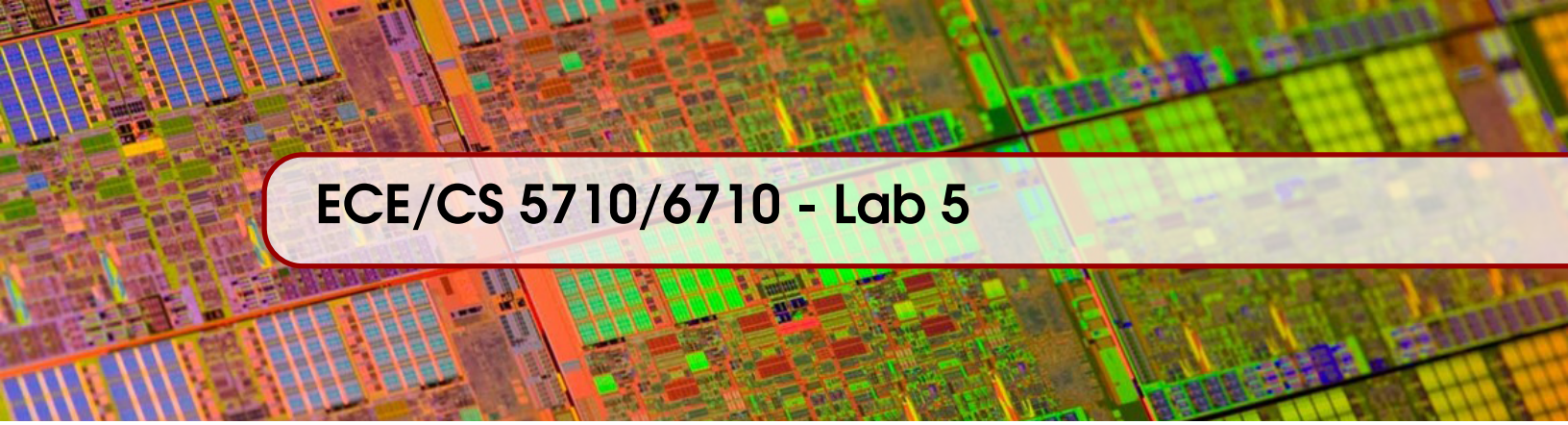


ECE/CS 5710/6710 - Lab 5

Floorplanning and Place & Route of a MIPS Processor

Pre-lab Assignment: Check for the date on Canvas.
Lab Report: Check for the date on Canvas.

1 Objective

This lab presents the main steps for performing the Place & Route (P&R) step of the 8-bit MIPS microprocessor you synthesized in the previous lab, using the TSMC 180 *nm* design kit. This lab details the typical steps of the back-end flow in VLSI design: floorplan, placement, clock tree synthesis and routing. The tool you will use is Cadence Innovus®, which is widely used in industry and academia. Innovus being a very complete and complicated tool, you will only use its basic functions in this lab, but remember that many other options and optimizations are available for each step.

R Note that this lab has been written with the verilog gate-level netlist obtained after the synthesis step with a very relaxed timing constraint. As such, some values such as the timing reports may differ from what you will get. Also, keep in mind that while your synthesized design was meeting your timing constraint, it might not meet it anymore after the P&R flow. In that case, you need to re-synthesized it with a more relaxed timing constraint and redo the P&R flow.

2 Pre-lab Assignment

Answer the following question and submit a **.pdf** file through Canvas:

1. What is the difference between a semi-custom and a full-custom design?
2. Describe briefly the main steps of the back-end flow.
3. What are clock jitter and clock skew and how can they be minimized?
4. What does the core density of your chip depend on?
5. What is IR drop? How can it be mitigated?

3 Directory Organization

For the Place & Route flow, you will have to work in your *innovus* folder. This folder contains different subdirectories as follows:

- *DBS*: design database, where you will save your design after the different step of the P&R.
- *RPT*: report files (area, number of gates, hierarchy etc.) created by the tool.
- *CONF*: Design import configuration files.

- *SCRIPTS*: Contains some script(s) used in this lab.
- *GDS*: Used to export your final design as a GDSII file.
- *SDF*: Timing files created by the tool for functional verification.

4 Lab Assignment

4.1 Starting Cadence Innovus®

Launch the Cadence Innovus® GUI by going in your *innovus* folder and doing:

```
innovus
```

You will obtain a window as shown in Fig 1. The console terminal is used for entering Innovus® commands, to source Tcl scripts or to run Linux commands. Pay attention to it since it contains crucial information when something went wrong in your flow (everything displayed in the current terminal is also stored in the *innovus.log* file in the *innovus* directory. **In case of errors, it is very helpful to open this *.log* in a text editor in order to check what went wrong**). When you will perform actions on the GUI, the equivalent commands will be recorded and stored in the *innovus.cmd* file in the *innovus* directory. The equivalent commands will also be given in this tutorial.

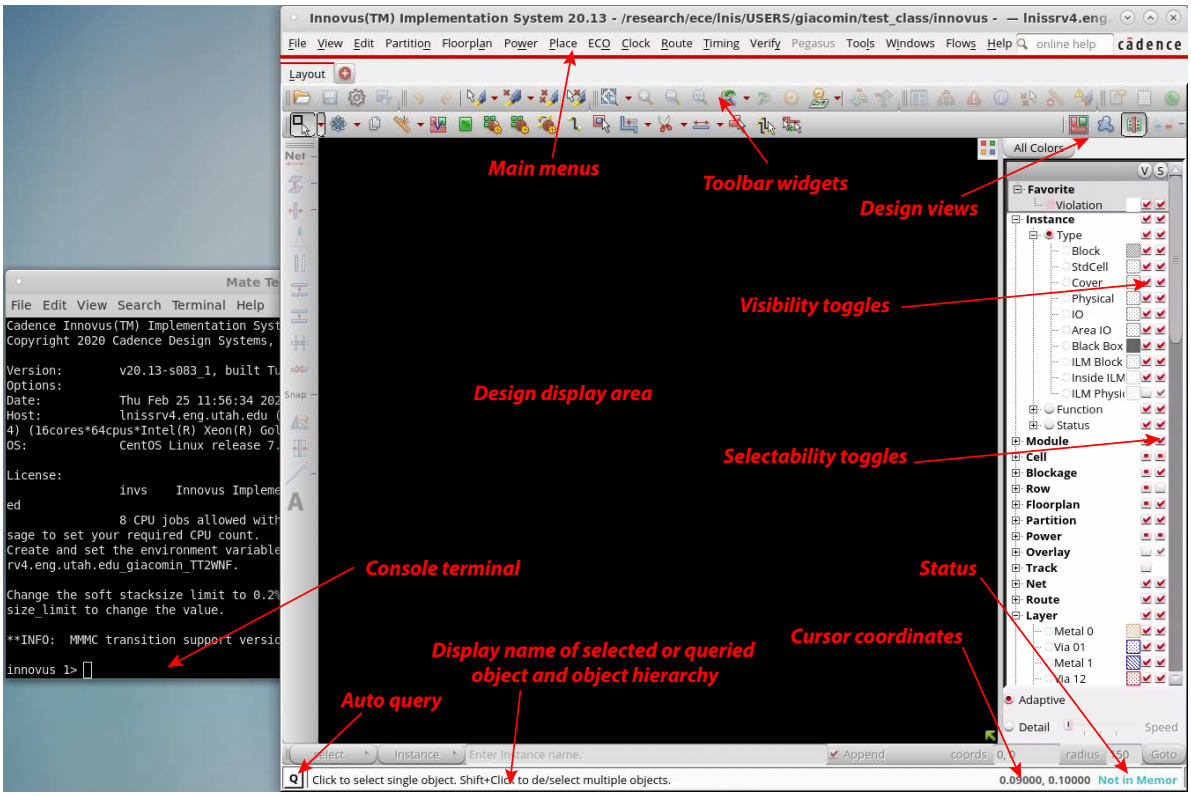


Figure 1: Cadence Innovus window organization.

The main window includes three different design views: the Floorplan view, the Amoeba view, and the Physical view. You can switch the view whenever you want by clicking on the icon view, as shown in Fig. 2.

Figure 2: Switch the design view

- *The Floorplan view* displays the hierarchical module and block guides, floorplan objects, the connection lines and power/ground nets.
- *The Amoeba view* displays the outline of the modules and submodules after placement, showing the physical locality of the module.
- *The Physical view* displays the detailed placements of the module blocks, standard cells, nets, and interconnects.

As in Virtuoso®, there are many useful shortcuts you can use. The most important ones are presented in Table 1.

Table 1: Main Innovus® shortcuts.

