# OpenROAD Tutorial

Open-Source ASIC Design for Computer Architects

Austin Rovinski
Tutu Ajayi
Christopher Batten

# Presenters / Organizers

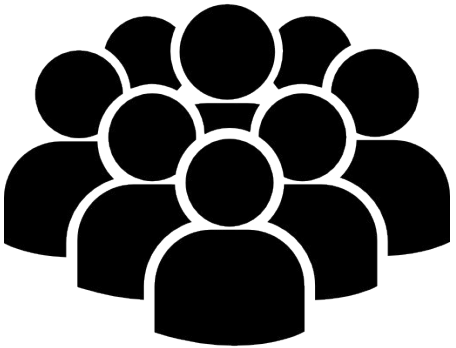
**Austin Rovinski**
Cornell University

**Tutu Ajayi**
University of Michigan

**Chris Batten**
Cornell University

# Audience

Computer architects looking to:

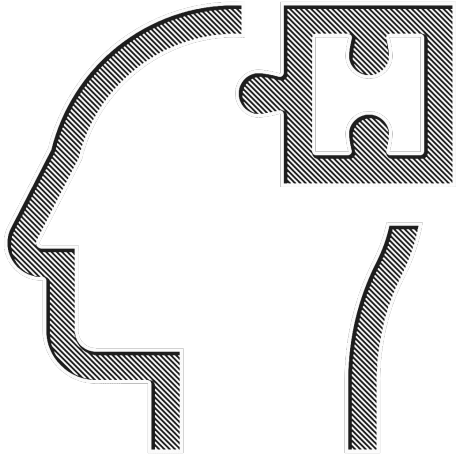


- Enhance research with accurate modeling
- Learn the basics about chip design
- Explore OpenROAD and other open-source resources
- Explore chip design techniques and algorithms
    - Improve hardware design across the stack
    - Improve EDA tools

# Motivation

Why should computer architects and researchers care?

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

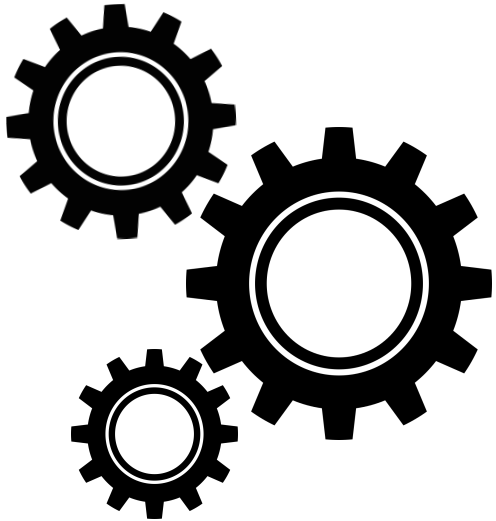


Why choose Open Source?

- 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!

