`openlane.config` API

The Configuration Module

This modules includes various functions for importing and/or generating OpenLane configuration objects. Configuration objects are the primary input to a flow.

```
class openlane.config.Instance(location: Tuple[Decimal, Decimal] | None,
    orientation: Orientation | None)
```

Bases: `object`

Location information for an instance of a Macro.

PARAMETERS:

- **location** (*Tuple[Decimal, Decimal] | None*) – The physical co-ordinates of the Macro's origin. Leave empty for automatic placement.
- **orientation** (*Orientation | None*) – The orientation of the macro's placement. 'N'/'R0' by default.

```
class openlane.config.Macro(gds:
    ~typing.List[~openlane.common.types.Path], lef:
    ~typing.List[~openlane.common.types.Path], instances:
    ~typing.Dict[str, ~openlane.config.variable.Instance] = <factory>,
    vh: ~typing.List[~openlane.common.types.Path] = <factory>, nl:
    ~typing.List[~openlane.common.types.Path] = <factory>, pnl:
    ~typing.List[~openlane.common.types.Path] = <factory>, spef:
    ~typing.Dict[str, ~typing.List[~openlane.common.types.Path]] =
    <factory>, lib: ~typing.Dict[str,
    ~typing.List[~openlane.common.types.Path]] = <factory>, spice:
    ~typing.List[~openlane.common.types.Path] = <factory>, sdf:
    ~typing.Dict[str, ~typing.List[~openlane.common.types.Path]] =
    <factory>, json_h: ~openlane.common.types.Path | None = None)
```

Bases: `object`

A data structure for storing definitions of Macro

latest

As it is typically stored in a dictionary in its supe
not stored in any of these fields.

You will note most fields correspond to a `openlane.state.DesignFormat` entry

Read the Docs latest

IDs. This is not coincidental.

PARAMETERS:

- **gds** (*List[Path]*) – A list of GDSII files representing the design. At least one is required.
- **lef** (*List[Path]*) – A list of LEF files representing the design. At least one is required.
- **instances** (*Dict[str, Instance]*) –

  A dictionary of `Instance` objects representing the instances of said macro.

  The keys for the dictionaries are the names of the instances.
- **nl** (*List[Path]*) –

  A list of netlists constituting the design.

  The netlists must be valid Verilog netlists readable by tools such as OpenSTA.

  Can be empty, however SPEF-based hierarchical static timing analysis will be unavailable.
- **spef** (*Dict[str, List[Path]]*) –

  A dictionary of parasitics annotations for the various netlists of the Macro.

  The keys are wildcards for timing corners supported by a certain SPEF file.

  Can be empty, however SPEF-based hierarchical static timing analysis will be unavailable.
- **lib** (*Dict[str, List[Path]]*) –

  A dictionary of timing library files.

  The keys are wildcards for timing corners supported by a certain LIB file.

  If both SPEF and LIB views are empty, the design may be black-boxed during STA.
- **spice** (*List[Path]*) – A list of SPICE netlists cc latest
  be useful in some flows.
- **sdf** (*Dict[str, List[Path]]*) –

A dictionary of standard delay format files. May be useful in some flows.

The keys are wildcards for timing corners supported by a certain SPEF file.

- **json_h** (*Path* | *None*) – A JSON file as generated by Yosys. Helpful in some flows.
- **vh** (*List[Path]*)
- **pnl** (*List[Path]*)

```
class openlane.config.Variable(name: str, type: ~typing.Any, description:
    str, default: ~typing.Any | None = None, deprecated_names:
    ~typing.List[str | ~typing.Tuple[str, ~typing.Callable]] =
    <factory>, units: str | None = None, pdk: bool = False)
```

Bases: `object`

An object encapsulating metadata on an OpenLane configuration variable, which is used to name, document and validate values supplied to `openlane.steps.Step`s or `openlane.flows.Flow`s.

latest

Values supplied for configuration variables are the primary interface by which users configure OpenLane flows.

PARAMETERS:

- **name** (*str*) – A string name for the Variable. Because of backwards compatibility with OpenLane 1, the convention is `UPPER_SNAKE_CASE`.
- **type** (*Any*) –

  A Python type object representing the variable.

  Supported scalars:

  - `int`
  - `decimal.Decimal`
  - `bool`
  - `str`
  - `Path`

  Supported products:

  - `Union` (incl. `Optional`)
  - `List`
  - `Tuple`
  - `Dict`
  - `Enum`

  Other:

  - `dataclass` types composed of the above.
- **description** (*str*) – A human-readable description of the variable. Used to generate help strings and documentation.
- **default** (*Any*) –

  A default value for the variable.

  Optional variables have an implicit default value of `None`.
- **deprecated_names** (*List[str | Tuple[str, Callable]]*) –

  A list of deprecated names for said variable.

