MACQUARIE University

SILICON PLATFORMS LAB

SPI Design using openROAD

This document provides a step-by-step guide on the process from RTL synthesis to GDS generation using OpenROAD, demonstrated through the design of a SPI (Serial Peripheral Interface) block with GlobalFoundries 180nm (GF180) technology.

SPI (Serial Peripheral Interface)

The Serial Peripheral Interface (SPI) is a synchronous serial communication interface commonly used for short-distance communication, primarily in embedded systems. It was originally developed by Motorola in the late 1980s and has since become a standard for connecting microcontrollers to peripherals like sensors, memory chips, displays, and other devices.

How SPI Works

SPI operates in a master-slave configuration, where one device (the master) controls the communication, and one or more devices (slaves) respond to the master's commands. It uses a full-duplex communication model, meaning data can be sent and received simultaneously.

Key Features

1. Four Signal Lines:
   - SCLK (Serial Clock): Generated by the master to synchronize data transmission.
   - MOSI (Master Out Slave In): Data line from the master to the slave.
   - MISO (Master In Slave Out): Data line from the slave to the master.
   - CS/SS (Chip Select or Slave Select): A dedicated line for each slave, used by the master to select which slave to communicate with (active low in most cases).

2. Synchronous Communication: Data transfer is timed with the clock signal.
3. High Speed: SPI typically supports faster data rates than I2C.

Advantages of SPI

- Simple and straightforward protocol.
- High-speed data transfer.
- Full-duplex communication.
- Flexible data frame sizes (not fixed to 8 bits).

Disadvantages

- Requires more pins than I2C (especially with multiple slaves due to separate CS lines).
- No built-in error checking (e.g., no acknowledgment mechanism).
- Short-distance communication (typically within a single PCB or a few centimeters).

Common Applications

- Interfacing microcontrollers with SD cards, flash memory, or EEPROM.
- Connecting to sensors like accelerometers, gyroscopes, or temperature sensors.
- Driving displays (e.g., TFT or OLED screens).
- Communication between microcontrollers in multi-processor systems.

SPI Design using OpenROAD

Design setup and run

1) The SPI block is designed using Global foundries 180nm, the constraints, config files, log files, reports and results will be available inside the **gf180** directory which is the **PLATFORM**
2) The netlists should be in the **/foss/designs/flow/designs/src/spi** directory
3) The constraint and design config files should be in **/foss/designs/flow/designs/gf180/spi**
4) In openROAD flow, the constraint and the design config file (config.mk) should be in the **/foss/designs/flow/designs/<PLATFORM>/<DESIGN_NAME>** directory.
5) Design variables must be defined in the design config **config.mk** file in the **/foss/designs/flow/designs/<PLATFORM>/<DESIGN_NAME>** directory for each design.
6) Design variables with explanation is available in this link - Environment Variables for the OpenROAD
7) The autotuner.json file should be in **/foss/designs/flow/designs/gf180/spi** directory.
    a. The autotuner.json file is typically related to OpenROAD's autotuning feature, which optimizes certain parameters for physical design tasks. The autotuner.json file is specific for each process node.
8) The rules-base.json file should be in **/foss/designs/flow/designs/gf180/spi** directory.
    a. rules-base.json file typically contains a set of design rules and constraints that guide the optimization and physical design process for specific process nodes.
        i. **Layer usage rules:** Constraints on the use of different metal layers for routing and placement.
        ii. **Minimum spacing:** Rules that define the minimum allowable distance between two objects (such as nets or vias).
        iii. **Design rule violations:** These rules help in identifying and correcting any violations of the specified design constraints during the flow.
        iv. **Technology parameters:** Rules relating to the physical properties of the technology being used (e.g., transistor size, gate length, etc.).
9) The flow scripts are available in the **/foss/designs/flow/scripts** directory. Modification of files in the scripts directory is not permitted. Customization is only possible through the **config.mk** file available in the **/foss/designs/flow/designs/<PLATFORM>/<DESIGN_NAME>** directory
10) Timing and design results will be generated in the **/foss/designs/flow/reports/<PLATFORM>/<DESIGN_NAME>** directory.
11) The log files will be generated in **/foss/designs/flow/logs/<PLATFORM>/<DESIGN_NAME>**
12) The synthesized netlist, ODB design databases, final def, final spef and GDS will be generated in the **/foss/designs/flow/results/<PLATFORM>/<DESIGN_NAME>** directory.
13) The design configuration file must be defined in **Makefile** in the **/foss/designs/flow** directory to ensure that the desired design is run in openROAD using the correct netlist and constraints. **DESIGN_CONFIG=./designs/gf180/spi/config.mk**

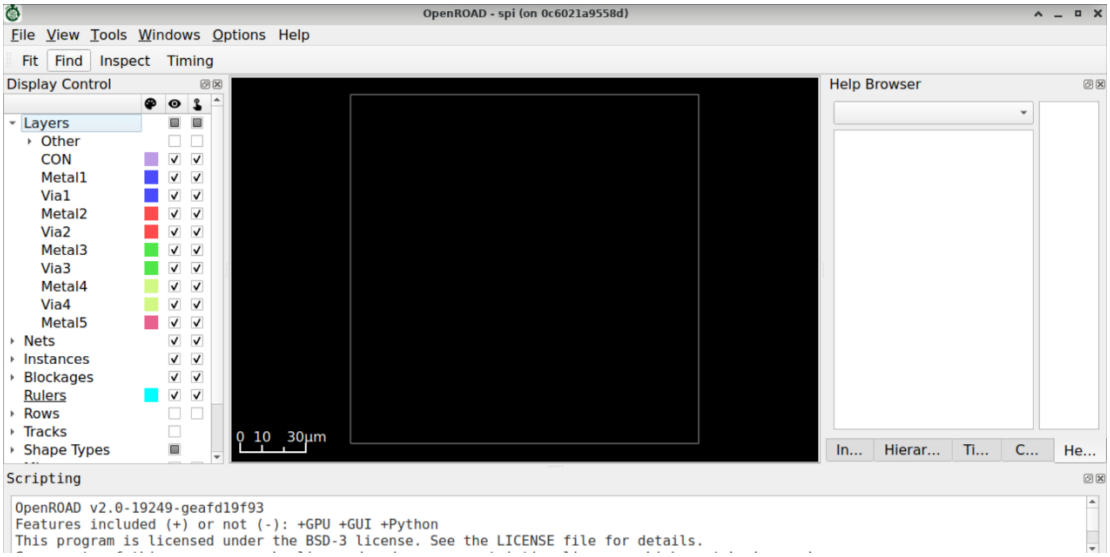
14) At the end of the setup phase, you should have the four files as below
    a. **/foss/designs/flow/designs/gf180/spi**
        i. rules-base.json
        ii. autotuner.json
        iii. constraint.sdc
        iv. config.mk
15) Run **make** in the **/foss/designs/flow** directory
16) OpenROAD is an automated flow and the GDS will be generated in the **results** directory.