Shortcut	Action
b	display the list of binding keys
d	(de)select or delete objects
f	zoom the display to fit the core area
g	ascend in hierarchy
shift+g	descend in hierarchy
k	create a ruler
maj+k	remove last ruler displayed
q display	the object attribute editor form for the selected object. click the left-button mouse to select an object shift-click to select or deselect an object
u	undo last command
shift+u	redo last command
z	2× zoom-in
shift+z	2×zoom-out
shift+ctrl+r	refresh the display

4.2 Defining the Design Import Configuration

Before starting the P&R steps, you must import all the different files:

- *Technology information (.lef files)*: those files include the technology process design rules for placement and routing (name, spacing, width of routing layers, vias definition, core and pad site definitions, placement orientation *etc.*) and the physical informations about the standard cells (width and height, used metals etc.). Those files are supplied by the library cell provider.
- *Timing libraries*: those files contain the timing information for all the standard cells (path delays, setup and hold times *etc.*). Those are the same files you used in Synopsys DC® but here in the Liberty text (*.lib*) format. Those files are supplied by the library cell provider.
- *Capacitance tables*: Those files contains the necessary information for accurate capacitance extraction for the routing preparation and optimization steps. The tables are generally supplied as text files without standard file extension (e.g., .CapTbl) by the foundry, or can be generated with the process design kit.

- *Gate-level netlist*: this is the design netlist (*.v*) which will be placed and routed. This file has been generated during the synthesis (in the previous lab).
- *Timing constraints*: this file (*.sdc*) relates to the constraints you used during the synthesis and has been generated by Synopsys DC®.

The first task is to create a design configuration file which will define all the previous file in order to properly load the design for the back-end steps. Once such a configuration file exists, it can be simply loaded to import the design directly. To do so:

1. *File -> Import Design...*
2. Click on the *Load...* button, as shown in Fig 3 and load the *CONF/mips.globals* file. After this step, some settings should be applied to the design import window. The loaded file includes some configuration file for the TSMC 180 *nm* process. In particular, it contains the information and paths to the *.lib* and *.lef* files from the TSMC library required by the P&R tool. It also contains the path to your synopsys *.sdc* file so you will not need to add it yourself in the configuration step.

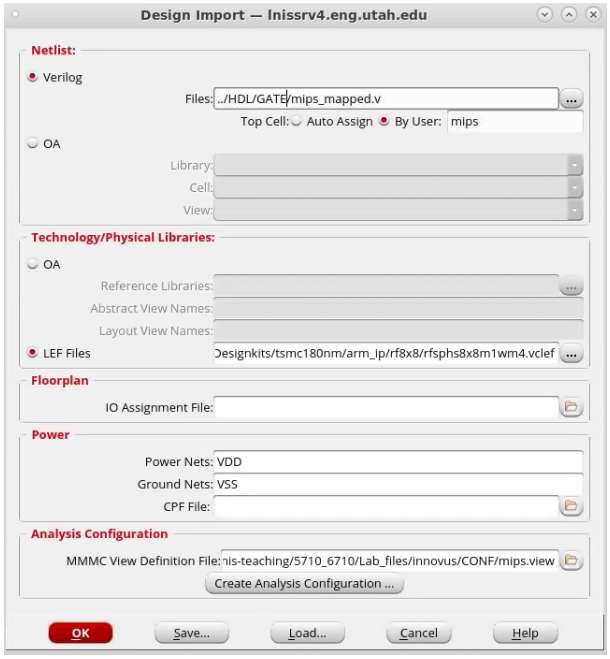


Figure 3: Design import window.

The design import phase still needs to be completed with the design data which is to be placed and routed. More particularly, you need to add the gate-level Verilog file netlist you synthesized in the previous lab:

1. From the design import window, next to the *Verilog* field, click on the **...** button as illustrated in Fig. 4.
2. Click on the >> button to get the *Netlist Selection* pane. Select the Verilog gate-level netlist in *HDL/GATE/mips_mapped.v*.
3. Click on *Add* then *Close*.

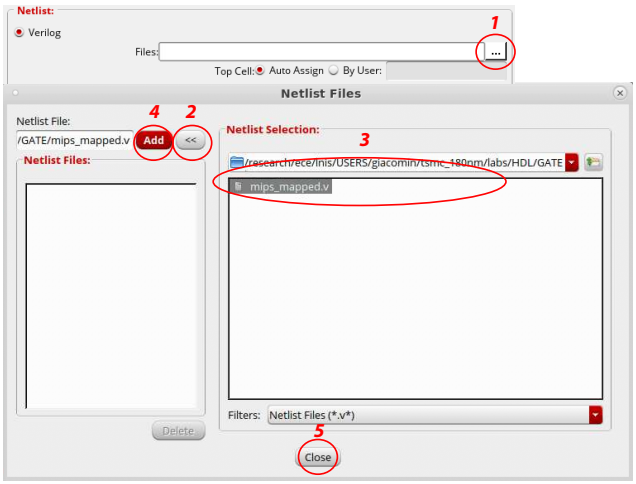


Figure 4: Adding the gate-level verilog file.

4. Next, fill the *Power Nets* and *Ground Nets* fields by *VDD* and *VSS* respectively, as in Fig. 3.
5. From the design import window, click on *Save* to save your configuration as *mips_final.globals* in the *CONF* directory. In case you need to modify some configuration later on, it will save all the settings you defined.
6. Finally, click on the *OK* button from the design import window the perform the design importation.

> **R** It is important to check that everything went smoothly after the importing step. Check the terminal to see if there was any error (a summary should be displayed at the very last lines of the terminal). If there is one, you need to check what went wrong by scrolling up into the terminal (you might have specified a wrong verilog file path for instance).

4.3 Importing the Design

This step needs only to be done when you will launch Innovus® again, to reload your design in its initial configuration. To import directly the design, without doing the previous step:

1. Go to *File -> Import Design...*
2. Click on the *Load..* button and select the *mips_final.globals* in the *CONF* directory.
3. Click on *OK*.

Equivalent Innovus command:

```
source CONF/mips_final.globals
init_design
```

4.4 Saving and Restoring the Design

The back-end flow contains several steps. Therefore, it is important to save your design after each step if you want to restore a specific state and restart from it without having to redo all the previous steps.

To save the current design:

1. Go to *File -> Save Design...*
2. Select the *Innovus* Data Type as shown in Fig. 5 and specify the file name *mips_nameofstep.enc*
3. Click on *OK*.

Equivalent Innovus command:

```
saveDesign DBS/mips_nameofstep.enc
```

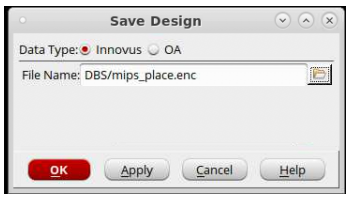


Figure 5: Saving your design.

> **R** When saving your design after each step, do not forget to change the name of the file (for instance, *mips_place.enc* after the placement step).

To restore a previous design:

1. Go to *File -> Restore Design...*
2. Select the *DB/Processor_nameofstep.enc* file to restore as depicted in Fig. 6.
3. Click on *OK*.

Equivalent Innovus command:

```
restoreDesign
DBS/mips_nameofstep.enc mips
```

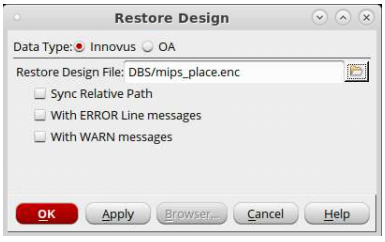


Figure 6: Restoring a previous design.

4.5 Floorplanning the Design

In this step, you will define the *core area* of your design, where the logic cells from your gate-level netlist from synthesis will be placed. You can define the aspect ratio, the form of the core, the distance between the core and the pads *etc.*. After the design import, a default floorplan is shown in the display area.

4.5.1 Defining the Floorplan Settings

To specify the floorplan:

1. Go to *Floorplan-> Specify Floorplan...*
2. Define the settings as in Fig. 7.
   The *Aspect ratio (H/W)*: means that your core area will have a square form.
   Core Utilization: 70% of the core area will be used for placing the standard cells and the 20% remaining will be free for power routing, buffer insertion etc.
   Core Margins: means that the distance between the core area to the core bounding box will be 20 μm.
3. Click on *OK*.

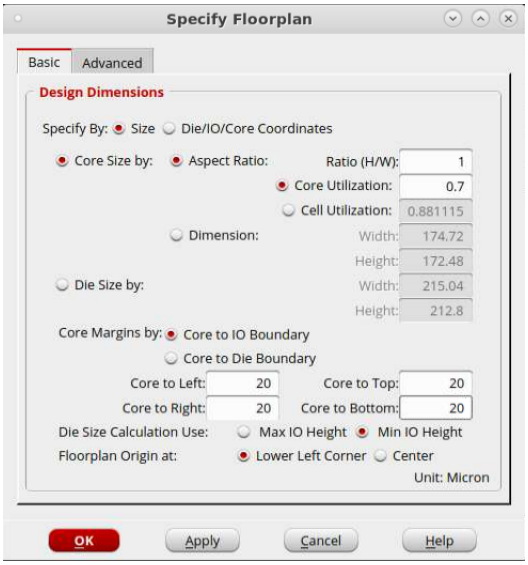


Figure 7: Specifying the floorplan.

Equivalent Innovus command:

```
floorPlan -site core7T -r 1 0.7 20
20 20 20
```

Before going any further, a setting has to be made in order for you to be able to display all the verilog modules on your floorplan, even the ones with a small size. To do so:

1. *View -> Set Preferences....*
2. Select the *Display* tab.
3. Next to the *Min. Floorplan Module Size*, replace 100 by 1.
4. Click Ok.

