Lab #6
OpenROAD

ECE4810J System-on-Chip Design

Yihua Liu
UM-SJTU Joint Institute
ayka_tsuzuki@sjtu.edu.cn
Nov. 7, 2022

Overview

Overview

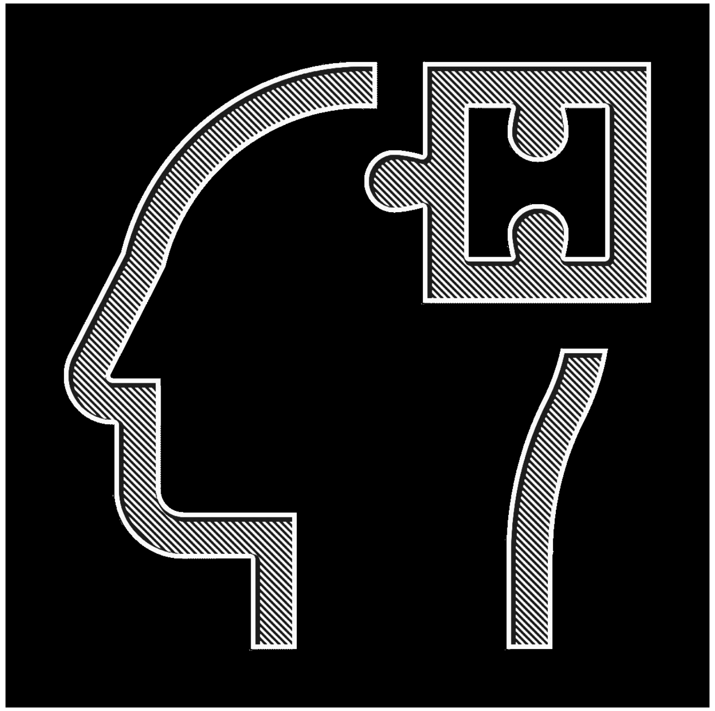
Motivation

Why should SoC designer and researchers care?

- Extending algorithms and techniques to real hardware designs
- More accurate design space exploration
- Hands on experience for job opportunities

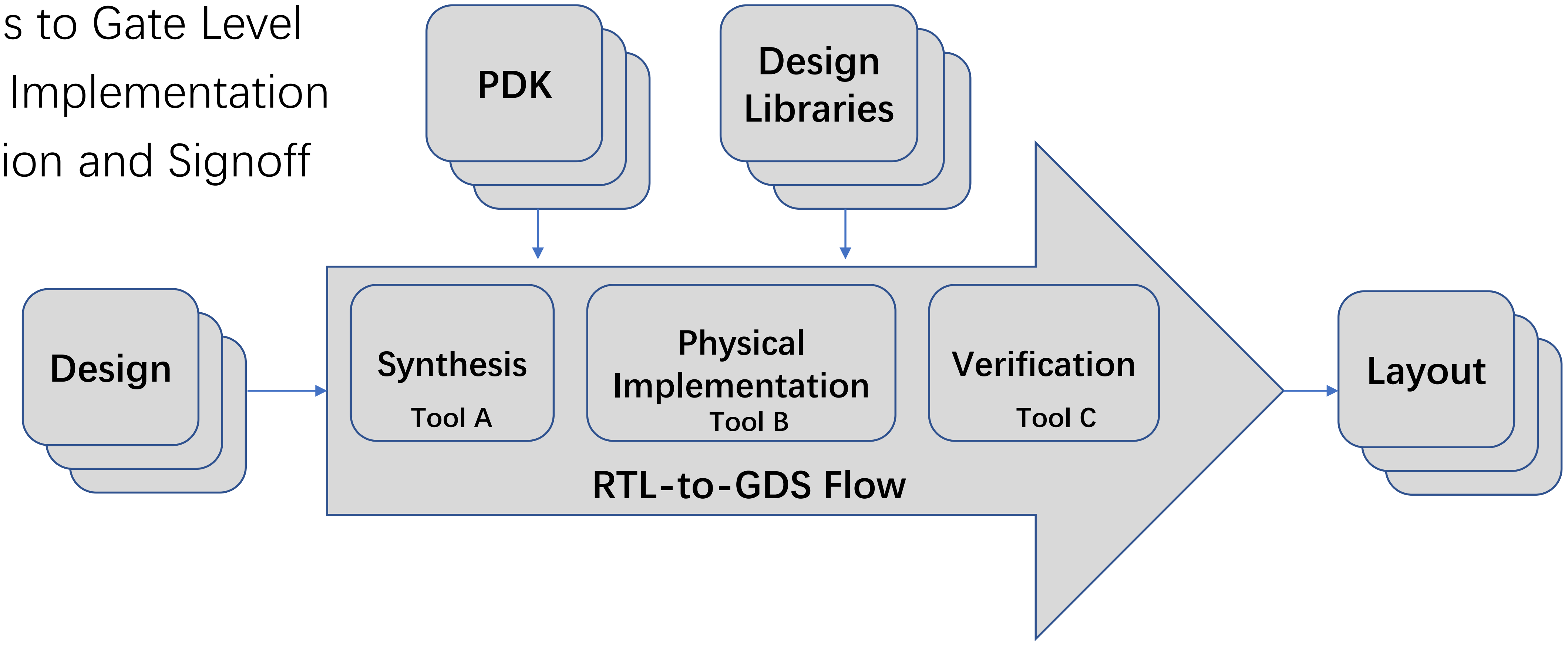
Why choose Open Source?

- Easier collaboration using publicly available IP and kits
- Reproducibility and Apples-to-Apples comparison of new implementations
- Easily re-use publicly available flows, best practices, designs and IP cores
- Support form the open-source community
- Opportunities for free/sponsored tape-outs
- FREE!