RTL to GDS flow - OpenROAD

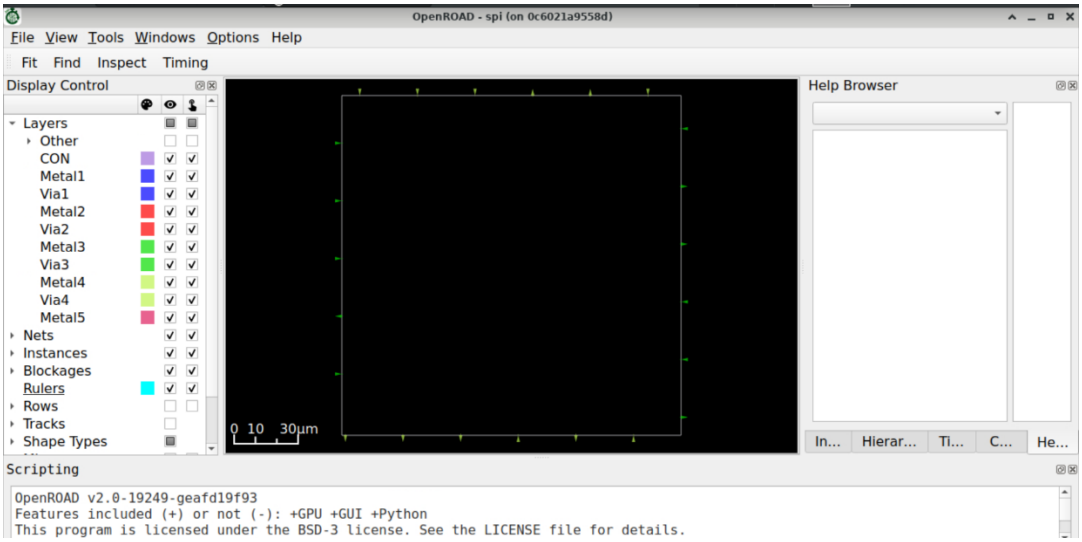
1) Detailed explanation on steps involved in the openROAD flow are available in the link: *OpenROAD Flow Scripts Tutorial*
2) openROAD flow includes a sequence of steps in the RTL to GDS flow.
    a. 1_1_Yosys - **RTL synthesis**
    b. 2_1_floorplan – **Initialize floorplan and size**
    c. 2_2_floorplan_io – **IO pin initialization**
    d. 2_3_floorplan_macro – **macro placement**
    e. 2_4_floorplan_tapcell – **tapcell placement**
    f. 2_5_floorplan_pdn – **power distribution**
    g. 2_flooplan – **full floorplan**
    h. 3_1_place_gp_skip_io – **global placement skipping IO pin placement**
    i. 3_2_place_iop – **IO pin placement**
    j. 3_3_place_gp – **Global placement**
    k. 3_4_place_resized – **legalization and resizing of cells**
    l. 3_5_place_dp – **Detailed placement**
    m. 3_place – **final placement**
    n. 4_1_cts – **cts**
    o. 4_cts – **final cts**
    p. 5_1_grt – **Global route**
    q. 5_2_route – **Detailed route**
    r. 5_3_fillcell – **Filler cell insertion**
    s. 5_route – **Final route**
    t. 6_1_fill – **metal fill**
    u. 6_final – **final database**
    v. 6_1_merge – **chip finishing GDS**

3) openroad -gui is the command used to open the GUI.
4) In the flow directory, use **$TOOLS/openroad-latest/bin/openroad -gui**
5) Select file -> OpenDB -> results and then the respective odb to analyze the GUI

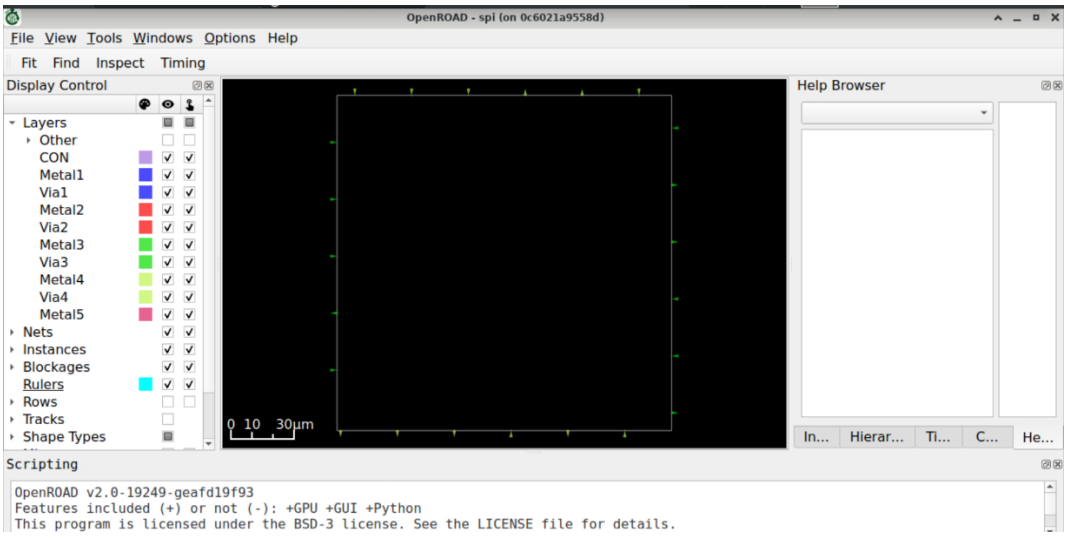
6) 2_1_floorplan – **floorplan is initialized, and size is finalized**



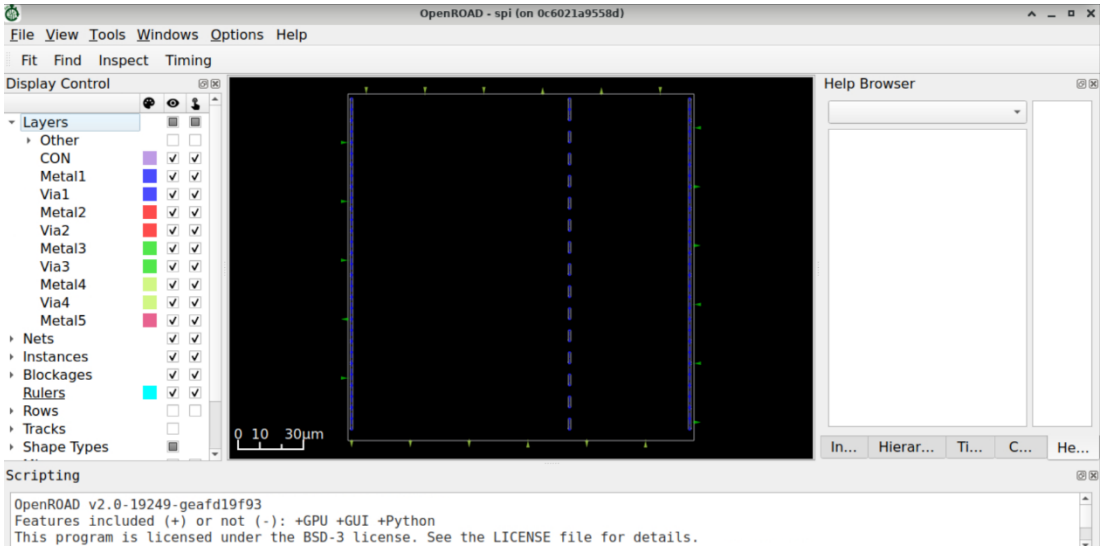
7) 2_2_floorplan_io – **IO pin initialization**



