ECE4810J SoC Design

Fall 2024

Lab #5 Advanced Explorations with the Automated ASIC Design Flow based on OpenROAD

Due: 11:59pm Nov. 25th, 2024

Logistics:

- This lab is a team exercise.
- Please use the discussion board on Piazza for Q&A.
- All reports and code (if available) MUST be submitted to the EDA platform.
- Internet usage is allowed and encouraged.
- Late penalty (10% per day) will be applied for this lab.

Contents

1 Overview

In this lab, you will get more familar with the ASIC design flow. The goals of this lab are to:

- Explore the influence of design constraints and technology nodes on the PPA.
- Learn about the macro placement flow in OpenROAD.
- Learn about setting up a new design with OpenROAD.

2 OpenROAD

NOTE: Please work and submit in the Lab5 section in the EDA Platform.

2.1 Exercise 1: Creating a Pareto Curve

Copy files in `/data/public/lab5_starter` to your workspace.

In ASIC design, the applied design constraints can hugely influence the design results. In this exercise, adjust the constraints on a design to observe the impact on power, performance, and area (PPA).

`../../exercise1/` provides a simple integer arithmetic logic unit (ALU). The default bitwidth is 12, and the default clock constraint is 7ns (~143 MHz). These parameters allow for RTL-to-GDS in under 1 minute. Run the design with:

```
make DESIGN_CONFIG=../../lab5_starter/exercise1/config.mk
```

Once complete, observe the final report at `logs/nangate45/alu/base/6_report.json` or `logs/nangate45/alu/base/6_report.log`.

Record the power, frequency, and area. Then, open the constraint file (`../../lab5_starter/exercise1/constraint.sdc`) with your favorite editor and adjust the clock period to 6ns.

Clean the design and rerun using the new constraint:

```
make DESIGN_CONFIG=../../lab5_starter/exercise1/config.mk clean_all
make DESIGN_CONFIG=../../lab5_starter/exercise1/config.mk
```

Once complete, you can plot this data using your favorite software (Google Sheets, Microsoft Excel, `matplotlib`, etc.). Use max frequency as the independent variable.

2.2 Exercise 2: Scaling a Design Across Technologies

Observe the differences when a design is implemented in different technologies.

OpenROAD-flow-scripts provides three open-source PDKs to implement designs in SkyWater 130nm, Nangate 45nm, and ASAP 7nm. RTL is easily portable across technologies if it does not contain technology-specific cells (such as I/O pads, SRAM, clock-gate cells, etc.).

The `../../lab5_starter/exercise2/` directory contains the same ALU design from Exercise 1. However, this time you will change the config to alter the target technology. Adjust the `PLATFORM` variable in `../../lab5_starter/exercise2/config.mk` to to one of the technologies (`sky130hd`, `nangate45`, `asap7`). Keep in mind that:

- You may need to clean the design data from Exercise 1, because the platform and design name (`nangate45/alu`) are reused:

  ```
  make DESIGN_CONFIG=../../lab5_starter/exercise1/config.mk clean_all
  ```

- The time units for `sky130hd` and `nangate45` are both in ns, but the units for `asap7` are in ps. In order to maintain parity, you will need to adjust `../../lab5_starter/exercise2/constraint.sdc`.

Then, run the design using:

```
make DESIGN_CONFIG=../../lab5_starter/exercise2/config.mk
```

Record the power, frequency, and area for each technology. You need not clean the design between runs because changing the platform changes the output directory. You can again graph the data using your favorite graphing software.

2.3 Exercise 3: Setting Up a New Design with OpenROAD-flow-Scripts

Use the provided `counter.v` file for this exercise. If you have your own RTL, now's the time to use it!

The directory `../../lab5_starter/exercise3/` contains a blank config.mk template and blank constraint.sdc template.

If you have Verilog RTL file(s), place it (them) in `../../lab5_starter/exercise3/`. If you don't have your own RTL, you can use the provided counter.v file instead. Feel free to use other designs as a reference! Use any platform you want.

