to begin the implementation flow.



First, it is important to specify the process technology because it sets capacitance filters and extraction effort level based on the process node.

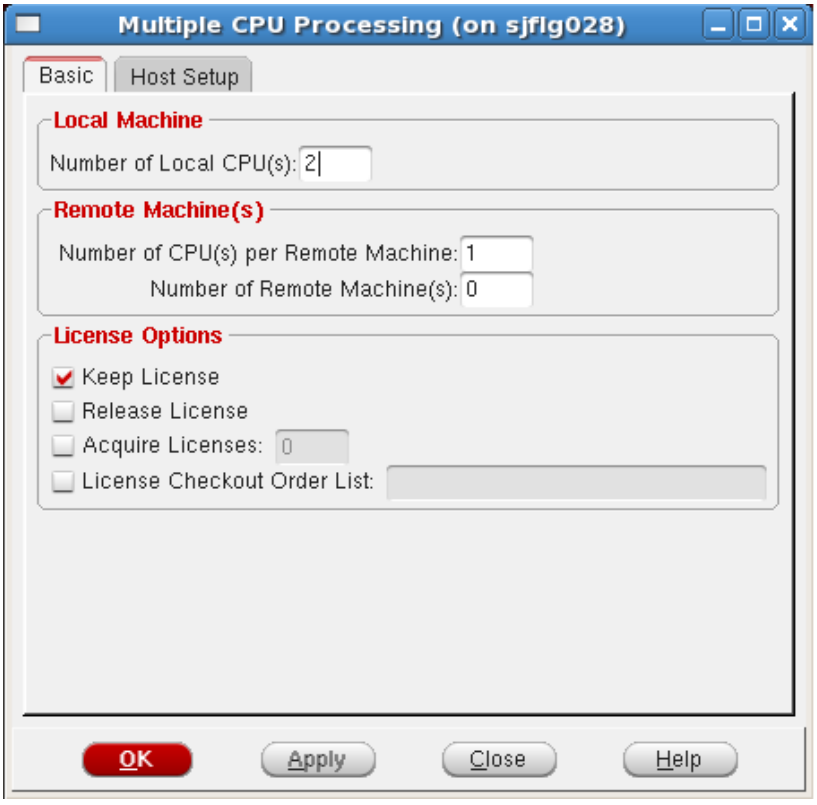
9. Specify the process:

```
setDesignMode -process 45
```

Using multiple CPUs can significantly decrease run time to implement a design. Two CPUs can be used with the base license. Additional licenses are needed beyond two CPUs as explained in the *Accelerating the Design Process By Using Multiple-CPU Processing* chapter of the *Innovus User Guide*.

10. If the machine you are running on has two CPUs available select **Tools – Set Multiple CPU Usage**.

11. Set **Number of Local CPU(s)** to 2.

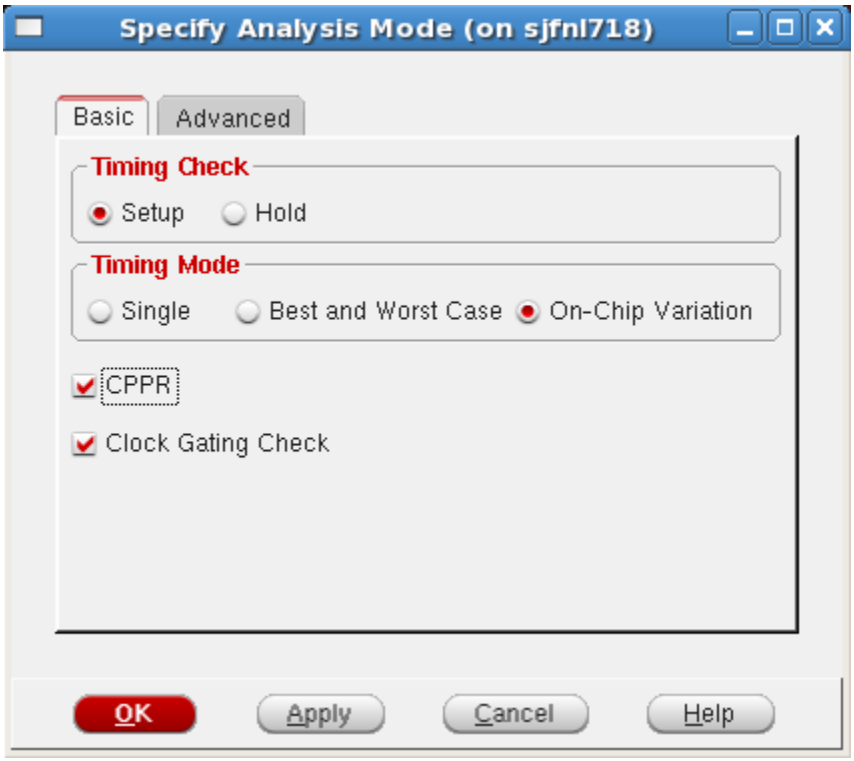


12. Click **OK**. This will enable the use of two CPUs by commands in the flow that run multi-threaded. You're welcome to set this value to a higher number of CPUs if you have the resources and licenses available.

Update the timing analysis mode to On-Chip Variation.

13. Select **Tools – Set Mode – Specify Analysis Mode**.

14. In the **Timing Mode** select **On-Chip Variation** and select **CPPR**.



15. Click **OK**

3-2 Running Placement + Pre-CTS Optimization

GigaPlace engine performs standard cell placement and IO pin assignment while GigaOpt engine performs preCTS, postCTS, and postRoute optimization.

Command `place_opt_design` executes pre-CTS flow by running both placement and pre-CTS optimization. Optimization at this stage is performed with ideal clocks because the clock tree has not been inserted yet.

Alternatively, placement and optimization can be run separately using the commands as follows, but `place_opt_design` is recommended as it provides better integration between placement and optimization to achieve faster runtime and better PPA.

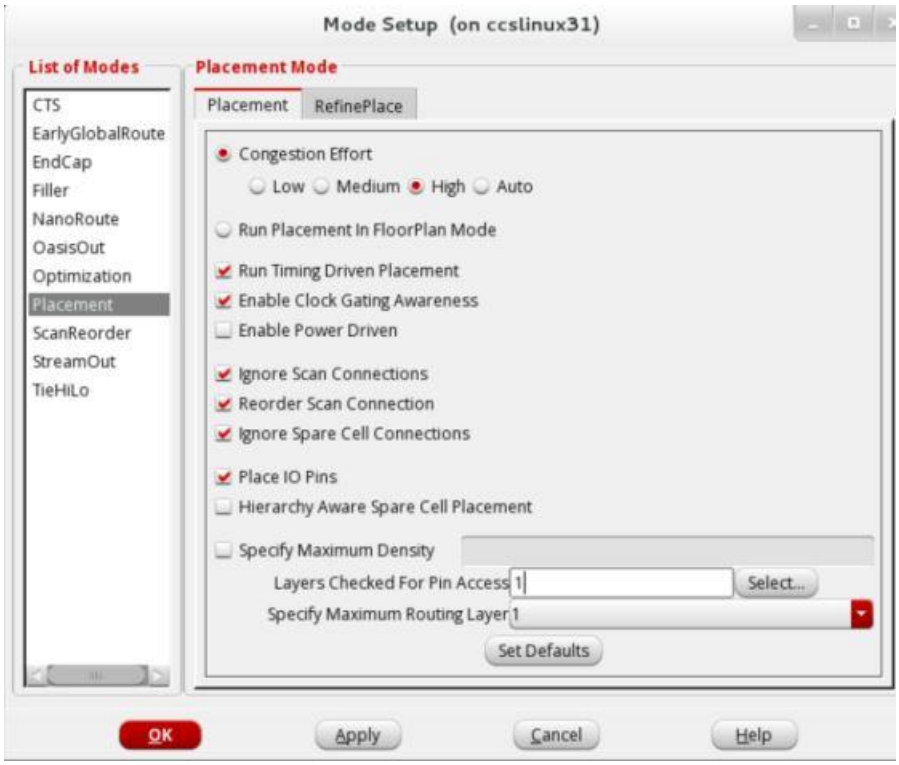
```
placeDesign
optDesign -preCTS
```

1. You can specify placement and optimization mode settings prior to running `place_opt_design`. Mode settings allow you to customize how commands run.

   Select **Tools – Set Mode – Mode Setup** to specify the mode settings for placement and optimization.

2. On the **Mode Setup** form select **Placement** on left side to see the mode settings available for placement.

    - Select **Place IO Pins** if it's not selected so that the IO pins get placed.
    - Select **Optimization** to see the optimization settings. You will use the default settings for optimization.



Tip: The equivalent text commands are `setPlaceMode` and `setOptMode` to specify these mode settings. `getPlaceMode` and `getOptMode` report what the current settings are. Other commands have similar `set*Mode` and `get*Mode` commands.

3. Close the **Mode Setup** form by clicking **OK**.

4. Specify cells with small drive strength should not be used by running the following setDontUse commands:

```
set_dont_use *XL true
set_dont_use *X1 true
```

5. Enable Unified Metrics by running the following command,

```
enable_metrics -on
```

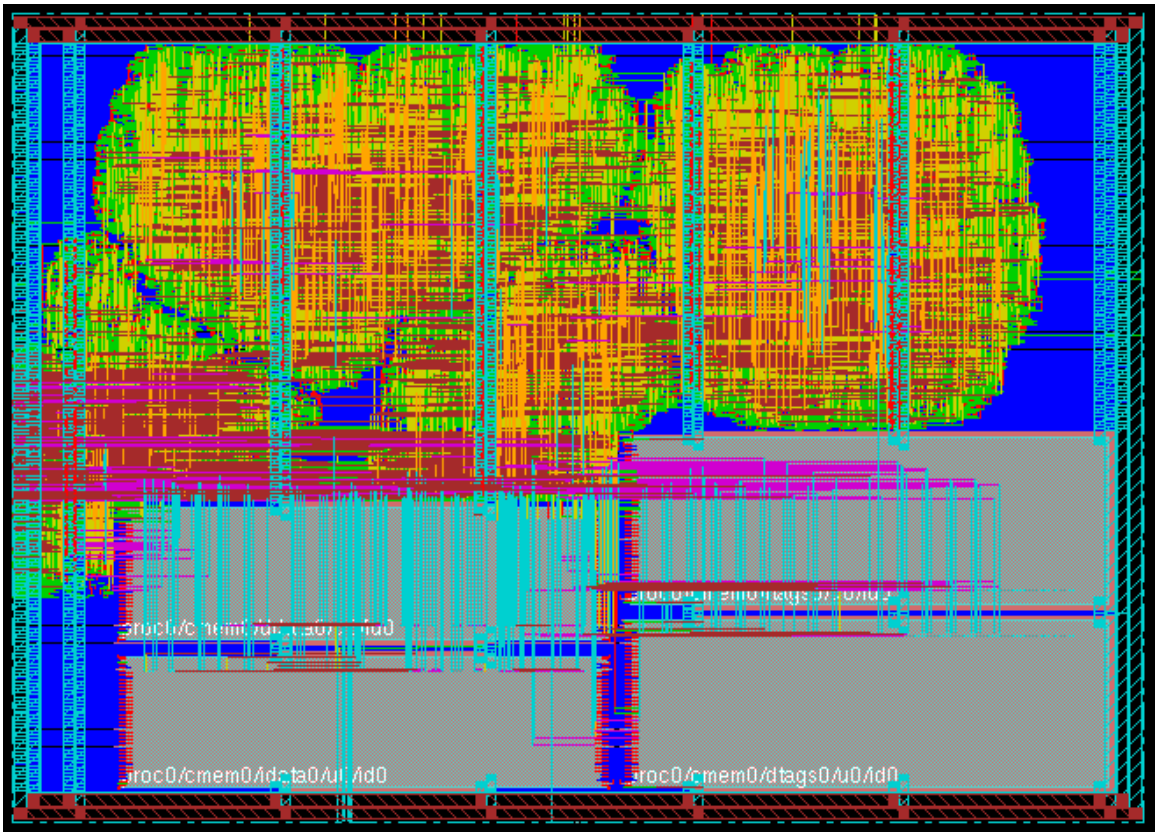
6. Start a new metric track using the following command,

```
push_snapshot_stack
```

7. Run placement and pre-CTS optimization by running the following command:

```
place_opt_design
```

Once placement and preCTS optimization is done, you can view the placed design in the Amoeba and Physical views. Observe the placed standard cells are visible in the Physical view (you may have to zoom in or turn off the display of Nets to see the placed standard cells). The reason there are routed nets is because placement automatically calls Early Global Route (eGR) so you can analyze congestion and estimate parasitics. In the next section you'll manually run Early Global Route to analyze congestion.



In the console you'll see a timing summary output so you can quickly see timing results after optimization as well as standard cell utilization (density) and EarlyGlobalRoute overflow values:

```
+--------------------+---------+---------+---------+---------+
|     Setup mode     |   all   | reg2reg |reg2cgate| default |
+--------------------+---------+---------+---------+---------+
|           WNS (ns):| -0.002  | -0.002  |  0.329  |  0.165  |
|           TNS (ns):| -0.003  | -0.003  |  0.000  |  0.000  |
|    Violating Paths:|    2    |    2    |    0    |    0    |
|          All Paths:|  12273  |  11430  |   33    |  1052   |
+--------------------+---------+---------+---------+---------+

+----------------+-----------------------------+-----------------+
|                |             Real            |      Total      |
|    DRVs        +------------------+----------+-----------------|
|                | Nr nets(terms)   | Worst Vio | Nr nets(terms) |
+----------------+------------------+-----------+-----------------+
|   max_cap      |      0 (0)       |   0.000   |      8 (8)      |
|   max_tran     |      0 (0)       |   0.000   |      6 (6)      |
|   max_fanout   |      0 (0)       |     0     |      0 (0)      |
|   max_length   |      0 (0)       |     0     |      0 (0)      |
+----------------+------------------+-----------+-----------------+

Density: 46.432%
Routing Overflow: 0.00% H and 0.00% V
```

Use the **checkPlace** command to check for placement violations including overlapping instances, instances out of the core area, or off-grid placement.

8. Select **Place - Check Placement** and click **OK**.

If there are any placement violations, they should be reviewed and fixed

9. Save the placed design:

```
saveDesign DBS/prects.enc
```

Note: The `check_design` command checks the preconditions for major flow steps before they are run. If there are errors identified by `check_design`, the current script will stop check.

