Introduction:

During this summer, I updated the PDK and workflow for the course ELEC527. The original PDK was AMI05 (500nm), and I replaced it with the Cadence SkyWater 130nm PDK. Dr. Cavallaro was looking for a tape-out opportunity since the foundry no longer supports the AMI05 technology.

The original ASIC flow was:

Design Compiler (synthesis) → Questa (post-DC simulation) → Innovus (place and route) → Magic (extraction) → IRSIM (digital simulation)

The new ASIC flow is:

Design Compiler/Genus (synthesis) → Questa (post-DC simulation) → Innovus (place and route) → Pegasus (DRC & LVS) → Quantus (extraction) → Spectre (analog simulation) → Xcelium Logic Simulator (digital simulation)

Method & Result:

Synthesis

For the synthesis part, I updated the Design Compiler script to better support two-phase clocks and also wrote a Genus script that achieves the same functionality. Since I'm using the Cadence SkyWater PDK, Genus provides better compatibility, while Design Compiler serves as a backup. All the code were in **Appendix** and the results were shown as below:

```
set_driving_cell -lib_cell INVX1 [get_ports in_clkb]
set_driving_cell -lib_cell INVX1 [get_ports in_restart]
set_driving_cell -lib_cell INVX1 [get_ports in_load]
set_driving_cell -lib_cell INVX1 [get_ports {in_d1_in[3]}]
set_driving_cell -lib_cell INVX1 [get_ports {in_d1_in[2]}]
set_driving_cell -lib_cell INVX1 [get_ports {in_d1_in[1]}]
set_driving_cell -lib_cell INVX1 [get_ports {in_d1_in[0]}]
set_driving_cell -lib_cell INVX1 [get_ports {in_d2_in[3]}]
set_driving_cell -lib_cell INVX1 [get_ports {in_d2_in[2]}]
set_driving_cell -lib_cell INVX1 [get_ports {in_d2_in[1]}]
set_driving_cell -lib_cell INVX1 [get_ports {in_d2_in[0]}]
create_clock [get_ports in_clka]  -period 20  -waveform {0 10}
create_clock [get_ports in_clkb]  -period 20  -waveform {0 10}
set_input_delay -clock in_clka  1  [get_ports in_restart]
set_input_delay -clock in_clka  1  [get_ports in_load]
set_input_delay -clock in_clkb  1  [get_ports {in_d1_in[3]}]
set_input_delay -clock in_clkb  1  [get_ports {in_d1_in[2]}]
set_input_delay -clock in_clkb  1  [get_ports {in_d1_in[1]}]
set_input_delay -clock in_clkb  1  [get_ports {in_d1_in[0]}]
set_input_delay -clock in_clkb  1  [get_ports {in_d2_in[3]}]
set_input_delay -clock in_clkb  1  [get_ports {in_d2_in[2]}]
set_input_delay -clock in_clkb  1  [get_ports {in_d2_in[1]}]
set_input_delay -clock in_clkb  1  [get_ports {in_d2_in[0]}]
set_output_delay -clock in_clka  1  [get_ports out_start]
set_output_delay -clock in_clka  1  [get_ports out_done]
set_output_delay -clock in_clkb  1  [get_ports {out_state_main[1]}]
set_output_delay -clock in_clkb  1  [get_ports {out_state_main[0]}]
set_output_delay -clock in_clkb  1  [get_ports {out_d_out[3]}]
set_output_delay -clock in_clkb  1  [get_ports {out_d_out[2]}]
set_output_delay -clock in_clkb  1  [get_ports {out_d_out[1]}]
set_output_delay -clock in_clkb  1  [get_ports {out_d_out[0]}]
set_clock_groups  -logically_exclusive -name in_clka_1  -group [get_clocks
in_clka] -group [get_clocks in_clkb]
```

Figure 1. sdc file

Double phase clocks were created in sdc file by Design Compiler/Genus.

Innovus

For the Innovus part, I updated the flow to support Multi-Mode Multi-Corner (MMMC) analysis. I created .view files and edited I/O pins. Then I implemented optimized, timing-aware placement. I also added Clock Tree Synthesis (CTS) at the end of the script.

