

Lab 0

Welcome & Tools Setup

Davit Markarian

Lab Logistics

- Labs release after the class ends
 - Check the Schedule to see their release dates
- Labs can be found on [Website > Labs](#)
 - Deadlines found in the schedule
- Labs are due at 11:59 PM on their listed due date
 - Lab 0 is due on April 7th
 - 3 AM late due time (on April 8th, no penalty)
- Start labs early
- Please review the Syllabus collaboration and GenAI policies

Lab 0 Rundown

- [Software Setup Guide](#)
 - ChipsHub
 - Git/GitHub
 - (Later, for miniprojects) Docker
- [OpenROAD-flow-scripts](#)
 - Running OpenROAD for the first time
 - Making changes to design parameters
 - Using the GUI to make analyses
- Submission with Google Docs/[Gradescope](#) and GitHub Classroom
 - Get the Gradescope invitation via Canvas or via the [Syllabus](#)

Realizing the Flow

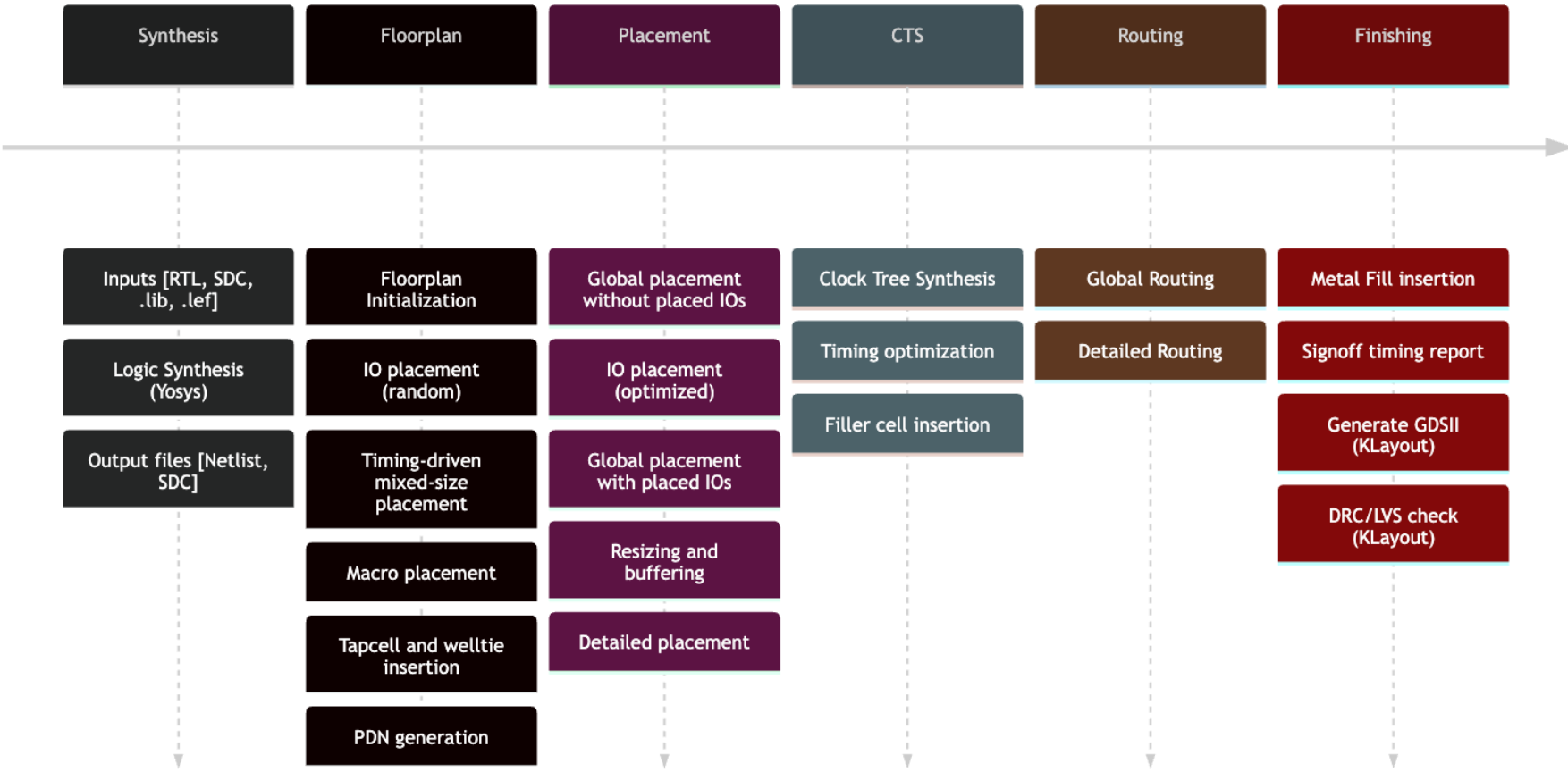
- OpenROAD, like the commercial tools, is inert without scripting → we must write tcl or Python scripts to realize implementation
- In 260B, you achieved this by guided writing or copying of scripts
 - then, running them stage-by-stage.
- This is an error-prone process
 - Stages must be executed in the correct order
 - Tools are large and complex: many configurable variables with scattered documentation
- Need for a starting-point “reference flow”