Chip Design Flow

1. Design Specification and Algorithm
2. RTL Implementation and Simulation
3. Synthesis to Gate Level
4. Physical Implementation
5. Verification and Signoff



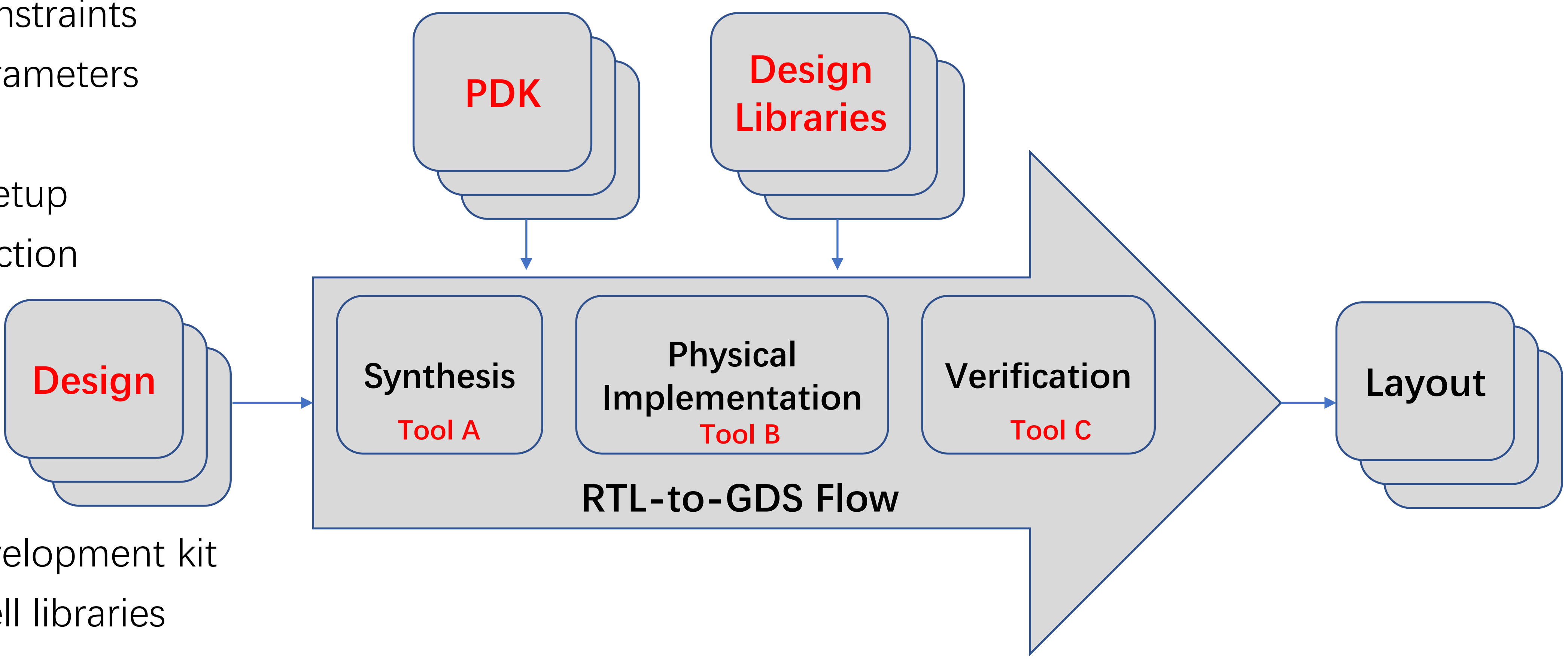
Design and Flow Preparation

Design Preparation

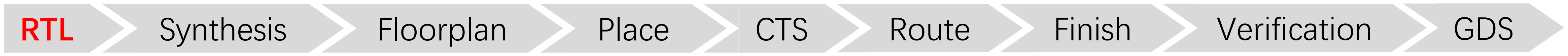
- RTL Files
- Timing Constraints
- Design Parameters

Flow Setup

- EDA Tool Setup
- Design selection



- Process development kit
- Standard cell libraries



Design Synthesis

Synthesis transforms RTL to netlist

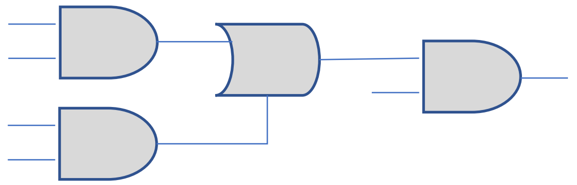
- RTL Parsing and Design Elaboration
- Generic Mapping
- Generic Optimizations
- Technology Mapping
- Technology Driven Optimization
- Constraint Checking / Adherence

OpenROAD flows leverages Yosys

```
x = a'bc + a'bc'
y = b'c' + ab' + ac
```

```
x = a'b
y = b'c' + ac
```

```
INVX1SC(.A(a),.Z(U1));
AND2X1SC(.A(U1),.B(b),.Z(U55));
AND2X1SC(.A(U2),.B(U3),.Z(U23));
ORX1SC(.A(U23),.B(U21),.Z(y));
```



RTL | **Synthesis** | Floorplan | Place | CTS | Route | Finish | Verification | GDS

