LAB TUTORIAL #4 – VLSI DESIGN FLOW: BACK-END

Original: F. Campi, Adapted By: A. Tino

In tutorial 1, 2, and 3, we learned about various aspects concerning front-end design and synthesis: the importance of synthesizable RTL, having a good stdcell library, and the impact of constraints on our design. Until this point however, we have not considered the importance of the chip's floorplan, i.e. cell placement, routing of interconnects between the cells, and the clock. Therefore, we must be aware that the conditions we implement in the Back-End (BE) will dramatically change from our Front-End results:

I. Wire delays will not be estimated by models, but rather extracted from real wire resistance/capacitance.
II. The clock tree will not be ideal as we have assumed, but rather described by a tree of buffers, with a given skew between each leaf (FF).
III. Subsequent to clock tree definition, hold violations will need to be fixed (further affecting timing)

The BE tools will allow us to:

a) Support the user in defining a floorplan, considering silicon area, with I/O pins and macros, and determine the location of where cells shall be placed
b) Remove all existing buffers from the synthesis netlist, and place remaining cells on the floorplan
c) Perform clock tree synthesis
d) Route connections between cells

We will likely incur exceptions at this step due to such Place&Route (P&R) activity. Namely setup, hold, max capacitance, and max transition violations, where the tools will insert buffers to fix these issues. Consequently:

- The timing results obtained during synthesis may suffer degradation, which is often very significant. During synthesis, the wire parasitics are only roughly estimated. For instance, in Verilog, there is no way to specify the length of a net, or the metal type used to build it. Hence wire parasitic information is only available during P&R.
- The tool will need space to place additional buffers, and therefore we must reserve floorplan area accordingly.
- Routing will require slight additional space to avoid congestion, depending on the number of metal layers available and used.

Overall, we can expect both a timing and estimated area degradation in comparison to our FE design. Our aim is to control this degradation in order to meet our design specification.

The input for our BE activity is as follows:

Design Information:

1. Verilog netlist generated/derived during synthesis (.v file)
2. Constraints file derived during synthesis (.sdc)
3. Specific set of constraints for Clock Tree Synthesis (.cts)

Technology Information

4. Timing libraries (liberty (.lib) files) for STA and Physical libraries (.lef files) for describing space requirements per stdcell
5. Capacitance and resistance information about wires (captables)

The tool we will be using for Back-End design is **Cadence Innovus**. We will start by setting up a specific environment, as we have in previous labs. It is advisable to **open a new terminal** specific for our BE activity.

Let us now prepare for the **BE environment**:

```
source /etc/profile.d/ensc-cmc.csh
```

```
source /CMC/setups/CDS_setup.csh
```

```
source /CMC/setups/BE_setup.sh
```

Going back to the rgb2grayscale folder from tutorial #1, and execute the following commands:

```
mkdir BE_045
```

```
cd BE_045
```

```
mkdir results
```

```
mkdir DBS
```

```
mkdir scripts
```

```
mkdir inputs
```

The back-end procedure is divided as follows:

1. Import Design
2. Floorplanning
3. Power Distribution
4. Placing
5. Post-Placing Optimization
6. Clock Tree Synthesis
7. Post-CTS Optimization
8. Routing
9. Finishing

We will use both Innovus' interactive GUI and shell scripts for developing our back-end design.

1. Import Design

Innovus has both a GUI and a shell for executing commands. Do not run Innovus in the background (do NOT append & to innovus&) to avoid the OS from freezing.

We will require the following files from our previous front-end design of the rgb2gray core during this tutorial: the Verilog netlist (**rgb2gray_ref.v), and the .sdc file in your results folder**. Copy these files to your "inputs" folder created in your BE_045 folder. Note: It's important that you analyze the content of the .sdc file when applying it to your design. Here are example commands to copy the two files:

```
cp syn_045/results/rgb2gray.ref.v     BE_045/inputs/
cp syn_045/results/rgb2gray.sdc      BE_045/inputs/
```

******ENSURE THAT YOUR NETLIST AND .SDC COPIED ABOVE ARE FROM THE CORRECT STDCELL LIB AND DESIGN POINT WHICH YOU ESTABLISHED/GENERATED DURING PHASE 1 FOR YOUR CORE******

Cd to your BE_045 folder, and launch the Innovus application by typing the following in the terminal:

```
innovus
```

Ignore the warnings as they pertain to running Innovus as a 32bit executable on our 64bit OS. Also ignore **ERROR: (IMPOAX-142/820/850) messages. Eventually a GUI and command shell window will open.

In the GUI's menu bar, click **File > Import Design.** A window will launch with resembles Fig. 1.

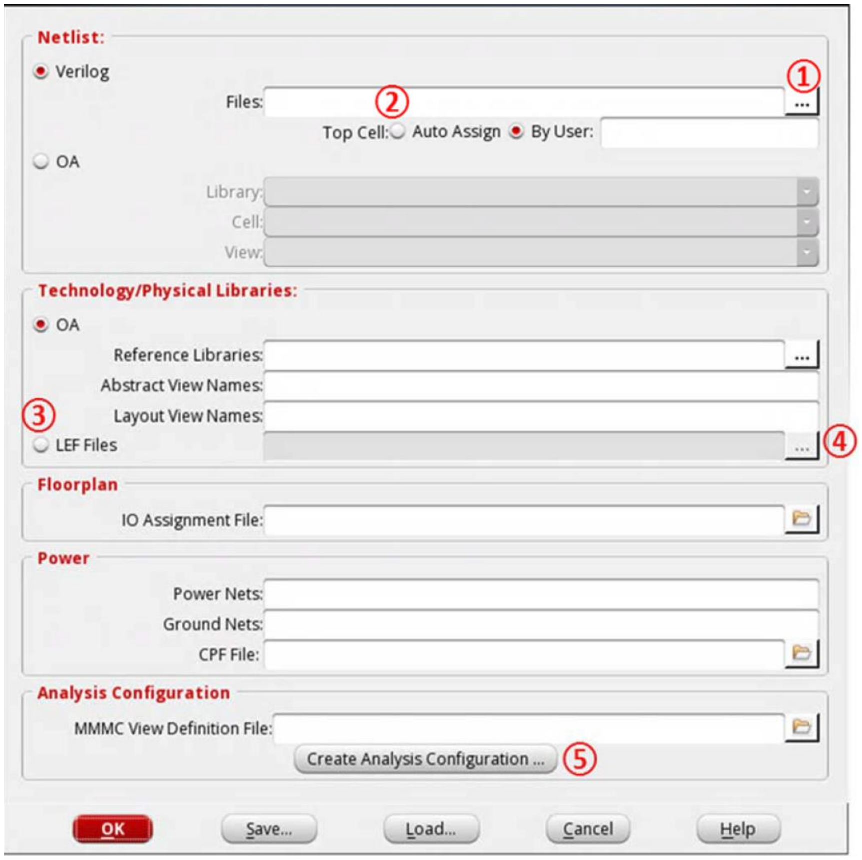
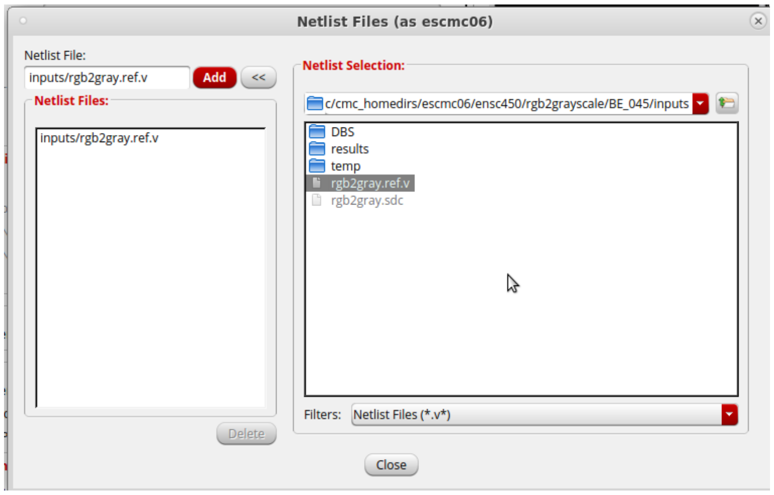


