ece 260c. **Lab 0** Welcome & Tools Setup

Shijie Sun / ysyx_25050135

Welcome to ECE 260C. In this lab, you will get started with the tools that you will later use in this course.

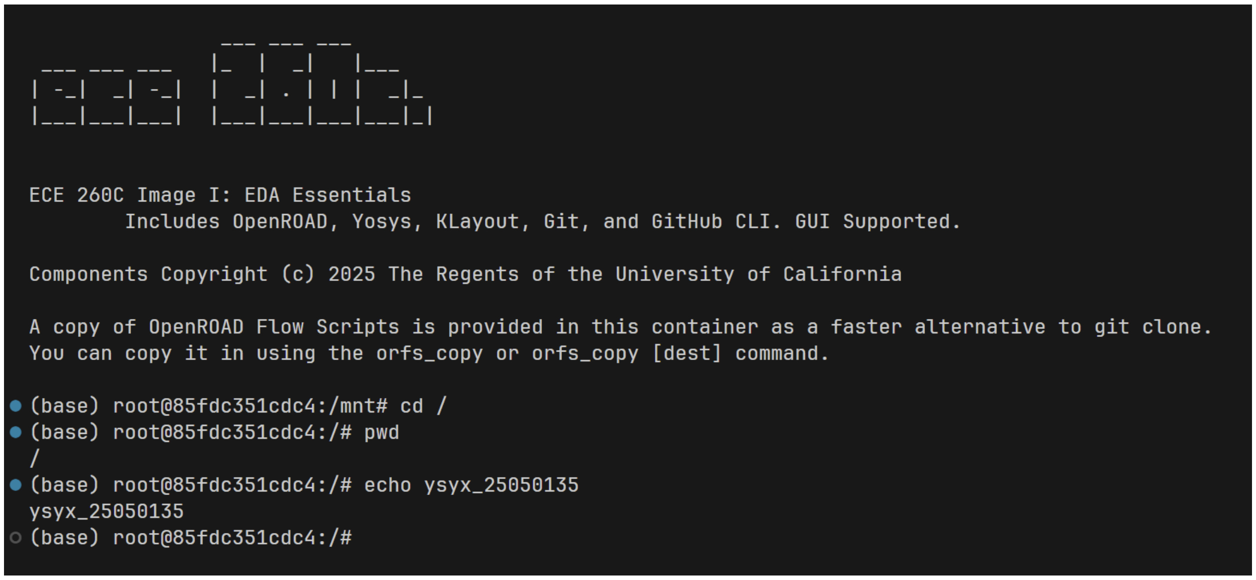
Section I Basic Setup

In this section, you will follow the Software Setup guide, linked here. from beginning to end. If you ever switch or reset machines, you should revisit this guide.

Q1.1

Get the container shell running interactively as noted in the guide. Run the following command and then paste a screenshot here:

```
echo <your PID>
```



```
                ___ ___ ___
 ___ ___ ___   |_  |  _|   |___
| -_|  _| -_|  |  _| . | | |  _|_
|___|___|___|  |___|___|___|___|_|


ECE 260C Image I: EDA Essentials
        Includes OpenROAD, Yosys, KLayout, Git, and GitHub CLI. GUI Supported.

Components Copyright (c) 2025 The Regents of the University of California

A copy of OpenROAD Flow Scripts is provided in this container as a faster alternative to git clone.
You can copy it in using the orfs_copy or orfs_copy [dest] command.

● (base) root@85fdc351cdc4:/mnt# cd /
● (base) root@85fdc351cdc4:/# pwd
/
● (base) root@85fdc351cdc4:/# echo ysyx_25050135
ysyx_25050135
○ (base) root@85fdc351cdc4:/#
```

Now, we need to load OpenROAD-flow-scripts, a comprehensive, configurable flow for OpenROAD that includes many example designs and PDKs. For this course, we will work with the IHP 130 open-source manufacturable process. First, however, we should sanity check ORFS.
For your convenience, **you should create a directory in the container's default root** called `/work`. `cd` into it and run the `orfs_copy` commands to load ORFS with a custom directory name:

```
mkdir /work
cd /work
orfs_copy orfs_lab0 # This is a convenience command provided by the
container to save time. You can also download ORFS using a git clone.
cd orfs_lab0/flow # The actual flow is a subdirectory of the ORFS
repo.
```

When we work with ORFS, we stay within the `flow/` directory so paths, unless otherwise specified, are relative to `flow/`.

Here, you can run `ls` and see the major elements of the flow:

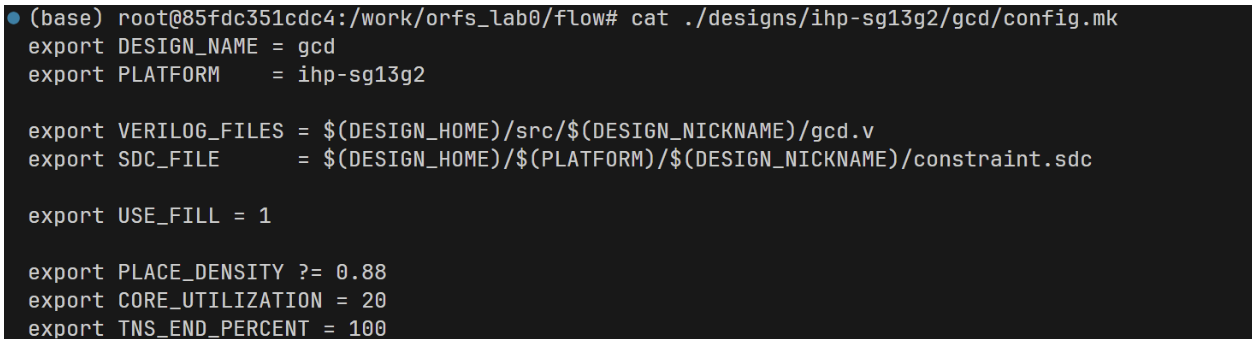
- `Makefile` – this is what you will use to invoke the flow.
- `scripts` – this contains the flow's actual behavior. These scripts contain mostly tcl commands for using OpenROAD and Yosys. They are invoked by the Makefile.
- `platforms` – this contains bundled PDKs, plus any special scripts needed to make them work, including ihp-sg13g2.
- `designs` – For each platform, this includes configurations for a set of bundled test designs that you can run ORFS on.
    - `designs/src` – the common Verilog for the bundled test designs within ORFS

Q1.2 Understanding the Makefile and the directory structure, pick a design and platform of your choice, and find the file named `config.mk`, then run

```
cat config.mk
```

Paste a screenshot here and describe the parameters you see, looking them up if need be in the ORFS Docs. Do you recognize any from previous courses?

If the config.mk contains many parameters or custom scripting, such as in `flow/designs/asap7/aes/config.mk`, describe only a few of the simpler ones.



```
● (base) root@85fdc351cdc4:/work/orfs_lab0/flow# cat ./designs/ihp-sg13g2/gcd/config.mk
  export DESIGN_NAME = gcd
  export PLATFORM    = ihp-sg13g2

  export VERILOG_FILES = $(DESIGN_HOME)/src/$(DESIGN_NICKNAME)/gcd.v
  export SDC_FILE      = $(DESIGN_HOME)/$(PLATFORM)/$(DESIGN_NICKNAME)/constraint.sdc

  export USE_FILL = 1

  export PLACE_DENSITY ?= 0.88
  export CORE_UTILIZATION = 20
  export TNS_END_PERCENT = 100
```

Variable	Description	Default
DESIGN_NAME	The name of the top-level module of the design.	
PLATFORM	Specifies process design kit or technology node to be used.	
USE_FILL	Whether to perform metal density filling.	0
PLACE_DENSITY	The desired average placement density of cells: 1.0 = dense, 0.0 = widely spread. The intended effort is also communicated by this parameter. Use a low value for faster builds and higher value for better quality of results. If a too low value is used, the placer will not be able to place all cells and a recommended minimum placement density can be found in the logs. A too high value can lead to excessive runtimes, even timeouts and subtle failures in the flow after placement, such as in CTS or global routing when timing repair fails. The default is platform specific.	
CORE_UTILIZATION	The core utilization percentage (0-100).	
TNS_END_PERCENT	Default TNS_END_PERCENT value for post CTS timing repair. Try fixing all violating endpoints by default (reduce to 5% for runtime). Specifies how many percent of violating paths to fix [0-100]. Worst path will always be fixed.	100

For your reference, you can always refer to these same files on GitHub without spinning up the shell session.

