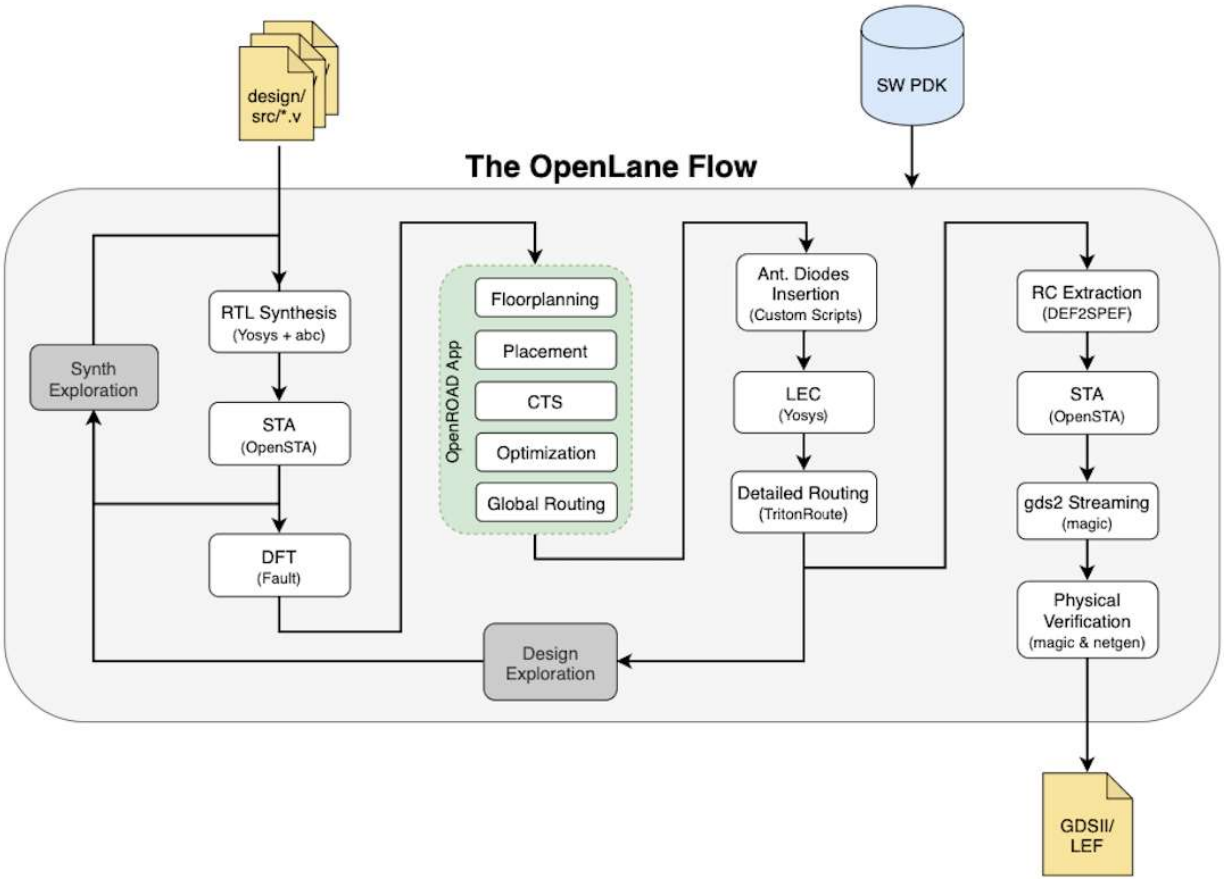
ASIC design flow for single cycle RISCV32I based processor using OpenLane



Synthesis -

Openlane uses yosys open source tool for netlist synthesis and abc open source tool for technology mapping.

In the OpenLane ASIC design flow, synthesis is the step where a Register Transfer Level (RTL) design written in a hardware description language (HDL) like Verilog or VHDL is converted into a gate-level netlist. The synthesized netlist contains a list of cells, each of which represents a logic gate in the design, along with their connections. The synthesis process also optimizes the design for timing, power, and area, and generates timing reports that help in further analysis and optimization. OpenLane uses the Yosys synthesis tool for RTL synthesis, which can also perform

technology mapping and optimize for area and timing constraints. After synthesis, the netlist is ready for the next step in the ASIC design flow, which is placement and routing.

Results

1) sdf file : sdf file is Standard Delay Format file . Header section of .sdf file contains several information like shown below

```
(DELAYFILE
 (SDFVERSION "3.0")
 (DESIGN "risc_v32")
 (DATE "Tue Jul 11 15:07:22 2023")
 (VENDOR "Parallax")
 (PROGRAM "STA")
 (VERSION "2.4.0")
 (DIVIDER .)
 (VOLTAGE 1.800::1.800)
 (PROCESS "1.000::1.000")
 (TEMPERATURE 25.000::25.000)
 (TIMESCALE 1ns)
 (CELL
  (CELLTYPE "risc_v32")
```

Voltage = 1.8v(delay depends on supply voltage).

Timescale = 1ns (unit of time)

Date and time of report creation .

The file also contains the information about the interconnect and standard cell delay.

The .v file contains the information about the technology mapping done by tool abc and this represents the gate level netlist in which gates will be represented by the standard cells.
The report section of synthesis contains the information about the number of wires used,number of standard cells of each type, core area etc.

Synthesis report

```
53. Printing statistics.

=== risc_v32 ===

   Number of wires:              14212
   Number of wire bits:          16326
   Number of public wires:          71
   Number of public wire bits:    2182
   Number of memories:               0
   Number of memory bits:            0
   Number of processes:              0
   Number of cells:              16312
     $_ANDNOT_                    2083
     $_AND_                         75
     $_MUX_                       5544
     $_NAND_                       116
     $_NOR_                        323
     $_NOT_                       3190
     $_ORNOT_                      404
     $_OR_                        1237
     $_XNOR_                       219
     $_XOR_                       1037
     sky130_fd_sc_hd__dfxtp_2     2080
     sky130_fd_sc_hd__dlxtn_1        4
```

Power Report

Group	Internal Power	Switching Power	Leakage Power	Total Power (Watts)	
Sequential	1.07e-02	6.94e-04	1.76e-08	1.14e-02	26.8%
Combinational	1.63e-02	1.48e-02	4.23e-08	3.11e-02	73.2%
Macro	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.0%
Pad	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.0%
Total	2.70e-02	1.55e-02	5.99e-08	4.25e-02	100.0%
	63.5%	36.5%	0.0%		

.