Option `-type` can be used to specify the category of the flow to be checked. For example, `check_design -type` place checks for possible issues related to placement such as PG rail alignment and pin access issues. This check is run before placement.

3-3 Running Early Global Route

Early Global Route is a combination of global routing and track assignment which correlates well to detail route while running in a fraction of the time. In this module, Early Global Route is run twice. The first run will demonstrate route congestion by limiting the number of metal routing layers to 3 instead of 9 layers. The second run will use all 9 metal layers to complete the prototyping of the design.

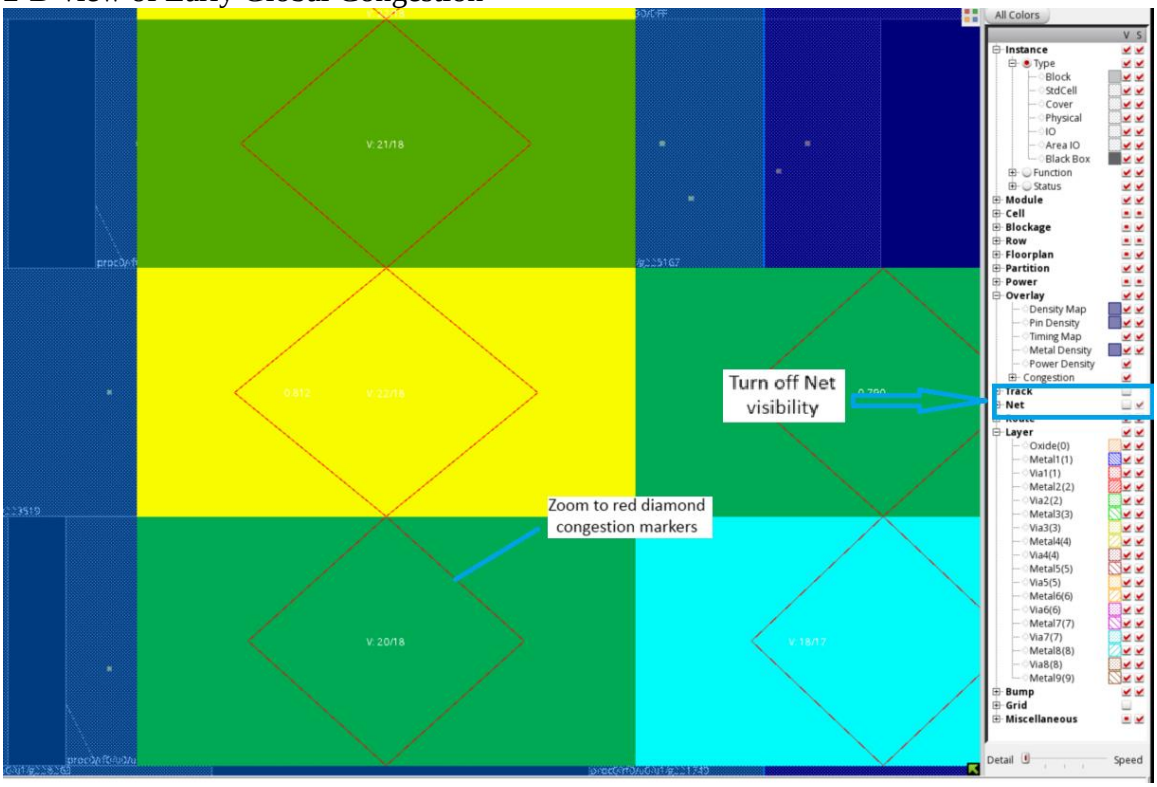
1. Select **Route – Early Global Route**

2. Change – **Routing Layer: max: to Metal3(3)**

3. Click **OK** to run Early Global Route.

Once Early Global route completes, analyze the congestion.

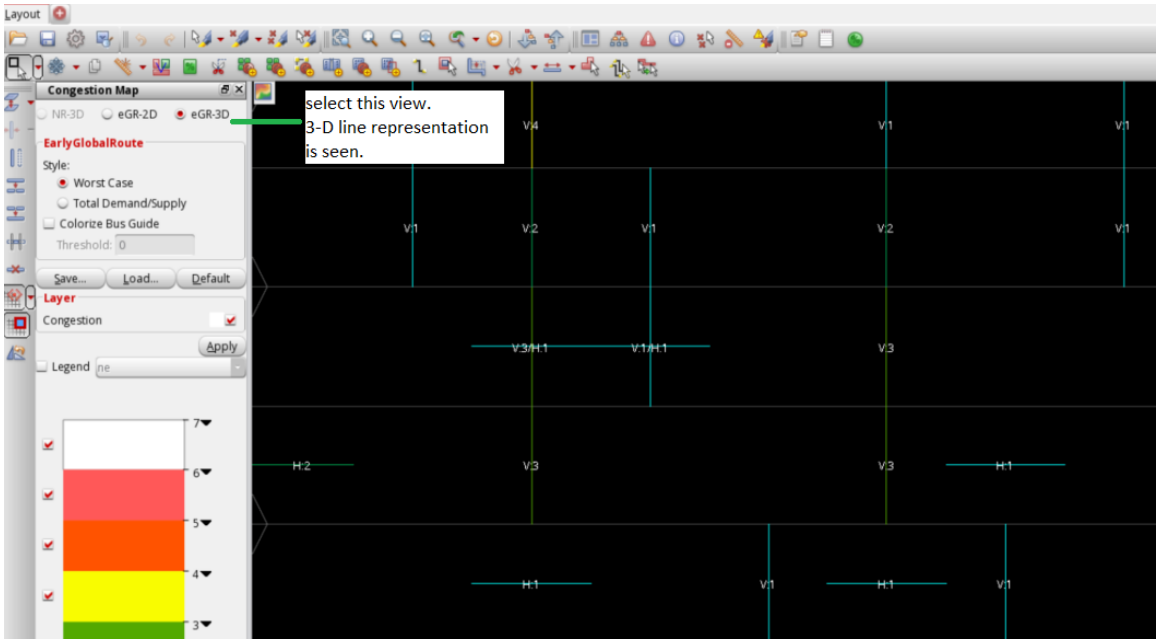
4. Turn off the visibility of nets by deselecting the checkbox next to **Net** on the right side of the GUI. Turn ON the overlay checkbox if its already not there. Then zoom into the red diamond shapes until you can read the numbers inside the diamond.
5. 2-D view of Early Global Congestion



The red color diamond shapes which are Early Route congestion markers. Zoom-in to view the congestion markers. Inside the diamond marker, there is a set of numbers in the format of either **H: #-top/#-bottom** or **V: #-top/#- bottom**. H stands for horizontal congestion and V stands for vertical congestion. The #- top is for the required number of routing tracks used in this area and #-bottom is the available routing tracks. These diamond shaped congestion locators represent an average in the area. Since there is only 1 vertical route layer (metal 2), most congestion markers are vertical.

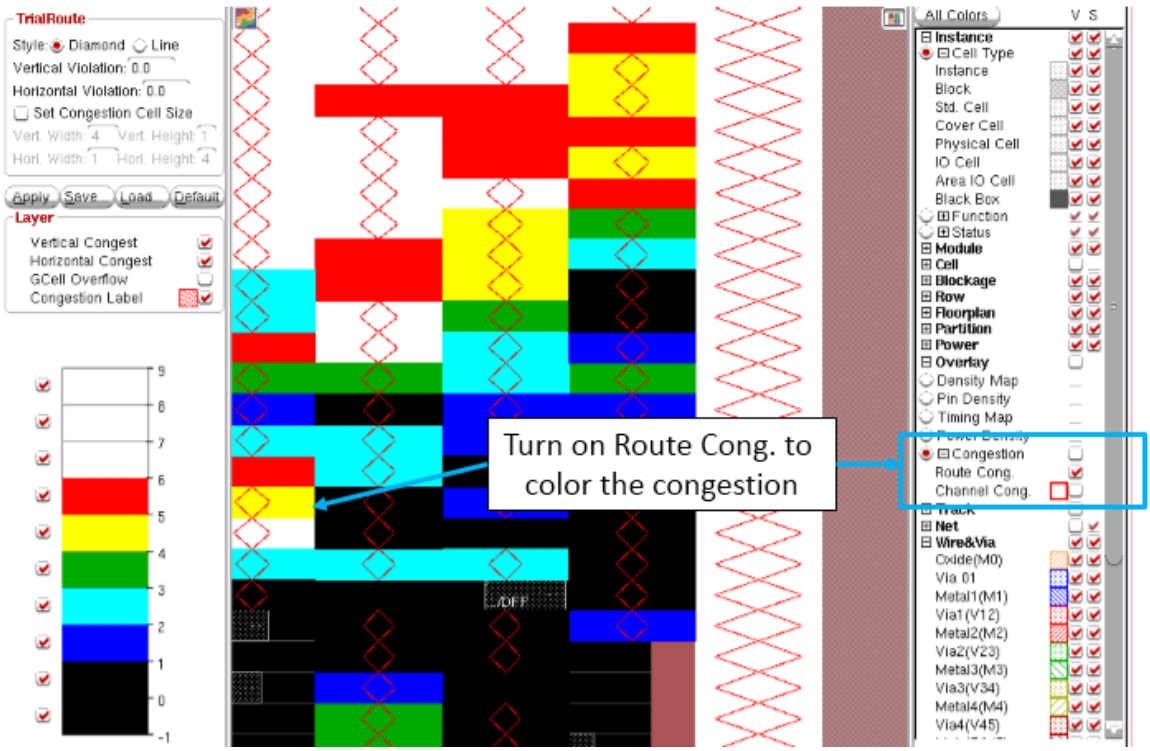
Another display of congestion will color the area based on the level of over congestion. This makes it easier to distinguish whether an area is highly or moderately congested.

In the below eGR-3D view, line representation is used.
Ex: V4 represents Vertical routing is congested by 4 tracks instead of #top /#bottom. In the legend 3-4 indicates yellow. So, the V4 line is in yellow color.



6. Expand **Overlay** in the Layer Control Bar then select the **Congestion** radio button and then the **Route Cong.** checkbox as shown below if its already checked ON.

   Observe a new pane appears on the left side of GUI. This pane has controls for displaying horizontal and vertical congestion separately, as well as how many tracks a GCell is overcongested. For example, green indicates overcongestion of 3 tracks. Zoom and pan around to see areas of overcongestion.



7. Deselect **Route Cong.** in the Layer Control bar to turn off the congestion coloring.

8. Make nets visible again by selecting the visibility checkbox for **Net** on the Layer Control bar.

9. Another way to analyze congestion is to view the log file or the Innovus console for the congestion table produced by EarlyGlobalRoute. The label of the table is:

```
Congestion distribution:

Overflow after earlyGlobalRoute 0.55% H + 2.15% V
```

For details on how Overflow values are computed, refer to section titled "Congestion Distribution Report" in Innovus User Guide 18.1

10. Run command `reportCongestion -hotspot` in the terminal to display the congestion values.
    The maximum, total hotspot and the top 5 congestion hotspot values are displayed as below

```
[hotspot] +------------+---------------+---------------+
[hotspot] |            |   max hotspot | total hotspot |
[hotspot] +------------+---------------+---------------+
[hotspot] | normalized |         92.13 |        204.13 |
[hotspot] +------------+---------------+---------------+
```

Local HotSpot Analysis: normalized max congestion hotspot area = 92.13, normalized total congestion hotspot area = 204.13 (area is in unit of 4 std-cell row bins)

```
[hotspot] top 5 congestion hotspot bounding boxes and scores of
normalized hotspot
[hotspot] +-----+-----------------------------------+---------------+
[hotspot] | top |            hotspot bbox           | hotspot score |
[hotspot] +-----+-----------------------------------+---------------+
[hotspot] |  1  |   466.18   452.58   520.90   575.70 |       88.59   |
[hotspot] +-----+-----------------------------------+---------------+
[hotspot] |  2  |   657.70   452.58   698.74   575.70 |       81.97   |
[hotspot] +-----+-----------------------------------+---------------+
[hotspot] |  3  |   219.94   384.18   247.30   411.54 |        6.89   |
[hotspot] +-----+-----------------------------------+---------------+
[hotspot] |  4  |   274.66   397.86   302.02   425.22 |        6.30   |
[hotspot] +-----+-----------------------------------+---------------+
[hotspot] |  5  |   657.70   425.22   685.06   452.58 |        3.93   |
[hotspot] +-----+-----------------------------------+---------------+
```

These top hotspot scores indicates that congestion is localized in the mentioned bound box area. Analysis must be done to avoid these congestion hotspots.

Now, run Early Global Route with 9 metal routing layers.

11. Select **Route – Early Global Route**

12. Change – **Routing Layer: max: to Metal9(9)**

13. Click **OK**. Observe the overflow is much more reasonable (actual numbers will vary but should be near 0%):

    ```
    Overflow after earlyGlobalRoute 0.00% H + 0.00% V
    ```

14. Run command `reportCongestion -hotspot.` The hotspot values are returned 0 as shown below,

```
[hotspot] +------------+---------------+---------------+
[hotspot] |            |   max hotspot | total hotspot |
[hotspot] +------------+---------------+---------------+
[hotspot] | normalized |          0.00 |          0.00 |
[hotspot] +------------+---------------+---------------+
```

3-4 Extracting RC Data

The capacitance and resistance values for all the nets in the design are extracted by the **Extract RC** form.

Extraction is run in pre-route mode prior to signal routing and in post-route mode after the signals are routed with NanoRoute. In post-route mode there are four effort levels to choose from (low, medium, high and signoff) which increase with accuracy at the expense of longer run-times.

The RC extraction mode can be changed by the **Tools - Set Mode - Specify RC Extraction Mode** form. Since the design has not been routed we will leave the default mode set which is pre-route mode.

