Automatic Placement & Routing with IC Compiler 2

Instructor: Lih-Yih Chiou

Speaker: Justin

Date: 2024/12/18

EE

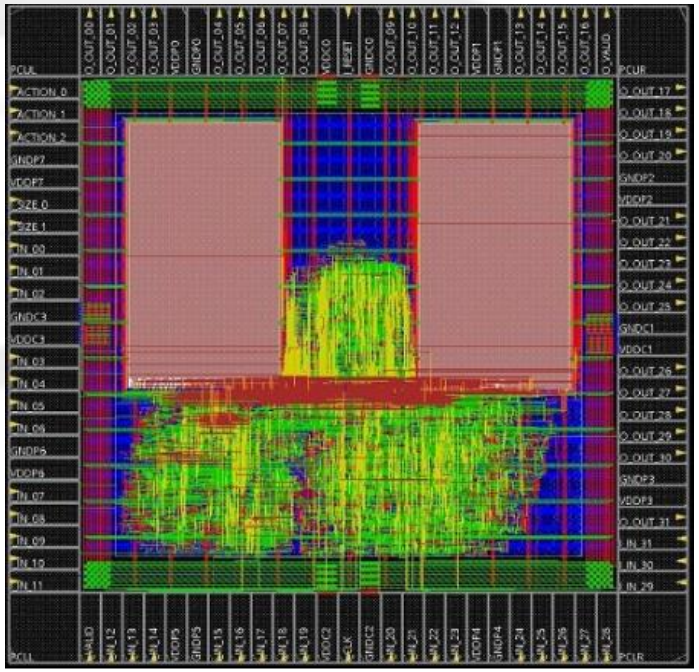
LPHPLAB VLSI Design LAB

Outline

- APR Introduction
  - Design setup
  - Design planning
  - Placement
  - CTS
  - Route
  - Chip finishing
- ICC2 tool introduction
- DRC check
- LVS check
- Area and cost

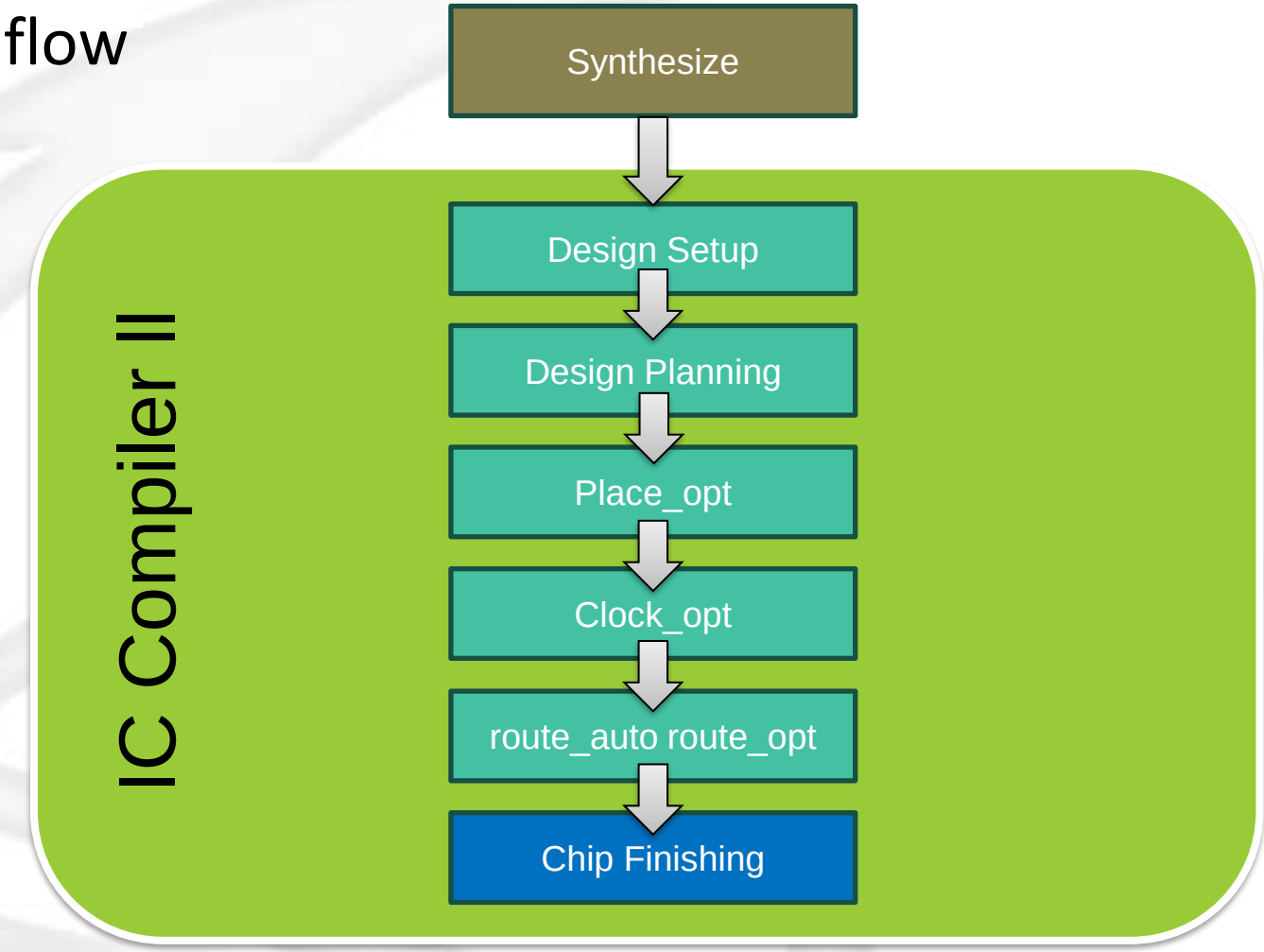
Introduction

- APR (Automatic Placement and Routing) is a critical step in the physical design phase of IC design.
- Primary Responsibility:
  - Maps the logical structure of a design onto the actual physical layout of a chip.
- Key Goals:
  - Ensure correct functionality of the design.
  - Achieve optimal performance.
  - Minimize power consumption.
  - Ensure manufacturability of the chip.

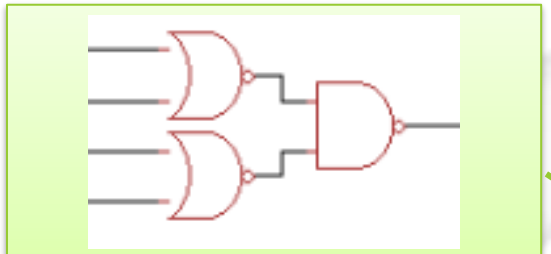


Introduction

- IC Compiler II flow



Design Setup



Design Setup

- **Multi-Corner Multi-Mode (MCMM)** is a key concept in the field of IC design, particularly in the static timing analysis (STA) and verification of digital circuits.

- **Corner** refers to a set of environmental or operating conditions that could affect the performance of the IC.
  - **Voltage (V)**: The supply voltage of the chip.
  - **Temperature (T)**: The temperature at which the chip operates.

- **Mode** refers to different operating modes or configurations of the chip.
  - **Function Mode**: the normal operating state of the chip.
  - **Test Mode**: a special configuration used for **testing** and **debugging**.(no test mode in HW4)
- Concurrent optimization under multiple mode and corner combinations, called **scenarios**.

Design Setup

- **MCMM : CHIP_func.sdc**
  - CHIP_func.sdc should be same as DC.sdc
  - period can set larger than DC.sdc
  - Remove syn only constraint
    - Don't touch network
    - Fix hold
    - Ideal network

```
set cpu_clk_period 5.0

set axi_clk_period 2.5
set rom_clk_period 50.1
set dram_clk_period 5.0

create_clock -name cpu_clk  -period $cpu_clk_period  [get_ports cpu_clk]
create_clock -name axi_clk  -period $axi_clk_period  [get_ports axi_clk]
create_clock -name rom_clk  -period $rom_clk_period  [get_ports rom_clk]
create_clock -name dram_clk -period $dram_clk_period  [get_ports dram_clk]
set_clock_groups -asynchronous -group {cpu_clk} -group {axi_clk} -group {rom_clk} -group {dram_clk}

set_clock_uncertainty  0.1  [all_clocks]
set_clock_latency      0.5  [all_clocks]
set_input_transition   0.2  [all_inputs]
set_clock_transition   0.1  [all_clocks]
```