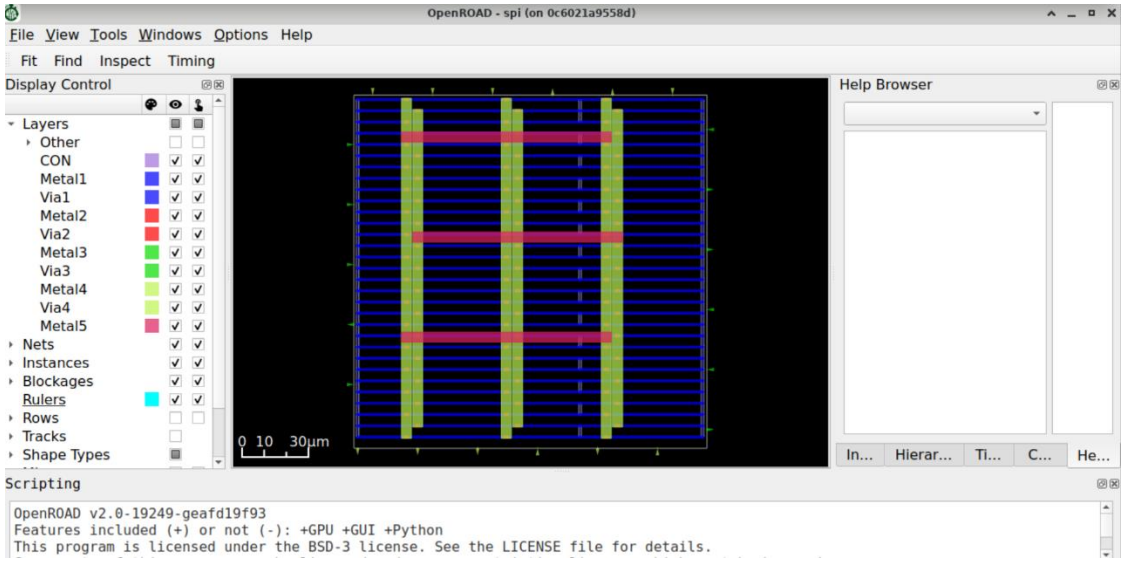
8) 2_3_floorplan – **macro placement**



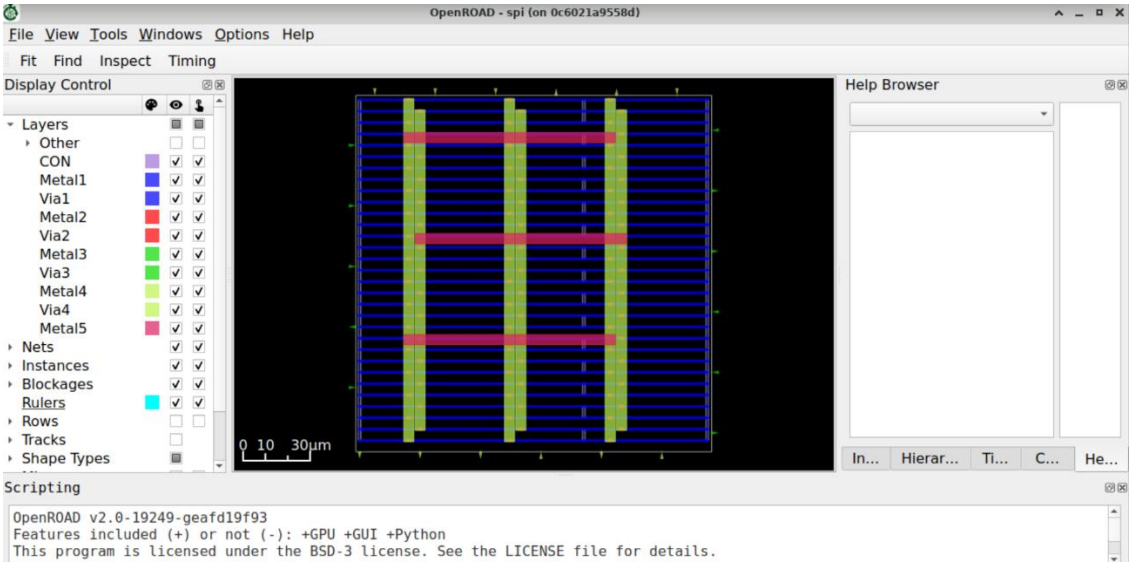
9) 2_4_floorplan– **tapcell placement**



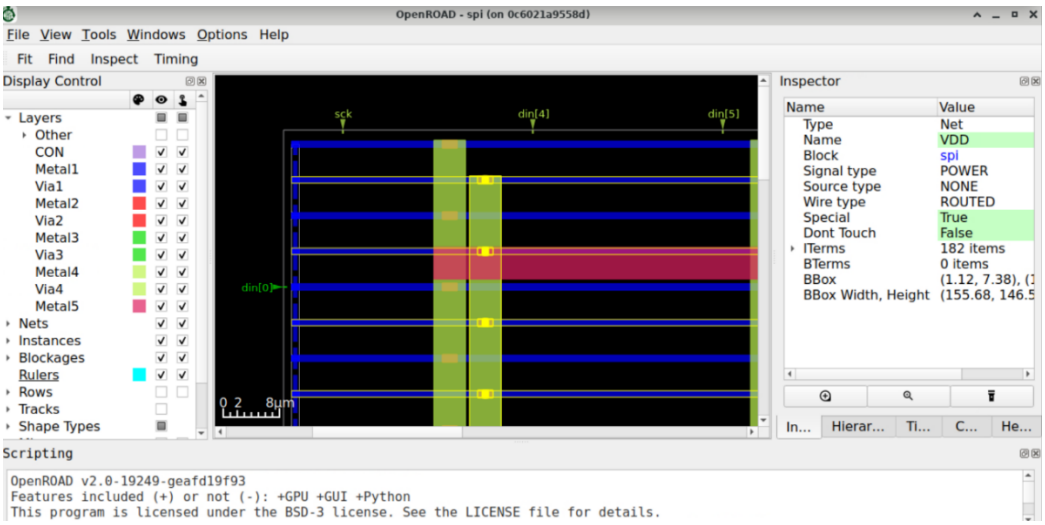
10) 2_5_floorplan_pdn – **power distribution**



