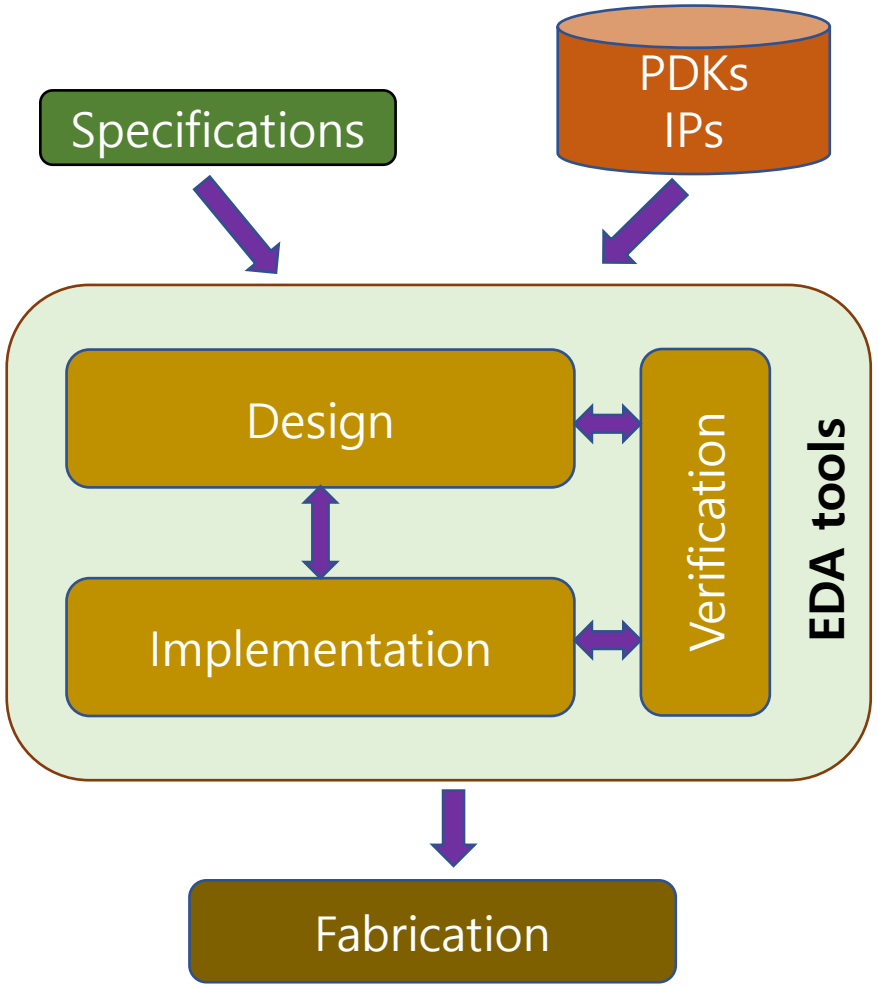
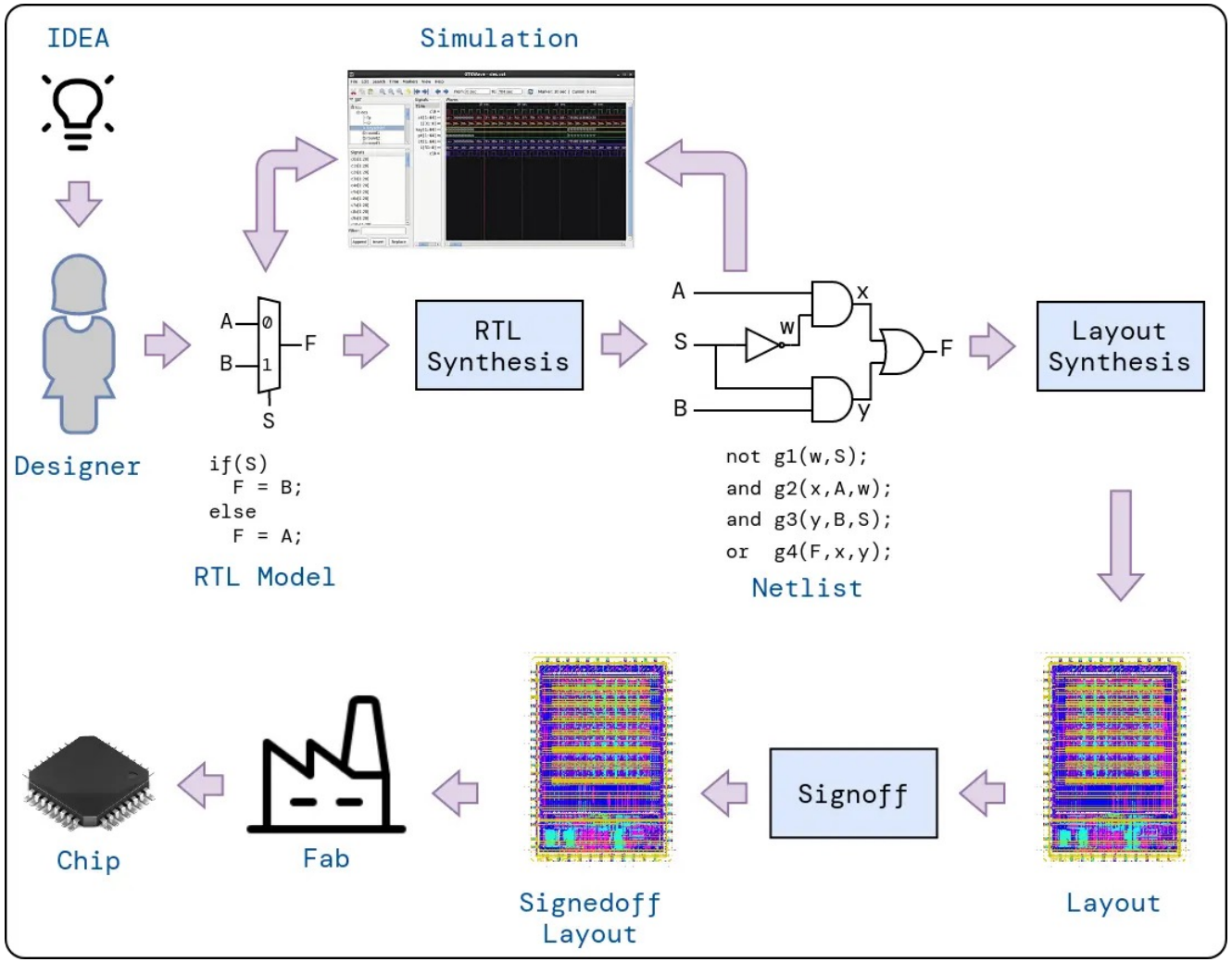
ASIC Physical Implementation Using Openlane2

Nguyen Dao
nguyen.dao@manchester.ac.uk

ASIC Design Flow





Opensource EDA tools and PDKs

Opensource tools

- Xschem – Analog/mixed signal design
- Yosys – RTL synthesis
- Icarus/Iverilog + GTKWave – synthesis/simulation and waveform viewer
- Openroad – Physical implementation
- OpenSTA – Timing/Power analysis
- Magic/Klayout – Layout/DRC check
- Netgen – LVS check
- CVC – Circuit Validity checker

- https://xschem.sourceforge.io/stefan/xschem_man/xschem_man.html
- https://github.com/YosysHQ/yosys
- http://iverilog.icarus.com
- https://gtkwave.sourceforge.net
- https://github.com/The-OpenROAD-Project/OpenLane
- http://opencircuitdesign.com/magic/
- https://www.klayout.de
- https://github.com/hpretl/iic-osic-tools

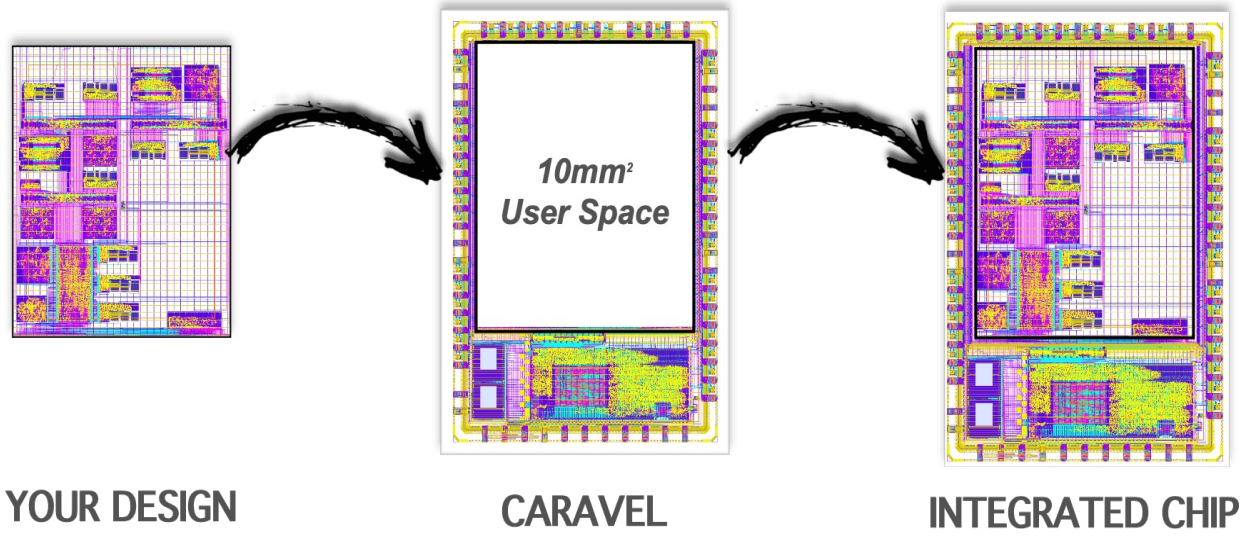
Opensource PDKs

- GF180MCU (GlobalFoundries 180nm)
- Sky130 – Sky90-FDSOI (Skywater 130nm – 90nm)
- FreePDK45 (45nm) - FreePDK15 (15nm)
- ASAP7 (Predictive 7nm Process)
- FreePDK3 (Predictive 3nm Process)

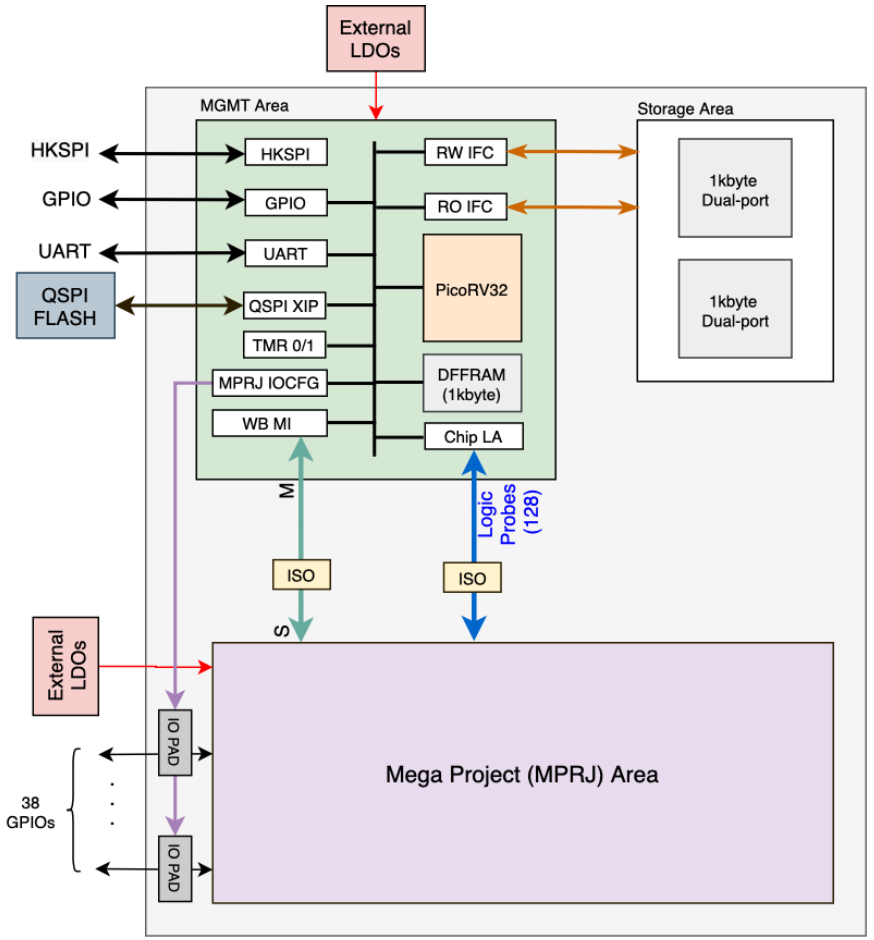
- https://skywater-pdk.readthedocs.io/en/main/
- https://opensource.googleblog.com/2022/07
- https://eda.ncsu.edu/
- https://asap.asu.edu

eFabless Caravel SoC

- **Caravel SoC** is composed of the harness frame, the **management area** and the **user project area**
- The management SoC is a RISC-V based SoC that includes several peripherals such as UART, GPIOs etc.
- The management SoC runs firmware that can be used to configure the IOs, control the power supply and observe/control signals to/from User project wrapper