Equivalent Innovus command:

```
setPreference MinFPModuleSize 1
```

Now, you can look at the floorplan overall organization. Select the *Floorplan view icon* to display all the modules and blocks, as illustrated in Fig. 8. The objects on the left are the **modules** from your Verilog netlist. They can be moved and reshaped. The objects on the right of the core area are unplaced **blocks** that can only be moved (since their physical design is already defined by the IP provider). Those are generally the IP components such as an RF macro or an SRAM array. Select the *dp* module on the left, and press **shift+g**. You just now descended in its hierarchy and can observe the verilog modules it contains. This can be done for all the hierarchical levels. To revert this action and only display the higher module in terms of hierarchy, select the appropriate module and press **g**.

Each module shows a target utilization (TU) value that represents its physical design size. Left-clicking on a module or a block displays the pins and connection flight lines, which are the connections and number of connections between the selected module or block to any other modules and blocks.

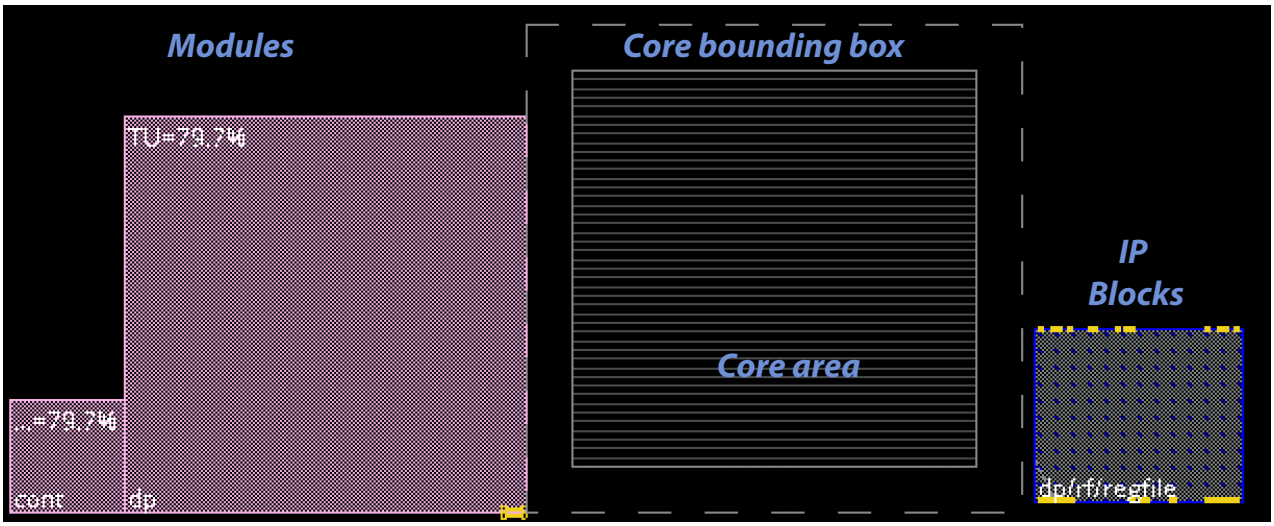
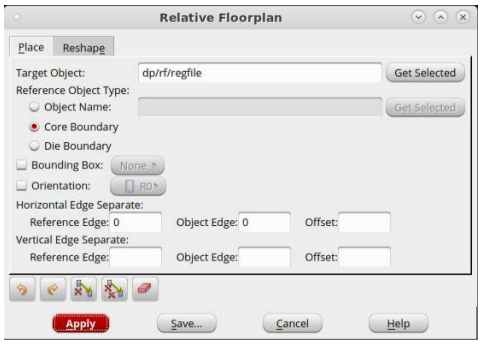


Figure 8: Floorplan View.

4.5.2 Placing the Register Block

In large designs, the cache or registers of your processor are generally provided as an IP block (or hard macro) whose layout and internal timing are already defined. In this lab, the *mips* processor uses a small 8×8 register, which is the IP block you can see on the right side of the floorplan view. To place the register IP:

1. Select the register IP by left clicking on it.
2. Go to *Floorplan-> Relative Floorplan -> Edit Constraints...*
3. Define the settings as in Fig. 9. It will place the register module on the left side of the core area, aligned with the bottom side.
4. Click on *Apply*. The register module is now placed. Close the window.



Equivalent Innovus command:

```
create_relative_floorplan -place dp/rf/regfile -ref_type core_boundary
-horizontal_edge_separate {0 "" 0} -vertical_edge_separate {"" "" ""}
```

Then, a placement halo around the register IP block has to be defined. It will prevent standard cells to be placed too closely to the block. Also, since there will be power rings around the block, this will ensure that the power rails of the standard cells are properly connect to the block power rings (see Section 4.6). To do so:

1. Right click on the register IP block and go to: *Edit Halo....*
2. Define the settings as in Fig. 10. It will add a placement halo on each sides of 2 μm. Note that the *Snap to Site* option was selected, so the Halo is rounded up to the nearest site, so the enarest standard cell row. In P&R, a SITE has a SIZE which is the base area that the SITE occupies. A macro or a standard cell sit on a SITE, and its size must match or be an integral multipliers of the base SITE. In this lab, you are using a SITE of size 0.560 $\times$ 3.920, meaning that each standard cell has a height of 3.920 μm and a width of a multiple of 0.560 μm.
3. Click on *OK*.

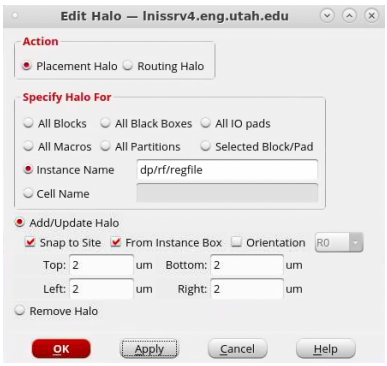


Figure 10: Adding a placement halo around the register block.

Equivalent Innovus command:

```
addHaloToBlock 2 2 2 2 -fromInstBox -snapToSite dp/rf/regfile
```

Before going any further, you can notice that some gap between the register IP and the core boundary. This is because the register file has some integrated power rings. To display those, in the *Visibility Toggles* on the right side, expend *Cell* and check *Pin Shapes* and uncheck *Cell Blockage*, as shown in Fig. 11. You can now observe the power rings around the IP block.

Figure 11: Visibility toggles for the IP block.

4.6 Defining the Power Structure

This step will define the power structure of the chip, such as the power rings and the power stripes around the core and will also route all the power nets.

4.6.1 Logically Connecting all the Power/Ground Nets

The netlist does not include any power and ground connection but the cells that will be placed do have power and ground connections. Therefore, you need to logically connect all the power nets of the cells to the global power net you defined during the import phase (V_{DD} and V_{SS}).

To do so:

1. Go to *Power-> Connect Global Nets...*
2. For each global net (V_{DD} and V_{SS}), do as in Fig. 12.
3. Click on *OK* then *Cancel*.

Equivalent Innovus command:

```
globalNetConnect VDD -type pgpin -pin VDD \
-instanceBasename * -hierarchicalInstance {}
globalNetConnect VSS -type pgpin -pin VSS \
-instanceBasename *-hierarchicalInstance {}
```

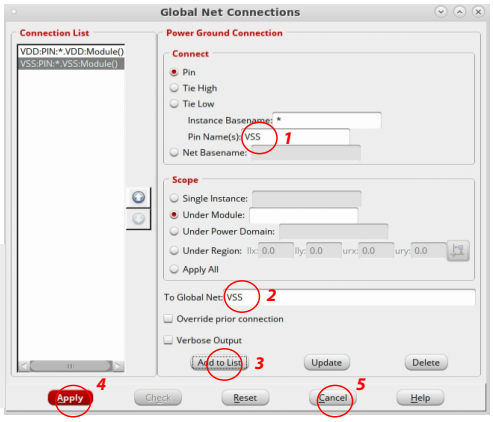


Figure 12: Connecting the power nets.

4.6.2 Adding the Power Rings Around the Core

Now that the power and ground nets are logically assigned, power planning can be done. First, lets add the power ring around the core area.

1. Go to *Power -> Power Planning -> Add Ring...*
2. Specify the power and ground nets as in Fig. 13.
3. Specify the layers and the width of each ring as well as the spacing between each ring, as in Fig. 13. The *Ring Configuration* defines the ring metal wires widths and spacing. Here, the horizontal power metal wires will be in metal5 and the vertical ones in metal4. The ring will be placed in the center of the channel between the core and the chip boundary (or the I/O pads, if there are some).
4. Click on *OK*.

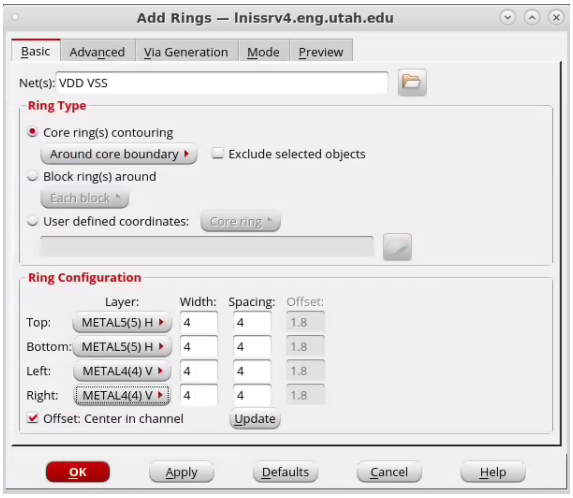


Figure 13: Adding the power ring around the core area.

