Built-in Flows and their Configuration Variables

These flows come included with OpenLane. They use a variety of built-in steps to either provide a general RTL-to-GDSII flow or more specific niches.

Each Flow's list of configuration variables is essentially a sum of:

- Universal Flow Configuration Variables
- Universal Flow PDK Configuration Variables

AND

- Flow-specific Configuration Variables
- All included Step Configuration Variables

If you're looking for documentation for the `Flow` Python classes themselves, check the API reference here.

Flows

> **Tip**
>
> For a table of contents, press the following button on the top-right corner of the page:

Classic

Note: While OpenLane 2 has a stable release, the default flow is in beta pending silicon validation. Use at your own risk.

A flow of type `openlane.flows.SequentialFlow` that is the most similar to the original OpenLane 1.0 flow, running the Verilog RTL through Yosys, OpenROAD, KLayout and Magic to produce a valid GDSII for simpler designs.

This is the default when using OpenLane via the command-line.

Using from the CLI

```
openlane --flow Classic [...]
```

Importing

```
from openlane.flows import Flow

Classic = Flow.factory.get("Classic")
```

Flow-specific Configuration Variables

Variable Name	Type	Description	Default	Units
`RUN_TAP_ENDCAP_INSERTION`	bool	Enables the OpenROAD.TapEndcapInsertion step.	`True`	
`RUN_POST_GPL_DESIGN_REPAIR`	bool	Enables resizer design repair after global placement using the OpenROAD.RepairDesignPostGPL step.	`True`	
`RUN_POST_GRT_DESIGN_REPAIR`	bool	Enables resizer design repair after global placement using the OpenROAD.RepairDesignPostGPL step. This is experimental and may result in hangs and/or extended run times.	`False`	
`RUN_CTS`	bool	Enables clock tree synthesis using the OpenROAD.CTS step.	`True`	
`RUN_POST_CTS_RESIZER_TIMING`	bool	Enables resizer timing optimizations after clock tree synthesis using the OpenROAD.ResizerTimingPostCTS step.	`True`	
`RUN_POST_GRT_RESIZER_TIMING`	bool	Enables resizer timing optimizations after global routing using the OpenROAD.ResizerTimingPostGRT step. This is experimental and may result in hangs and/or extended run times.	`False`	
`RUN_HEURISTIC_DIODE_INSERTION`	bool	Enables the Odb.HeuristicDiodeInsertion step.	`False`	
`RUN_ANTENNA_REPAIR`	bool	Enables the OpenROAD.RepairAntennas step.	`True`	
`RUN_DRT`	bool	Enables the OpenROAD.DetailedRouting step.	`True`	
`RUN_FILL_INSERTION`	bool	Enables the OpenROAD.FillInsertion step.	`True`	
`RUN_MCSTA`	bool	Enables multi-corner static analysis using the OpenROAD.STAPostPNR step.	`True`	

Variable Name	Type	Description	Default	Units
`RUN_SPEF_EXTRACTION`	bool	Enables parasitics extraction using the OpenROAD.RCX step.	`True`	
`RUN_IRDROP_REPORT`	bool	Enables generation of an IR Drop report using the OpenROAD.IRDropReport step.	`True`	
`RUN_LVS`	bool	Enables the Netgen.LVS step.	`True`	
`RUN_MAGIC_STREAMOUT`	bool	Enables the Magic.StreamOut step to generate GDSII.	`True`	
`RUN_KLAYOUT_STREAMOUT`	bool	Enables the KLayout.StreamOut step to generate GDSII.	`True`	
`RUN_MAGIC_WRITE_LEF`	bool	Enables the Magic.WriteLEF step.	`True`	
`RUN_KLAYOUT_XOR`	bool	Enables running the KLayout.XOR step on the two GDSII files generated by Magic and Klayout. Stream-outs for both KLayout and Magic should have already run, and the PDK must support both signoff tools.	`True`	
`RUN_MAGIC_DRC`	bool	Enables the Magic.DRC step.	`True`	
`RUN_KLAYOUT_DRC`	bool	Enables the KLayout.DRC step.	`True`	
`RUN_EQY`	bool	Enables the Yosys.EQY step. Not valid for VHDLClassic.	`False`	
`RUN_LINTER`	bool	Enables the Verilator.Lint step and associated checker steps. Not valid for VHDLClassic.	`True`	