Section II Running the Flow

For the next part of our sanity check, we can try **running an actual design**.
By default, running make without any parameters will build the GCD design on the Nangate45 process. GCD is useful for testing because it's small and will thus finish implementation quickly – we will use it here.
In place of Nangate45, however, which is a non-manufacturable academic PDK, we will use IHP 130.
Run the following to perform the flow:

```
DESIGN_CONFIG=./designs/ihp-sg13g2/gcd/config.mk make
```

Depending on your machine, this can take up to 10 minutes. Please keep in mind that you need GUI working for the flow to complete without error. Please review the end of the output log carefully to ensure it's successful.
When the run is successfully completed, you can `ls` again. You'll notice 4 new directories each with nested subdirectories organized by platform, design, and a variant (defaults to "base") i.e. ./logs/`ihp-sg13g2/gcd/base`:

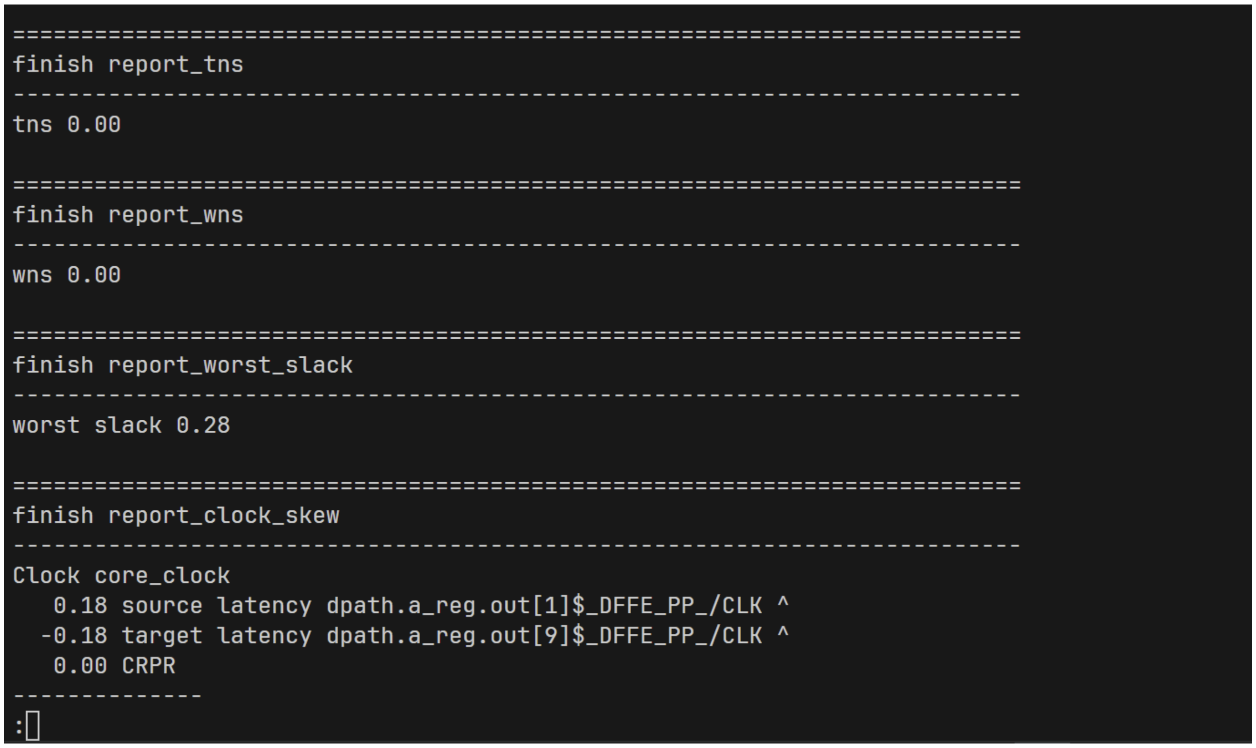
- `logs` – contains the tool logs and machine-readable metrics from OpenROAD and yosys, organized by stage of the flow
- `objects` – this typically contains PDK data copied in temporarily by the flow
- `reports` – this contains Quality-of-Results reports from different stages of the flow. Also contains rendered images
- `results` – this contains the most vital data to the flow, separated by stage:
    - synthesized netlist Verilog files
    - flow-generated tcl scripts for IO placement
    - timing/parasitics .sdc/.spef files
    - OpenDB database files that you can reload into OpenROAD
    - The all-important `6_final.def` and `6_final.gds`, the final result of the implementation.

Q2.1 Now, we can view our final report by running:

```
cat reports/ihp-sg13g2/gcd/base/6_finish.rpt | less
```

With the less command, you can just press enter or use the arrow keys to scroll around the report.

What are your WNS and TNS values? Does the design, as it is, pass timing?



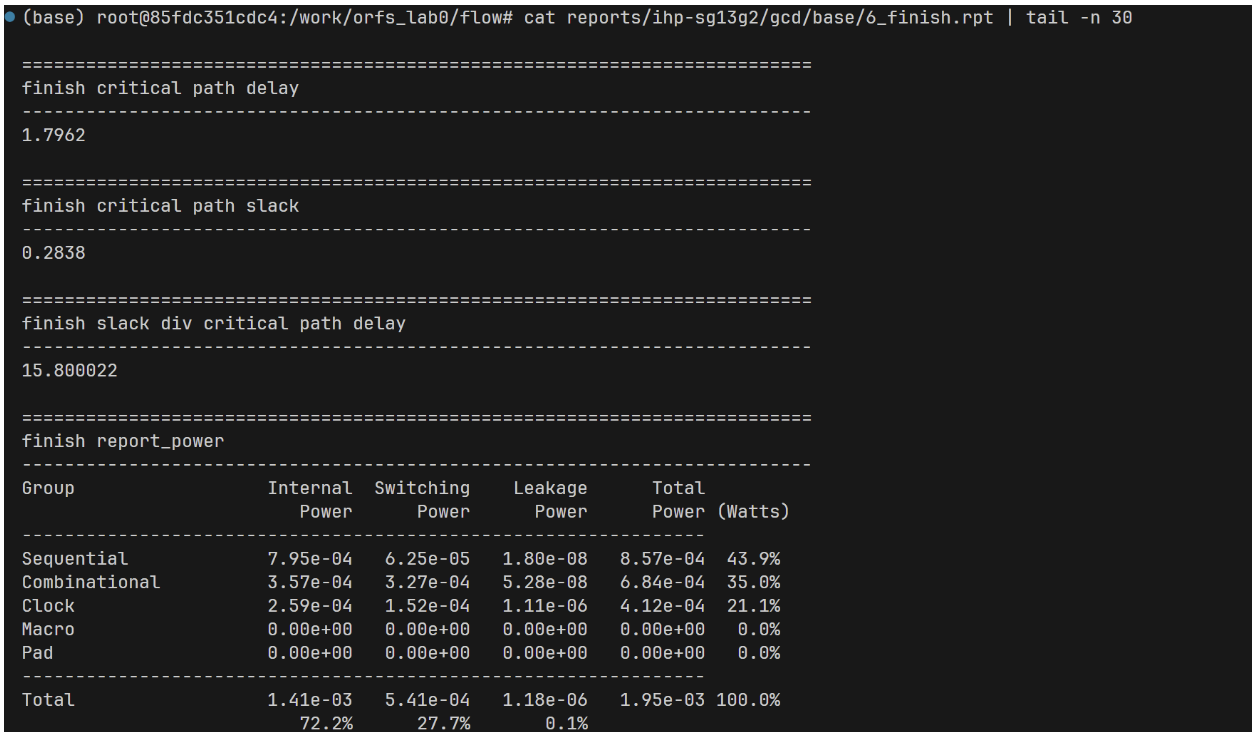
```
==========================================================================
finish report_tns
--------------------------------------------------------------------------
tns 0.00

==========================================================================
finish report_wns
--------------------------------------------------------------------------
wns 0.00

==========================================================================
finish report_worst_slack
--------------------------------------------------------------------------
worst slack 0.28

==========================================================================
finish report_clock_skew
--------------------------------------------------------------------------
Clock core_clock
   0.18 source latency dpath.a_reg.out[1]$_DFFE_PP_/CLK ^
  -0.18 target latency dpath.a_reg.out[9]$_DFFE_PP_/CLK ^
   0.00 CRPR
--------------
:
```

Pass timing.