Floorplan -

Die area 0.0 0.0 546.55 557.27

Core_area 5.52 10.88 540.96 544.0

Openlane uses following tools to do the floorplan

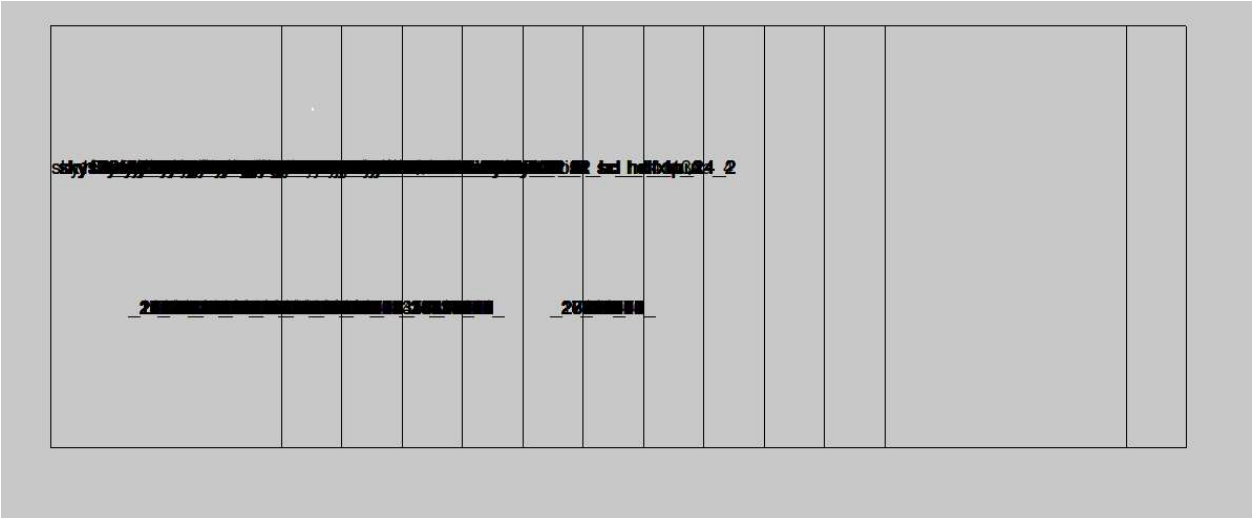
1. init_fp - Defines the core area for the macro as well as the rows (used for placement) and the tracks (used for routing)

2. ioplacer - Places the macro input and output ports

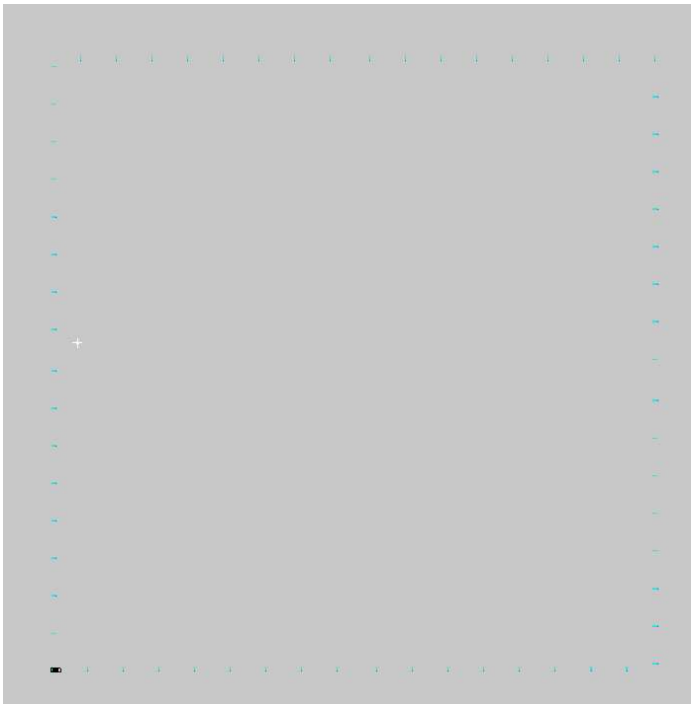
3. pdngen - Generates the power distribution network

4. tapcell - Inserts well tap and decap cells in the floorplan

1. init_fp: This is short for "initial floor plan". In chip design, a floor plan refers to the physical layout of the chip's components, including the placement of macros (large functional units, such as processors or memory blocks) and the routing of connections between them. The init_fp step defines the core area of the chip where the macros will be placed, as well as the rows (horizontal partitions) and tracks (vertical partitions) that will be used for routing.

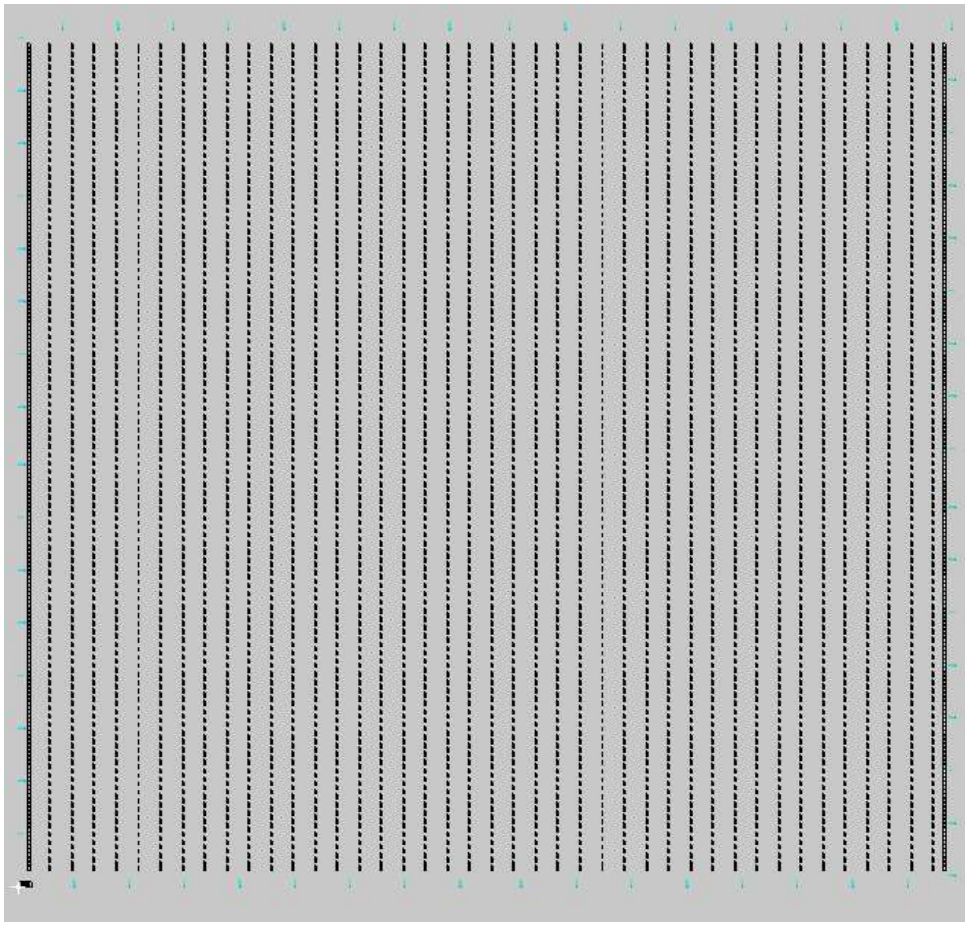


2. ioplacer: Once the core area and routing resources have been defined, the next step is to place the input and output ports of each macro. This is done by the ioplacer tool, which determines the optimal location for each port based on factors such as signal timing, power consumption, and physical proximity to other components.