Included Steps

- `Verilator.Lint`
- `Checker.LintTimingConstructs`
- `Checker.LintErrors`
- `Checker.LintWarnings`
- `Yosys.JsonHeader`
- `Yosys.Synthesis`
- `Checker.YosysUnmappedCells`
- `Checker.YosysSynthChecks`
- `Checker.NetlistAssignStatements`
- `OpenROAD.CheckSDCFiles`
- `OpenROAD.CheckMacroInstances`
- `OpenROAD.STAPrePNR`
- `OpenROAD.Floorplan`
- `Odb.CheckMacroAntennaProperties`
- `Odb.SetPowerConnections`
- `Odb.ManualMacroPlacement`
- `OpenROAD.CutRows`
- `OpenROAD.TapEndcapInsertion`
- `Odb.AddPDNObstructions`
- `OpenROAD.GeneratePDN`
- `Odb.RemovePDNObstructions`
- `Odb.AddRoutingObstructions`
- `OpenROAD.GlobalPlacementSkipIO`
- `OpenROAD.IOPlacement`
- `Odb.CustomIOPlacement`
- `Odb.ApplyDEFTemplate`
- `OpenROAD.GlobalPlacement`
- `Odb.WriteVerilogHeader`
- `Checker.PowerGridViolations`
- `OpenROAD.STAMidPNR`
- `OpenROAD.RepairDesignPostGPL`
- `OpenROAD.DetailedPlacement`
- `OpenROAD.CTS`
- `OpenROAD.STAMidPNR-1`
- `OpenROAD.ResizerTimingPostCTS`
- `OpenROAD.STAMidPNR-2`

- `OpenROAD.GlobalRouting`
- `OpenROAD.CheckAntennas`
- `OpenROAD.RepairDesignPostGRT`
- `Odb.DiodesOnPorts`
- `Odb.HeuristicDiodeInsertion`
- `OpenROAD.RepairAntennas`
- `OpenROAD.ResizerTimingPostGRT`
- `OpenROAD.STAMidPNR-3`
- `OpenROAD.DetailedRouting`
- `Odb.RemoveRoutingObstructions`
- `OpenROAD.CheckAntennas-1`
- `Checker.TrDRC`
- `Odb.ReportDisconnectedPins`
- `Checker.DisconnectedPins`
- `Odb.ReportWireLength`
- `Checker.WireLength`
- `OpenROAD.FillInsertion`
- `OpenROAD.RCX`
- `OpenROAD.STAPostPNR`
- `OpenROAD.IRDropReport`
- `Magic.StreamOut`
- `KLayout.StreamOut`
- `Magic.WriteLEF`
- `Odb.CheckDesignAntennaProperties`
- `KLayout.XOR`
- `Checker.XOR`
- `Magic.DRC`
- `KLayout.DRC`
- `Checker.MagicDRC`
- `Checker.KLayoutDRC`
- `Magic.SpiceExtraction`
- `Checker.IllegalOverlap`
- `Netgen.LVS`
- `Checker.LVS`
- `Yosys.EQY`
- `Checker.SetupViolations`
- `Checker.HoldViolations`

- `Checker.MaxSlewViolations`
- `Checker.MaxCapViolations`
- `Misc.ReportManufacturability`

VHDLClassic

A variant of Classic that accepts VHDL files for Synthesis instead of Verilog files (and removes Verilog linting/equivalence steps.)

Using from the CLI

```
openlane --flow VHDLClassic [...]
```

Importing

```
from openlane.flows import Flow

VHDLClassic = Flow.factory.get("VHDLClassic")
```

Flow-specific Configuration Variables

Variable Name	Type	Description	Default	Units
`RUN_TAP_ENDCAP_INSERTION`	bool	Enables the OpenROAD.TapEndcapInsertion step.	`True`	
`RUN_POST_GPL_DESIGN_REPAIR`	bool	Enables resizer design repair after global placement using the OpenROAD.RepairDesignPostGPL step.	`True`	
`RUN_POST_GRT_DESIGN_REPAIR`	bool	Enables resizer design repair after global placement using the OpenROAD.RepairDesignPostGPL step. This is experimental and may result in hangs and/or extended run times.	`False`	
`RUN_CTS`	bool	Enables clock tree synthesis using the OpenROAD.CTS step.	`True`	
`RUN_POST_CTS_RESIZER_TIMING`	bool	Enables resizer timing optimizations after clock tree synthesis using the OpenROAD.ResizerTimingPostCTS step.	`True`	
`RUN_POST_GRT_RESIZER_TIMING`	bool	Enables resizer timing optimizations after global routing using the OpenROAD.ResizerTimingPostGRT step. This is experimental and may result in hangs and/or extended run times.	`False`	
`RUN_HEURISTIC_DIODE_INSERTION`	bool	Enables the Odb.HeuristicDiodeInsertion step.	`False`	
`RUN_ANTENNA_REPAIR`	bool	Enables the OpenROAD.RepairAntennas step.	`True`	
`RUN_DRT`	bool	Enables the OpenROAD.DetailedRouting step.	`True`	
`RUN_FILL_INSERTION`	bool	Enables the OpenROAD.FillInsertion step.	`True`	
`RUN_MCSTA`	bool	Enables multi-corner static analysis using the OpenROAD.STAPostPNR step.	`True`	