1. Select **Timing - Extract RC**

   Select – **Save SPEF to** and specify file *leon.spef*
   Select – **RC Corner to Output**: *rc_worst*

2. Click **OK**.

If you review the log file you'll see extraction is run and then a SPEF file is written out. The equivalent text commands to run extraction and export SPEF are:

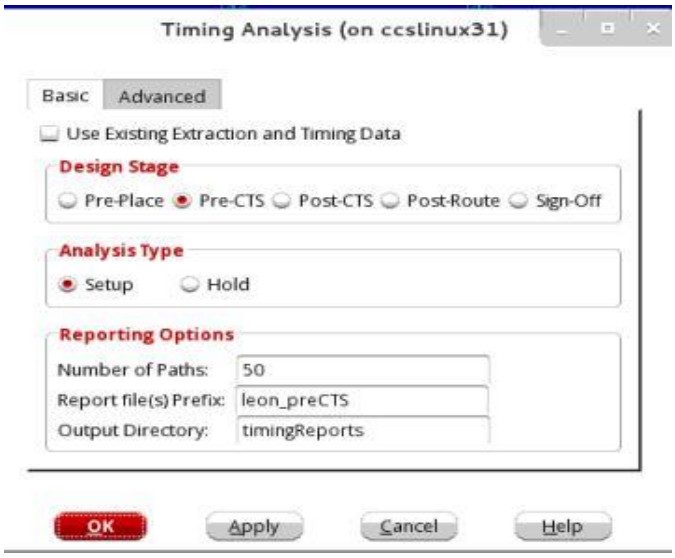
```
extractRC
rcOut -spef leon.spef -rc_corner rc_worst
```

3-5 Running Timing Analysis

Timing analysis can be run after extracting RC. Note if extraction was not run prior to timing analysis, timing analysis will automatically run RC extraction followed by timing analysis. Since clock tree synthesis was not run yet, an ideal clock is used and the ideal clock transition delay value is 0.1ps.

1. Select **Timing – Report Timing**

2. In the Design Stage section, Select – **Pre-CTS**.
   In the Analysis Type section, Select – **Setup** which is default.



3. Click the **Advanced** tab and review the options. Use default options and click **OK**.

When timing analysis is done, a timing summary is printed to the Innovus shell. This file is saved under–directory *timingReports*. The ***timingReports*** directory contains several additional reports commonly used for debugging:

The file with suffix `.summary.gz`, is created after each phase once timing analysis is run. As shown below, file `leon_preCTS.summary.gz`, shows timing and DRV information at preCTS stage (In Pre-CTS stage focus is on setup whereas hold is ignored).

```
------------------------------------------------------------
          timeDesign Summary
------------------------------------------------------------

+--------------------+---------+---------+---------+---------+
|     Setup mode     |   all   | reg2reg |reg2cgate| default |
+--------------------+---------+---------+---------+---------+
|           WNS (ns):|  0.019  |  0.019  |  0.332  |  0.201  |
|           TNS (ns):|  0.000  |  0.000  |  0.000  |  0.000  |
|    Violating Paths:|    0    |    0    |    0    |    0    |
|          All Paths:|  12273  |  11430  |   33    |  1052   |
+--------------------+---------+---------+---------+---------+

+----------------+-----------------------------------+------------------+
|                |              Real                 |       Total      |
|    DRVs        +------------------+------------+------------------+
|                |  Nr nets(terms)  | Worst Vio  |  Nr nets(terms)  |
+----------------+------------------+------------+------------------+
|   max_cap      |      0 (0)       |   0.000    |      8 (8)       |
|   max_tran     |      0 (0)       |   0.000    |      6 (6)       |
|   max_fanout   |      0 (0)       |     0      |      0 (0)       |
|   max_length   |      0 (0)       |     0      |      0 (0)       |
+----------------+------------------+------------+------------------+

Density: 46.421%
Routing Overflow: 0.00% H and 0.00% V
------------------------------------------------------------
```

File - *leon_preCTS_default.tarpt.gz* shows details of the top critical setup paths in the design.

```
Path 1: MET Setup Check with Pin proc0/cmem0/dtags0/u0/id0/CK1
Endpoint:   proc0/cmem0/dtags0/u0/id0/A1[3] (v) checked with  leading edge of
'test_clk'
Beginpoint: scan_clk                        (v) triggered by trailing edge of
'test_clk'
Path Groups: {default}
Analysis View: func_slow_max
Other End Arrival Time          0.000
- Setup                         0.263
+ Phase Shift                   8.000
+ CPPR Adjustment               0.000
= Required Time                 7.737
- Arrival Time                  7.536
= Slack Time                    0.201
     Clock Fall Edge                      4.000
     + Drive Adjustment                   0.001
     = Beginpoint Arrival Time            4.001
     Timing Path:
     +-------------------------------------------------------------------------------------------------------+
     |              Pin               | Edge |          Net          |     Cell     | Delay | Arrival | Required |
     |                                |      |                       |              |       |  Time   |   Time   |
     |--------------------------------+------+-----------------------+--------------+-------+---------+----------|
     | scan_clk                       |  v   | scan_clk              |              |       |   4.001 |    4.201 |
     | clk_div_out_mux/B              |  v   | scan_clk              | MX2X4        | 0.002 |   4.003 |    4.204 |
     | clk_div_out_mux/Y              |  v   | clk_half              | MX2X4        | 1.215 |   5.218 |    5.419 |
     | mcore0/a0/g1626/B              |  v   | clk_half              | NOR2BX2      | 0.002 |   5.220 |    5.421 |
     | mcore0/a0/g1626/Y              |  ^   | mcore0/a0/n_58        | NOR2BX2      | 0.050 |   5.270 |    5.471 |
     | mcore0/a0/g985/AN              |  ^   | mcore0/a0/n_58        | NOR2BX2      | 0.000 |   5.271 |    5.471 |
     | mcore0/a0/g985/Y               |  ^   | mcore0/a0/n_110       | NOR2BX2      | 0.121 |   5.391 |    5.592 |
     | mcore0/a0/g974/B               |  ^   | mcore0/a0/n_110       | AND2X2       | 0.000 |   5.391 |    5.592 |
     | mcore0/a0/g974/Y               |  ^   | mcdo_12               | AND2X2       | 0.182 |   5.573 |    5.774 |
     | proc0/c0/dcache0/g20437/AN     |  ^   | mcdo_12               | NOR2BX2      | 0.000 |   5.574 |    5.774 |
     | proc0/c0/dcache0/g20437/Y      |  ^   | proc0/c0/dcache0/n_957| NOR2BX2      | 0.176 |   5.750 |    5.951 |
     | proc0/c0/dcache0/g3579/A       |  ^   | proc0/c0/dcache0/n_957| INVX2        | 0.000 |   5.750 |    5.951 |
     | proc0/c0/dcache0/g3579/Y       |  v   | proc0/c0/dcache0/n_407| INVX2        | 0.076 |   5.826 |    6.027 |
     | proc0/c0/dcache0/g20016/B      |  v   | proc0/c0/dcache0/n_407| AND2X2       | 0.000 |   5.826 |    6.027 |
     | proc0/c0/dcache0/g20016/Y      |  v   | proc0/c0/dcache0/n_602| AND2X2       | 0.107 |   5.934 |    6.134 |
     | proc0/c0/dcache0/g21030/A      |  v   | proc0/c0/dcache0/n_602| OR3X2        | 0.000 |   5.934 |    6.135 |
     | proc0/c0/dcache0/g21030/Y      |  v   | proc0/c0/dcache0/n_677| OR3X2        | 0.233 |   6.166 |    6.367 |
     | proc0/c0/dcache0/g20442/AN     |  v   | proc0/c0/dcache0/n_677| NOR2BX2      | 0.000 |   6.167 |    6.367 |
     | proc0/c0/dcache0/g20442/Y      |  v   | proc0/c0/dcache0/n_441| NOR2BX2      | 0.098 |   6.265 |    6.465 |
     | proc0/c0/dcache0/g20782/C      |  v   | proc0/c0/dcache0/n_441| OR3X2        | 0.000 |   6.265 |    6.465 |
     | proc0/c0/dcache0/g20782/Y      |  v   | proc0/c0/dcache0/n_785| OR3X2        | 0.218 |   6.483 |    6.683 |
     | proc0/c0/dcache0/g18988/B      |  v   | proc0/c0/dcache0/n_785| AND2X2       | 0.000 |   6.483 |    6.684 |
     | proc0/c0/dcache0/g18988/Y      |  v   | proc0/c0/dcache0/n_959| AND2X2       | 0.139 |   6.622 |    6.823 |
     | proc0/c0/dcache0/g20689/A1     |  v   | proc0/c0/dcache0/n_959| AO22X2       | 0.000 |   6.622 |    6.823 |
     | proc0/c0/dcache0/g20689/Y      |  v   | proc0/c0/dcache0/n_830| AO22X2       | 0.200 |   6.822 |    7.023 |
     | proc0/c0/dcache0/FE_RC_43_0/B  |  v   | proc0/c0/dcache0/n_830| NOR2X2       | 0.000 |   6.823 |    7.023 |
     | proc0/c0/dcache0/FE_RC_43_0/Y  |  ^   | proc0/c0/dcache0/n_861| NOR2X2       | 0.301 |   7.123 |    7.324 |
     | proc0/c0/dcache0/g3475/A       |  ^   | proc0/c0/dcache0/n_861| INVX2        | 0.006 |   7.129 |    7.330 |
     | proc0/c0/dcache0/g3475/Y       |  v   | proc0/crami_47        | INVX2        | 0.398 |   7.527 |    7.728 |
     | proc0/cmem0/dtags0/u0/id0/A1[3]|  v   | proc0/crami_47        | MEM2_128X32  | 0.009 |   7.536 |    7.737 |
     +-------------------------------------------------------------------------------------------------------+
     Clock Rise Edge                      0.000
     = Beginpoint Arrival Time            0.000
     Other End Path:
```

Other files which are generated and saved under directory timingReports are:

```
leon_preCTS.cap            - max capacitance violations
leon_preCTS.fanout         - max fanout violations
leon_preCTS.slk            - slack report for each endpoint
leon_preCTS.summary        - timing summary
leon_preCTS.tran           - max transition violations
leon_preCTS_all.tarpt      - timing report for all path groups
leon_preCTS_in2out.tarpt   - report for input to output paths
leon_preCTS_clkgate.tarpt  - report for clock gating paths
leon_preCTS_in2reg.tarpt   - report for input to register paths
leon_preCTS_reg2out.tarpt  - report for register to output paths
leon_preCTS_reg2reg.tarpt  - report for register to register paths
```

4. After running timing analysis you typically want to explore and debug certain timing paths. Run `report_timing` to report timing for the worst path.

```
report_timing
```

`report_timing` reports timing for the worst path by default. You can report specific paths using the `report_timing` options. Type `report_timing -help` for a complete list of its options.

5. Close the metric track for pre_cts by running the following command

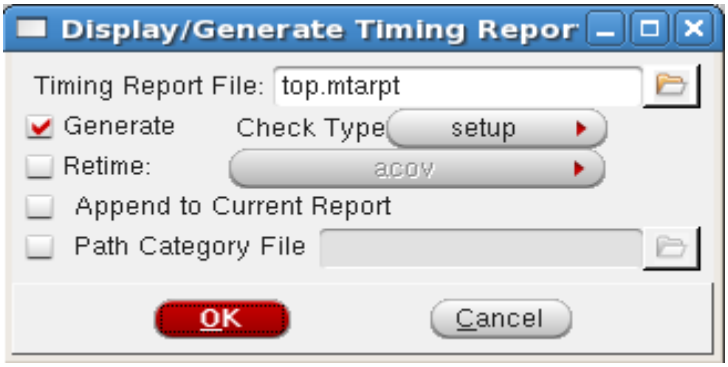
```
pop_snapshot_stack
```

6. Create the snapshot using unified metrics command

```
create_snapshot -name pre_CTS
```

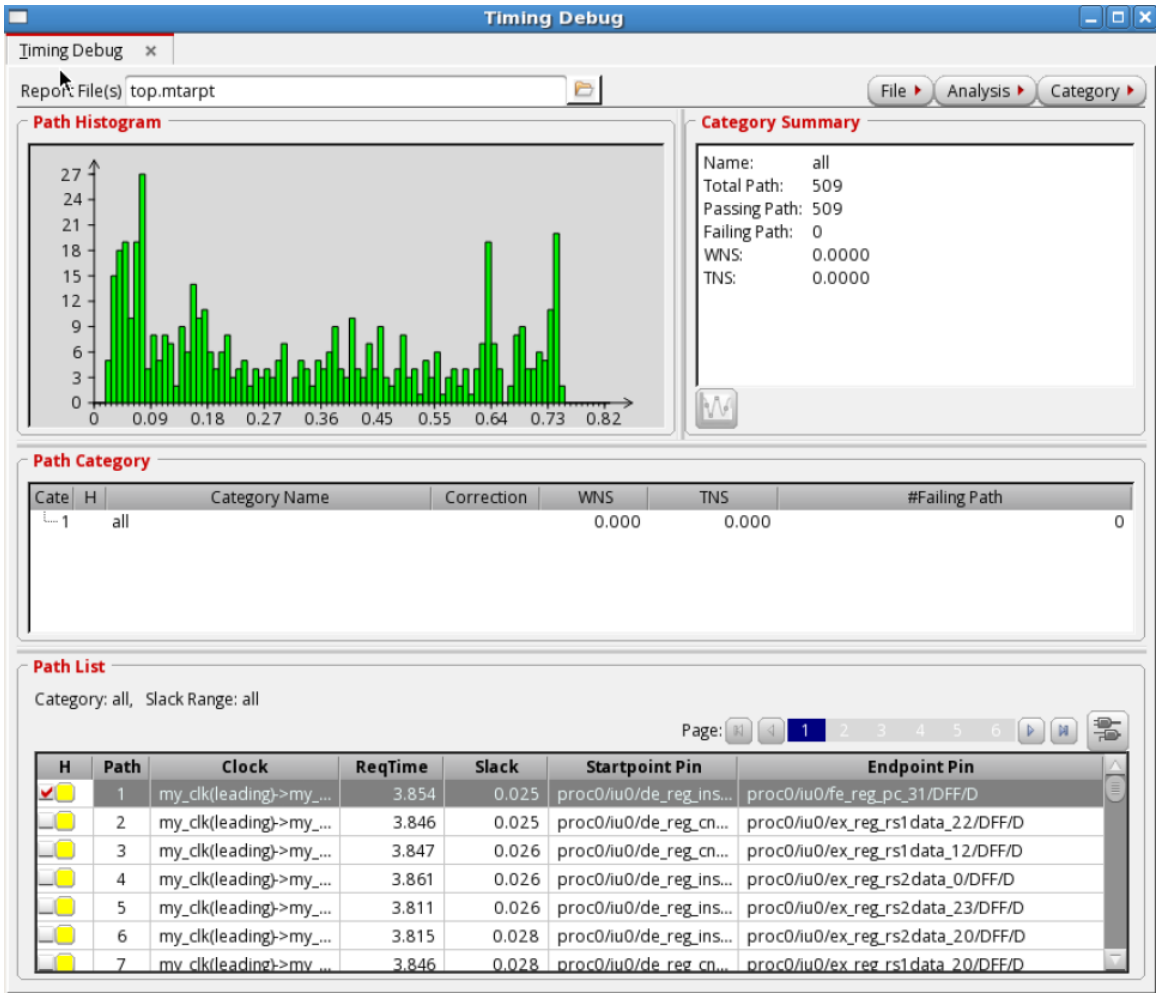
The **Global Timing Debug Browser** provides a visual display of the timing paths. This browser is a powerful debug tool providing cross-probing between the report information and the Innovus design display area. You can group timing violations into categories making it easier to debug timing paths in groups rather than individual paths. Perform the following to open the Global Timing Debug Browser:

2. Select **Timing - Debug Timing**



3. Click **OK**.

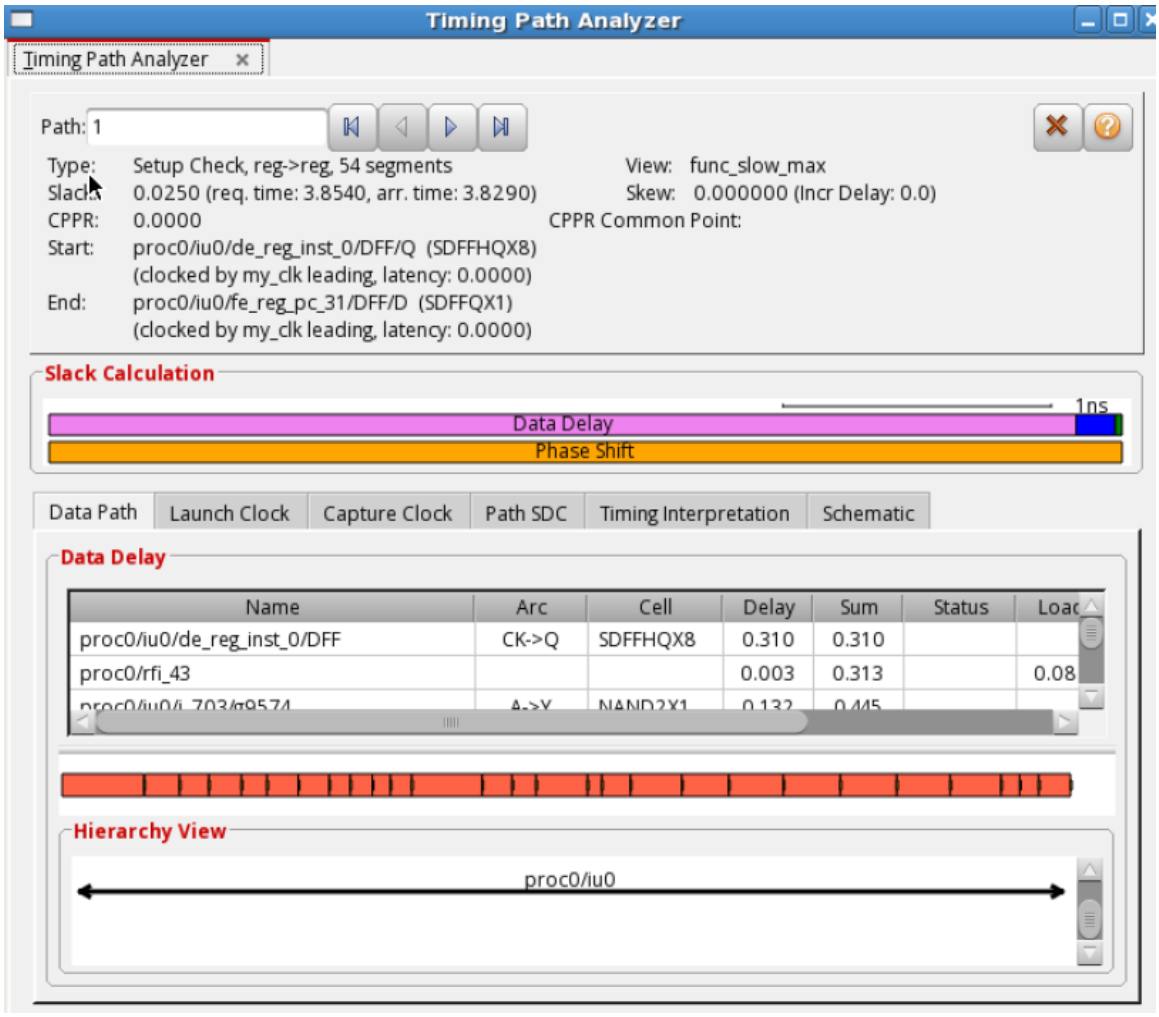
A Timing Debug form displays path histogram and a Timing Browser displays in the Innovus main window.



The Global Timing Debug interface is a powerful tool for visualizing and debugging timing issues. Features include:

- **Path Histogram** - Displays overall timing picture
- **Category Summary** – Summary of selected category
- **Path Category** – List of current categories. Customer categories are created using the **Category** menu, or based on predefined criteria and analysis using the **Analysis** menu.
- **Path List** – Shows paths in selected category. Double-click a path to open it in the Path Analyzer
- **Cross probing** - Selected paths are highlighted in the design

3. Double-click the first path in the **Path List** to open it in the Path Analyzer.



The **Path Analyzer** shows details of the path including:

- Header shows start point/end point, slack, clock edges involved, skew and view.
- Slack Calculation shows components of slack calculation for launch and capture paths.
- Use tabs in the center of the form to show different details of the path. Selecting an instance or net highlights it in the GUI.
- Data delay part shows proportional delay for each instance and net in the path. Hover over a bar to show delay value and percentage of total path delay.
- Hierarchy View shows hierarchical boundaries the path crosses.

This tutorial does not go into detail on using Global Timing Debug but the Rapid Adoption Kit (RAK) titled *Global Timing Debug (GTD) using Innovus or Tempus* takes you through a lab showing the details of debugging timing using GTD.

4. Close the Timing Debug window(s).

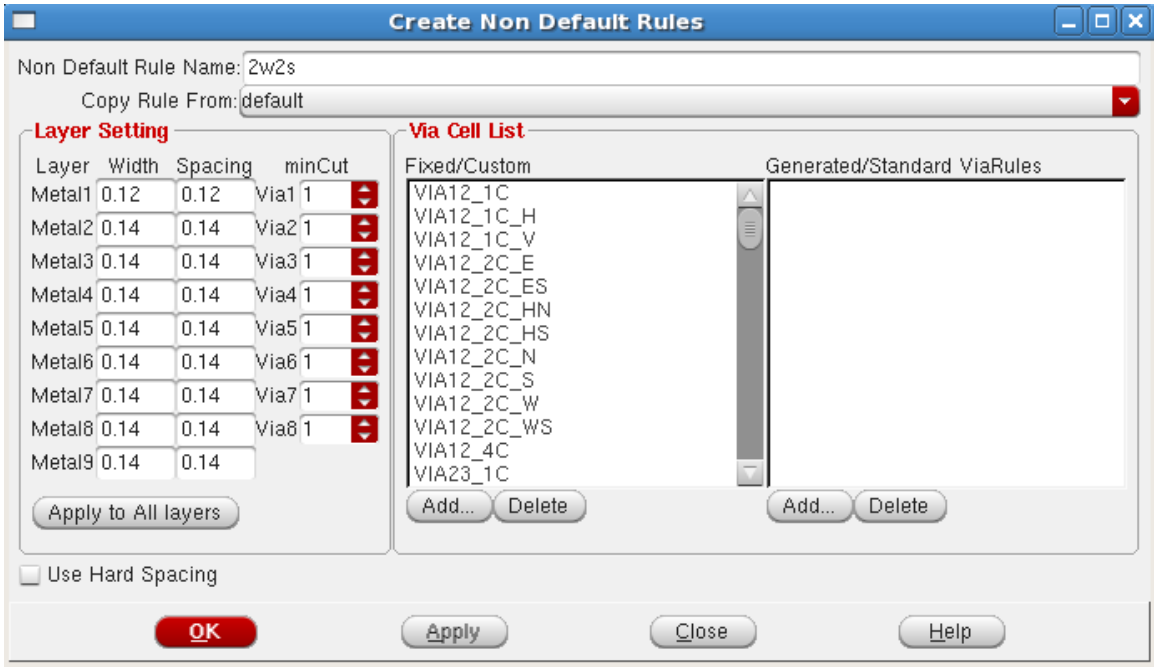
3-7 Running Clock Tree Synthesis (CTS)

Clock Tree Synthesis using CCOpt-CTS is run to build a clock tree which delivers the clock signal to the sequential elements within the specified constraints. To synthesize the clock tree you will:

- Define a non-default rule (NDR) that has double the width and spacing of the default rule. This will be used to route the clocks.
- Configure CCOpt and generate a CCOpt specification file.
- Run CCOpt-CTS to build the clock tree.

1. Select **Edit – Create Non Default Rule**

2. On the **Create Non Default Rules** form you can seed the width and spacing fields with the values of an existing rule. The default width and spacing are shown by default. Specify the rule name 2w2s and specify double the width and spacing of the default rule as shown below:



3. Click **OK** to save the rule.

4. Create a route type to define the NDR and layers to use for routing the clock tree:

```
create_route_type -name clkroute -non_default_rule 2w2s \
                  -bottom_preferred_layer Metal5 \
                  -top_preferred_layer Metal6
```

5. Specify this route type should be used for trunk and leaf nets:

```
set_ccopt_property route_type clkroute -net_type trunk
set_ccopt_property route_type clkroute -net_type leaf
```

6. Specify the buffers, inverters and clock gating cells to use:

```
set_ccopt_property buffer_cells {CLKBUFX8 CLKBUFX12}
set_ccopt_property inverter_cells {CLKINVX8 CLKINVX12 }
set_ccopt_property clock_gating_cells TLATNTSCA*
```

7. Generate the CCOpt spec file and source it:

```
create_ccopt_clock_tree_spec -file ccopt.spec
source ccopt.spec
```

8. Start a new metric track using the command

```
push_snapshot_stack
```

9. Run CCOpt-CTS:

```
ccopt_design -cts
```

10. Save the design by typing the following at the Innovus command prompt:

```
saveDesign DBS/cts.enc
```

3-8 Running Post-CTS Timing Optimization

At this point the clock tree has been synthesized and we can analyze timing based on the actual clock tree delay and skew.

1. Run post-CTS timing analysis by typing the following command in the Innovus console:

```
timeDesign -postCTS
```

2. If timing violations exist run the following to fix them:

```
optDesign -postCTS
```

3. Run post-CTS hold timing analysis by typing the following command in the Innovus console:

```
timeDesign -postCTS -hold
```

A small number of hold violations likely exist because hold fixing has not been performed. If there were a large number of hold violations you would want to perform hold fixing using `optDesign -postCTS -hold`. But since there is only a small number we will wait to fix them during post-route optimization.

4. Close the metric track by running the command

```
pop_snapshot_stack
```

5. Create a snapshot by running the unified metrics command

```
create_snapshot -name post_CTS
```

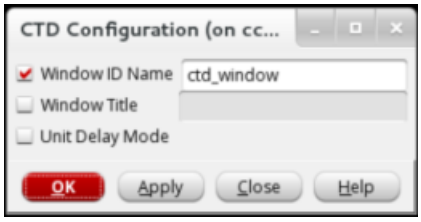
6. Save the design:

```
saveDesign DBS/postcts.enc
```

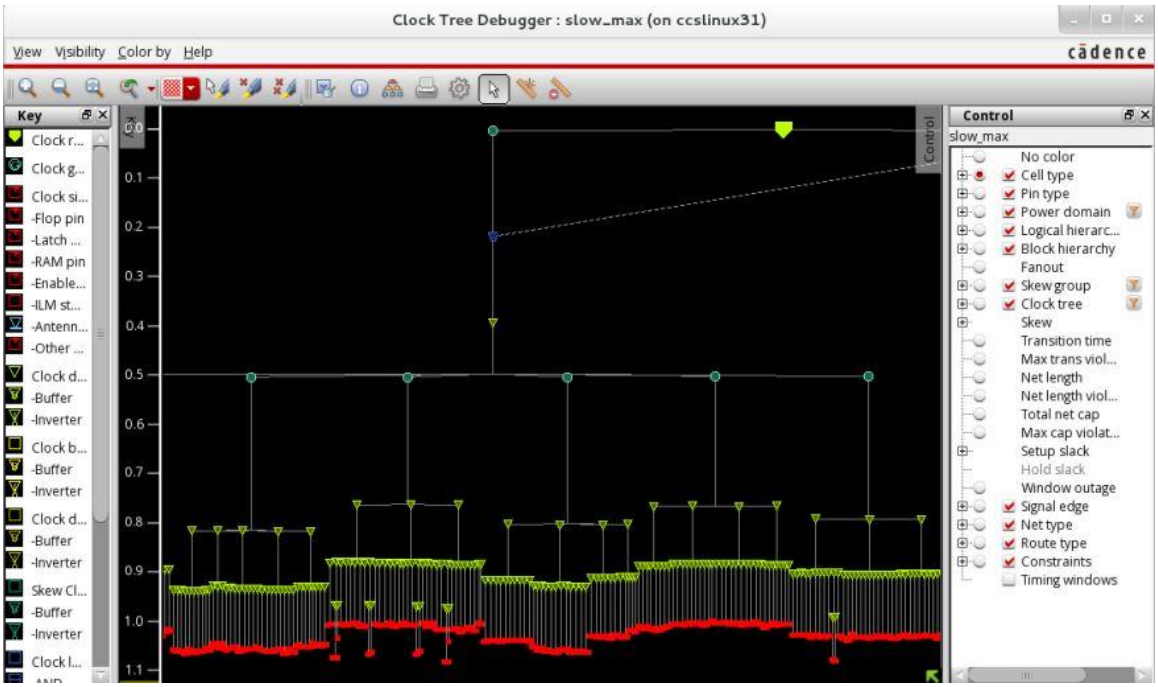
3-9 Running CCOpt Clock Tree Debugger (CTD)

The CCOpt Clock Tree Debugger is a graphical tool for analyzing and debugging the clock tree results.

1. Select **Clock - CCOpt Clock Tree Debugger**. The CCOpt Clock Tree Debugger appears.
2. The dialog box **CTD Configuration** is shown. Click **OK** without changing any settings.