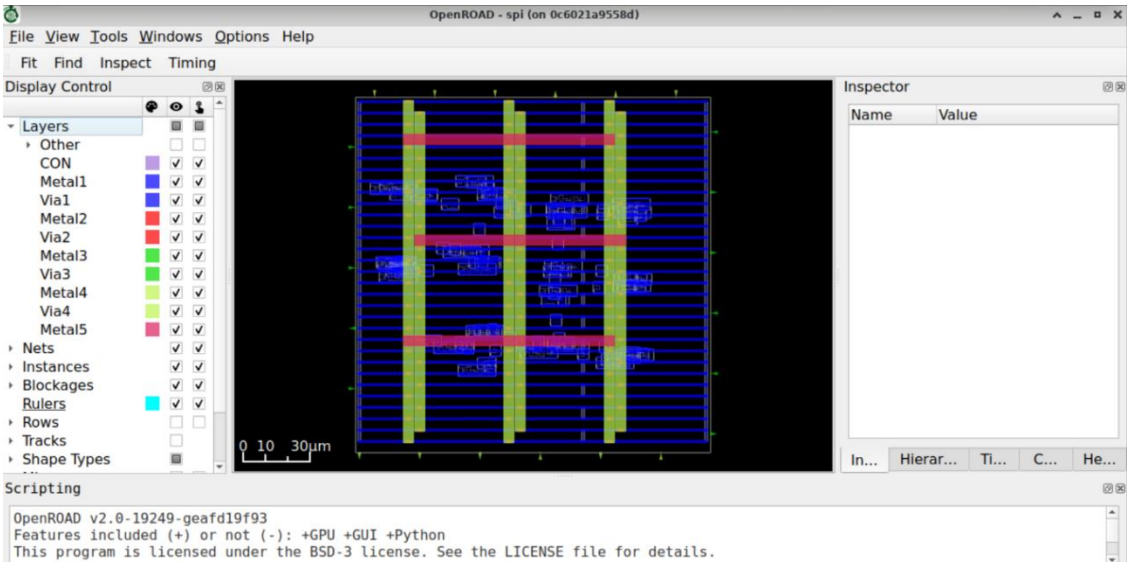
11) 2_flooplan – **full floorplan**



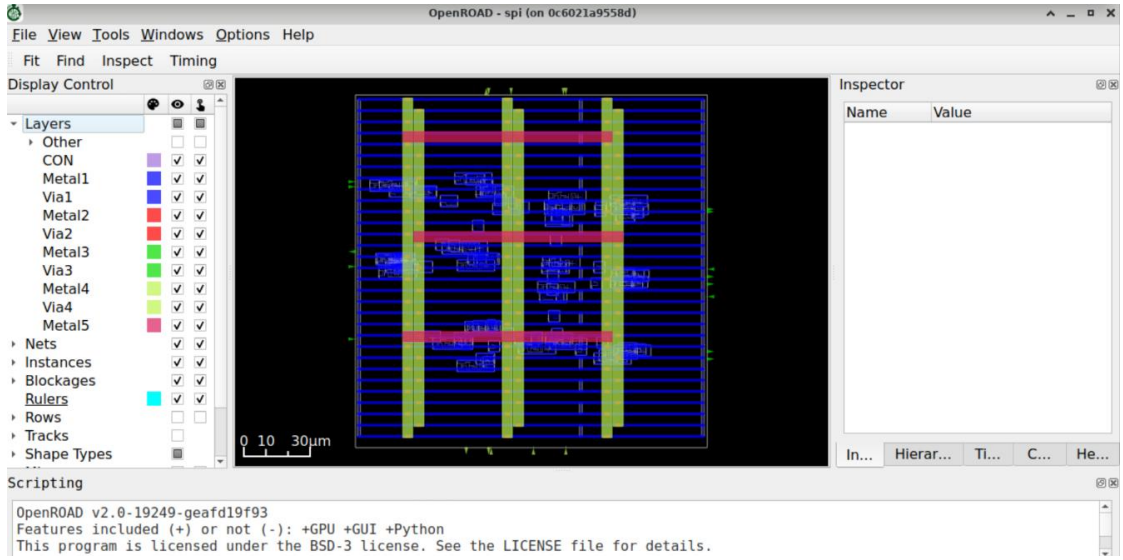
12) 2_flooplan – **full floorplan** (specific metal layers can be viewed by disabling the rest - M5 and M1 is enabled and the VDD power grid is highlighted)



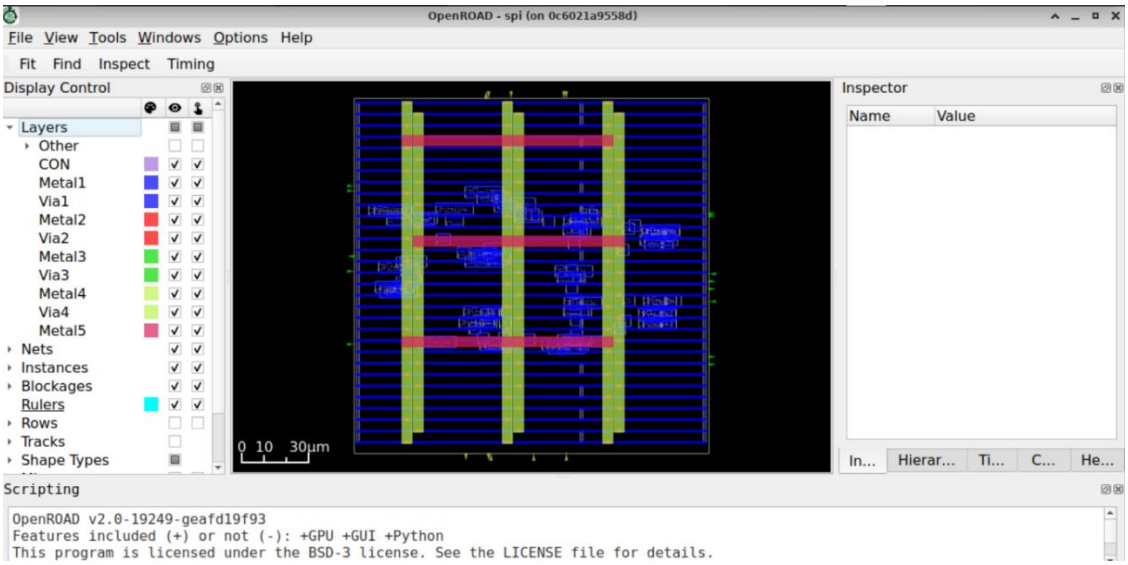
13) 3_1_place_gp_skip_io – **global placement skipping IO pin placement**



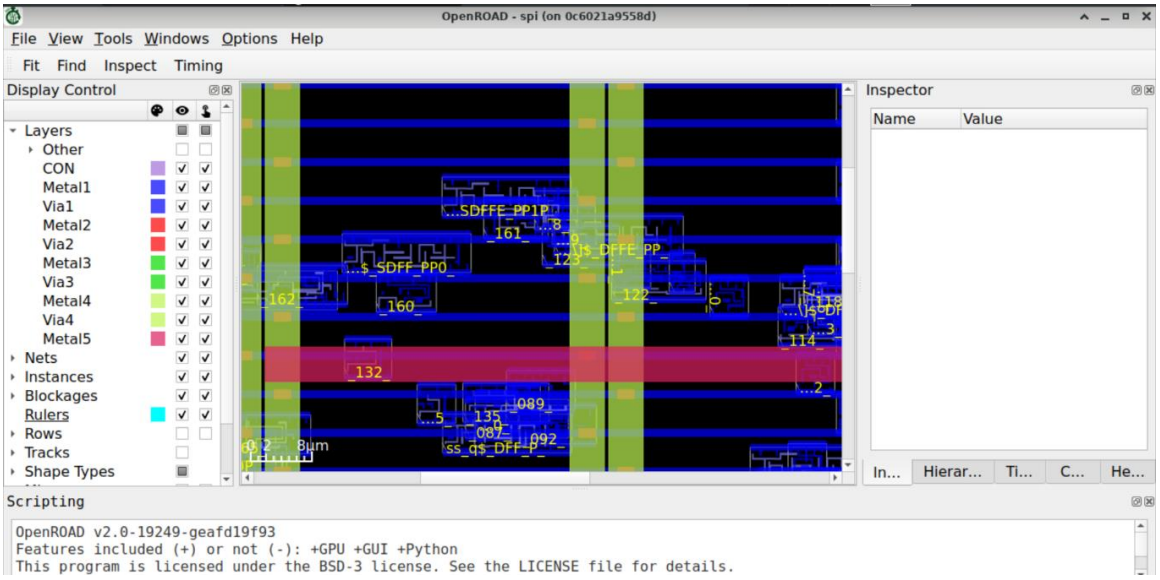
14) 3_2_place_iop – **IO pin placement**



