

LIBRELANE FLOW

A practical guide to chip design flow

Daniel Arevalos

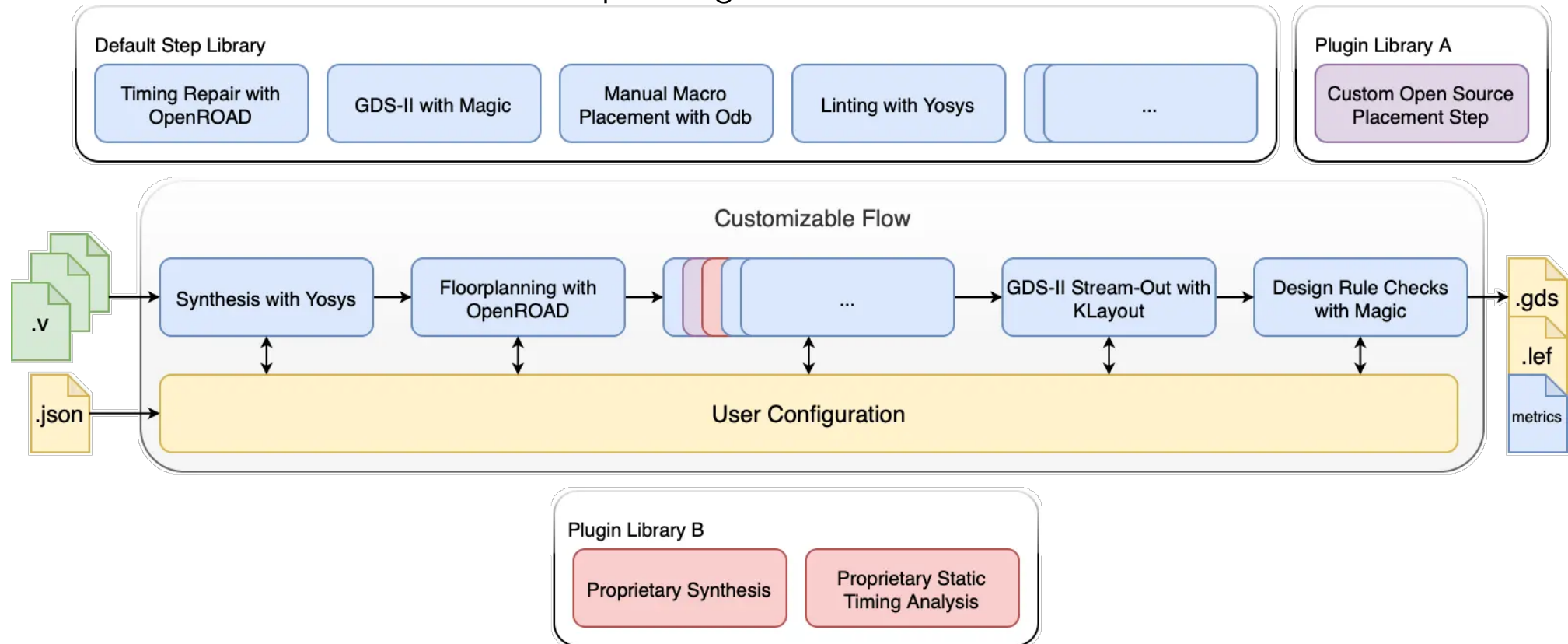
Fakultät 07 – Hochschule München

Table of Contents

- 1. What is Librelane?
- 2. Architectural Overview
- 3. Librelane Flow
 - 3.1 Config File
 - 3.2 Macros
 - 3.3 Classic vs Chip flow
 - 3.4 Steps
- 4. Quick tutorial
 - 4.1 Classic Flow
 - 4.2 Chip Flow

What is Librelane?

An open-source, configurable digital ASIC flow built on open-source EDA tools, designed for education and real-world chip design.



Architectural Overview

Encapsulates the state of the design as an **immutable dictionary** of paths to various design formats.

state
module

Encapsulates a configurable **transformation** on the State object.

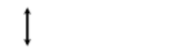
step
module

Encapsulates aggregations of steps for ease of configuration and ease-of-execution.

