Contributing Code

We'd love to accept your patches and contributions to this project. There are just a few small guidelines you need to follow.

Branching

For various reasons, it's recommended to call working branches, even in your forks, something else other than `master` or `main`, as those two branch names do have some special behavior associated with them.

Testing

Before you submit your changes, it's prudent to perform some kind of smoke test. `python3 -m openlane ./designs/spm/config.json` tests a simple spm design to ensure nothing has gone horribly wrong.

Language Standards

Python

Python code should be run on Python 3.8+, and be **typed**. i.e., we require explicit type annotations for all major API functions.

You will need to ensure that your Python code passes linting with our three chosen tools:

Tool	Kind	Command	Description
black	Formatter	`black .`	Ensures indentation and whitespace follow a strict standard without having you lift a finger.
flake8	Linter	`flake8 .`	Finds a number of common programming pitfalls.
mypy	Type-Checker	`mypy .`	Ensures that you're using compatible types, i.e., you are not passing a `string` to a function that accepts an `int`, or passing `None` to a non-optional variable, and such.

Do all arithmetic either in integers or using the Python `decimal` library. All (numerous) existing uses of IEEE-754 are bugs we are interested in fixing.

Tcl

Only use Tcl to interface with tools that only have a Tcl interface (or have an immature Python interface)- i.e., Yosys, OpenROAD and Magic.

1TBS-indented, four spaces, `lower_snake_case` for local/global variables and `UPPER_SNAKE_CASE` for environment variables. Unfortunately it is impossible to add any other guidelines or standards to the Tcl code considering it is Tcl code. Please exercise your best judgment.

Yosys, OpenROAD and Magic Scripts

There are some special guidelines for scripts in `scripts/yosys`, `scripts/openroad`, and `scripts/magic`:

- The scripts for each tool are a self-contained ecosystem: do not `source` scripts from outside their directories.
  - You may duplicate functionality if you deem it necessary.
- Do not reference the following environment variables anywhere in this folder to avoid causing recursion when generating issue reproducibles:
  - $PWD
  - $RUN_DIR
  - $DESIGN_DIR

Nix

Binary utilities used by OpenLane must utilize Nix derivations. We do have some conventions:

- All packages must accept a `pkgs` argument with a default value of `import ./pkgs.nix`.
  - https://nix.dev/tutorials/towards-reproducibility-pinning-nixpkgs
- All packages must use `fetchFromGitHub` with a commit-based `rev` and `sha256`, in addition to using `name` instead of `pname`.
  - We don't keep track of versions, only commits, so it doesn't matter. In other words, `version` should (in most cases) be `null`.
  - This will ultimately help us implement automatic tool update checks
- Packages must not enable tests.
  - OpenLane should be unit-testing everything it needs separately, and unlike the nixpkgs repo proper, we make no guarantees that the actual tools fully runs beyond what not OpenLane needs.
  - Therefore, running tests would essentially just be a waste of compute.

Submissions

Make your changes and then submit them as a pull requests to the `master` branch.

Consult GitHub Help for more information on using pull requests.

The Approval Process

For a PR to be merged, there are two requirements:

- There are two automated checks, one for linting and the other for functionality. Both must pass.
- An OpenLane team member must inspect and approve the PR.

Licensing and Copyright

Please add you (or your employer's) copyright headers to any files to which you have made major edits.

Please note all code contributions must have the same license as OpenLane, i.e., the Apache License, version 2.0.