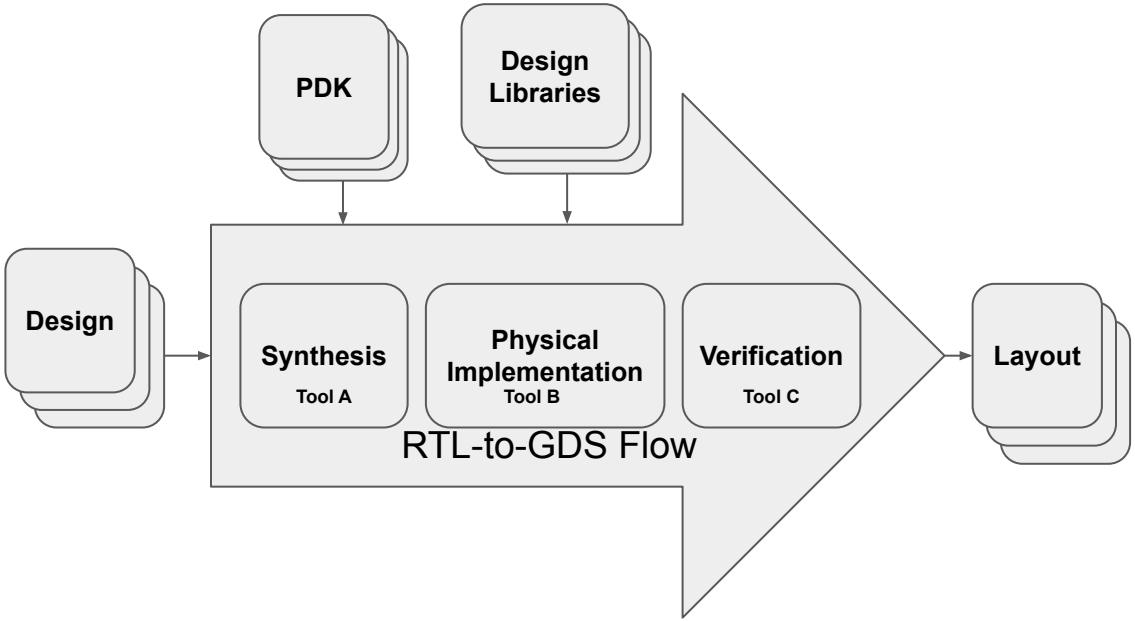
# Goals / Schedule

- Introduction to chip design and flow
  - Basic demonstration and illustration
- OpenROAD Tutorial
  - Overview of OpenROAD flows and abstractions
  - Demos and exercises using the OpenROAD flow
- Further discussions
  - OpenROAD limitations
  - OpenROAD roadmap?
  - How can I contribute to OpenROAD?
  - Additional information on other open-source resources



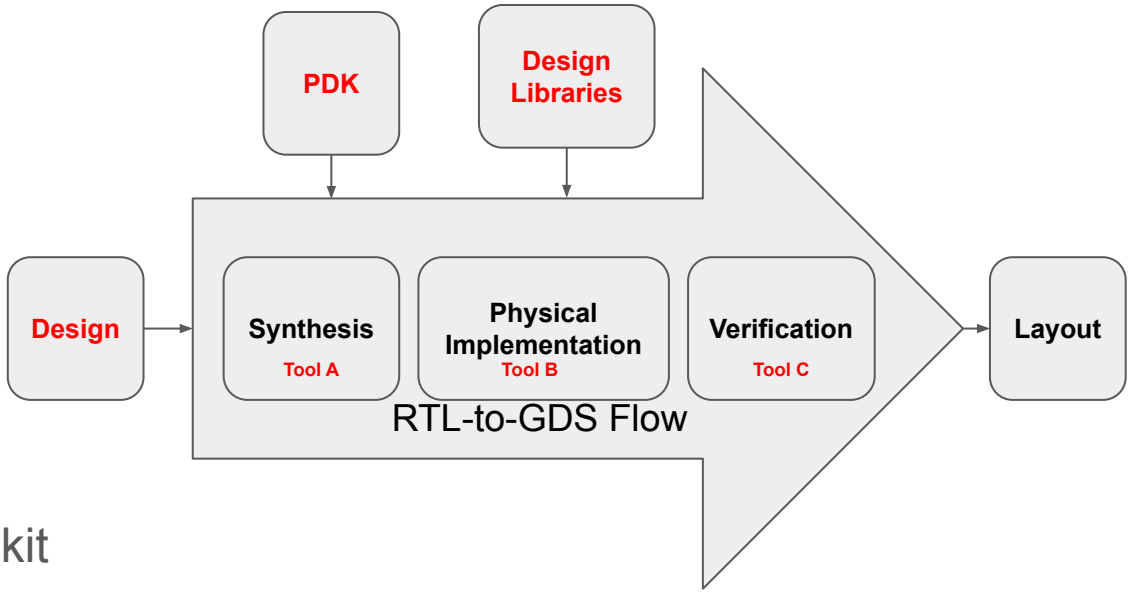
# Chip Design Flow

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



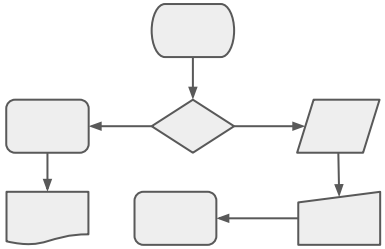
# Design and Flow Preparation

- Design Preparation
  - RTL Files
  - Timing Constraints
  - Design Parameters
- Flow Setup
  - EDA Tool Setup
  - Design selection
  - Process development kit
  - Standard cell libraries




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

# Design Synthesis

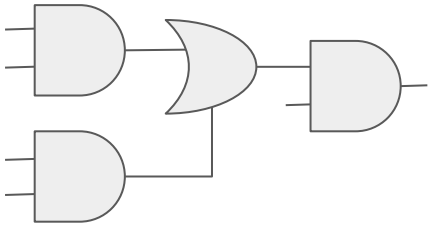


- Synthesis transforms RTL to netlist
  - RTL Parsing and Design Elaboration
  - Generic Mapping
  - Generic Optimizations
  - Technology Mapping
  - Technology Driven Optimization
  - Constraint Checking / Adherence
- OpenROAD flows leverages Yosys

```
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));
OR2X1SC(.A(U23),.B(U21),.Z(y));
```