OpenROAD-flow-scripts

- ORFS is a batteries-included flow for OpenROAD
 - It includes highly configurable scripts that you can use to complete implementation
 - Driven by [Makefiles](#) with [well-documented variables](#).
- It also includes test designs and [PDKs](#)
 - Great for testing, exploration, and research
 - For this course, we will use the included open IHP 130nm SG12G2 process
- Includes scripting [hooks](#) between flow stages
 - You can your custom implementation behavior on top of ORFS

OpenROAD-flow-scripts

RTL-GDSII Using OpenROAD-flow-scripts



config.mk

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

export VERILOG_FILES = spi.v
export SDC_FILE       = constraint.sdc

export USE_FILL = 1

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

- [config.mk](#) is what control an implementation in ORFS
 - Just set [flow variables](#)
- ORFS is driven by [flow/Makefile](#), which pulls your design's config.mk
- Implementing a design is as easy as:
DESIGN_CONFIG=path/to/config.mk
make

Outputs

- ORFS will output what you expect from a flow:
 - GDS, DEF, SPEF, Verilog
 - for every stage
- ORFS will also output reports
 - QoR
 - + stage-by-stage logs/renders
 - + machine-readable reports (JSON)
- You can open interactive sessions for any stage
 - `make open_final /`
 - `make gui_final`

```
{  
  ...  
  "finish__design__instance__count": 2021,  
  "finish__design__instance__area": 23939.2,  
  "finish__timing__setup__tns": 0,  
  "finish__timing__hold__tns": 0,  
  "finish__timing__setup__ws": 0.0957634,  
  "finish__timing__hold__ws": 0.369189,  
  "finish__clock__skew__setup": 0.0141816,  
  "finish__clock__skew__hold": 0.0141816,  
  "finish__power__internal__total": 0.00121364,  
  "finish__power__switching__total": 0.000514615,  
  "finish__power__leakage__total": 1.17737e-06,  
  "finish__power__total": 0.00172943,  
  "finish__design__io": 54,  
  "finish__design__die__area": 39917.8,  
  "finish__design__core__area": 23939.2,  
  ...  
}
```

OpenROAD GUI

The screenshot displays the OpenROAD GUI interface. The central window shows a detailed routing layout with various layers and components. On the left, a 'Display Control' panel lists layers such as Metal1 through Metal5, TopVia1 through TopVia2, and TopMetal1 through TopMetal2, each with a corresponding color and a checked box. Below this, there are sections for Nets, Instances, Blockages, Rulers, Tracks, Shape Types, Misc, and Timing Path. A scale bar at the bottom left of the layout indicates 0 to 20 micrometers.