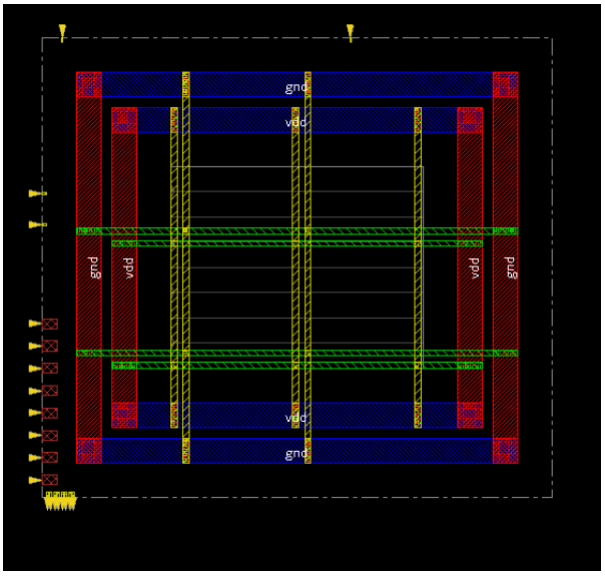


Figure 2. Innovus layout after edit pin

After clock three synthesis in Innovus by using above sdc file, there were two clock tree as below:

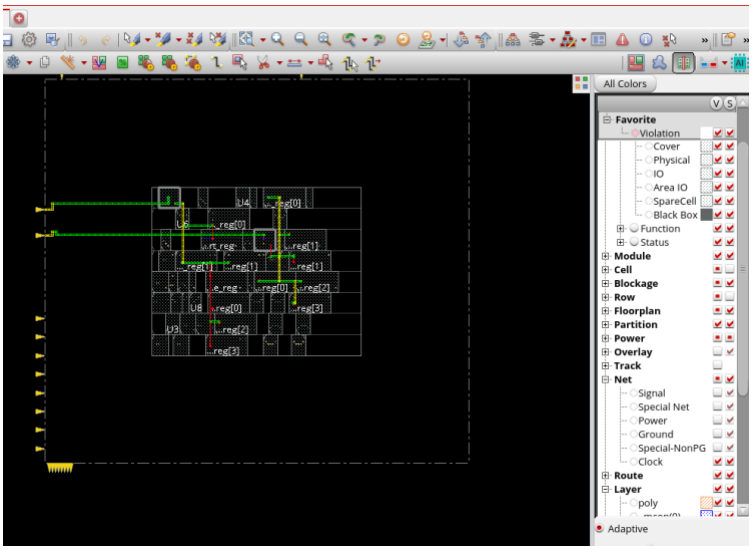


Figure 2. Innovus layout after cts

The CTS also added two drivers for clock tree:

Cell type	Count	Area	Capacitance
Buffers	0	0.000	0.000
Inverters	2	34.279	0.042
Integrated Clock Gates	0	0.000	0.000
Discrete Clock Gates	0	0.000	0.000
Clock Logic	0	0.000	0.000
All	2	34.279	0.042

Figure 3. CTS report

Pegasus

Because the new PDK provides detailed design rules, I used Pegasus to perform DRC and LVS checks. I then fixed reported errors within Innovus. The Pegasus lvs tutorial was in **Appendix B**.