Fig. 1: Design Import Window

Notice the numbered labels on Fig. 1. Click the GUI button indicated by **1** on Fig. 1 **(…)** to open the "Netlist Files" window in Innovus. In the Netlist Files window, expand the window by clicking the ">>" button. A sub-window entitled **Netlist Selection** will appear as shown in Fig. 2.


Netlist Files (as escmc06)

Netlist File: inputs/rgb2gray.ref.v

Netlist Files:
inputs/rgb2gray.ref.v

Netlist Selection:
c/cmc_homedirs/escmc06/ensc450/rgb2grayscale/BE_045/inputs
DBS
results
temp
rgb2gray.ref.v
rgb2gray.sdc

Filters: Netlist Files (*.v*)

Fig. 2: Netlist Files Windows

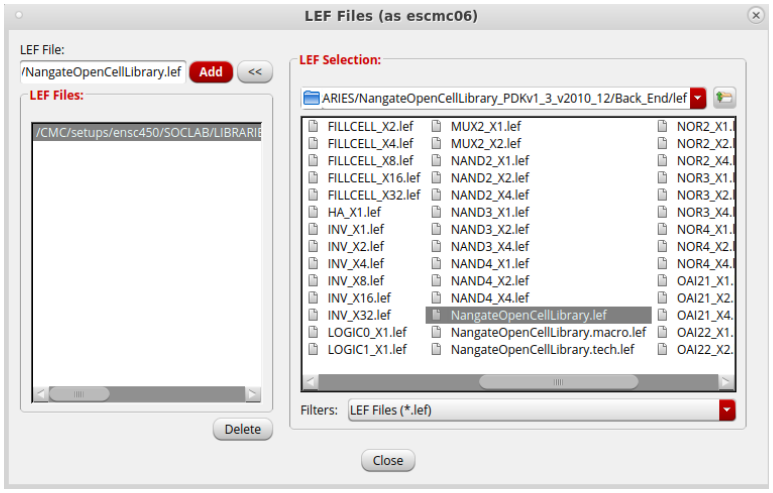
Navigate to the input/ folder you created in BE_045, and select your netlist file rgb2gray.ref.v. Highlight this file and double-click to transfer the file to the "Netlist Files" sub-window. Click **Close** once completed.

This will bring you back to the **Import Files** window of Fig. 1. Next, as indicated by 2 in the **Import Design** window, under **Top Cell**, select **By user** and type **rgb2gray** (i.e. your TOP entity).

Next in the current **Import Files** window, click the button as indicated by **3** in Fig. 1 to select the LEF file option, and click 4 to open the LEF Files window as shown in Fig. 3. Again, select ">>" to expand this window.

Under the **LEF Selection** sub-window, click on the red downward icon, select "/" and navigate to /CMC/setups/ensc450/SOCLAB/LIBRARIES/NangateOpenCellLibrary_PDKv1_3_v2010_12/Back_End/lef/

Select NangateOpenCellLibrary.lef. Click "<<" (or simply double click the file) to bring the .lef file to the LEF Files sub-window. Click **Close**.


LEF Files (as escmc06)

LEF File: /NangateOpenCellLibrary.lef

LEF Files:
/CMC/setups/ensc450/SOCLAB/LIBRARIE

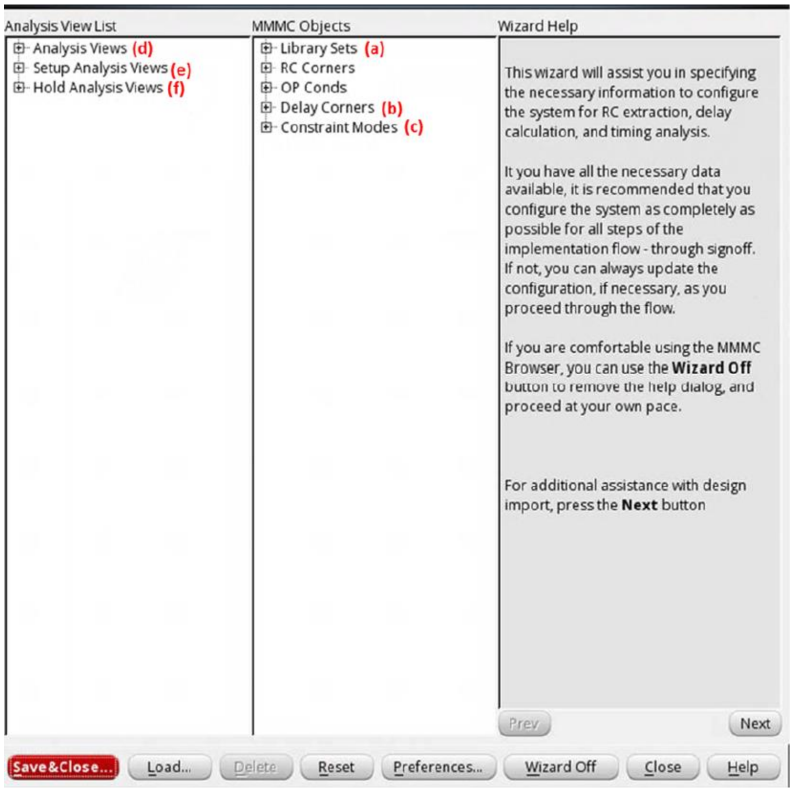
LEF Selection:
ARIES/NangateOpenCellLibrary_PDKv1_3_v2010_12/Back_End/lef