Equivalent Innovus command:

```
addRing -nets {VDD VSS} -type core_rings -follow core -layer \
{top METAL5 bottom METAL5 left METAL4 right METAL4} -width \
{top 4 bottom 4 left 4 right 4} -spacing \
{top 4 bottom 4 left 4 right 4} -offset \
{top 1.8 bottom 1.8 left 1.8 right 1.8} -center 1 -extend_corner { }\
-threshold 0 -jog_distance 0 -snap_wire_center_to_grid None
```

Assignment

Which metal layers should be used for power routing and for the CTS and why (it is not necessarily the default one used in this lab)?

4.6.3 Adding Power Stripes

Power stripes are vertical or horizontal power lines spanning across the core area. They can be needed to ensure the proper power distribution in large cores to avoid IR drop. To do so:

1. Go to *Power -> Power Planning -> Add Stripes...*
2. Define the settings as in Fig. 14. It will create 2 pairs of VDD and VSS power stripes in metal4, starting and ending 50 μm of the left and right sides.
3. Click on *OK*. You can now see the vertical power stripes on the floorplan.

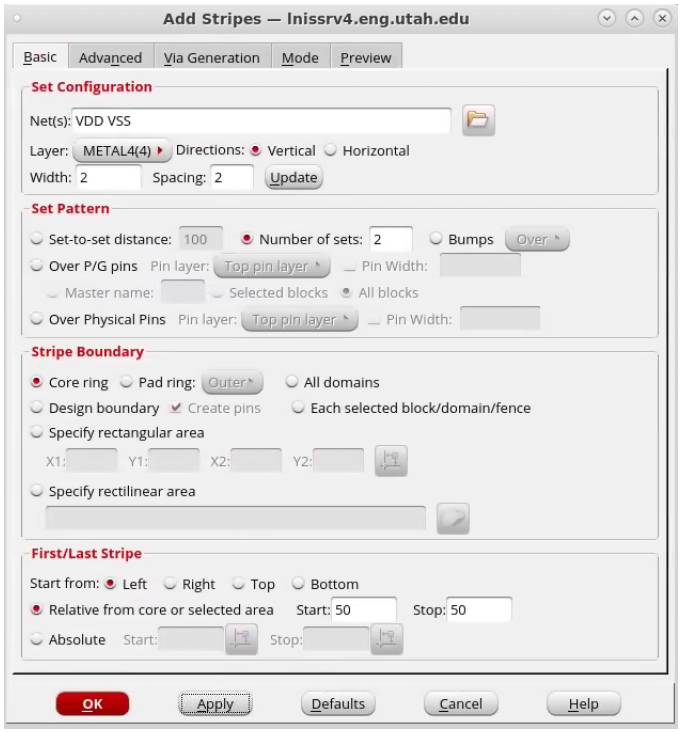


Figure 14: Adding Power Stripes.

Equivalent Innovus command:

```
addStripe -nets VDD VSS -layer METAL4 -direction vertical -width 2 -spacing 2
-number_of_sets 2 -start_from left -start_offset 50 -stop_offset 50 -switch_layer_over_obs
false -max_same_layer_jog_length 2 -padcore_ring_top_layer_limit METAL6 -padcore_ring_bottom_
METAL1 -block_ring_top_layer_limit METAL6 -block_ring_bottom_layer_limit METAL1
-use_wire_group 0 -snap_wire_center_to_grid None
```

4.6.4 Routing the Power Nets

Now, the power net can be routed. This step will create the vertical V_{DD} and V_{SS} power lines on the entire core area. This is why it is important, when designing full custom cells (as you did in the previous labs), to use the same height for each of your cells so they fit between those power lines.

1. Go to *Route -> Special Route...*
2. In the *Basic* tab, specify the power net name as shown in Fig. 15. Only check the *Block Pins* and *Follow Pins* boxes.
3. Click on *OK*. You can now see the horizontal power lines on the floorplan.

Equivalent Innovus command:

```
sroute -connect {blockPin corePin} -layerChangeRange {METAL1(1) METAL6(6)} \
-blockPinTarget { nearestTarget } -corePinTarget { firstAfterRowEnd } \
-allowJogging 1 -crossoverViaLayerRange {METAL1(1) METAL6(6)} \
-nets { VDD VSS } -allowLayerChange 1 -blockPin useLef \
-targetViaLayerRange {METAL1(1) METAL6(6)}
```

The floorplan of your design is done. It should looks like Fig. 16. Save your design as *DBS/mips_fplan_power.enc*, as explained in Section 4.4. For more complex design, the floorplanning is generally the most important step since many particular modules (such as IP, RF modules or caches) have to be placed, requiring many halo for routing and placement, special power rings and stripes, *etc.*

```
saveDesign DBS/mips_fplan_power.en
```

Checkpoint
Please call an assistant and show him that your floorplan looks correct.

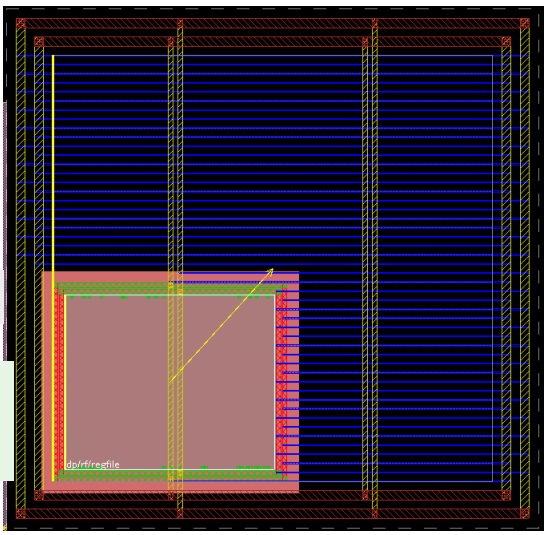


Figure 16: Design after the floorplaning step.

4.7 Placing the Standard Cells

This step will place the standard cells (contained in your Verilog netlist) in the floorplan rows.

1. Go to *Place -> Place standard Cells...*
2. Click on *Mode....*
3. In the *Placement* tab, select *Place IO pins*, as shown in Fig. 17.
4. Click on *Ok* then on *Ok*.

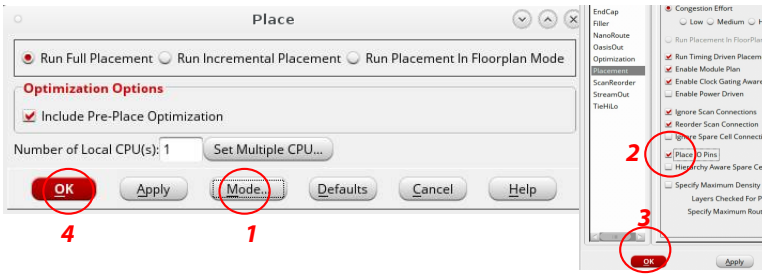


Figure 17: Placement settings.

Equivalent Innovus command:

```
setPlaceMode -placeIOPins 1
place_design
```

Select the Physical view icon to see the result of the placement in the display area. It should look something like Fig. 18 The placement has been ran in *Timing-Driven mode*. That means the Placement program balances the importance of meeting setup timing constraints (from the *.sdc* file) with routability. This is why you can see routing wires. However, those are not the final ones. The final ones will be obtained after the Routing step.
As you can see, some rows have holes. This is because you specified a core utilization of 80% when defining the floorplan properties. This space will be filled by buffers or inverters (for the clock tree). Some cells will also be replaced (by the same cell but with a higher driving capability) when performing timing-driven optimization in the next steps.

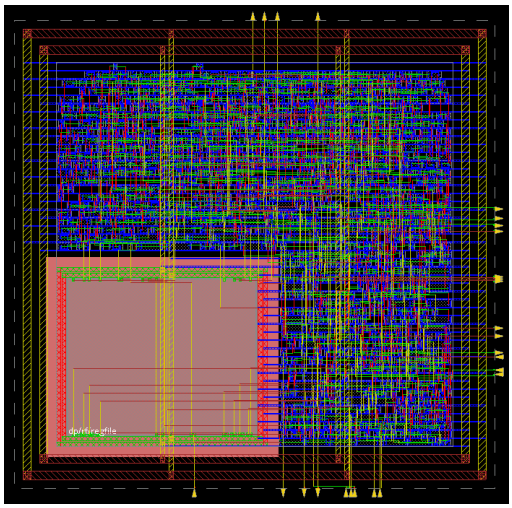


Figure 18: Placed design.