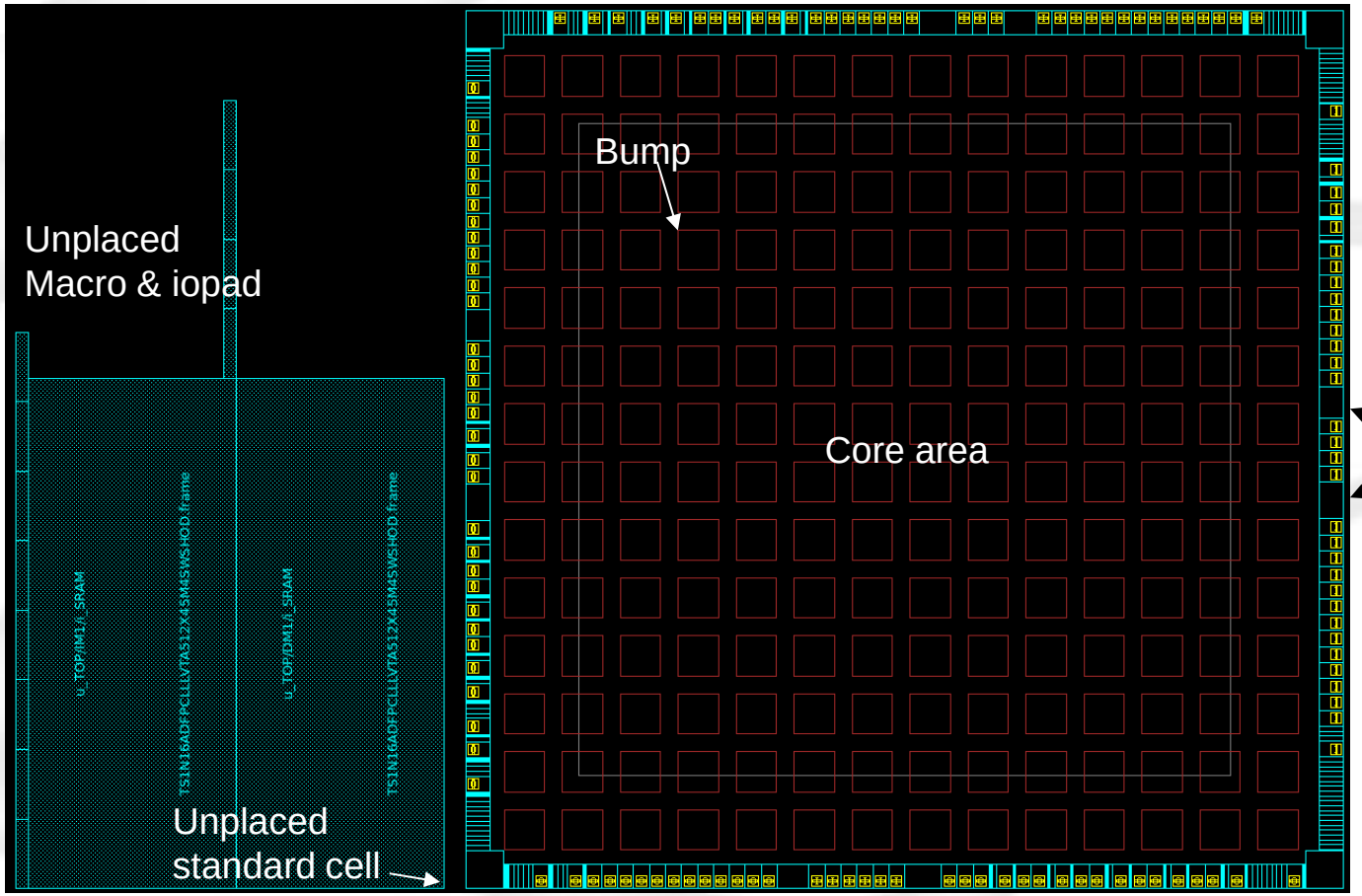
CHIP_func.sdc

```
set cpu_clk_period 4.0

set axi_clk_period 2.5
set rom_clk_period 50.1
set dram_clk_period 5.0

create_clock -name cpu_clk  -period $cpu_clk_period  [get_ports cpu_clk]
create_clock -name axi_clk  -period $axi_clk_period  [get_ports axi_clk]
create_clock -name rom_clk  -period $rom_clk_period  [get_ports rom_clk]
create_clock -name dram_clk -period $dram_clk_period  [get_ports dram_clk]
set_clock_groups -asynchronous -group {cpu_clk} -group {axi_clk} -group {rom_clk} -group {dram_clk}

set_dont_touch_network      [all_clocks]
set_fix_hold                [all_clocks]
set_clock_uncertainty  0.1  [all_clocks]
set_clock_latency      0.5  [all_clocks]
set_input_transition   0.2  [all_inputs]
set_clock_transition   0.1  [all_clocks]
set_ideal_network           [all_clocks]
set_ideal_network           [get_pins ipad_cpu_clk/C]
set_ideal_network           [get_pins ipad_axi_clk/C]
```

DC.sdc

Design planning

- Floorplan



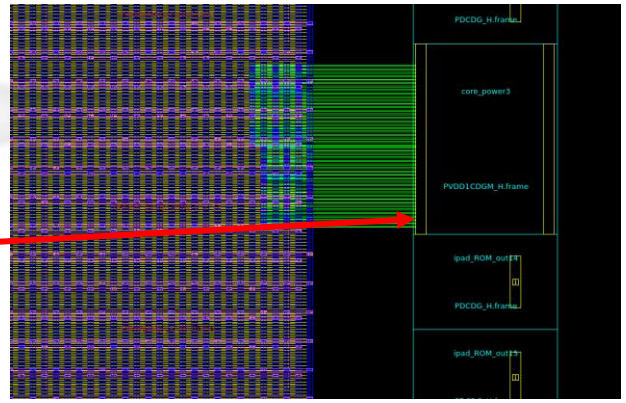
Design planning

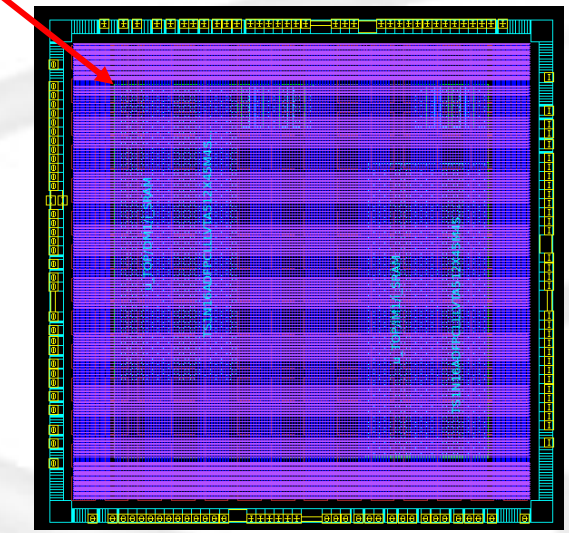
- Set IO sequence
  - Setup IO sequence in 02_design_planning.tcl by your own
  - Different sequence will affect chip performance
  - **Please pay attention to poly direction**

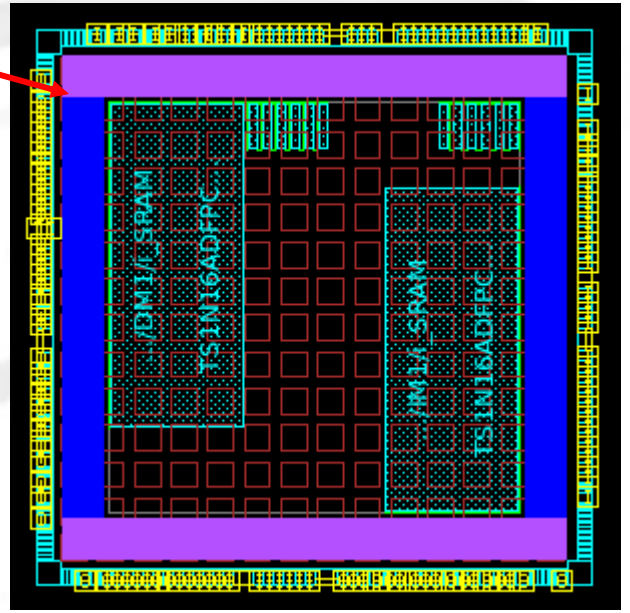
```
#set_signal_io_constraints -file ../design_data/CHIP.io
set_signal_io_constraints -io_guide_object ioring.left -constraint {{order_only}
opad_DRAM_CSn
io_power1
opad_DRAM_D1
core_power1
opad_DRAM_D14
}
set_signal_io_constraints -io_guide_object ioring.top -constraint {{order_only}
io_power2
core_power2
}
set_signal_io_constraints -io_guide_object ioring.right -constraint {{order_only}
core_power3
io_power3
}
set_signal_io_constraints -io_guide_object ioring.bottom -constraint {{order_only}
io_power4
core_power4
}

#place_io

initialize_floorplan -honor_pad_limit -core_offset {233.864} -core_utilization 1.0
create_io_ring -name ioring -corner_height 78.864

source -echo ../scripts/create_corner_pad.tcl

#set_signal_io_constraints -file ../design_data/CHIP.io
set_signal_io_constraints -io_guide_object ioring.left -constraint {{order_only}
opad_DRAM_CSn
io_power1
opad_DRAM_D1
core_power1
opad_DRAM_D14
}
set_signal_io_constraints -io_guide_object ioring.top -constraint {{order_only}
io_power2
core_power2
}
set_signal_io_constraints -io_guide_object ioring.right -constraint {{order_only}
core_power3
io_power3
}
set_signal_io_constraints -io_guide_object ioring.bottom -constraint {{order_only}
io_power4
core_power4
}
```

Design planning

- Build power network
  - Chip power pad
  - Power ring
  - Power mesh(strap)
  - Power rail