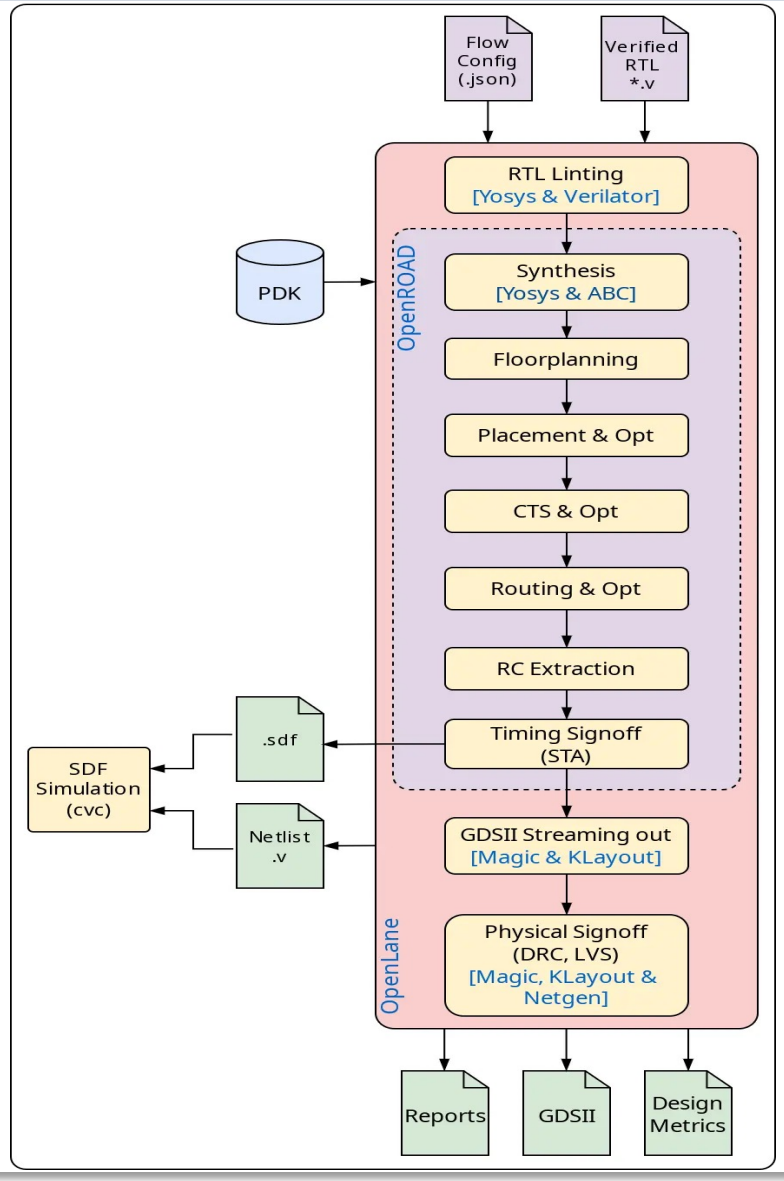


- User project area (2.92mm x 3.52 mm) has fixed 38 GPIOs, 128 Logic analyzer probes and Wishbone port connections to management SoC



Openlane2 Design Flow

1. **Synthesis**
   - o Yosys & Verilator – RTL synthesis
   - o ABC – technology mapping
   - o OpenSTA – static timing analysis
2. **Floorplan and PDN**
   - o Init_fp – core area planning
   - o Ioplacer – macros/Ios placement
   - o Pdn – implement power distribution network
   - o Tapcell – insert welltap/decap cells
3. **Placement**
   - o RePlace – perform global placement
   - o Resizer – optimize the design
   - o OpenDP – perform detailed placements
4. **CTS**
   - o TritionCTS – Clock Tree Synthesis
5. **Routing**
   - o FastRout/CU-GR – perform global routing
   - o TritonRoute – perform detailed routing
   - o SPEF-Extractor - perform parasitic extraction
6. **GDSII Generation**
   - o Magic/Klayout – stream out the final GDSII layout file

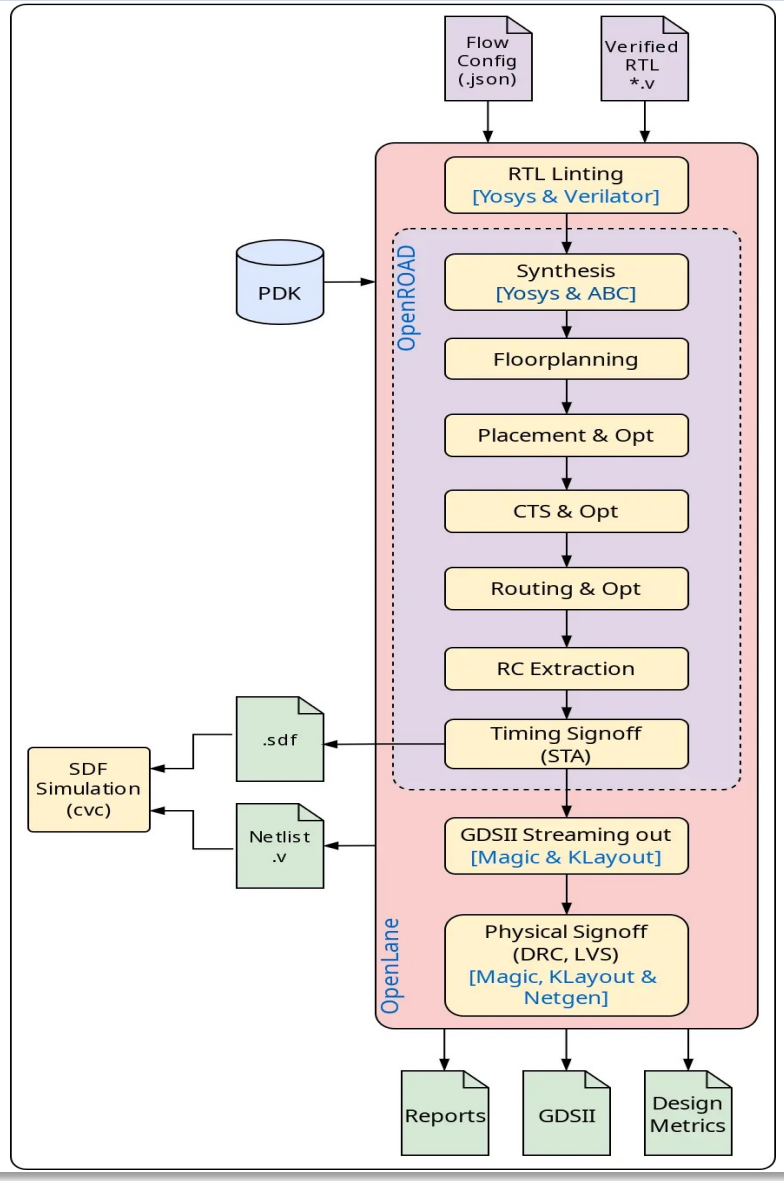


https://openlane2.readthedocs.io/en/latest/getting_started/newcomers/index.html

Openlane2 Design Flow

Hardening strategies:

1. **Maro-First Hardening:** Harden the user macro(s) initially and incorporate them into the user project wrapper without top-level standard cells. Ideal for a smaller designs, as this approach significantly reduces Placement and Routing (OnR) and signoff time.
2. **Full-Wrapper Flattening:** Merge the user macro(s) with the user_project_wrapper, covering the entire wrapper area. While this method demands more time and iterations for PnR and signoff, it ultimately enhances performance, making it suitable for design requiring the full wrapper area.
3. **Top-Level Integration:** Place the user macro(s) within the wrapper alongside standard cells at the top level, This method is typically chosen to introduce buffering at the top level, fitting scenarios where such an approach is necessary



https://openlane2.readthedocs.io/en/latest/getting_started/newcomers/index.html

Openlane2 Design Flow

Hardening strategies:

1. Prepare the design (RTL)

2. Set the flow constraints .json, timing constraints .sdc, pin placement .cfg

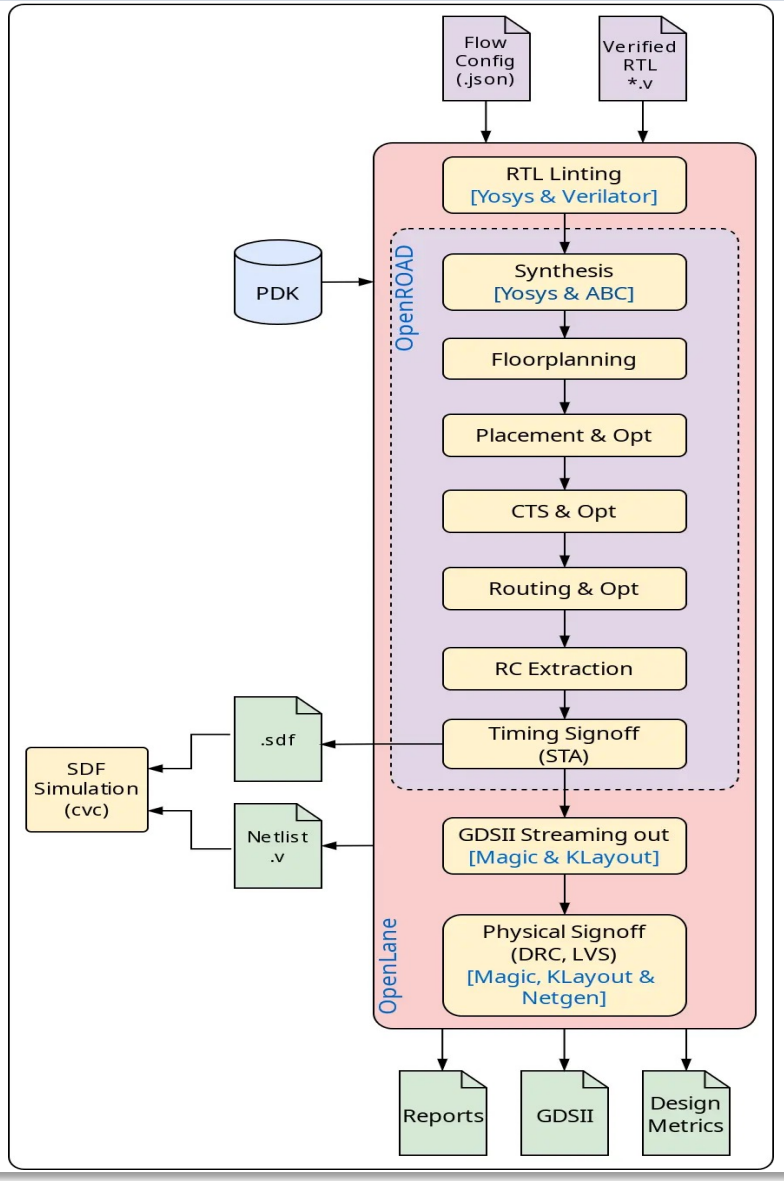
3. Run the flow

$openlane <dir_to_json_file>/config.json

4. Check the results

```
{
    "DESIGN_NAME": "aes_wb_wrapper",
    "FP_PDN_MULTILAYER": false,
    "CLOCK_PORT": "wb_clk_i",
    "CLOCK_PERIOD": 25,
    "VERILOG_FILES": [
        "dir::../../../secworks_aes/src/rtl/aes.v",
        "dir::../../../secworks_aes/src/rtl/aes_core.v",
        "dir::../../../secworks_aes/src/rtl/aes_decipher_block.v",
        "dir::../../../secworks_aes/src/rtl/aes_encipher_block.v",
        "dir::../../../secworks_aes/src/rtl/aes_inv_sbox.v",
        "dir::../../../secworks_aes/src/rtl/aes_key_mem.v",
        "dir::../../../secworks_aes/src/rtl/aes_sbox.v",
        "dir::../../verilog/rtl/aes_wb_wrapper.v"
    ],
    "FP_CORE_UTIL": 40
}
```

config.json



https://openlane2.readthedocs.io/en/latest/getting_started/newcomers/index.html

Openlane2 Design Flow

Hardening strategies:

1. Prepare the design (RTL)

2. Set the flow constraints .json, timing constraints .sdc, pin placement .cfg

3. Run the flow

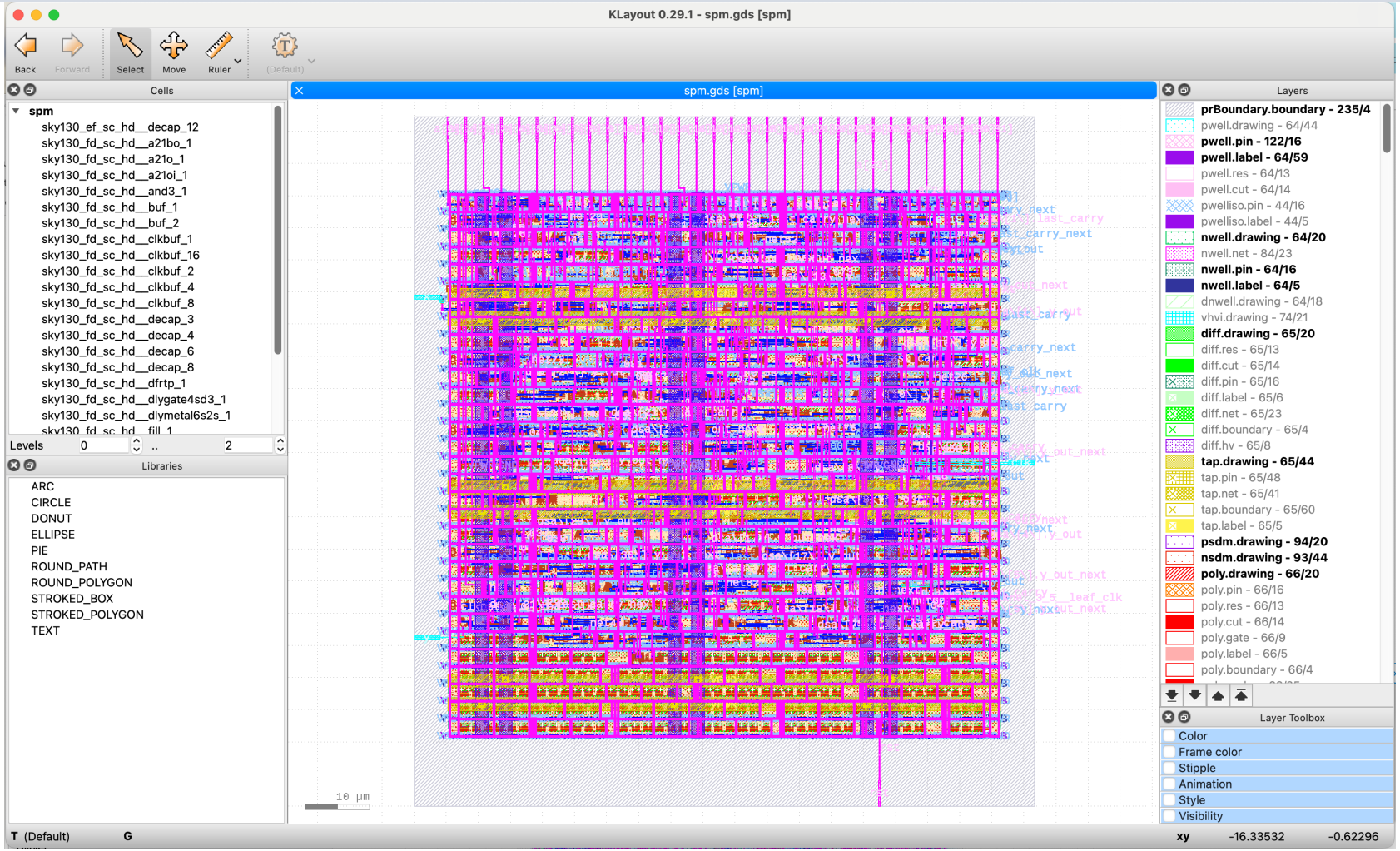
$openlane <dir_to_json_file>/config.json

4. Check the results

```
E-LOSXGBE1WFV:runs x19637nd$ ls RUN_2024-08-08_00-02-53/
01-verilator-lint                       21-odb-removepdnobstructions          41-openroad-stamidpnr-3               61-klayout-drc
02-checker-linttimingconstructs         22-odb-addroutingobstructions         42-openroad-detailedrouting           62-checker-magicdrc
03-checker-linterrors                   23-openroad-globalplacementskipio     43-odb-removeroutingobstructions      63-checker-klayoutdrc
04-checker-lintwarnings                 24-openroad-ioplacement               44-openroad-checkantennas-1           64-magic-spiceextraction
05-yosys-jsonheader                     25-odb-customioplacement              45-checker-trdrc                      65-checker-illegaloverlap
06-yosys-synthesis                      26-odb-applydeftemplate               46-odb-reportdisconnectedpins         66-netgen-lvs
07-checker-yosysunmappedcells           27-openroad-globalplacement           47-checker-disconnectedpins           67-checker-lvs
08-checker-yosyssynthchecks             28-odb-writeverilogheader             48-odb-reportwirelength               68-checker-setupviolations
09-checker-netlistassignstatements      29-checker-powergridviolations        49-checker-wirelength                 69-checker-holdviolations
10-openroad-checksdcfiles               30-openroad-stamidpnr                 50-openroad-fillinsertion             70-checker-maxslewviolations
11-openroad-checkmacroinstances         31-openroad-repairdesignpostgpl       51-openroad-rcx                       71-checker-maxcapviolations
12-openroad-staprepnr                   32-openroad-detailedplacement         52-openroad-stapostpnr                72-misc-reportmanufacturability
13-openroad-floorplan                   33-openroad-cts                       53-openroad-irdropreport              error.log
14-odb-checkmacroantennaproperties      34-openroad-stamidpnr-1               54-magic-streamout                    final
15-odb-setpowerconnections              35-openroad-resizertimingpostcts      55-klayout-streamout                  flow.log
16-odb-manualmacroplacement             36-openroad-stamidpnr-2               56-magic-writelef                     resolved.json
17-openroad-cutrows                     37-openroad-globalrouting             57-odb-checkdesignantennaproperties   tmp
18-openroad-tapendcapinsertion          38-openroad-checkantennas             58-klayout-xor                        warning.log
19-odb-addpdnobstructions               39-odb-diodesonports                  59-checker-xor
20-openroad-generatepdn                 40-openroad-repairantennas            60-magic-drc
```

https://openlane2.readthedocs.io/en/latest/getting_started/newcomers/index.html

Openlane2 Design Flow



check the final layout gds with Klayout:

$openlane --last-run --flow openinklayout openlane/examples/spm/config.json

https://openlane2.readthedocs.io/en/latest/getting_started/newcomers/index.html

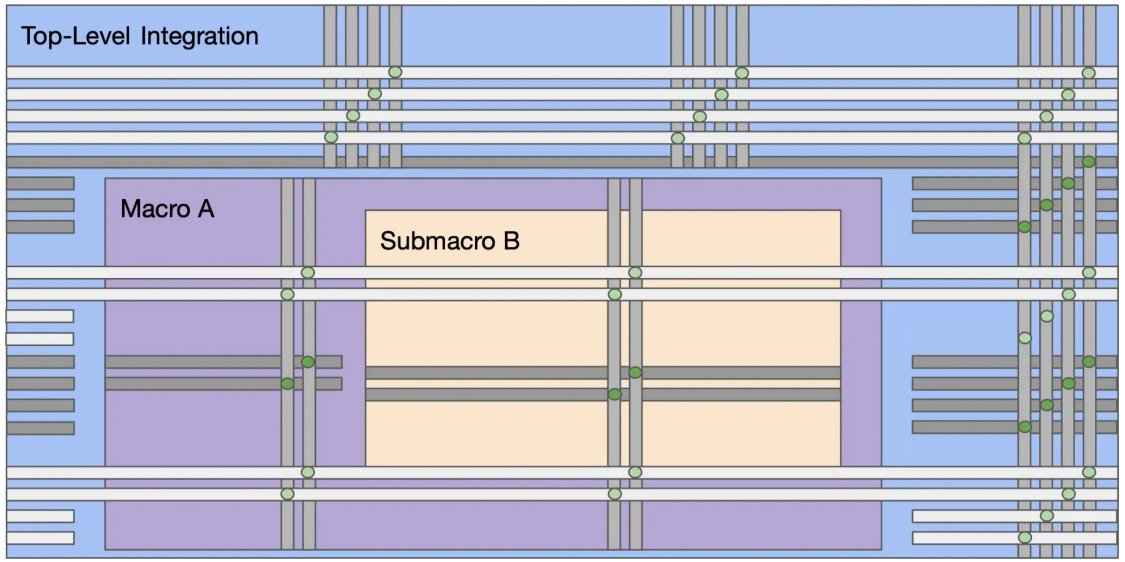
Openlane2 Design Flow

Macro integration:

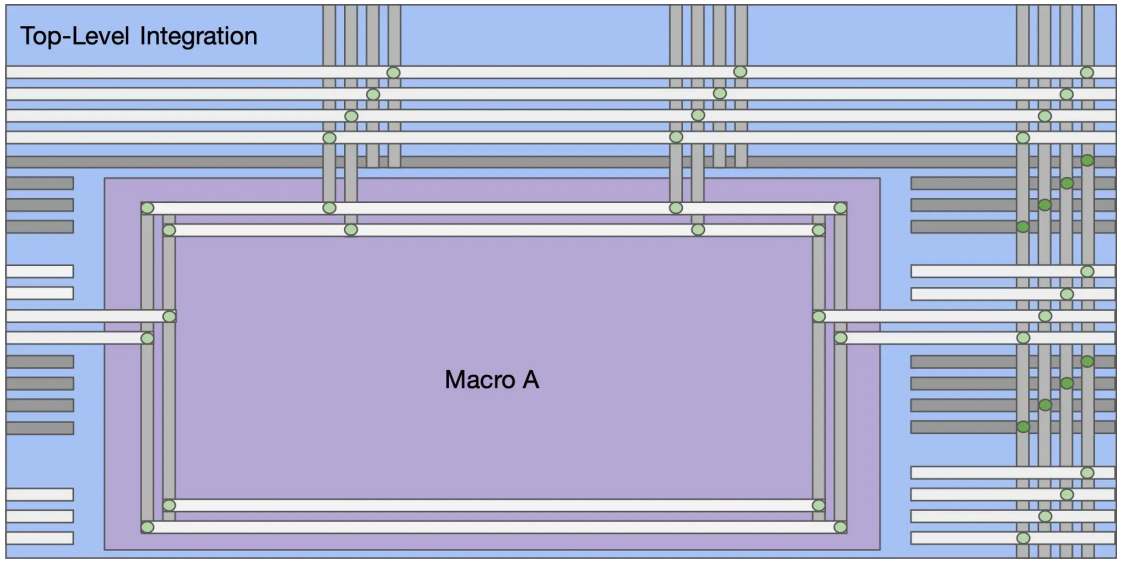
- **Hierarchical method:** saves space, but less routing layers available for the macros
- **Ring method:** Uses more space, allow any arbitrarily-nested macro to use the full routing layer stack. Useful if routing very complex macros.