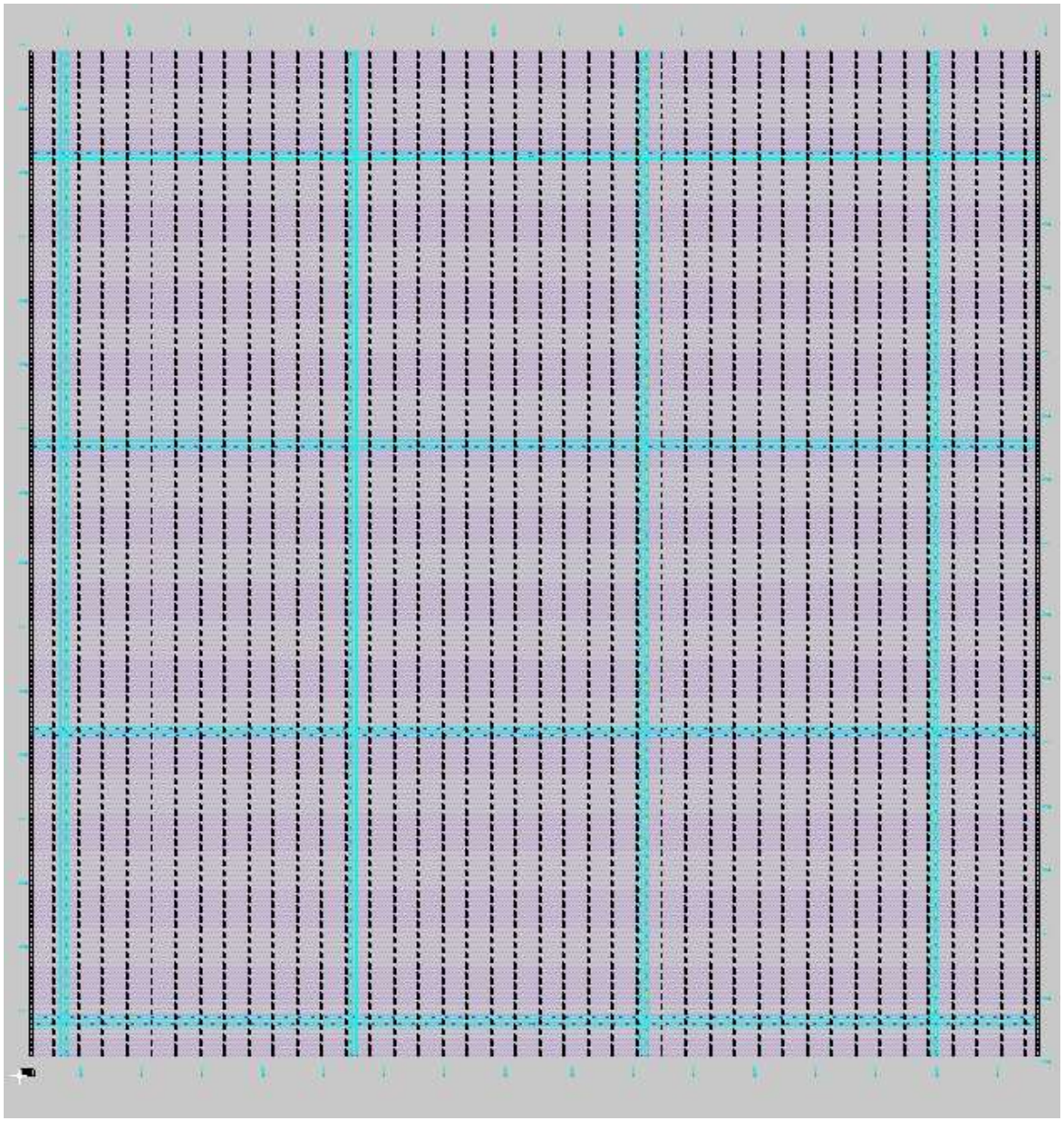


(blue dashed lines are visible in the above diagram in a rectangular shape)

3. pdngen: After the macros and their input/output ports have been placed, the next step is to generate the power distribution network. This network ensures that each component receives the correct amount of power, while minimizing power consumption and minimizing noise that can interfere with signal quality. The pdngen tool creates the network by placing power and ground pins throughout the chip and connecting them with metal routing layers.



4. tapcell: Finally, the tapcell tool inserts well tap and decap cells into the floorplan. Well tap cells provide a connection between the substrate of the chip and the power supply, which helps to prevent parasitic effects that can degrade signal quality. Decap cells are used to smooth out fluctuations in the power supply, which can occur due to changes in the activity level of different parts of the chip. The tapcell tool determines the optimal locations for these cells based on the layout of the other components in the chip.