After the placement step, you can obtain some information about the design:

1. Go to *Place -> Check Placement....*
2. You can check or not the *Check Placement report to* if you want to keep the report in a file (the recommended location is then in the file *RPT/mips.Checkplace.rpt*.
3. Click on *Ok*. You can now see the density of your design (which should be close to 80%), the number of placed cells, *etc.*
   Equivalent Innovus command:

```
checkPlace RPT/mips.checkPlace.rpt
```

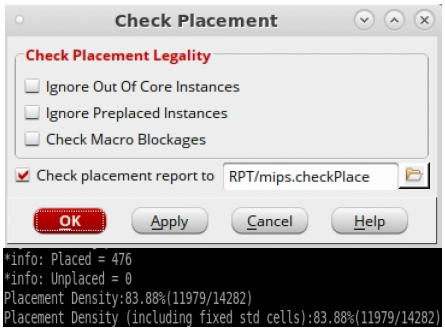


Figure 19: Checking the placement.

After the placement step, you can obtain a first estimation of the critical path. This is only an estimation since the routing step has not been completed yet. To do so:

1. The first step is to tell Innovus® which technology node you are using. It will ensure better accuracy results when extracting the timing delay. In the terminal, write:

```
setDesignMode -process 180
```

2. Go to *Timing -> Report Timing...*
3. Keep the same settings as depicted in Fig. 20. Especially, keep the *Pre-CTS* box checked since we want to get the timing before the *CTS* step.
4. Click on *Ok*.

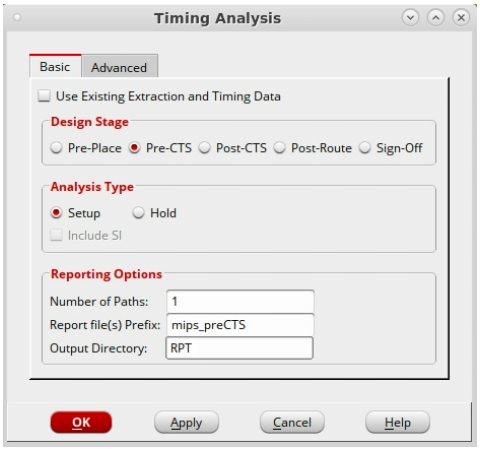
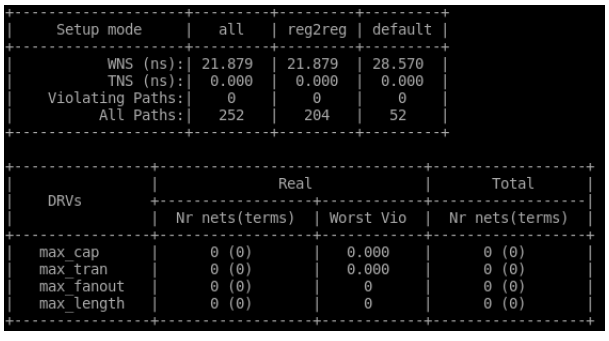


Figure 20: Timing analysis settings.

Equivalent Innovus command:

```
timeDesign -preCTS -pathReports -drvReports -slackReports -numPaths 1 \
-prefix mips_preCTS -outDir RPT
```

On the console terminal, you can now see some timing information, as sown in Fig. 21. The *Worst Negative Slack* (WNS) is the slack of the critical path in the design. Here, it has a positive value so the timing constraints are already met. Even if it had a negative value, the slack could still become positive after the next optimization step. On the *DRC* table, you might see a few violations such as *max_cap, max_tran, etc.*. Those can be resolved by optimizing your design.



Setup mode	all	reg2reg	default
WNS (ns):	21.879	21.879	28.570
TNS (ns):	0.000	0.000	0.000
Violating Paths:	0	0	0
All Paths:	252	204	52

DRVs	Real		Total
	Nr nets(terms)	Worst Vio	Nr nets(terms)
max_cap	0 (0)	0.000	0 (0)
max_tran	0 (0)	0.000	0 (0)
max_fanout	0 (0)	0	0 (0)
max_length	0 (0)	0	0 (0)

Figure 21: Pre-CTS slack report.

To optimize your design after the place step:

1. Go to: *ECO -> Optimize Design...*
2. Select *Pre-CTS* as in Fig. 22 and click *Ok*.
3. When it is done, observe how the slack changed on the terminal. Depending on if you had a large slack or a negative slack, the optimization results can be very different. If you had a large slack, the tool will not be able to optimize a lot since the design already meets the timing constraints easily.

```
optDesign -preCTS
```

4. Save your design as *DBS/mips_place.enc*, as explained in Section 4.4.

```
saveDesign DBS/mips_place.enc
```

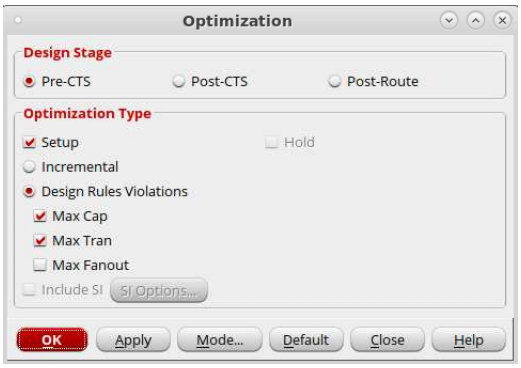


Figure 22: Optimizing the design.

Assignment

1. Report the critical path timing before and after optimization after the place step.
2. During the optimization step, how do you think Innovus® optimized the design in order to improve the slack?

4.8 Synthesizing the Clock Tree

Since the paths propagating the *clock* signal are not all balanced, some registers will receive the *clock* signal after or before the others. If that happens, that means that the design would not be synchronous anymore. To correct this, the tool will insert some inverters in the design to add some delay so each register will receive the *clock* signal at the same time. First, you can take a quick look at the initial clock tree as follow:

1. Go to *Tools -> Design Browser...*.
2. Under *Nets*, select *clk*.
3. On the visibility toggles (on the right part of the screen, as explained in Fig. 1), deselect *Net* and *Cell* and select the *Floorplan* view. That way, you can see the clock tree, as depicted in Fig. 23.
4. Before going to the next step, don't forget to select back the *Net* and *Cell* fields in the visibility toggles.
5. Before synthesizing the clock tree, several options can be performed such as specifying the list of usable inverters or buffers or specifying the width and spacing of the metals the clock tree should use. In this lab, we will use the default settings so there is nothing to specify. To do the *Clock Tree Synthesis* (CTS), do:

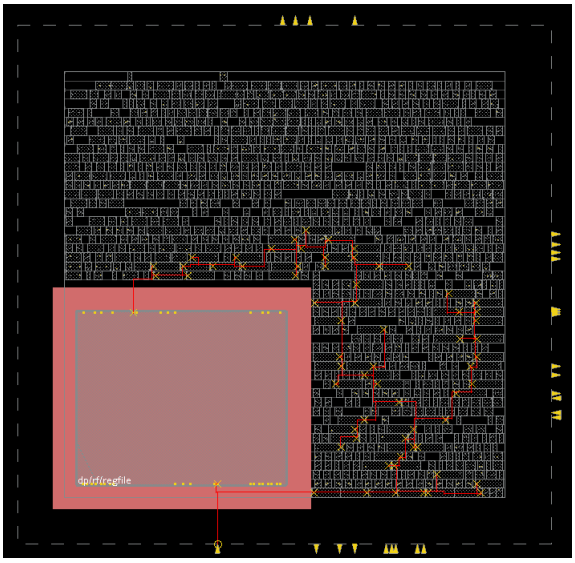


Figure 23: Clock tree path.

```
ccopt_design -cts
```

Assignment

Report the critical path timing before and after optimization after the CTS step, for both setup and hold mdoes (you will have to report the timing twice). Do not forget to select the appropriate *Design Stage* for both the *Timing Analysis* and *Optimization* windows since the CTS step is now done. **When optimizing the design, select both setup and hold fields in the *Optimization Type* field.**

```
timeDesign -postCTS
timeDesign -postCTS -hold
optDesign -postCTS -setup -hold
```

6. Save your design as *DBS/mips_cts.enc*, as explained in Section 4.4.

```
saveDesign DBS/mips_cts.enc
```

4.9 Routing the Design

This step will connect all your standard cell together as well as the I/O pins and will create the resulting wires. To route your design:

1. Go to *Route -> NanoRoute -> Route...*.
2. Select the same options as in Fig. 24.
3. Click on *OK*.

Equivalent Innovus command:

```
setNanoRouteMode \
-routeWithTimingDriven true \
-routeTdrEffort 5 \
-routeWithSiDriven true \
-drouteFixAntenna true \
-routeInsertAntennaDiode true \
-routeAntennaCellName ANTENNA
 routeDesign -globalDetail -wireOpt \
-viaOpt
```

Figure 24: Routing option window.

4. You can now observe the final routed design on the design display area.
5. As before, report the timing (and select the *Post-Route* option). If necessary, optimize your design as before.

```
setAnalysisMode -analysisType onChipVariation
timeDesign -postRoute
timeDesign -postRoute -hold
optDesign -postRoute -setup -hold
```

6. Save your design as *DBS/mips_route.enc*, as explained in Section 4.4.

```
saveDesign DBS/mips_route.enc
```

4.10 Adding the Filler Cells

Since you specified a core utilization of 80%, there is still some empty space in your design. The ensure the continuity of the doping and well zones, some filler cell (which do not have any logic function) have to be adder.