All macros should be hardened with the following options:

- FP_PDN_CORE_RING
- FP_PDN_HORIZONTAL_LAYER
- FP_PDN_CORE_HWIDTH
- FP_PDN_CORE_VWIDTH
- FP_PDN_CORE_HOFFSET
- FP_PDN_CORE_VOFFSET
- FP_PDN_CORE_HSPACING
- FP_PDN_CORE_VSPACING



Macro integration using power straps



Macro integration using power ring

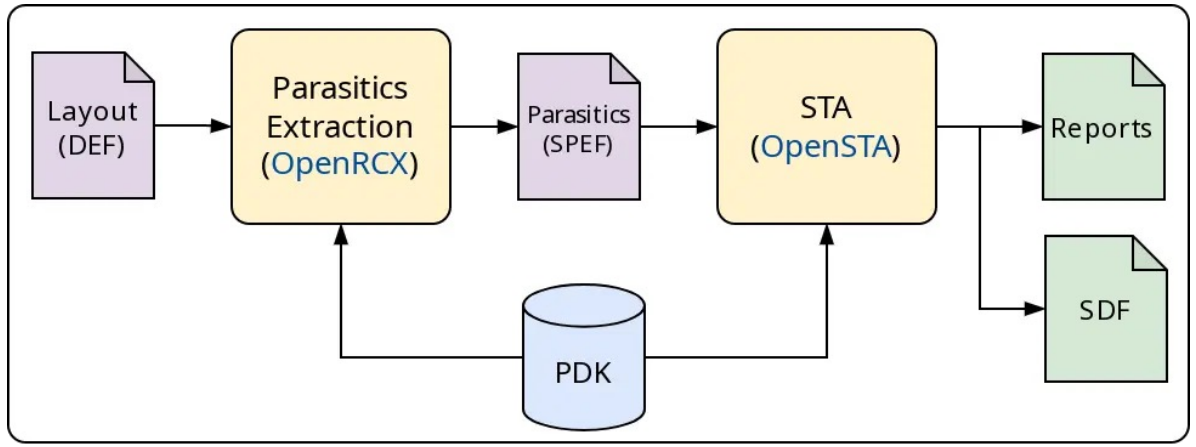
https://openlane2.readthedocs.io/en/latest/getting_started/newcomers/index.html

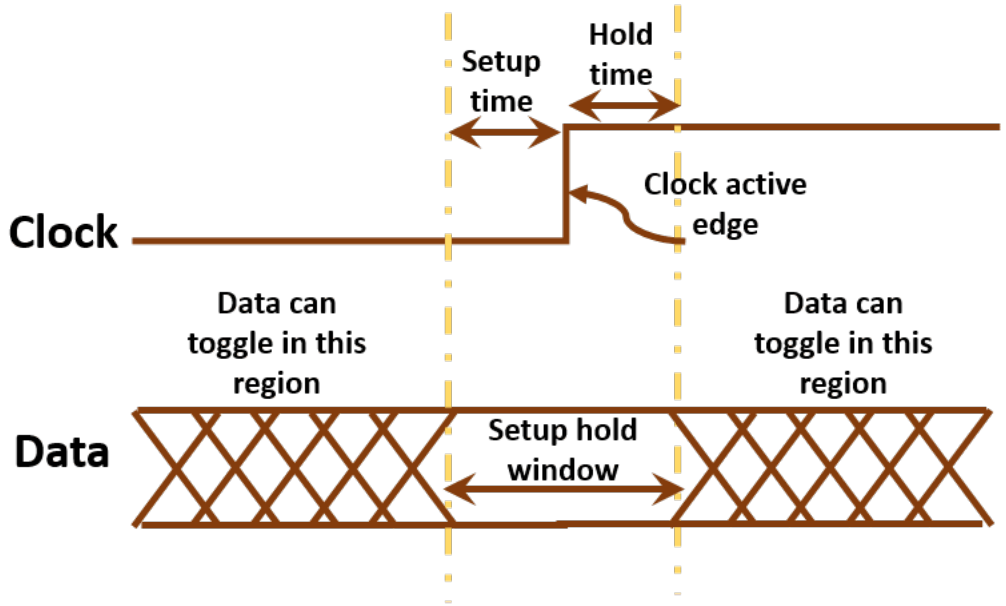
Openlane2 Design Flow

Static Timing Analysis (STA): the STA tool identified the design timing paths and then calculates the data's earliest and largest actual and required arrival times at every timing path endpoint.

If the data arrives after (setup checking) or before (hold checking) it is required, we have a timing violation (negative slack)

STA makes sure that a circuit will correctly perform its function (but tells nothing about the correctness of that function)





https://openlane2.readthedocs.io/en/latest/getting_started/newcomers/index.html

Openlane2 Design Flow

Timing corner: to ensure a chip can work properly under various conditions, the design must be analysed in different timing corners.

- Parasitic/Interconnect corners (capacitance, resistance)
- Transistor corners (fast, typical, slow)
- Temperature
- Voltage

Common EDA files incorporate these corners:

.spef – Parasitic/interconnect corners

.spice – interconnect and transistor corners

.lib – characteristics of a cell/macro at a full corner

Name	Process {NMOS, PMOS}	Voltage (V)	Temperature (C)	Corresponding File
"tt_025C_1v80"	{T, T}	1.8	25	`sky130_fd_sc_hd__tt_025C_1v80.lib`
"ss_100C_1v60"	{S, S}	1.6	100	`sky130_fd_sc_hd__ss_100C_1v60.lib`
"ff_n40C_1v95"	{F, F}	1.95	-40	`sky130_fd_sc_hd__ff_n40C_1v95.lib`

PVT: corner data stored in LIB

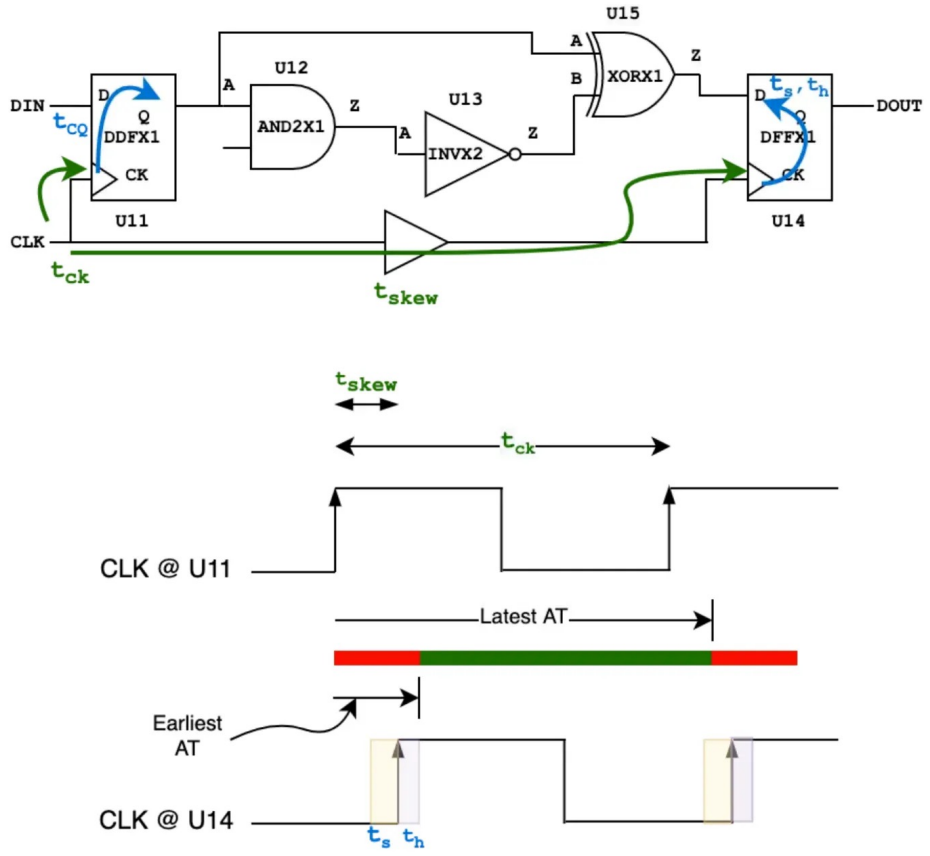
Name	Description	Corresponding Technology LEF	Corresponding Ruleset
"nom"	The nominal interconnect corner	`sky130_fd_sc_hd__nom.tlef`	`rules.openrcx.sky130A.nom.calibre`
"min"	The minimal interconnect corner	`sky130_fd_sc_hd__min.tlef`	`rules.openrcx.sky130A.min.calibre`
"max"	The maximum interconnect corner	`sky130_fd_sc_hd__max.tlef`	`rules.openrcx.sky130A.max.calibre`

Corner data stored in TECH_LEFS and RCX_TULESETS

https://openlane2.readthedocs.io/en/latest/getting_started/newcomers/index.html

Openlane2 Design Flow

Timing closure:



Timing constraints need to be set to help the tool analyse and optimize the design. The timing constraints are defined in SDC file (.sdc) such as create_clock, set_input_delay, set_output_delay, set_load, set_drive, etc.

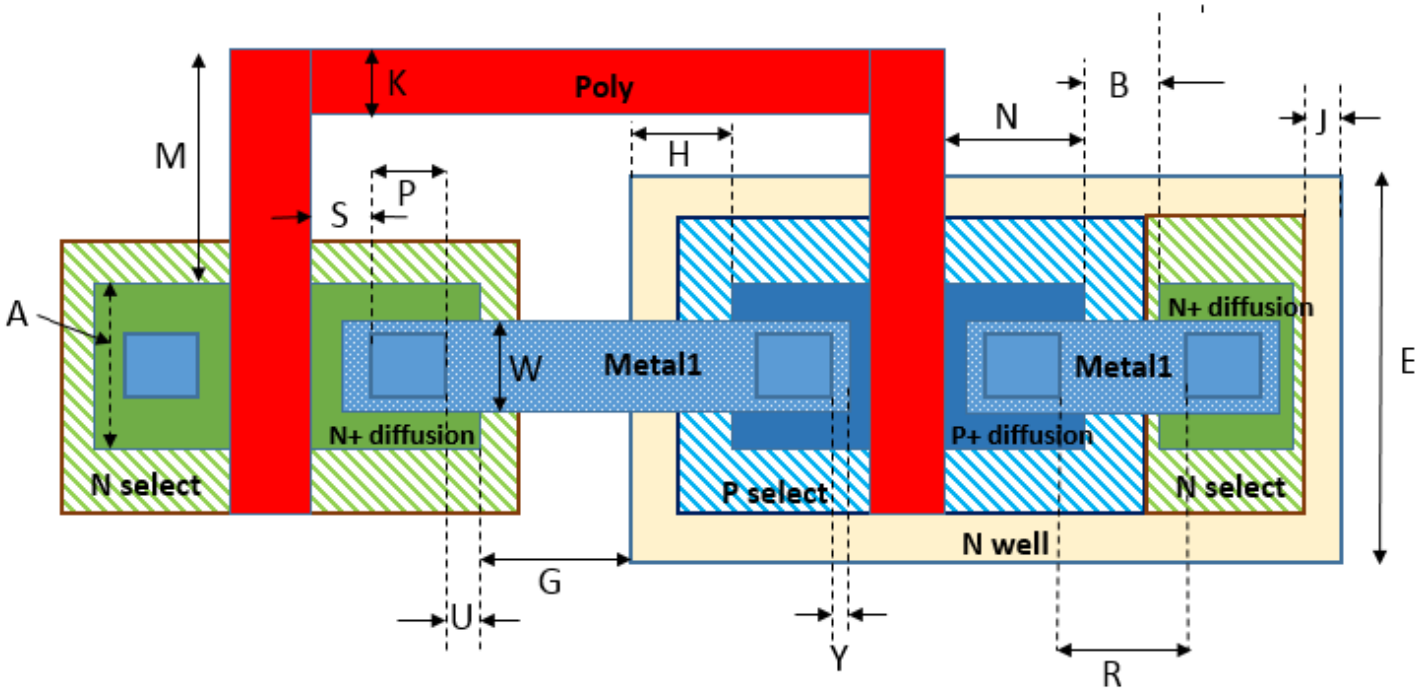
Parameter	Use
`RUN_POST_CTS_RESIZER_TIMING`	Specifies whether design timing optimizations should be performed immediately after placement and CTS or not.
`PL_RESIZER_SETUP_SLACK_MARGIN`	Used to guide timing optimization after placement. It instructs the optimizer not to stop at zero slack and try to achieve a positive timing slack.
`PL_RESIZER_HOLD_SLACK_MARGIN`	Used to guide timing optimization after placement. It instructs the optimizer not to stop at zero slack and try to achieve a positive timing slack.
`PL_RESIZER_HOLD_MAX_BUFFER_PCT`	Maximum % of hold buffers to insert to fix hold vios. (PL/CTS)
`PL_RESIZER_SETUP_MAX_BUFFER_PCT`	Maximum % of hold buffers to insert to fix setup vios. (PL/CTS)
`PL_RESIZER_ALLOW_SETUP_VIOS`	Allow setup violations while fixing hold violations after placement and CTS.
`RUN_POST_GRT_RESIZER_TIMING`	Enable/disable timing optimizations after global routing. This is an experimental feature in OpenROAD and crashes may occur.
`GRT_RESIZER_SETUP_SLACK_MARGIN`	Used to guide timing optimization after global routing. It instructs the optimizer not to stop at zero slack and try to achieve a positive slack (the specified margin.)
`GRT_RESIZER_HOLD_SLACK_MARGIN`	Used to guide timing optimization after global routing. It instructs the optimizer not to stop at zero slack and try to achieve a positive slack (the specified margin.)
`GRT_RESIZER_HOLD_MAX_BUFFER_PCT`	Maximum % of hold buffers to insert to fix hold vios after global routing.
`GRT_RESIZER_SETUP_MAX_BUFFER_PCT`	Maximum % of hold buffers to insert to fix setup vios after global routing.
`GRT_RESIZER_ALLOW_SETUP_VIOS`	Allow setup violations while fixing hold violations after global routing.