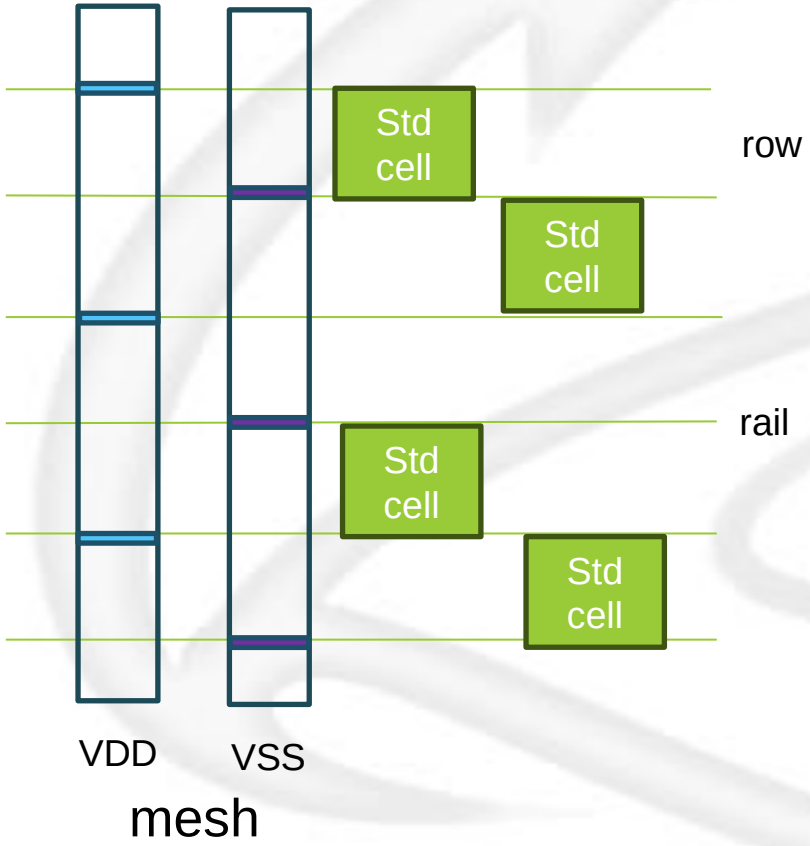


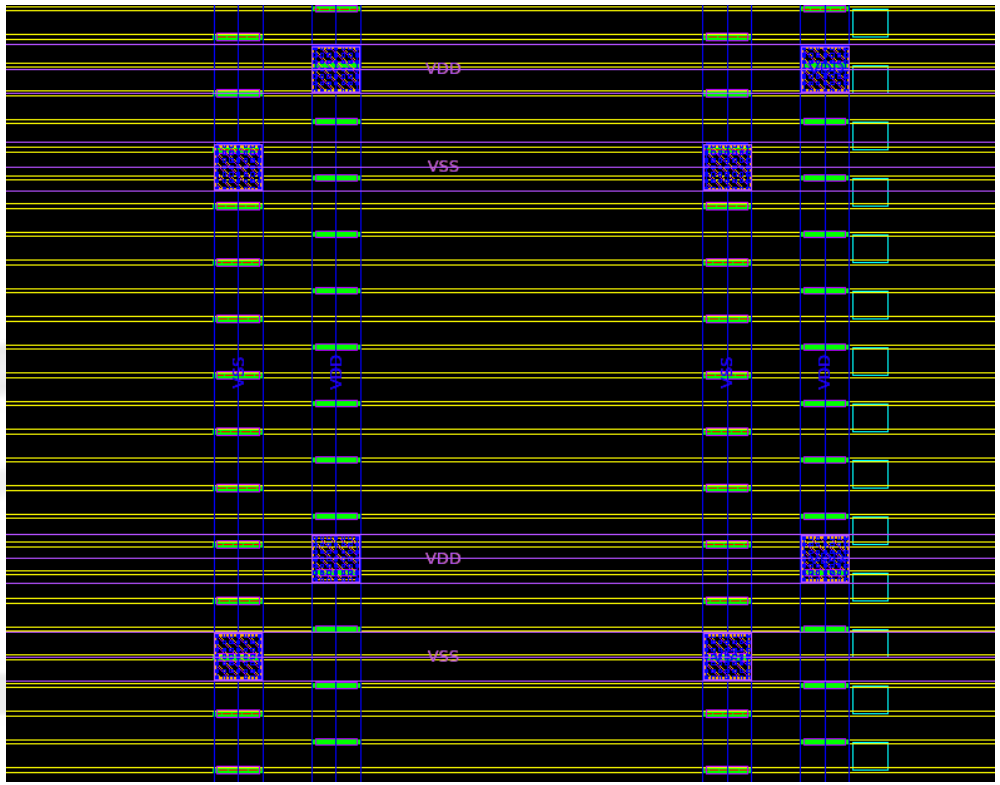




Design planning

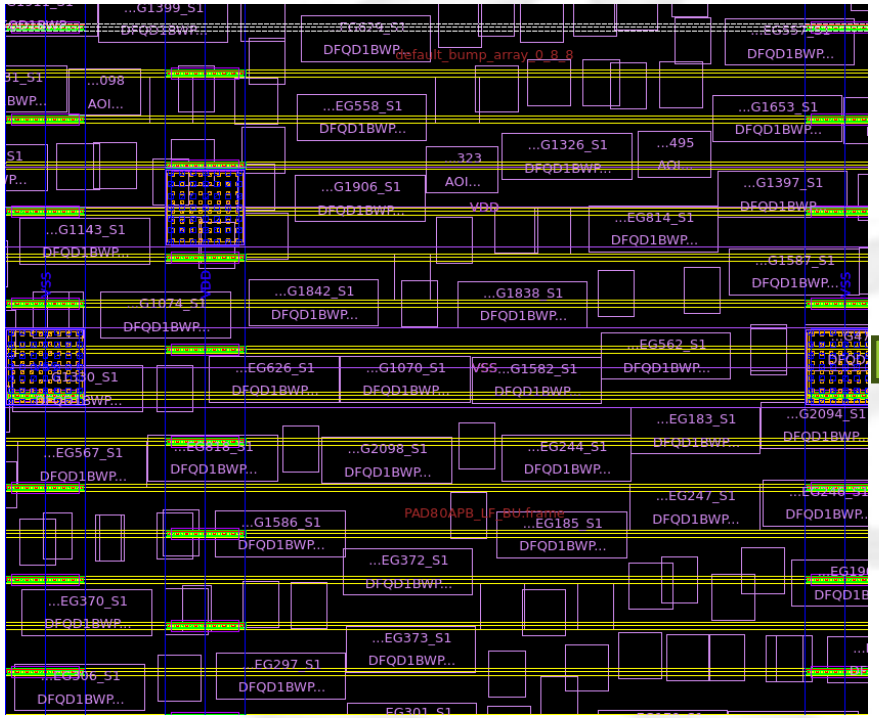
- Build power network
  - Power rail



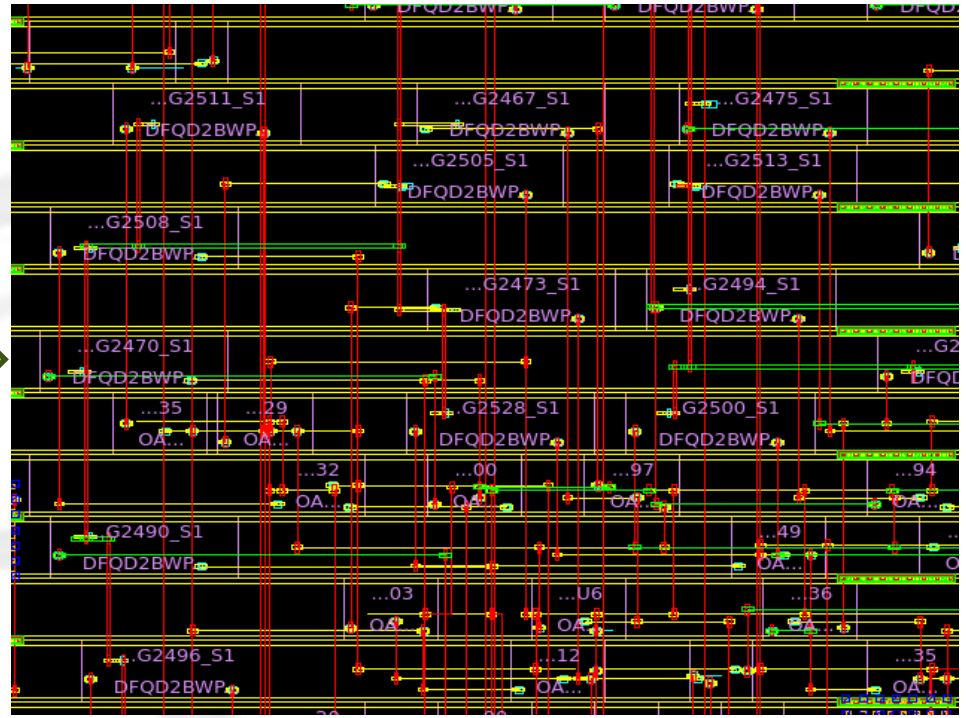


Placement

- **Placement** step involves determining the physical locations of the logical elements (gates, registers, etc.) on the chip. This step considers several factors, including: Power, Delay, Transition…