On the right side, a 'Timing Report' window is open, showing a table of timing data. The table has columns for Capture Clock, Required, Arrival, Slack, Skew, Logic Delay, and Load. The data is as follows:

Capture Clock	Required	Arrival	Slack	Skew	Logic Delay	Load
core_clock	2.080	2.122	-0.042	0.000	1.660	
core_clock	2.080	2.109	-0.029	0.000	1.649	
core_clock	2.080	2.102	-0.022	0.000	1.639	
core_clock	2.080	2.093	-0.013	0.000	1.438	
core_clock	2.080	2.088	-0.008	0.000	1.629	
core_clock	2.080	2.088	-0.008	0.000	1.624	
core_clock	2.080	2.087	-0.007	0.000	1.622	
core_clock	2.080	2.085	-0.005	0.000	1.621	
core_clock	2.080	2.084	-0.004	0.000	1.603	
core_clock	2.080	2.082	-0.002	0.000	1.617	
core_clock	2.080	2.070	0.010	0.000	1.593	

Below the timing report, there is a 'Data Path Details' section with a table showing pin details:

Pin	Fanout	Time	Delay	Slew	Load
clk	1	0.000	0.000	0.000	0.020
clock network delay		0.387	0.387		
dpath.b_reg.out{2}\$_DFFE_PP_CLK (sg13g2_dfrbp_1)		0.387	0.001	0.102	
dpath.b_reg.out{2}\$_DFFE_PP_Q (sg13g2_dfrbp_1)	7	0.701	0.314	0.129	0.030
353/A (sg13g2_xor2_1)		0.703	0.002	0.129	
353/X (sg13g2_xor2_1)	2	0.826	0.123	0.083	0.010
355/A (sg13g2_nor2_1)		0.826	0.000	0.083	
355/Y (sg13g2_nor2_1)	2	0.949	0.123	0.114	0.011
356/D (sg13g2_nand4_1)		0.950	0.001	0.114	
356/Y (sg13g2_nand4_1)	3	1.169	0.219	0.187	0.014

At the bottom of the GUI, a 'Scripting' window shows the following commands:

```
read spef ./results/ihp-sg13g2/gcd/base/6_final.spef
Populating timing paths...OK
gui::update_timing_report
```

The bottom right corner of the GUI shows the coordinates -2.48, 94.26.

Git & GitHub

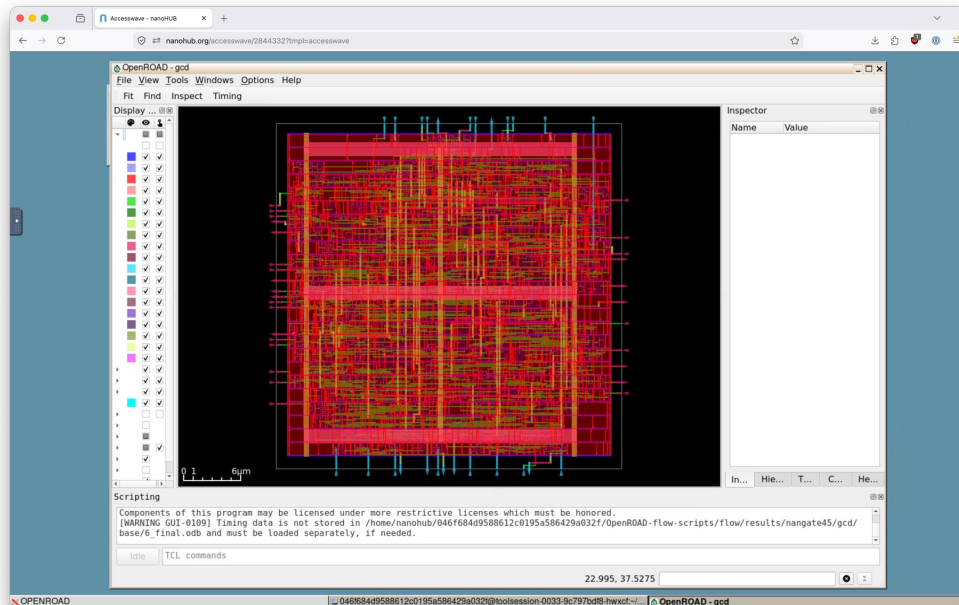
- GitHub Classroom lets you submit Git repos containing your work for this course
 - You will get invitation links to each lab submission within the lab report template
 - Some labs will have starter code too
 - For the first time, you must associate your GitHub account to your name on the class roster. Please do this carefully.
- The Git and GitHub command line tools are part of our container