Design Floorplanning

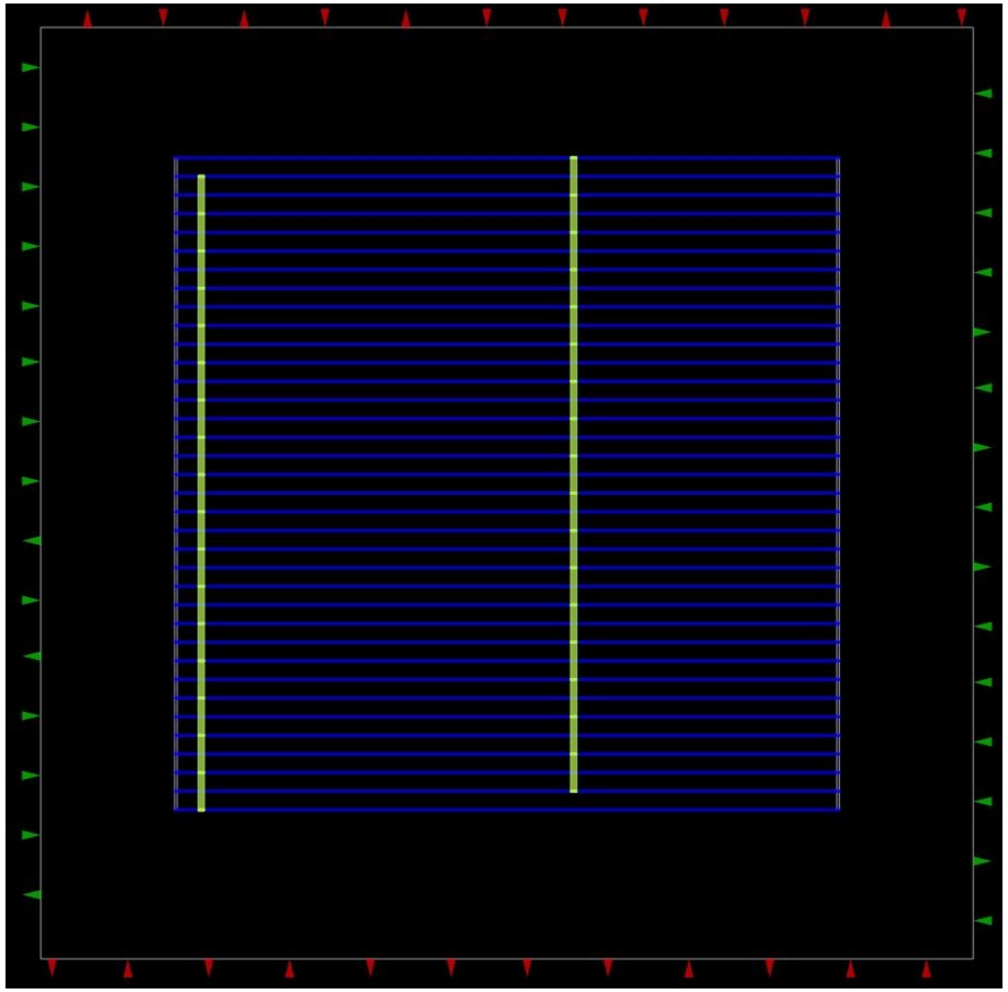
JOINT INSTITUTE 交大密西根学院

ASIC Fundamentals

- Standard Cells
- Standard Cell Rows
- Metal Stack

Power Grid

- Macros
- Design Specific
- Setting Die Area
- Assigning pin locations
- Placing hard macros
- Placing “guides” for cell placement



RTL → Synthesis → **Floorplan** → Place → CTS → Route → Finish → Verification → GDS