Q2.2 Another useful command for shortening reports is tail -n <N>, which retrieves the last N lines. Run it like so:

```
cat reports/ihp-sg13g2/gcd/base/6_finish.rpt | tail -n 30
```

and paste a screenshot here.



```
(base) root@85fdc351cdc4:/work/orfs_lab0/flow# cat reports/ihp-sg13g2/gcd/base/6_finish.rpt | tail -n 30

==========================================================================
finish critical path delay
--------------------------------------------------------------------------
1.7962

==========================================================================
finish critical path slack
--------------------------------------------------------------------------
0.2838

==========================================================================
finish slack div critical path delay
--------------------------------------------------------------------------
15.800022

==========================================================================
finish report_power
--------------------------------------------------------------------------
Group                  Internal  Switching    Leakage      Total
                          Power      Power      Power      Power (Watts)
----------------------------------------------------------------
Sequential             7.95e-04   6.25e-05   1.80e-08   8.57e-04  43.9%
Combinational          3.57e-04   3.27e-04   5.28e-08   6.84e-04  35.0%
Clock                  2.59e-04   1.52e-04   1.11e-06   4.12e-04  21.1%
Macro                  0.00e+00   0.00e+00   0.00e+00   0.00e+00   0.0%
Pad                    0.00e+00   0.00e+00   0.00e+00   0.00e+00   0.0%
----------------------------------------------------------------
Total                  1.41e-03   5.41e-04   1.18e-06   1.95e-03 100.0%
                          72.2%      27.7%       0.1%
```

Before we begin to deep-dive into some of the most useful flow commands, let's clean up our environment and prepare ourselves to make edits.

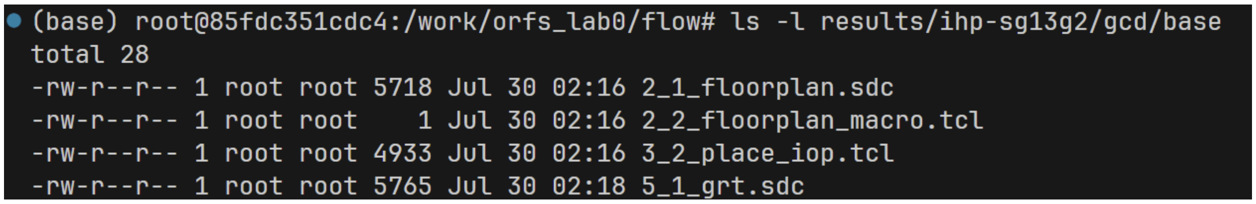
Typically, whenever we edit our design or perhaps upgrade OpenROAD, we need to perform a full clean run. To do this, we run the following with the same `DESIGN_CONFIG` as before (**please fill in the design path in place of ...**):

```
DESIGN_CONFIG=.../config.mk make clean_all
```

All the intermediate data from the aforementioned 4 folders (specifically, the subfolders containing our specific platform-design-variant combo) will be cleared out. **From this point on, remember to include the same `DESIGN_CONFIG` before all make commands otherwise ORFS will default to trying to work with `nangate45/gcd/base`.**

Now, if you run ls `logs/ihp-sg13g2/gcd/base`, you'll notice that the directory still exists but is completely empty. Similarly, in the `results` directory, only some leftover `.tcl` or `.sdc` files remain that will be cleanly overwritten the next time you run the flow.

Q2.3 Paste a screenshot here of the `results` directory.



```
(base) root@85fdc351cdc4:/work/orfs_lab0/flow# ls -l results/ihp-sg13g2/gcd/base
total 28
-rw-r--r-- 1 root root 5718 Jul 30 02:16 2_1_floorplan.sdc
-rw-r--r-- 1 root root    1 Jul 30 02:16 2_2_floorplan_macro.tcl
-rw-r--r-- 1 root root 4933 Jul 30 02:16 3_2_place_iop.tcl
-rw-r--r-- 1 root root 5765 Jul 30 02:18 5_1_grt.sdc
```

Section III Editing Flow Parameters

For this section, you'll need an editor. We highly recommend you follow the Software Setup guide to use Dev Containers in Visual Studio Code.
If you have your own preferred editor, like `vim` or `nano`, you may set it up but keep in mind you will also need an image viewer such as `sxiv` (vim, nano, and sxiv are installed in the container).
The following lab instructions will assume you're using VS Code but, when asked for screenshots, you can paste your equivalent.

You've already seen creating or editing designs centered around the `config.mk` file. Now, we'll edit the configuration for the IHP 130 GCD design we've been using. Alongside a basic set of flow environment variables you see here, more are described in the ORFS docs.

For the following exercise, **we'll modify the aspect ratio (AR) of the core**, which describes the relationship between the height and the width from 0.0 to 1.0 – i.e. an AR of 0.5 means the height of the core is half the width of the core. Typically, this AR, alongside CORE_UTILIZATION – the percentage indicating how much of the core's total area should be taken up by standard cells, are used to size a core area.

Look for the flow variable controlling Aspect Ratio in the ORFS docs linked above, then add it in the same `export <VARIABLE> = VALUE` format seen in the rest of the config.mk file.

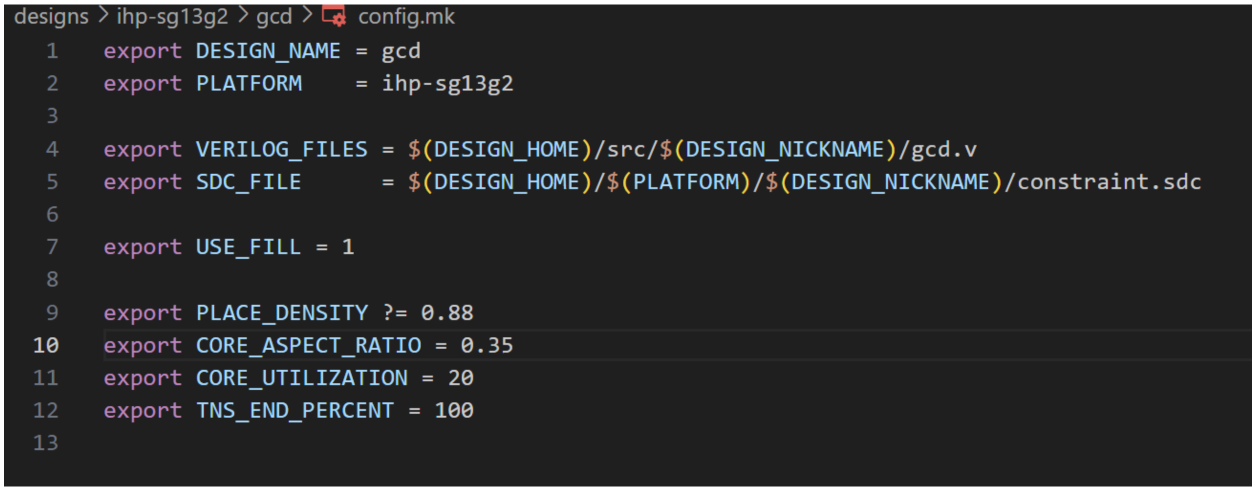
The value you use for this exercise **will actually be determined by your PID** using a simple algorithm: use the last two digits of your PID to form a decimal number. For example, if your PID is A35971234 → your AR would be 0.34.

If the number is less than 0.25, however, add 0.3 to it so if your PID is A16808614 then use the value 0.44.