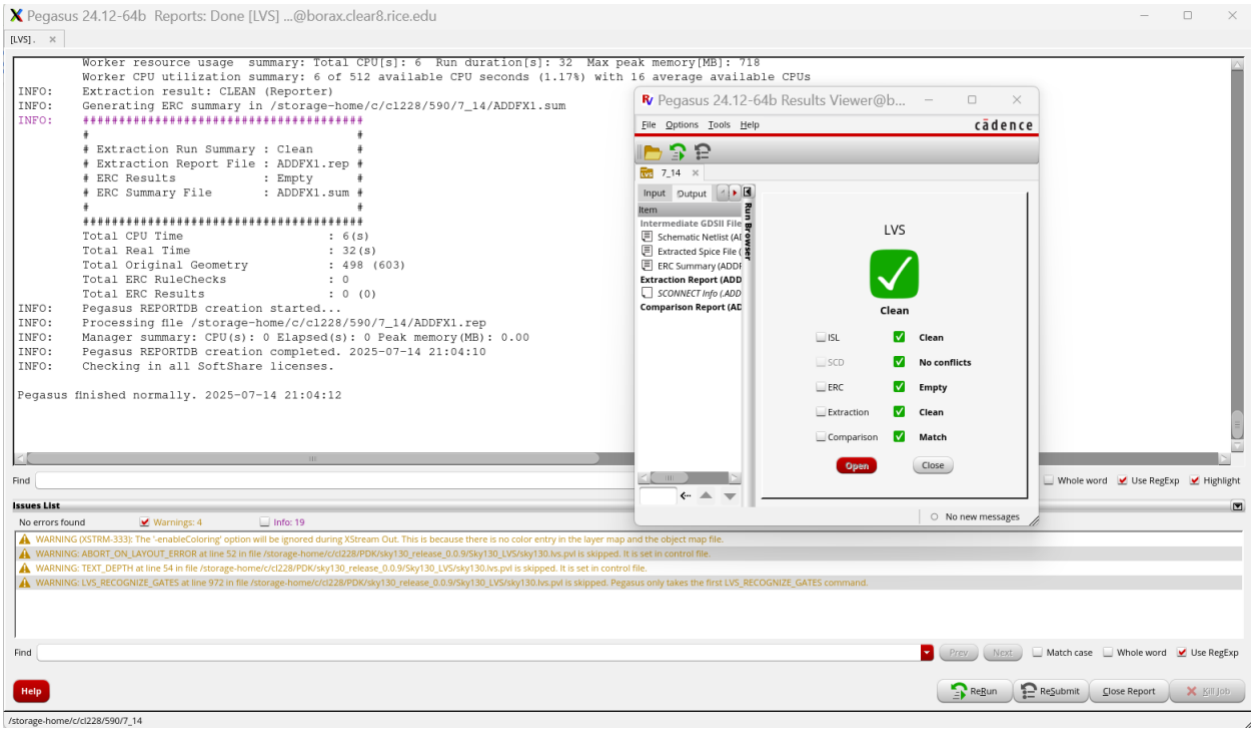


Figure 4. LVS result

DRC results were shown as below:

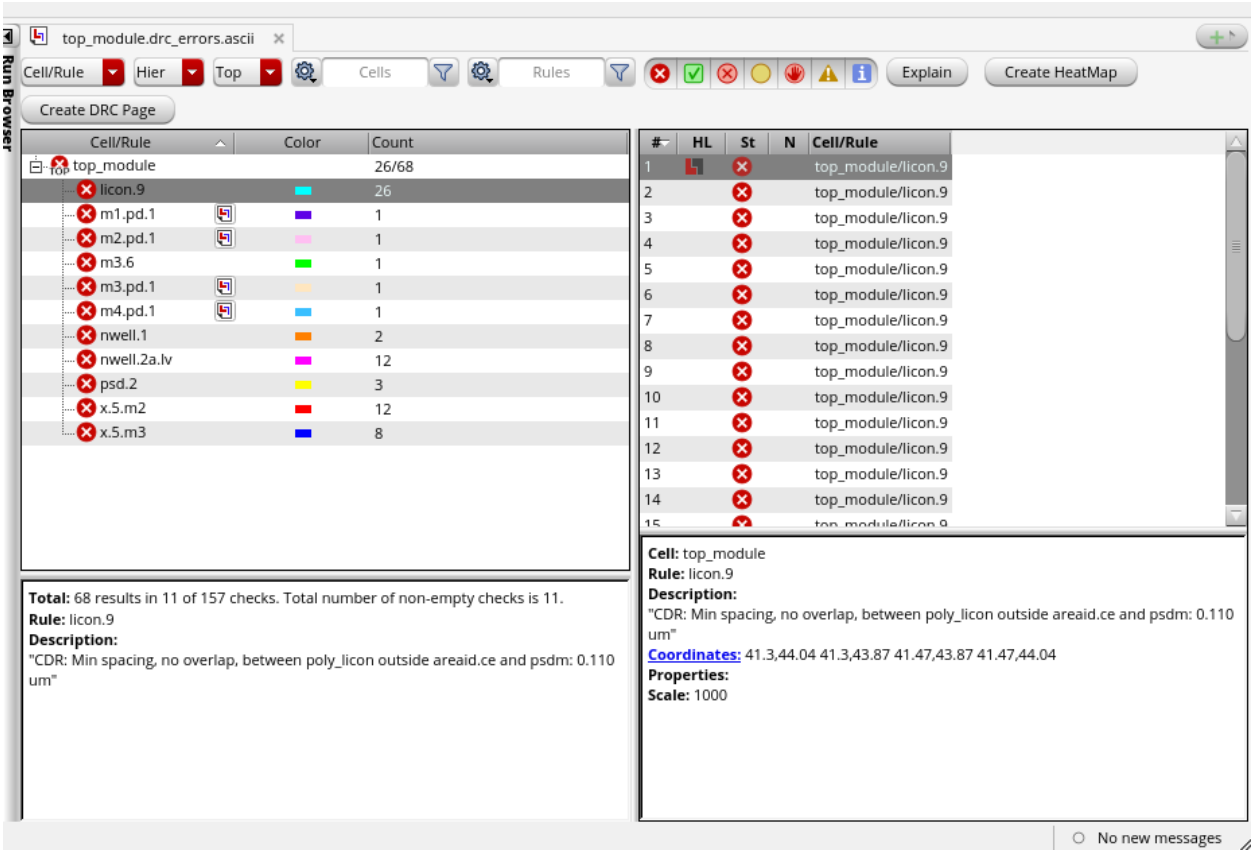


Figure 5. DRC result

- 1. Licon.9: Min spacing, no overlap, between poly_licon outside areaid.ce and psdm: 0.110 um (Unsolved)
- 2. m1.pd.1: CDR: m1.pd.1: Min met1 oxide pattern density: 80% (Unsolved)
- 3. X5.m2: 'All "pin" polygons must be within the 'drawing' polygons of the layer - met3\' (Can be manually solved)
- 3. nwell/psdm error (Should be solved after PDK updated)

Quantus

The Quantus part is not finished yet, as some required technology files are missing from the PDK. I will continue once Cadence updates their PDK.