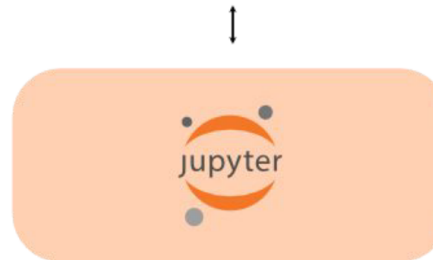
flow
module

Performs validation of various **variables** used to configure flows and steps, incl. type checking.

config
module



The LibreLane Python API



Config File

```
{  
  "DESIGN_NAME": "my_design",  
  "VERILOG_FILES": [  
    "src/my_design.v"  
  ],  
  "CLOCK_PORT": "clk",  
  "CLOCK_PERIOD": 10,  
  
  "FP_CORE_UTIL": 50,  
  "FP_ASPECT_RATIO": 1.0  
  
  "PL_TARGET_DENSITY": 0.55,  
}
```

More Information on: Config File

- ▶ **Design Definition:** Defines the logical identity of the design and specifies the RTL sources that will be used as input for the flow.
- ▶ **Clock Definition:** Describes the main clock interface and timing target that guide synthesis, timing analysis, and optimization throughout the flow.
- ▶ **Floorplanning (High-level):** Sets high-level constraints that shape the core area, controlling utilization and overall layout proportions.
- ▶ **Placement Control:** Guides how densely standard cells are packed during placement to balance area efficiency and routability.

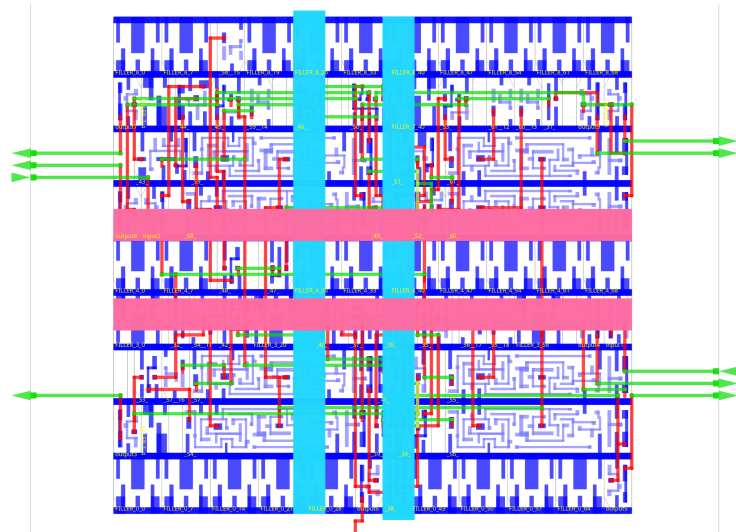
Macros

```
"MACROS": {  
  "your_module": {  
    "gds": "path/your_module.gds",  
    "lef": "path/your_module.lef",  
    "nl": "path/your_module.pnl.v",  
  
    "lib": {  
      "nom_typ_1p20V_25C": "path/your_module.lib",  
      "nom_fast_1p32V_m40C": "path/your_module.lib",  
      "nom_slow_1p08V_125C": "path/your_module.lib"  
    },  
  
    "spef": {  
      "nom_typ_1p20V_25C": "path/your_module.spef"  
    },  
  
    "instances": {  
      "inst_0": { "location": [640, 630], "orientation": "N" },  
      "inst_1": { "location": [440, 430], "orientation": "N" }  
    }  
  }  
}
```

