Recent Experiences with OpenROAD- and IHP130/Croc-based Courses

140 M.S. Students, 2 Courses
Using OpenROAD / OS EDA

Davit Markarian
UC San Diego

ECE 260C: VLSI Advanced Topics

- Following intro RTL2GDS course, using OpenROAD
- Enrollment – 120 Graduate Students

- Lectures on
  - Flow components – focused on OpenROAD's implementation
  - ML and research topics, DFM, ...
- 5 Labs covering flow, synthesis, placement, ML, and source code modification
  - See backup slides
- Final Project – SoC Implementation Optimization based on Croc
  - Teams of 1-3, selecting tasks from a pool of design optimizations
- IHP130-based – small, digestible PDK

- **See backup slides for more details**
  - Course goals/philosophy, novel lab packaging, final project details

OpenROAD: 4 Key Benefits for EDU

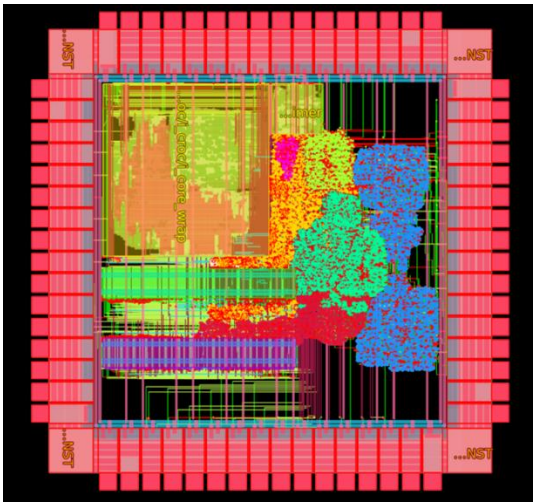
- Open-source: Look inside + Extend
  - Code links in lectures/Q&A + code exercises in labs
- Python: familiar scripting + comprehensive APIs
  - Students already know Python → hit the ground running on complex scripting tasks
- ML-ready
  - DB access with Python + permissive licensing
  - OpenROAD-flow-scripts as a dataset
- Concision
  - Smaller command set & fewer knobs → easier to comprehend

Challenges & Lessons

- Distribution = major challenge with OS EDA !
  - Containerization – Docker for maximum reproducibility
    - Can build on top with GitHub Codespaces, ...
  - Amplifying effect
    - Fix infrastructure early, write bulletproof guides
    - Balancing ease-of-use with opportunity for teaching
- OSS Contributions = hard to encourage
  - Students prefer to workaround issues rather than report them, even when incentivized
  - Analyzing crash reports is a teachable skill
- See backup slides for more
  - Course Design, PDK selection, ...

ECE 260C Final Project Example

- Team of 3 students implemented the following on top of Croc:
  - **Hierarchical floorplanning**, separating CPU Core, RF, and timer from top-level
  - **Design Space Exploration** for:
    - Aspect ratio
    - SRAM capacity switching
      - Deploying OpenROAD's automatic macro placer
  - **Retiming** with yosys/ABC



- Achieved Design/PPA improvements
  - This example can be downloaded from the BoF site

CSE 291: ML for Chip Design

Course Summary

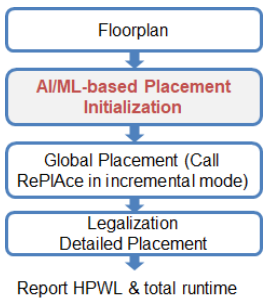
- **19 students across 8 teams** built ML projects using **OpenROAD**
  - Topic #1: AL/ML-Based Placement Initialization
  - Topic #2: LLM-powered Autonomous Agents

Skills Developed

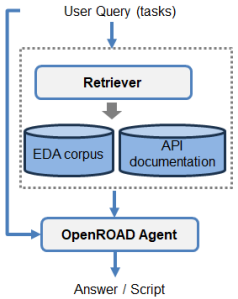
- Collected and labeled **data** to **support model training**
- Integrated **ML models** into full EDA flows
- Students executed projects from literature review onwards

To Improve Next Time …

- Enable **more flexible, student-friendly** customization of ORFS
- Provide more detailed **API documentation** with examples
- **Data, compute** to match course scope



Topic #1



Topic #2

Releases & Future Work

- Today: ECE 260C Labs
  - See “Labs Compass” backup slide
  - Links can be found on the BoF site.
- Today: ECE 260C Final Project Example
- Future: Adapting existing labs
  - Parallels between Proprietary EDA and OS EDA
    - Bringing in more tools, like NGSPICE/Xyce or KLayout
- Future: More turnkey tooling
  - Simplified packaging and delivery will open new opportunities – OpenROAD for short tutorials, seminars
  - Better autograding – automatic build/test for source editing labs, database/metrics comparison for grading, GenAI for grading student written answers, …

