Using PDKs

Process Design Kits, or PDKs, are files provided by a foundry that enables chips to be implemented for a specific foundry process.

OpenLane supports multiple process design kits, so long that an OpenLane PDK configuration file is made for them.

Specifying your PDK

There are multiple ways to specify a PDK from the command-line interface, ordered by priority (from highest to lowest):

- Within the `config.json` file: Recommended only for designs that are compatible with exactly one PDK.
- The `--pdk` command-line flag
- The `PDK` environment variable
- The default PDK, i.e., the Skywater/Google 130nm PDK.

Note that the PDK provided must be a fully qualified PDK variant, i.e.

- `sky130` ❌
- `sky130A` ⭕

Standard Cell Libraries

Standard cells are small silicon patterns that are used to implement basic functionality throughout your design that have already been designed and analyzed for timing and other physical information. For example, the standard cell `sky130_fd_sc_hd__and1` implements a logical AND.

Each PDK comes bundled with collections of standard cells known as “standard cell libraries” or SCLs for short, some of which are foundry-provided and some of which are third-party. Each OpenLane PDK configuration specifies a default

standard cell library, so you don't have to worry about picking one- but if you want to anyway, ordered by priority (from highest to lowest):

- Within the `config.json` file
- The `--scl` command-line flag
- The `STD_CELL_LIBRARY` environment variable
- The default SCL as dictated by the OpenLane PDK configuration files.
  - For sky130, that's `sky130_fd_sc_hd`.

Building, location and downloading

Most open-source PDKs are not usable in their default formats. This is the reason we rely on Open PDKs to build the PDK, which will not only place it an OpenLane-friendly layout but also generate views of the PDK that are usable by the aforementioned tools.

Because of the extended build times, the two PDKs supported by Efabless, i.e., the Skywater/Google 130nm PDK and the GlobalFoundries/Google GF180MCU PDK, are built and cached using Volare, a version manager by the OpenLane team built on top of Open PDKs.

There are multiple variants of each PDK (reflecting different metal stack configurations):

- `sky130`
  - `sky130A`: The default. 5 metal layers and a local interconnect.
  - `sky130B`: Similar to `sky130A`, but also includes a Resistive RAM (ReRAM) layer between `metal1` and `metal2`. See here for more information. This affects timing as, for example, `via1` between `metal1` and `metal2` is twice as high.

Both variants are supported by OpenLane.

> **Tip**
>
> The variants are identical on the front end of line, and Efabless shuttles are “split-lot”, i.e., two identical wafers come out of FEOL are then subjected to different back end of line processes, one for `A` and one for `B`.
>
> Designs targeting a specific variant are packaged only from the appropriate wafer, with its “twin” on the other wafer, which has timing issues at best and is a dead chip at worst, is discarded.
>
> All this to say: you can use either variant on any Efabless `sky130` shuttle (but not within the same design.)

- `gf180mcu`
  - `gf180mcuA`: 3 metal layers.
  - `gf180mcuB`: 4 metal layers.
  - `gf180mcuC`: 5 metal layers.
  - `gf180mcuD`: 5 metal layers, with a slightly thicker top metal layer. The thinner top metal layer has issues with delamination on the pads according to GlobalFoundries.

GFMPW0 was run on `C`, but any and all future shuttles by Efabless will require `gf180mcuD` as per GlobalFoundry’s recommendation. `A` and `B` were never used and are not supported by OpenLane.

OpenLane is tested with a specific revision of Open PDKs. By default, OpenLane will attempt to download a build done with that revision using Volare to the **PDK root**.

The PDK Root

The PDK root is a directory containing all of your open-source PDKs. OpenLane is configured to use the PDK root at the following options (from highest to lowest priority)”

- The `--pdk-root` command-line flag
- The `PDK_ROOT` environment variable
- A folder named `.volare` in your home directory.

The PDK root stores downloaded versions of the PDK as well as the Volare metadata information.

Each PDK family, e.g., `sky130` or `gf180mcu`, has one active version at a time. Versions of the PDKs are exactly equal to the `open_pdks` revision they were built with.

> **Tip**
>
> PDKs range from 700MiB to 1.5GiB, and they add up *fast*.
>
> You can delete all non-active PDKs by invoking `volare prune` in your terminal.

Using non-Volare PDKs

If you want to use PDKs without Volare (built manually), you will need to pass this flag:

`--manual-pdk`

This will bypassing checking and attempting to download the PDK using Volare, so you can add custom PDKs to the PDK root.

The PDKs must provide the following:

- An OpenLane PDK configuration file at the path `{pdk_root}/{pdk}/libs.tech/openlane/config.tcl`.

This Tcl-based (sorry) config file must include values for all non-optional variables in the PDK-Level.

- An standard cell library configuration file at the path `{pdk_root}/{pdk}/libs.tech/{scl}/openlane/config.tcl`.

This Tcl-based config file must include values for all non-optional variables in the SCL-Level.

The PDK must also include any step-specific PDK variables for the `Classic` flow to be compatible with the `Classic` flow.

...it may be prudent to just copy these files from `sky130A` and use that as a starting point, replacing files and values as you go.

Made with Sphinx and @pradyunsg's Furo