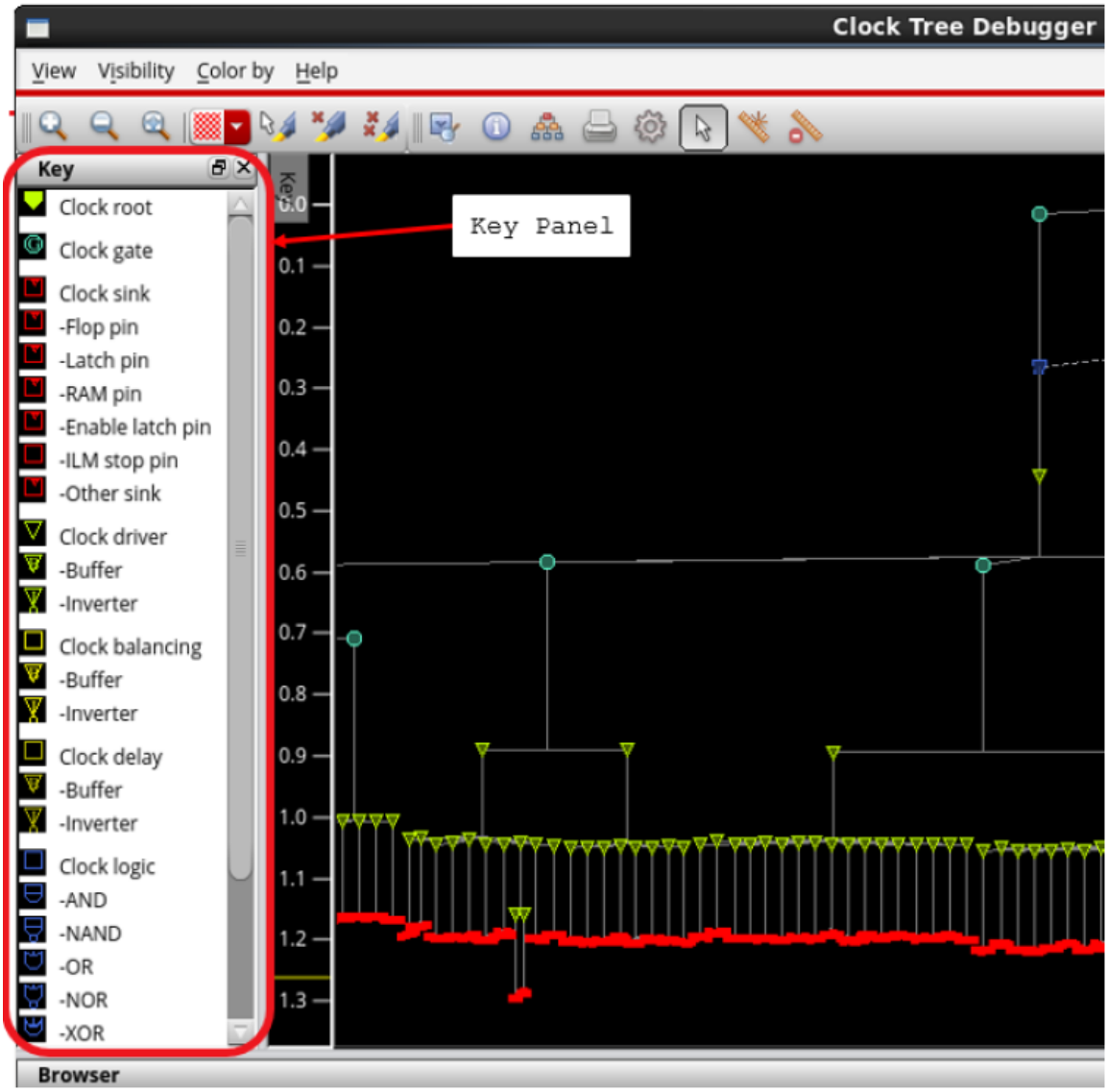


3. In **view - enable browser**.



The debugger shows the logic in the clock tree with the clock roots at the top and the leaf cells at the bottom. Analysis is performed by coloring the clock tree based on selected criteria such as objects, paths, groups, and so on.

4. Display the Key Panel by selecting **View -Key Panel** or by selecting the Key tab in the upper-left corner.

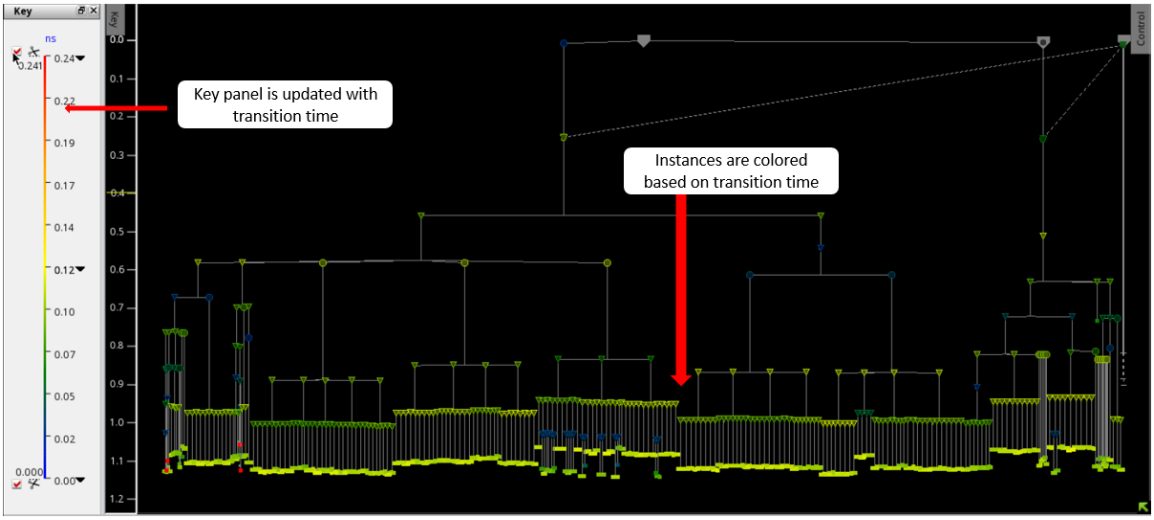


Initially, the clock tree is colored based on the cell type. Use the **Key Panel** to determine the symbols that represent each cell type.

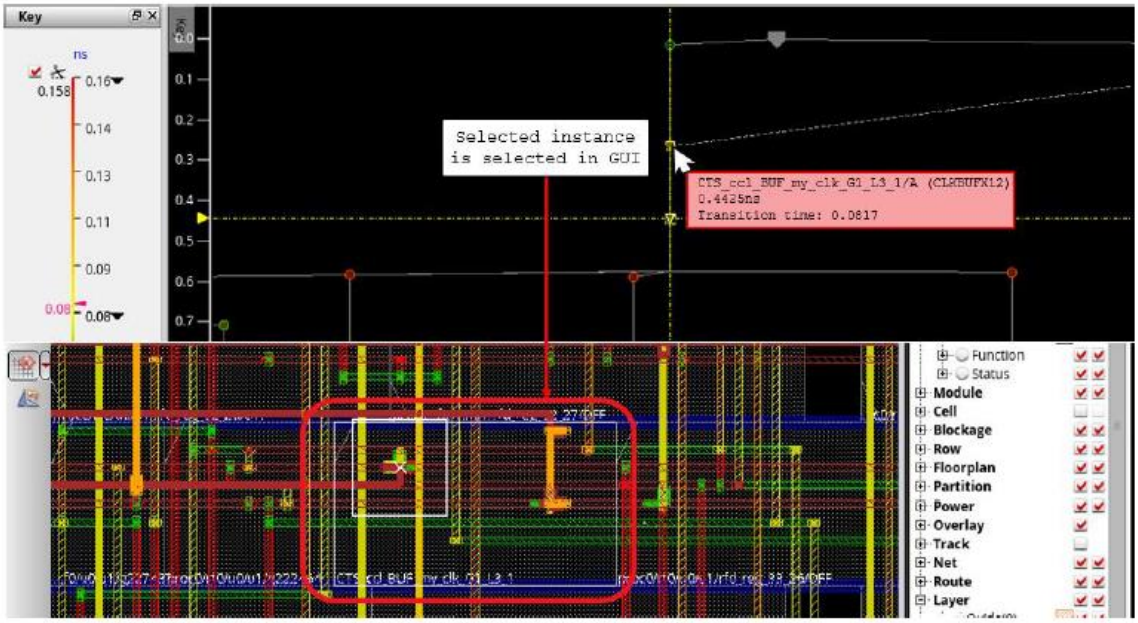
The Visibility menu filters which colored objects are to be made visible.

5. Enable the check box next to **Visibility - Cell type -Clock sink** to disable the coloring of all the clock sinks.

6. Select **Color by - Transition Time** to color the clock tree based on the transition time arrival at each cell.



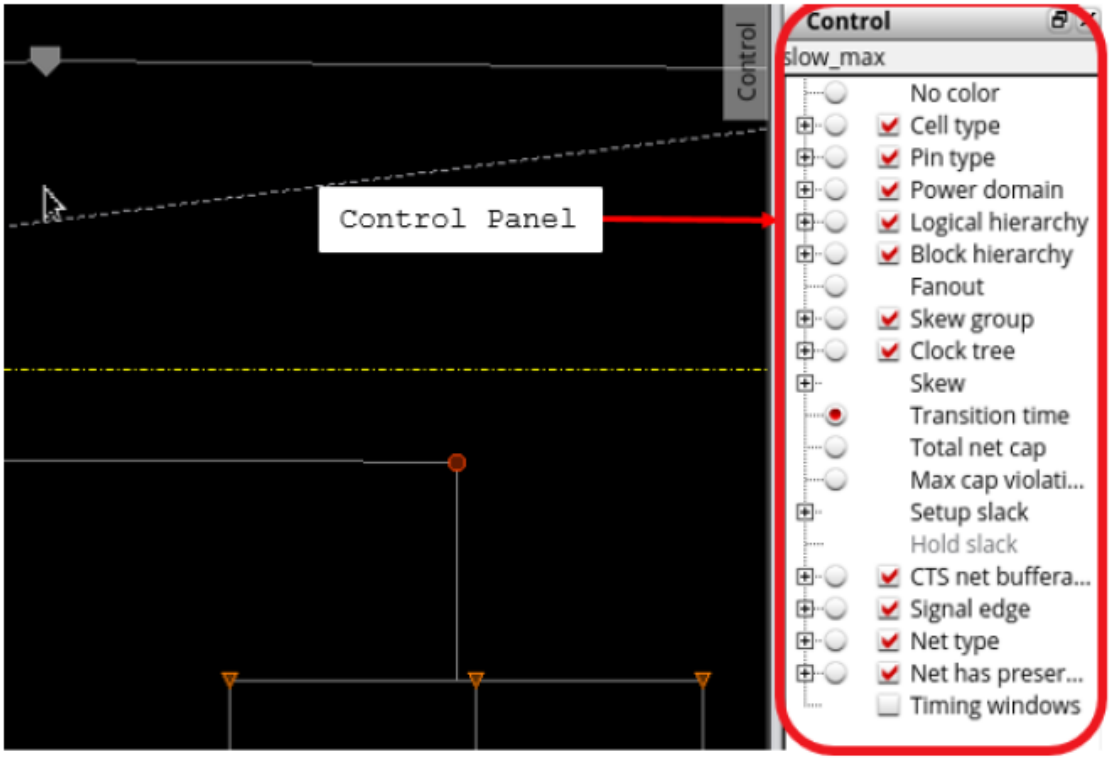
7. Select an instance or net in the debugger and observe its selection in the Innovus Implementation System GUI.



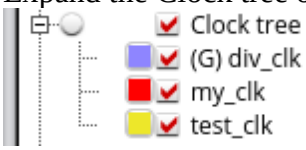
Tip: Cross-probing can be enabled/disabled using **View-Select -Enable crossprobing.**

The Control Panel combines the functionality of the Visibility and Color by menus into a single form.