Placement -

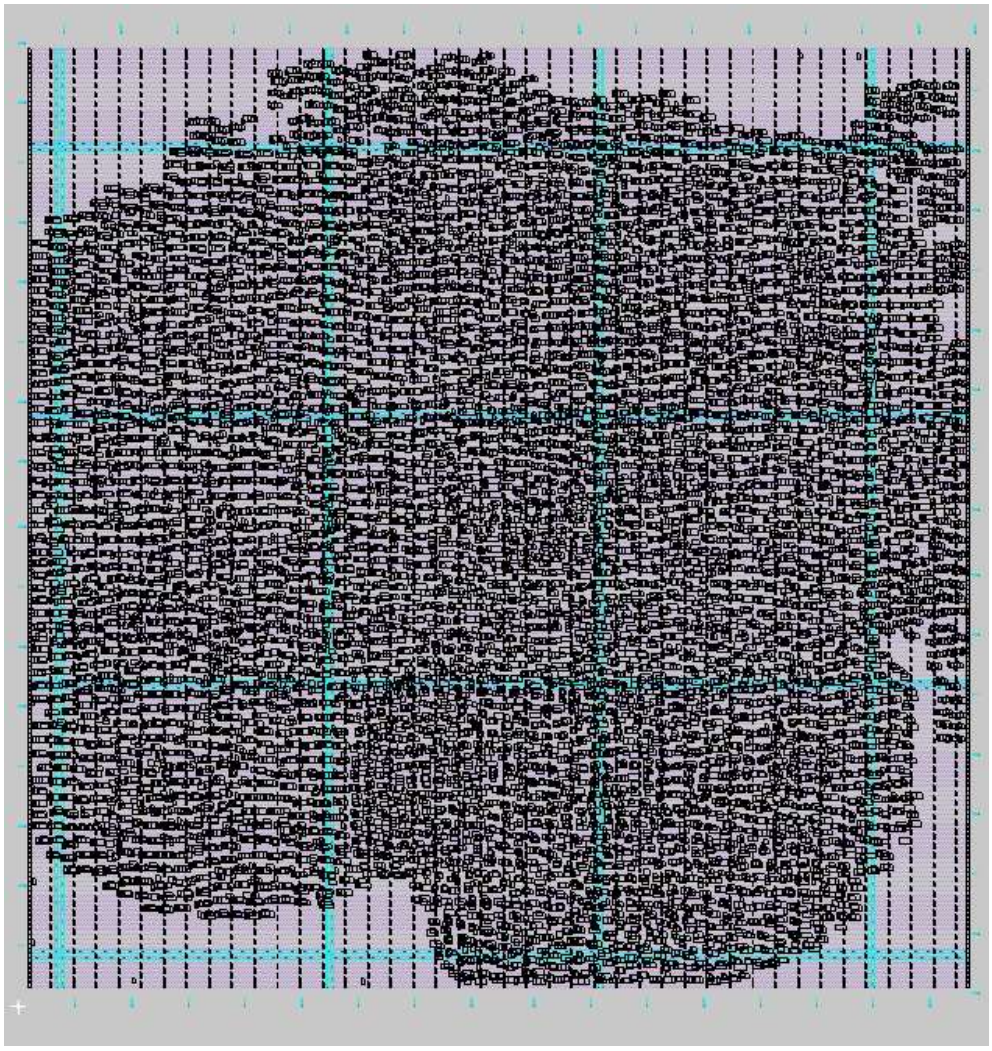
Placement is a critical step in the physical design of a chip, where the position of each component is determined to optimize the chip's performance and minimize its area and power consumption. There are several tools used in the placement process, including RePLace, Resizer, and OpenDP.

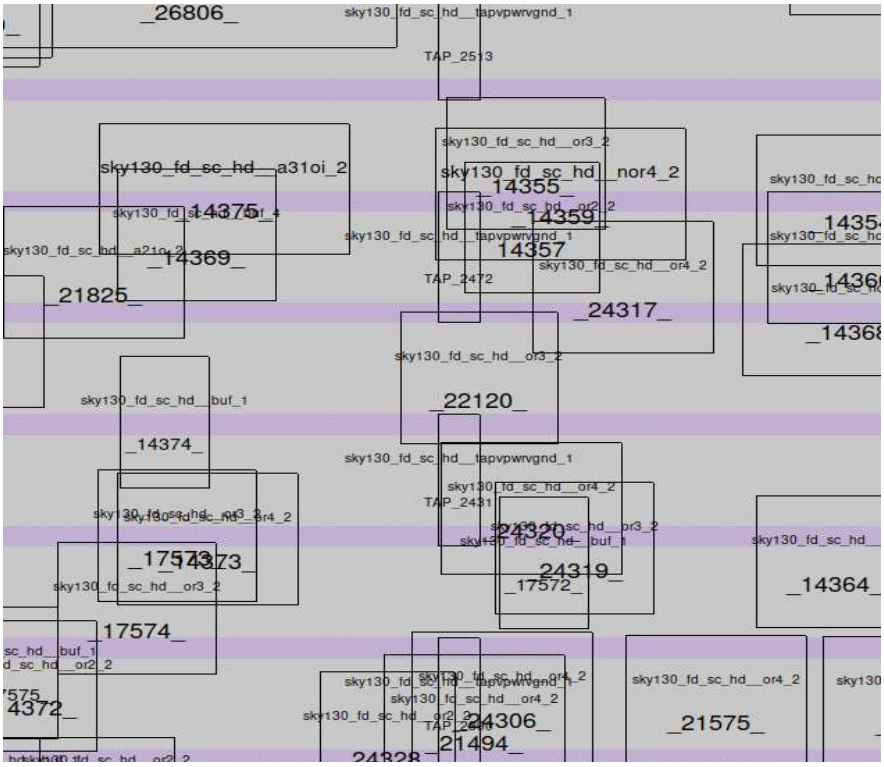
1. RePLace - Performs global placement

2. Resizer - Performs optional optimizations on the design

3. OpenDP - Performs detailed placement to legalize the globally placed components

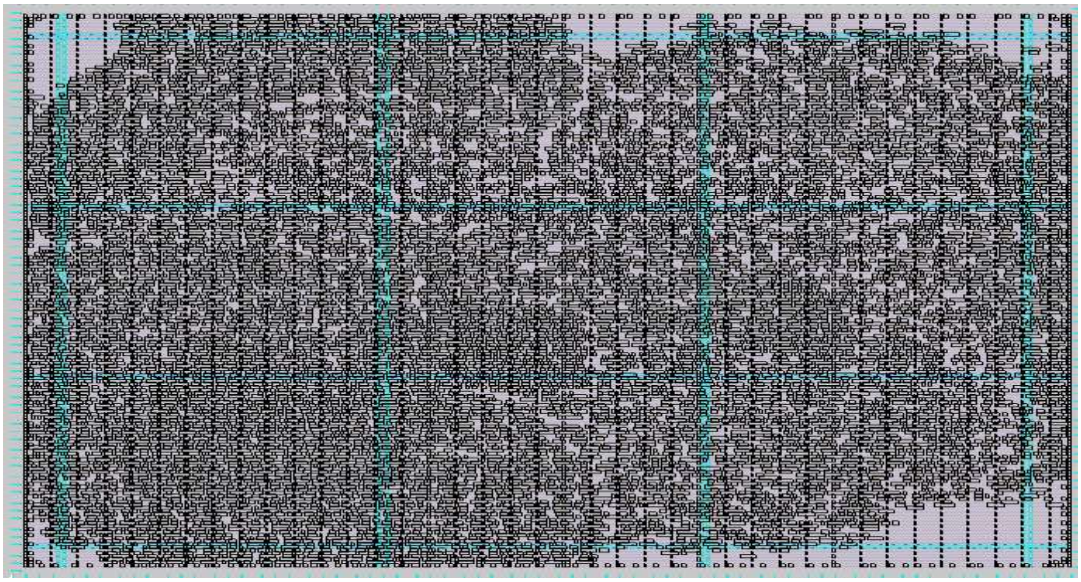
1. RePLace: This tool performs global placement, which involves determining the approximate location of each component on the chip's surface. The goal is to minimize the total wire length of the interconnects between components, as longer wires can introduce signal delays and increase power consumption. RePLace uses algorithms such as simulated annealing and gradient descent to iteratively adjust the placement of components until a satisfactory result is achieved.