FILLCELL_X2.lef
FILLCELL_X4.lef
FILLCELL_X8.lef
FILLCELL_X16.lef
FILLCELL_X32.lef
HA_X1.lef
INV_X1.lef
INV_X2.lef
INV_X4.lef
INV_X8.lef
INV_X16.lef
INV_X32.lef
LOGIC0_X1.lef
LOGIC1_X1.lef
MUX2_X1.lef
MUX2_X2.lef
NAND2_X1.lef
NAND2_X2.lef
NAND2_X4.lef
NAND3_X1.lef
NAND3_X2.lef
NAND3_X4.lef
NAND4_X1.lef
NAND4_X2.lef
NAND4_X4.lef
NangateOpenCellLibrary.lef
NangateOpenCellLibrary.macro.lef
NangateOpenCellLibrary.tech.lef
NOR2_X1.
NOR2_X2.
NOR2_X4.
NOR3_X1.
NOR3_X2.
NOR3_X4.
NOR4_X1.
NOR4_X2.
NOR4_X4.
OAI21_X1.
OAI21_X2.
OAI21_X4.
OAI22_X1.
OAI22_X2.

Filters: LEF Files (*.lef)

Fig. 3: Lef File Window

Closing this window will bring you back to the **Import Design** window of Fig. 1. Finally, click on the icon indicated by **5** in Fig. 1 (**Create Analysis Configuration**…) which opens the MMMC Browser window displayed in Fig. 4.



This wizard will assist you in specifying the necessary information to configure the system for RC extraction, delay calculation, and timing analysis.

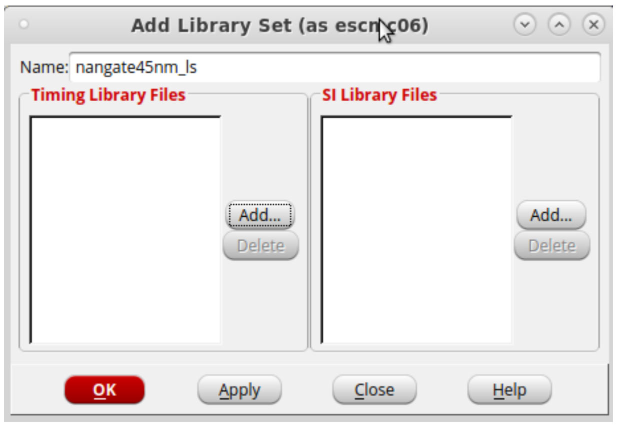
It you have all the necessary data available, it is recommended that you configure the system as completely as possible for all steps of the implementation flow - through signoff. If not, you can always update the configuration, if necessary, as you proceed through the flow.

If you are comfortable using the MMMC Browser, you can use the **Wizard Off** button to remove the help dialog, and proceed at your own pace.

For additional assistance with design import, press the **Next** button

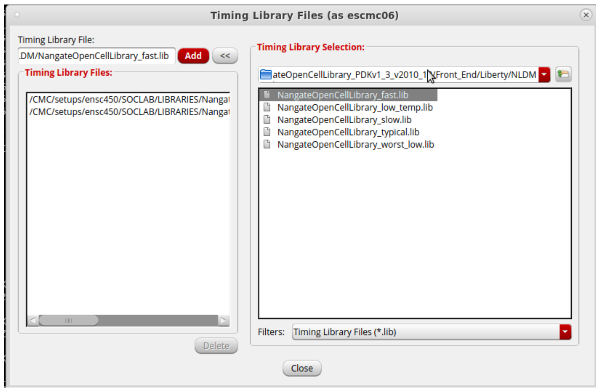
Fig. 4: MMMC Browser Window

Double-click the Library Sets option as indicated by (a) in Fig. 4 (do not click the'+' button). This will open the **Add Library Set** window of Fig. 5. In the Name field, type "nangate45nm_ls". Click the Add button in the **Timing Library Files** sub-window which will open another window called **Timing Library Files** as seen in Fig. 6.



Name: nangate45nm_ls

Fig. 5: Add Library Set Window


Timing Library Files (as escmc06)

Timing Library File: .DM/NangateOpenCellLibrary_fast.lib

Timing Library Files:
```
/CMC/setups/ensc450/SOCLAB/LIBRARIES/Nangat
/CMC/setups/ensc450/SOCLAB/LIBRARIES/Nangat
```

Timing Library Selection: ateOpenCellLibrary_PDKv1_3_v2010_1 \Front_End/Liberty/NLDM
```
NangateOpenCellLibrary_fast.lib
NangateOpenCellLibrary_low_temp.lib
NangateOpenCellLibrary_slow.lib
NangateOpenCellLibrary_typical.lib
NangateOpenCellLibrary_worst_low.lib
```

Filters: Timing Library Files (*.lib)

Fig. 6: Timing Library Files

In the **Timing Library Files** window, expand the window by clicking “>>”. Under Filters at the bottom right, select “All Files (*)”. Navigate to:

/CMC/setups/ensc450/SOCLAB/LIBRARIES/NangateOpenCellLibrary_PDKv1_3_v2010_12/Front_End/Liberty/NLDM

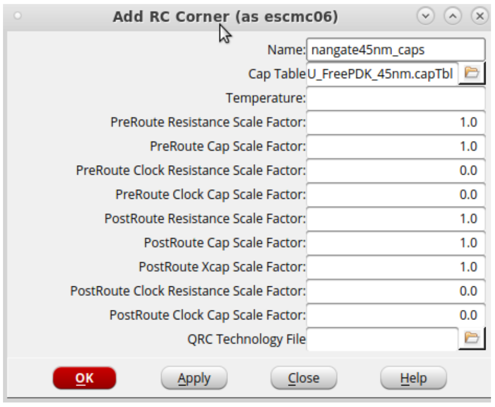
Select NangateOpenCellLibrary_slow.lib

Double click these files to bring them to the **Timing Library Files** sub-window, and click **Close**. This will bring you back to the window displayed in Fig. 5. Click **OK** to close the window. You will be brought back to the MMMC Browser window shown in Fig. 4.