When you are setting up a new design with the design flow, you may change your time constraint file to maximize the performance of your design. An easy way to ensure this is to iterate your time constraints several times so that the final slack reach exactly zero. This is also the most accurate way to evaluate the maximum frequency of the design. In Lab 4, you have already learn the formula for calculating masimum frequency. You can use it to guide your time constraints for the next interation until your worst slack is exactly zero.

For the submission, take notes and make analysis on the timing constraints and timing report for your iterations and final result.

2.4 Building Complex Designs

Follow along as the slides explain how to incorporate macros into your design.

For designs to scale to larger sizes, additional layers of abstraction are required. Macros are special cells that are not logic gates and aren't automatically generated from synthesis. Macros are often much larger than standard cells and therefore require special handling. Macros are often used for several reasons:

1. Using SRAM or register files for large memories
2. Encapsulating a module that is instantiated multiple times
3. I/O pad cells for off-chip power and communication
4. Fiducial cells required by the manufacturer for fabrication
5. Intellectual property (IP) provided by a third-party vendor
6. And more!

2.4.1 How can I generate macros?

- OpenRAM is an open-source SRAM generator
  - Requires bitcells and sense amplifiers; creates implementations suitable for fabrication
- bsg_fakeram is a blackbox SRAM generator
  - Creates a blackbox implementation which is useful for modeling; cannot be used for fabrication
- Generate a block using OpenROAD
  - Use OpenROAD to create a hardened macro, then instantiate the block in a parent module
- Acquire third-party IP
  - Many commercial vendors provide RAM generators, I/O pad cells, analog macros, and more

`nangate45/tinyRocket` is a CPU core which incorporates SRAM macros generated by `bsg_fakeram`. While OpenROAD-flow-scripts already includes platform files necessary for standard cells, designers must specify macro files in the design config.

`flow/designs/nangate45/tinyRocket/config.mk`:

```
export ADDITIONAL_LEFS = $(sort $(wildcard
↪  ./designs/$(PLATFORM)/$(DESIGN_NICKNAME)/*.lef))
export ADDITIONAL_LIBS = $(sort $(wildcard
↪  ./designs/$(PLATFORM)/$(DESIGN_NICKNAME)/*.lib))
```

The config file uses the variable `ADDITIONAL_LEFS` and `ADDITIONAL_LIBS` to reference the abstract physical views (`.lef`) and timing models (`.lib`) for the macros. The wildcard commands above are shorthand for:

```
export ADDITIONAL_LEFS =
↪  ./designs/nangate45/tinyRocket/fakeram45_1024x32.lef
↪  ./designs/nangate45/tinyRocket/fakeram45_64x32.lef
export ADDITIONAL_LIBS =
↪  ./designs/nangate45/tinyRocket/fakeram45_1024x32.lib
↪  ./designs/nangate45/tinyRocket/fakeram45_64x32.lib
```

Notice, however, that these RAMs are generated by `bsg_fakeram` and do not have physical implementation files (`.gds`). Normally, this would create an error during the GDS merge step, however the platform configuration for nangate45 downgrades this to a warning by setting `GDS_ALLOW_EMPTY` on these instances:

```
# Allow empty GDS cell
export GDS_ALLOW_EMPTY = fakeram.*
```

If the macro does have a physical implementation (`.gds`), it can be added to the design config with:

```
export ADDITIONAL_GDS = /path/to/macro1.gds /path/to/macro2.gds ...
```

Now, build tinyRocket with:

```
# Build takes several minutes
make DESIGN_CONFIG=./designs/nangate45/tinyRocket/config.mk
```

Once done, you can see that new steps in the flow were used:

1. `2_3_tdms_place.log`: timing-driven mixed-size place
2. `2_4_mplace.log`: macro place

`tdms_place` performs a rough initial placement of both macros and standard cells. This is used as a seed for the macro placer. `mplace` performs macro placement. The placer tries to ensure that macros block as little design area as possible while still allowing connectivity to the macro.