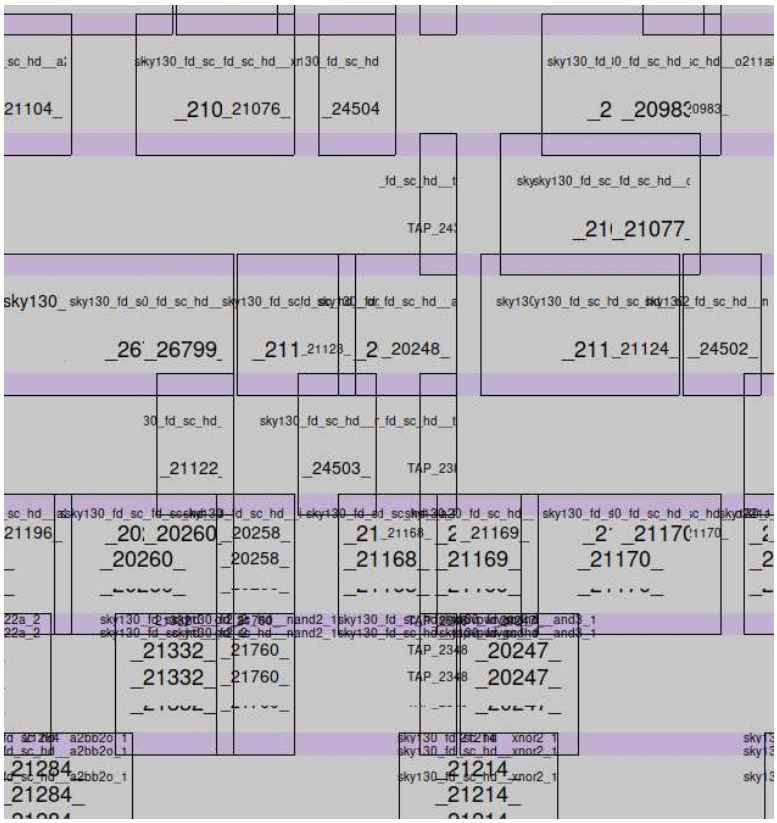


(zoomed in version of above image)

2. Resizer: After global placement, Resizer is used to perform optional optimizations on the design. These optimizations can include buffering, sizing, and cell swapping to improve timing, area, and power consumption. Resizer can also generate multiple placement solutions with different trade-offs to help designers select the optimal design.



3. OpenDP: Finally, OpenDP performs detailed placement to legalize the globally placed components. This involves ensuring that the placement is compliant with design rules, such as minimum spacing between components and metal routing layers. OpenDP also optimizes the placement of standard cells and macros to further improve the chip's performance and minimize its area and power consumption.

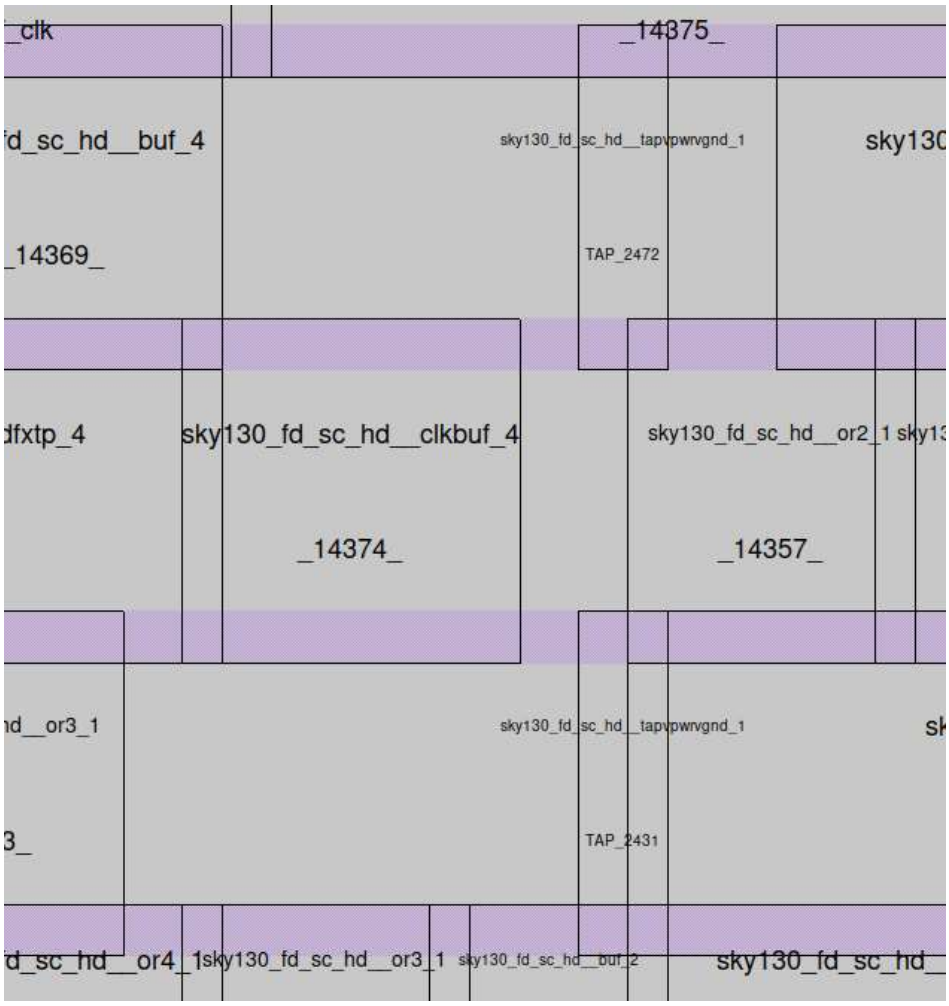


Clock Tree Synthesis (CTS) -

In digital chip design, clock distribution is a crucial part of the design process as it determines the timing accuracy and synchronicity of the entire system. The Clock Tree Synthesis (CTS) stage creates a hierarchical tree-like structure that distributes the clock signal to all sequential elements in the design with minimum clock skew and power dissipation.

One of the tools used for CTS is TritonCTS, which is a tool used for synthesizing the clock distribution network (the clock tree). It takes the logical netlist of the design and creates a clock tree with low skew, minimal delay, and low power consumption.

TritonCTS uses advanced algorithms to create the clock tree such as Steiner routing algorithms, buffer insertion, and wire sizing to meet the timing constraints, improve signal integrity, and reduce power consumption. Additionally, it performs optimization such as buffer insertion, clock gating, and power optimization, which helps to minimize clock power and reduce the dynamic power of the chip.



Clk buffer got inserted in the above step.