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

# OpenROAD Synthesis

Demo

# OpenROAD Synthesis

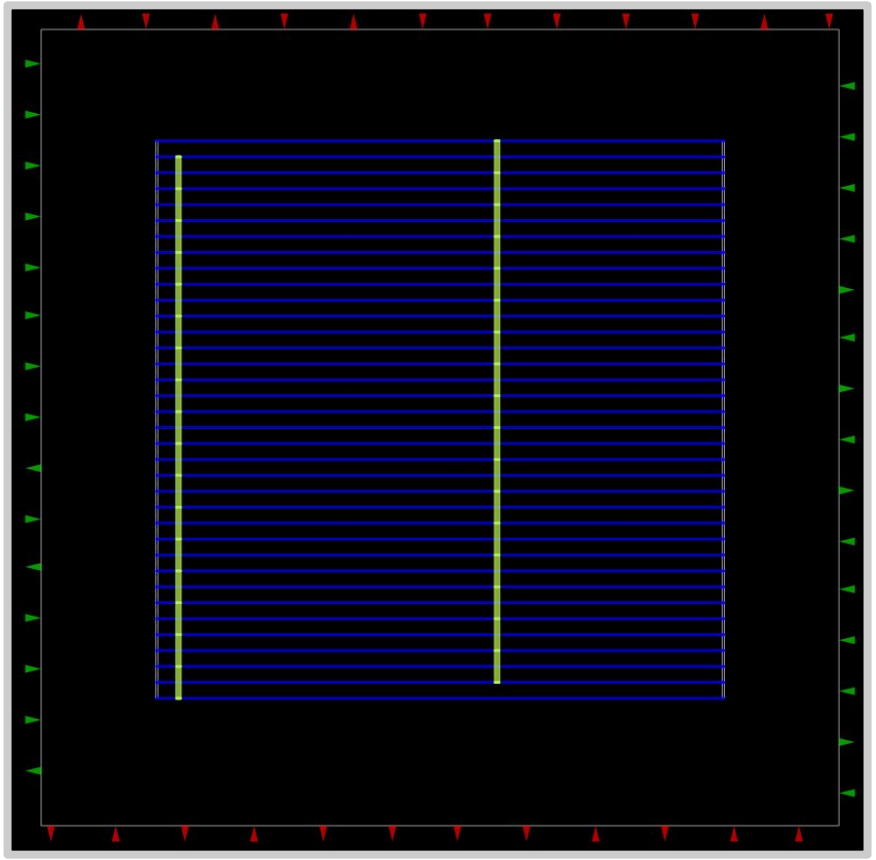
1. Perform file preprocessing (mainly for yosys)
2. Parse input files
3. Elaborate the design
4. Optimize the netlist
5. Map the generic netlist cells to technology specific cells
6. Generate Verilog netlist

```
17. Executing Verilog backend.
```