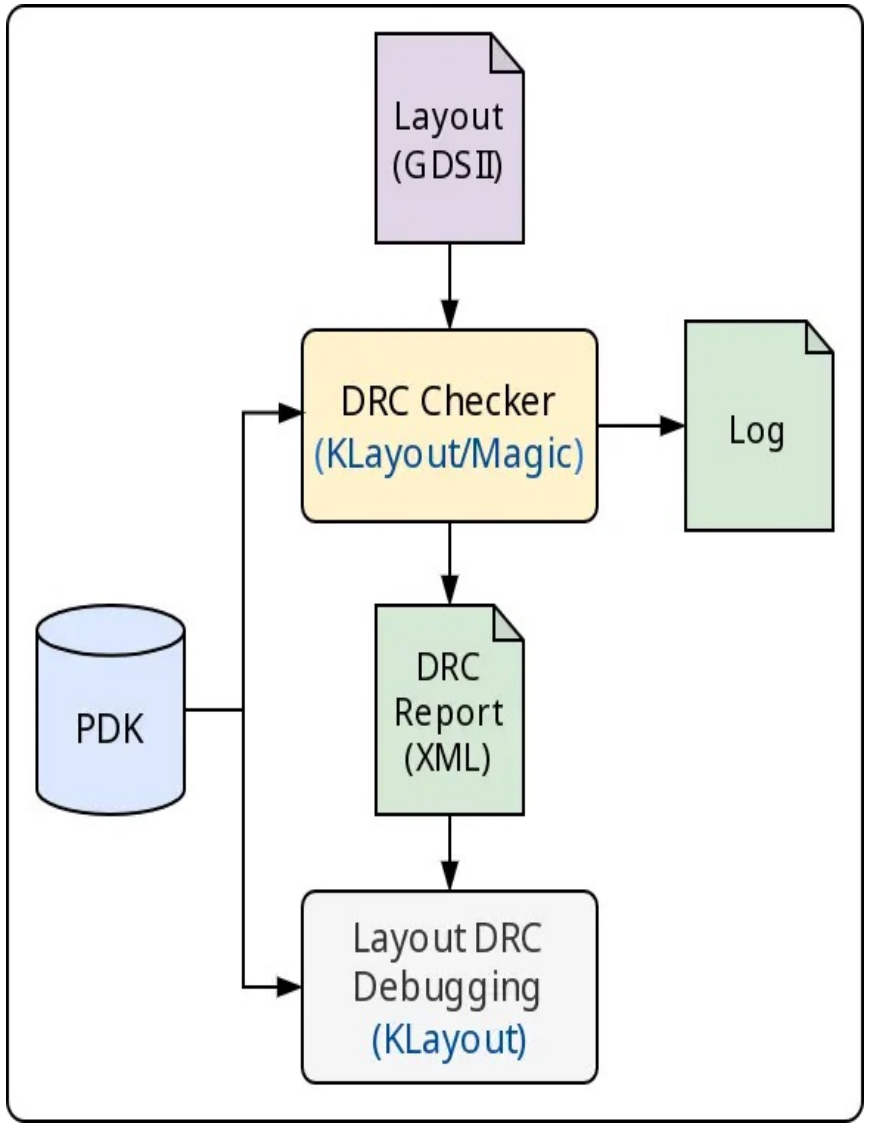
https://openlane2.readthedocs.io/en/latest/getting_started/newcomers/index.html

Openlane2 Design Flow

Design Rule Check (DRC):

The design layout has to be checked against rules set by chip foundries to ensure that it can be manufacturable.





https://openlane2.readthedocs.io/en/latest/getting_started/newcomers/index.html

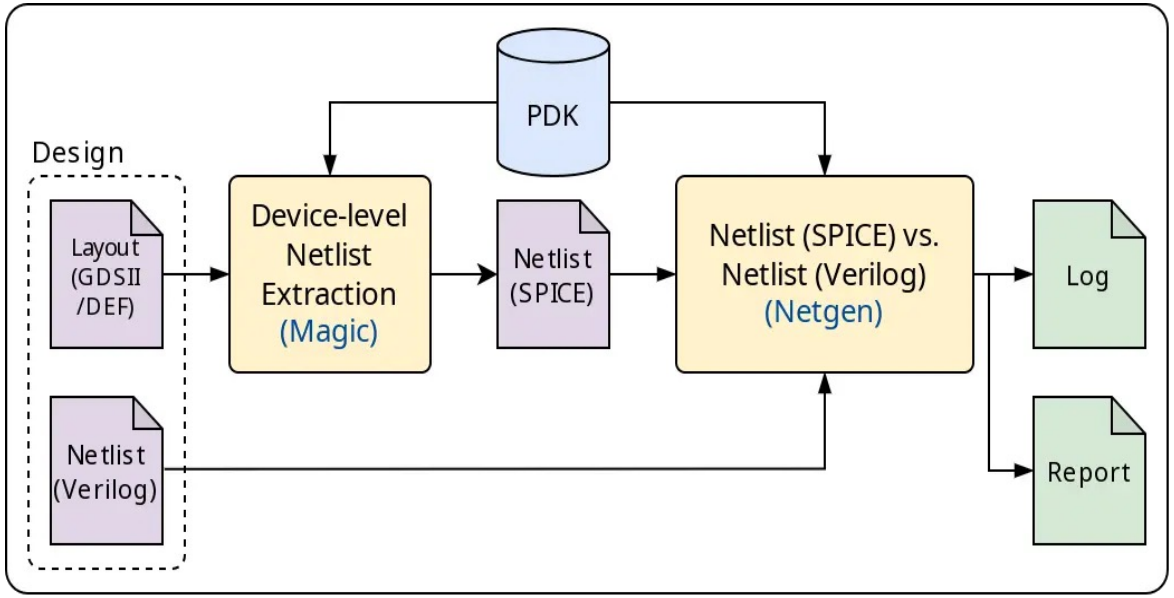
Openlane2 Design Flow

Layout Versus Schematic (LVS):
It compares the layout GDSII or DEF/LEF, with the schematic to ensure the connectivity in both views matches.

Some common LVS errors such as:

- Short: Two or more wires that should not be connected have been and must be separated. The most problematic is power and ground shorts.
- Open: Wires or components that should be connected are left dangling or only partially connected.
- Missing components: An expected components has been left out of the layouts. Normally some parasitic/dummy components need to be added to match with the layout

Netgen.LVS is the Step run for LVS using a tool called Netgen. First the layout is converted to SPICE netlist, Next, the layout and the schematic are inputted to Netgen



Subcircuit summary:

Circuit 1: pm32	Circuit 2: pm32
sky130_fd_sc_hd__tapvpwrvgnd_1 (102->1)	sky130_fd_sc_hd__tapvpwrvgnd_1 (102->1)
sky130_fd_sc_hd__decap_3 (144->1)	sky130_fd_sc_hd__decap_3 (144->1)
sky130_fd_sc_hd__inv_2 (64)	sky130_fd_sc_hd__inv_2 (64)
sky130_fd_sc_hd__nand2_1 (31)	sky130_fd_sc_hd__nand2_1 (31)
sky130_fd_sc_hd__dfrtp_1 (64)	sky130_fd_sc_hd__dfrtp_1 (64)
sky130_ef_sc_hd__decap_12 (132->1)	sky130_ef_sc_hd__decap_12 (132->1)

https://openlane2.readthedocs.io/en/latest/getting_started/newcomers/index.html

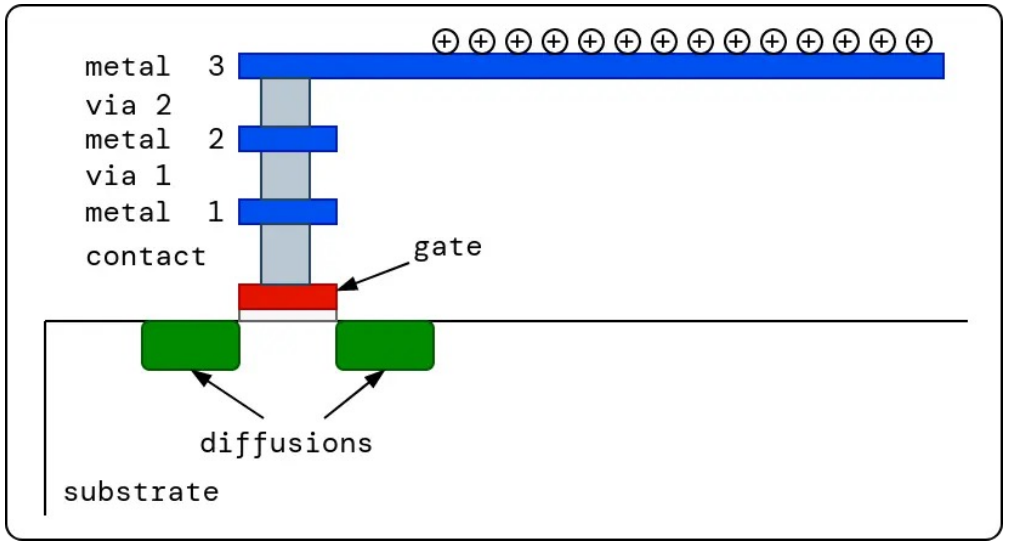
Openlane2 Design Flow

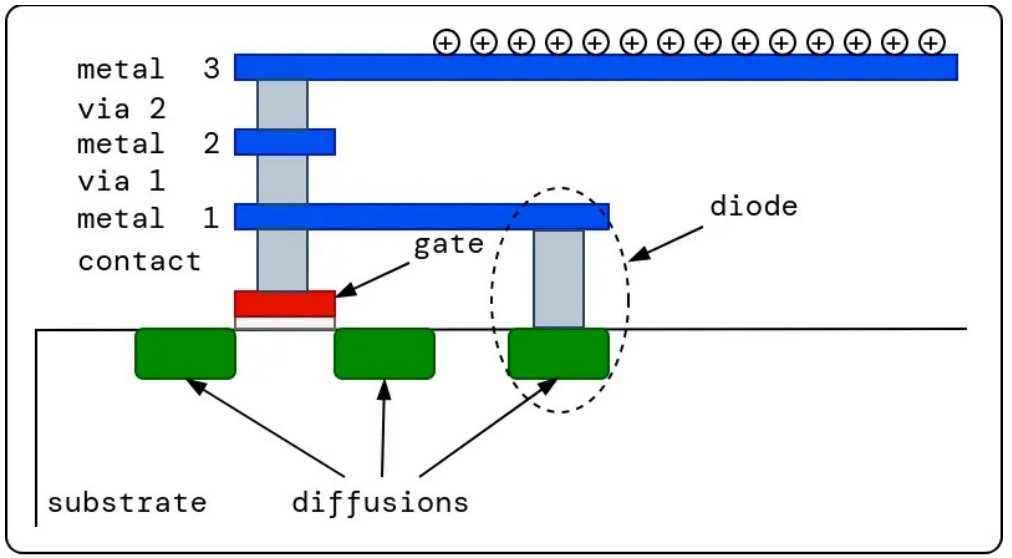
Antenna Check:
Long metal wire segments that are connected to a transistor gate may damage the transistor's thin gate oxide during the fabrication process due to its collection of charges from the processing environment

Chip foundries normally supply antenna rules, which set the limit of the ratio of collection area and drainage (thin oxide) area.