Below is the screenshot of def file after cts , where clk buffers locations are specified

```
- _17147_ sky130_fd_sc_hd__clkbuf_1 + PLACED ( 181700 193120 ) S ;
- _17148_ sky130_fd_sc_hd__mux2_1 + PLACED ( 181700 144160 ) FS ;
- _17149_ sky130_fd_sc_hd__clkbuf_1 + PLACED ( 181240 149600 ) FS ;
- _17150_ sky130_fd_sc_hd__mux2_1 + PLACED ( 173420 163200 ) N ;
- _17151_ sky130_fd_sc_hd__clkbuf_1 + PLACED ( 170660 163200 ) N ;
- _17152_ sky130_fd_sc_hd__mux2_1 + PLACED ( 181240 201280 ) N ;
- _17153_ sky130_fd_sc_hd__clkbuf_1 + PLACED ( 181240 206720 ) N ;
- _17154_ sky130_fd_sc_hd__mux2_1 + PLACED ( 181240 103360 ) N ;
- _17155_ sky130_fd_sc_hd__clkbuf_1 + PLACED ( 181240 108800 ) N ;
- _17156_ sky130_fd_sc_hd__mux2_1 + PLACED ( 179860 78880 ) S ;
- _17157_ sky130_fd_sc_hd__clkbuf_1 + PLACED ( 184000 78880 ) S ;
- _17158_ sky130_fd_sc_hd__mux2_1 + PLACED ( 181240 84320 ) FS ;
- _17159_ sky130_fd_sc_hd__clkbuf_1 + PLACED ( 179400 84320 ) S ;
- _17160_ sky130_fd_sc_hd__mux2_1 + PLACED ( 177560 89760 ) FS ;
- _17161_ sky130_fd_sc_hd__clkbuf_1 + PLACED ( 171580 87040 ) N ;
- _17162_ sky130_fd_sc_hd__mux2_1 + PLACED ( 173420 114240 ) N ;
- _17163_ sky130_fd_sc_hd__clkbuf_1 + PLACED ( 171580 114240 ) N ;
- _17164_ sky130_fd_sc_hd__mux2_1 + PLACED ( 155940 239360 ) FN ;
- _17165_ sky130_fd_sc_hd__clkbuf_1 + PLACED ( 158240 244800 ) N ;
- _17166_ sky130_fd_sc_hd__mux2_1 + PLACED ( 163760 193120 ) FS ;
- _17167_ sky130_fd_sc_hd__clkbuf_1 + PLACED ( 162840 198560 ) FS ;
- _17168_ sky130_fd_sc_hd__or3_2 + PLACED ( 122360 204000 ) S ;
- _17169_ sky130_fd_sc_hd__buf_4 + PLACED ( 100280 231200 ) S ;
- _17170_ sky130_fd_sc_hd__mux2_1 + PLACED ( 106260 310080 ) N ;
- _17171_ sky130_fd_sc_hd__clkbuf_1 + PLACED ( 106260 312800 ) S ;
- _17172_ sky130_fd_sc_hd__mux2_1 + PLACED ( 95220 383520 ) FS ;
- _17173_ sky130_fd_sc_hd__clkbuf_1 + PLACED ( 94300 388960 ) S ;
- _17174_ sky130_fd_sc_hd__mux2_1 + PLACED ( 73600 350880 ) FS ;
- _17175_ sky130_fd_sc_hd__clkbuf_1 + PLACED ( 73140 356320 ) FS ;
- _17176_ sky130_fd_sc_hd__mux2_1 + PLACED ( 74060 380800 ) N ;
```

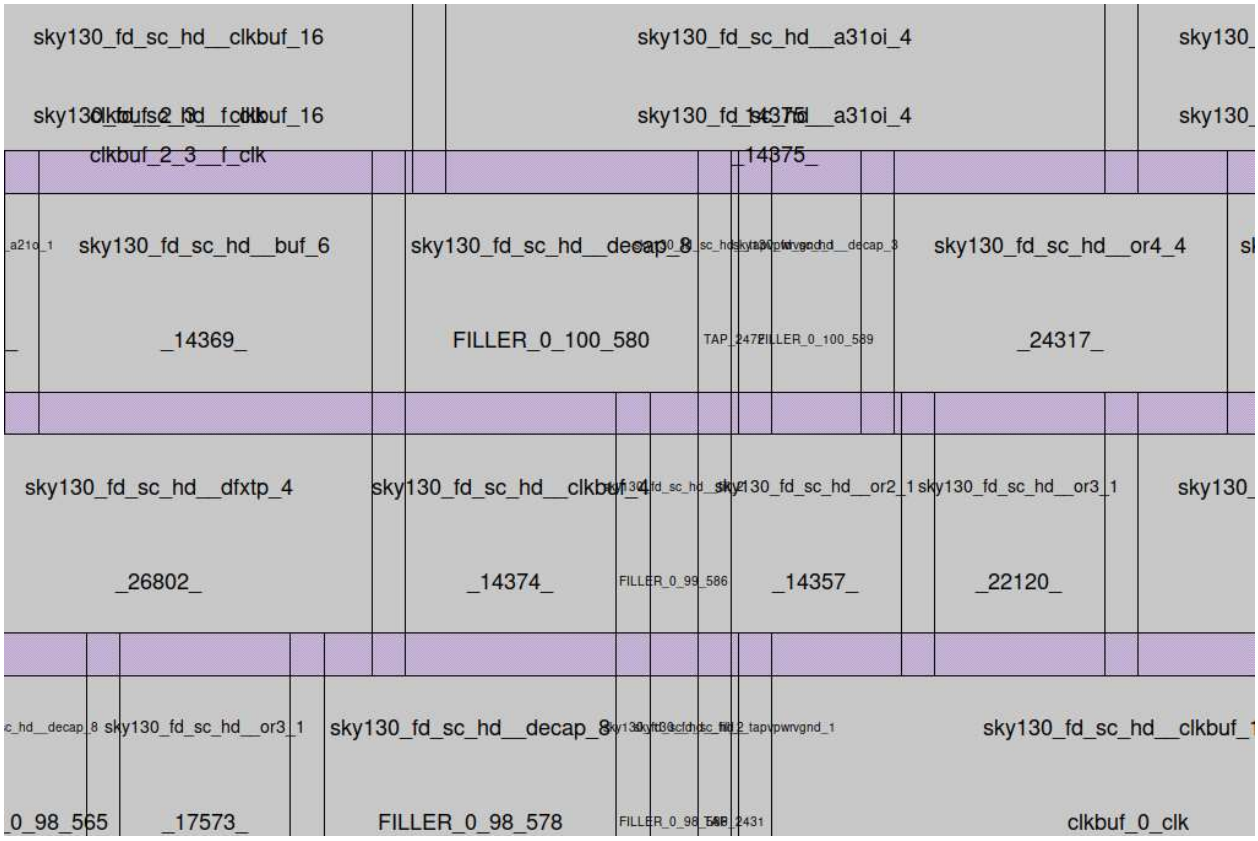
Routing -

Routing is a crucial step in the physical design of integrated circuits (ICs). It involves connecting the various components of the IC layout through metal wires to create a functioning circuit.