BACKUP

ECE 260C Course Goals

- Course Goals: Following on from RTL2GDS introductory course, we wanted to empower students to:
  - Dive into more advanced flow/CAD scripting
  - Get a deeper understanding of the tool algorithms/implementation
  - Read and modify OpenROAD sources
  - Experiment with ML methods
  - Apply these concepts to SoC Implementation
- For 120 students, we delivered:
  - 5 Labs
    - Based on IHP130. Delivered with GH Classroom, Docker, Codespaces, ...
    - Covering flow setup, synthesis/DSE, custom placement, Graph-based ML, and Source editing to implement custom modules/commands
  - Final Project – SoC Implementation Optimization based on Croc
    - Teams of 2/3 selecting from a set of 8 optimizations

On Building a VLSI / OS EDA Course

- Chance to bring together both VLSI Concepts and EDA SWE
  - Focus on flow knobs? EDA algorithms? PD methods?
    - OS EDA reduces friction between discussing algorithmic theory and discussing the behavior of the real tool
  - Fluid spectrum between VLSI labs and CAD programming assignments
    - Intersection of EE and CS: hard to assume knowledge
    - Open-endedness makes student skill/time commitment harder to estimate – we likely overcorrected for this.
- Direct paper/code references in slides/resources pages
  - Ability to answer student questions on OpenROAD's implementation with direct GitHub permalinks to source lines

ECE 260C Labs Compass

- Some of these labs will be available to download. See BoF site for links.

- Lab 0 – Intro to OpenROAD/OpenROAD-Flow-Scripts
  - Used in conjunction with the Software Setup Guide.
- Lab 1 – Design Space Exploration
  - Teaches synthesis/techmapping concepts through building a DSE script that uses synthesis results. Students finish by taking their optimal (by some PPA metric) design through ORFS.

- Cont. on next slide.

ECE 260C Lab Compass (cont.)

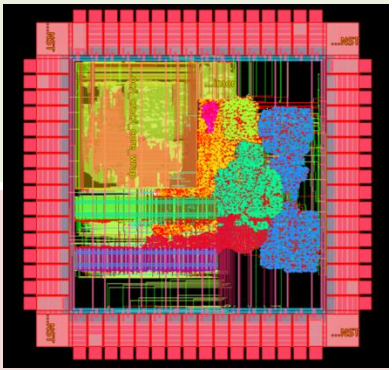
- Lab 2 – Scripting with the Database
  - Explores OpenROAD's Python Database APIs by having students make small DB queries and manipulations, culminating in them building a simple structured datapath placement script.
- Lab 3 – Machine Learning (not released)
  - Students use graph attention networks to compute the number of buffers in a given netlist, using net-level statistics like capacitance and slew.
- Lab 4 – Source Editing
  - Students write a custom module "ToySizer" for OpenROAD that implements ERC-compliance resizer functionality using DB and STA network queries

Adventures in Distributing Labs

- We tried different methods for packaging each lab.
  - Nearly all used GitHub Classroom as a way for us distribute PDK/databases and as a way for students to submit code
  - Labs 0/1/2 used standard Google Docs-based reports
    - Shared Docker Container providing OpenROAD/Yosys binaries
  - Lab 3 (not released today) – focused on Machine Learning – code and written questions in one notebook served in Google Colab
    - We parsed/processed these with nbformat library and GenAI
    - Prepared dataset for students and shared on HuggingFace
    - Dataset collection may be something for the student to explore in future
  - Lab 4 – source editing
    - Served with GitHub Classroom/Codespaces –"Open in Codespaces" for instant web-based source editing
    - Prepared a Docker container with cached OpenROAD sources
      - Pre-built once to allow incremental compiles
    - Students interacted with lab through special makefile
      - `make build` / `make test` for building/interacting with customized OpenROAD
      - `make turnin` to autogenerate a git patch containing student's customizations and commit/push to GitHub

Benefits of Croc & IHP130

- Final Project: Optimizing ETHZ's Croc SoC
  - Teams of 1-3 students, selecting 2/3 out of a pool of optimizations
    - Small design – faster iteration on students' computers
    - Flat floorplan with SRAM macros – opportunities for hierarchical design and floorplanning automation
    - Large unutilized "user" area allows more additions
      - SRAM sizing automation
      - Experimentation with arithmetic units
      - More IPs – some teams ported FPGA IPs



- IHP130 PDK – Already a part of ORFS
  - Used in final project and all labs
  - Great for teaching – linking lines in DEF/LIB
  - Missing cells and missing multi-Vt – fewer project opportunities

ECE 260C Final Project Optimizations List

Students had the following optimizations to choose from:

1. Hierarchical Floorplanning
2. Design Parametrization & DSE
3. Core: Optimized Multipliers & Adders
4. Core: Optimized Datapaths using SDPs
5. Core: Optimized Datapaths using Repipelining – using ABC
6. Automated Test – using Fault
7. Formal Verification – using SymbiYosys
8. IP Adaptation & Verification

Teams of 1 or 2 students were required to execute **2** of these.
Teams of 3 students executed **3** of these.