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

# Design Floorplanning

- 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



# OpenROAD Floorplanning

Demo

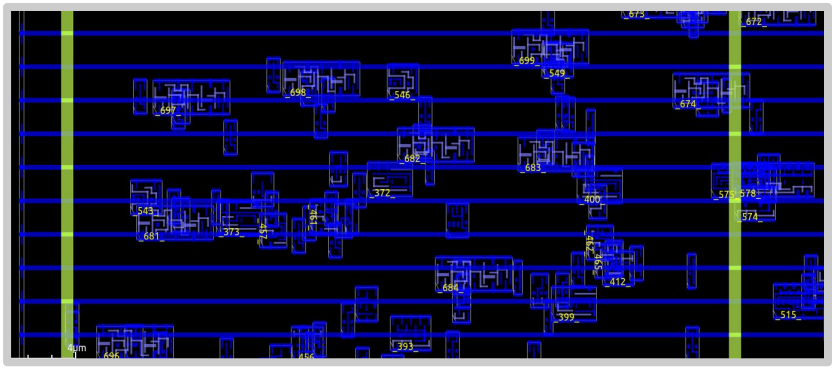
# OpenROAD Floorplanning

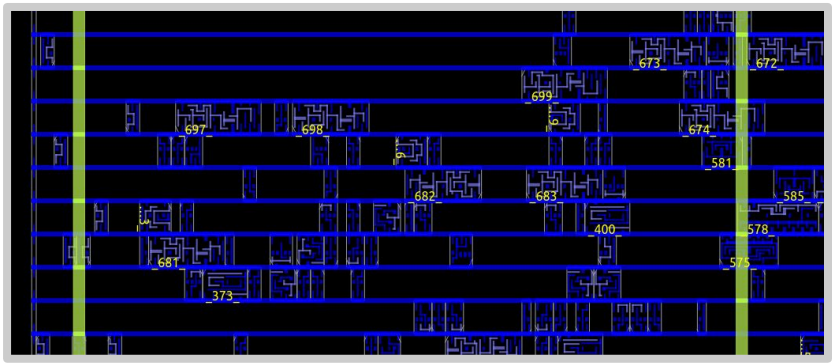
1. Initialize chip area
2. I/O pin placement
3. Insert tapcells and endcaps
4. Generate power grid

```
[INFO PDN-0001] Inserting grid: grid
```

# 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

# OpenROAD Placement

Demo

# OpenROAD Placement

▼ Global Placement

2. Nesterov gradient descent (with timing-driven weighting)

3. Timing optimization and electrical rule fixing

```
Perform port buffering...
[INFO RSZ-0027] Inserted 35 input buffers.
[INFO RSZ-0028] Inserted 18 output buffers.
Perform buffer insertion...
[INFO RSZ-0058] Using max wire length 661um.
[INFO RSZ-0039] Resized 39 instances.
Repair tie lo fanout...
Repair tie hi fanout...
```

# OpenROAD Placement

## Detailed Placement

1. Optimize and legaliize placement

```
Detailed placement improvement.
```

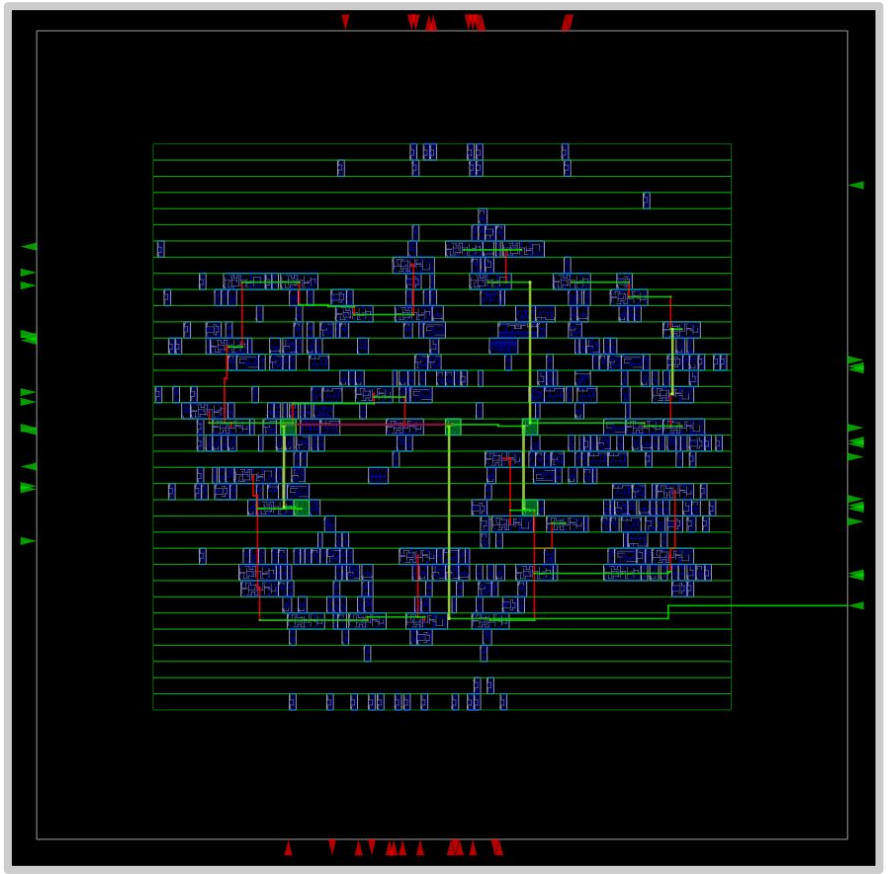
2. Cell mirroring

```
INFO DPL-0020] Mirrored 20 instances
[INFO DPL-0021] HPWL before            2703.4 u
[INFO DPL-0022] HPWL after             2700.8 u
[INFO DPL-0023] HPWL delta               -0.1 %
```