There are different stages involved in routing, including global routing, detailed routing, and SPEF (Standard Parasitic Extraction Format) extraction. Here's a brief overview of the three tools you mentioned:

1. FastRoute is a tool used for global routing. Global routing involves determining the approximate path that interconnects various components of an IC. It is a critical step in IC design because it impacts the timing, power, and area of the circuit. FastRoute is designed

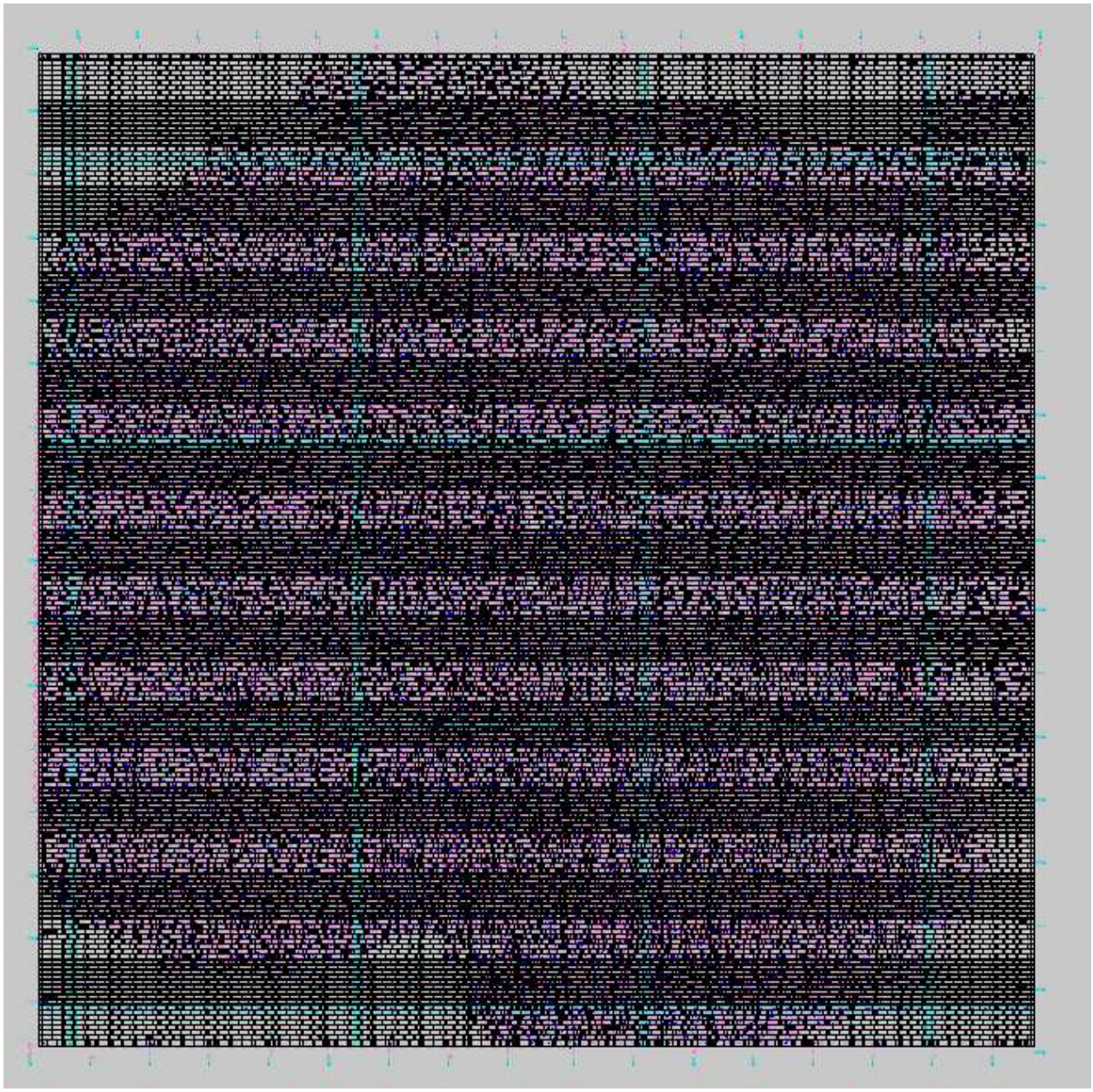
to perform fast and efficient global routing, generating a guide file that is used by the detailed router.

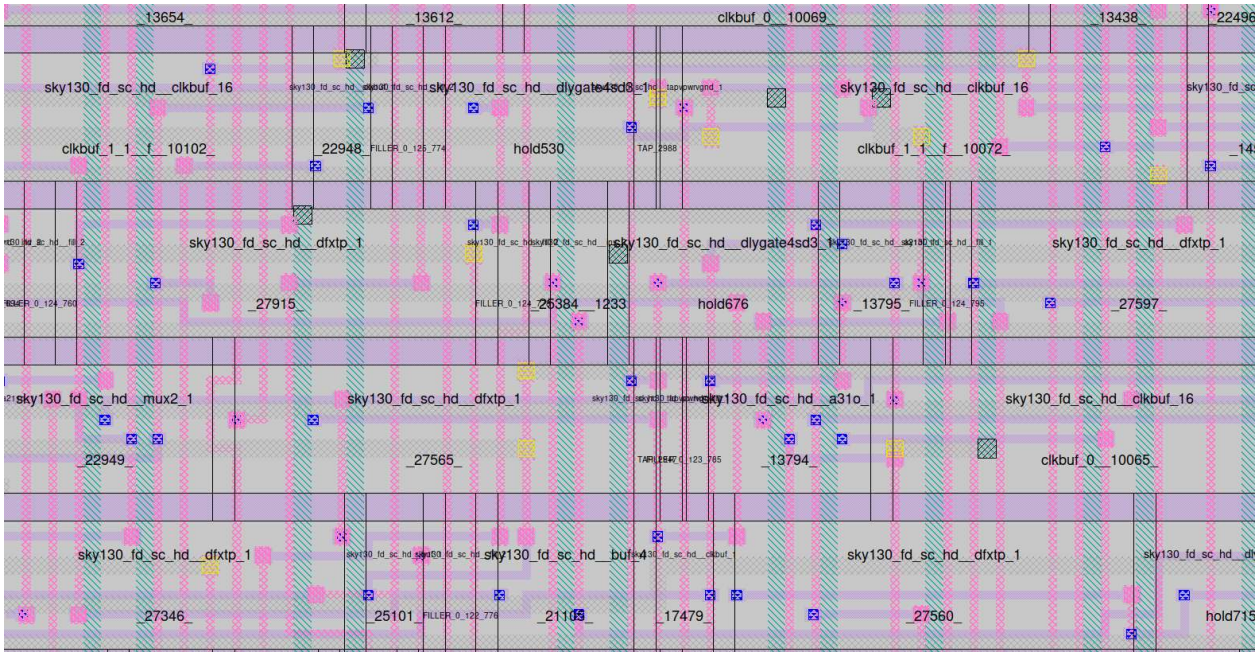


Filler cells are cells that are inserted into integrated circuits (ICs) to fill empty spaces or gaps between functional cells. They are also known as "dummy cells" or "spacer cells". Filler cells are typically used in the design of digital ICs to improve chip density, reduce timing violations, and improve manufacturing yields.