Antenna effect can be avoided by instructing the router to use short wire segments and to create bridges to disconnect long from transistor gates during fabrication (this is not supported by Openlane flow)

Openlane uses another approach that involves the insertion of an antenna diode (provided as a standard cell) next to the cell input pin that suffers from the antenna effect.

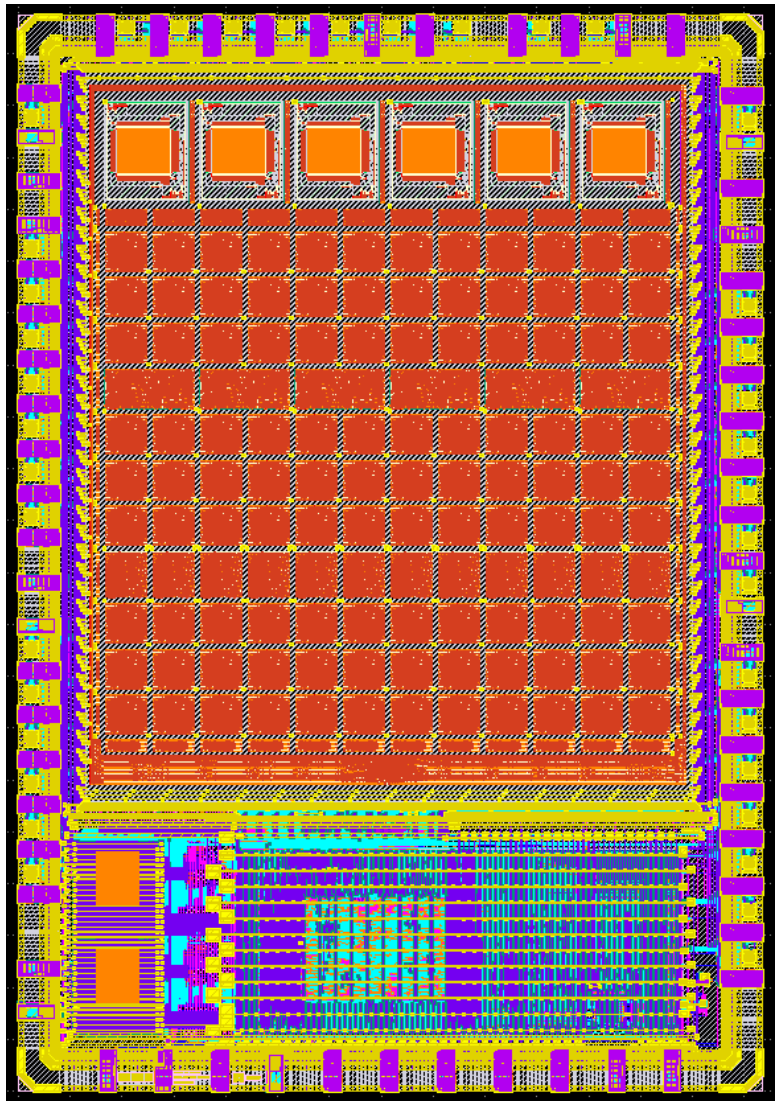




https://openlane2.readthedocs.io/en/latest/getting_started/newcomers/index.html

eFPGA design flow using Openlane

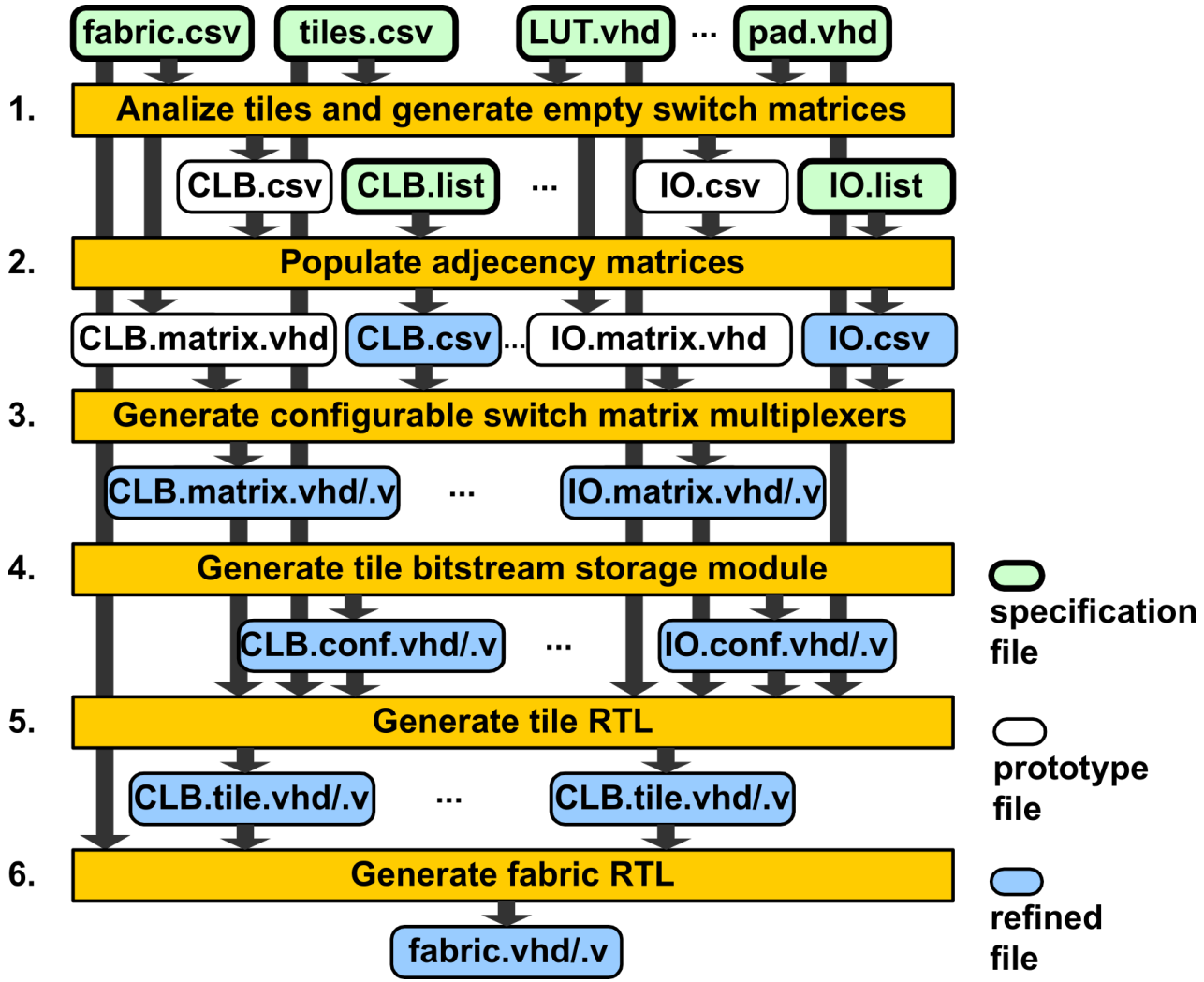
1. Create/Generate RTLs
2. Customize the cells (optional)
3. Hardening tiles (optional)
4. Hardening the fabric (eFPGA_top)/User project wrapper
5. Caravel Integration



eFPGA design flow using Openlane

1. Generate the fabrics (RTLs)

More details at
https://fabulous.readthedocs.io/en/latest/Building%20fabric.html



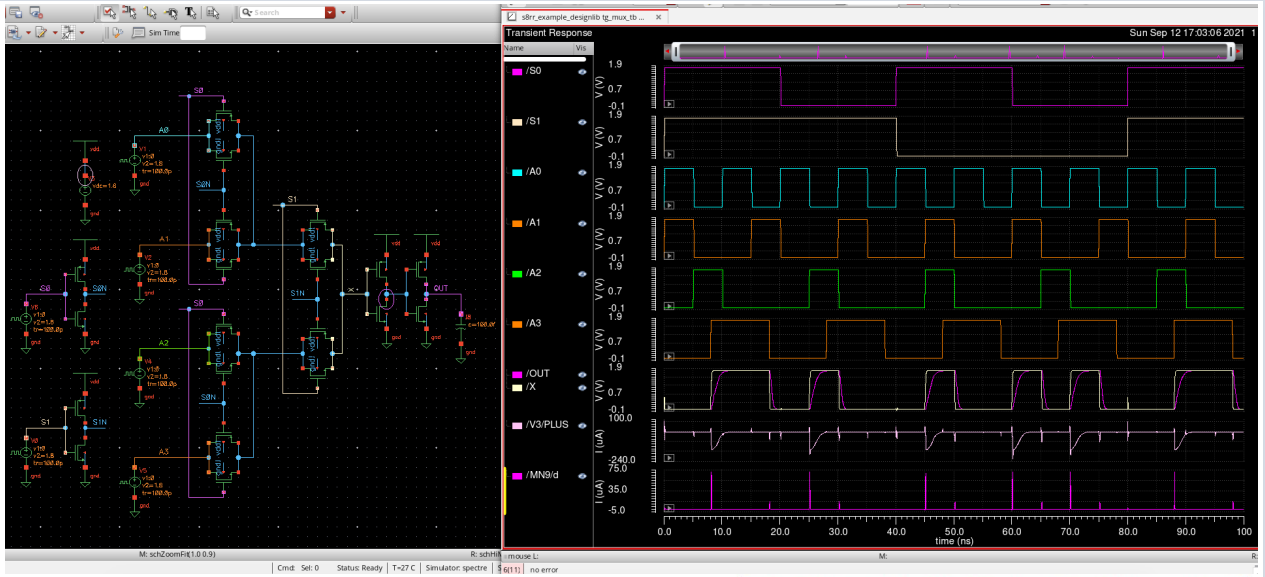
eFPGA design flow using Openlane

2. Customize the cells (optional)

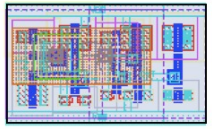
- 2.1. Create schematic
- 2.2. Simulation
- 2.3. Layout
- 2.4. Extract and do post-layout simulation
- 2.5. Export LEF/lib/GDS

```
VERSION 5.7 ;
  NOWIREEXTENSIONATPIN ON ;
  DIVIDERCHAR "/" ;
  BUSBITCHARS "[]" ;
MACRO cus_tg_mux41_buf
  CLASS CORE ;
  FOREIGN cus_tg_mux41_buf ;
  ORIGIN 0.000 0.000 ;
  SIZE 6.440 BY 2.720 ;
  SYMMETRY X Y R90 ;
  SITE unithd ;
  PIN S0
    ANTENNAGATEAREA 0.216000 ;
    DIRECTION INPUT ;
    USE SIGNAL ;
    PORT
      LAYER met2 ;
        RECT 1.980 1.310 2.300 1.570 ;
        RECT 2.035 0.800 2.245 1.310 ;
        RECT 1.995 0.480 2.255 0.800 ;
    END
    PORT
      LAYER met1 ;
        RECT 0.145 1.550 0.375 1.700 ;
        RECT 1.980 1.550 2.300 1.570 ;
        RECT 0.145 1.410 2.300 1.550 ;
        RECT 1.980 1.310 2.300 1.410 ;
      LAYER via ;
        RECT 2.010 1.310 2.270 1.570 ;
    END
  END S0
...
  END
END cus_tg_mux41_buf
```