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

# Clock Tree Synthesis (CTS)

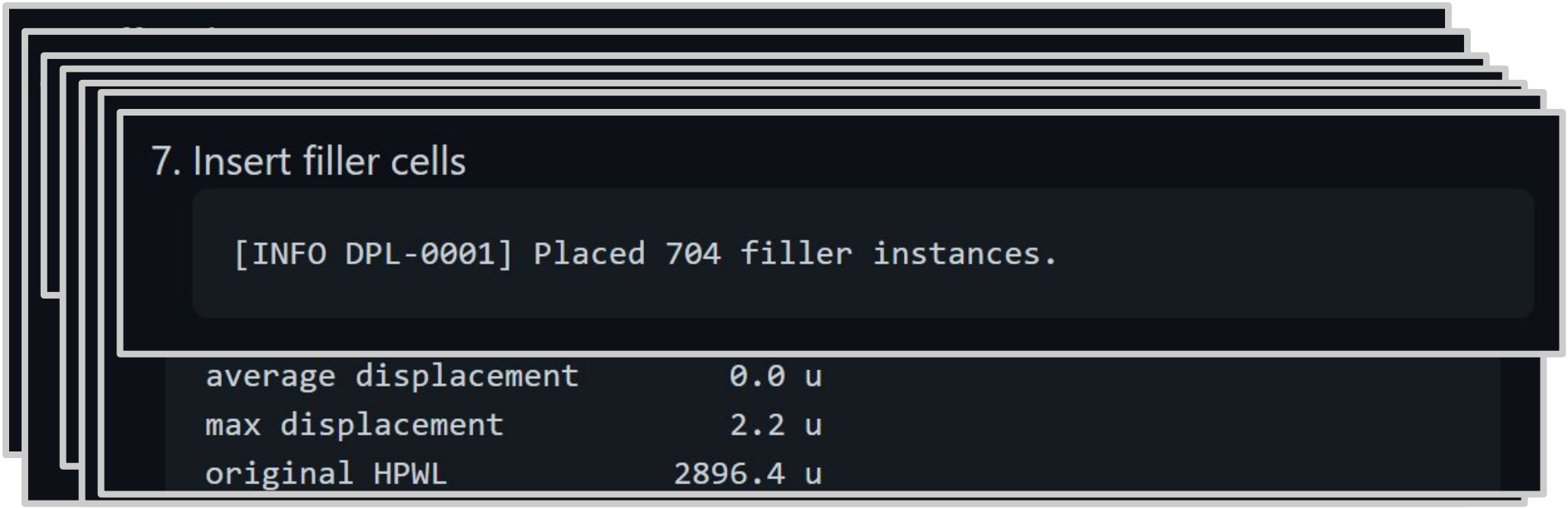
- Clock trees are built and buffered
- Reducing Skew (setup/hold time)
- Inserting buffers for high fanout signals