Spectre(Virtuoso ADE)

Spectre was used for analog simulation. After importing the Verilog file from Innovus into Virtuoso, I used Spectre to simulate the generated schematic. The Spectre tutorial was in **Appendix C**.

The schematic was shown as below:

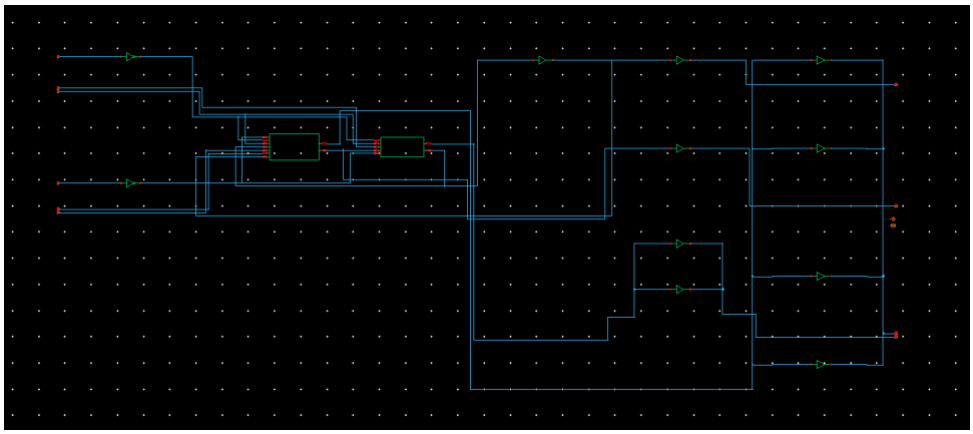


Figure 6. Virtuoso schematic

By manually adding input source:

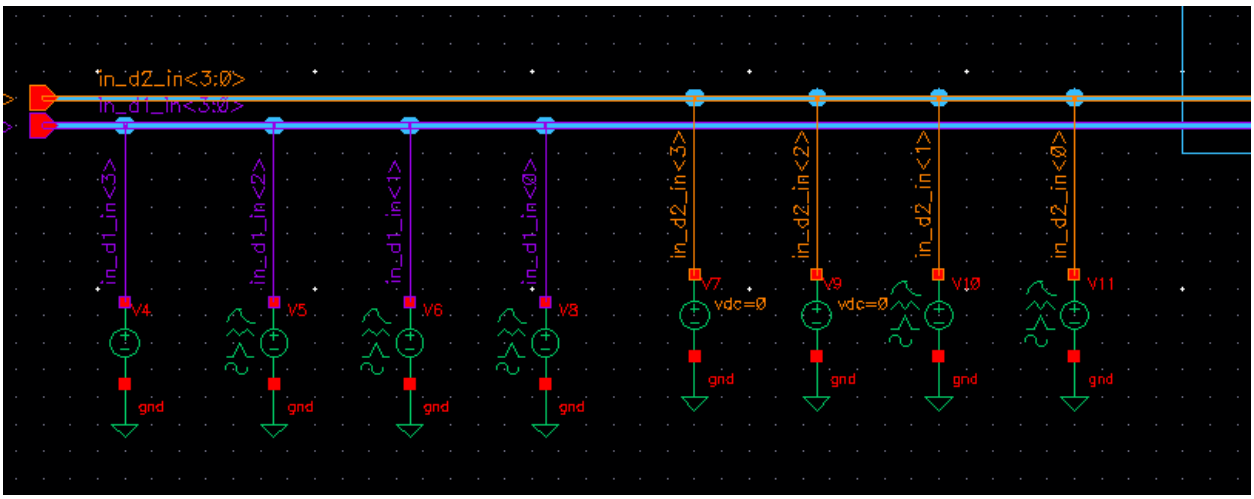


Figure 7. Virtuoso schematic input

Then run spectre simulation:

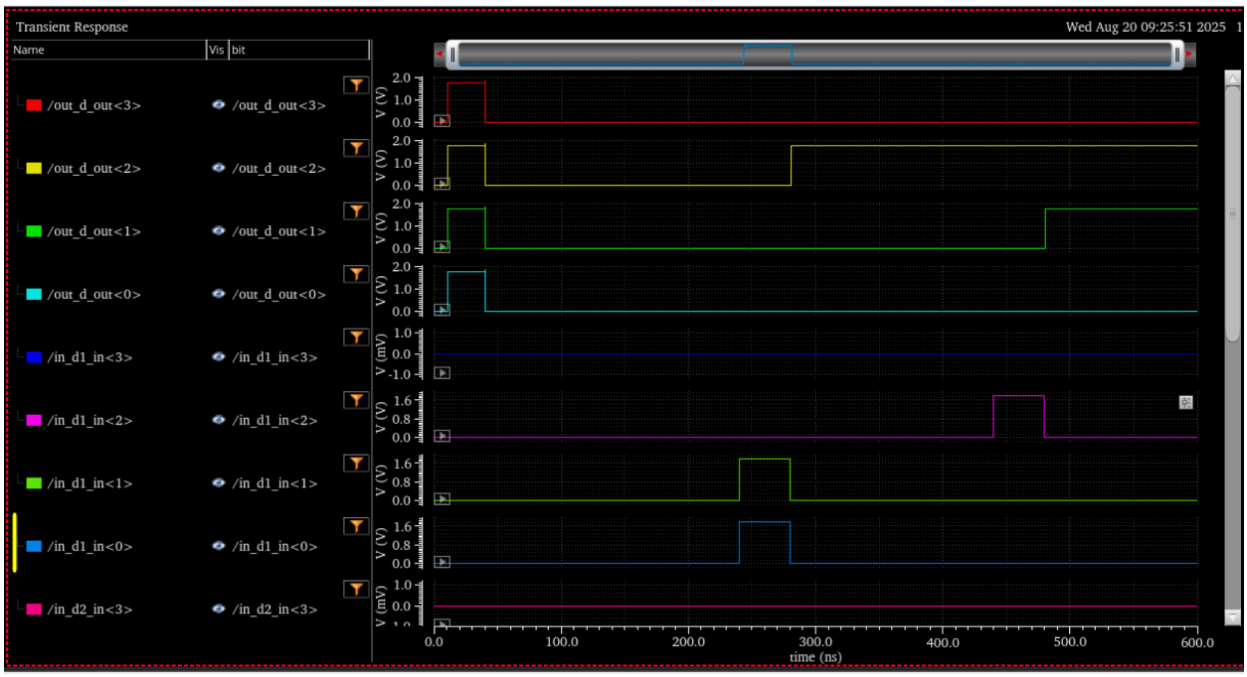


Figure 8. Spectre sate result

The state machine changed from 0 to 1 to 2, which aligns with results from Questa.

Xcelium Logic Simulator

Xcelium Logic Simulator has not been installed yet. I plan to proceed with it once the installation is complete.

Virtuoso

Additionally, I worked with Virtuoso by manually building several standard cells using the Cadence PDK. I ran DRC and LVS checks on them. My future plan is to simulate these cells, similar to what we did with Magic and Irsim in ELEC527. The Virtuoso tutorial was in Appexdix D.