Design Placement

Global Placement

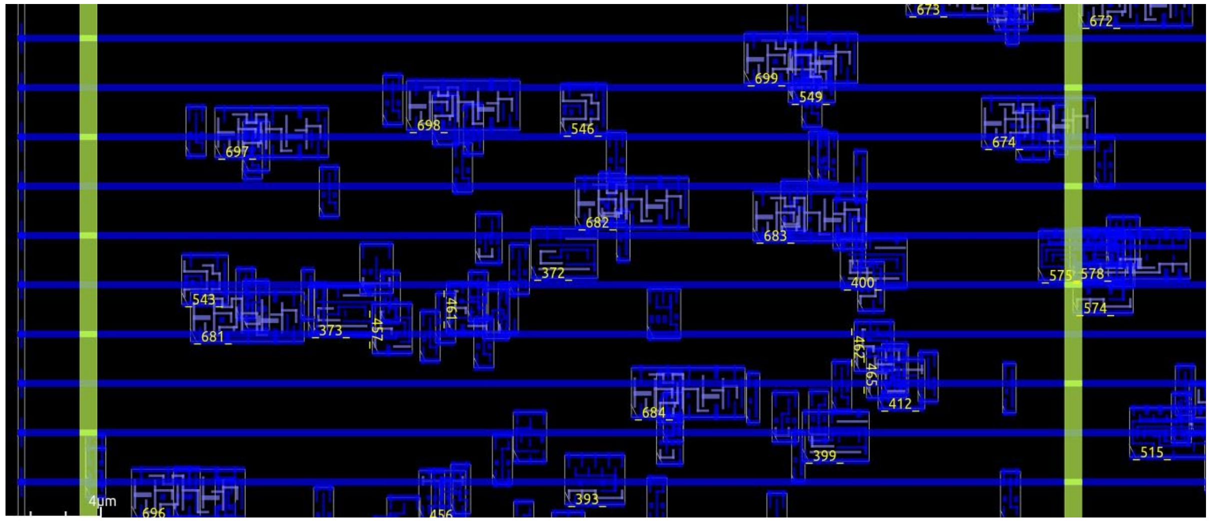
- Minimize congestion and log wires

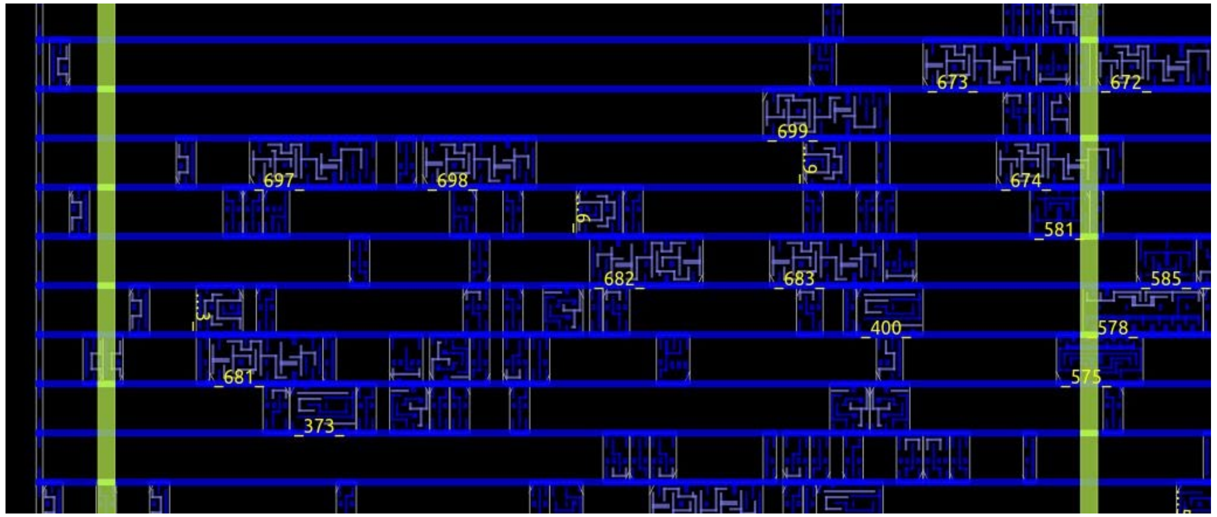
Placement Optimizations

- Resizing
- Buffering

Detail Placement

- Overlap
- Orientation

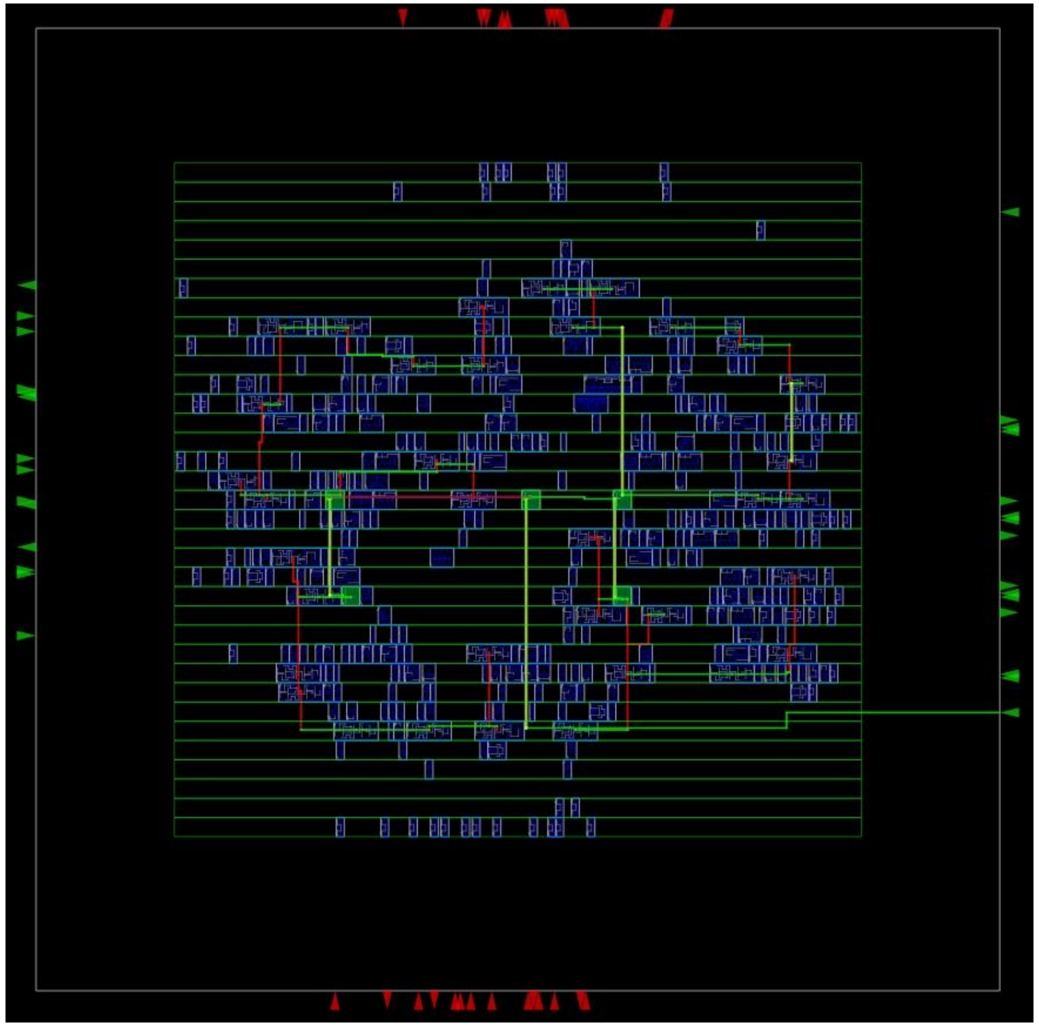




RTL | Synthesis | Floorplan | Place | CTS | Route | Finish | Verification | GDS

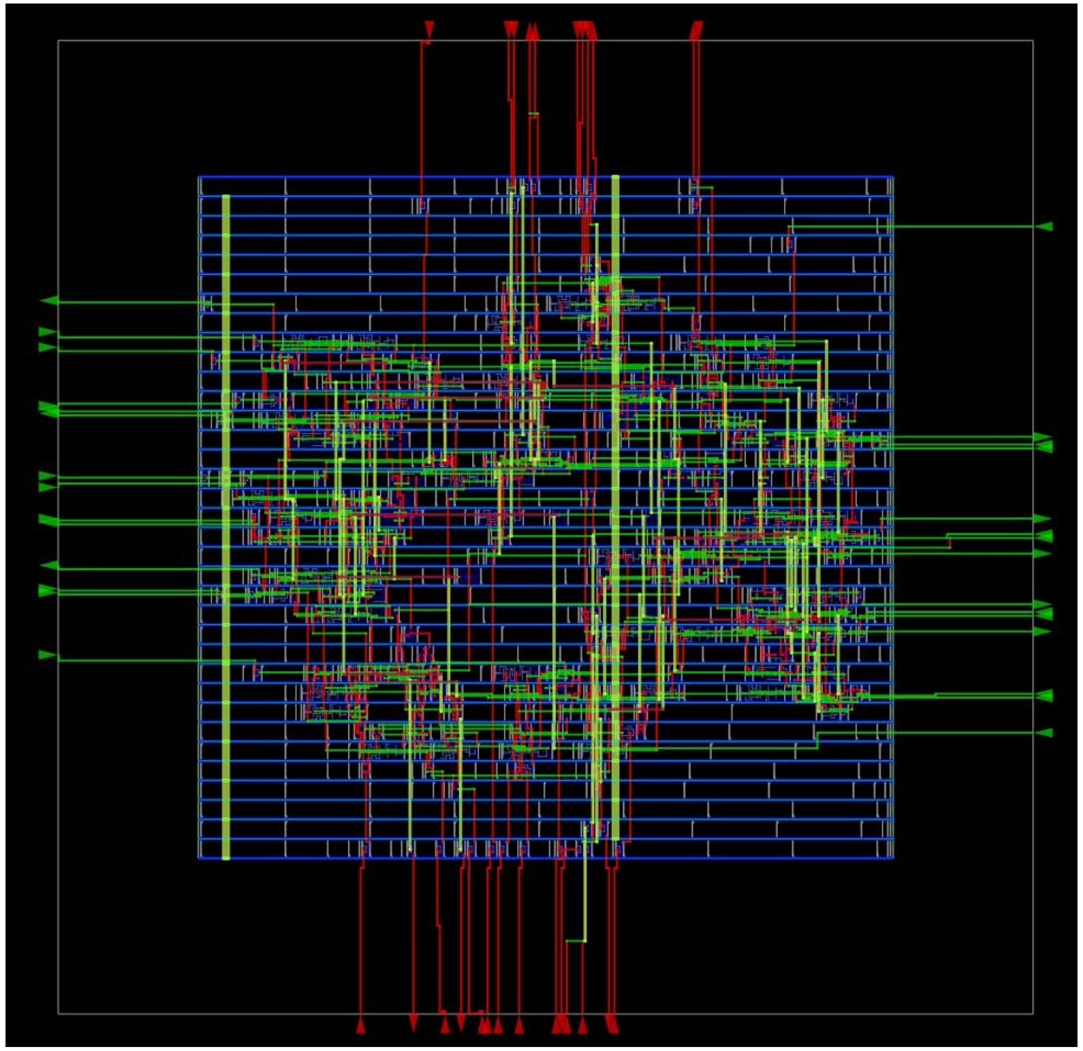
Clock Tree Synthesis (CTS)

- Clock trees are built and buffered
- Reducing Skew (setup/hold time)
- Inserting buffers for high fanout signals