Back in our MMMC Browser, we must specify the rc corners (captables) that Innovus should consider. Double click on **RC Corners**, which will open the **Add RC Corner** window as shown in Fig. 7. Under name, type “nangate45nm_caps”. Under the option **Cap Table**, navigate to:

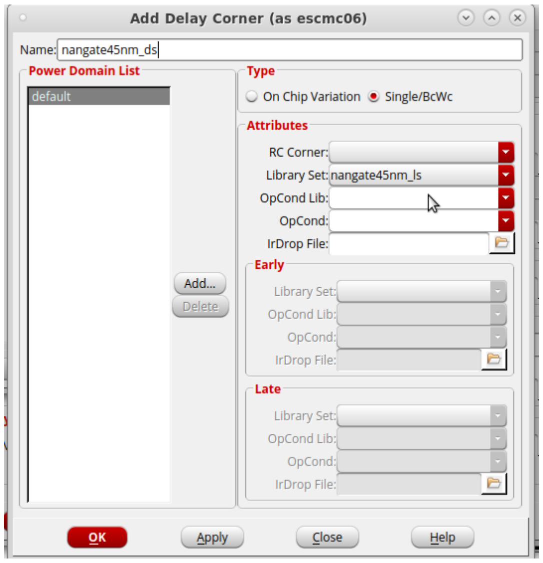
/CMC/setups/ensc450/SOCLAB/LIBRARIES/NangateOpenCellLibrary_PDKv1_3_v2010_12/Back_End/captables

And select **NSCU_FreePDK_45nm.capTbl**. Under temperature, specify 125C (your operating temperature). Leave the rest of the defaults as they are, and press OK to exit the window and go back to the MMMC browser window.


Add RC Corner (as escmc06)

Name:	nangate45nm_caps
Cap Table	U_FreePDK_45nm.capTbl
Temperature:	
PreRoute Resistance Scale Factor:	1.0
PreRoute Cap Scale Factor:	1.0
PreRoute Clock Resistance Scale Factor:	0.0
PreRoute Clock Cap Scale Factor:	0.0
PostRoute Resistance Scale Factor:	1.0
PostRoute Cap Scale Factor:	1.0
PostRoute Xcap Scale Factor:	1.0
PostRoute Clock Resistance Scale Factor:	0.0
PostRoute Clock Cap Scale Factor:	0.0
QRC Technology File	

Fig. 7: RC Corner window


Add Delay Corner (as escmc06)

Name: nangate45nm_ds

Power Domain List: default

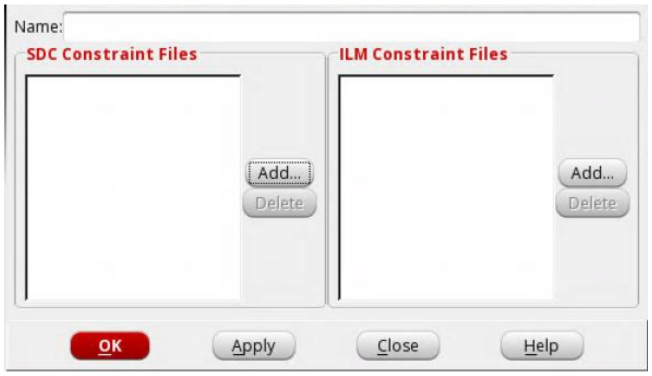
Type: On Chip Variation [✓ Single/BcWc]

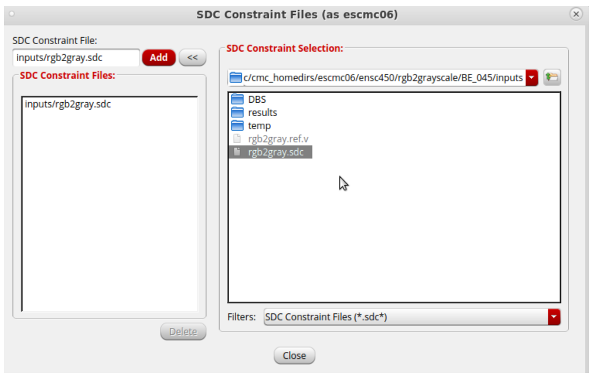
Attributes
RC Corner:
Library Set: nangate45nm_ls
OpCond Lib:
OpCond:
IrDrop File:

Fig. 8: Add Delay Corner Window

Next, click **(b)** in the MMMC Browser to select **Delay Corners**, which will open the **Add Delay Corner** window of Fig. 8. Type "nangate45nm_dc" in the **Name** field. Under the Library Set field, select **nangate45nm_ls**, and under RC Corner select **nangate45nm_caps**. Close this window by clicking **OK**, which will bring you back to the MMMC browser.

In the MMMC Browser, click **(c) Constraint Modes** as indicated on Fig. 6, which opens the **Add Constraint Mode** window of Fig. 9. In the **Name** field, type rgb_cm. Next, click the **Add** button in the **SDC Constraint Files** sub-window, which will open a new window as shown in Fig. 10.




SDC Constraint Files (as escmc06)

SDC Constraint File: inputs/rgb2gray.sdc

SDC Constraint Files: inputs/rgb2gray.sdc

SDC Constraint Selection: c/cmc_homedirs/escmc06/ensc450/rgb2grayscale/BE_045/inputs

DBS
results
temp
rgb2gray.ref.v
rgb2gray.sdc

Filters: SDC Constraint Files (*.sdc*)

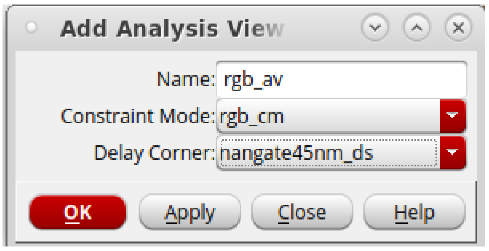
Fig. 9: Add Constraint Mode window

Fig. 10: SDC Constraint Files

Expand the **SDC Constraints Files** window by clicking ">>" and navigate to your input/ folder. Double-click your **rgb2gray.sdc** file to display the file in the **SDC Constraints Files** sub-window. This .sdc file contains the core's constraints for period and duty cycle, and includes other timing properties, similar to the synth.tcl script for the front-end design.

Click **Close** to close the window. This will take you back to the **Add Constraint Mode** window illustrated in Fig. 9. Click OK to close the window, which will take you back to the MMMC Browser window.

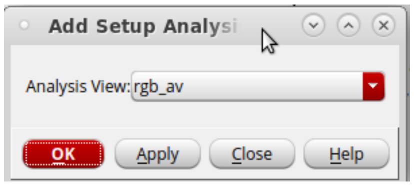
Back in the MMMC Browser, click the option indicated by **(d)** as indicated in Fig. 4, entitled **Analysis Views** which opens the **Add Analysis View** window as illustrated in Fig. 10. Under the **Name** field, type rgb_av, and ensure nangate45nm_dc is selected in **Delay Corner**, and rgb_cm in **Constraints Mode**. Click OK to close the window, bringing you once again to the MMMC Browser.