1. Go to *Place -> Physical Cell -> Add Filler...*.
2. Click on the *Select* button and choose the appropriate filler cells in the list, as in Fig. 27.
3. Click on *Close* and then *OK*.

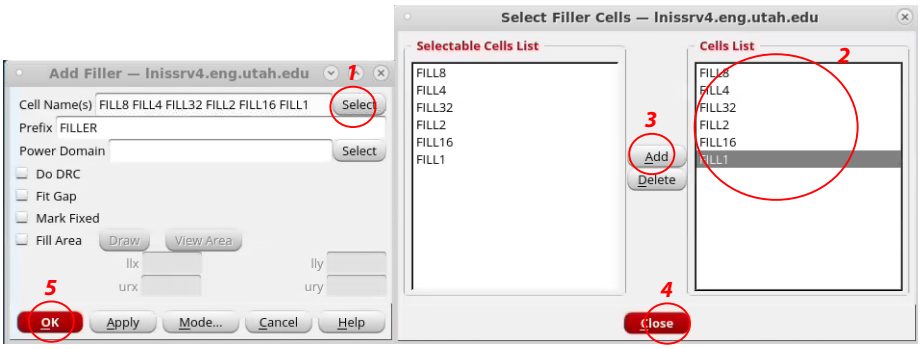


Figure 25: Adding the filler cells.

Equivalent Innovus command:

```
addFiller -cell FILL32 FILL16 FILL8 FILL4 FILL2 FILL1 -prefix FILLER
```

4. Observe the new layout. There should not be any empty space anymore. You can also look at the console which displays which filler cells have been added.
5. Save your design as *DBS/mips_fill.enc*, as explained in Section 4.4.

```
saveDesign DBS/mips_filler.enc
```

4.11 Verifying the Design

4.11.1 Verifying the Connectivity

This step will check your design for open or unconnected wires, unconnected pins, partial routing or unrouted nets.

1. Go to *Verify -> Verify Connectivity...*.
2. Select the same options, as shown in Fig. 26.
3. Specify the *Verify Connectivity Report* as *RPT/mips_conn.rpt* in case you want to save the report.
4. Click *Ok*.
5. Check the console terminal for possible errors.

Figure 26: Verifying the connectivity.

Equivalent Innovus command:

```
verifyConnectivity -noAntenna -type all \
-report RPT/mips_conn.rpt -error 1000 \
-warning 50
```

4.11.2 Verifying the Geometry

This step will check several DRC rules such as the metal minimum widths, shorts, minimum cut, *etc.*

1. Go to *Verify -> Verify DRC...*.
2. Specify the *Verify DRC Report* field in the *Advanced* tab as *RPT/mips_drc.rpt* in case you want to save the report.
3. In the *Advanced* tab, check the *Regular Routing* option under *Check Objects*.
4. Click *Ok*.
5. Check the console terminal for possible errors.

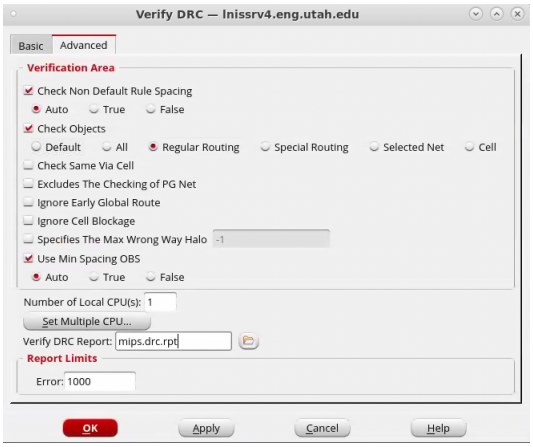


Figure 27: Verifying the geometry.

Equivalent Innovus command:

```
set_verify_drc_mode -check_only regula
verify_drc -report RPT/mips_geo.rpt
```

Assignment
If you were to have a lot of DRC errors after your P&R flow due to many metal wires being to close to each other, what could you do to address this issue?

Checkpoint
Please call an assistant and show him that your final design looks correct after the verifying steps.

4.12 Generating the Reports

The previous steps should have generated several reports in the *RPT* folder. It is possible to generate additional report.

4.12.1 Netlist Statistics

Select *File -> Report -> Netlist Statistics*. It will report several information about your design, as shown in Fig. 28. Equivalent Innovus command:

```
reportNetStat
```

If you want to save the output somewhere, do:

```
reportNetStat » RPT/name_of_your_file
```

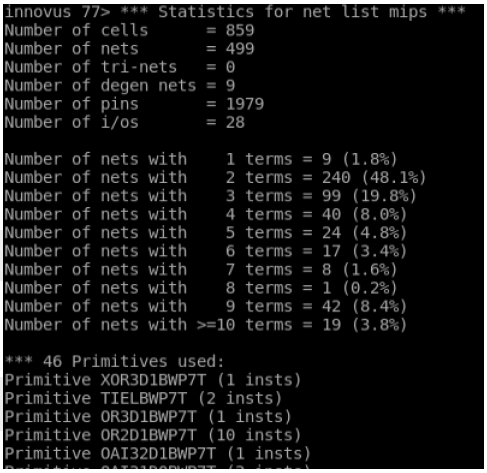


Figure 28: Netlist statistic report.