Design Routing

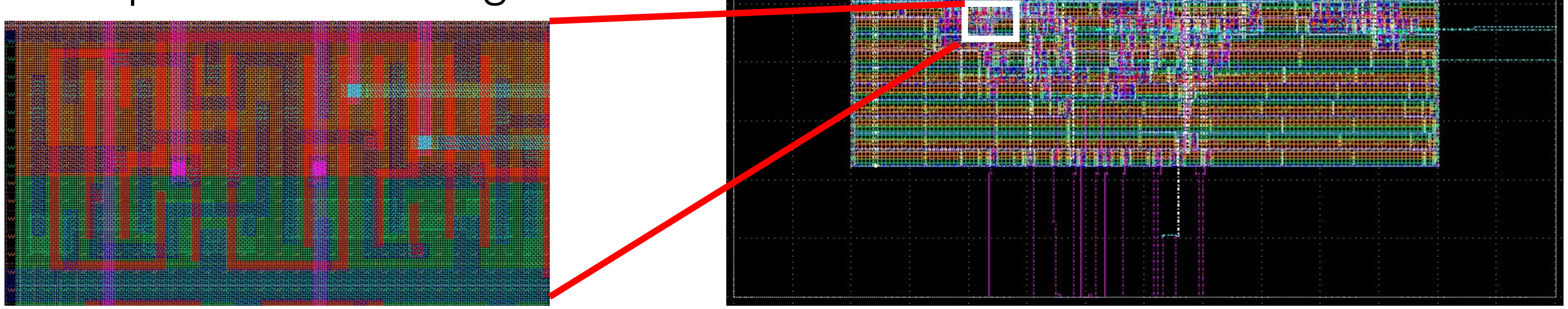
- Global Routing
- Detail Routing
- Routing optimization/fixing



RTL › Synthesis › Floorplan › Place › CTS › Route › Finish › Verification › GDS

Design Finish

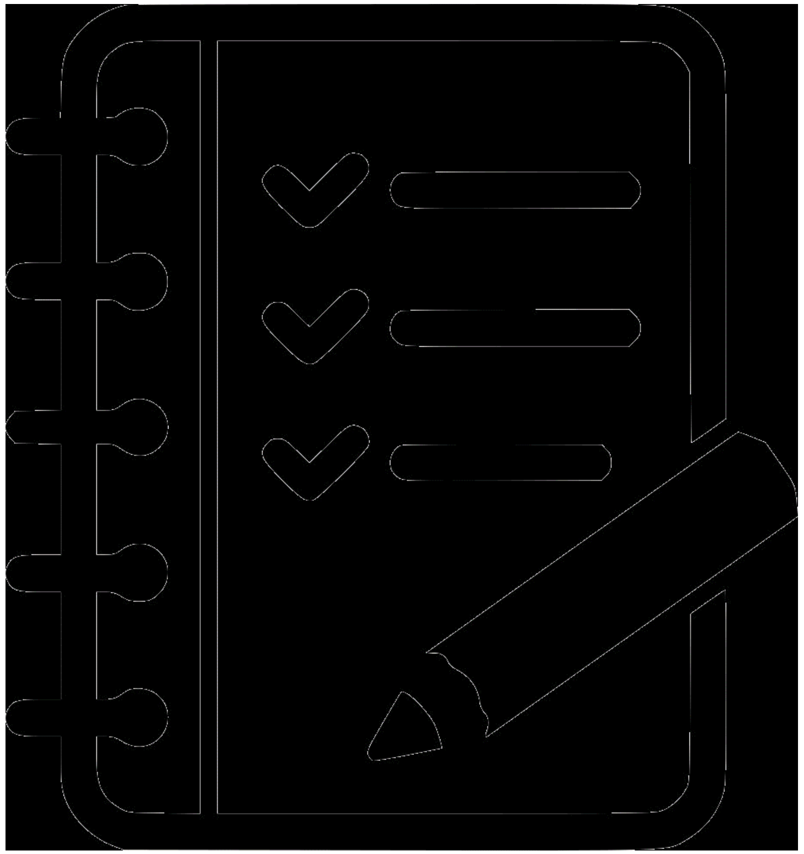
- Parasitic extraction
- Timing Signoff
- Dummy Metal Fill
- Export
  - ☐ Layout (GDS)
  - ☐ Netlist (Verilog)
  - ☐ Reports
- KLayout for GDS Export and Viewing



RTL > Synthesis > Floorplan > Place > CTS > Route > **Finish** > Verification > GDS

Design Verification

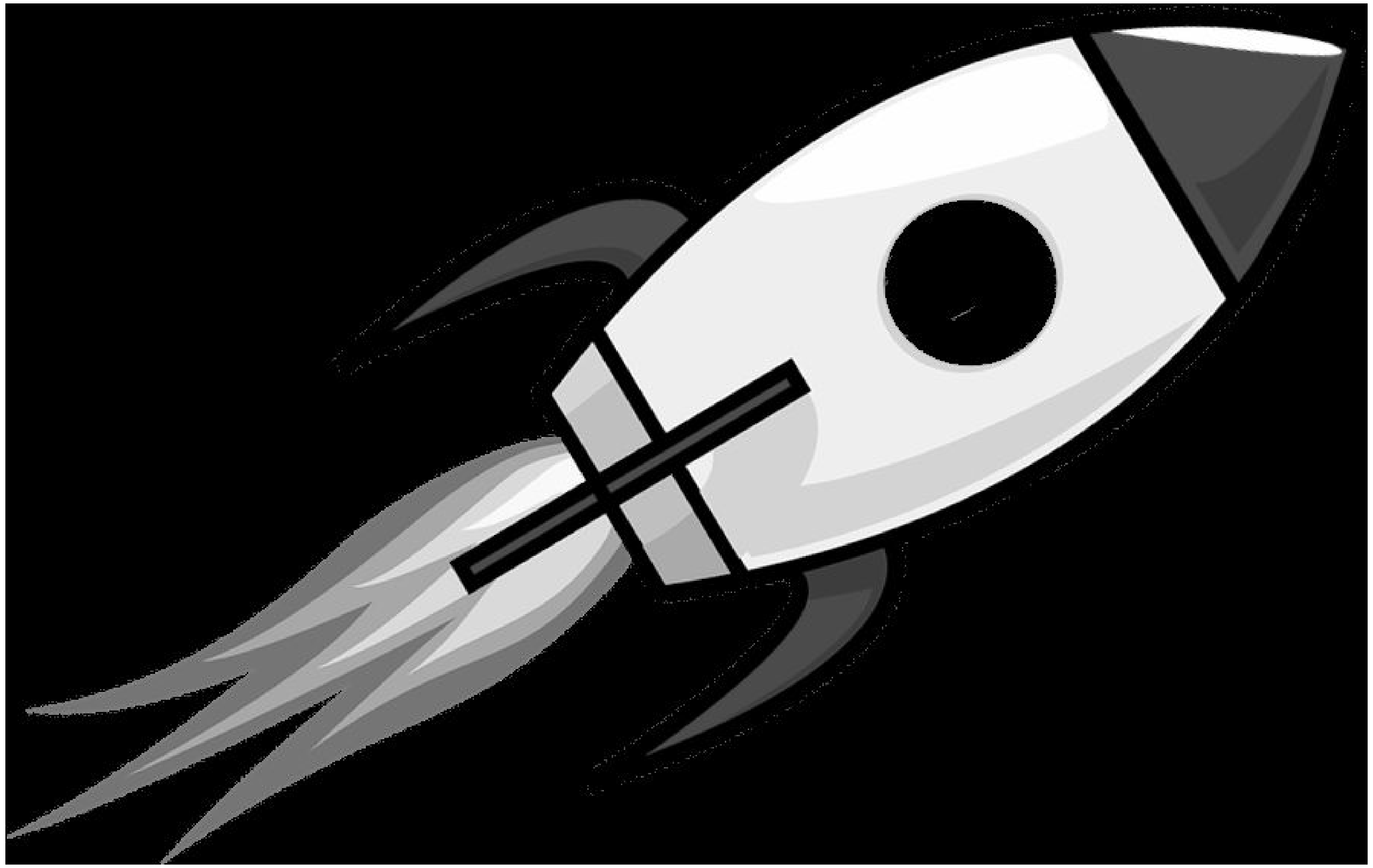
- Design Rule Check (DRC)
- Layout vs Schematic (LVS)
- Back-annotated Simulations