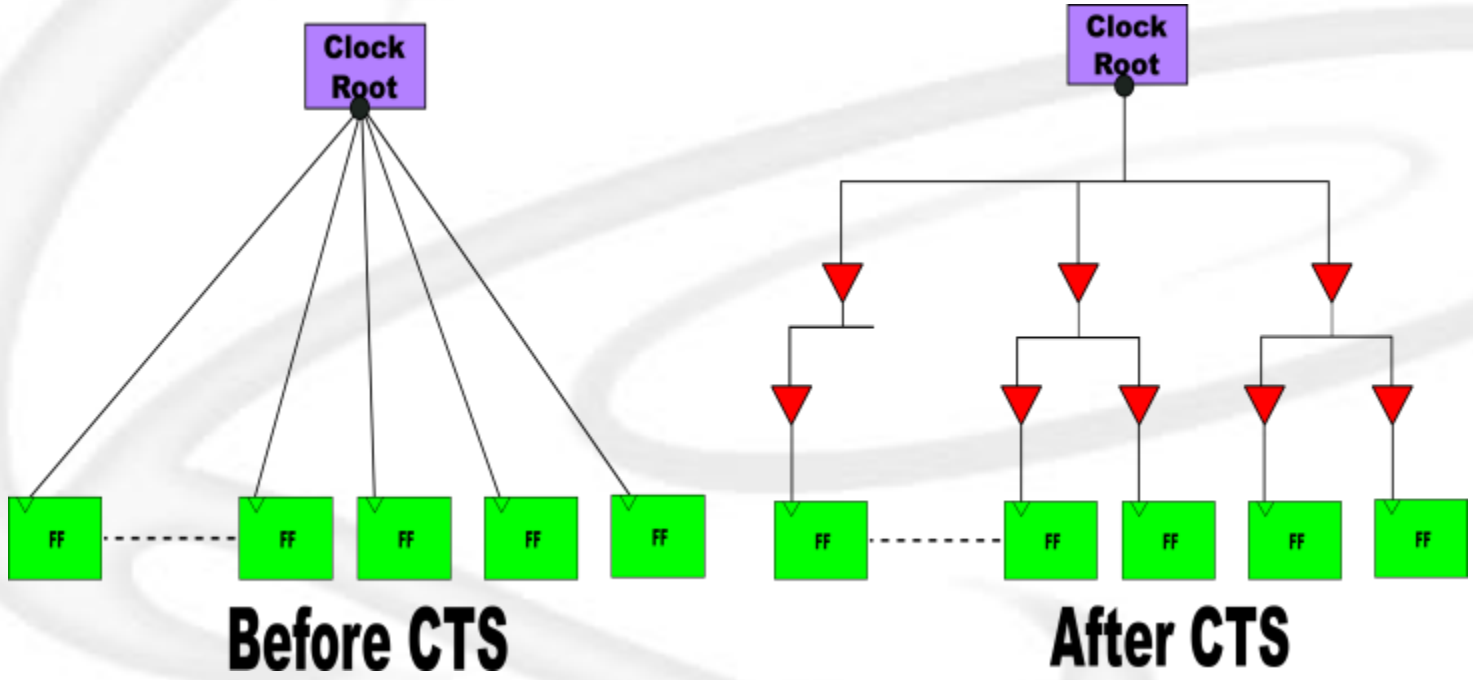
before



after

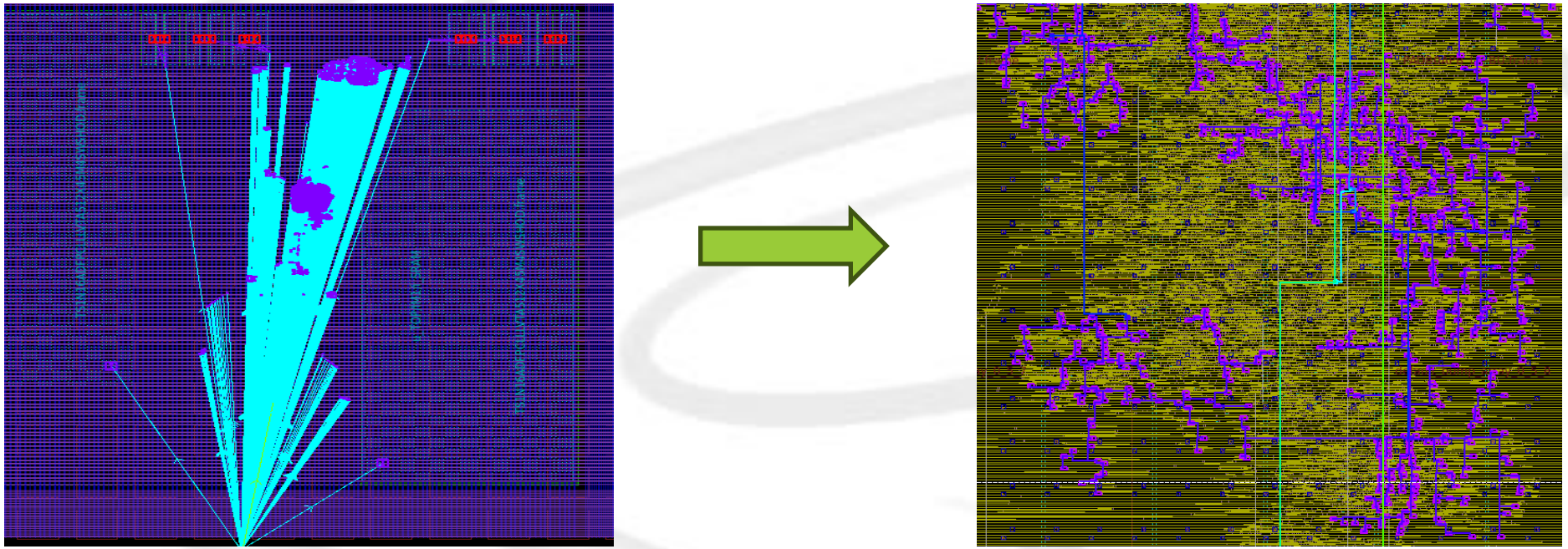
Clock Tree Synthesis

- During **CTS**, a clock network is generated. This includes the clock root (typically the clock source) and various buffers that are added to distribute the clock signal evenly across the chip.



Clock Tree Synthesis

- After the clock tree is synthesized, post-CTS verification is performed to ensure that the **timing** and **signal integrity** requirements are met.