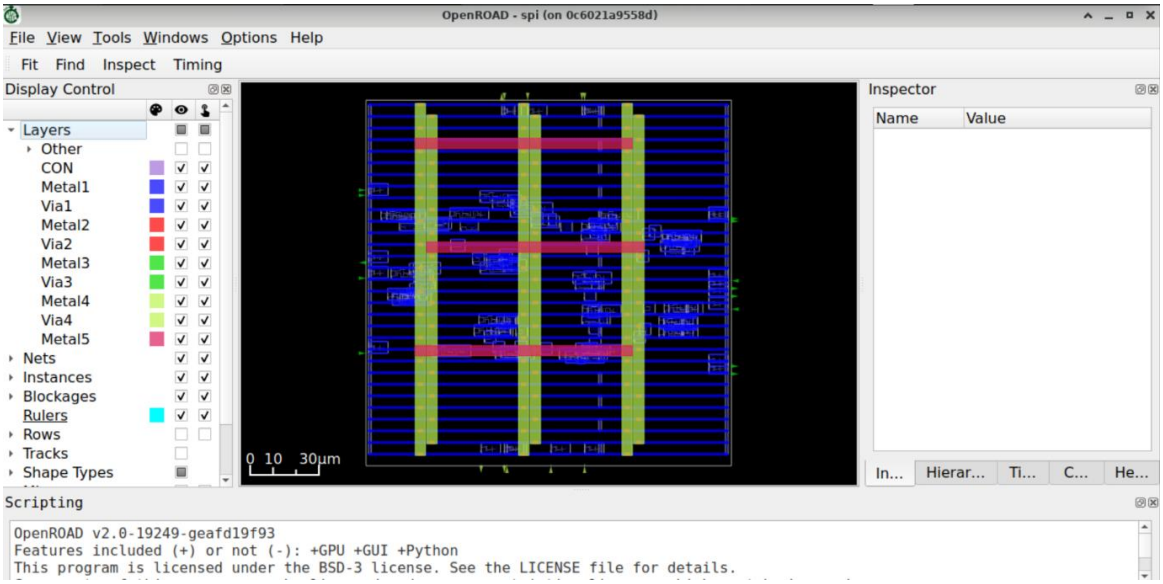
15) 3_3_place_gp – **Global placement**



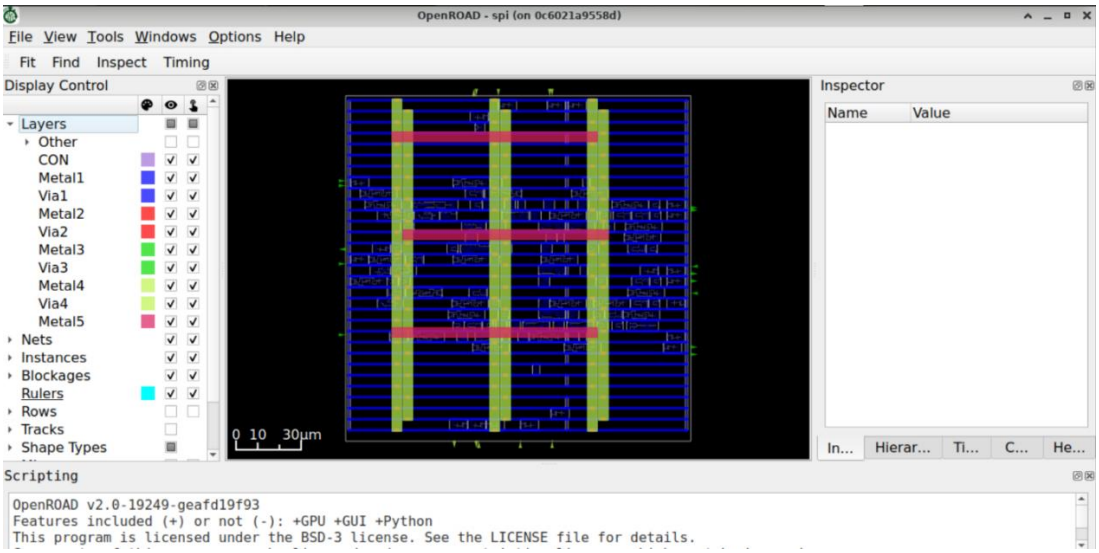
16) **Global placement** (zoomed – cells are overlapping and not aligned to the power rails)



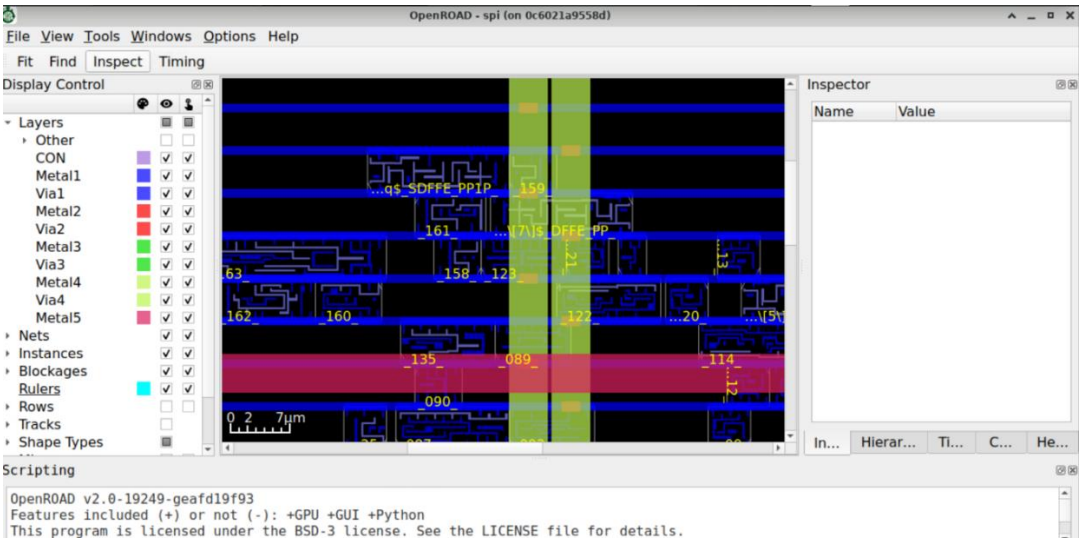
17) 3_4_place_resized – **legalization and resizing of cells**



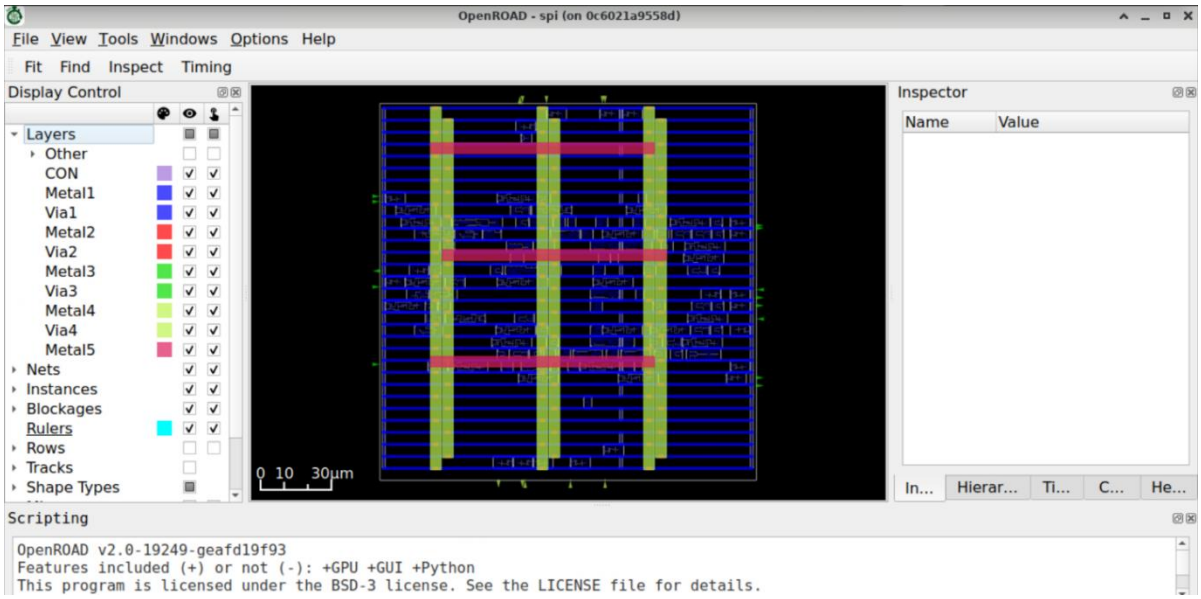
18) 3_5_place_dp – **Detailed placement**