Variable Name	Type	Description	Default	Units
`RUN_SPEF_EXTRACTION`	bool	Enables parasitics extraction using the OpenROAD.RCX step.	`True`	
`RUN_IRDROP_REPORT`	bool	Enables generation of an IR Drop report using the OpenROAD.IRDropReport step.	`True`	
`RUN_LVS`	bool	Enables the Netgen.LVS step.	`True`	
`RUN_MAGIC_STREAMOUT`	bool	Enables the Magic.StreamOut step to generate GDSII.	`True`	
`RUN_KLAYOUT_STREAMOUT`	bool	Enables the KLayout.StreamOut step to generate GDSII.	`True`	
`RUN_MAGIC_WRITE_LEF`	bool	Enables the Magic.WriteLEF step.	`True`	
`RUN_KLAYOUT_XOR`	bool	Enables running the KLayout.XOR step on the two GDSII files generated by Magic and Klayout. Stream-outs for both KLayout and Magic should have already run, and the PDK must support both signoff tools.	`True`	
`RUN_MAGIC_DRC`	bool	Enables the Magic.DRC step.	`True`	
`RUN_KLAYOUT_DRC`	bool	Enables the KLayout.DRC step.	`True`	
`RUN_EQY`	bool	Enables the Yosys.EQY step. Not valid for VHDLClassic.	`False`	
`RUN_LINTER`	bool	Enables the Verilator.Lint step and associated checker steps. Not valid for VHDLClassic.	`True`	

Included Steps

- `Yosys.VHDLSynthesis`
- `Checker.YosysUnmappedCells`
- `Checker.YosysSynthChecks`
- `Checker.NetlistAssignStatements`
- `OpenROAD.CheckSDCFiles`
- `OpenROAD.CheckMacroInstances`
- `OpenROAD.STAPrePNR`
- `OpenROAD.Floorplan`
- `Odb.CheckMacroAntennaProperties`
- `Odb.ManualMacroPlacement`
- `OpenROAD.CutRows`
- `OpenROAD.TapEndcapInsertion`
- `Odb.AddPDNObstructions`
- `OpenROAD.GeneratePDN`
- `Odb.RemovePDNObstructions`
- `Odb.AddRoutingObstructions`
- `OpenROAD.GlobalPlacementSkipIO`
- `OpenROAD.IOPlacement`
- `Odb.CustomIOPlacement`
- `Odb.ApplyDEFTemplate`
- `OpenROAD.GlobalPlacement`
- `Checker.PowerGridViolations`
- `OpenROAD.STAMidPNR`
- `OpenROAD.RepairDesignPostGPL`
- `OpenROAD.DetailedPlacement`
- `OpenROAD.CTS`
- `OpenROAD.STAMidPNR-1`
- `OpenROAD.ResizerTimingPostCTS`
- `OpenROAD.STAMidPNR-2`
- `OpenROAD.GlobalRouting`
- `OpenROAD.CheckAntennas`
- `OpenROAD.RepairDesignPostGRT`
- `Odb.DiodesOnPorts`
- `Odb.HeuristicDiodeInsertion`
- `OpenROAD.RepairAntennas`
- `OpenROAD.ResizerTimingPostGRT`

