

Gate-Level to GDSII with OpenLane

The usual openlane-flow is RTL to GDSII, but we have no RTL file in this example. Instead, we want to skip Synthesis and convert a **Gate-Level-Design to GDSII** without further optimization and altering of cells.

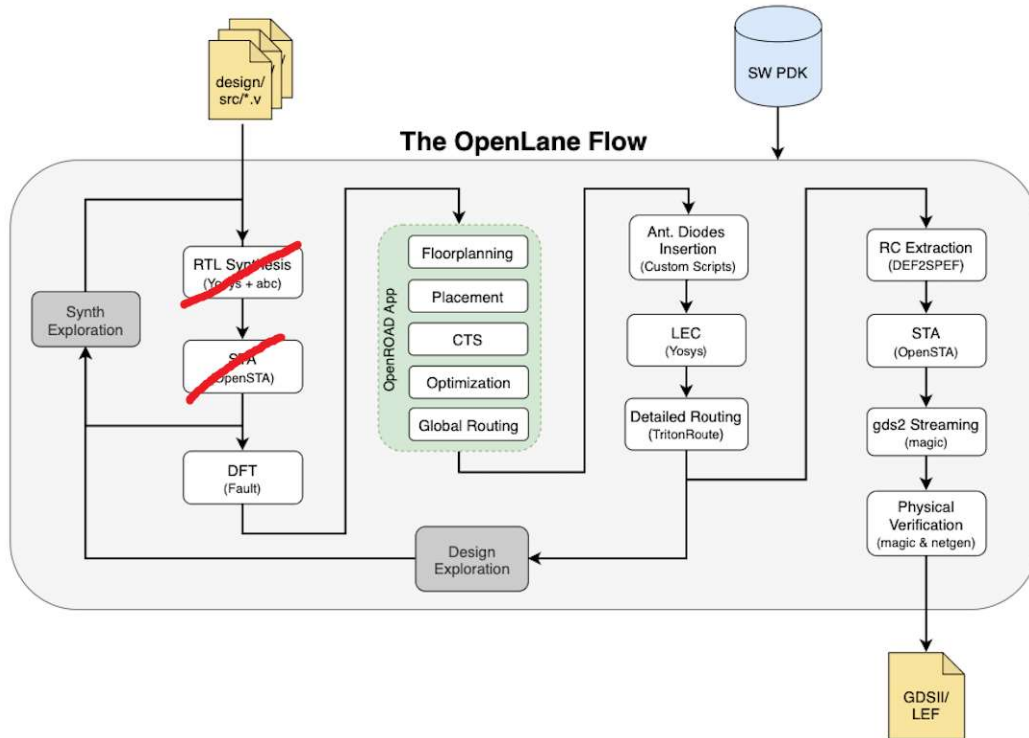


Image from <https://github.com/The-OpenROAD-Project/OpenLane/raw/master/docs/static/openlane.flow.1.png>

To achieve this, we must skip RTL-synthesis manually and prevent the tool from altering our cells. This is a possible workflow:

New openlane designs are usually initialized with `./flow.tcl -init-design-config`, but this does not work in the docker environment. Instead, copy one of the design templates

- from `/foss/tools/openlane/2022.07/designs/<designname>`
- to `/designs/<project-name>/openlane/<yournewdesignname>`

Write your Verilog HDL-file in the format as shown in the image below. You can find names for the standard cells in <https://antmicro-skywater-pdk-docs.readthedocs.io> and the PDK Verilog-library at `/foss/pdk/sky130A/libs.ref/sky130_fd_sc_hd/verilog/sky130_fd_sc_hd.v`

```

module mymodulename
(
    output wire ena_out,
    input wire ena_in,
);
wire net1 ;

sky130_fd_sc_hd__inv_1 x6 (
    .A( net1 ),
    .Y( ena_in )
);

sky130_fd_sc_hd__clkdybuf4s50_2 x1600 (
    .A( ena_out ),
    .X( net1 )
);

endmodule