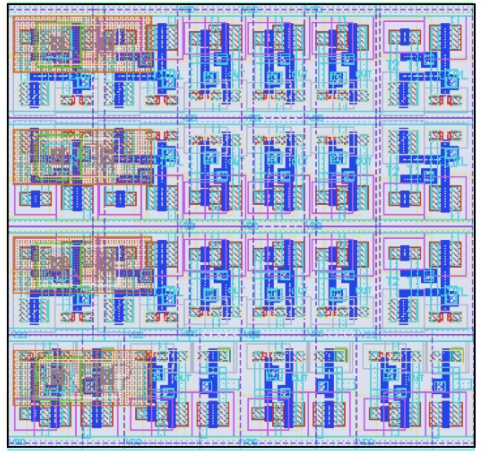
```
library ("custom_mux") {
    define(def_sim_opt,library,string);
    ...
    time_unit : "1ns";
    voltage_unit : "1V";
    leakage_power_unit : "1nW";
    current_unit : "1mA";
    pulling_resistance_unit : "1kohm";
    capacitive_load_unit(1.0000000000, "pf"); ...
    default_arc_mode : "worst_edges";
    default_constraint_arc_mode : "worst";
    default_leakage_power_density : 0.0000000000;
    default_operating_conditions : "tt_025C_1v80";
    operating_conditions ("tt_025C_1v80") {
        voltage : 1.8000000000;
        process : 1.0000000000;
        temperature : 25.000000000;
        tree_type : "balanced_tree";
    }
    power_lut_template ("power_inputs_1") {
        variable_1 : "input_transition_time";
        index_1("1, 2, 3, 4, 5, 6, 7");
cell ("cus_tg_mux41_buf") {
        leakage_power () {
            value : 0.0137458000;
            when : "!A0&!A1&!A2&!A3&!S0&S1";
        }
                related_pin : "S1";
                rise_transition ("del_1_7_7") {

        ...
                timing_sense : "negative_unate";
                timing_type : "combinational";
            }
}
```

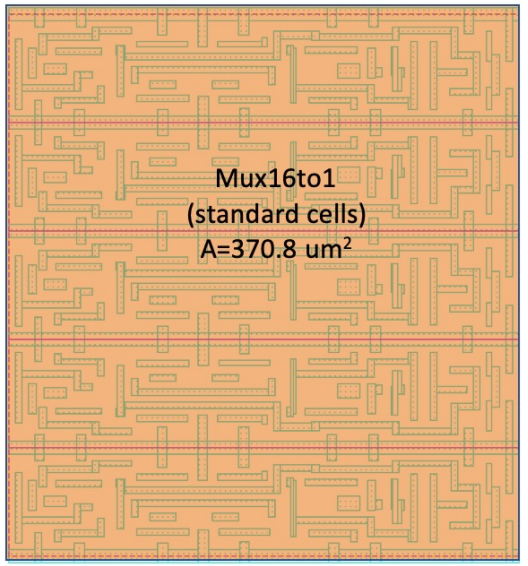


Mux2to1
A=27.3 um^2



Mux16to1
A =272.45 um^2

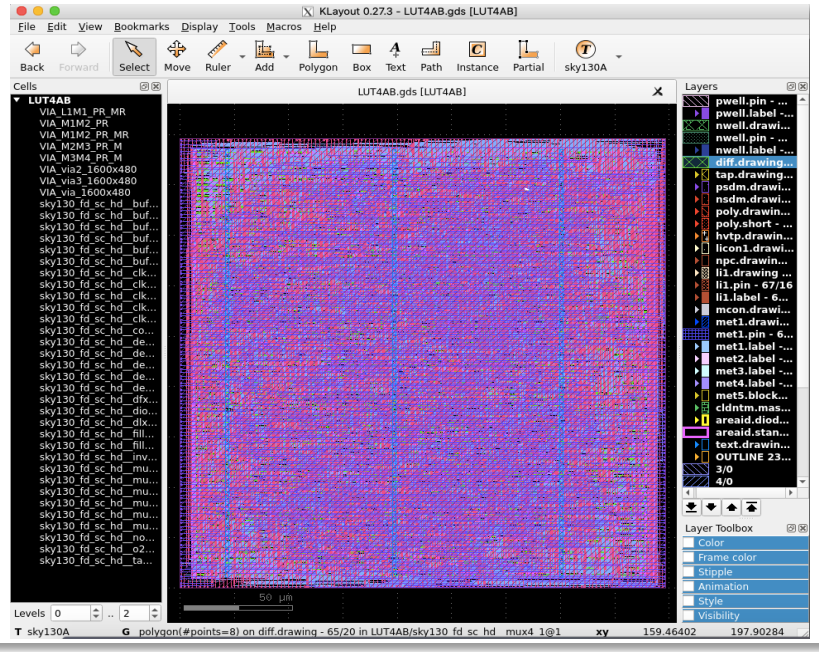




eFPGA design flow using Openlane

3. Hardening tiles (optional)

- Configure the flow and design constraints (config.tcl)
- Initial/Set Technology/Lib and Top design
- Set area/density
- Set clock constraints
- Set technology/custom gates mapping
- Set IO pins arrangement
- Set Timing constraints (disable timing loops)
- Set routing constraints (layers/halos)

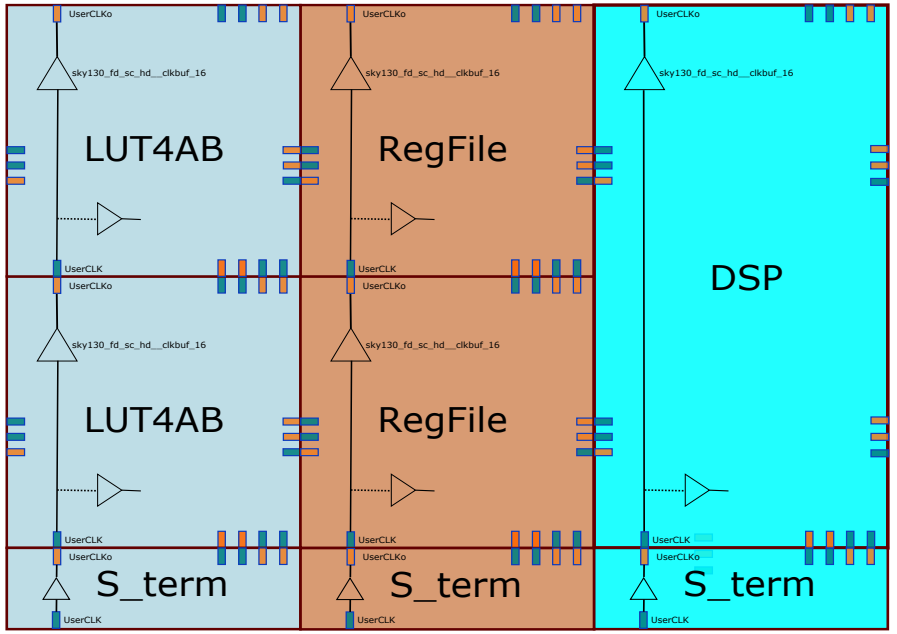



```
# User config
set ::env(DESIGN_NAME) LUT4AB

# Change if needed
set ::env(VERILOG_FILES) [glob $::env(DESIGN_DIR)/src/*.v]

# Use FP_CORE_UTIL to experiment the tile's size
#set ::env(FP_CORE_UTIL) 55

# Use FP_SIZING to fix the tile's area
set ::env(FP_SIZING) "absolute"
set ::env(DIE_AREA) "0 0 223.275 223.115"; #baseline LUT4AB(223.275 223.115)

#set ::env(PL_TARGET_DENSITY) [ expr ($::env(FP_CORE_UTIL)+5) / 100.0 ]
set ::env(PL_TARGET_DENSITY) 0.6

# Clock config
set ::env(CLOCK_PERIOD) "40"
set ::env(CLOCK_PORT) "UserCLK"
set ::env(CLOCK_TREE_SYNTH) 1

# Synthesis mode - should disable flattening the hierarchy that helps setting timing contraints later
# It also requires remove "-flatten" option at line 353 in scripts/yosys/synth.tcl
set ::env(SYNTH_NO_FLAT) 1

# DESIGN_IS_CORE 1 default, 0 is a macro
set ::env(DESIGN_IS_CORE) 0
set ::env(FP_PDN_CORE_RING) 0
set ::env(GLB_RT_MAXLAYER) 5

# Specify latch gate for mapping
set ::env(SYNTH_LATCH_MAP) $::env(DESIGN_DIR)/gate_map.v

# Change sdc file
set ::env(BASE_SDC_FILE) $::env(DESIGN_DIR)/LUT4AB.sdc

# Change the size and arrange the IO pins
set ::env(FP_IO_VLENGTH) 0.8
set ::env(FP_IO_HLENGTH) 0.8
set ::env(FP_IO_HTHICKNESS_MULT) 2
set ::env(FP_IO_VTHICKNESS_MULT) 2
set ::env(FP_IO_MODE) 0
set ::env(FP_PIN_ORDER_CFG) $::env(DESIGN_DIR)/pin_order.cfg

# Adjust the floorplan
set ::env(TOP_MARGIN_MULT) 2
set ::env(BOTTOM_MARGIN_MULT) 2

# Set the power pins
set ::env(VDD_PINS) "vccd1"
set ::env(GND_PINS) "vssd1"

set ::env(ROUTING_CORES) 12
```

eFPGA design flow using Openlane

3. Hardening tiles - Notes

- Need to add the lib/lef of the custom cells to the sky130 tech files
- Enable latch mapping – need to specify the latch used for configurations and specify the custom cells be used (gate_map.v)
- Enable hierarchical synthesis (SYNTH_NO_FLAT=1) to resist changing the module name during yosys synthesis
- Set Mux4 as preferable for better density (Yosys uses Mux2 as the default)
- RTL syntax, limited support for SystemVerilog e.g., global param/inherited param (#)
- Disable combinational loops by replacing the default *base.sdc*
- IO pins placement is limited (e.g., single metal layer only)
- Clock tree synthesis is limited (cannot handle a very large number of connections)



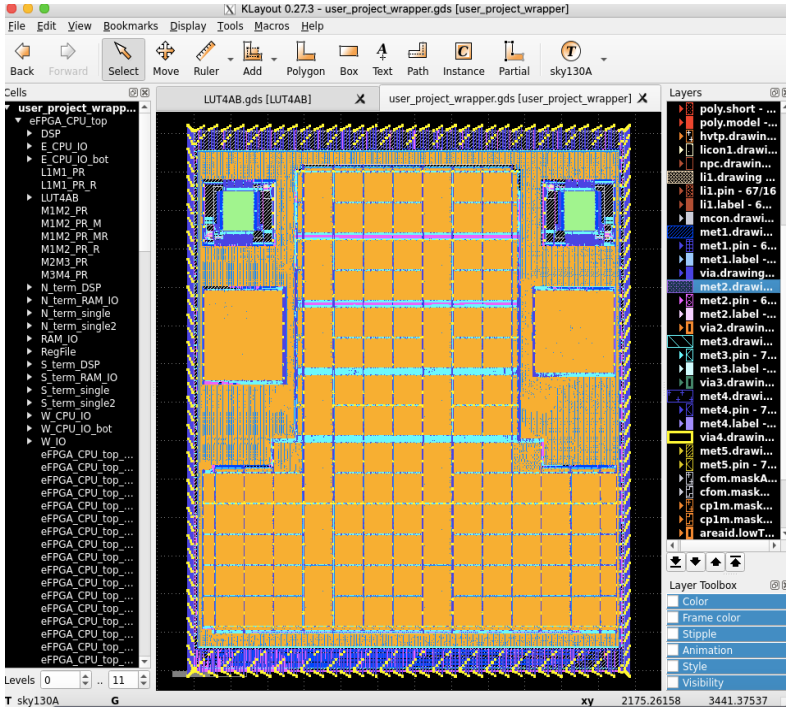
https://github.com/nguyendao-uom/open_eFPGA

eFPGA design flow using Openlane

4. Hardening the User project (eFPGA_top fabric)

- Instantiate and connect the fabric (eFPGA_top) in User_project_wrapper.v
- Floorplan and Placement constraints
- Configure the Openlane flow and the design constraints
- Hardening the User_project_wrapper