4.12.2 Reporting the Critical Path

To report the critical path timing, type in the terminal:

```
report_timing
```

```
Endpoint:   dp/pcreg/q_reg[1]/D          (^) checked with  leading edge of 'clk'
Beginpoint: cont/statelog/state_reg[1]/Q (^) triggered by  leading edge of 'clk'
Path Groups: {clk}
Analysis View: wc
Other End Arrival Time          0.104
- Setup                         0.173
+ Phase Shift                  20.000
= Required Time                19.930
- Arrival Time                 10.357
= Slack Time                    9.573
     Clock Rise Edge                0.000
     + Clock Network Latency (Prop)  0.003
     = Beginpoint Arrival Time      0.003
     +-----------------------------------------------------------------------------------------------------+
     |                 Instance                 |     Arc      |   Cell   | Delay | Arrival | Required |
     |                                          |              |          |       |  Time   |   Time   |
     |------------------------------------------+--------------+----------+-------+---------+----------|
     | cont/statelog/state_reg[1]               | CLK ^        |          |       |   0.003 |    9.576 |
     | cont/statelog/state_reg[1]               | CLK ^ -> Q ^ | DFFQQBX1 | 0.568 |   0.571 |   10.144 |
     | cont/outputlog/U13                       | A ^ -> Z v   | INVX2    | 0.233 |   0.804 |   10.377 |
     | cont/outputlog/U36                       | A v -> Z ^   | NAND3X1  | 0.177 |   0.981 |   10.554 |
     | cont/outputlog/U12                       | A ^ -> Z v   | INVX2    | 0.168 |   1.149 |   10.722 |
     | cont/outputlog/U35                       | A v -> Z ^   | NAND2X1  | 0.129 |   1.278 |   10.851 |
     | cont/outputlog/U34                       | C ^ -> Z v   | NAND3X1  | 0.478 |   1.756 |   11.329 |
     | dp/src1mux/U1                            | A v -> Z ^   | INVX2    | 0.385 |   2.141 |   11.714 |
     | dp/src1mux/U21                           | B ^ -> Z v   | NAND2X1  | 0.135 |   2.276 |   11.849 |
     | dp/src1mux/U20                           | B v -> Z ^   | NAND2X1  | 0.609 |   2.884 |   12.457 |
     | dp/alunit/FE_OFC42_srca_1                | A ^ -> Z v   | INVX1    | 0.349 |   3.234 |   12.807 |
     | dp/alunit/FE_OFC43_srca_1                | A v -> Z ^   | INVX1    | 0.913 |   4.147 |   13.720 |
     | dp/alunit/addblock/add_1_root_add_645_2/U5  | A ^ -> Z v   | INVX2    | 0.181 |   4.328 |   13.901 |
     | dp/alunit/addblock/add_1_root_add_645_2/U41 | B v -> Z ^   | NAND2X1  | 0.167 |   4.495 |   14.068 |
     | dp/alunit/addblock/add_1_root_add_645_2/U40 | B ^ -> Z v   | NAND2X1  | 0.089 |   4.584 |   14.157 |
     | dp/alunit/addblock/add_1_root_add_645_2/U39 | B v -> Z ^   | NAND2X1  | 0.236 |   4.821 |   14.394 |
     | dp/alunit/addblock/add_1_root_add_645_2/U34 | A ^ -> Z ^   | OR2X1    | 0.208 |   5.029 |   14.602 |
     | dp/alunit/addblock/add_1_root_add_645_2/U33 | B ^ -> Z v   | NAND2X1  | 0.088 |   5.116 |   14.689 |
     | dp/alunit/addblock/add_1_root_add_645_2/U32 | B v -> Z ^   | NAND2X1  | 0.212 |   5.328 |   14.901 |
     | dp/alunit/addblock/add_1_root_add_645_2/U4  | A ^ -> Z v   | INVX2    | 0.159 |   5.488 |   15.061 |
     | dp/alunit/addblock/add_1_root_add_645_2/U29 | A v -> Z ^   | NAND2X1  | 0.116 |   5.604 |   15.177 |
     | dp/alunit/addblock/add_1_root_add_645_2/U28 | B ^ -> Z v   | NAND2X1  | 0.086 |   5.689 |   15.262 |
     | dp/alunit/addblock/add_1_root_add_645_2/U27 | B v -> Z ^   | NAND2X1  | 0.199 |   5.889 |   15.462 |
     | dp/alunit/addblock/add_1_root_add_645_2/U22 | A ^ -> Z ^   | OR2X1    | 0.211 |   6.100 |   15.673 |
     | dp/alunit/addblock/add_1_root_add_645_2/U21 | B ^ -> Z v   | NAND2X1  | 0.078 |   6.178 |   15.751 |
     | dp/alunit/addblock/add_1_root_add_645_2/U20 | B v -> Z ^   | NAND2X1  | 0.155 |   6.333 |   15.906 |
     | dp/alunit/addblock/add_1_root_add_645_2/U2  | A ^ -> Z v   | INVX2    | 0.136 |   6.469 |   16.042 |
     | dp/alunit/addblock/add_1_root_add_645_2/U17 | A v -> Z ^   | NAND2X1  | 0.111 |   6.580 |   16.153 |
     | dp/alunit/addblock/add_1_root_add_645_2/U16 | B ^ -> Z v   | NAND2X1  | 0.086 |   6.665 |   16.238 |
     | dp/alunit/addblock/add_1_root_add_645_2/U15 | B v -> Z ^   | NAND2X1  | 0.209 |   6.874 |   16.447 |
     | dp/alunit/addblock/add_1_root_add_645_2/U11 | A ^ -> Z ^   | OR2X1    | 0.204 |   7.078 |   16.651 |
     | dp/alunit/addblock/add_1_root_add_645_2/U10 | B ^ -> Z v   | NAND2X1  | 0.074 |   7.152 |   16.725 |
     | dp/alunit/addblock/add_1_root_add_645_2/U9  | B v -> Z ^   | NAND2X1  | 0.132 |   7.284 |   16.857 |
     | dp/alunit/addblock/add_1_root_add_645_2/U7  | A ^ -> Z ^   | XOR2X1   | 0.211 |   7.496 |   17.069 |
     | dp/alunit/resultmux/U175                 | A ^ -> Z v   | NAND2X1  | 0.115 |   7.610 |   17.183 |
     | dp/alunit/resultmux/U172                 | B v -> Z ^   | NAND3X1  | 0.183 |   7.793 |   17.366 |
     | dp/alunit/resultmux/U162                 | A ^ -> Z v   | NOR2X1   | 0.182 |   7.976 |   17.549 |
     | dp/alunit/resultmux/U144                 | A v -> Z ^   | NAND2X1  | 0.310 |   8.285 |   17.858 |
     | dp/alunit/zd/U4                          | B ^ -> Z v   | NOR2X1   | 0.138 |   8.424 |   17.997 |
     | dp/alunit/zd/U2                          | A v -> Z ^   | NAND2X1  | 0.159 |   8.583 |   18.156 |
     | dp/alunit/zd/U1                          | B ^ -> Z v   | NOR2X1   | 0.163 |   8.745 |   18.318 |
     | cont/U2                                  | A v -> Z v   | AND2X1   | 0.172 |   8.917 |   18.490 |
     | cont/U1                                  | A v -> Z v   | OR2X1    | 0.163 |   9.080 |   18.653 |
     | dp/pcreg/U3                              | A v -> Z ^   | INVX2    | 0.056 |   9.136 |   18.709 |
     | dp/pcreg/U29                             | A ^ -> Z v   | NOR2X1   | 0.352 |   9.488 |   19.061 |
     | dp/pcreg/U28                             | B v -> Z ^   | NOR2X1   | 0.571 |  10.059 |   19.632 |
     | dp/pcreg/U9                              | B ^ -> Z v   | NAND2X1  | 0.117 |  10.176 |   19.749 |
     | dp/pcreg/U7                              | A v -> Z ^   | NAND2X1  | 0.181 |  10.357 |   19.930 |
     | dp/pcreg/q_reg[1]                        | D ^          | DFFQX1   | 0.000 |  10.357 |   19.930 |
     +-----------------------------------------------------------------------------------------------------+
```

Figure 29: Critical path timing report.

4.12.3 Reporting the Total Area

To report the area of your design, type in the terminal:

```
report_area
```

It will report the area of each module (*mips* being the top module) and specify its hierarchy level.

Depth	Name	#Inst	Area (um^2)
0	mips	478	12075.7952
1	cont	70	928.5696
1	dp	407	11098.9312
2	dp/srclmux	9	147.0784
2	cont/statelog	30	509.2864
2	dp/pcreg	9	480.7488

Figure 30: Area report.

4.13 Exporting the Final Design

4.13.1 Generating the SDF timing file

This step will generate the *Standard Delay Format* (SDF) file which includes the gate and wire delays. It is then used for post-backend functional verification (with Modelsim® for instance). To do so:

1. Note: The SDF can only be exported if a timing analysis has been previously done (through *TimeDesign* or *report_timing* for instance.
2. Go to *Timing -> Write SDF...*
3. Select the *wc* active view (worst case, so for setup analysis).
4. Specify the output file as SDF/mips_placed.sdf
5. Click *Ok*.

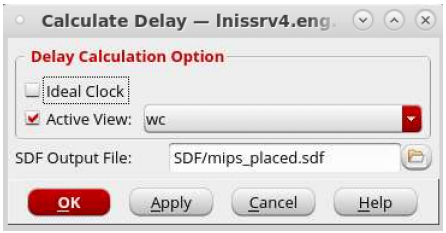


Figure 31: Writing the SDF file.

Equivalent Innovus command:

```
write_sdf -view wc SDF/mips_placed.sdf
```

4.13.2 Generating the Verilog Gate-level Netlist

As you did after the synthesis step, you can generate a verilog gate-level netlist after the P&R flow. This netlist might differ from the initial netlist since some gates could have been resized or added (during the CTS for instance). To generate the file:

1. Go to *File -> Save -> Netlist...*
2. Deselect the *Include Leaf Cell Definition* field.
3. Specify the netlist file as HDL/GATE/mips_placed.v
4. Click *Ok*.

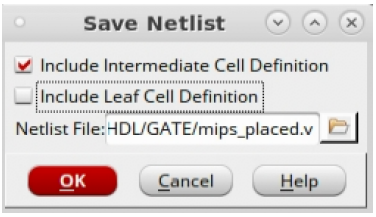


Figure 32: Generating the final verilog netlist.

Equivalent Innovus command:

```
saveNetList -excludeLeafCell HDL/GATE/mips.placed.v
```

4.13.3 Generating the GDSII File

The design can be exported as a GDSII file which is a binary file format representing the planar geometric shapes, text labels and other information about your design. It is generally used to be imported into Virtuoso® or to be sent to the foundry for fabrication.

1. Go to *File -> Save -> GDS/OASIS...*
2. As in Fig. 33, fill out the appropriate fields.
3. Click *Ok*.
4. Check in the terminal to verify that the *Streamout* phase has been done without any issue.
   Equivalent Innovus command:

```
streamOut GDS/mips.gds -mapFile
GDS/gds2.map -libName DesignLib
-structureName mips -units 2000 -mode
ALL
```

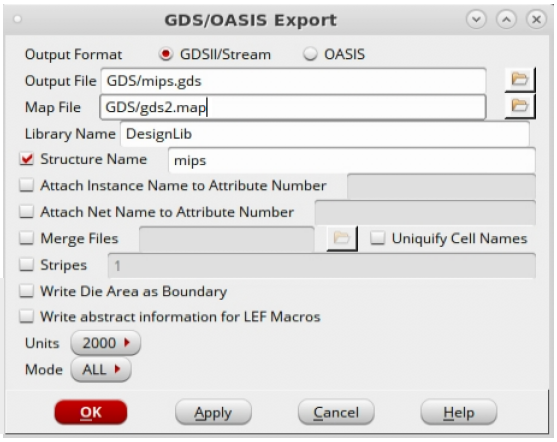


Figure 33: Generating the GDSII file.

4.14 Importing the Final Design into Virtuoso®

To import the design after the P&R steps into Virtuoso®:

1. From the virtuoso CIW window, *File -> Import -> Stream...*.
2. In the *XStream In* window, specify the path of your GDSII file (which should be in the *GDS* folder of your *innovus* directory) as the *Stream File*.
3. Specify a library name. If this library does not exist, it will be created.
4. Specify the *Top Cell* name which should match the one you used when exporting the GDSII file from Innovus®.
5. In the **Technology** tab, select *tsmc18* so your new library will be attached to the TSMC 180nm technology library.
6. Click on *More Options*. Next to the *Ref Lib File Name*, click on . Select the *SC_LIB* library and click on the right arrow. Do the same for the *PADS_LIB* and *rf2hsm1wm1* libraries. This will ensure that this standard cell library is linked to your design during the importation phase. Since your top design is using many of those cells, they would be considered as black boxes otherwise. Click on *Save As and Exit*, set the file name to *reflib.list* and click on *Save*.
7. Click on *Ok* then on *Translate* in the XStream In window.
8. The translation should start and you should not get any error.