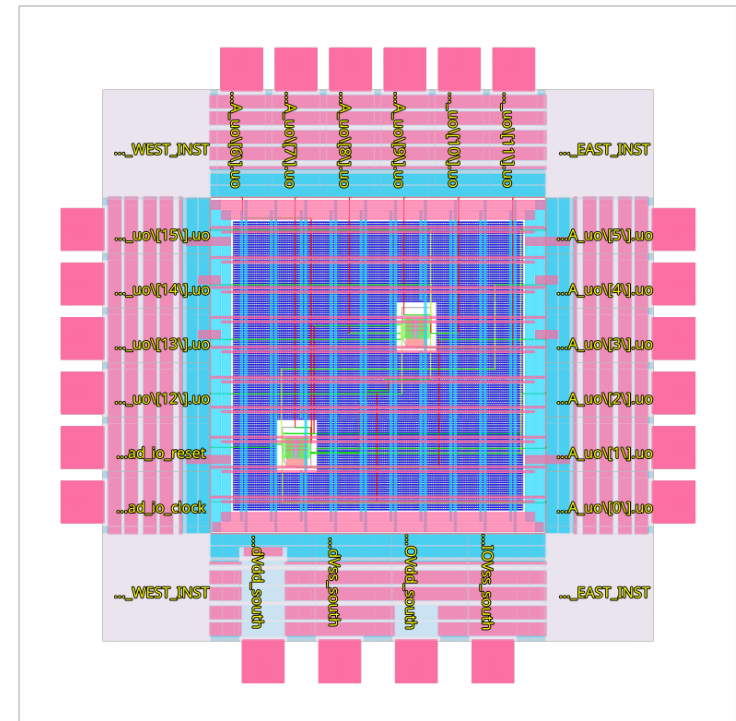
- ▶ A physical abstract view (LEF) to define area, pins, and routing blockages.
- ▶ A final layout (GDS) used for stream-out and sign-off verification.
- ▶ A logical netlist to connect the macro at the top level.
- ▶ Timing models across relevant corners for chip-level timing analysis.
- ▶ Optional parasitic information to improve accuracy during sign-off.
- ▶ Fixed placement information when the macro must be placed explicitly.

Classic vs Chip flow

```
{
  "meta": {
    "version": 3,
    "flow": "Classic"
  },
  ...
}
```

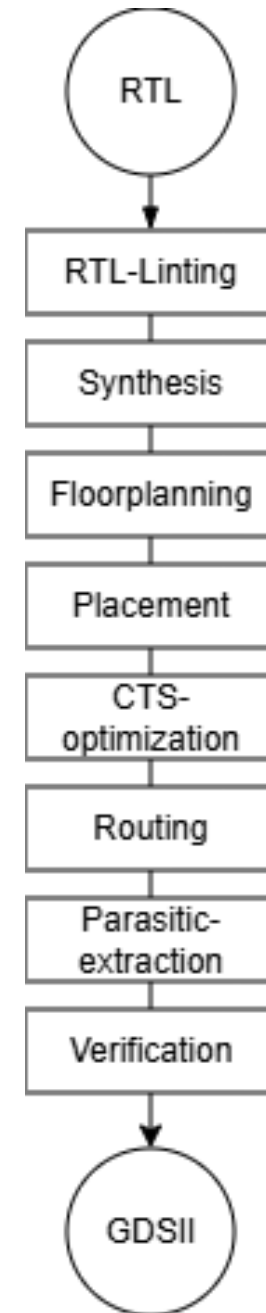


```
{
  "meta": {
    "version": 3,
    "flow": "Chip"
  },
  ...
}
```



Steps

- ▶ Generally used to harden a single design block from RTL to GDS.
- ▶ Assumes a self-contained design.



Quick tutorial - Classic Flow

counter_8bit.v

```
{
module counter_8bit (
    input logic      clk_i,
    input logic      rst_ni,
    output logic [7:0] count_o
);

    always_ff @(posedge clk_i) begin
        if (!rst_ni) begin
            count_o <= '0;
        end else begin
            count_o <= count_o + 1;
        end
    end

endmodule
}
```

config.json

```
{
    "DESIGN_NAME": "counter_8bit",
    "VERILOG_FILES": [
        "src/counter_8bit.v"
    ],
    "CLOCK_PORT": "clk",
    "CLOCK_PERIOD": 10,
    "FP_CORE_UTIL": 50,
    "FP_ASPECT_RATIO": 1.0,
    "PL_TARGET_DENSITY": 0.55
}
```