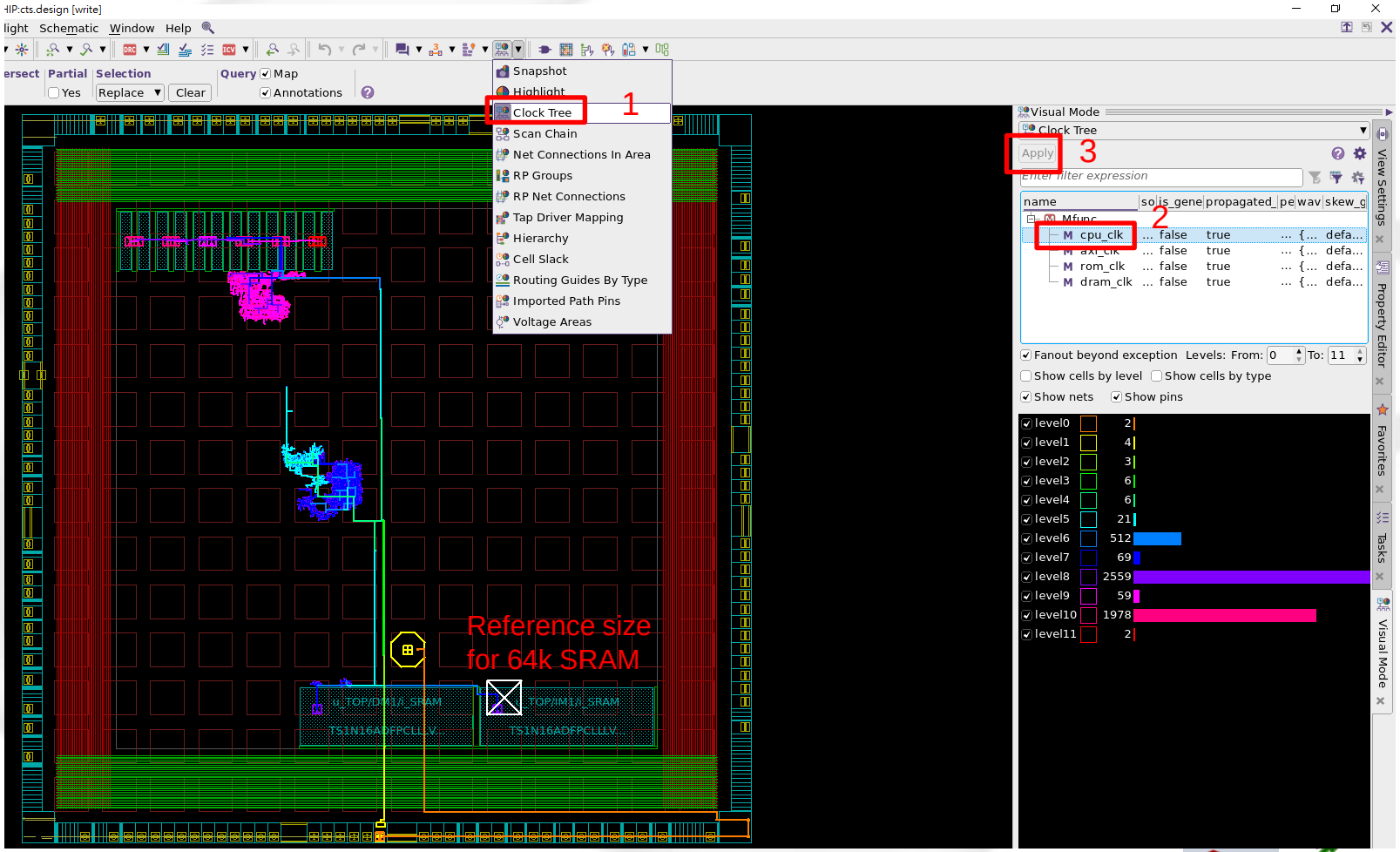


before

after

Clock Tree Synthesis

- Clock Tree viewer

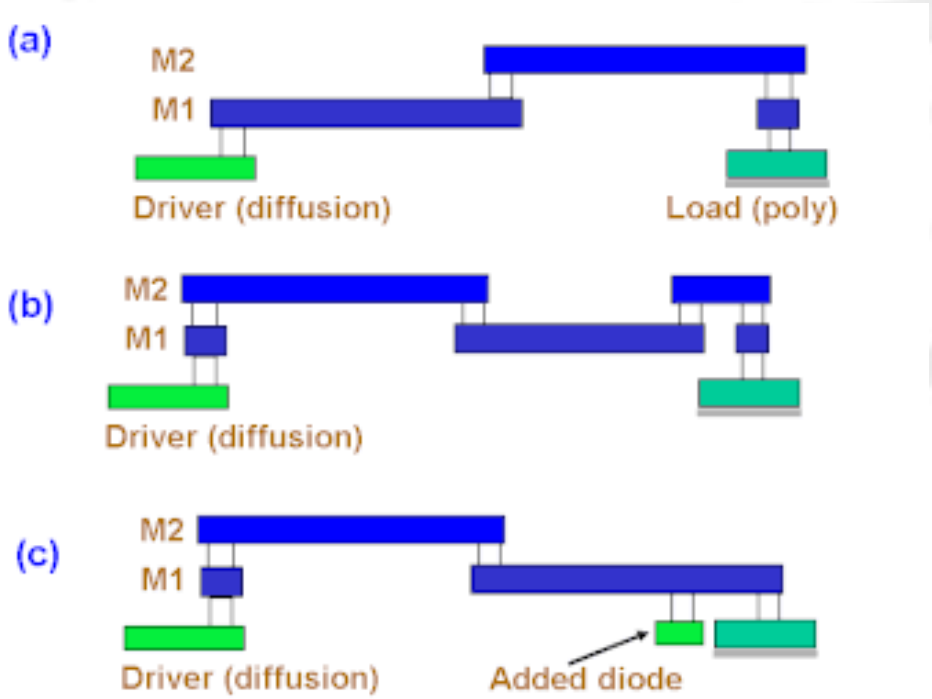


Routing

- **Routing** involves the creation of the physical connections (wires) between the cells or components after placement has been completed. Routing determines how to connect these cells in an efficient and optimal manner.
- Routing stage includes two parts:
  - route_auto :
    - initial routing entails global routing, track assignment and detail routing
  - route_opt :
    - Optionally enable power, CTO(Clock Tree Optimization) and CCD(Concurrent Clock & Data flow) optimization.

Routing

- Antenna effect
  - Antenna rule violations can be fixed by layer jumping and diode insertion
  - Layer jumping is recommended during detail route.
  - Diode insertion is recommended post-route.



Chip Finishing

- Design For Manufacturing :
  - Gate oxide integrity => Antenna rule / Insert antenna diode
  - Via resistance and reliability => insert redundant via (double via)
  - metal over-etching => insert metal filler

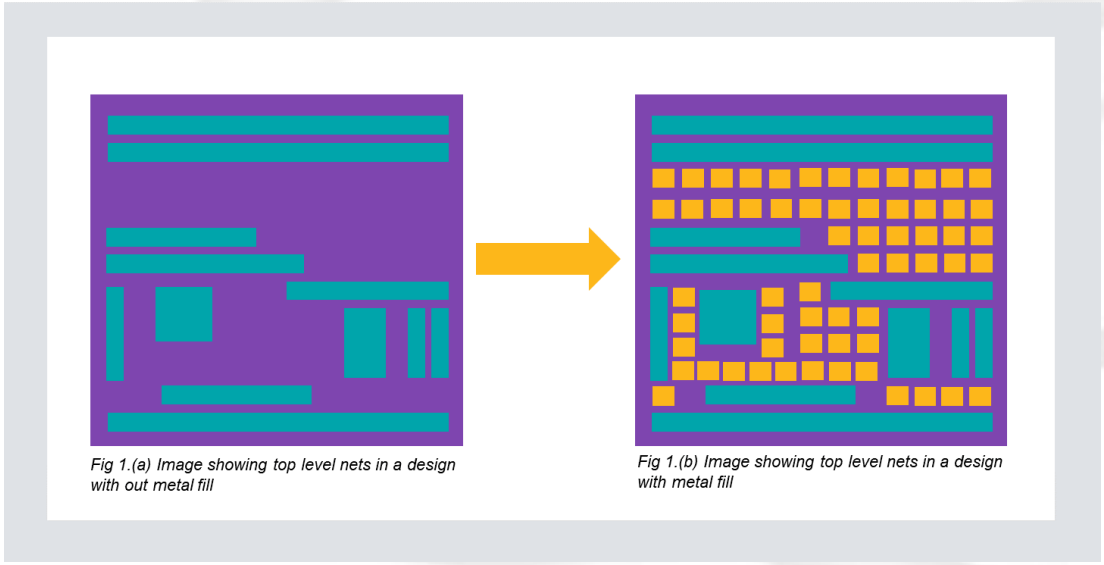
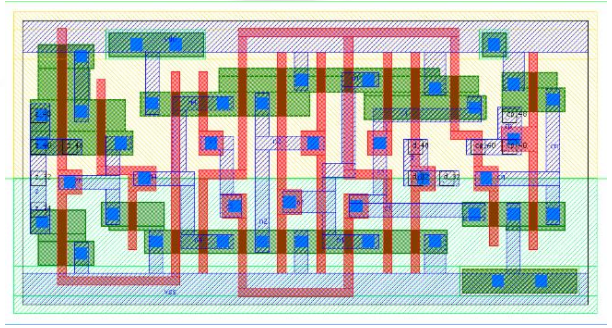


Fig 1.(a) Image showing top level nets in a design with out metal fill

Fig 1.(b) Image showing top level nets in a design with metal fill

Chip Finishing

- Stream out GDS2
  - **GDSII** (Graphic Data System II) is a file format used to represent the layout of an IC. It is the industry standard for mask data exchange between IC design tools and semiconductor foundries.
  - Merging GDS files for **Standard Cells**, **Standard I/O Cells**, and **Macros**



```
write_gds  -merge_files  { \
/usr/cad/CBDK/Executable_Package/Collaterals/IP/stdcell/N16ADFP_StdCell/GDS/N16ADFP_StdCell.gds \
/usr/cad/CBDK/Executable_Package/Collaterals/IP/stdio/N16ADFP_StdIO/GDS/N16ADFP_StdIO.gds \
/usr/cad/CBDK/Executable_Package/Collaterals/IP/bondpad/N16ADFP_BondPad/GDS/N16ADFP_BondPad.gds \
/usr/cad/CBDK/Executable_Package/AVSD_cell_lib/data_array/N16ADFP_data_array_100a.gds \
/usr/cad/CBDK/Executable_Package/AVSD_cell_lib/tag_array/N16ADFP_tag_array_100a.gds \
/usr/cad/CBDK/Executable_Package/AVSD_cell_lib/SRAM/N16ADFP_SRAM_100a.gds \
} \
-layer_map /usr/cad/CBDK/Executable_Package/Collaterals/Tech/APR/N16ADFP_APR_ICC2/N16ADFP_APR_ICC2_Gdsout_11M.10a.map \
-units 1000 CHIP.gds \
-long_names
```

Chip Finishing

- Output Verilog file
  - CHIP_pr.v :
    - a Verilog netlist of the physical design
  - CHIP_pr_lvs.v :
    - post-layout **Verilog netlist** used for running **Layout Versus Schematic (LVS)** checks.
  - CHIP_pr.sdf :
    - **timing information** (delays, setup, hold times, etc.) of the design