```

Change the config /designs/<project-name>/openlane/<yourdesignname>/config.tcl

- Links to your Verilog-file
- Name of your project
- Prevent the router and placer from altering your cells by specifying
 - set ::env(PL_RESIZER_TIMING_OPTIMIZATIONS) {0}
 - set ::env(PL_RESIZER_DESIGN_OPTIMIZATIONS) {0}
 - set ::env(GLB_RESIZER_TIMING_OPTIMIZATIONS) {0}
- Optional: set your pin order and fanout orientations with
 - set ::env(FP_PIN_ORDER_CFG) \$::env(DSIGN_DIR)/pin_order.cfg

start a terminal at /designs/<project-name>/openlane/

run **flow-tcl -interactive -design <yourdesignname> -tag <run-name>**

type **package require openlane**, console should return **0.9**

Before you start synthesis, copy your already synthesized file into

../openlane/<yourdesignname>/runs/<run-name>/results/synthesis

Type **run_synthesis** in the console, synthesizer should recognize there is already a file (which is exactly what we want)

```

% run_synthesis
[STEP 1]
[INFO]: Running Synthesis...
[WARNING]: A netlist at /foss/designs/SKY130_SAR-ADC/openlane/yourdesignname/runs/run-name/results/synthesis/uit.v already exists. Synthesis will be skipped.
[STEP 2]
[INFO]: Running Single-Corner Static Timing Analysis...

```

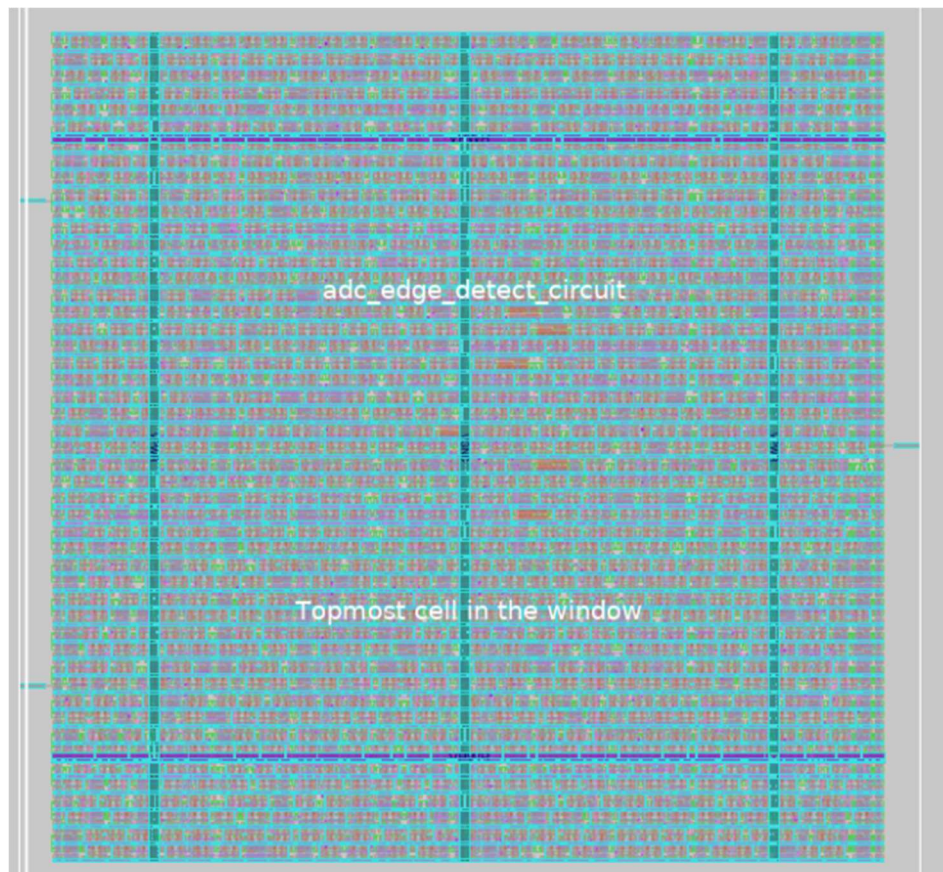
Type **run_floorplan**

Type **run_placement**

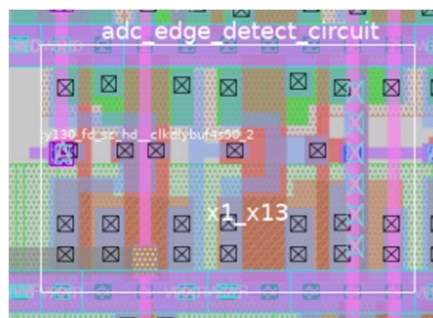
Type **run_routing**

Type **run_magic**

Result in runs/<run-name>/results/signoff/<project-name>.mag



...which is using the exact cell-type specified in our gate-level-design.