Completion & Submission

- Go to the website and follow the instructions to make a copy of the lab report template on Google Docs
- You will follow the lab instructions to
 - Perform the full software setup as noted in the Guide.
 - Run the tool and experiment with ORFS and the OpenROAD GUI
 - Join the GitHub Classroom assignment, clone the repo to your container, and push your results into it
 - Export your report to PDF and upload it to Gradescope

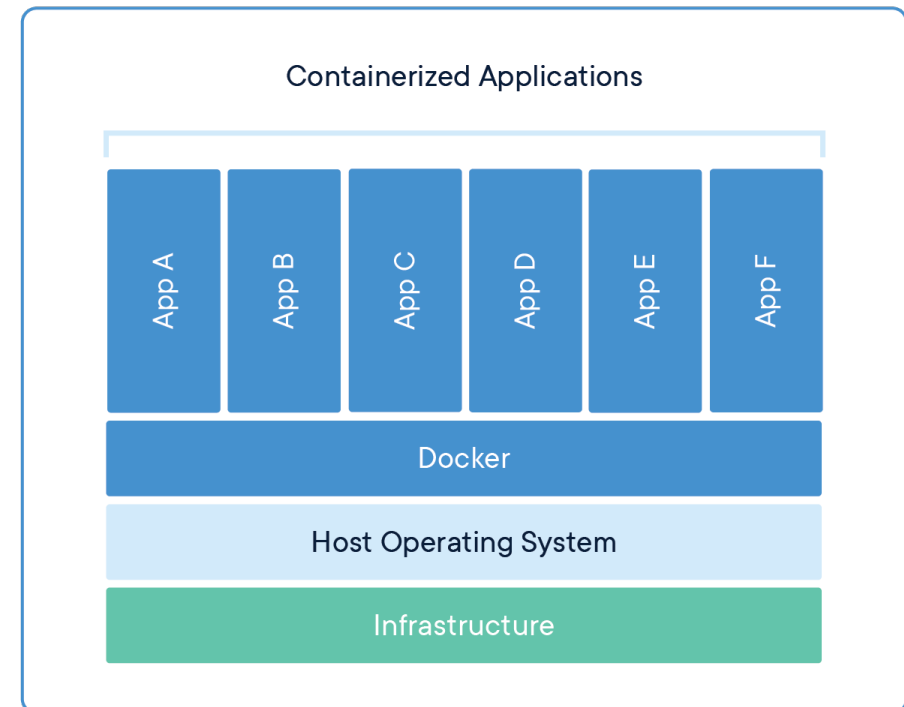
Chipshub

- A new online chip design hub lead by Purdue University as an NSF project
 - Provides web-based remote access to a Linux environment with OpenROAD
- For Lab 0, you will follow the Software Setup Guide to start using OpenROAD on Chipshub.



Docker

- [Docker](#) is an app that lets you spin up “containers”
 - isolated copies of Linux-based operating systems prepackaged with apps
- For miniprojects, we provide a downloadable container image for OpenROAD and OpenROAD-flow-scripts
 - You can run OpenROAD locally without a Linux machine and without spending time on software dependencies



Final Notes

- With this lab, now is the time to iron out software issues
 - Use Piazza and Office Hours
- Skim both the Software Setup Guide and the Lab 0 template before getting started
- Start early
 - Leave time in case you need software support