# OpenROAD CTS

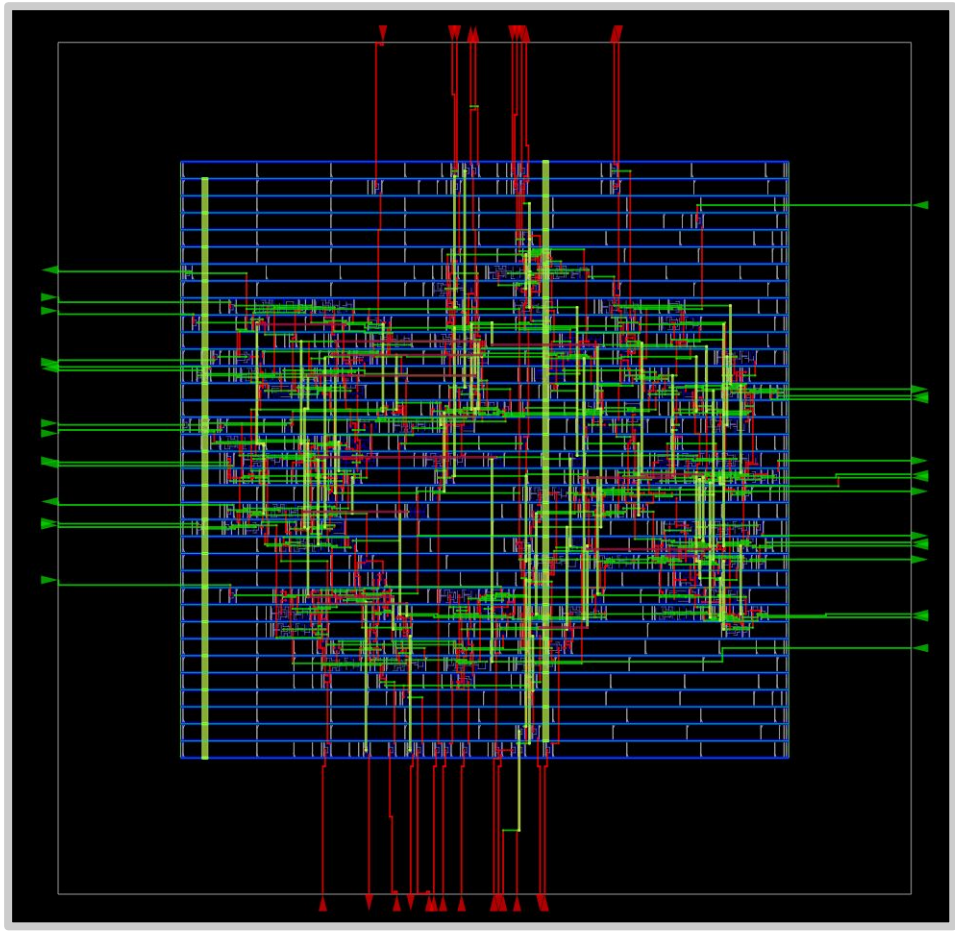
Demo

# OpenROAD CTS



# Design Routing

- Global Routing
- Detail Routing
- Routing optimization/fixing



# OpenROAD Routing

Demo

# OpenROAD Routing

1. Generate routing grid
2. Perform global routing
3. Check for antenna violations

```
[INFO ANT-0002] Found 0 net violations.
[INFO ANT-0001] Found 0 pin violations.
```