RTL → Synthesis → Floorplan → Place → CTS → Route → Finish → Verification → GDS

GDS

- Ready to send to fab!



RTL › Synthesis › Floorplan › Place › CTS › Route › Finish › Verification › GDS

OpenROAD-flow-scripts Structure

```
OpenROAD-flow-scripts/        Flow repository
|── docker         Dockerfiles (containerization)
|── flow           Flow – everything happens here!
|──  |── designs      Source RTL, configs, constraints for sample designs
|──  |── platforms    Platform data (.lib, .lef, .gds, etc.), configs
|──  |── scripts      Tcl scripts for OpenROAD, yosys
|──  |── test         Test scripts and run directory
|──  |── tutorials    Tutorials (WIP)
|──  |── util         Utility scripts (package issues, collect data, other misc.
|── jenkins        Continuous integration (CI)
|── tools          OpenROAD, yosys source repos; binaries
```

Platform Configs vs. Design Configs

```
|—— platforms
|    |——— [PLATFORM]
|    |     |—— config.mk
|—— designs
     |——— [PLATFORM]
           |—— config.mk
```

```
export TECH_LEF = …
export SC_LEF = …
export LIB_FILES = …
export GDS_FILES = …
```

Technology files

```
export CELL_PAD_IN_SITES_GLOBAL_PLACEMENT ?= …
export CELL_PAD_IN_SITES_DETAIL_PLACEMENT ?= …
export PLACE_DENSITY ?= …
```

Good default parameters

```
export DESIGN_NAME   = …
export PLATFORM      = …
export VERILOG_FILES = …
export SDC_FILE      = …
export DIE_AREA      = …
export CORE_AREA     = …
```

Design files

```
export PLACE_DENSITY = …
```

Parameter overrides

Debugging Common Design Problems

What Do Messages Mean?

- INFO: Report data, status, or current progress
- WARNING: Unexpected situation, but tools will do best to continue
  - Designer should fix warnings or validate they are benign
- ERROR: Unexpected situation, tools cannot work around issue
- CRIT: openroad must exit immediately (rare)
  - All segfaults / asserts / crashes are bugs :)

Debugging Strategy

- Review error which caused flow to abort
- Check warnings and errors starting from beginning of flow
  - ◻ Early warnings can be cause of later errors
- Try to identify root cause of issue
  - ◻ Design problem?
  - ◻ Tool problem?
  - ◻ Unrealistic expectations?

Common Problems and Solutions

- Utilization too high - fails placement
  - Increase die area or decrease core utilization
- Utilization too high - fails resizing
  - Check for proper SDC constraints
  - Check that user-generated macros have reasonable constraints (e.g. good .lib files)
- Congestion too high - fails global routing
  - Try previous fixes
  - Try decreasing layer adjustment
- Congestion too high - fails detail routing
  - Try previous fixes
  - Try adding cell padding to space cells further apart
  - If violations always occur on same cell(s), try marking those cells as dont_use
- Design too small - fails PDN generation
  - Try increasing design size or reducing power grid pitch

Common Problems and Solutions

- Design runtime too long
  - Increase utilization if too low
  - Relax timing constraints
  - Reduce design complexity
  - Faster machine :)
- Failing setup time
  - Hard problem - may just need to reduce constraints
  - Change architecture: more pipelining, reduce complexity
- Failing hold time
  - Check that user cells (e.g. SRAM) are properly constrained
  - Check design constraints are valid (SDC)
    - Designs with multiple clocks are tricky!
  - Check that your PDK has properly correlated parasitics

Analyzing Your Design

Reporting Chip Metrics – Area

- Different area numbers mean different things
- Some metrics assume 100% utilization – 70-90% more typical
- Buffering and clock tree can add significant area (20%+)
- Chip I/O (pad rings, etc.) & fab markers (fiducials, etc.) rarely accounted for
- Test interfaces can add significant area too!

	Logic	SRAM	Buffers	Clock tree	Chip I/O	Fab Markers	Unutilized Space
Synthesized	√	√	Some	×	Usually no	×	×
Places & Routed	√	√	√	√	Usually no	×	×
“Die area”	√	√	√	√	Sometimes	Usually no	√
“Die size”	√	√	√	√	√	√	√

Reporting Chip Metrics – Power

- Buffers and clock tree consume significant power (40%+)
- Chip I/O can be simulated but usually isn't
- Simulation type makes a huge difference!
  - Activity factor vs. switching activity (SAIF) vs. vector (VCD)

	Logic	SRAM	Buffers	Clock tree	Chip I/O	Supply losses
Synthesized	√	√	Some	×	Usually no	×
Places & Routed	√	√	√	√	Usually no	×
"Die area"	√	√	√	√	Sometimes	×
"Die size"	√	√	√	√	√	√

Reporting Chip Metrics – Frequency

- Classic synthesis can provide mediocre/poor estimates of real chip frequency
- Physical synthesis provides much better estimates
- Place & route offers excellent estimates
  - Typical, best, worst, and other modeling corners
- Real chips have a distribution of frequencies and are binned

