`openlane.state` API

The State Module

This module manages the State of a Design before and after the execution of an OpenLane step. The State is essentially a list of views in various formats in addition to the cumulative set of metrics created by previous Steps.

`class openlane.state.DesignFormat(value)`

Bases: `Enum`

An enumeration of a number of `openlane.state.DesignFormatObject`s representing the various possible text or binary representations (views) supported by OpenLane states.

Members of this enumeration are used as the keys of `openlane.state.State` objects.

`class openlane.state.DesignFormatObject(id: str, extension: str, name: str, folder_override: str | None = None, multiple: bool = False)`

Bases: `object`

Metadata about the various possible text or binary representations (views) of any design.

Read the Docs latest

For example, `DesignFormat.NETLIST.value` has the metadata for Netlist views.

PARAMETERS:

- **id** (*str*) – A lowercase alphanumeric identifier for the design format. Some IDs in OpenLane 2.X use dashes. This is an inconsistency that will be addressed in the next major version of OpenLane as it would be a breaking change.
- **extension** (*str*) – The file extension for designs saved in this format.
- **name** (*str*) – A human-readable name for this design format.
- **folder_override** (*str* | *None*) – The subdirectory when `openlane.state.State.save_snapshot()` is called on a state. If unset, the value for `id` will be used.
- **multiple** (*bool*) – Whether this view may have multiple files (typically, files that are different across multiple corners or similar.)

```
class openlane.state.State(copying: Mapping[str, Path | List[Path] |
    Dict[str, Path | List[Path]] | None] | Mapping[DesignFormat, Path |
    List[Path] | Dict[str, Path | List[Path]] | None] | None = None,
    *args, overrides: Mapping[str, Path | List[Path] | Dict[str, Path |
    List[Path]] | None] | Mapping[DesignFormat, Path | List[Path] |
    Dict[str, Path | List[Path]] | None] | None = None, metrics:
    Mapping[str, Any] | None = None, **kwargs)
```

Bases: `GenericImmutableDict`[`str`, `Path` | `List`[`Path`] | `Dict`[`str`, `Path` | `List`[`Path`]] | `None`]

Basically, a dictionary from `DesignFormat`s and values of (nested dictionaries of) `openlane.common.Path`.

The state is the only thing that can be altered by steps other than the filesystem.

States are **immutable**. To construct a new state with some modifications, you may do so as follows:

```
state_b = State(
    copying=state_a,
    overrides={
        …
    },
    metrics=GenericImmutableDict(copying=state_a.metrics, overrides={
        …
    })
)
```

Though in the majority of cases, you do not have to construct States on your own: after executing a Step, you only return your deltas and then the Flow is responsible for the creation of a new Step object.

PARAMETERS:

- **copying** (*Optional[Union[Mapping[str, StateElement], Mapping[DesignFormat, StateElement]]]*) – A mutable or immutable mapping to use as the starting value for this State.
- **overrides** (*Optional[Union[Mapping[str, StateElement], Mapping[DesignFormat, StateElement]]]*) – A mutable or immutable mapping to override the starting values with.
- **metrics** (*Optional[Mapping[str, Any]]*) – A dictionary that carries statistics about the design: area, wire length, et cetera, but also miscellaneous data, for example, whether it passed a certain check or not.

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

RETURNS:

A copy of the underlying Python built-in `dict` for this class.

PARAMETERS:

metrics (*bool*)

RETURN TYPE:

Dict[str, *Any*]

`copy() → State`

Convenience replacement for *object.__class__(object)*, which would create a copy of the `GenericDict` object.

RETURNS:

The copy

PARAMETERS:

self (*State*)

RETURN TYPE:

State

`save_snapshot(path: str | PathLike)`

Validates the current state then saves all views to a folder by design format, including the metrics.

PARAMETERS:

path (*str* | *PathLike*) – The folder that would contain other folders.

`validate()`

Ensures that all paths exist in a State.

`exception openlane.state.InvalidState`

Bases: `RuntimeError`