# OpenROAD Routing

1. Region query

4. Track assignment

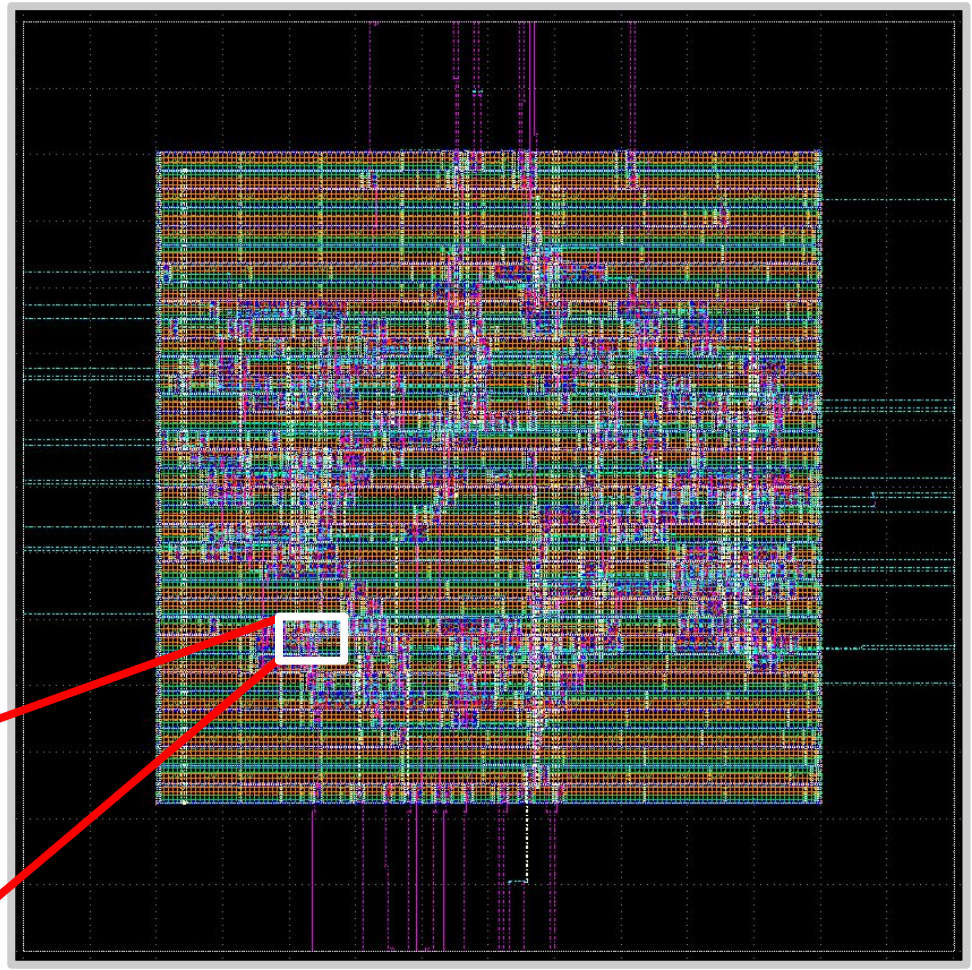
5. Detailed routing

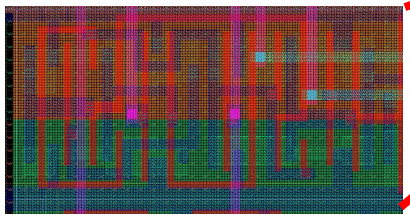
```
[INFO DRT-0194] Start detail routing.
[INFO DRT-0195] Start 0th optimization iteration.
    Completing 10% with 0 violations.
    elapsed time = 00:00:00, memory = 96.41 (MB).
    Completing 20% with 0 violations.
    elapsed time = 00:00:00, memory = 96.70 (MB).
```

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

# OpenROAD Finishing

Demo

# OpenROAD Finishing

1. Report final timing
2. Report final electrical violations

```
finish max_slew_violation_count
---------------------------------------------------------------------
max slew violation count 0

=====================================================================
finish max_fanout_violation_count
---------------------------------------------------------------------
max fanout violation count 0
```