Q3.1
If you found a number from your PID, write down your PID here, underlining the two digits you used here: ysyx_25050135

If a number like this cannot be made from your PID or if the later run fails, select a custom AR value and note it down here:

Q3.2 Paste a screenshot of your editor here with your modified `ihp-sg13g2/gcd/config.mk` displayed, showing your augmented AR variable.

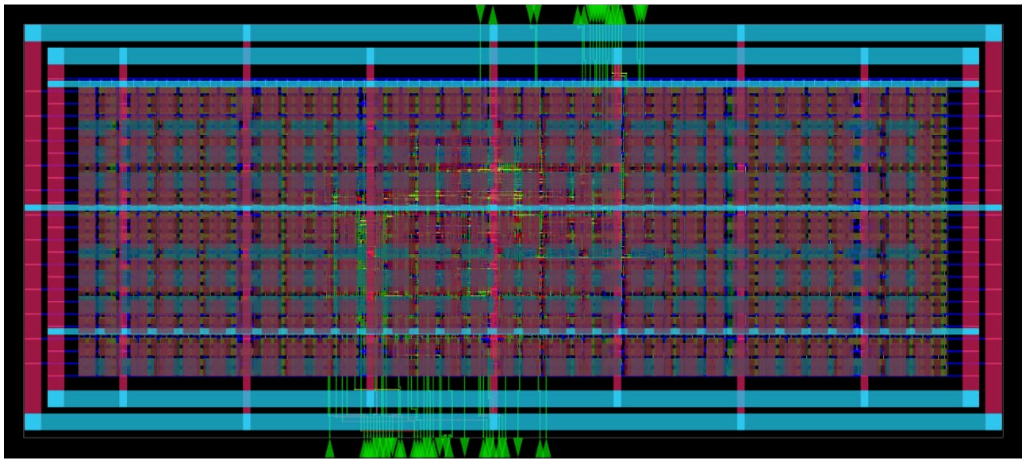


```
export DESIGN_NAME = gcd
export PLATFORM    = ihp-sg13g2

export VERILOG_FILES = $(DESIGN_HOME)/src/$(DESIGN_NICKNAME)/gcd.v
export SDC_FILE      = $(DESIGN_HOME)/$(PLATFORM)/$(DESIGN_NICKNAME)/constraint.sdc

export USE_FILL = 1

export PLACE_DENSITY ?= 0.88
export CORE_ASPECT_RATIO = 0.35
export CORE_UTILIZATION = 20
export TNS_END_PERCENT = 100
```

Now, making sure you `clean`ed, run the make command again:

```
DESIGN_CONFIG=./designs/ihp-sg13g2/gcd/config.mk make
```

Q3.3 Now, with the flow run, let's use VS Code to view the final PNG render in `reports/ihp-sg13g2/gcd/base/final_all.webp.png`
Paste a screenshot of the viewer window here.

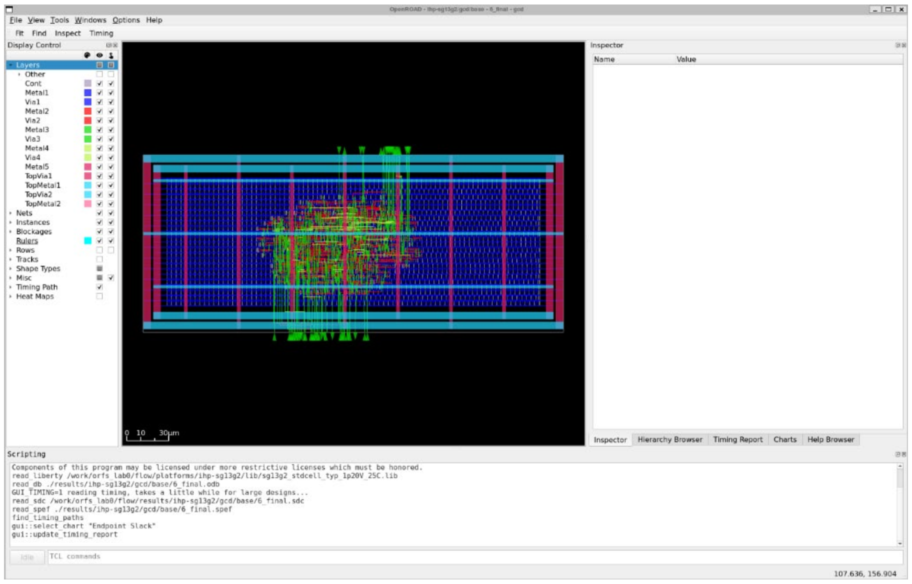


You can look at the other images rendered out to `reports` but, for more advanced interaction, we will now look at two useful commands (remember to use the correct design path in place of ...):

```
DESIGN_CONFIG=.../config.mk make open_final
DESIGN_CONFIG=.../config.mk make gui_final
```

These two commands allow you to open CLI (command-line interface) and GUI OpenROAD, respectively, with the correct DB and timing data automatically loaded. **It's also possible to load the database at different stages** by replacing the `final` suffix with floorplan/place/cts/route/grt/final.

Q3.4 Open the GUI for the final design and paste a screenshot here.



```
Components of this program may be licensed under more restrictive licenses which must be honored.
read_liberty /work/orfs_lab0/flow/platforms/ihp-sg13g2/lib/sg13g2_stdcell_typ_1p20V_25C.lib
read_db ./results/ihp-sg13g2/gcd/base/6_final.odb
GUI_TIMING=1 reading timing, takes a little while for large designs...
read_sdc /work/orfs_lab0/flow/results/ihp-sg13g2/gcd/base/6_final.sdc
read_spef ./results/ihp-sg13g2/gcd/base/6_final.spef
find_timing_paths
gui::select_chart "Endpoint Slack"
gui::update_timing_report
```

Sometimes, you may need to resize the OpenROAD window to make it render correctly.

With the GUI open, you have many options. In the main design viewer, you can left-click on cells/macros, nets, tracks, and ports. You can also right-click-drag to zoom in. **You may need to do this for your design depending on Aspect Ratio**. You can hit F to zoom back out and fit the whole chip on the screen.

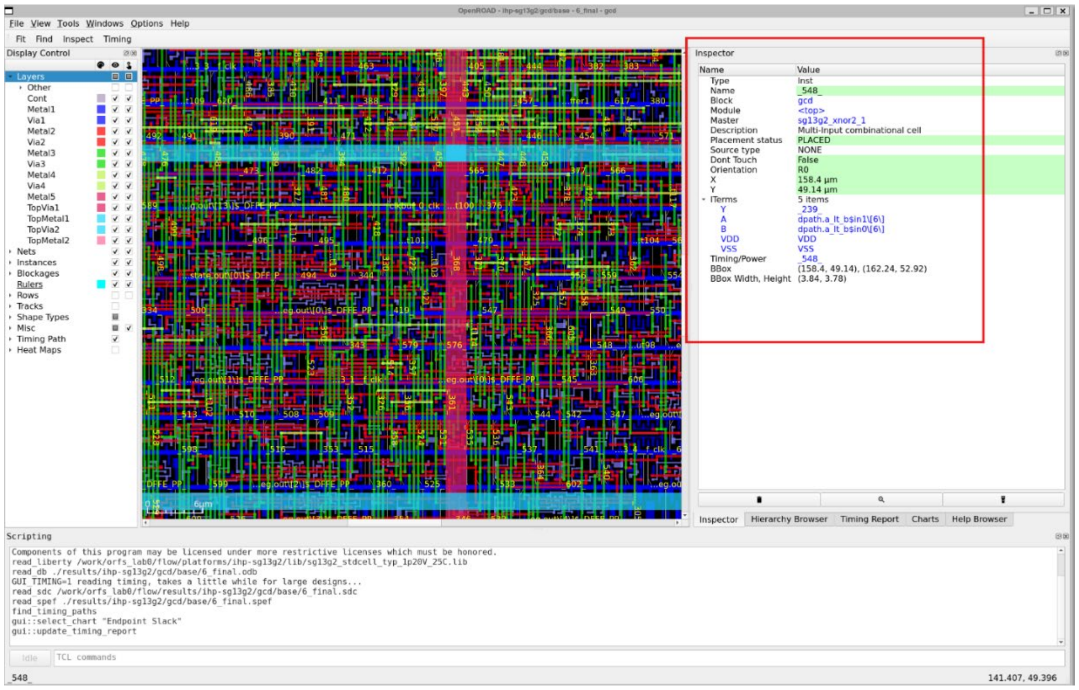
Q3.5 <u>What do you notice about the pin placements? Are they done by an engineer or by the tool? Is there any method to their placement?</u>

Click on a few cells, generically called Instances, to see their database entries on the right side in the Inspector. If you click on a wire net, you can see all the connected cells listed.