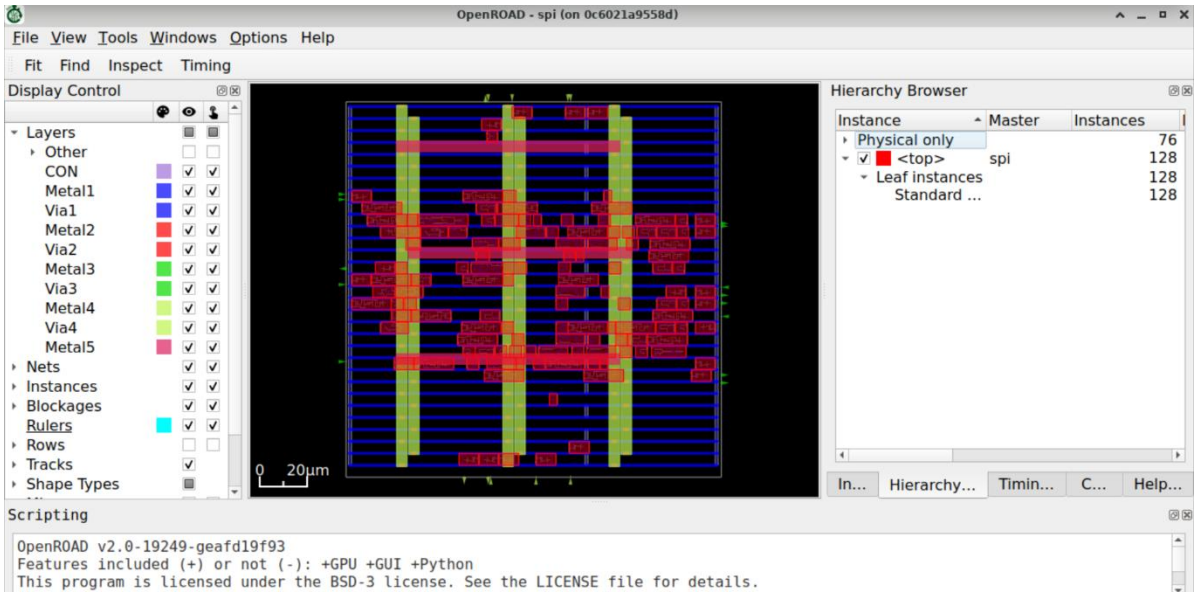
19) **Detailed placement** (zoomed – cells are not overlapping and are aligned to power rails)



20) 3_place – **final placement**

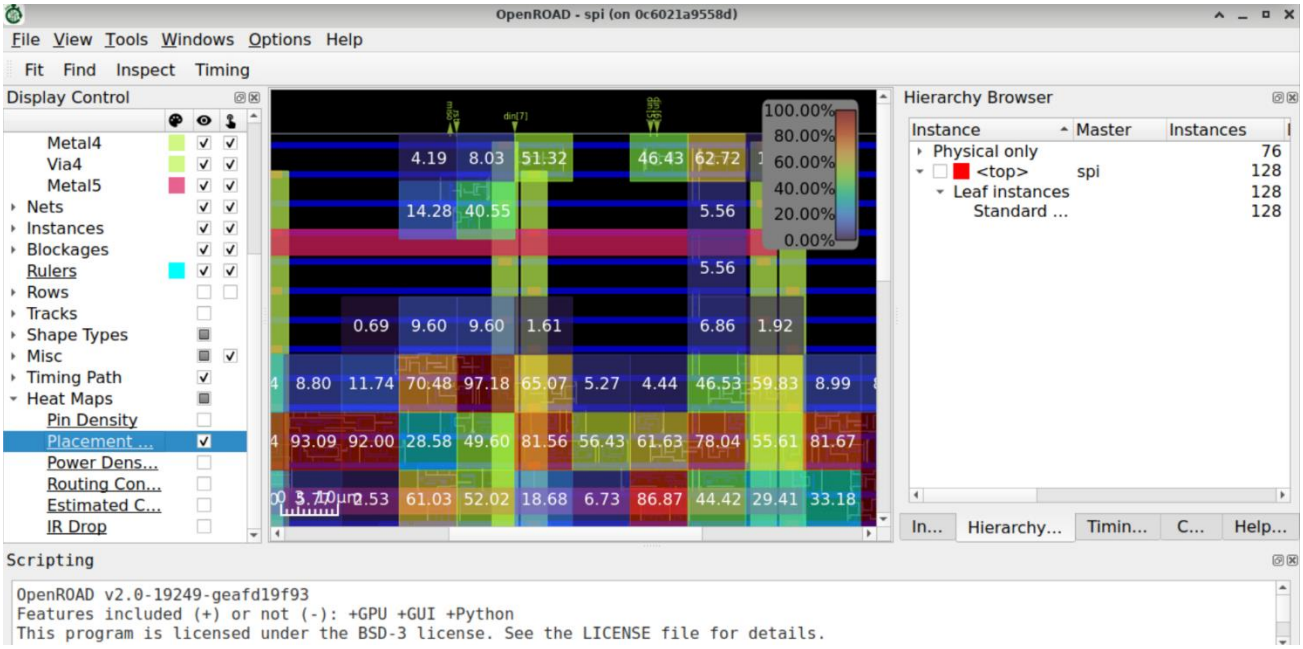


21) 3_place – **final placement** (Enable module view to have a look at the standard cell placement)