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

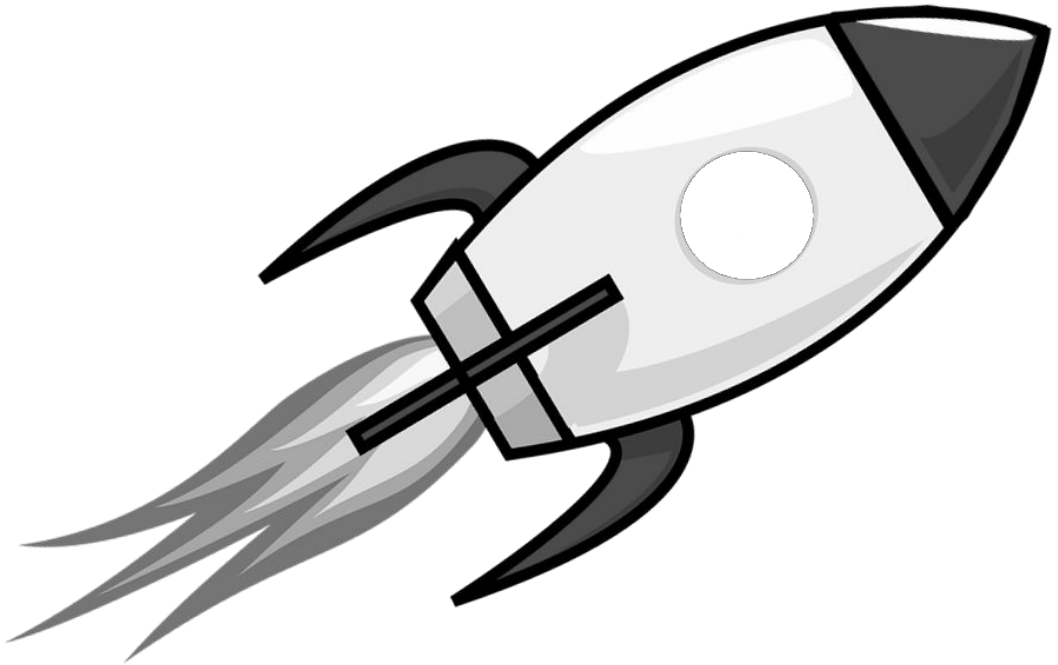
# Design Verification

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

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

# GDS

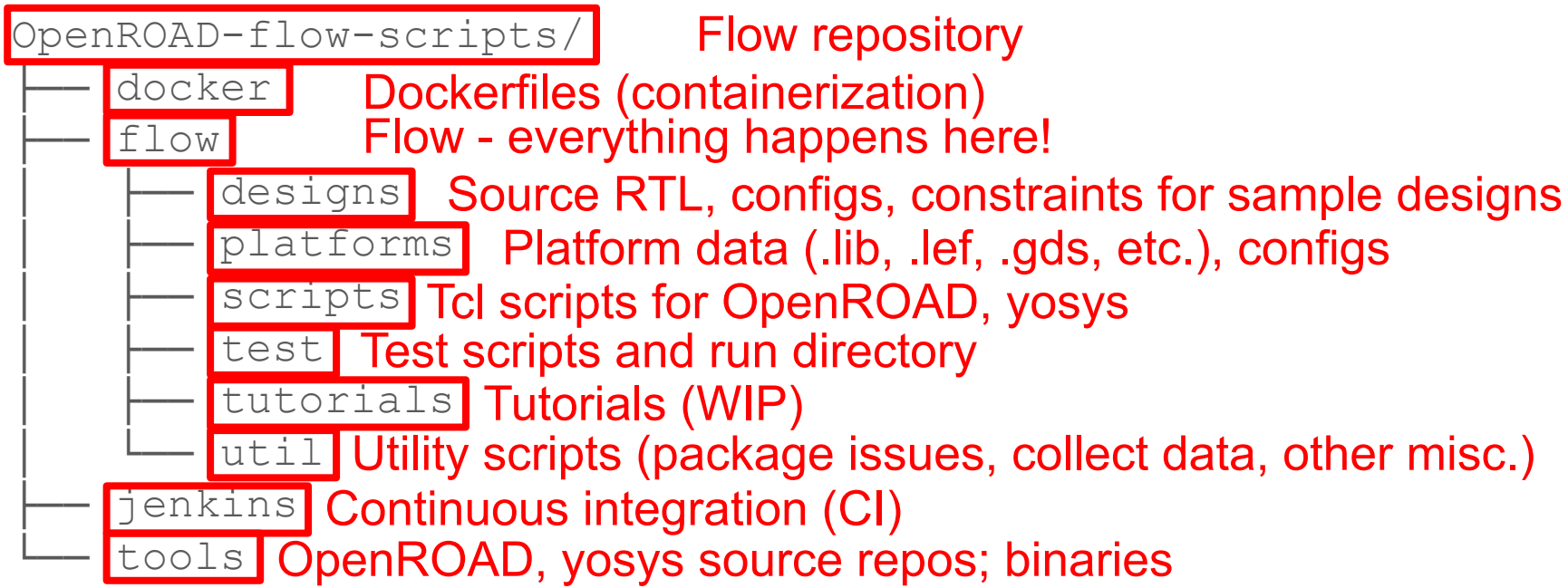
- Ready to send to fab!



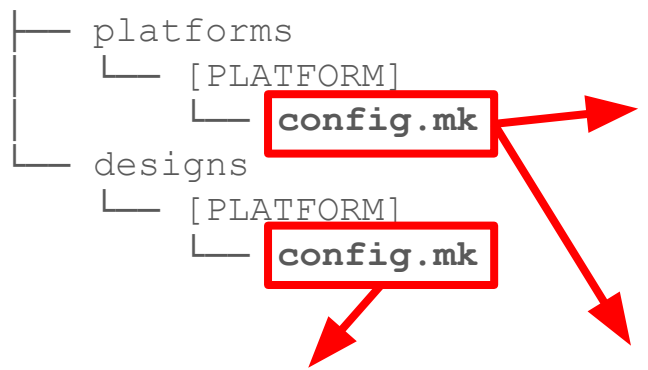
RTL | Synthesis | Floorplan | Place | CTS | Route | Finish | Verification | GDS

# Break

# OpenROAD-flow-scripts Structure



# Platform Configs vs. Design Configs




```
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
  - Designer 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

# Exercises 1 & 2

# 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 | ✖ | ✖ |
| Placed & 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 | ✖ |
| Placed & Routed | ✓ | ✓ | ✓ | ✓ | Usually no | ✖ |
| Real chip | ✓ | ✓ | ✓ | ✓ | Sometimes | ✖ |
| Wall power | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |

# 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 |
|---|---|---|---|
| Synthesis | Wire-load | Usually “typical corner” | Ideal |
| Physical Synthesis | Estimated | Usually “typical corner” | Estimated |
| Placed & Routed | Extracted | Usually “typical corner” | Extracted |
| Real chip | Binned | | |

# Demo 2

# Exercise 3

# Exercise 4

# Demo 3

# Exercise 5

# 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

# Demo 4

# OpenLane vs. OpenROAD-flow-scripts

- 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

- 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

# Thank you for attending!