Here are some key cell entries visible in this view :

- Block/module are used in hierarchical design, which we will discuss later.
- Master is the specific cell in use. For this PDK, all masters are prefixed with `sg13g2_`. You can click on this entry to get details about the master itself.
- ITerms are the actual connections to the cell, power and clock included where applicable.
  Hover over or click on these to see the nets they are connected to.
- Placement Status/Don't Touch are used by the actual OpenROAD application components to mark what should be modified. For example, setting Placement Status to FIRM before running Global Placement will keep the placer from moving that cell around. We will take advantage of this in a future lab.
- Clicking on Timing/Power allows you to access a report including arrival times, power activities, and total power, as well as references to the Liberty entry/library.
- Source for some D Flip Flops will show the register in Verilog source that the FF backs.

Q3.6 <u>Select a random cell and paste a screenshot here of the Inspector.</u>



Inspector

Name	Value
Type	Inst
Name	_548_
Block	gcd
Module	<top>
Master	sg13g2_xnor2_1
Description	Multi-Input combinational cell
Placement status	PLACED
Source type	NONE
Dont Touch	False
Orientation	R0
X	158.4 µm
Y	49.14 µm
ITerms	5 items
Y	_239_
A	dpath.a_lt_b$in1\[6\]
B	dpath.a_lt_b$in0\[6\]
VDD	VDD
VSS	VSS
Timing/Power	_548_
BBox	(158.4, 49.14), (162.24, 52.92)
BBox Width, Height	(3.84, 3.78)

```
Components of this program may be licensed under more restrictive licenses which must be honored.
read_liberty /work/orfs_lab0/flow/platforms/ihp-sg13g2/lib/sg13g2_stdcell_typ_1p20V_25C.lib
read_db ./results/ihp-sg13g2/gcd/base/6_final.odb
GUI_TIMING=1 reading timing, takes a little while for large designs...
read_sdc /work/orfs_lab0/flow/results/ihp-sg13g2/gcd/base/6_final.sdc
read_spef ./results/ihp-sg13g2/gcd/base/6_final.spef
find_timing_paths
gui::select_chart "Endpoint Slack"
gui::update_timing_report
```

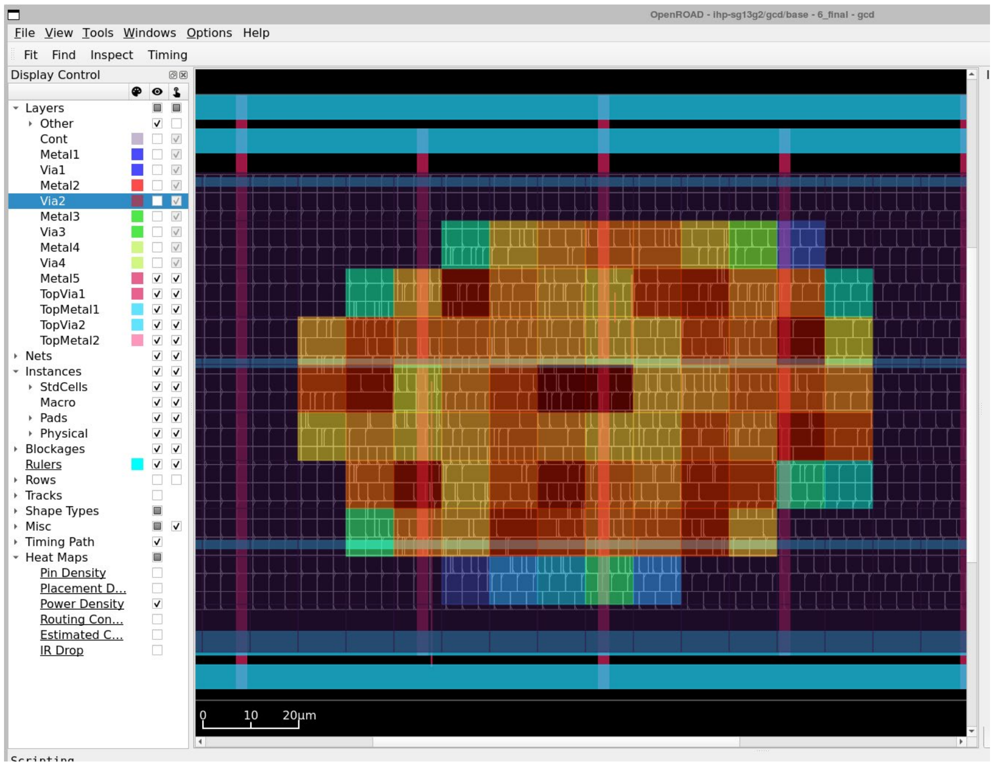
_548_ 141.407, 49.396

The Display Control area on the left-side. Allows you to hide or show different layers and views. You can use this to get clearer insight into how the design was placed and routed. For example, you can hide Layers to see just where Instances were placed or you can hide the top metal layers to see the design without the large power delivery network obscuring cells.
You can also control whether a layer is clickable (the rightmost column of checkboxes). When a layer is deselected here (for example, the Metal5 vertical power wire), you can click on the elements under it.

Another useful feature are heatmaps, which provide key spatial statistics. Open the Heatmaps section and select one at a time (except IR Drop, which may not be populated). If a heatmap is ever too granular, too coarse, or including elements you do not want, you can go to Tools > Heatmaps and change the grid size or other options. For our purposes, the default 10x10 heatmap is sufficient.

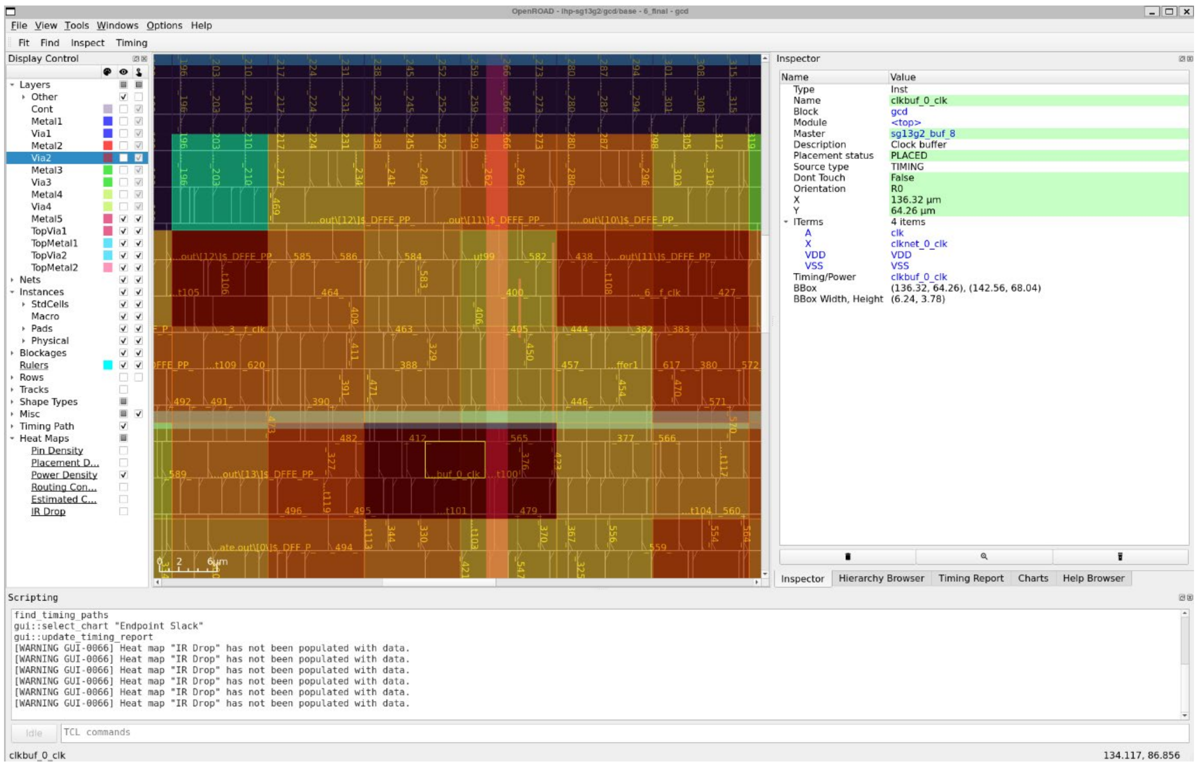
Q3.7 <u>Select a random heatmap (except Placement Density or IR Drop) and paste a screenshot here. Discuss what it suggests about the design.</u>