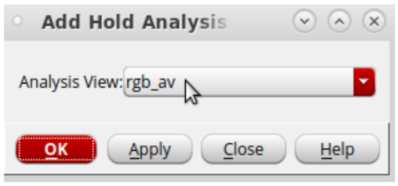
Name: rgb_av
Constraint Mode: rgb_cm
Delay Corner: nangate45nm_ds

Fig. 10: Add Analysis View window

Back in the MMMC Browser, click the option indicated by **(e)** as indicated in Fig. 4, entitled **Setup Analysis Views** which opens the **Add Setup Analysis View** window of Fig. 11(a). Confirm that rgb_av is selected in the **Analysis View** field. Click OK to close the window.



Analysis View: rgb_av



Analysis View: rgb_av

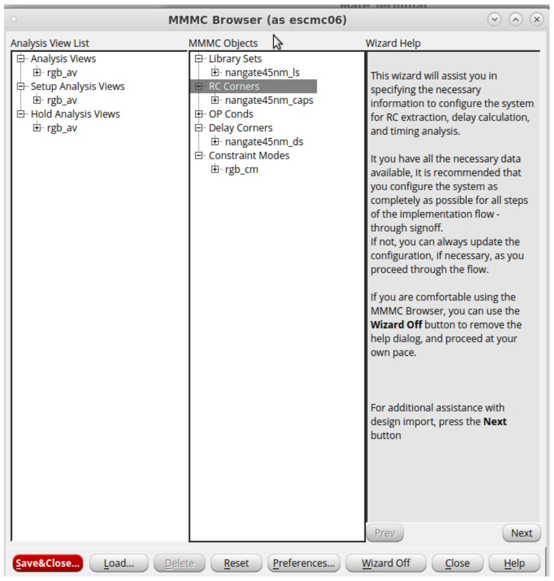
Fig. 11(a): Add Setup Analysis View window   Fig. 11(b): Add Hold Analysis View window

In the MMMC Browser, select the option indicate by **(f)** on Fig. 4, to open the **Hold Analysis Views** window of Fig. 11(b). Ensure that **rgb_av** is also selected in the **Analysis View** drop down menu. Click **Ok** to close the window and return to MMMC.

Finally, double click the Op Conds option in the MMMC Browser. In this window, we may specify our operating conditions PVT. For process, input **1** (implying our first process testing condition), under V specify **1.05**, and for T, specify **125**. Under Library file, navigate to the folder:

/CMC/setups/ensc450/SOCLAB/LIBRARIES/NangateOpenCellLibrary_PDKv1_3_v2010_12/Front_End/Liberty/NLDM/NangateOpenCellLibrary_slow.lib

Double click the file and press OK. The resulting MMMC Browser window should look like the figure provided in Fig. 13. Click **Save & Close,** and save this configuration as **Default.view**.



```
Analysis Views
  rgb_av
Setup Analysis Views
  rgb_av
Hold Analysis Views
  rgb_av
```

```
Library Sets
  nangate45nm_ls
RC Corners
  nangate45nm_caps
OP Conds
Delay Corners
  nangate45nm_ds
Constraint Modes
  rgb_cm
```

This wizard will assist you in specifying the necessary information to configure the system for RC extraction, delay calculation, and timing analysis.

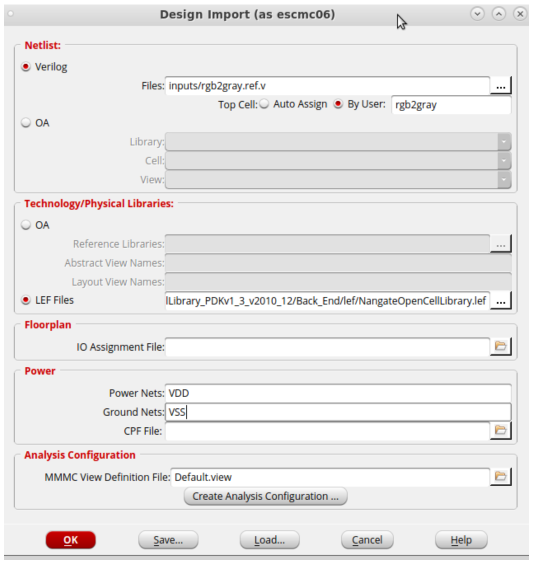
It you have all the necessary data available, it is recommended that you configure the system as completely as possible for all steps of the implementation flow - through signoff.
If not, you can always update the configuration, if necessary, as you proceed through the flow.

If you are comfortable using the MMMC Browser, you can use the **Wizard Off** button to remove the help dialog, and proceed at your own pace.

For additional assistance with design import, press the **Next** button

Fig. 13: Final MMMC Browser – final configuration window

We should now be back in the **Design Import** window. Under the **Power** tab, specify the **Power Nets** as VDD, and **Ground Nets** as VSS. Your **Design Import** window should now look like Fig. 14.



Netlist:
[✓ Verilog]
Files: inputs/rgb2gray.ref.v
Top Cell: Auto Assign [✓ By User]: rgb2gray
OA
Library:
Cell:
View:

Technology/Physical Libraries:
OA
Reference Libraries:
Abstract View Names:
Layout View Names:
[✓ LEF Files] lLibrary_PDKv1_3_v2010_12/Back_End/lef/NangateOpenCellLibrary.lef

Floorplan
IO Assignment File:

Power
Power Nets: VDD
Ground Nets: VSS
CPF File:

Analysis Configuration
MMMC View Definition File: Default.view
Create Analysis Configuration ...

Fig. 14: Design Import – Final Window

Note: if you click **Save** at this point to save the design import parameters as a .global file, you may reload this view later to continue your lab, and perform any necessary changes for your design.

Click **OK** in the **Design Import** window to import your design into Innovus. Next, switch to the command shell window, where Innovus will output information as it imports your design. Check to see the report. You will likely see some warnings which Innovus will automatically correct. You will observe the odd error pertaining to OA which we selected not to use during P&R – we are using lef files instead. Ignore these errors.

Innovus' main window will now display rows, indicating where the stdcells of your design will be placed during the placement phase. You will notice a gold and red polygon shapes at the bottom of the grid – these are your IO pins, which will be connected during P&R. Feel free to zoom up and observe the labeled pins.

2. Floorplanning

The previous import design step built a basic square floorplan with the density information provided. We will override this information using the GUI's floorplan option. Back in the Innovus GUI window, in the menu bar select **Floorplan > Specify Floorplan …**

In the **Basic** tab, we have an option to specify the core size by either *1) Aspect Ratio (AR)* or *2) Dimension*. "Core" signifies the region which the cells can be placed. For the purpose of this tutorial, we will specify the aspect ratio of our core region. It is always possible to go back to this window later (if necessary) to change the values or options and Innovus will update the floorplan as desired.