Appendix A

1. Design Compiler code

Tool path: /clear/apps/elec8/bin/dc_shell-t

```
#/************************************************/
#/* Compile Script for Synopsys                  */
#/*                                              */
#/* dc_shell-t -f compile_dc.tcl                 */
#/*                                              */
#/* OSU FreePDK 45nm                             */
```

```
#/* Modified for OSU ami05 - Rice U. 2018          */
#/*                                               */
#/*************************************************/

#/* User Input Data Section: List files, module, clock, frequency below. */
#/* Edit this part only of the file.                                     */
#/* Add all verilog files, separated by spaces after keyword "list"      */

set my_verilog_files [list main_FSM.v dp.v top_module.v]

#/* Top-level Module Name update                                         */
set my_toplevel top_module

#/* The name of the clock pin(s). For multiple clocks, list them all. */
#/* Example for two phase clock: [list clka clkb]                     */
set my_clock_pins [list in_clka in_clkb]

#/* Target frequency in MHz. If clocks have different frequencies,   */
#/* list them in the same order as my_clock_pins.                    */
#/* Example: [list 50 50]                                            */
set my_clk_freqs_MHz [list 50 50]

#/* Delay of input signals (Clock-to-Q, Package etc.)  */
set my_input_delay_ns 1

#/* Reserved time for output signals (Holdtime etc.)   */
set my_output_delay_ns 1


#/*********************************************************************/
#/*********************************************************************/
#/* No modifications needed below. Do not edit or remove.             */
#/*                                                                   */
#/* Paths updated for Rice U. on CLEAR cluster 2018.                  */
#/*********************************************************************/
#/*********************************************************************/

#/* Start of Synopsys library cells location data.                    */
set skywater "/clear/apps/cadence-2025/rhel8/sky130-data/sky130_scl_9T"

set search_path [concat  $search_path $skywater]
set alib_library_analysis_path $skywater
```

```
set link_library [set target_library [concat  [list sky130_tt_1.8_25_nldm_lines_deleted.db] [list dw_foundation.sldb]]]
set target_library "sky130_tt_1.8_25_nldm_lines_deleted.db"
#/* End of Synopsys library cells location data.                */

define_design_lib WORK -path ./WORK
set verilogout_show_unconnected_pins "true"

analyze -f verilog $my_verilog_files

elaborate $my_toplevel

current_design $my_toplevel

link
uniquify


##################### MODIFICATION START ####################

# --- Create all clocks defined in the user list ---
set clock_names [list]
set i 0
foreach clock_pin $my_clock_pins {
   set freq [lindex $my_clk_freqs_MHz $i]
   set period [expr 1000.0 / $freq]

   set find_clock [find port [list $clock_pin]]
   if { $find_clock != [list] } {
      echo "Info: Found clock port '$clock_pin'. Creating clock with period ${period}ns."
      create_clock -name $clock_pin -period $period [get_ports $clock_pin]
      lappend clock_names $clock_pin
   } else {
      echo "Warning: Clock port '$clock_pin' not found in design. Skipping clock creation for it."
   }
   incr i
}

# --- Define the relationship between the two clocks ---
# For a typical two-phase non-overlapping clock, they are logically exclusive.
if { [llength $clock_names] > 1 } {
   echo "Info: Defining clock groups as logically exclusive."
   set_clock_groups -logically_exclusive -group [get_clocks [lindex $clock_names 0]] -group [get_clocks [lindex $clock_names 1]]
```

```
}


# --- Set I/O constraints with respect to the correct clock phase ---
# The I/O ports are grouped based on their likely corresponding clock phase.
# Please verify this matches your design's specific architecture.
if { [llength $clock_names] > 0 } {

    echo "Info: Setting I/O constraints for the multi-clock design."

    # Set a default driving cell for all inputs
    set_driving_cell  -lib_cell INVX1 [all_inputs]

    # --- Constrain I/Os related to 'in_clka' (Control Signals) ---
    # Define the list of control inputs and outputs
    set clka_inputs [get_ports {in_restart in_load}]
    set clka_outputs [get_ports {out_start out_done}]

    if {[llength $clka_inputs] > 0} {
        set_input_delay $my_input_delay_ns -clock [get_clocks in_clka] $clka_inputs
    }
    if {[llength $clka_outputs] > 0} {
        set_output_delay $my_output_delay_ns -clock [get_clocks in_clka] $clka_outputs
    }

    # --- Constrain I/Os related to 'in_clkb' (Data Signals) ---
    # Define the list of data inputs and outputs
    set clkb_inputs [get_ports {in_d1_in[*] in_d2_in[*]}]
    set clkb_outputs [get_ports {out_state_main[*] out_d_out[*]}]

    if {[llength $clkb_inputs] > 0} {
        set_input_delay $my_input_delay_ns -clock [get_clocks in_clkb] $clkb_inputs
    }
    if {[llength $clkb_outputs] > 0} {
        set_output_delay $my_output_delay_ns -clock [get_clocks in_clkb] $clkb_outputs
    }

}
##################### MODIFICATION END ####################


compile -ungroup_all -map_effort medium

compile -incremental_mapping -map_effort medium
```

```
check_design
report_constraint -all_violators

set filename [format "%s%s"  $my_toplevel ".vh"]
write -f verilog -output $filename

set filename [format "%s%s"  $my_toplevel ".sdc"]
write_sdc $filename

report_cell
report_area
report_timing -path full -delay max -max_paths 3 -nworst 1
report_power

redirect timing.rep { report_timing }
redirect cell.rep { report_cell }
redirect power.rep { report_power }

quit
```

2. Genus code
Tool path: /clear/apps/elec8/bin/genus-2025

```
set_db common_ui false

set skywater "/clear/apps/cadence-2025/rhel8/sky130-data/sky130_scl_9T"



set_attr init_hdl_search_path {./}
set_attr init_lib_search_path "$skywater/sky130_scl_9T/lib"

set_attribute library {sky130_tt_1.8_25_nldm.lib}

set_attribute information_level 6

set rtlFiles [list top_module.v dp.v main_FSM.v]
set basename top_module     ;# name of top level module
set runname gates_genus     ;# name appended to output files(every file will have the same name
for runname appended)


set frequency    25.0          ;# Clock frequency in MHz
set clk1_port_name  "in_clka"  ;# Physical port name for clk1 in Verilog
```

```
set clk1_logic_name "clka"     ;# Logical name for clk1 in constraints
set clk2_port_name  "in_clkb"  ;# Physical port name for clk2 in Verilog
set clk2_logic_name "clkb"     ;# Logical name for clk2 in constraints
set input_setup  1             ;# delay from clock to inputs valid
set output_delay 1             ;# delay from clock to output valid
set clock_period [expr 1000000.0 / $frequency]   ;# Clock period in ps (picosecond)
set half_period [expr ${clock_period} / 2.0]

read_hdl -sv ${rtlFiles}
elaborate ${basename}

#sizing of the transistors part of research, try changing these values randomly
set_drive     2.0 [all_inputs]
set_load      4.0 [all_outputs]
set_max_fanout 5   [all_inputs]


# Step 1: Create the first phase clock (phi1), high for the first half of the period.


set clock [define_clock -period ${clock_period} -name ${clk1_logic_name} \
   -waveformr 0 -waveformf ${half_period} [get_ports ${clk1_port_name}]]

# Step 2: Create the second phase clock (phi2), high for the second half of the period.
set clock [define_clock -period ${clock_period} -name ${clk2_logic_name} \
   -waveformr ${half_period} -waveformf ${clock_period} [get_ports ${clk2_port_name}]]


# Step 4: Set I/O Delays, making sure to exclude clock ports
set data_inputs [remove_from_collection [all_inputs] [get_ports "${clk1_port_name}
${clk2_port_name}"]]
set data_outputs [all_outputs]

set_input_delay $input_setup -clock ${clk1_logic_name} $data_inputs
set_output_delay $output_delay -clock ${clk2_logic_name} $data_outputs



# Step 5: Check design and timing constraints
check_design -unresolved
report timing -lint

#--------------------------------------------------------------------
# --- Synthesis ---
```

```
#--------------------------------------------------------------------
# Synthesize the design to the target library
# synthesize -effort medium -to_mapped
set_attribute syn_generic_effort medium
set_attribute syn_map_effort medium
set_attribute syn_opt_effort medium
syn_generic
syn_map
syn_opt

#--------------------------------------------------------------------
# --- Reporting and Output ---
#--------------------------------------------------------------------
# Write out the reports
report timing > ${basename}_${runname}_timing.rep
report gates  > ${basename}_${runname}_cell.rep
report power  > ${basename}_${runname}_power.rep

# Write out the structural Verilog and sdc files
write_hdl -mapped >  ${basename}_${runname}.vh
write_sdc >  ${basename}_${runname}.sdc

# show result
gui_show
```