8. Select **View - Control Panel** or click the **Control tab** in the upper-right to open the Control Panel.

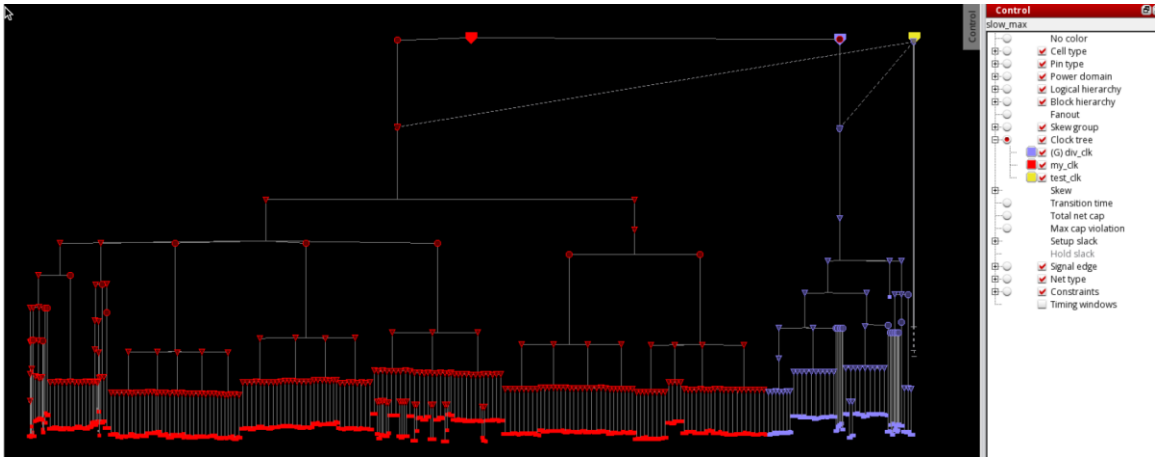


9. Expand the Clock tree object by clicking the '+' symbol next to it.

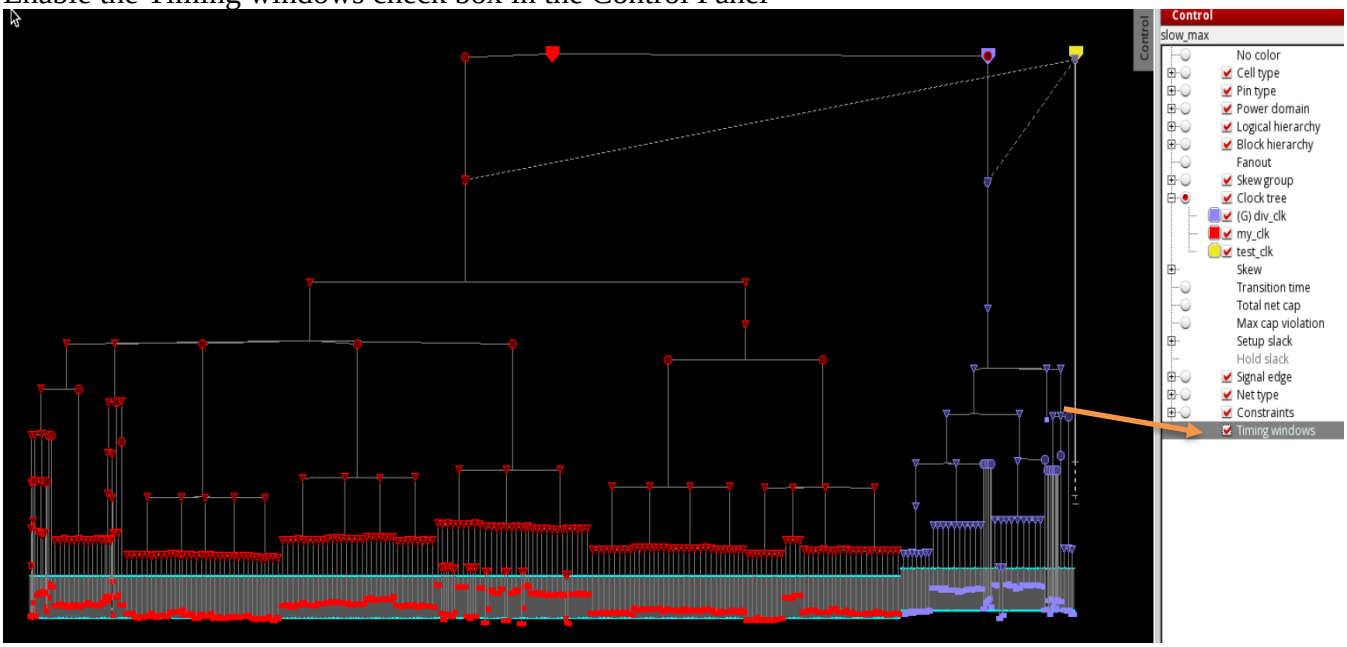


The radio buttons specify what the coloring is based on. You can change the color manually.

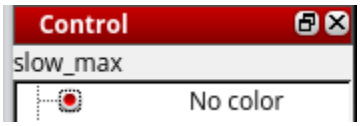
10. Enable the Clock tree radio button. The clock tree will be colored with the default color.



11. Enable the Timing windows check box in the Control Panel

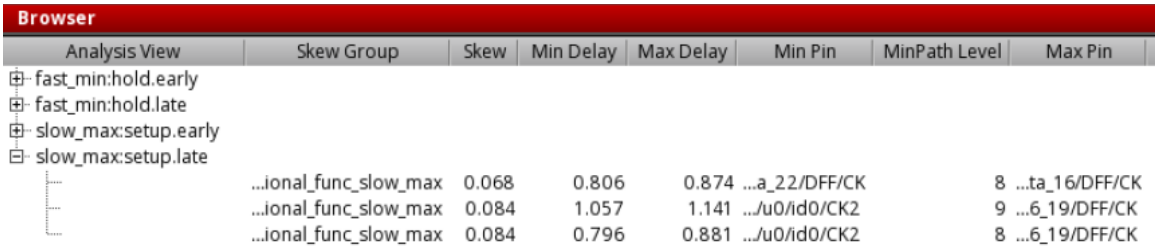


12. Select No Color in the Control Panel in preparation for the next steps.

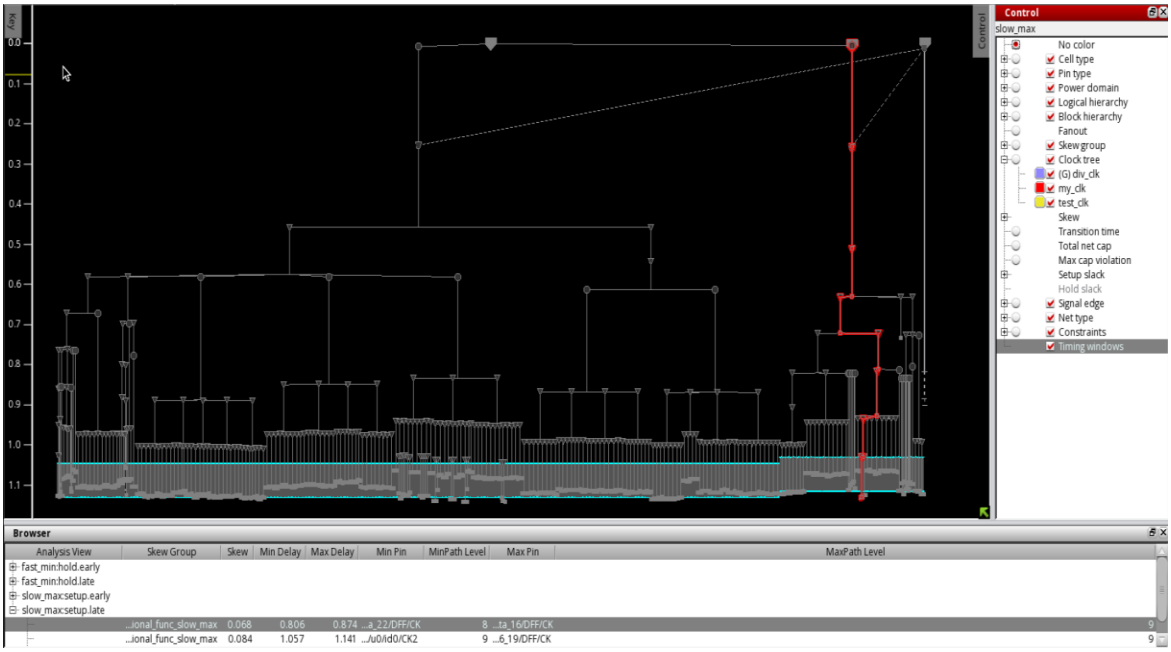


13. The Clock Path Browser displays the clock path data in a table and provides the option for bringing up a clock path analyzer. This is done either from its context menu or by double-clicking on a row in the table.

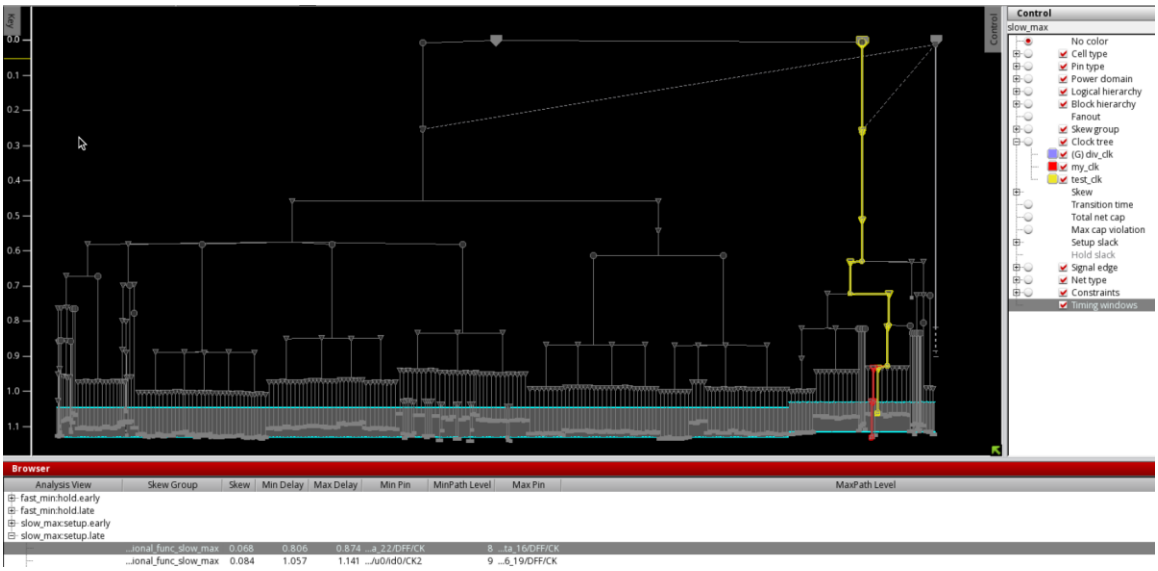
14. In the Browser, minimize the Analysis Views except for slow_max:setup:late, by clicking the '-' symbol



15. Click the RMB on my_clk/functional_func_slow_max and select a color under **Highlight - Max Path.**



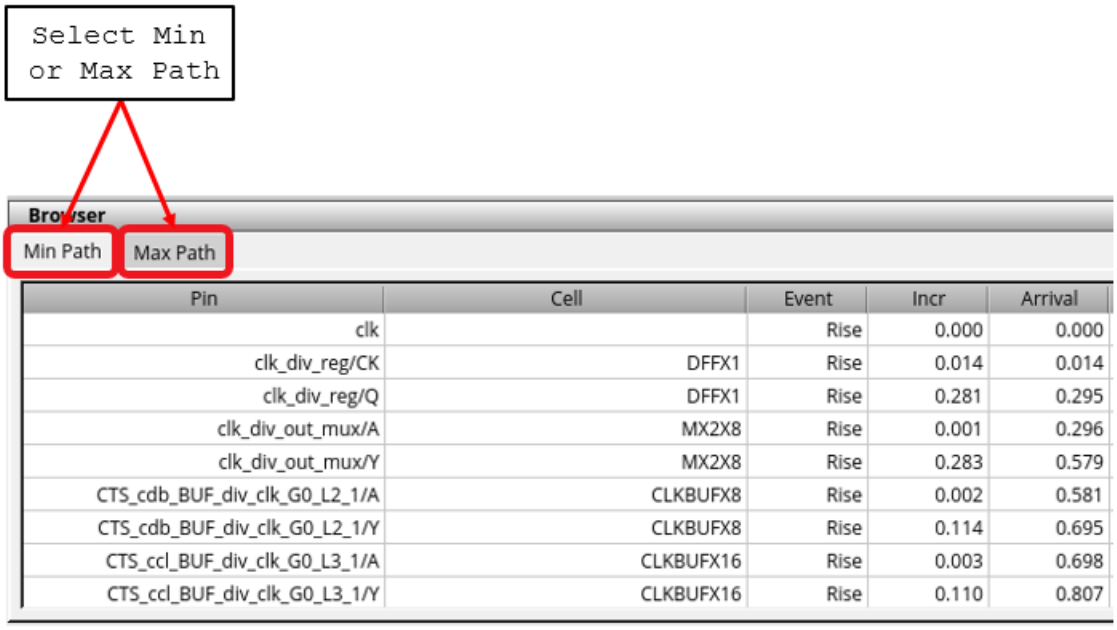
16. Highlight the minimum path in a similar way by clicking the RMB on my_clk/functional_func_slow_max and selecting a color under **Highlight -Min Path**.



17. Clear the highlighting by clicking the RMB in the display area and selecting **Dehighlight All.**

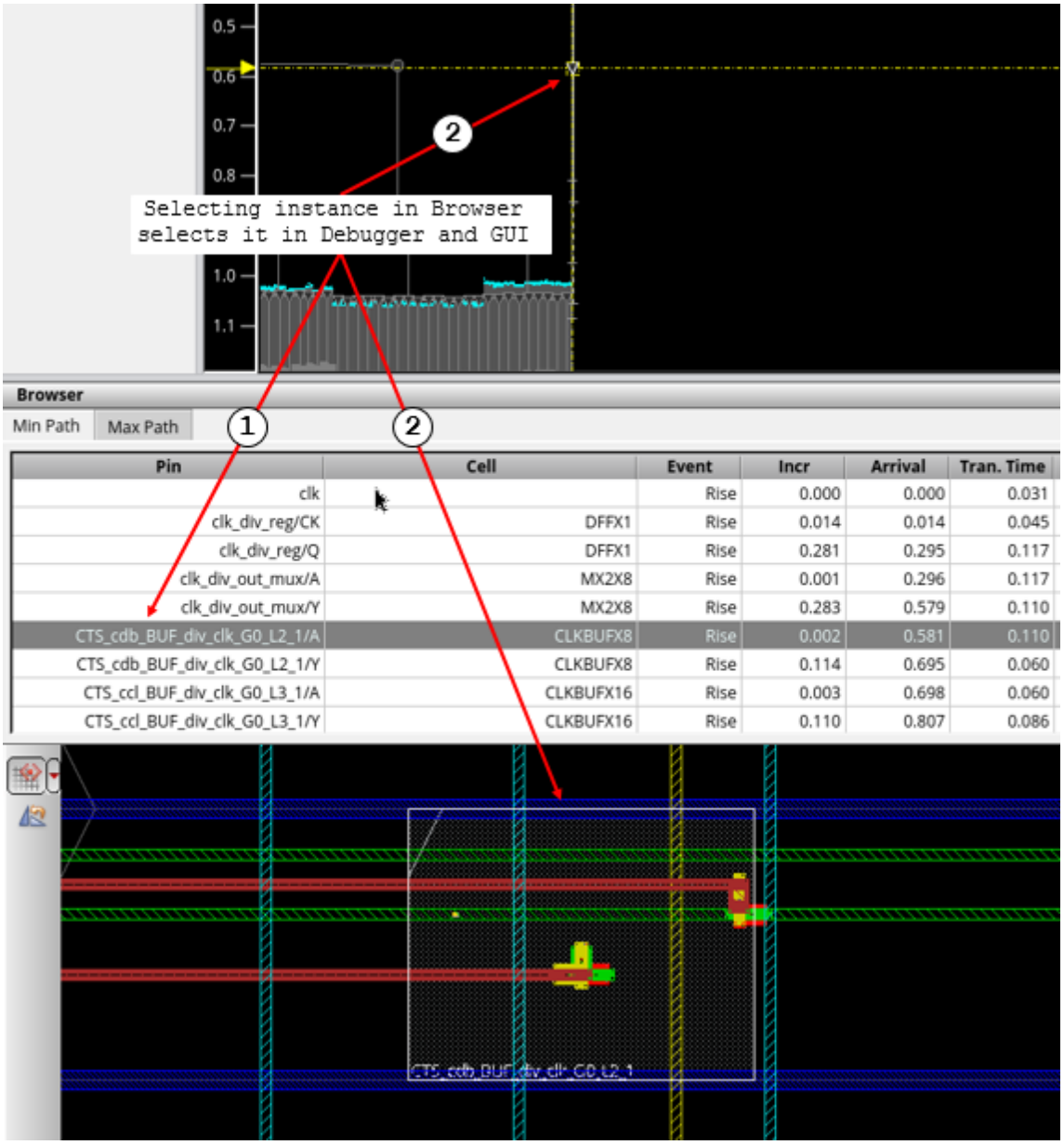
18. The Path Browser enables you to see the detailed path delays for the min and max paths

- Double-click on my_clk/functional_func_slow_max in the Browser to drill down and view the details of the min and max paths.