Power Density



The overall power density distribution is relatively uniform, with no significant hotspots observed. The center of the core region exhibits the highest power density, highlighted in deep red, indicating a concentration of active cells in that area. In contrast, the edges of the core show noticeably lower power density with lighter colors, which aligns with the expected trend of gradually decreasing activity toward the periphery.

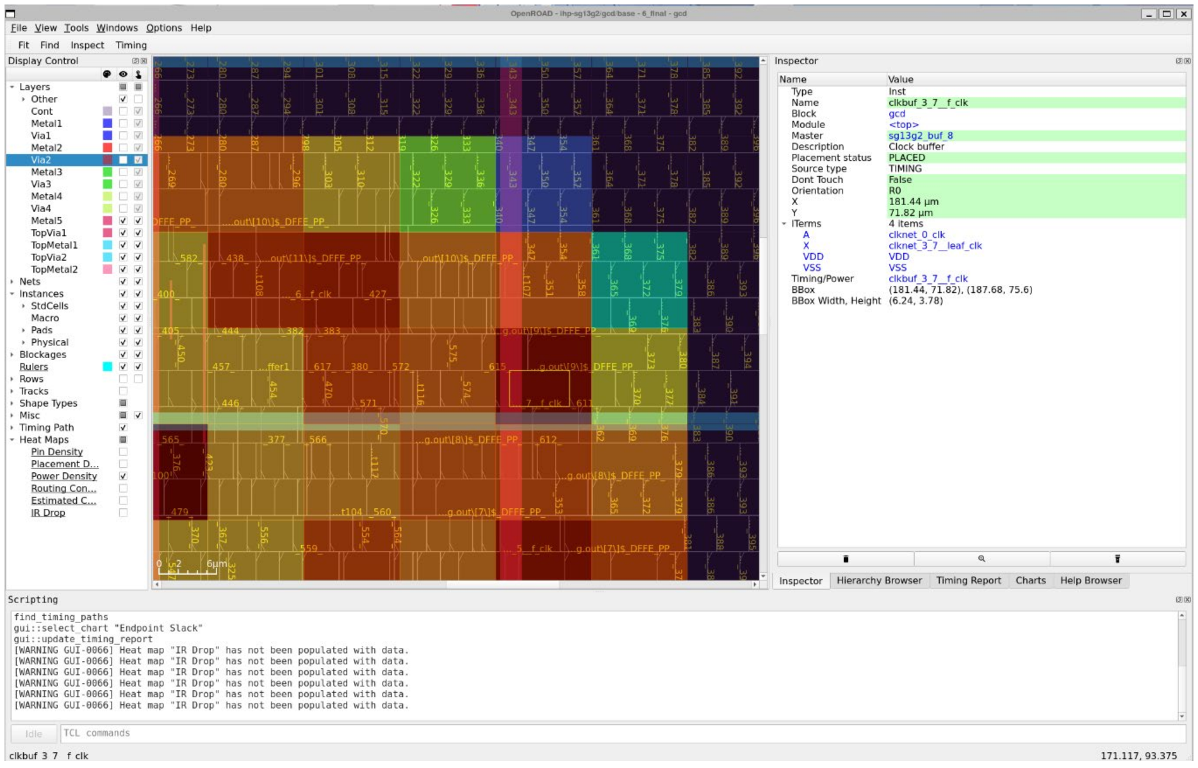
Q3.8 Compare a “hot” area (i.e. a dark red grid square) to surrounding cold(er) ones – <u>what differences can you see between their interior instances and/or nets?</u> Remember, you can toggle Layers in Display Control to more easily select instances or certain nets.



Name	Value
Type	Inst
Name	clkbuf_0_clk
Block	gcd
Module	<top>
Master	sg13g2_buf_8
Description	Clock buffer
Placement status	PLACED
Source type	TIMING
Dont Touch	False
Orientation	R0
X	136.32 µm
Y	64.26 µm
ITerms	4 items
A	clk
X	clknet_0_clk
VDD	VDD
VSS	VSS
Timing/Power	clkbuf_0_clk
BBox	(136.32, 64.26), (142.56, 68.04)
BBox Width, Height	(6.24, 3.78)

```
find_timing_paths
gui::select_chart "Endpoint Slack"
gui::update_timing_report
[WARNING GUI-0066] Heat map "IR Drop" has not been populated with data.
[WARNING GUI-0066] Heat map "IR Drop" has not been populated with data.
[WARNING GUI-0066] Heat map "IR Drop" has not been populated with data.
[WARNING GUI-0066] Heat map "IR Drop" has not been populated with data.
[WARNING GUI-0066] Heat map "IR Drop" has not been populated with data.
[WARNING GUI-0066] Heat map "IR Drop" has not been populated with data.
```