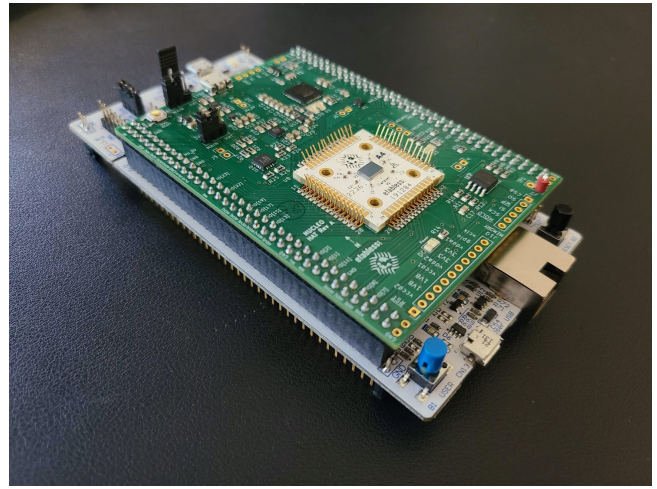
(Note: the power rails can be unconnected to some tiles/macros if they are not in the range of PDN pitch)




```
set script_dir [file dirname [file normalize [info script]]]
source $::env(CARAVEL_ROOT)/openlane/user_project_wrapper/fixed_wrapper_cfgs.tcl
source $::env(CARAVEL_ROOT)/openlane/user_project_wrapper/default_wrapper_cfgs.tcl

set ::env(DESIGN_NAME) user_project_wrapper
set ::env(FP_PDN_ENABLE_RAILS) 1
set ::env(GLB_RT_OBS) "met1 0 0 $::env(DIE_AREA),\
                       met2 0 0 $::env(DIE_AREA),\
                       met3 0 0 $::env(DIE_AREA),\
                       met4 0 0 $::env(DIE_AREA),\
                       met5 0 0 $::env(DIE_AREA)"

set ::env(CLOCK_PORT) "user_clock2"
set ::env(CLOCK_NET) "inst_eFPGA_top.user_clock2"
set ::env(CLOCK_PERIOD) "40"

set ::env(PL_OPENPHYSYN_OPTIMIZATIONS) 0
set ::env(DIODE_INSERTION_STRATEGY) 5
set ::env(MAGIC_WRITE_FULL_LEF) 0

set ::env(SYNTH_FLAT_TOP) 1
set ::env(CLOCK_TREE_SYNTH) 1
set ::env(DESIGN_IS_CORE) 1
set ::env(STA_REPORT_POWER) 0
set ::env(SYNTH_USE_PG_PINS_DEFINES) "USE_POWER_PINS"
set ::env(VDD_NETS) {vccd1 vdda1 vdda2 vccd2}
set ::env(GND_NETS) {vssd1 vssa1 vssa2 vssd2}
set ::env(VDD_PIN) "vccd1"
set ::env(GND_PIN) "vssd1"
set ::env(PL_TARGET_DENSITY) 0.45
set ::env(CTS_TARGET_SKEW) 200
set ::env(CTS_SINK_CLUSTERING_SIZE) 100
set ::env(CTS_SINK_CLUSTERING_MAX_DIAMETER) 1000
set ::env(ROUTING_CORES) 12
set ::env(GLB_RT_MAXLAYER) 5
set ::env(FP_PDN_CHECK_NODES) 0
set ::env(FP_PDN_IRDROP) 0

set ::env(FP_TAP_HORIZONTAL_HALO) 20
set ::env(FP_TAP_VERTICAL_HALO) 20
set ::env(FP_PDN_HORIZONTAL_HALO) 30
set ::env(FP_PDN_VERTICAL_HALO) 30
set ::env(FP_PDN_VWIDTH) 1.6
set ::env(FP_PDN_VPITCH) 2800

set ::env(SYNTH_READ_BLACKBOX_LIB) 1
set ::env(VERILOG_FILES) [glob $script_dir/../../verilog/rtl/defines.v $script_dir/../../../verilog/rtl/*.v ]
## Internal Macros
### Macro Placement
set ::env(MACRO_PLACEMENT_CFG) "$script_dir/../../../openlane/user_project_wrapper/macros/placements/macro_placement.cfg"
### Black-box verilog and views
set ::env(VERILOG_FILES_BLACKBOX) [glob $script_dir/../../../verilog/rtl/BB/*.v]
set ::env(EXTRA_LEFS) [glob $script_dir/../../../openlane/user_project_wrapper/macros/lef/*.lef]
set ::env(EXTRA_GDS_FILES) [glob $script_dir/../../../openlane/user_project_wrapper/macros/gds/*.gds]
### Macro PDN Connections
set ::env(FP_PDN_MACRO_HOOKS) "\
        inst_eFPGA_top.Inst_eFPGA.Tile_X0Y1_W_IO vccd1 vssd1 \
        inst_eFPGA_top.Inst_eFPGA.Tile_X0Y2_W_IO vccd1 vssd1 \
        inst_eFPGA_top.Inst_eFPGA.Tile_X0Y3_W_IO vccd1 vssd1 \
                                                              51,1          9%
```

eFPGA design flow using Openlane

5. Caravel Integration
6. Run local precheck
7. Submit precheck/tapeout job on eFabless portal
   >>> DONE !!!



MPW-7 | Open MPW

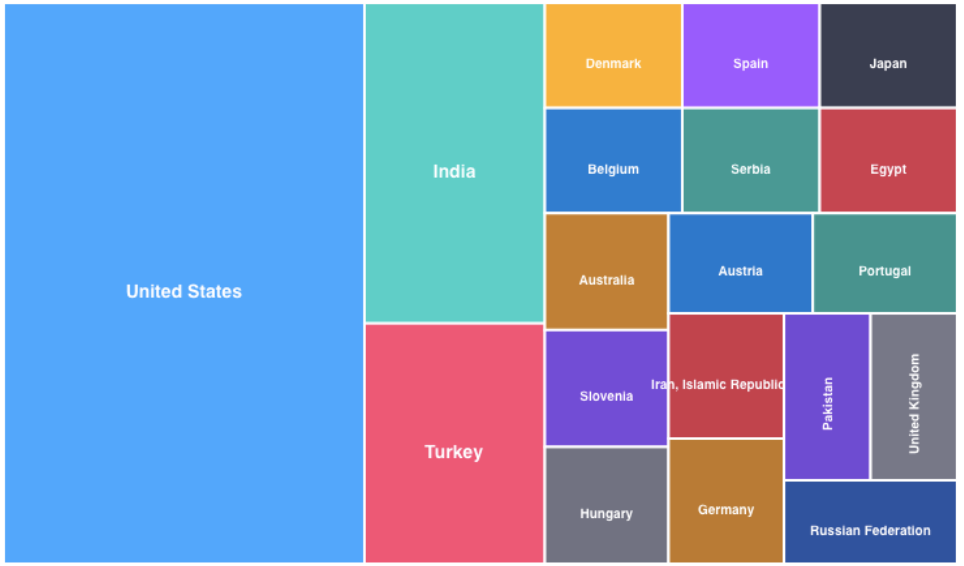
Process
SKY130B
Skywater 130nm

Participants
89
+ 37 New Users

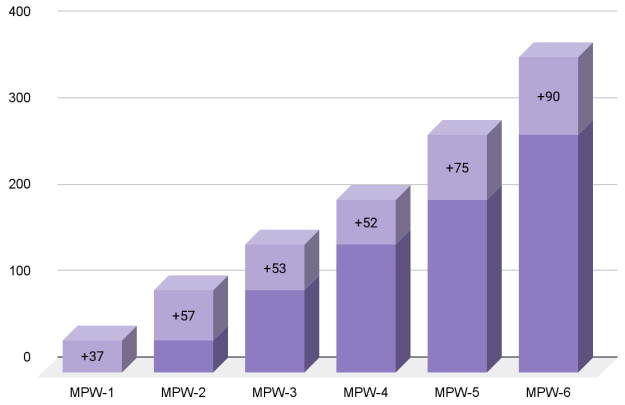
Projects
106
48 Categories

Capacity
106 / 40
265.0 %
Oversubscribed

Community Diversity for Projects Submitted (19)
Note - not all projects make it to fabrication

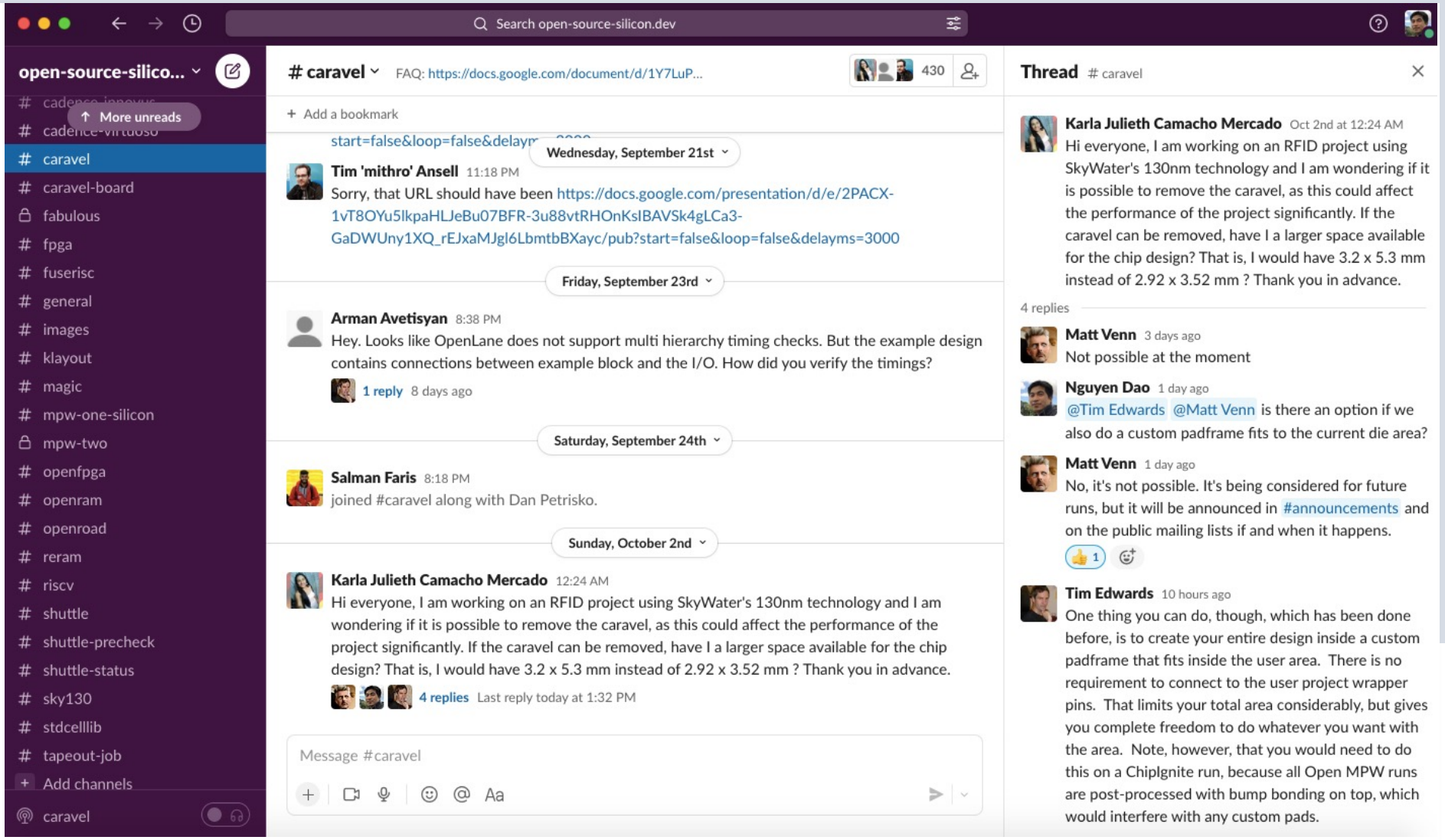


Total number of Open MPW designs



https://platform.efabless.com/shuttles/MPW-7?active_tab=summary

Questions?



https://join.slack.com/t/open-source-silicon/shared_invite/zt-1hb6gydjo-C2NCyrjGtkAwWcaaRTSbNQ