Pin	Cell	Event	Incr	Arrival
clk		Rise	0.000	0.000
clk_div_reg/CK	DFFX1	Rise	0.014	0.014
clk_div_reg/Q	DFFX1	Rise	0.281	0.295
clk_div_out_mux/A	MX2X8	Rise	0.001	0.296
clk_div_out_mux/Y	MX2X8	Rise	0.283	0.579
CTS_cdb_BUF_div_clk_G0_L2_1/A	CLKBUFX8	Rise	0.002	0.581
CTS_cdb_BUF_div_clk_G0_L2_1/Y	CLKBUFX8	Rise	0.114	0.695
CTS_ccl_BUF_div_clk_G0_L3_1/A	CLKBUFX16	Rise	0.003	0.698
CTS_ccl_BUF_div_clk_G0_L3_1/Y	CLKBUFX16	Rise	0.110	0.807

19. Select an instance in the path and observe that it is highlighted in the display.



3-11 Running NanoRoute

Timing is not closed on a design unless crosstalk is prevented, analyzed, and then fixed. To really achieve the above, the design is run with global and detail routing using NanoRoute. When running NanoRoute, we enable both the Timing Driven and SI Driven (Signal Integrity) options in the NanoRoute form. These options are important in helping to close timing and preventing crosstalk.

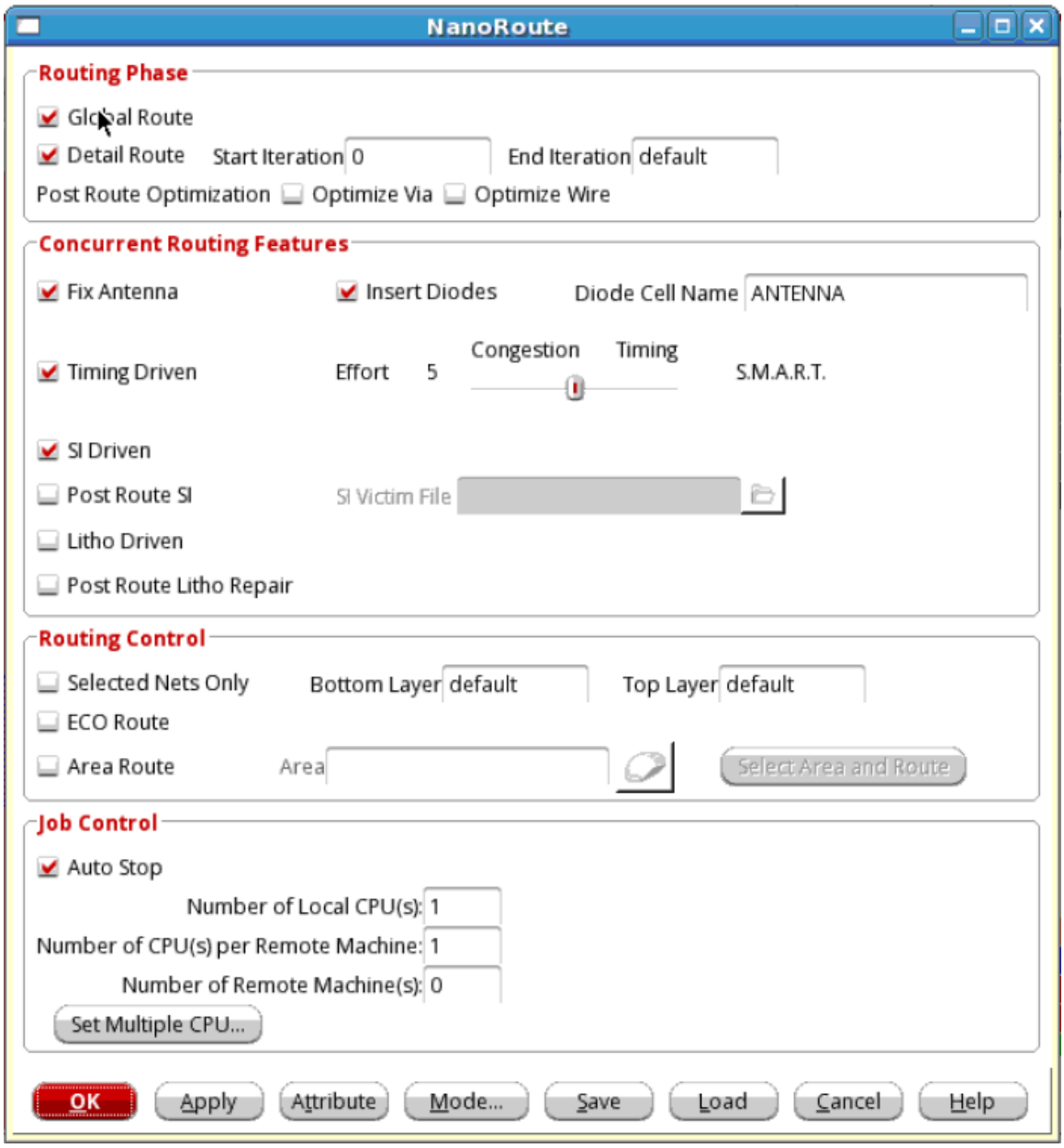
Run NanoRoute in Timing and SI driven mode:

1. Start a new metric track using the command

   ```
   push_snapshot_stack
   ```

2. Select **Route - NanoRoute - Route**

3. In the Concurrent Routing Features section,
   - Select – **Fix Antenna**
   - Select – **Timing Driven**
   - Select – **SI Driven**
   - Select – **Insert Diodes**
   - Enter Diode Cell Name as **ANTENNA**



4. Click **OK**.

Running NanoRoute with the SI Driven option requires the capacitance table file (or QRC technology file), and this file was read in during design import. Choosing SI Driven option in NanoRoute is our first line of defense against noise.

5. Save the design:

```
saveDesign DBS/route.enc
```

3-12 Post-Route Timing and SI Optimization

The design is fully routed and timing analysis should be run to analyze timing based on the actual routes.

At the Innovus prompt run Post-Route timing analysis to report any setup or hold violations. The following increases the RC extraction effort level to medium so turbo-QRC is run which correlates better to signoff extraction. Note an effort level of medium or higher requires a QRC technology file and effort level of `high` or `signoff` also requires a QRC license.

1. Analyze post-route setup timing:

```
setExtractRCMode –engine postRoute
setExtractRCMode –effortLevel medium
timeDesign –postRoute
```

2. Analyze post-route hold timing:

```
timeDesign –postRoute –hold
```

3. Run setup and hold optimization simultaneously to fix any remaining timing violations:

```
optDesign –postRoute –setup –hold
```

   A small number of violations may remain. If so, run `optDesign` again as above and they should be resolved.

4. Close the metric track by running the command

```
pop_snapshot_stack
```

5. Create a snapshot by running the unified metrics command

```
create_snapshot –name post_ROUTE
```

6. Save the design:

```
saveDesign DBS/postroute.enc
```

7. We can generate the reports from the specified metrics by running the following command

```
report_qor -file metrics.html -format html
```

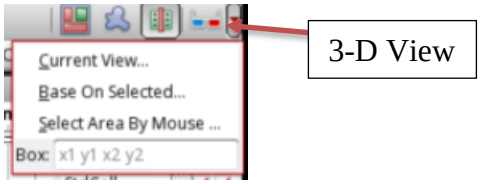
8. Open the `metrics.html` file.

Snapshot	Setup (all)			Setup (reg2reg)			Hold (all)			Hold (reg2reg)			DRV			Clock		Design			Power	Route		Tool		
	WNS (ns)	TNS (ns)	FEPS	WNS (ns)	TNS (ns)	FEPS	WNS (ns)	TNS (ns)	FEPS	WNS (ns)	TNS (ns)	FEPS	Tran	Load	Fanout	Insts	Area (um^2)	Density (%)	Insts	Area (um^2)	Leakage (mW)	DRC	WL (um)	Errors	Wall (s)	Memory (MB)
pre_CTS	0.019	0	0	0.019	0	0							6	8	0			46.4	35,465	391,639				e2 w:0	1:09:25	1,646
post_CTS	0.008	0	0	0.008	0	0	-0.036	-1	19	0.071	0	0	0	0	0	322	1,714.446	46.8	35,757	393,131		0	70,403		0:13:50	1,943
post_ROUTE	0.140	0	0	0.140	0	0	0.000	0	0	0.072	0	0	0	0	0	322	1,719.576	46.8	35,794	393,178		0	1,356,559	e1 w:0	0:23:52	2,319
min	0.008	0	0	0.008	0	0	-0.036	-1	0	0.071	0	0	0	0	0	322	1,714.446	46.4	35,465	391,639		0	70,403		0:13:50	1,646
avg	0.056	0	0	0.056	0	0	-0.018	-0	10	0.071	0	0	2	3	0	322	1,717.011	46.7	35,672	392,650		0	713,481		0:35:42	1,969
max	0.140	0	0	0.140	0	0	0.000	0	19	0.072	0	0	6	8	0	322	1,719.576	46.8	35,794	393,178		0	1,356,559		1:09:25	2,319
graph																										