	Parasitics model	Gate timing model	Clock tree model
Synthesized	Wire-load	Usually “typical corner”	Ideal
Places & Routed	Estimated	Usually “typical corner”	Estimated
“Die area”	Extracted	Usually “typical corner”	Extracted
“Die size”		Binned	

Limitations and Future Directions

OpenROAD Roadmap – Active Projects

- Ease of use
  - ☐ Simplify install process
  - ☐ Broaden OS support
  - ☐ Python API, Python module
  - ☐ Documentation improvements
- Improved support
  - ☐ Support and tune additional PDKs
  - ☐ Support additional technology rules
- Enhanced features
  - ☐ Hierarchical implementation
  - ☐ Universal Power Format (UPF) support
- Maintenance
  - ☐ Code cleanup and optimization

OpenROAD Roadmap – Long-term Projects

- Enhanced Features
  - ❑ Vector-based power calculation
  - ❑ CCS timing engine
  - ❑ Incremental implementation
- COPILOT: >100x improvement to tool throughput
  - ❑ Massively distributed workloads
- ML-based EDA
  - ❑ Interfaces for data collection
  - ❑ ML-guided optimization
- Education and outreach
  - ❑ Courses, tutorials, and more!

OpenROAD Limitations

- Ease of use
  - Mediocre support for SystemVerilog (yosys)
  - Lack of design checking / sanity checking
- Quality of Results
  - No multi-Vt flow yet
  - No automatic clock gating yet
  - Lacking quality hierarchical implementation
    - Slow runtime on large designs
- Design features
  - Hierarchical extraction accuracy is limited
  - Analog / mixed signal support is very preliminary

OpenROAD Advantages

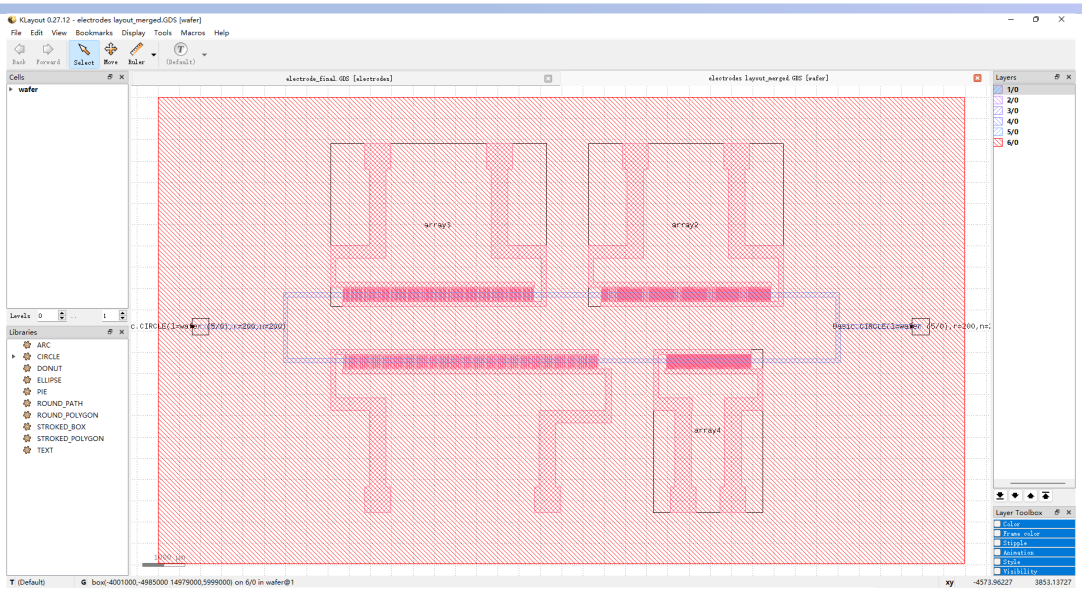
- Accessibility
  - No license limitations / license servers!
    - Run 100s of OpenROAD instances for free
  - Access to source code for debugging / modification
  - Share and get help with tool questions (no paywalls)
- Active community
  - Updates nearly daily
  - Issues fixed and upstreamed in days, not months
  - Pull requests accepted for any useful fixes / features
- Reproducibility
  - Easy to package designs and reproduce exactly
  - Able to validate other’s research

OpenLane vs. OpenROAD-flow-scripts

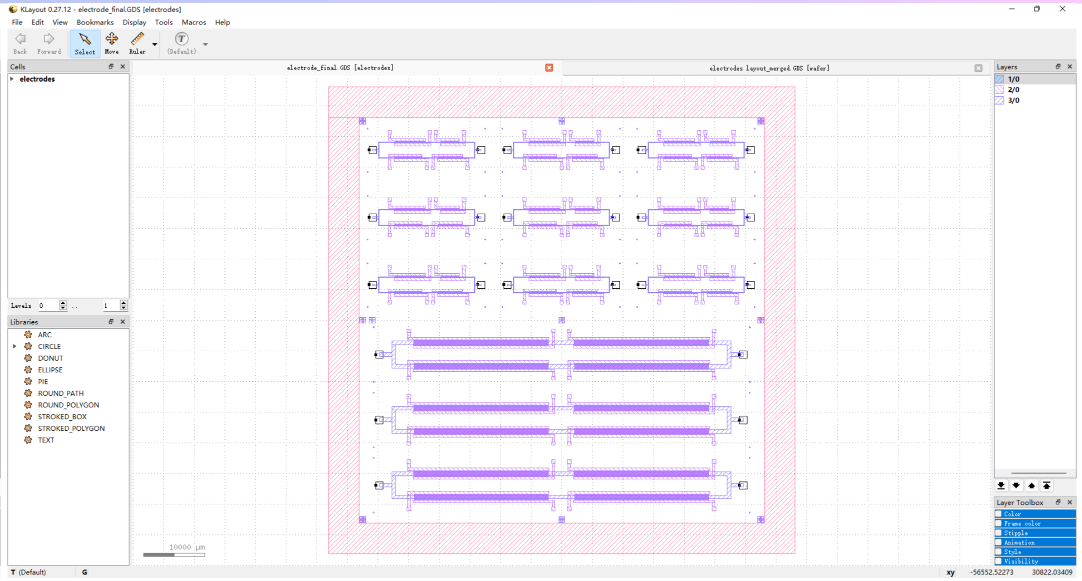
- Based off OpenRoad, yosys
- Support only for sky130
- Focus full-chip open-source signoff for sky130
- Tcl-based flow
- Only supports Docker
- Maintained by Efabless

- Based off OpenRoad, yosys
- Support for several PDKs
- Focus on full-chip closed-source signoff
- Make-based flow
- Supports Docker, native execution
- Maintained by OpenROAD team