  An element of the list can alternative be a tuple of a name and a Callable used to perform a translation for w
  also slightly modified.

latest

- **units** (*str* | *None*) – Used only in documentation: the unit corresponding to this object, i.e., μm, pF, etc. Can be any string, but for consistency, SI units must be represented in terms of their official symbols.
- **pdk** (*bool*) –

  Whether this variable is expected to be given a default value by a PDK or not.

  If this is true, and the variable is not of an option type, a PDK *must* give this variable a default value in order to be marked compatible with a step.

  If this is true and the variable is of an option type, a PDK may optionally provide a default value for this variable, however steps must presume it is `null`.

  If this is false, a PDK is not allowed to set a default value for this variable. In current versions of OpenLane, the value will be silently ignored, but warnings or errors may occur in future versions.

`property optional: bool`

RETURNS:

Whether a variable's type is an Option type.

`property some: Any`

RETURNS:

The type of a variable presuming it is not None.

If a variable is not Optional, that is simply the type specified in the `type` attribute.

`type_repr_md(for_document: bool = False) → str`

PARAMETERS:

for_document (*bool*) – Adds HTML line breaks between sum type separators for easier wrapping by web browsers/PDF renderers/what have you

RETURNS:

A pretty Markdown string representati latest

RETURN TYPE:

str

`desc_repr_md() → str`

RETURNS:

The description, but with newlines escaped for Markdown.

RETURN TYPE:

str

```
class openlane.config.Meta(version: int = 1, flow: None | str | List[str]
    = None, substituting_steps: None | Dict[str, str | None] = None,
    step: str | None = None, openlane_version: None | str = '2.1.3')
```

Bases: `object`

Constitutes metadata for a configuration object.

PARAMETERS:

- **version** (*int*)
- **flow** (*None* | *str* | *List[str]*)
- **substituting_steps** (*None* | *Dict[str, str* | *None]*)
- **step** (*None* | *str*)
- **openlane_version** (*None* | *str*)

```
class openlane.config.Config(*args, meta: Meta | None = None, **kwargs)
```

Bases: `GenericImmutableDict`[`str`, `Any`]

A map from OpenLane configuration variable keys to their values.

latest

It is recommended that you use `load()` to create new, validated configurations from dictionaries or files.

PARAMETERS:

- **meta** (*Meta*) – The `Meta` object for this configuration. If `None` is passed, the default Meta object will be assigned.
- **final** –

  Whether the configuration is final (i.e. has been pre-assembled for an entire flow) or may be incremented per-step.

  Final configurations may not be adjusted or incremented.

`copy(**overrides) → Config`

Produces a *shallow* copy of the configuration object.

PARAMETERS:

overrides – A series of configuration overrides as key-value pairs. These values are NOT validated and you should not be overriding these haphazardly.

RETURN TYPE:

Config

`to_raw_dict(include_meta: bool = True) → Dict[str, Any]`

PARAMETERS:

include_meta (*bool*) – Whether to include the “meta” object or not

RETURNS:

A raw dictionary representation including the `meta` object.

RETURN TYPE:

Dict[str, *Any*]

`dumps(include_meta: bool = True, **kwargs) → str`

PARAMETERS:

- **include_meta** (*bool*) – Whether to include the `meta` object in the serialized string.
- **kwargs** – Passed to `json.dumps`.

latest

RETURNS:

A JSON string representing the the GenericDict object.

RETURN TYPE:

str

`copy_filtered(config_vars: Sequence[Variable], include_flow_variables: bool = True) → Config`

Creates a new copy of the configuration object, but only with the configuration variables defined by the parameter.

PARAMETERS:

- **config_vars** (*Sequence[Variable]*) – A list of configuration variables to include in the filtered copy.
- **include_flow_variables** (*bool*) –

  Whether to include the common flow variables in the copy or not.

  This parameter is deprecated as of OpenLane 2.0.0b5 and should be set to `False` by callers.

RETURNS:

The new copy

RETURN TYPE:

Config

`with_increment(config_vars: Sequence[Variable], other_inputs: Mapping[str, Any], config_quiet: bool = False) → Config`

Creates a new `Config` object by copying all values from the original in addition to any new variables (and removing any variables not in *config_vars*).

Furthermore, inputs can be provided incrementally by passing the object `other_inputs`, which will also use these as overrides to the values in the base `Config` object.

latest

All values, including those in the base `Config` object and in `other_inputs`, will be re-validated.

PARAMETERS:

- **config_vars** (*Sequence[Variable]*) – A list of configuration variables to include and validate.
- **other_inputs** (*Mapping[str, Any]*) – A mapping of other inputs.
- **config_quiet** (*bool*)

RETURNS:

The new `Config` object

RETURN TYPE:

Config

```
classmethod get_meta(config_in: str | PathLike | Mapping[str, Any],
    flow_override: str | None = None) → Meta
```

Returns the Meta object of a configuration dictionary or file.