- `OpenROAD.STAMidPNR-3`
- `OpenROAD.DetailedRouting`
- `Odb.RemoveRoutingObstructions`
- `OpenROAD.CheckAntennas-1`
- `Checker.TrDRC`
- `Odb.ReportDisconnectedPins`
- `Checker.DisconnectedPins`
- `Odb.ReportWireLength`
- `Checker.WireLength`
- `OpenROAD.FillInsertion`
- `OpenROAD.RCX`
- `OpenROAD.STAPostPNR`
- `OpenROAD.IRDropReport`
- `Magic.StreamOut`
- `KLayout.StreamOut`
- `Magic.WriteLEF`
- `Odb.CheckDesignAntennaProperties`
- `KLayout.XOR`
- `Checker.XOR`
- `Magic.DRC`
- `KLayout.DRC`
- `Checker.MagicDRC`
- `Checker.KLayoutDRC`
- `Magic.SpiceExtraction`
- `Checker.IllegalOverlap`
- `Netgen.LVS`
- `Checker.LVS`
- `Checker.SetupViolations`
- `Checker.HoldViolations`
- `Checker.MaxSlewViolations`
- `Checker.MaxCapViolations`
- `Misc.ReportManufacturability`

OpenInKLayout

This 'flow' actually just has one step that opens the LEF/DEF from the KLayout. Fancy that.

Intended for use with run tags that have already been run with another flow, i.e.:

```
openlane [...]
openlane --last-run --flow OpenInKLayout [...]
```

Using from the CLI

```
openlane --flow OpenInKLayout [...]
```

Importing

```
from openlane.flows import Flow

OpenInKLayout = Flow.factory.get("OpenInKLayout")
```

Included Steps

- `KLayout.OpenGUI`

OpenInOpenROAD

This 'flow' actually just has one step that opens the ODB from the initial state object in OpenROAD.

Intended for use with run tags that have already been run with another flow, i.e.

```
openlane [...]
openlane --last-run --flow OpenInOpenROAD [...]
```

Using from the CLI

```
openlane --flow OpenInOpenROAD [...]
```

Importing

```
from openlane.flows import Flow

OpenInOpenROAD = Flow.factory.get("OpenInOpenROAD")
```

Included Steps

- `OpenROAD.OpenGUI`

OpenInMagic

This 'flow' actually just has one step that opens the GDS or DEF from the initial state object in Magic.

Intended for use with run tags that have already been run with another flow, i.e.

```
openlane [...]
openlane --last-run --flow OpenInMagic [...]
```

Using from the CLI

```
openlane --flow OpenInMagic [...]
```

Importing

```
from openlane.flows import Flow

OpenInMagic = Flow.factory.get("OpenInMagic")
```

Included Steps

- `Magic.OpenGUI`

SynthesisExploration

Synthesis Exploration is a feature that tries multiple synthesis strategies (in the form of different scripts for the ABC utility) to try and find which strategy is better by either minimizing area or maximizing slack (and thus frequency.)

The output is represented in a tabulated format, e.g.:

```
┏━━━━━━━━━━━━━━━━┳━━━━━━━┳━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━┓
┃                ┃       ┃              ┃ Worst Setup       ┃ Total Negative    ┃
┃ SYNTH_STRATEGY ┃ Gates ┃ Area (µm²)   ┃ Slack (ns)        ┃ Setup Slack (ns)  ┃
┡━━━━━━━━━━━━━━━━╇━━━━━━━╇━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━┩
│ AREA 0         │ 8781  │ 97141.916800 │ 6.288794          │ 0.0               │
│ AREA 1         │ 8692  │ 96447.500800 │ 6.434102          │ 0.0               │
│ AREA 2         │ 8681  │ 96339.897600 │ 6.276806          │ 0.0               │
│ AREA 3         │ 11793 │ 111084.038400│ 7.011374          │ 0.0               │
│ DELAY 0        │ 8969  │ 101418.518400│ 6.511191          │ 0.0               │
│ DELAY 1        │ 8997  │ 101275.881600│ 6.656564          │ 0.0               │
│ DELAY 2        │ 9013  │ 101177.036800│ 6.691765          │ 0.0               │
│ DELAY 3        │ 8733  │ 99190.131200 │ 6.414865          │ 0.0               │
│ DELAY 4        │ 8739  │ 101011.878400│ 6.274565          │ 0.0               │
└────────────────┴───────┴──────────────┴───────────────────┴───────────────────┘
```

You can then update your config file with the best `SYNTH_STRATEGY` for your use-case so it can be used with other flows.

Using from the CLI

```
openlane --flow SynthesisExploration [...]
```

Importing

```
from openlane.flows import Flow

SynthesisExploration = Flow.factory.get("SynthesisExploration")
```

Included Steps

- `Yosys.Synthesis`
- `OpenROAD.CheckSDCFiles`
- `OpenROAD.STAPrePNR`