Appendix A: list of interactive openlane commands

https://github.com/The-OpenROAD-Project/OpenLane/blob/master/docs/source/advanced_readme.md

Interactive Mode

You may run the flow interactively by using the `-interactive` option:

```
./flow.tcl -interactive
```

A tcl shell will be opened where the openlane package is automatically sourced:

```
% package require openlane
```

Then, you should be able to run the following main commands:

0. Any tcl command.
1. `prep -design <design> -tag <tag> -config <config> -init_design_config -overwrite` similar to the command line arguments, design is required and the rest is optional
2. `run_synthesis`
3. `run_floorplan`
4. `run_placement`
5. `run_cts`
6. `run_routing`
7. `write_powered_verilog` followed by `set_netlist $::env(routing_logs)/$::env(DESIGN_NAME).powered.v`
8. `run_magic`
9. `run_magic_spice_export`
10. `run_magic_drc`
11. `run_lvs`
12. `run_antenna_check`

The above commands can also be written in a file and passed to `flow.tcl`:

```
./flow.tcl -interactive -file <file>
```

A more detailed list of all the commands supported by openlane could be found [here](#).

Note 1: Currently, configuration variables have higher priority over the above commands so if `RUN_MAGIC` is 0, command `run_magic` will have no effect.

Note 2: Currently, most of these commands must be run in the flow sequence and no steps should be skipped.

Note 3: You can pass the `-design`, `-tag`, etc. flags to `./flow.tcl -interactive` directly without the need of entering the interactive mode and then executing the `prep` command.

Appendix B: Configuration file of a CLKDLYBUF_4S50_2 combinatorial Delay-cell

Folder structure:

```
/foss/designs/SKY130_SAR-ADC
-/verilog
-/adc_edge_detect_circuit/ adc_edge_detect_circuit.v
-/openlane
-/adc_edge_detect_circuit/
-/runs/
-/config.tcl
-/pin_order.cfg
```

Contents of config.tcl:

```
set ::env(DSIGN_NAME) "adc_edge_detect_circuit"

set filename $::env(DSIGN_DIR)/$::env(PDK)_$::env(STD_CELL_LIBRARY)_config.tcl
if { [file exists $filename] == 1 } {
    source $filename
}

set ::env(VERILOG_FILES) "/foss/designs/SKY130_SAR-
ADC/verilog/adc_edge_detect_circuit/adc_edge_detect_circuit.v"

# set ::env(CLOCK_PERIOD) "10.000"
set ::env(CLOCK_PORT) "clk"
set ::env(CLOCK_NET) $::env(CLOCK_PORT)

# Floorplanning
set ::env(FP_SIZING) "relative"
set ::env(FP_ASPECT_RATIO) {1}
set ::env(FP_CORE_UTIL) {75}
set ::env(FP_PDN_VPITCH) {100}
set ::env(FP_PDN_HPITCH) {100}
set ::env(FP_PIN_ORDER_CFG) $::env(DSIGN_DIR)/pin_order.cfg

# Placement
set ::env(PL_TARGET_DENSITY) {0.80}
# set ::env(PL_TIME_DRIVEN) {0}
set ::env(PL_RESIZER_TIMING_OPTIMIZATIONS) {0}
set ::env(PL_RESIZER_DESIGN_OPTIMIZATIONS) {0}

#Clocktreesynthesis - unused
# set ::env(CLOCK_TREE_SYNTH) {0}
# set ::env(FILL_INSERTION) {0}

# Router
set ::env(GLB_RESIZER_TIMING_OPTIMIZATIONS) {0}

set ::env(CELL_PAD) {0}
```