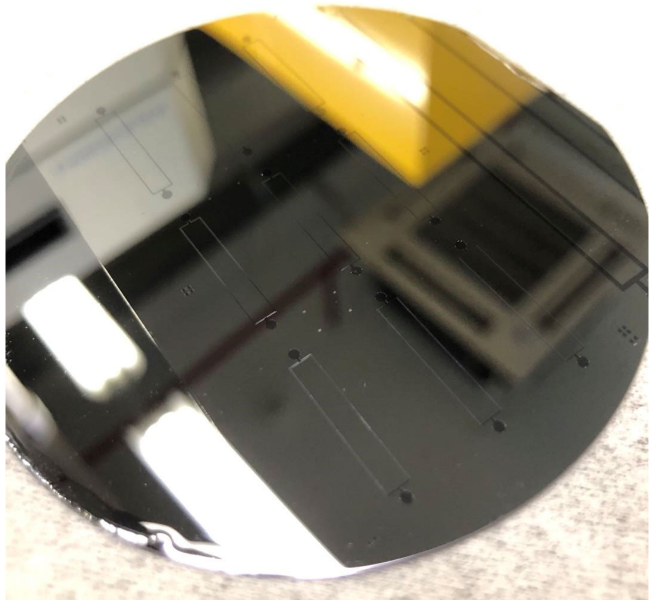
KLayout

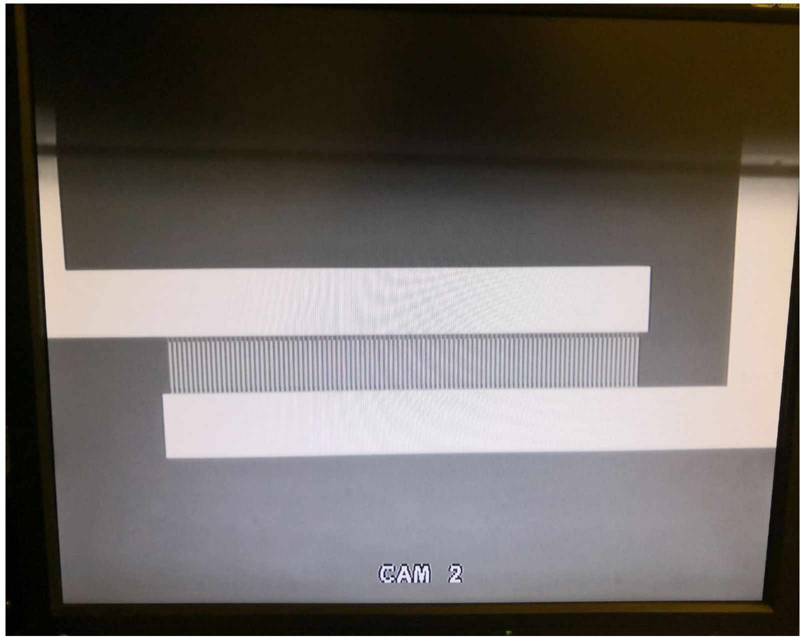


KLayout 0.27.12 - electrodes layout_merged.GDS [wafer]
File Edit View Bookmarks Display Tools Macros Help
Back Forward Select Move Ruler (Default)
Cells
wafer
electrode_final.GDS [electrodes]
electrodes layout_merged.GDS [wafer]
Layers
1/0
2/0
3/0
4/0
5/0
6/0
array1
array2
array3
array4
Libraries
ARC
CIRCLE
DONUT
ELLIPSE
PIE
ROUND_PATH
ROUND_POLYGON
STROKED_BOX
STROKED_POLYGON
TEXT
Layer Toolbox
box(-4001000,-4985000 14979000,5999000) on 6/0 in wafer@1



KLayout 0.27.12 - electrode_final.GDS [electrodes]
File Edit View Bookmarks Display Tools Macros Help
Back Forward Select Move Ruler (Default)
Cells
electrodes
electrode_final.GDS [electrodes]
electrodes layout_merged.GDS [wafer]
Layers
1/0
2/0
3/0
Libraries
ARC
CIRCLE
DONUT
ELLIPSE
PIE
ROUND_PATH
ROUND_POLYGON
STROKED_BOX
STROKED_POLYGON
TEXT
Layer Toolbox
10000 μm





CAM 2

PyMTL3 Supplements

`pytest ../regincr`

- `-s --capture=no`: generate a line trace for a test
- `--tb=short`: produce more detailed error output (short)
- `--tb=long`: produce more detailed error output (long)
- `--dump-vcd`: generate (VCD) waveforms for a test
- `-v --verbose`: verbose output where each test case is listed on a separate line; passing test cases are marked with PASSED and failing test cases are marked with FAILED.
- `-k`: select just a few test cases to run and debug in more detail
- `-x`: have pytest stop after the very first failing test case

⚠ Options like `–test-verilog` requires `verilator` which is only available on Linux
Only `pip install pymtl3` (and `pytest)` in Python virtual environments! Do not install in user directories like `/home/<username>/pymtl3/lib/python3.10/site-packages/` or `sudo` install in root directories like `/usr/local/lib/python3.10/dist-packages/` and `/usr/local/bin`. Follow the instruction on pymtl/pymtl3: Pymtl 3 (Mamba), an open-source Python-based hardware generation, simulation, and verification framework (github.com). `pytest` should be located at `<venvname>/bin/pytest`.

PyMTL3 Supplements

Pay attention to the `py.py` file and the `py` module. PyMTL3 uses the `py` module in `pymtl3/passes/sim/PrepareSimPass.py`. In case the `py` module is name-conflicted with the `<venvname>/lib/python3.10/site-packages/py.py` file, you should delete the `py.py` file.

PyMTL3 officials require the specific version of verilator: Verilator 4.036, while the latest Verilator in the APT repository is Verilator 4.038 2020-07-11 rev v4.036-114-g0cd4a57ad. In most cases, this version works well, but if you encounter problems using this version, you should better build Verilator 4.036 from source. Besides, the build of Verilator 4.036 can also be problematic. See Bug#966909: Fixed upstream (mail-archive.com) and Fix build with Bison 3.7 and newer by rswarbrick · Pull Request #2505 · verilator/verilator (github.com). You should patch the fix manually or use Bison version < 3.7.

Reference

① Kexin Li and Yihua Liu. “Microfluidic robot with AC osmosis-based asymmetric electrode pair array fabricated by integrated circuit technologies.” Apr. 30, 2021.

② Austin Rovinski, Tutu Ajayi, and Christopher Batten. “OpenROAD Tutorial: Open-Source ASIC Design for Computer Architects. Oct. 1, 2022.