Variable Migration Guides

These are migration guides for some particularly complex variables from OpenLane 1.

> **Note**
>
> In this document, OpenLane 1 refers to OpenLane versions before the release of 2.0.0, and OpenLane 2 refers to 2.0.0 or higher.

Migrating `FP_PDN_VOFFSET` / `FP_PDN_HOFFSET` / `FP_PDN_VPITCH` / `FP_PDN_HPITCH` for smaller designs

When working with smaller designs, OpenLane 1 used to scale down the PDN pitch/offset values by a factor of 4 so at least one set of PDN straps would be able to connect to the design.

This was a bit too magical and not entirely deterministic, and thus OpenLane 2 will no longer automatically do this scaling down for you. You will have to manually calculate the scale.

See Power Distribution Networks for more info.

Migrating `DIODE_INSERTION_STRATEGY`

As they were extremely complex, the OpenLane 1 diode insertion strategies were replaced by two flags in OpenLane 2's "Classic" flow:

- `GRT_REPAIR_ANTENNAS`: Attempts to repair antennas by enabling the step `OpenROAD.RepairAntennas` that invokes OpenROAD's `repair_antennas` function
- `RUN_HEURISTIC_DIODE_INSERTION`: Runs a custom script by Sylvain Munaut that inserts diodes based on a net's Manhattan distan

The mapping for the strategies is as follows:

- Strategies 1, 2 and 5 will throw an error.
  - 1 is an unreasonable brute force approach that adds a diode to every single net. 2 and 5 utilize replaceable “fake diode” fill cells, which are a hack that will not be available in all PDKs.
- Strategy 0:
  - Both flags are set to `false`.
- Strategy 3:
  - `GRT_REPAIR_ANTENNAS` is set to `true`.
- Strategy 4:
  - `RUN_HEURISTIC_DIODE_INSERTION` is set to `true`.
- Strategy 6:
  - Both flags are set to `true`.

Although for now OpenLane 2 will attempt the conversion for you automatically, it is recommended you update your designs as this feature will get removed.

Migrating `MACROS` (Optional, but highly recommended)

OpenLane 2+ allows you to configure macros you use in a configuration object called `MACROS`.

This object collects all information about macros and instances of macros into one object.

For example, where an SPM macro in OpenLane 1 would be defined across multiple variables in this manner:

```
{
  "VERILOG_FILES_BLACKBOX": ["dir::macros/spm/nl/spm.nl.v"],
  "EXTRA_GDS_FILES": ["dir::macros/gds/spm.gds"],
  "EXTRA_LEFS": ["dir::macros/lef/spm.lef"],
  "EXTRA_SPEFS": "spm dir::macros/spm/spef/nom_/spm.min.spef dir::macros/spm/sp
  "MACRO_PLACEMENT_CFG": "dir::macro_placement.cf
}
```

This information and more is laid out in the `MACROS` object as follows:

```
{
  "MACROS": {
    "spm": {
      "instances": {
        "spm_inst_0": {
          "location": [10, 150],
          "orientation": "N"
        },
        "spm_inst_1": {
          "location": [150, 150],
          "orientation": "N"
        }
      },
      "gds": ["dir::macros/spm/gds/spm.magic.gds"],
      "lef": ["dir::macros/spm/lef/spm.lef"],
      "nl": ["dir::macros/spm/nl/spm.nl.v"],
      "spef": {
        "nom_*": "dir::macros/spm/spef/nom_/spm.nom.spef",
        "min_*": "dir::macros/spm/spef/min_/spm.min.spef",
        "max_*": "dir::macros/spm/spef/max_/spm.max.spef"
      }
    }
  }
}
```

You will notice a number of things:

- Instances and their locations are no longer declared in a separate file.
- Multiple SPEF files can be declared to match various IPVT corner, i.e.
  - `nom_*` will match `nom_tt_025C_1V80` and `nom_ff_n40C_1v95`
  - `min_*` will match `min_ss_100C_1v60`
  - etc
- The previous bullet point also applies to `.lib` files (if available)

This configuration object helps keep all data related to instantiated macros in the same place. The `EXTRA_` variables still exist and are loaded indiscriminately in all operations where the appropriate files are loaded, but the `MACROS` object give you a little bit more control.

Tip

During STA, a Macro's `.lib` view or a combination of its Verilog netlist and `.spef` view could be used to provide the necessary timing information.

By default, netlist and `.spef` are prioritized over the `.lib` files because the characterized `.lib` files produced by OpenSTA are not the most accurate, but you may override this behavior by setting `STA_MACRO_PRIORITIZE_NL` to `false`.

As the name implies, this is merely a priority and if one view is available, that one will always be used.