Under the **Aspect Ratio** option, select 1. In the **core utilization** field, enter 0.7 (% of final chip that will contain useful standard cells vs filler/empty cells. The higher the %, the more time Innovus will spend optimizing your design)

Next, under **Core to IO Boundary** selection, set **Core to Left, Core to Top, Core To Right**, and **Core to Bottom** as 4. Specify the floorplan origin as **center**. Click **OK**, where Innovus will return to the main window and display the modified floorplan outline.

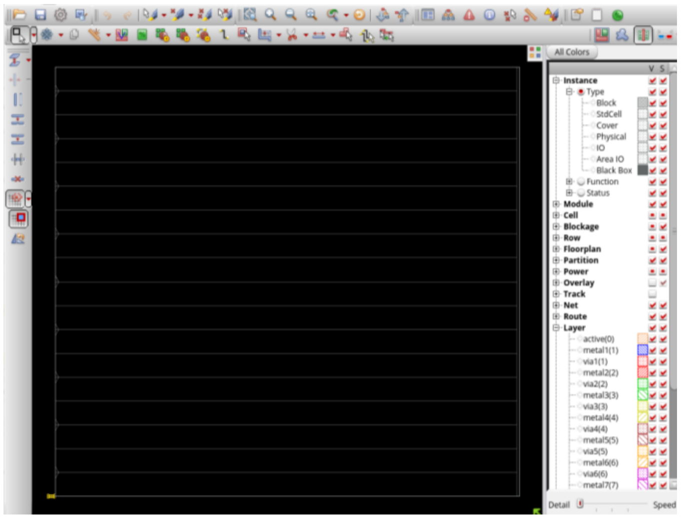


Fig. 15: Floorplan window

Note: It is possible to save what we have performed up to this step by navigating back to the Innovus command shell and typing: saveDesign rgb_01_floorplan.enc

If you would like to load your design later, upon launching Innovus, add the following option:

innovus -init rgb_01_floorplan.enc

On the contrary, if you're already in innovus (after launching the application), you may type the following command to load the floorplan at this point: source rgb_01_floorplan.enc

It is highly suggested you save your design using **File > Save Design,** select **Innovus** type.

For each run, Innovus will produce useful log files with the extensions .cmd# and .log#, where # is the number of times you have run Innovus within the same folder. Old logs can be useful as a reference, but don't accumulate too many within the folder as this can get very confusing. cmd files contain a list of all the commands launched during the session (history), while .log will keep track of all commands and their respective outputs. **HINT: You may use this file to create a script which will setup your environment and floorplan without the need to use the GUI. This will save you time.**

Back to scripting

As we saw in the first tutorial, scripting may be more of a convenient form of documentation and design execution in comparison to the GUI method which we have used thus far. If you would like to see the equivalent GUI commands which were executed by Innovus in the background, you may search the .log and .cmd files created in the work/ directory. These commands may be utilized as part of a script (if you wish), or may be launched one by one in Innovus' console window.

It is highly recommended that you check log files, especially when launching a rather long script. It may be difficult to follow all the steps individually, and hence log files may aid in your design's verification. You may use the Unix command "grep" in a free terminal (not the Innovus terminal) to navigate logs and search for the keywords "error" or "warnings" to spot any issues in your design Ex: grep -i "error" xxxxx.log Errors usually mean that something was wrong, or that the provided constraints (period, area) were too tight, and Innovus can not find a suitable solution. Warnings are often acceptable, but they should still be checked and understood. Again, errors IMPOAX-142/820/850 may be ignored as they pertain to the OS.

We will now switch over to scripting to complete the remainder of our back-end design for the rgb2gray tutorial. We will perform place and route by executing the appropriate tcl script located in the rgb2grayscale CMC folder under BE_045/scripts. At each step, we will request a timing analysis from the tool using a given script, in addition to a short analysis of the netlist.

Although we are switching to this automated method, it is a good idea to try and launch the commands one by one in Innovus, which will allow you to have a more in-depth understanding of the procedure and a better handle of the Innovus commands. You may use the man <command> utility in Innvous to obtain help with a particular command and its options.

3. Power Distribution

We will require a power **ring** for our design. A ring is used to connect the stdcells and macros of our design to Vdd and Vss. Since the ring may be quite distant for certain cells based on their placement, we also use **stripes** which extend the ring segments across the floorplan**.** Stripes are vertical/horizontal continuations of the power ring for improved current flow. It is important that when we place the ring, we consider our floorplan to ensure that we reserve enough space between the core and pins to build the **ring**.

In this rgb2gray example, we have no IO pads to place since we are building an embedded block. This particular block does not contain any analog macros such as PLL/DLL, or memory. The only significant object in our floorplan is therefore the IO pins. In addition, since we have a very small design, it is not necessary to add a full grid (horizontal and vertical) of stripes. To avoid routing congestion, we will only utilize vertical stripes. The technology we are using also does not have any well taps or end caps, so we will need to route Metal1 (M1) stripes to the ring.

If we observe our LEF file for the 45nm technology, we will see that M9 and M10 are the highest possible metals. Accordingly, we will form our ring with metals in the medium range and use their widths to nicely fit these metals into our floorplan's power ring. Also note that 1) since this is a smaller core, it will likely be integrated within an SoC. The highest metals will likely be used by the SoC. 2) We want to avoid unnecessary routing congestion, so we will select a metal from the mid range of available metals for now. If routing congesting becomes a problem, we can always revisit this step and adjust either the density (utilization) and/or the metals used for power planning.

With these factors in mind, let us add a power ring and stripes to our design. Copy the powerPlan-01.tcl file from the course folder. Ensure you are in your **BE_045** folder and execute the following commands:

```
cp /CMC/setups/ensc450/rgb2grayscale/BE_045/scripts/03-powerPlan.tcl   scripts/
```

Open the file to view the contents and verify the parameters which were discussed above. The script will modify the floorplan such that the pins will have a fixed positioning and add a power ring around the design. Note that as an alternative, a ring may also be added to the project using the GUI with *Power > Power Planning > Add Rings*. All commands may alternatively be setup in the GUI, however we will stick with scripting as it is much easier and efficient.