3. .view code

```
# Create RC corner
create_rc_corner -name RC_TYP -qx_tech_file /clear/apps/cadence-2025/rhel8/sky130-data/sky130_release/quantus/extraction/typical/qrcTechFile -pre_route_res 1.0 -pre_route_cap 1.0

create_rc_corner -name RC_SLOW -qx_tech_file /clear/apps/cadence-2025/rhel8/sky130-data/sky130_release/quantus/extraction/typical/qrcTechFile -pre_route_res 1.2 -pre_route_cap 1.2

# Create library set
create_library_set -name lib_tt -timing [list /clear/apps/cadence-2025/rhel8/sky130-data/sky130_scl_9T/sky130_scl_9T/lib/sky130_tt_1.8_25_nldm.lib]
```

```
create_library_set -name lib_ss -timing [list /storage-home/c/cl228/590/sky130_scl_9T_0.0.6/sky130_scl_9T/lib/sky130_ss_1.62_125_nldm.lib]


# Define constraints
create_constraint_mode -name func_mode -sdc ../genus/top_module_gates_genus.sdc


# Define timing corners
create_delay_corner -name tc_tt -library_set lib_tt -rc_corner RC_TYP

create_delay_corner -name tc_ss -library_set lib_ss -rc_corner RC_SLOW

# Define analysis views
create_analysis_view -name func_tt -constraint_mode func_mode -delay_corner tc_tt

create_analysis_view -name func_ss -constraint_mode func_mode -delay_corner tc_ss

# Set default views
set_analysis_view -setup [list func_tt] -hold [list func_ss]
```