PARAMETERS:

- **config_in** (*str* | *PathLike* | *Mapping[str, Any]*) – A configuration object or file.
- **flow_override** (*str* | *None*)

RETURNS:

Either a Meta object, or if the file is invalid, None.

RETURN TYPE:

Meta

```
classmethod interactive(DESIGN_NAME: str, PDK: str, STD_CELL_LIBRARY:
    str | None = None, PDK_ROOT: str | None = None, **kwargs) →
    Config
```

This constructs a partial configuration object that may be incrementally adjusted per-step, and activates OpenLane's **interactive mode**.

The interactive mode is overall less rigid than the pure mode, adding various references to global objects to make the REPL or Notebook experience more pleasant, however, it is not as resilient as the pure mode and should not be used in production code.

PARAMETERS:

- **DESIGN_NAME** (*str*) – The name of the design to be used.
- **PDK** (*str*) – The name of the PDK.
- **STD_CELL_LIBRARY** (*str* | *None*) –

  The name of the standard cell library.

  If not specified, the PDK's default SCL will be used.
- **PDK_ROOT** (*str* | *None*) –

  Required if Volare is not installed.

  If Volare is installed, this value can be used to optionally override Volare's default.
- **kwargs** –

  Any overrides to PDK values and/or common flow default variables can be passed as keyword arguments to this function.

  Useful examples are CLOCK_PORT, CLOCK_PERIOD, et cetera, which while not bound to a specific `Step`, affects most Steps' behavior.

RETURN TYPE:

Config

```
classmethod load(config_in: str | PathLike | Mapping[str, Any] |
    Sequence[str | PathLike | Mapping[str, Any]], flow_config_vars:
    Sequence[Variable], *, config_override_strings: Sequence[str] |
    None = None, pdk: str | None = None, pdk_root: str | None =
    None, scl: str | None = None, design_dir: str | None = None,
    _load_pdk_configs: bool = True) → Tuple[Config, str]
```

Creates a new Config object based on a Tcl file, a JSON file, or a dictionary.

The returned config object is locked and cannot be modified.

PARAMETERS:

- **config_in** (*str* | *PathLike* | *Mapping[str, Any]* | *Sequence[str* | *PathLike* | *Mapping[str, Any]]*) –

  Either a file path to a JSON file or a Python Mapping object (such as `dict`) representing an unprocessed OpenLane configuration object.

  Tcl files are also supported, but are deprecated and will be removed in the future.

- **config_override_strings** (*Sequence[str]* | *None*) – A list of “overrides” in the form of NAME=VALUE strings. These are primarily for running OpenLane from the command-line and strictly speaking should not be used in the API.
- **design_dir** (*str* | *None*) –

  The design directory for said configuration(s).

  If not explicitly provided, the design directory will be the directory holding the last file in the list.

  If no files are provided, this argument is required.

- **pdk** (*str* | *None*) – A process design kit to use. Required unless specified via the “PDK” key in a configuration object.
- **pdk_root** (*str* | *None*) –

  Required if Volare is not installed.

  If Volare is installed, this value can be used to optionally override Volare’s default.

- **scl** (*str* | *None*) – A standard cell library to use. If not specified, the PDK’s default standard cell library will be used instead.
- **flow_config_vars** (*Sequence[Variable]*)
- **_load_pdk_configs** (*bool*)

RETURNS:

A tuple containing a Config object and

latest

RETURN TYPE:

Tuple[*Config*, str]

exception openlane.config.InvalidConfig(config: str, warnings: List[str], errors: List[str], message: str | None = None, *args, **kwargs)

Bases: `ValueError`

An error raised when a configuration under resolution is invalid.

PARAMETERS:

- **config** (*str*) – A human-readable name for the particular configuration file causing this exception, i.e. whether it's a PDK configuration file or a user configuration file.
- **warnings** (*List[str]*) – A list of warnings generated during the loading of this configuration file.
- **errors** (*List[str]*) – A list of errors generated during the loading of this configuration file.
- **args** – Further arguments to be passed onto the constructor of `ValueError`.
- **message** (*str* | *None*) – An optional override for the Exception message.
- **kwargs** – Further keyword arguments to be passed onto the constructor of `ValueError`.

RETURN TYPE:

None

exception openlane.config.PassedDirectoryError(config: str | PathLike)

Bases: `ValueError`

When a passed configuration file is in fact a directory.

PARAMETERS:

config (*str* | *PathLike*)

RETURN TYPE:

None

exception openlane.config.UnknownExtensionError(config: str | PathLike)

Bases: `ValueError`

latest

When a passed configuration file has an unrecognized extension, i.e., not .json or .tcl.

PARAMETERS:

config (*str* | *PathLike*)

RETURN TYPE:

None