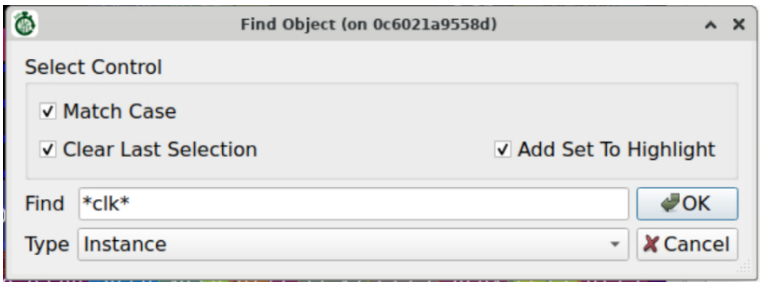


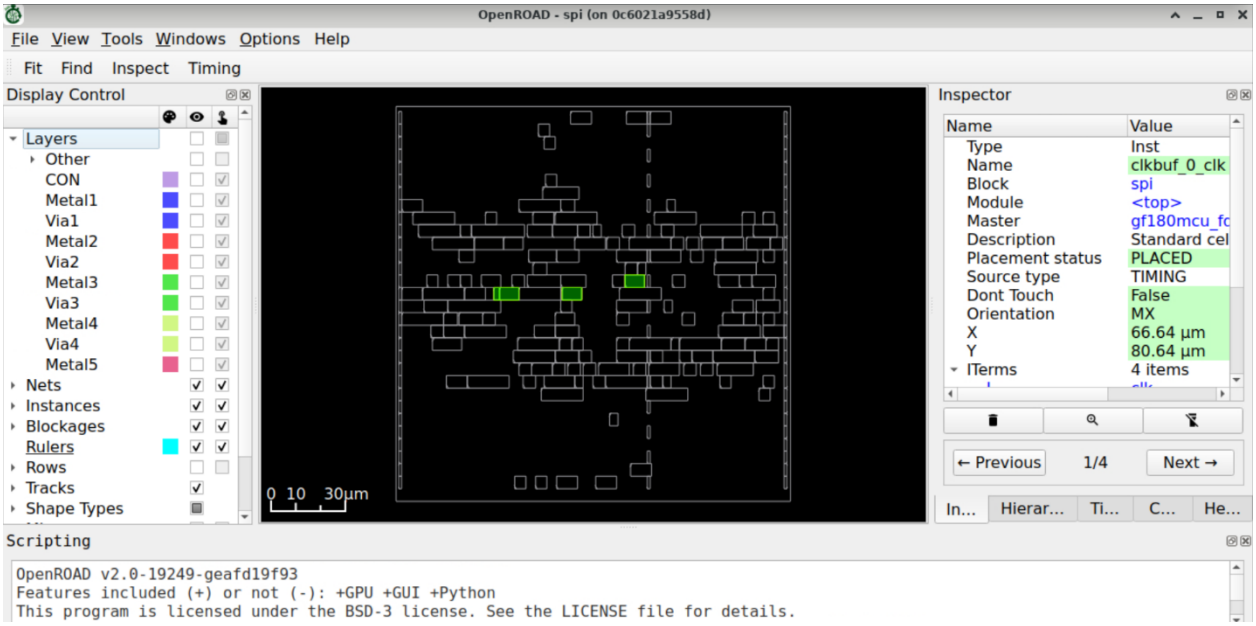
22) It is quite important to check placement density of standard cells using the heatmap option on the bottom left of the GUI. Routing congestion should be checked after 5_2_route step.

23) 3_place – **final placement** (Tools > Heat Maps > Placement Density> Enable numbers and legend) and it shows density in a numerical form. The areas with high numbers contribute to more density.

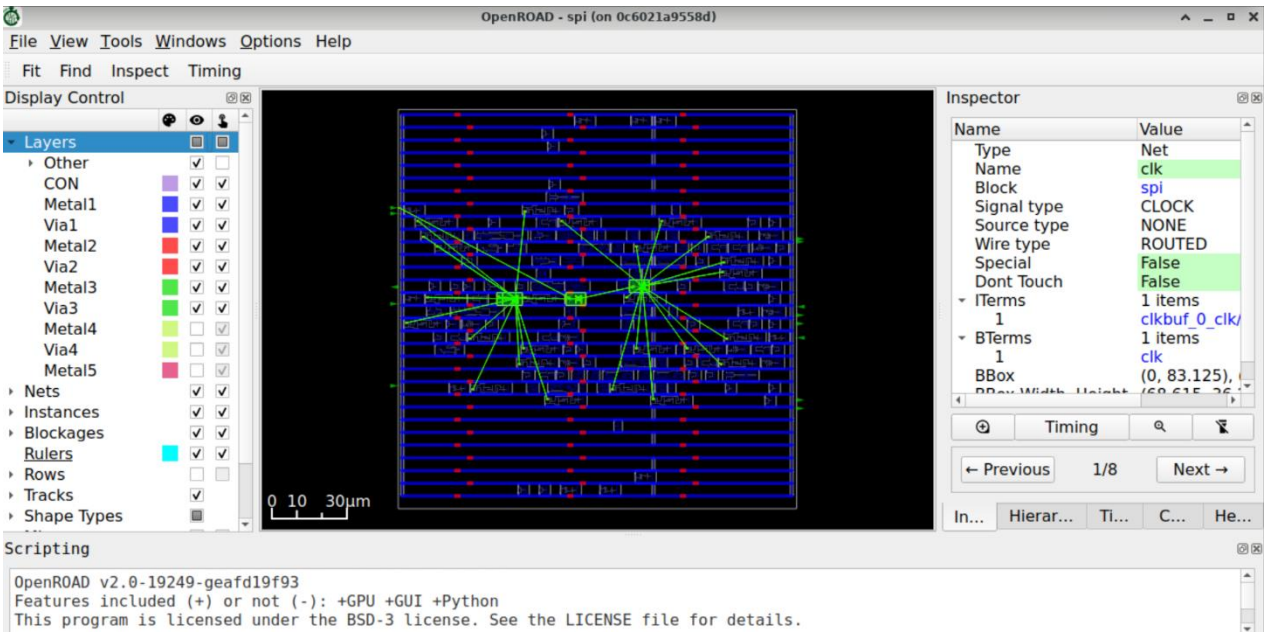


24) 4_1_cts – **cts** (clock buffers are highlighted). Information on clock buffers will be in the logs.



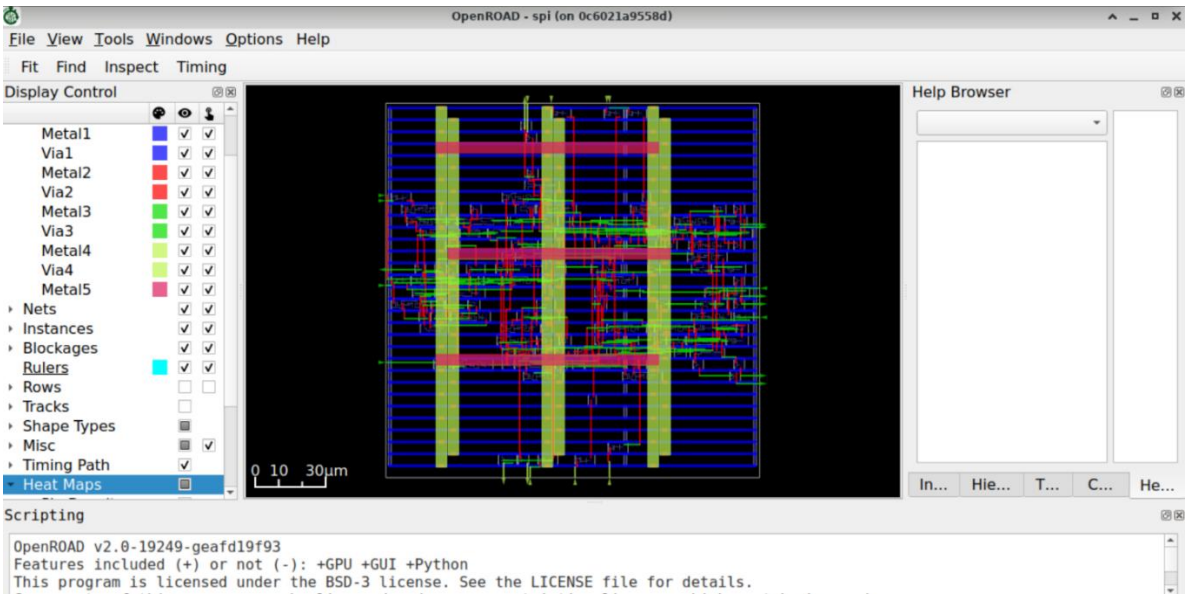


25) 4_1_cts – **cts** (The cts nets are highlighted)