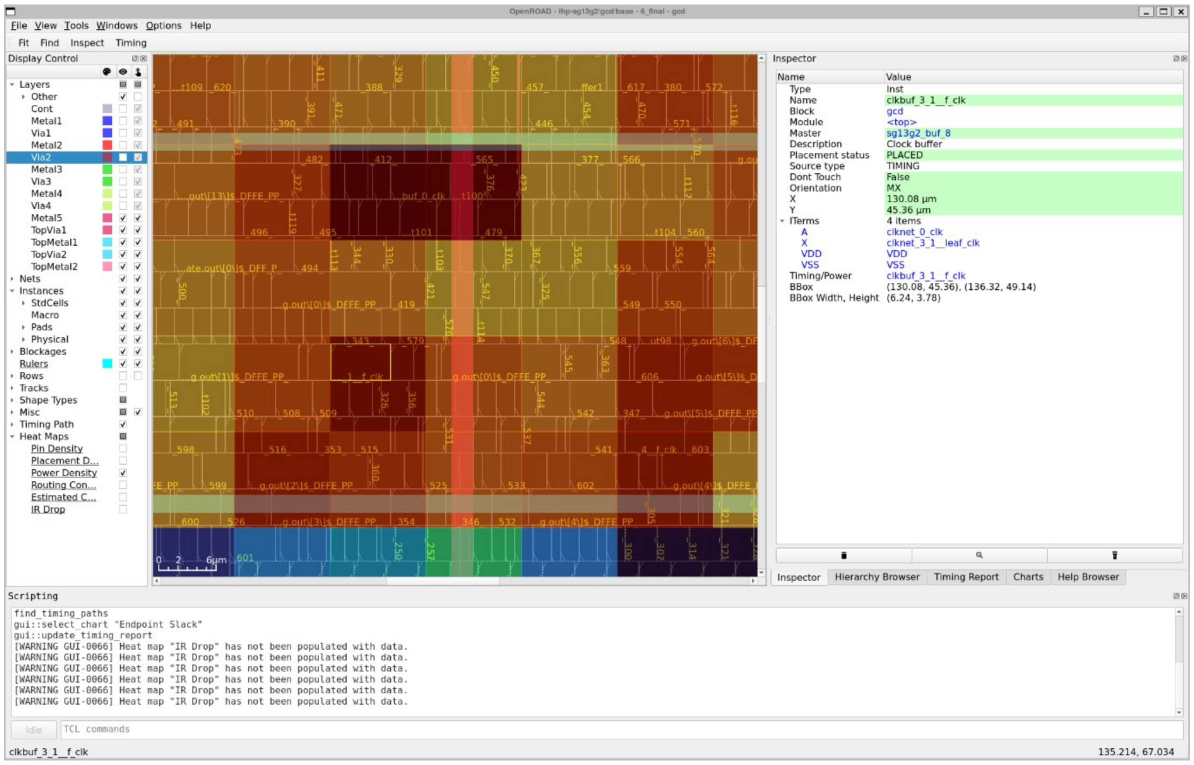
clkbuf_0_clk 134.117, 86.856



Name	Value
Type	Inst
Name	clkbuf_3_7__f_clk
Block	gcd
Module	<top>
Master	sg13g2_buf_8
Description	Clock buffer
Placement status	PLACED
Source type	TIMING
Dont Touch	False
Orientation	R0
X	181.44 µm
Y	71.82 µm
ITerms	4 items
A	clknet_0_clk
X	clknet_3_7__leaf_clk
VDD	VDD
VSS	VSS
Timing/Power	clkbuf_3_7__f_clk
BBox	(181.44, 71.82), (187.68, 75.6)
BBox Width, Height	(6.24, 3.78)

```
find_timing_paths
gui::select_chart "Endpoint Slack"
gui::update_timing_report
[WARNING GUI-0066] Heat map "IR Drop" has not been populated with data.
[WARNING GUI-0066] Heat map "IR Drop" has not been populated with data.
[WARNING GUI-0066] Heat map "IR Drop" has not been populated with data.
[WARNING GUI-0066] Heat map "IR Drop" has not been populated with data.
[WARNING GUI-0066] Heat map "IR Drop" has not been populated with data.
[WARNING GUI-0066] Heat map "IR Drop" has not been populated with data.
```

clkbuf_3_7__f_clk 171.117, 93.375


OpenROAD - ihp-sg13g2/gcd/base - 6_final - gcd
File View Tools Windows Options Help
Fit Find Inspect Timing
Display Control
Inspector
Inspector Hierarchy Browser Timing Report Charts Help Browser
135.214, 67.034

Name	Value
Type	Inst
Name	clkbuf_3_1__f_clk
Block	gcd
Module	<top>
Master	sg13g2_buf_8
Description	Clock buffer
Placement status	PLACED
Source type	TIMING
Dont Touch	False
Orientation	MX
X	130.08 µm
Y	45.36 µm
ITerms	4 items
A	clknet_0_clk
X	clknet_3_1__leaf_clk
VDD	VDD
VSS	VSS
Timing/Power	clkbuf_3_1__f_clk
BBox	(130.08, 45.36), (136.32, 49.14)
BBox Width, Height	(6.24, 3.78)

Scripting

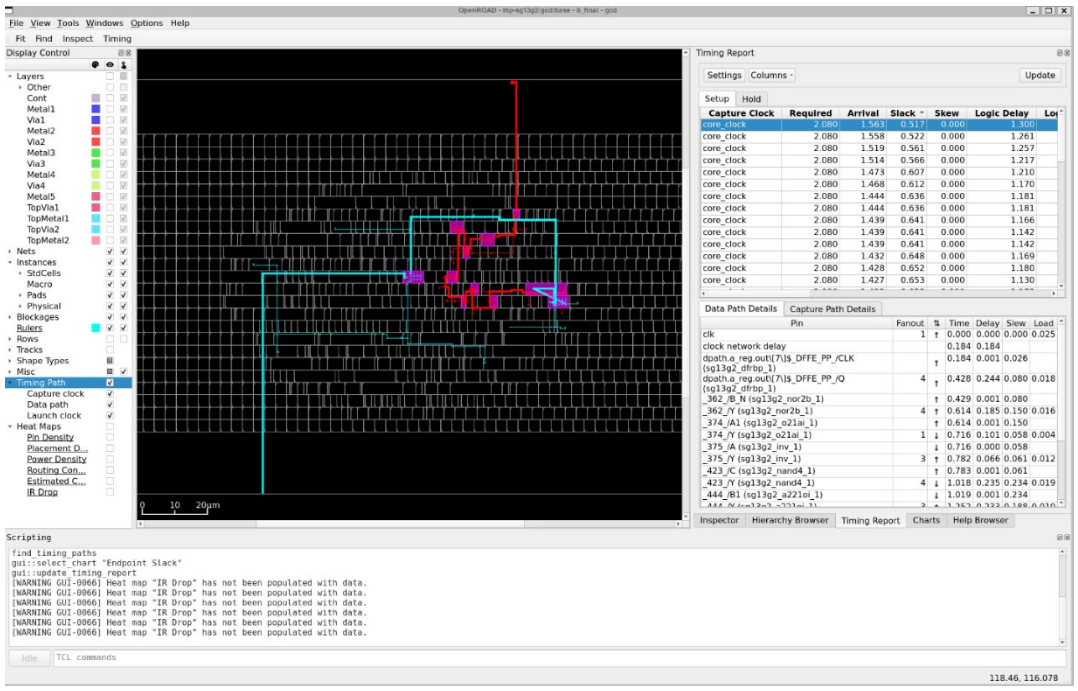
```
find_timing_paths
gui::select_chart "Endpoint Slack"
gui::update_timing_report
[WARNING GUI-0066] Heat map "IR Drop" has not been populated with data.
[WARNING GUI-0066] Heat map "IR Drop" has not been populated with data.
[WARNING GUI-0066] Heat map "IR Drop" has not been populated with data.
[WARNING GUI-0066] Heat map "IR Drop" has not been populated with data.
[WARNING GUI-0066] Heat map "IR Drop" has not been populated with data.
[WARNING GUI-0066] Heat map "IR Drop" has not been populated with data.
```

clkbuf_3_1__f_clk

Clock buffer (clkbuf) modules are found in multiple high power density hotspots, which is likely because these buffers need to drive a large number of flip-flops and other clock loads. This results in relatively high dynamic power consumption of the clkbuf itself, causing localized power concentration in these regions.

Return Display Control to how it was before (all Layers shown, heatmaps off, Timing Paths on).
Now, go to the Timing Report tab on the right panel of the GUI (where Inspector is). Here, you can see paths with the lowest slack (or highest WNS) for both Setup and Hold.
In the Setup section, Click on the Slack column header to sort from lowest to highest and click on the top entry. Assuming you're zoomed in enough, you should be able to see the clock path (including buffers) highlighted in blue and datapath highlighted in red (this can be hard to see in the default color scheme without hiding layers).

Q3.9 Without layers hidden, paste a screenshot of the app with the timing path visible and the Timing Report visible. Based on the slack, does the design meet timing at these constraints?



Capture Clock	Required	Arrival	Slack	Skew	Logic Delay
core_clock	2.080	1.563	0.517	0.000	1.300
core_clock	2.080	1.558	0.522	0.000	1.261
core_clock	2.080	1.519	0.561	0.000	1.257
core_clock	2.080	1.514	0.566	0.000	1.217
core_clock	2.080	1.473	0.607	0.000	1.210
core_clock	2.080	1.468	0.612	0.000	1.170
core_clock	2.080	1.444	0.636	0.000	1.181
core_clock	2.080	1.444	0.636	0.000	1.181
core_clock	2.080	1.439	0.641	0.000	1.166
core_clock	2.080	1.439	0.641	0.000	1.142
core_clock	2.080	1.439	0.641	0.000	1.142
core_clock	2.080	1.432	0.648	0.000	1.169
core_clock	2.080	1.428	0.652	0.000	1.180
core_clock	2.080	1.427	0.653	0.000	1.130