This script will also create stripes and make connections from the stripes to rings by using the Special Route feature in Innovus (sroute). We will also save our floorplan at this step and generate a summary report. To run a script in the Innovus command shell, use the **source** command:

```
source scripts/03-powerPlan.tcl
```

Verify that your floorplan resembles Fig. 17. Viewing issues? Click **Window > Design Browser + Physical**

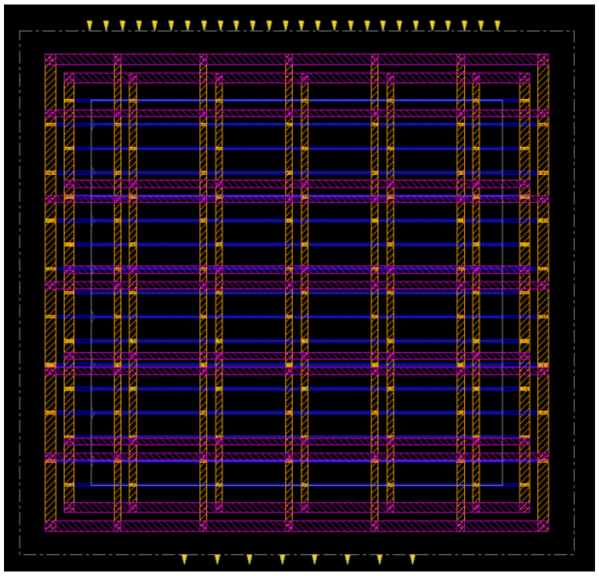


Fig. 17: Floorplan with Rings and Stripes

Verify that the design reports were properly generated as specified in the tcl script, and observe their contents. The floorplan will be saved in results/floorplan.enc should you want to restore the design at this point later, as well.

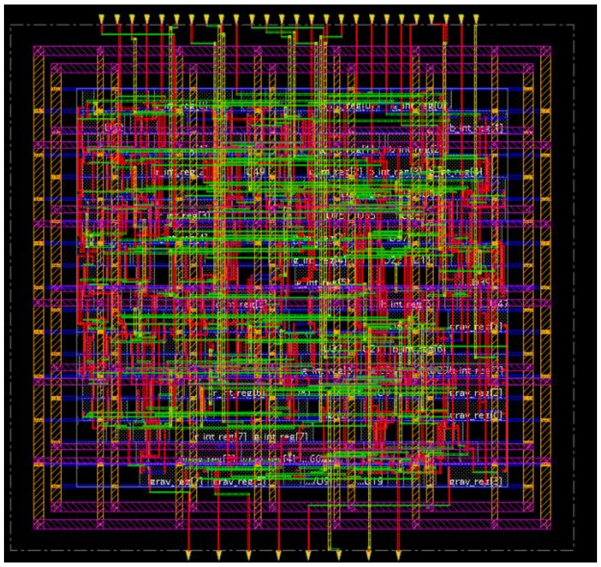


Fig. 18: Stdcell Placement and trial-route

4. Placement

Now that we have laid out our power nets and rings on the floorplan, we may place our cells. Copy the placement script and observe its contents:

```
cp /CMC/setups/ensc450/rgb2grayscale/BE_045/scripts/04-placement.tcl   scripts/
```

When running this script, Innovus will perform placement as well as trial-routing (not actual routing), generating a floorplan placement which resembles Fig. 19. Trial route in this case is used to obtain a sensible, pre-estimated wire delay for our placement, and to provide timing checks prior to extracting RCs. This step will route cells to other cells and power rails as required of the design. Thereafter, the 04-placement.tcl script will extract RC parasitic from the pre-route phase to get a more accurate, yet optimistic, timing report for the layout.

To obtain this placement and early route estimates, run the placement script:

```
source scripts/04-placement.tcl
```

Observe the reports generated by Innovus and ensure that your placement resembles Fig. 18.

5. Post Placement Optimization

To optimize the design prior to our next step, i.e. clock tree synthesis, we will run post placement optimization. Copy the post placement script to your scripts folder, and run:

```
cp /CMC/setups/ensc450/rgb2grayscale/BE_045/scripts/05-postPlaceOpt.tcl   scripts/
```

```
source scripts/05-postPlaceOpt.tcl
```

Your design should look like Fig. 19, which is extremely similar to Fig. 18. This step simply ran a timing analysis after cell placement, to fix (insert buffers) for any hold violations, break capacitance for any max cap violations, and decrease delay for any max path violations. Therefore you may see more buffer cells and paths added to your design at this point.

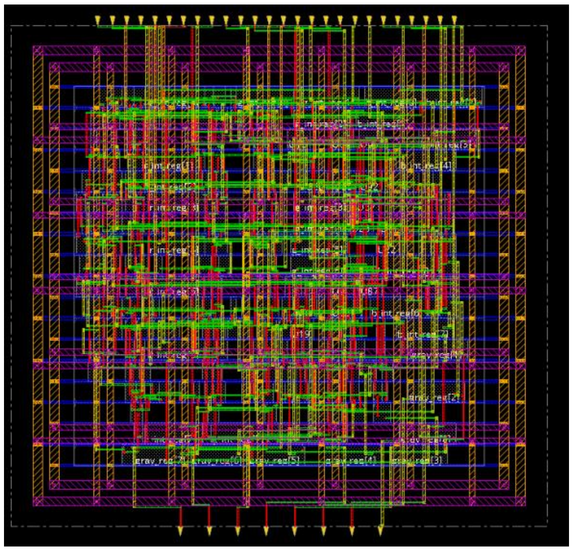


Fig. 19: Post Placement Optimized Layout

6. Clock Tree Synthesis (CTS)

Next, we must synthesize a clock tree for our design. In many libraries, there are specific buffers used for the Clock Tree. One could use any buffer (buf) or Inv to build a clock tree, but specific buffers are designed in this case to (i) have a better output transition on the rising edge, and (ii) be more resistant than the average buffer to on-chip variation i.e. limit variation on the clock skew that would impact all paths for a given design.

You are provided with a clock tree synthesis file in the rgb2gray folder. Copy and observe the file's contents. Within your BE_045 folder:

```
cp /CMC/setups/ensc450/rgb2grayscale/BE_045/inputs/rgb2gray.cts   inputs/
```

```
cp /CMC/setups/ensc450/rgb2grayscale/BE_045/scripts/06-cts.tcl   scripts/
```

```
source scripts/06-cts.tcl
```

A clock tree will be generated for the layout, in addition to several reports. A setup and hold detailed report may be found in the results/summary/ folder. Further, search for report details for rise and fall, min/max skew – these values represent your clock skew which must be accounted for in latency constraints and may be found in the .ctsrpt generated. Note that you will need to generate a .cts for your own design.

You may view the clock tree in the GUI by selecting: **Clock > Display > Display Clock Tree. Route Selection:** Post-Route, **Levels:** All levels. Click **OK**. You may also browse the clock tree – **Clock > Browse Clock Tree**, select clk as the Clock, with **Route Selection**: Post-Route (or as you wish), and click OK. A browser will demonstrate the clock distribution on the rgb2gray core. Feel free to play around with the tool.