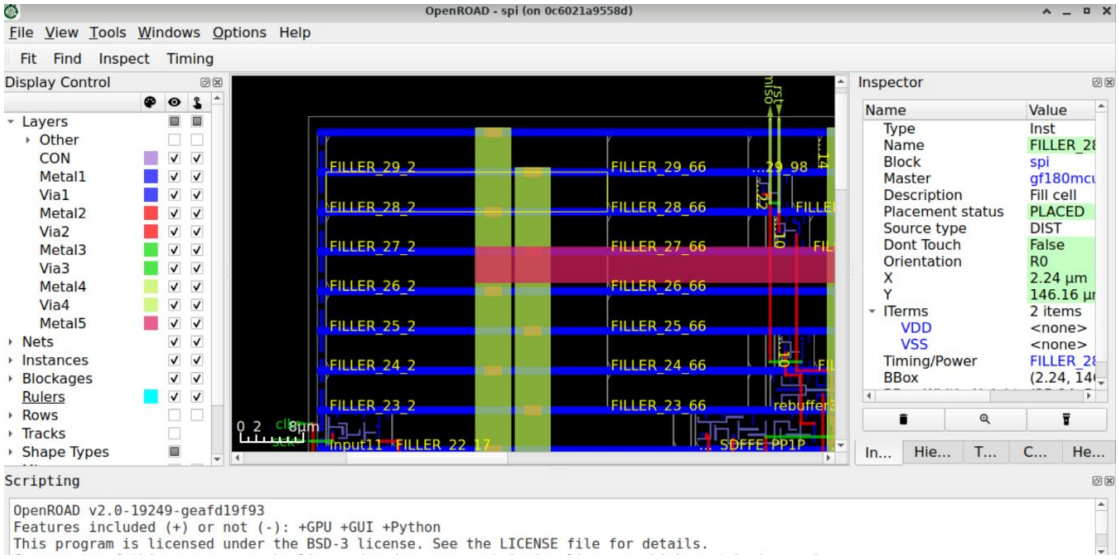
26) 5_1_grt – **Global route**

27) 5_2_route – **Detailed route**

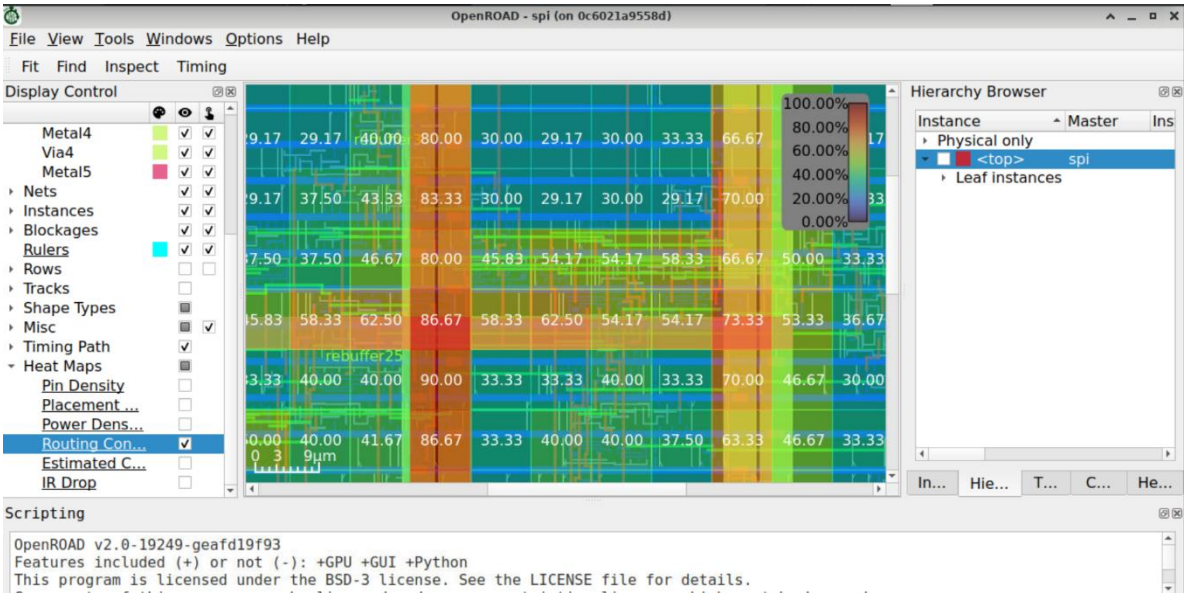


28) **Detailed route** (no filler cells)

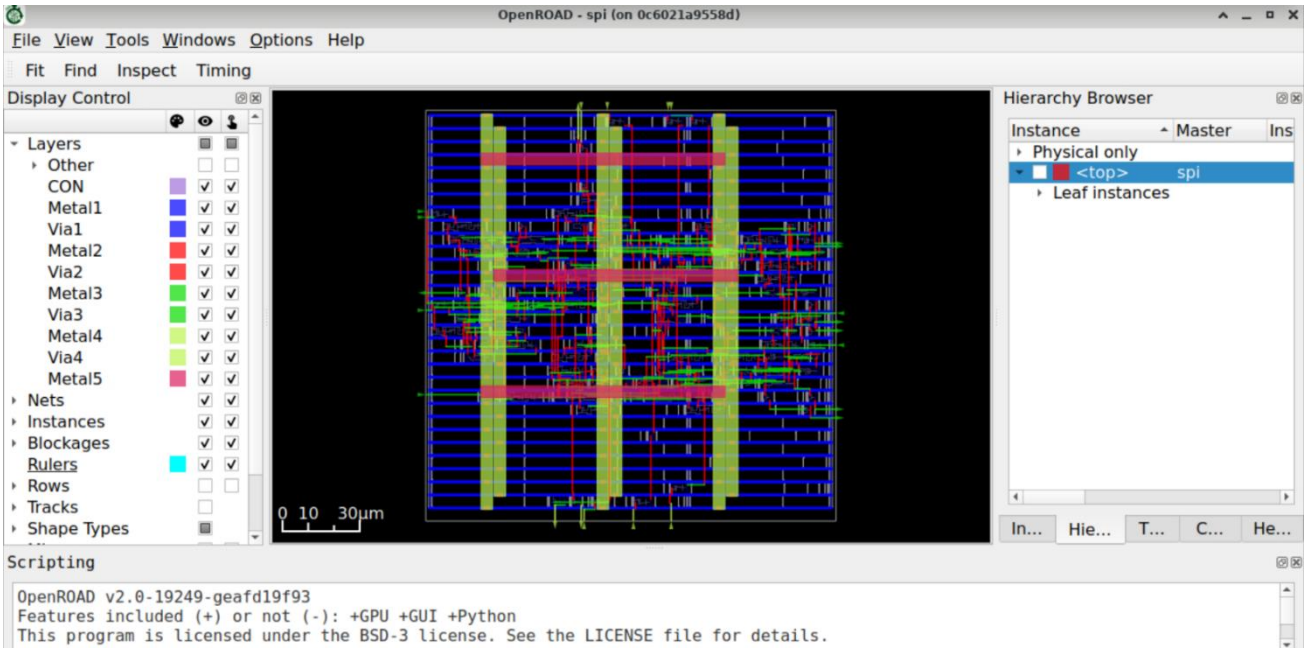
29) 5_3_fillcell – **Filler cell insertion**



30) **Routed database (Routing congestion in heat map)** - In case the density is already too high, then that would mean there is no space for routing.



31) 6_final – **final database**



32) For timing analysis, source the .lib and sdc file before loading the odb database and you should be able to view timing information and paths by using the update option in the timing browser.
    a. Timing information will be available in placement, cts and route step.