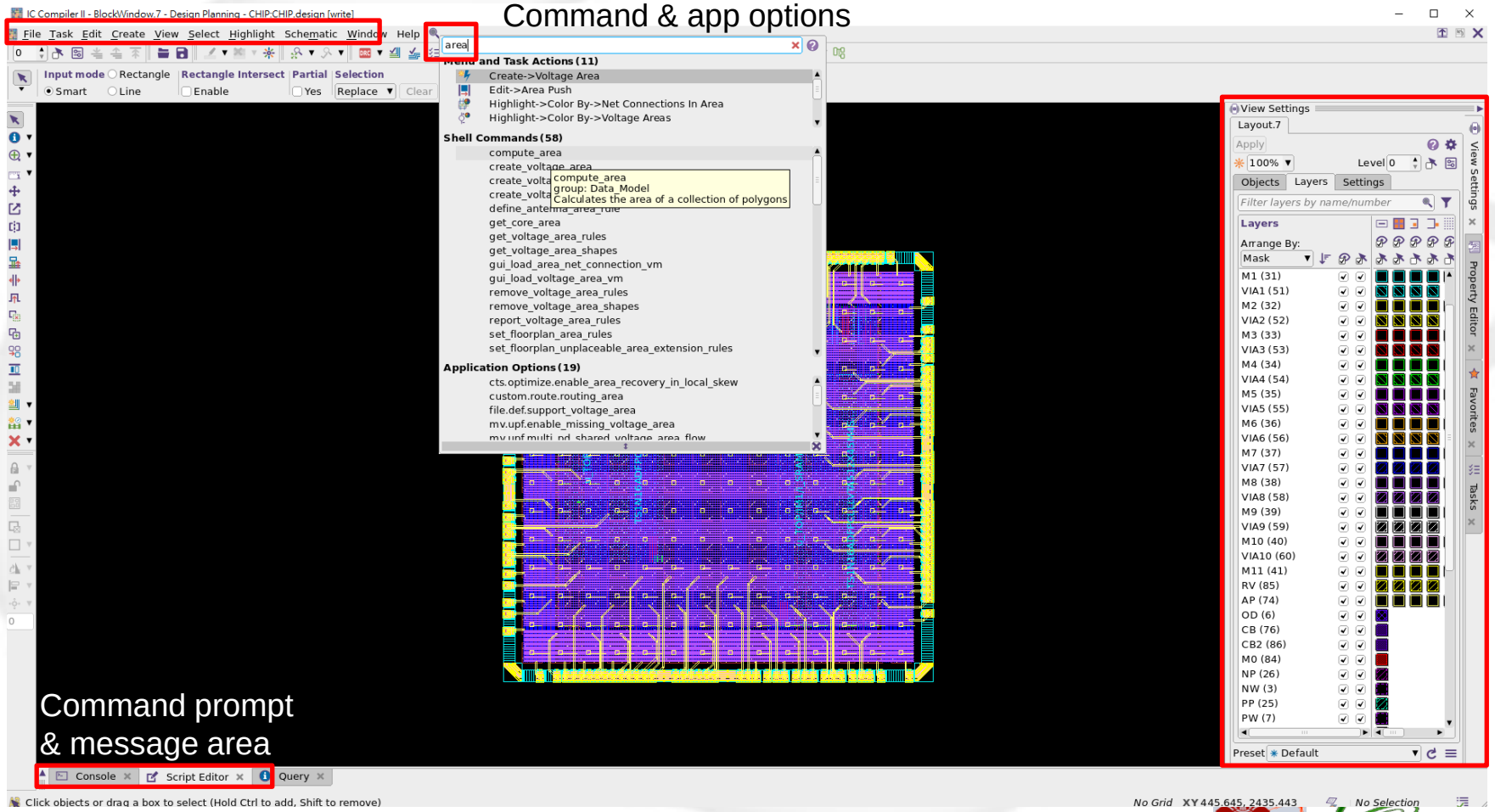
```
write_verilog -exclude {scalar_wire_declarations \
                        leaf_module_declarations \
                        pg_objects \
                        end_cap_cells \
                        well_tap_cells \
                        filler_cells \
                        pad_spacer_cells \
                        physical_only_cells \
                        empty_modules feedthrough_cells flip_chip_pad_cells \
                        cover_cells } ../../CHIP_pr.v

write_verilog -exclude {empty_modules feedthrough_cells flip_chip_pad_cells physical_only_cells} -force_reference {PVDD*} CHIP_pr_lvs.v

write_sdf ../../CHIP_pr.sdf
```

ICC2 tool introduction

- IC compiler II GUI

Menu commands

Search:
Command & app options

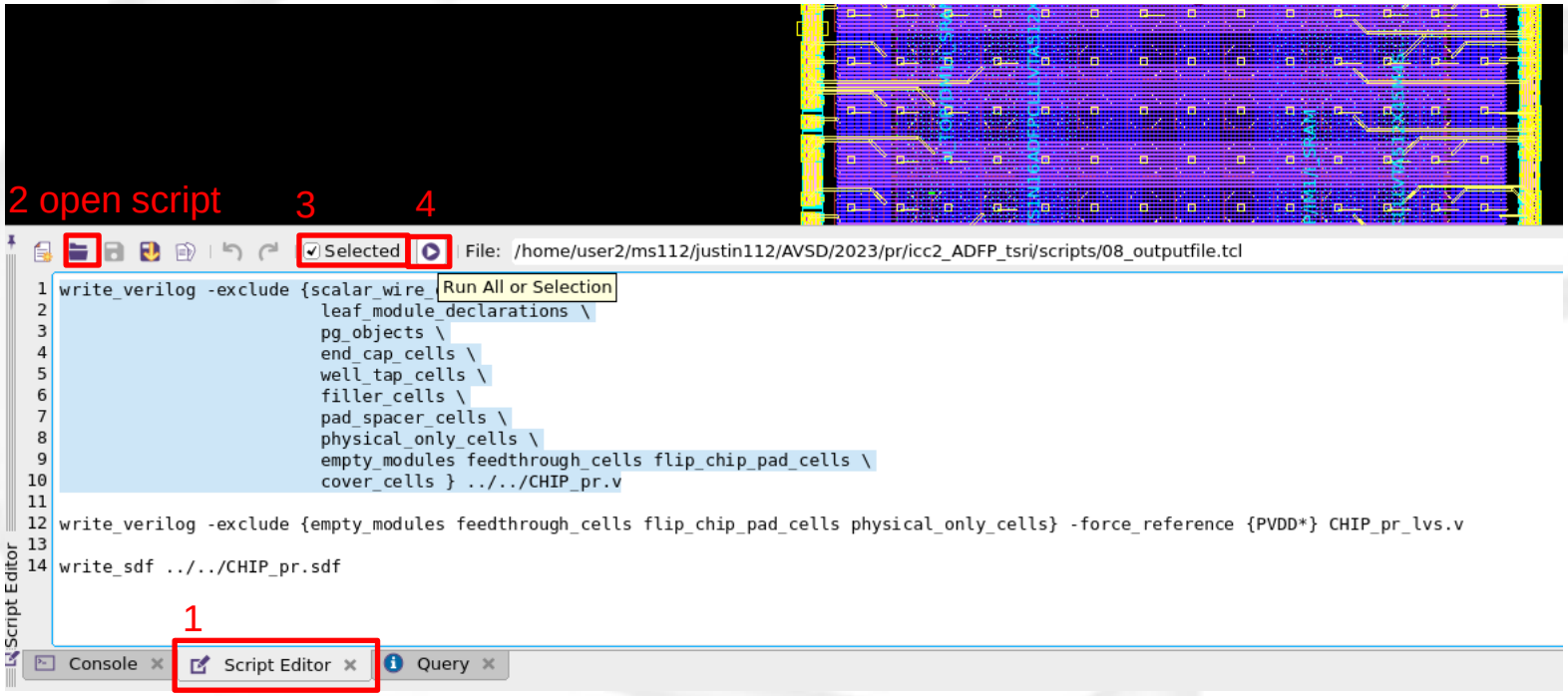


Visibility and selection control

Command prompt & message area

ICC2 tool introduction

- IC compiler II GUI
  - Script editor : run your script step by step(recommended)



ICC2 tool introduction

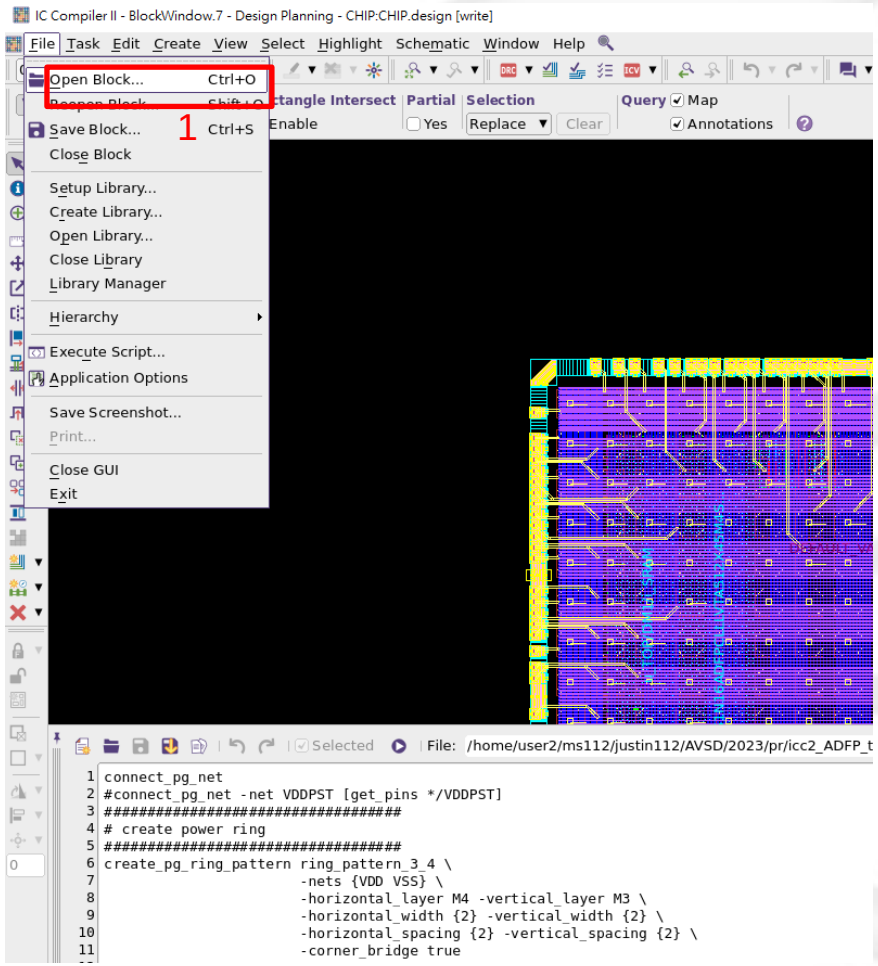
- IC compiler II script flow
  - Set app option / strategy
  - Run compile
  - Hw4 APR is incomplete, please check, otherwise APR can't run properly.