7. Post CTS Optimization

Next, we will fix any setup violations which may have generated after clock tree synthesis by running a post cts timing analysis. This will help us ensure that no violations have occurred before proceeding to our final routing stage. The script we use will also generate a post cts netlist, and a final timing report. Copy the script and execute:

```
cp /CMC/setups/ensc450/rgb2grayscale/BE_045/scripts/07-postCTSOpt.tcl   scripts/
```

```
source scripts/07-postCTSOpt.tcl
```

View the characteristics of your design by reading the reports generated in results/summary/07-postCTSOpt.rpt. A summary of area, utilization of metals, core size etc will be provided, in addition to timing analysis in results/timing/07-postCtsOpt-timeDesign.setup[hold]/xxxxxxx_tarpt.gz

8. Routing

Now that we have fully placed and verified our cells and clock tree, we may proceed to our final routing stage. Copy the route script and run:

```
cp /CMC/setups/ensc450/rgb2grayscale/BE_045/scripts/08-route.tcl   scripts/
```

```
source scripts/08-route.tcl
```

View the generate reports by switching to the other terminal window. The report will provide an analysis on your routing information, including total wirelength, wirelength for each metal layer, number of vias, the number of DRC violations, and other statistics.

Note: To capture nice screenshots of your floorplan, you may use Innovus' feature: **Tools > Screen Capture > Write to GIF file…**

9. Finishing

Our final design is now in place. We may save the final netlist which includes the generated clock tree and all optimizations, the final place and routed circuit design (.enc), and respective design exchange format (def) file that describes the P&R circuit. Copy the script and observe its contents:

```
cp /CMC/setups/ensc450/rgb2grayscale/BE_045/scripts/09-finishing.tcl   scripts/
```

```
source scripts/09-finishing.tcl
```

Although we have generated a full P&R of the chip, the next and final phase prior to fabrication would be to import this design into Cadence Virtuous, and generate a final layout of the chip and its stdcells. Therefore, to complete this flow, a gds file would be required for import into Cadence Virtuoso. To do this, save your design in Innovus using the GDSII file format - **File > Save > GDS**

You will also obtain a final.v file in your results/verilog folder (see 09-finishing.tcl script for the command and location). You may use this against your testbench to verify your final P&R core.

Notes About P&R

Do not change the setting of the scripts unless you are at a very advanced stage of VLSI design. If you are not happy with the results you obtain in your design, you may change the design by means of two options:

1. Change the floorplan by changing the placing density
2. Change the constraints – increase the period (sdc), change the IO delay, or change the clock tree specifications (skew and latency)

It is an interesting P&R activity to replicate this tutorial for different values of placement density, commencing from 2% up to 90%. It is expected that larger density causes higher delays initially (longer wires have greater capacitance), however at a certain point the routing congestion would need to be handled by creating longer wires even considering a smaller space. Longer wires yield an increase in max cap and max trans violations. Therefore, the tool will not have enough space to place the clock tree, to fix buffers for hold/setup violations, and would not be able to conclude the P&R activity.

Checking for errors and warnings

Violations may be analyzed in the Innovus GUI with **Tools > Violation Browser**.

Aside from the reports, you may also visually inspect the layout of the chip:

1. Are power stripes in place?
2. Are cells placed over the row area?

You may visualize how stdcells and macros are mapped across the chip using **Windows > Design Browser + Physical**. You will see your nets, stdcells, and modules listed. Browse the hierarchy as you wish. For instance, if you expand the module list, only the adder macro is listed. Right click this module, click **Highlight**, and select a colour (yellow for instance). The module will be highlighted on the layout. You may right-click it again to "De-Highlight" as they call it.

3. Is routing present?

You can use the following steps in Cadence Innovus to display where the critical path is on the actual chip: **Timing > Debug Timing** from the menu. Generate the report (this may take a minute). Right click on first path in the *Path List*. Choose ***Highlight > Only This Path > Color***

4. Are there any obvious violations highlighted by the tool?

In Innovus, go to **Window > Workspaces > Violation Browser + Physical** (should be none listed)

Analyzing Results

Timing: For a quick assessment of the timing behaviour of the design at various stages, check the file results/timing/<0#-step>.<hold/setup>/<design>.summary.gz

These files will briefly summarize the paths of the design and the relative violations. The setup analysis also contains Max Transition and Max Cap violations. More details of these parameters are available in the same folder, under results/timing/0#-<step>.hold|setup.rpt, containing an analysis of the critical path, with even more details provided in results/timing/0#-xxxxx-timeDesign.<hold/setup>/xxxx_all.tarpt.gz folders.

We can check how the number of cells in the block have evolved in our design with the Unix command **grep** to quickly browse our reports (Check the grep syntax online). For instance, in a Unix shell (NOT the Innovus console) we can run:

grep “Instances:” results/summary/*

grep “Standard cells(Subtracting Physical Cells)” results/summary/*.rpt

grep “Chip Density #2” results/summary/*.rpt

For our case, we obtain something similar to the following results:

Stage	Instances (rgb2gray)	Cell Area (um^2)	Density	Violations
03-floorplan	325			
04-Placing	297	464.968	0.39670	
05-PostPlace OPT	290	424.536	0.39670	
06-CTS	290	424.536	0.39670	
07-Post-CTS OPT	290	424.536	0.39670	
08-Routing/09-Signoff	712	424.536	1	

As seen above in some instances, Innovus fails to properly generate a reportSummary containing the above information. If you can not find the above criteria for certain P&R step, within the GUI select **File > Report > Summary** which will generate a comprehensive report for that stage of your design.

If you wish to obtain general values, within Innovus command line terminal, type “report_area” and “checkFPlan -reportUtil” to determine these values. You may also use “report_power” to get a detailed breakdown of power consumption, similar to dc_shell.

We can see that there is no cell increase in the P&R stages, contrary to our expectations (except for the clock tree insertion during (CTS)). The lack of cell increase is due to 1) that our design is very, very small (rgb2gray), and 2) the 45nm technology is so dense, that when implementing such a small design, the parasitics of wire loads are minor and almost negligible. This design therefore did not require any specific optimization and was able to run at 2ns without increasing cell area, while maintaining a very compact placing.

At the end of P&R, depending on the level of density chosen for the floorplan, the tool may cause a few DRC (Design Rule Check) violations, such as shorts, antenna errors, spacing errors etc. Depending on the time we wish to invest vs desired/required performance of our design, we may relax design specs (especially density) and run another P&R which eliminates the violations OR manually fix them in a layout (i.e. GDS imported into Cadence Virtuoso, which we will not cover in the scope of this course).

For the purpose of this exercise, it is not necessary to resolve DRCs via layout, especially if only a few exist. It is acceptable to understand the reason that the violations are caused, and to change design specs in order to resolve your issues.

In the reports generated, you will see the following information (random example):

Chip Area: 1190.19 um^2, Core Area: 689.472 um^2

Core Density: 100% ….

Please note: ***Chip density*** refers to the density including IO rings, whereas ***core density*** does not consider IO rings.