Common problems when introducing macros:

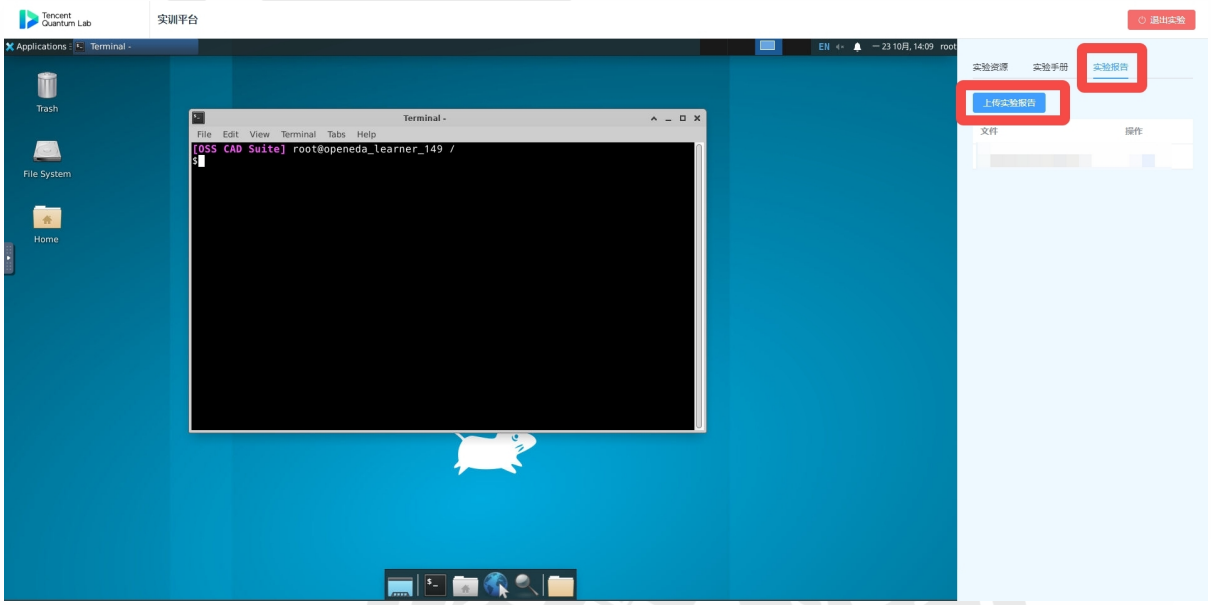
- "Channels" between macros need to be wide enough to not overcongest the router
- Slight changes to the design area can cause large changes in the macro placements
- Macros can block regions of the core area and make standard cell placement difficult
- Malformed macros can cause difficult-to-diagnose design problems

3 Deliverables

Please add comments or explanations to all the deliverables.

1. Section 2.1: In a single figure, the plot of the total power and area w.r.t max frequency or clock period for clock period = {3, 4, 5, 6, 7}.
2. Section 2.2: In a single figure, the plot of the total power and area w.r.t max frequency or clock period for each technology (`sky130hd`, `nangate45`, `asap7`).
3. Section 2.3: the `config.mk` file, the `constraint.sdc` file, the generated final GDS file, and the analysis document.

Note: We will try using the EDA platform for the submission. Please compress all the files into one zip file and upload it through "Experiment Report -> Upload Experiment Report" on your right hand side of the screen.



4 Grading policy

Factors	Percentage
Section 2.1	40%
Section 2.2	20%
Section 2.3	20%

A Peer Evaluation Form

Part	Your work	Your partner's work	Your score	Your partner's score
Section 2.1				
Section 2.2				
Section 2.3				

B Change Log

Fall 2023: Yichen Cai

- Break Lab 6 into Lab 4 and Lab 5
- updated Section 2.3

Fall 2022: Yihua Liu

- created this lab