CLI

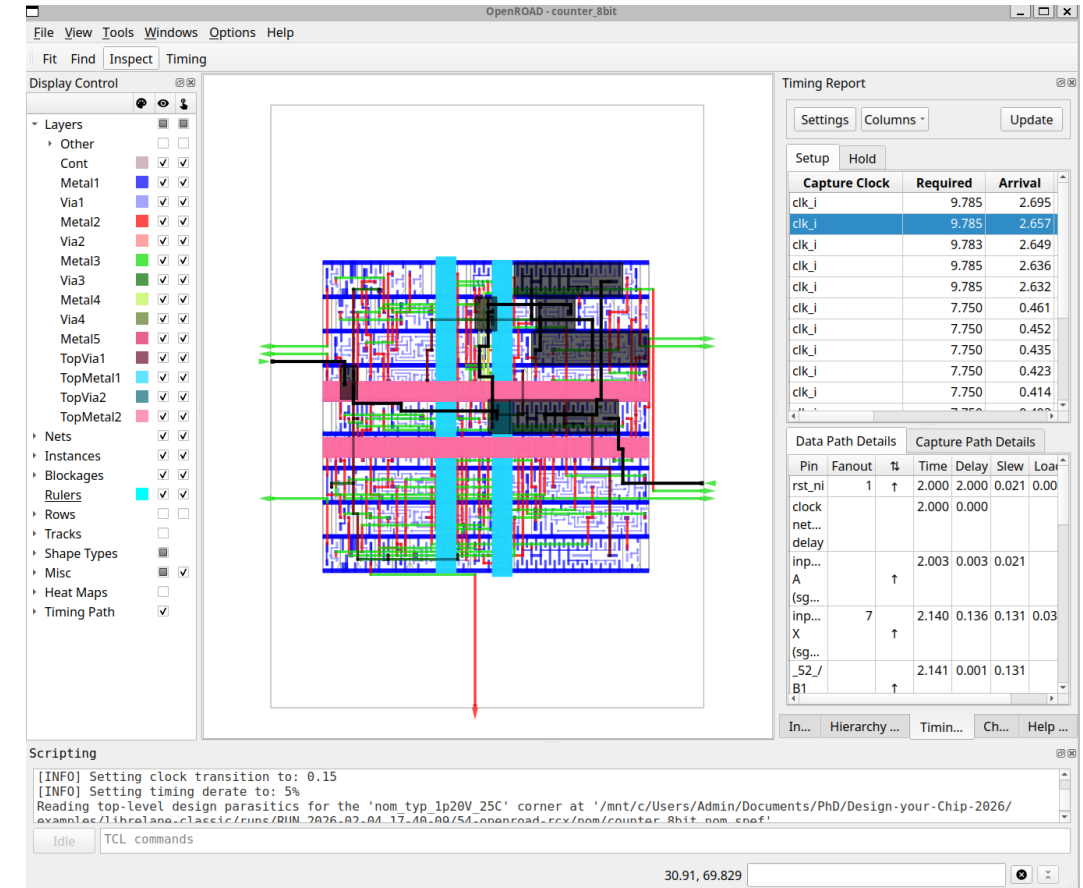
```
export PDK=ihp-sg13g2
librelane -pdk ihp-sg13g2 config.json
```

Visualization of Results

```
librelane --last-run config.json --flow OpenInOpenROAD
```

```
runs/<run_tag>
```

```
-- final
-- tmp
-- error.log
-- info.log
-- resolved.json
-- warning.log
-- 01-verilator-lint
-- 02-checker-linttimingconstructs
-- 03-checker-linterrors
-- 04-yosys-jsonheader
-- 05-yosys-synthesis
-- 06-checker-yosysunmappedcells
-- 07-checker-yosyssynthchecks
-- 08-openroad-checksdcfiles
-- 09-openroad-staprepnr
-- 10-openroad-floorplan
-- 11-odb-setpowerconnections
-- 12-odb-manualmacroplacement
-- 13-openroad-cutrows
-- 14-openroad-tapendcapinsertion
-- 15-openroad-globalplacementskipio
.
.
.
```



Quick tutorial - Chip Flow

- ▶ Full-chip design integrating two 8-bit counter macros.
- ▶ Macros are reused from the Classic flow and instantiated at the top level.
- ▶ Shared clock and reset, producing a 16-bit output.
- ▶ Dedicated core and IO power pads.