2. TritonRoute is a tool used for detailed routing. It involves routing each individual wire in the IC layout, based on the global route generated by FastRoute. TritonRoute is designed to perform fast and accurate detailed routing, ensuring that the connections between components are optimized for timing, power, and area.

3. OpenRCX is a tool used for SPEF extraction. SPEF extraction involves extracting the parasitic capacitances and resistances of the wires and other components in the circuit. These parasitics impact the timing and performance of the circuit, so it's important to extract them accurately. OpenRCX is designed to extract the SPEF format from the routed layout, which is used by other tools in the physical design flow.





Tapeout -

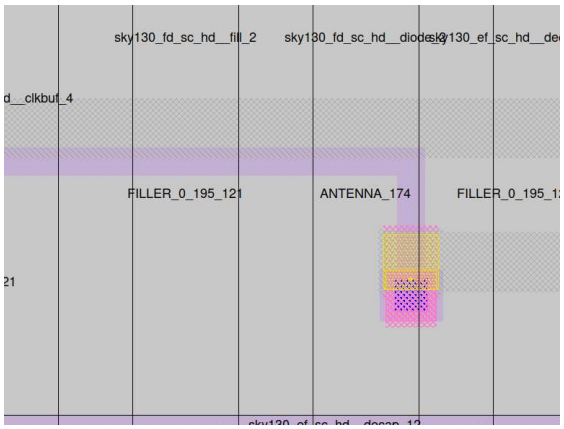
In the semiconductor industry, "tapeout" refers to the process of generating the final version of a chip's design layout in a format called GDSII, which is then used to manufacture the chip. The term originated from the practice of physically assembling the design files onto a magnetic tape before sending it to a semiconductor foundry for fabrication.

The tapeout process typically involves several stages of design verification, including circuit simulations, physical layout checks, and electrical rule checks, to ensure that the design is free of errors or manufacturing defects. Once the design is deemed ready for fabrication, the tapeout process involves converting the design files into the GDSII format, which is a standardized layout description language used by most foundries.

The final GDSII file contains all the information necessary to create the physical mask that will be used to pattern the chip's features onto a silicon wafer during the manufacturing process. Tapeout is a critical milestone in the chip design process, as it marks the point at which the design is frozen and ready for fabrication.

Signoff -

1. Magic - Performs DRC Checks & Antenna Checks



2. KLayout - Performs DRC Checks

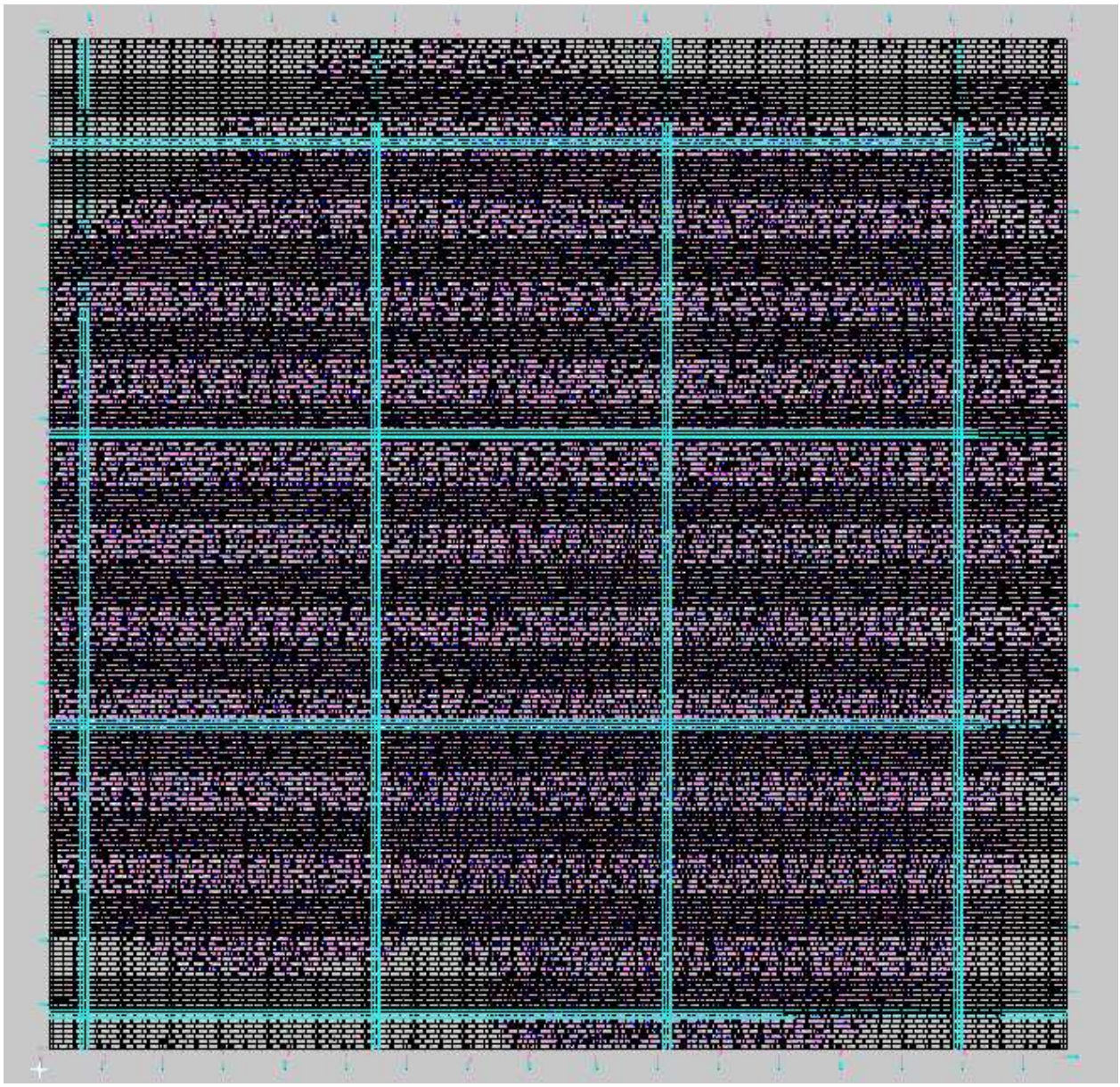
3. Netgen - Performs LVS Checks

   LVS reports no net, device, pin, or property mismatches.

   Total errors = 0

4. CVC - Performs Circuit Validity Checks

Final GDS view!!!



References

https://openlane.readthedocs.io/en/latest/reference/openlane_commands.html