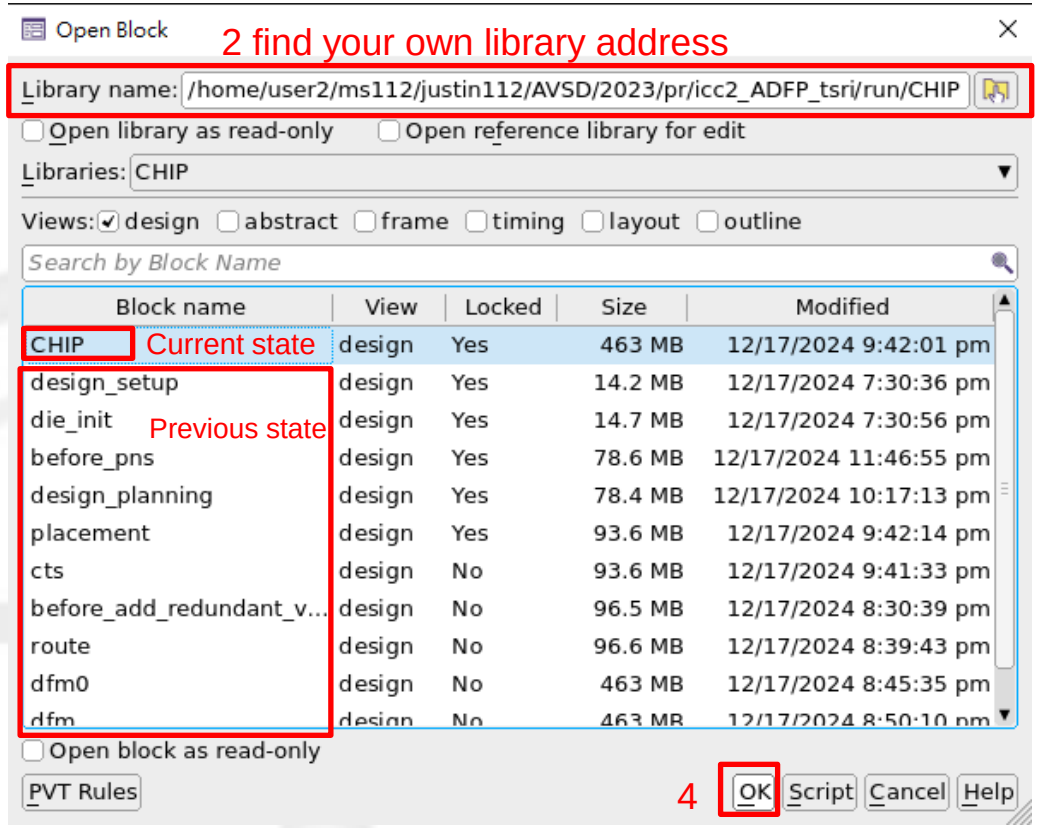
```
set_app_options -name route.common.concurrent_redundant_via_mode -value reserve_space
set_app_options -name route.common.post_detail_route_redundant_via_insertion -value off

source /usr/cad/CBDK/Executable_Package/Collaterals/Tech/APR/N16ADFP_APR_ICC2/N16ADFP_APR_ICC2_11M_Antenna.10a.tcl

route_auto
```

```
create_pg_ring_pattern ring_pattern_7_8 \
                       -nets {VDD VSS} \
                       -horizontal_layer M8 -vertical_layer M7 \
                       -horizontal_width {2} -vertical_width {2} \
                       -horizontal_spacing {2} -vertical_spacing {2} \
                       -corner_bridge true

set_pg_strategy Strategy_ring_7_8 -core -pattern {{name : ring_pattern_7_8}{nets

compile_pg -strategies {Strategy_ring_3_4 Strategy_ring_5_6 Strategy_ring_7_8}
```

ICC2 tool introduction

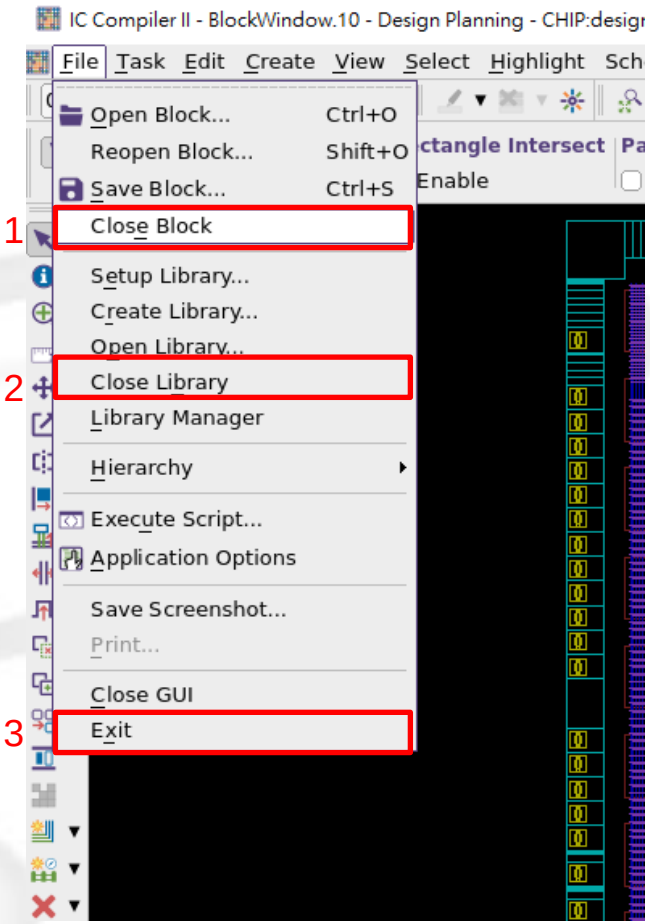
- Open block : open your current block state or saved block state





ICC2 tool introduction

- Exiting ICC2
  - Open design
    - Unix% icc2_shell –gui
    - icc2_shell> open_lib CHIP
    - icc2_shell> open_block CHIP
  - Save design
    - icc2_shell> save_block
    - icc2_shell> save_block –as CHIP:design_setup
    - icc2_shell> save_lib
  - Close design
    - icc2_shell> close_block
    - icc2_shell> close_lib
    - icc2_shell> exit

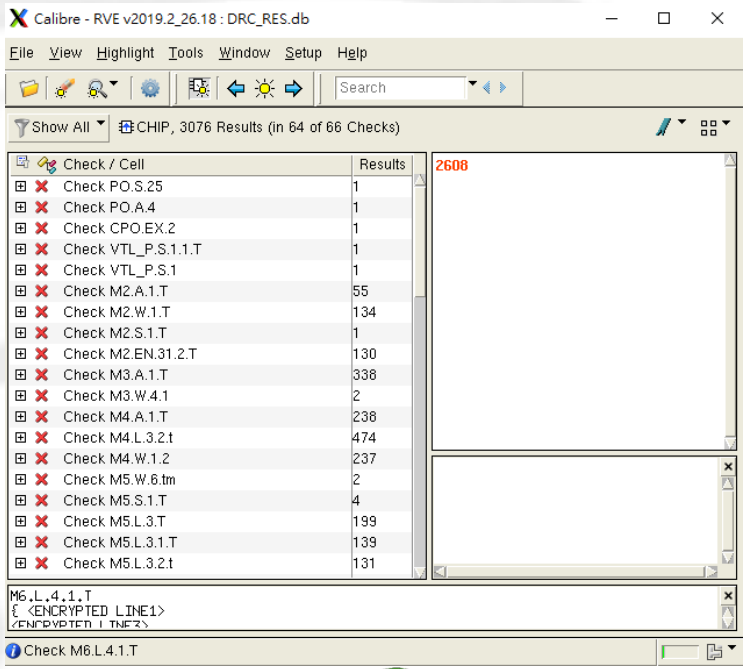


DRC check

- unix% cd ./pr/icc2_ADFP_tsri/verify/01_ipmerge_insertDummy/03_dummyMerge
- unix% ./addDummy.csh
- unix% cd ./pr/icc2_ADFP_tsri/verify/03_pv/02_drc
- unix% ./run.csh

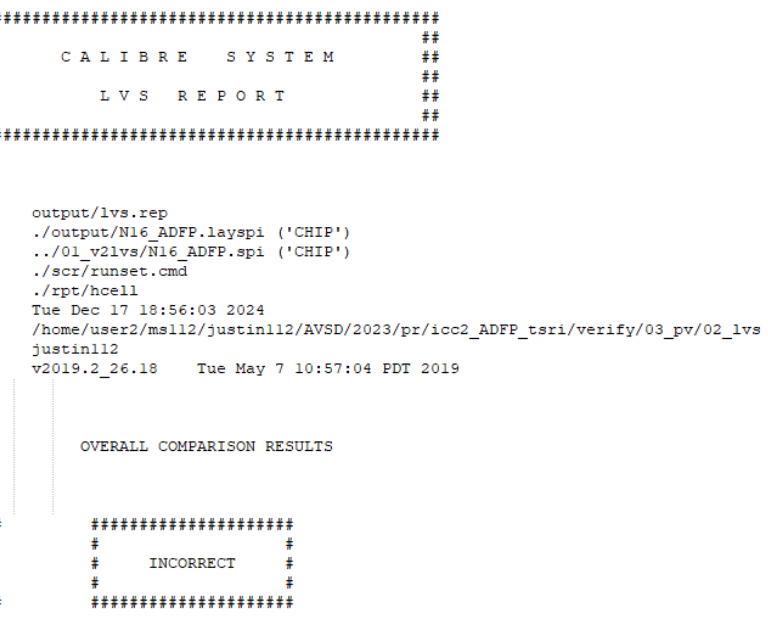
```
--- CALIBRE::DRC-H EXECUTIVE MODULE COMPLETED.  CPU TIME = 15961  REAL TIME = 573
--- TOTAL RULECHECKS EXECUTED = 2299
--- TOTAL RESULTS GENERATED = 3076 (3076)
--- DRC RESULTS DATABASE FILE = output/DRC_RES.db (ASCII)

--- CALIBRE::DRC-H COMPLETED - Tue Dec 17 18:15:23 2024
--- TOTAL CPU TIME = 15994  REAL TIME = 598
--- PROCESSOR COUNT = 32
--- SUMMARY REPORT FILE = output/DRC.rep
```