Now, go to the *Library Manager* and open the *mips* layout view in your newly imported library. You should obtain something similar as Fig. 35. If you only see metal wires, do **Shift+F** to also display the cells and the IP. If you zoom in, you can see all the standard cells from the standard cell library. As you can see, only the metal layers are shown for the IP block and nothing about the transistor or doping layers. This is because the companies selling IPs generally only provide *abstract* views for the P&R flow (which just needs the metal information from the standard cells) for confidentiality reasons.

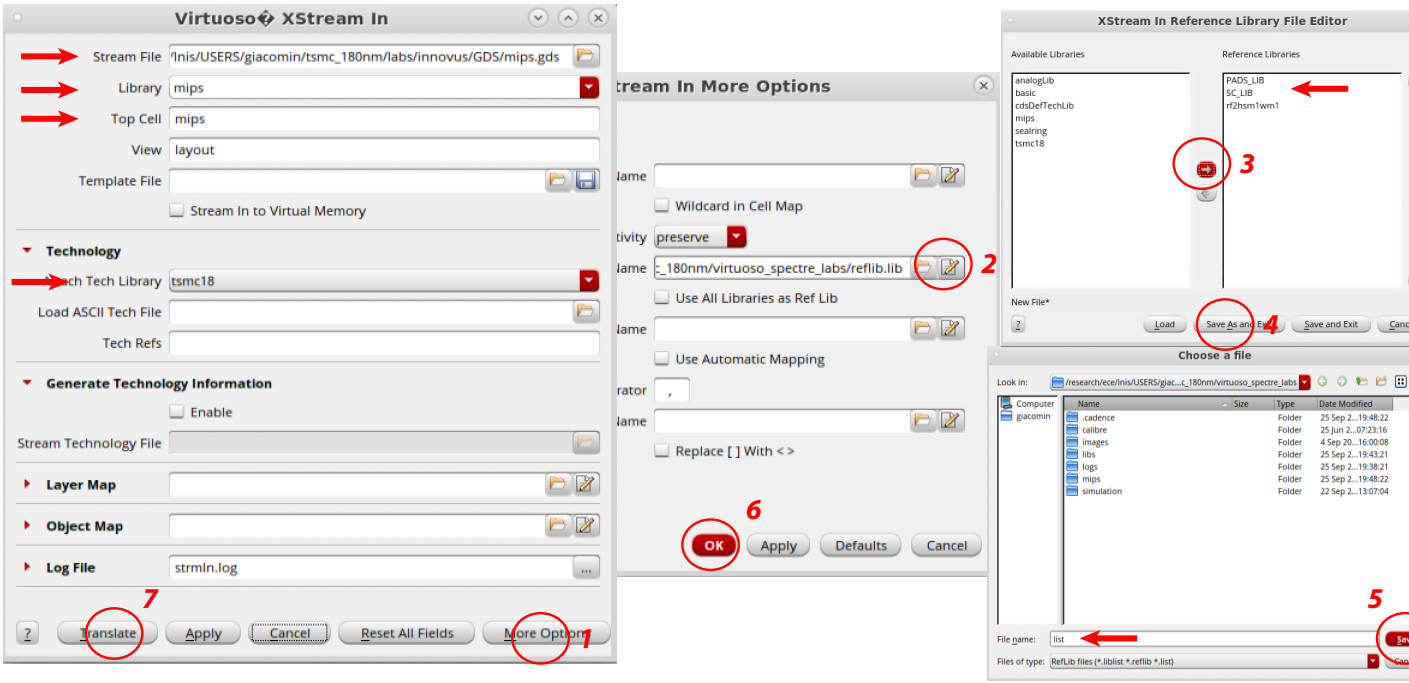


Figure 34: Importing a GDSII file into Virtuoso®.

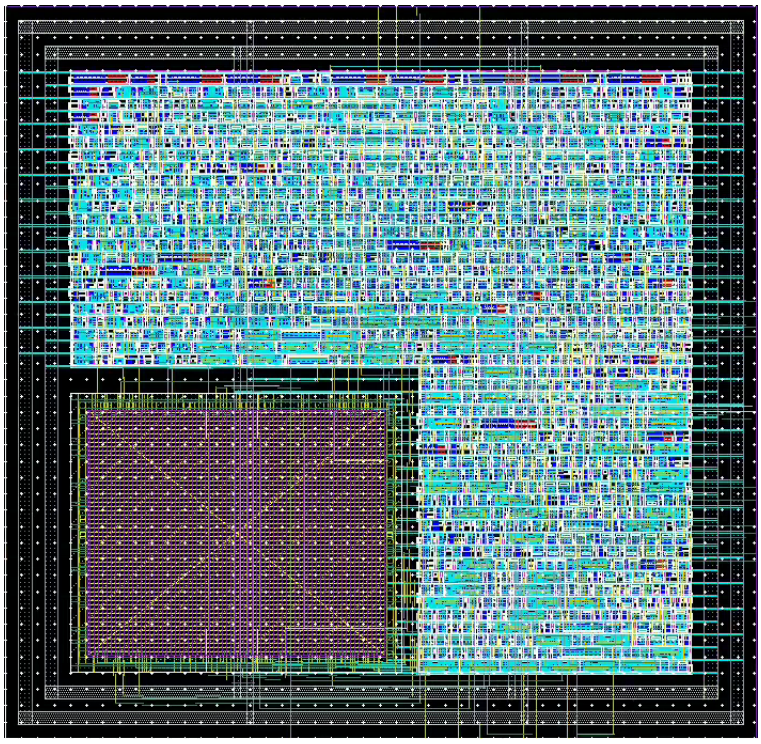


Figure 35: MIPS layout view from Virtuoso®.

4.14.1 Using a TCL script

As in the previous lab, instead of using the GUI, you could use a *.tcl* script to automatize your flow and optimize your design with tighter time constraints. The *place_route.tcl* script in your *innovus/SCRIPTS* directory contains all the commands of this lab. Open it and take a look at how it is organized. You can see that it contains several procedures, each of which will perform one step (importing the design, saving the design, place, route, CTS, *etc.*). Try to understand what each procedure is for, you should recognize most of the commands used in this lab. Also, pay attention where each report is created at the end of each step (more particularly for the *generate_reports* step). In each procedure, many variables are defined and then passed as parameters for the common Innovus commands®. The main procedure is called: **do_steps** and has two arguments: *first_step last_step*. Look at the very end of the *place_route.tcl* to see how it is defined: by passing only one argument number (2 for instance), a single step is performed (*floorplan_design* in this case). By passing two arguments (1 6 for instance), several steps are performed in order (*init_design* to *route_power_nets* for instance). In order to use this script, launch Innovus® and do:

```
source SCRIPTS/place_route.tcl
set_design mips
```

One way to do the whole flow would be to do:

```
do_steps 1 12
```

For the first time, it is advised to run the steps 1 by 1, to ease the debugging:

```
do_steps 1
do_steps 2
do_steps 3
...
```

If something is not working, check the terminal as it will indicate what went wrong and try to understand what is the issue and how to address it. Sometimes, a file name has not been specified correctly, or you could have forgotten to specify the *design_name* for instance. Some common errors could come from your *mips_final.globals* file which defines the path of many configuration file. The latter file depends on the *mips.view*, located in the */research/ece/lnis-teaching/5710_6710_F20/Lab_files/innovus/CONF/* directory. This *view* file depends on your constraint file generated after synthesis, so if you specified a wrong name, your P&*R* flow could go wrong.

Assignment

1. Try to optimize your design by specifying a tighter clock constraint (to do so, you will need re-synthesized it with the same clock constraint using the script of the previous lab). Do not forget to choose the appropriate *.sdc* and verilog gate-level netlist files when importing your design since they depends on the clock constraint.
2. Report the minimum clock period you achieved and its associated critical path, total number of cells as well as its total area (provide some screenshots of the appropriate reports).

5 Assignment and Checkpoint Summary

Write a report and answer the assignments asked during the lab, which are summarized below. Do not forget to validate the checkpoints, summarized below as well, by an assistant before the end of the lab.

Assignments

1. Which metal layers should be used for power routing and for the *Clock Tree Synthesis* (CTS) and why (it is not necessarily the default one used in this lab)?
2. Report the critical path timing before and after optimization after the place step.
3. During the optimization step, how do you think Innovus® optimized the design in order to improve the slack?
4. Report the critical path timing before and after optimization after the CTS step. Do not forget to select the appropriate *Design Stage* for both the *Timing Analysis* and *Optimization* windows since the CTS step is now done. **When optimizing the design, select both setup and hold fields in the *Optimization Type* field.**
5. If you were to have a lots of DRC errors after your P&R flow due to many metal wires being to close to each other, what could you do to address this issue?
6. Try to optimize your design by specifying a tighter clock constraint (to do so, you will need re-synthesized it with the same clock constraint using the script of the previous lab). Do not forget to choose the appropriate *.sdc* and verilog gate-level netlist files when importing your design since they depends on the clock constraint.
7. Report the minimum clock period you achieved and its associated critical path, total number of cells as well as its total area (provide some screenshots of the appropriate reports).

Checkpoints

1. Please call an assistant and show him that your floorplan looks correct.
2. Please call an assistant and show him that your final design looks correct after the verifying steps.