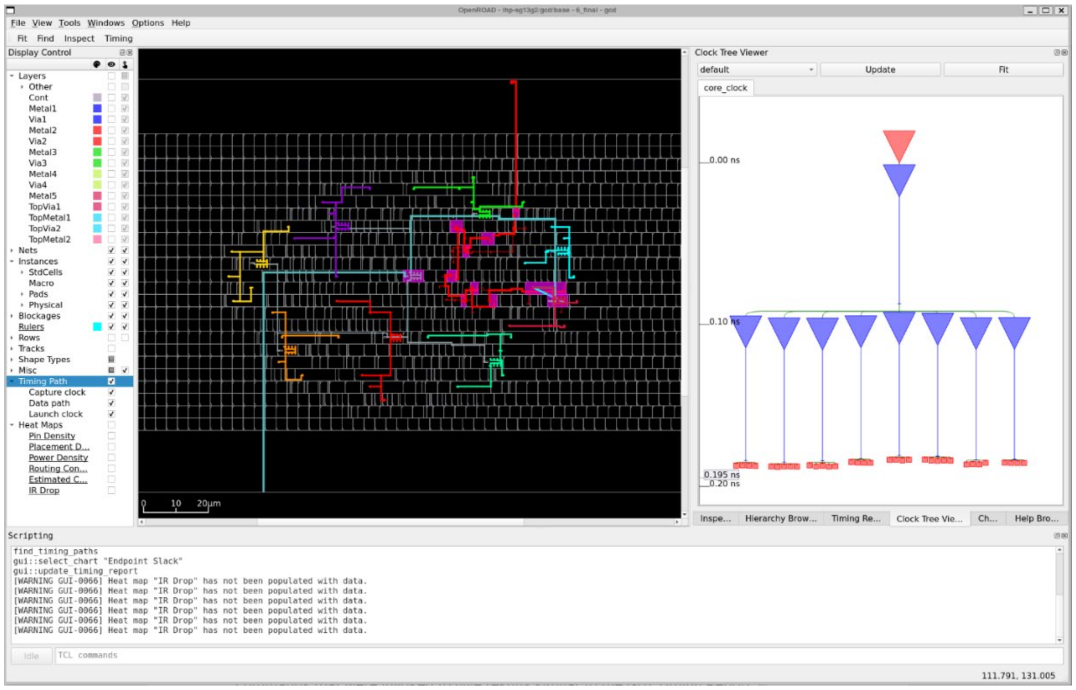
Pin	Fanout	⇅	Time	Delay	Slew	Load
clk	1	↑	0.000	0.000	0.000	0.025
clock network delay			0.184	0.184		
dpath.a_reg.out\[7\]$_DFFE_PP_/CLK (sg13g2_dfrbp_1)		↑	0.184	0.001	0.026	
dpath.a_reg.out\[7\]$_DFFE_PP_/Q (sg13g2_dfrbp_1)	4	↑	0.428	0.244	0.080	0.018
_362_/B_N (sg13g2_nor2b_1)		↑	0.429	0.001	0.080	
_362_/Y (sg13g2_nor2b_1)	4	↑	0.614	0.185	0.150	0.016
_374_/A1 (sg13g2_o21ai_1)		↑	0.614	0.001	0.150	
_374_/Y (sg13g2_o21ai_1)	1	↓	0.716	0.101	0.058	0.004
_375_/A (sg13g2_inv_1)		↓	0.716	0.000	0.058	
_375_/Y (sg13g2_inv_1)	3	↑	0.782	0.066	0.061	0.012
_423_/C (sg13g2_nand4_1)		↑	0.783	0.001	0.061	
_423_/Y (sg13g2_nand4_1)	4	↓	1.018	0.235	0.234	0.019
_444_/B1 (sg13g2_a221oi_1)		↓	1.019	0.001	0.234	

```
find_timing_paths
gui::select_chart "Endpoint Slack"
gui::update_timing_report
[WARNING GUI-0066] Heat map "IR Drop" has not been populated with data.
[WARNING GUI-0066] Heat map "IR Drop" has not been populated with data.
[WARNING GUI-0066] Heat map "IR Drop" has not been populated with data.
[WARNING GUI-0066] Heat map "IR Drop" has not been populated with data.
[WARNING GUI-0066] Heat map "IR Drop" has not been populated with data.
[WARNING GUI-0066] Heat map "IR Drop" has not been populated with data.
```

Slack ⩾ 0, the design meet timing at these constraints.

While you have OpenROAD open, you can also check out the following (no need to screenshot):

- The Charts tab in the same right panel as Inspector and Timing Report
- The Find tool
- The Clock Tree Viewer – enable this in the menubar under Windows > Clock Tree Viewer. From there, you can select the Clock Tree Viewer tab in the same right panel. Select "default" and click Update.
  This screen shows you the spread of the clock tree as it enters from a port (the red triangle), is buffered (the blue triangles), and is distributed to sets of FFs (highlighted in red at the bottom at the chart). You can click each of these entries to see them in the design and in Inspector.



```
find_timing_paths
gui::select_chart "Endpoint Slack"
gui::update_timing_report
[WARNING GUI-0066] Heat map "IR Drop" has not been populated with data.
[WARNING GUI-0066] Heat map "IR Drop" has not been populated with data.
[WARNING GUI-0066] Heat map "IR Drop" has not been populated with data.
[WARNING GUI-0066] Heat map "IR Drop" has not been populated with data.
[WARNING GUI-0066] Heat map "IR Drop" has not been populated with data.
[WARNING GUI-0066] Heat map "IR Drop" has not been populated with data.
```

You can now close out OpenROAD. If prompted whether you want to Hide GUI or Exit, select Exit.

Keep in mind that many of the features you see here are available in the CLI.
If you look at `6_finish.rpt` again (as you saw in **Q2.2**), you can see the timing commands that were invoked to give results similar to the GUI Timing Report.
Similarly, ORFS exports several views, including the IR drop heatmap, as png files in the same directory, as you saw in **Q3.2**.
As we continue this course, you will be able to write your own scripts that take advantage of these same features but often you will also be able to work on top of ORFS.

Section IV Submitting Your Design

Congratulations on running your customized design! In this final section, you will prepare some files for submission to GitHub Classroom. You'll use GitHub Classroom for your later labs and for your final project so this is a chance to figure it out.

First, complete the login procedure as noted in the *Continued Usage – Saving your Files with Git* section in the Software Setup guide. Then, `cd` back into your `/work` directory (or, if you didn't create one, your root directory / or home directory ~)

Now go to the assignment:
[Click here to unlock the assignment in GitHub Classroom.]()

Follow the instructions to join the class and create your submission repo. **Please be attentive when selecting your name.**

When your repo is created, you can clone it by running:

```
gh repo clone ABKCourses/e260c-lab0-YourUsername
```

If this doesn't work, make sure you're using the same account you logged in with.
If you used the wrong account, misselected your name, or could not find your name during the Class Join process, email a TA as soon as you can.

Now, you should see a repo that's empty except for maybe a README.md. You can use the cp command to copy files from ORFS like so (always verify with ls after):

```
cp /work/orfs_lab0/flow/...  . # using relative paths, copying into
your current dir
cp /work/orfs_lab0/flow/...  /work/e260c-lab0-YourUsername # using
absolute paths
```

Please copy the following files from the flow subdirectories into the root of the repo (you should have no directories in your repo):

- `designs/ihp-sg13g2/gcd/config.mk`
- `reports/ihp-sg13g2/gcd/base/6_finish.rpt`
- `reports/ihp-sg13g2/gcd/base/cts_core_clock_layout.webp.png`
- `logs/ihp-sg13g2/gcd/base/6_report.json`
- `results/ihp-sg13g2/gcd/base/6_final.gds`

Now, you need to stage, commit, and push these additions. Using VS Code, you can go to the Source Control tool on the left sidebar, hover over Changes, and click the plus button to stage all the files you just added at once.

Then, type in a commit message such as “Submitting my assignment” and hit Commit. Finally, hit Sync Changes.

If you want to use command-line Git, you can run the following within the repo directory:

```
git add .
git commit -m "Submitting my assignment"
git push
```

Tip: When you work on multiple machines or with teammates, ensure you’re communicating and syncing changes regularly to avoid merge conflicts when editing the same files.

When you’ve successfully pushed the changes from VS Code or the shell, refresh the GitHub repo webpage and verify that your commit, with all the required files, is present.

Take this lab document and export it as PDF.
<end>