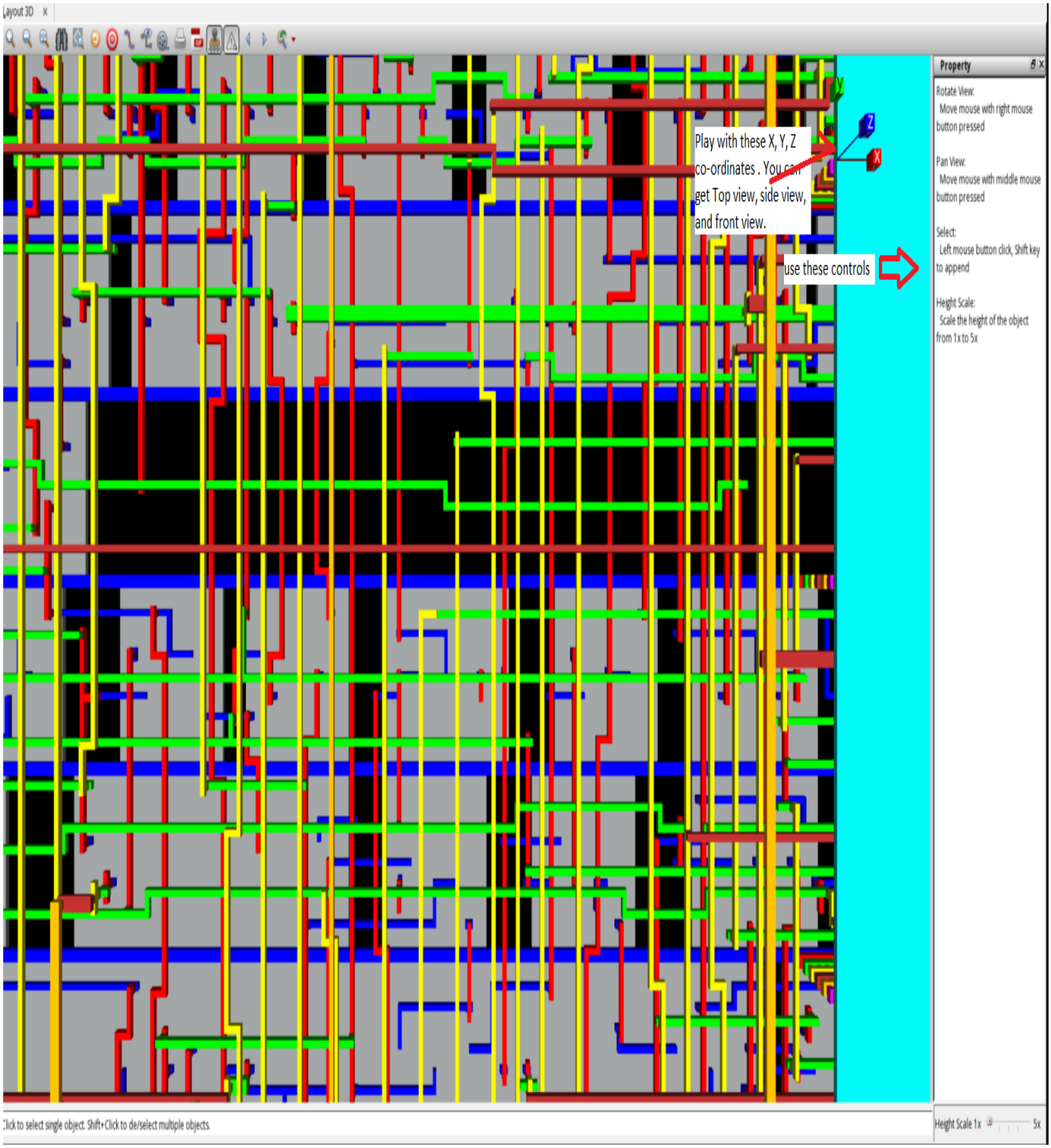
Stylus runtime summary

9. New 3D layout View is added to complement 2D display and to analyze more complicated issues like multi-DRC and so on.

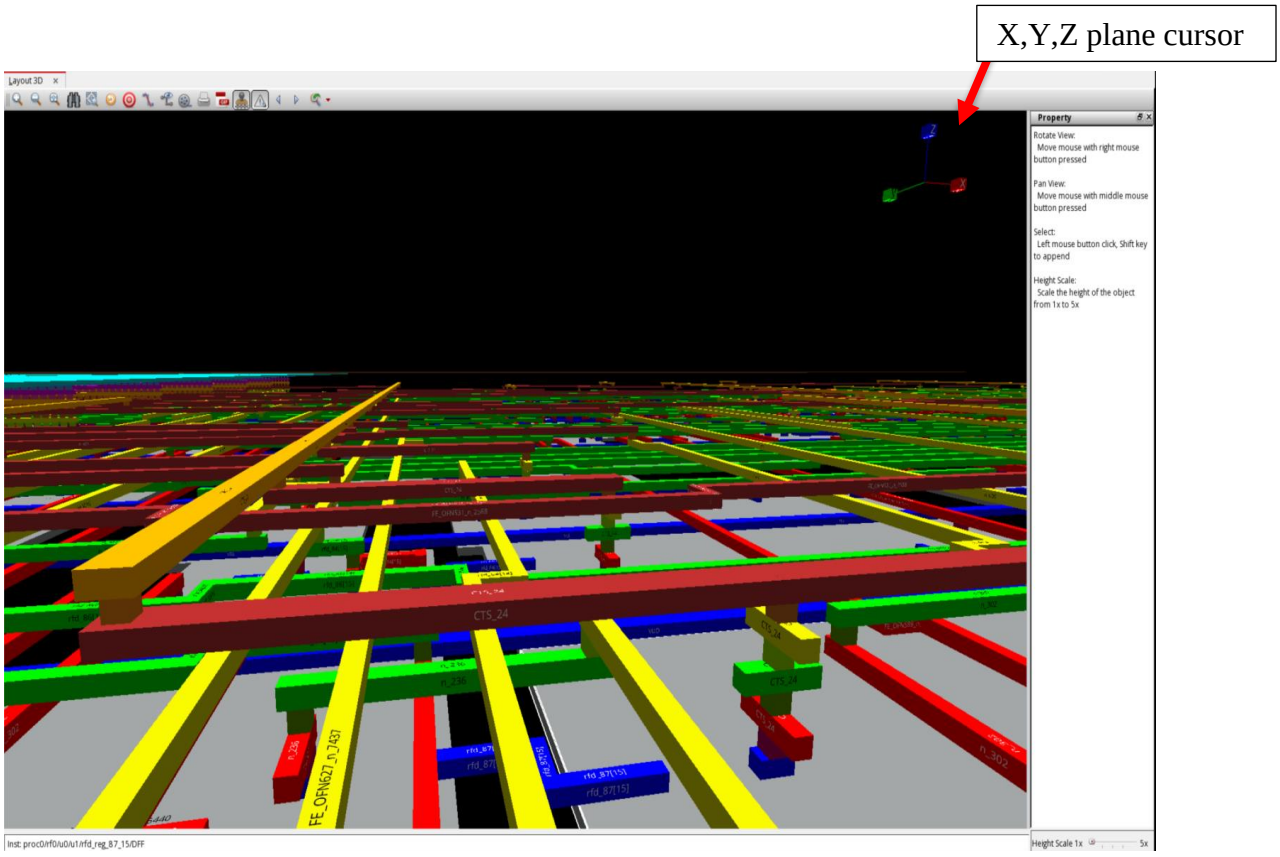
10. First select specific area which you want to open in 3D view using RMB. Next, on the right-top there select the 3D view option. In the dropdown, choose **Base on selected**



11. This will open the 3D view. Use Arrow keys to move top, bottom and Middle Mouse button to zoom in.



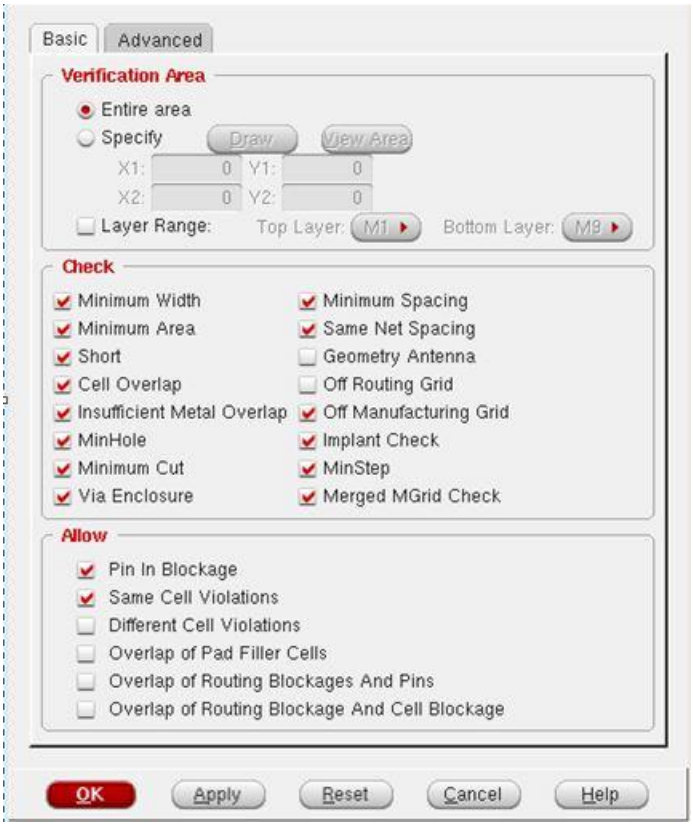
12. Move the X,Y,Z cursor on the right side top to see different views of the selected area. Below is the side view of the selected area.



3-13 Verifying the Design

The Verify menu provides several commands for verifying different aspects of the design.

1. Select **Verify – Verify Geometry** to verify physical design rules. **Verify Geometry** is used for designs above 20nm. For 20nm and below design use **Verify DRC**.

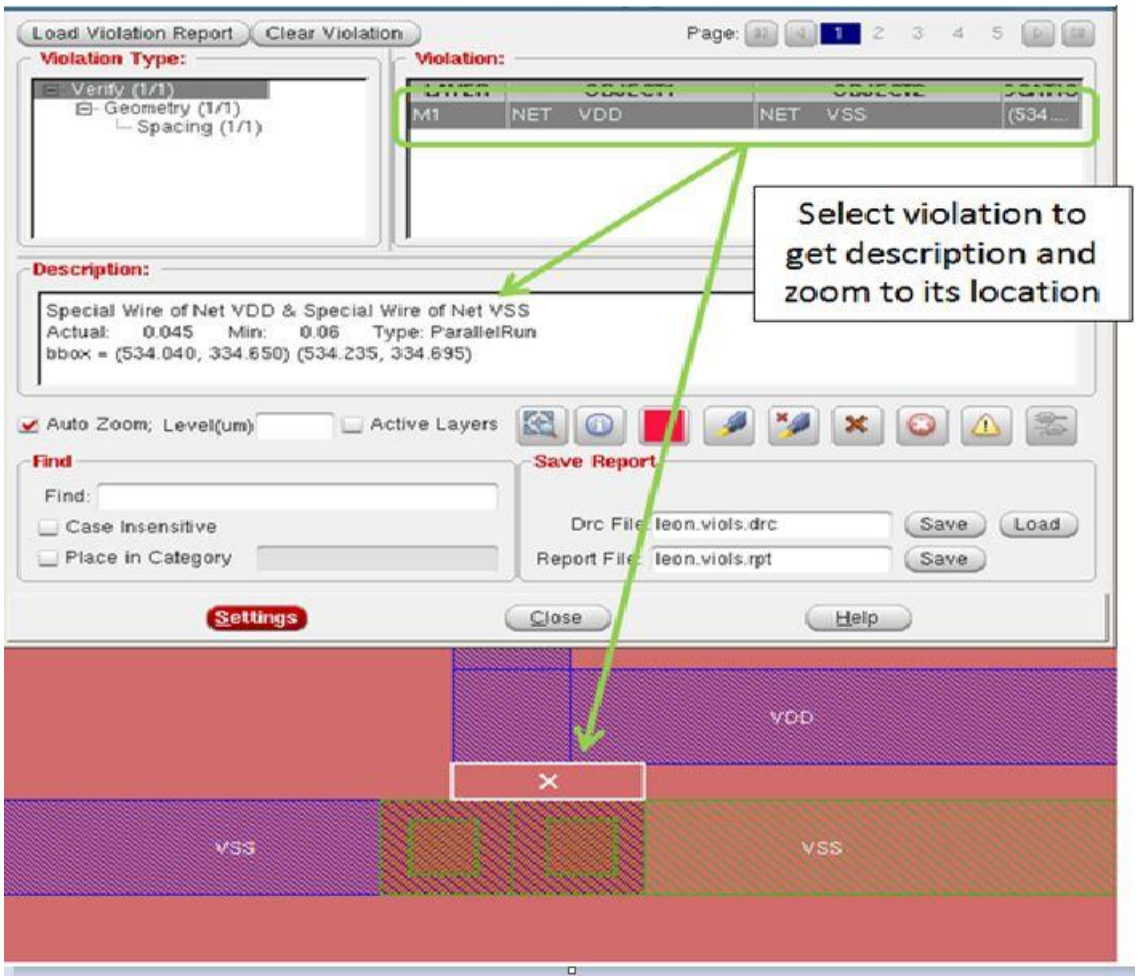


2. Click **OK**.

Review the summary in the console once Verify Geometry completes. Are there any violations reported? Use the **Violation Browser** to debug the violation.

3. Select **Tools – Violation Browser**.

4. Select the violation in the Violation Browser as shown below to zoom to the violation.



Observe a description of the violation is shown in the Violation Browser indicating the actual and required minimum spacing.

Tips:

- Double-clicking a violation marker in the GUI will automatically open the Violation Browser with that violation selected.
- Select **Load Violation Report** to load a violation report from a physical verification tool for debugging in Innovus.

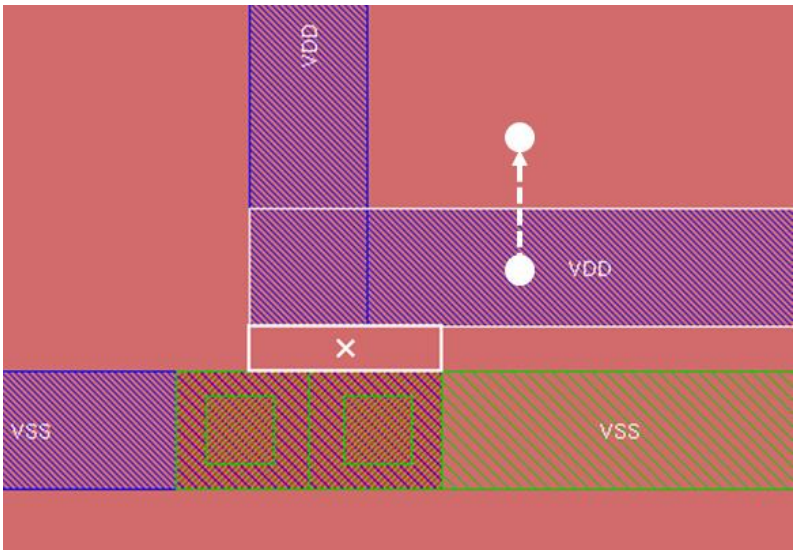
Do the following to manually move the wire and fix the violation.

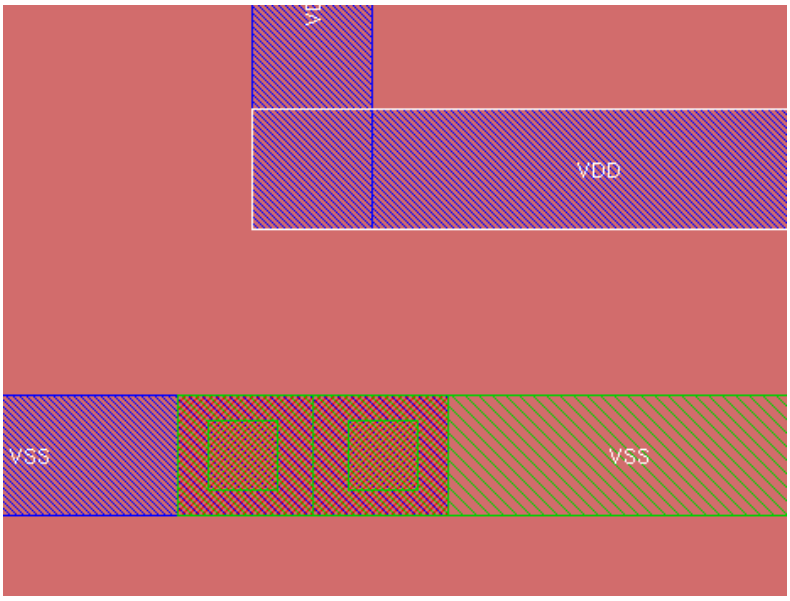
5. Select the **Move Wire** widget  or press **M** key.

6. Do the following to move the wire:

   - Click LMB once on the upper wire to select it.
   - Click LMB again to begin the move.

- Move the wire up and click LMB again to complete the move. See pictures below to show the move and the result.





7. Press **A** key to return to selection mode. Close the Violation Browser.

3-14 Exporting Design Data

The **File – Save** menu is used to export design data such as GDS, Verilog, etc. Export a GDS file.

1. Select **File – Save – GDS/Oasis…**

    - Specify **Output File:** *leon.gds*

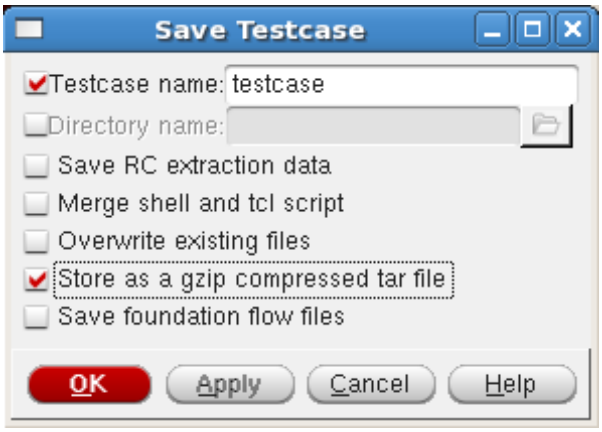
2. Click **OK**. This will generate a default mapping file called `streamOut.map` and then export GDS based on this mapping file. You can open `streamOut.map` to view the example mapping file.

3-15 Saving a Test Case

Innovus makes it easy to collect all the library and design data to make a test case. Use this feature to package test cases which you are providing to Cadence for debug.

1. Select **File – Save – Testcase**.

2. Enable **Testcase** name and specify *testcase*.
   Select Store as a gzip compressed tar file.



3. Click OK. This will copy the libraries and design data currently in memory to a directory called *testcase* then tar and gzip it into the file *testcase.tar.gz*. The design can then be easily restored by extracting the file and running `innovus -init testcase.tcl`.

This concludes this lab.

End of Lab