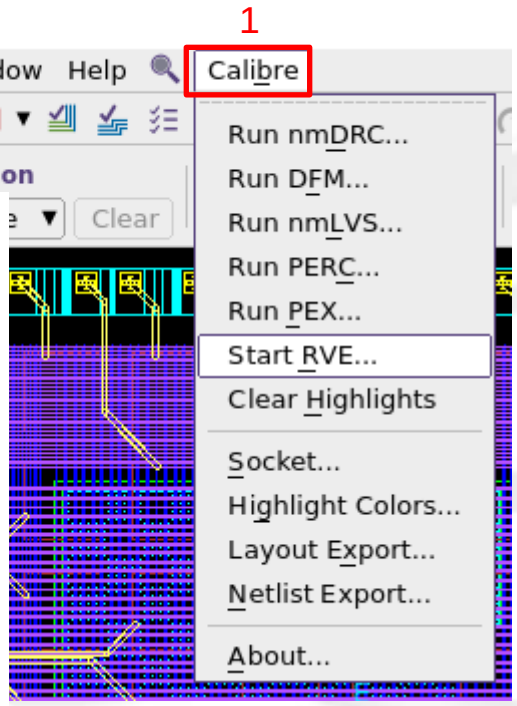
- unix% calibre –rve output/DRC_res.db
  - Check where the DRC are violated

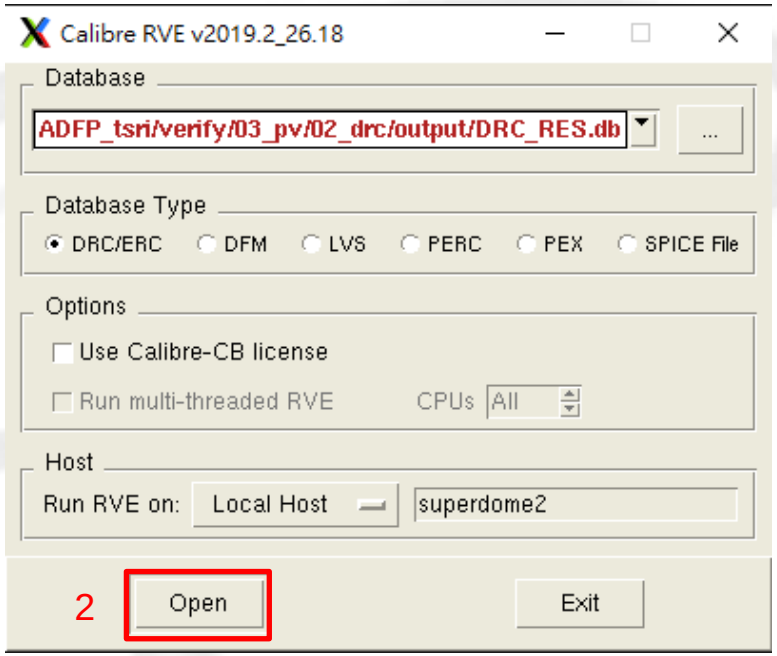


DRC check

- Read Calibre DRC result in ICC2
- icc2_shell> source /usr/cad/mentor/calibre/cur/lib/icc_calibre.tcl

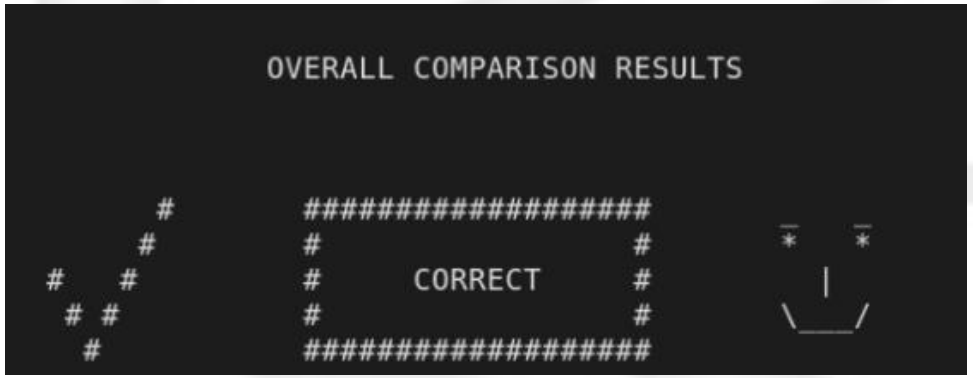






LVS check

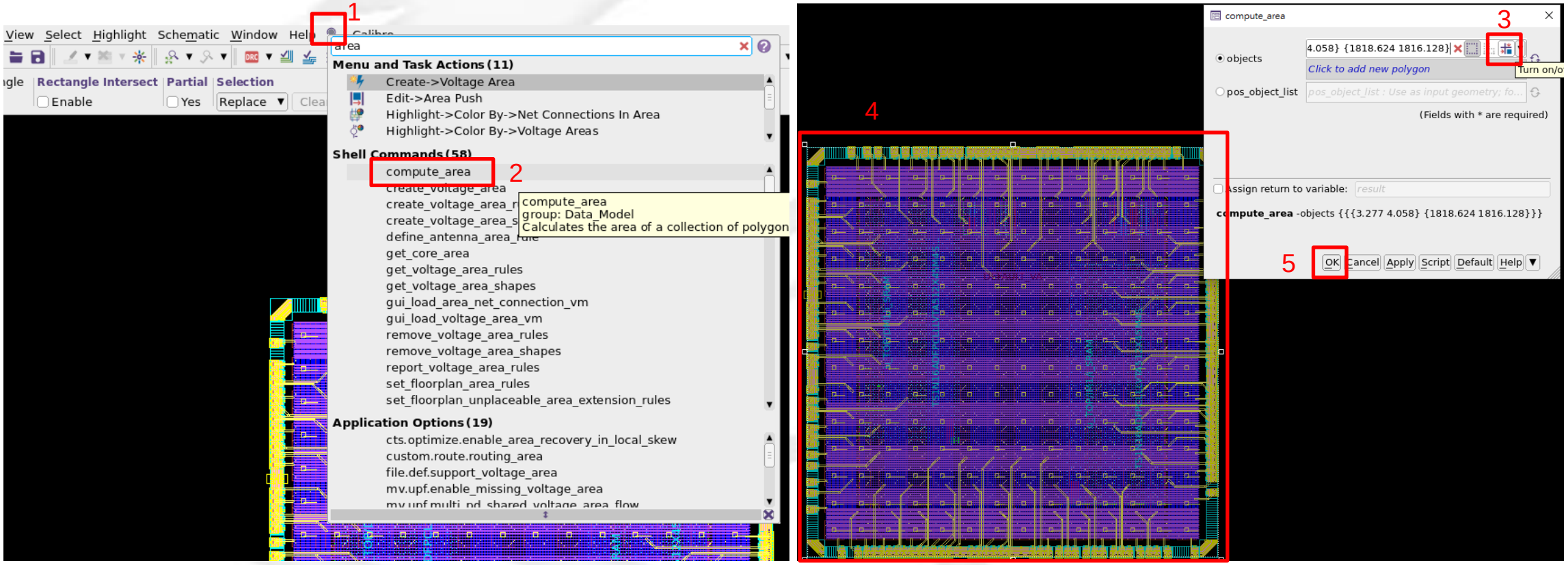
- unix% cd ./pr/icc2_ADFP_tsri/verify/03_pv/01_v2lvs
- unix% ./run.csh
- Unix% cd ./pr/icc2_ADFP_tsri/verify/03_pv/02_lvs
- unix% ./run.csh
- open ./output/lvs.rep with text editor




```
##################################################
##                                              ##
##        C A L I B R E   S Y S T E M           ##
##                                              ##
##            L V S   R E P O R T               ##
##                                              ##
##################################################


REPORT FILE NAME:         output/lvs.rep
LAYOUT NAME:              ./output/N16_ADFP.layspi ('CHIP')
SOURCE NAME:              ../01_v2lvs/N16_ADFP.spi ('CHIP')
RULE FILE:                ./scr/runset.cmd
HCELL FILE:               ./rpt/hcell
CREATION TIME:            Tue Dec 17 18:56:03 2024
CURRENT DIRECTORY:        /home/user2/ms112/justin112/AVSD/2023/pr/icc2_ADFP_tsri/verify/03_pv/02_lvs
USER NAME:                justin112
CALIBRE VERSION:          v2019.2_26.18    Tue May 7 10:57:04 PDT 2019


                    OVERALL COMPARISON RESULTS


                  ######################
                  #                    #
                  #     INCORRECT      #
                  #                    #
                  ######################
```

Area and cost

- Compute area



Area and cost

- Compute area
- Result will show at the terminal

```
icc2_shell> set_working_design_stack CHIP:placement.design
icc2_shell> set_working_design_stack CHIP:CHIP.design
icc2_shell> compute_area -objects {{{3.277 4.058} {1818.624 1816.128}}}
3289535.83829000 (um^2)
icc2_shell>
```

- N16 area cost
  - Please evaluate your chip area cost in your report

製程代號	面積 < 2.16 mm^2 單價 (元/mm^2)	2.16 mm^2 ≦面積 < 4mm^2 單一費率 (元)	面積 ≧ 4mm^2 單價 (元/mm^2)
TN16FFC	1,885,000	4,068,000	1,017,000

LPHPLAB
VLSI Design LAB

Thanks for listening