4. Innovus tcl code

Tool path: /clear/apps/elec8/bin/innovus-2025

```
##########################################################################
#
#  Innovus Command File - updated Feb. 22, 2018 for ELEC 422/527
#  Run the design through Innovus 22.14
#  Minor text typo update Feb. 12, 2019
#  Minor update for Innovus Apr. 06, 2024
#
##########################################################################

##########################################################################
##########################################################################
# IMPORTANT - IMPORTANT
# Change only these three lines each time to your top level cell names

#Set top module
set init_top_cell "top_module"
#Load netlist
set init_verilog "top_module.vh"
```

```
#Set mmmc file
set init_mmmc_file "top_module.view"

##########################################################################
##########################################################################
#
# No need to change anything below - standard design flow steps
# IMPORTANT - See note below about floorPlan if density error occurs
#
##########################################################################
##########################################################################
#

set skywater "/clear/apps/cadence-2025/rhel8/sky130-data/sky130_scl_9T"

# Set the Library Exchange Format cell library file wrt. the OSU root
set init_lef_file [list \
                "$skywater/sky130_scl_9T_tech/lef/sky130_scl_9T.tlef" \
                "$skywater/sky130_scl_9T/lef/sky130_scl_9T.lef" \
                "$skywater/sky130_scl_9T_tech/lef/sky130_scl_9T_phyCells.lef" \
]

set init_gnd_net VSS
set init_design_settop 0
set init_pwr_net VDD

init_design


getIoFlowFlag
setIoFlowFlag 0

floorPlan -r 1.0 0.6 21 21 21 21

globalNetConnect VDD -type pgpin -pin VDD -inst *
globalNetConnect VSS -type pgpin -pin VSS -inst *

set sprCreateIeRingNets {}
set sprCreateIeRingLayers {}
set sprCreateIeRingWidth 1.2
set sprCreateIeRingSpacing 1.2
set sprCreateIeRingOffset 1.2
set sprCreateIeRingThreshold 1.2
set sprCreateIeRingJogDistance 1.2
```

```
setPlaceMode -placeIoPins true

setAddRingMode -stacked_via_top_layer met5
setAddRingMode -stacked_via_bottom_layer met1

# Add Power and Ground rings around core
addRing -skip_via_on_wire_shape Noshape -skip_via_on_pin Standardcell -center 1 \
-type core_rings -jog_distance 1.8 -threshold 1.8 -nets {VDD VSS} -follow core \
-layer {bottom met1 top met1 right met2 left met2} -width 4.0 -spacing 1.8 -offset 1.8


addStripe -nets {VDD VSS} -layer met4 -direction vertical -width 1 -spacing 1 -set_to_set_distance 20
addStripe -nets {VDD VSS} -layer met3 -direction horizontal -width 1 -spacing 1 -set_to_set_distance 20

verifyConnectivity -type special -noAntenna -noWeakConnect -noUnroutedNet -error 1000 -warning 50 -net VDD


verifyConnectivity -type special -noAntenna -noWeakConnect -noUnroutedNet -error 1000 -warning 50 -net VSS
verify_drc



set_interactive_constraint_modes [all_constraint_modes -active]


set inputs [get_object_name [get_ports {in_clka in_clkb in_restart in_load in_d1_in[*] in_d2_in[*]}]]
set all_output_ports [get_object_name [all_outputs]]


editPin -spreadDirection counterclockwise -spacing 0.001 -layer 5 -pin $all_output_ports
editPin -spreadDirection clockwise -spacing 0.002 -layer 5 -pin $inputs


# Set data transition & clock transition
set_max_tran 0.3 [current_design]
set_max_transition 0.12 -clock [all_clocks]
# Set maximum fanout
set_max_fanout 24 [current_design]
```

```
# Set OCV
setAnalysisMode -analysisType onChipVariation -cppr both
# Crosstalk
setDelayCalMode -engine default -siAware true


Placement setting
setPlaceMode -reset
setPlaceMode -congEffort auto -timingDriven 1 -modulePlan 1 -clkGateAware 1 -powerDriven 0 -ignoreScan 1 -reorderScan 0 -ignoreSpare 1 -placeIOPins 0 -moduleAwareSpare 0 -preserveRouting 0 -rmAffectedRouting 0 -checkRoute 0 -swapEEQ 0

setPlaceMode -place_detail_legalization_inst_gap 2
#setPlaceMode -fp false
#setOptMode -allEndPoints true
#setPlaceMode -place_global_uniform_density true
#setPlaceMode -place_opt_effort high
#setOptMode -leakagePowerOptimization true
#setOptMode -powerEffort low


setRouteMode -earlyGlobalEffortLevel standard


checkPlace

place_opt_design
saveDesign ${init_top_cell}_placement.enc

sroute -nets {VDD VSS}


# Clock Tree Synthesis
# set_ccopt_property use_inverters true

# Set clock tree constrains
 set_ccopt_property target_max_trans 0.4
 set_ccopt_property target_skew 0.06
 set_ccopt_property max_fanout 24



set clock_buffers "CLKBUFX2 CLKBUFX4 CLKBUFX8"
set clock_inverters "CLKINVX1 CLKINVX2 CLKINVX4 CLKINVX8"
```

```
set_ccopt_property use_inverters true
set_ccopt_property inverter_cells $clock_inverters


#ccopt_design -cts
clock_opt_design

report_ccopt_skew_groups -summary
saveDesign ${init_top_cell}_cts.enc


setNanoRouteMode -route_extra_via_enclosure 1

routeDesign -globalDetail

setNanoRouteMode -route_detail_post_route_wire_widen 3

saveDesign ${init_top_cell}_route.enc

# Add Filler cells at edges of cell rows
#addFiller -cell FILL -prefix FILLER -markFixed

addFiller -cell FILL1 \
            FILL2  \
            FILL4  \
            FILL8  \
            FILL16 \
            FILL32 \
            FILL64 \
        -prefix FILL

#ecoRoute

# Run DRC and Connection checks
verifyGeometry
verifyConnectivity -type all -noAntenna


# Output GDSII and include layout of subcells to later view in magic
setStreamOutMode -snapToMGrid true -supportPathType4 false
streamOut  final.gds  -mapFile  $skywater/sky130_scl_9T/gds/sky130_stream.mapFile  -libName
DesignLib -units 1000 -merge $skywater/sky130_scl_9T/gds/sky130_scl_9T.gds -mode ALL
```

```
# Output a final Verilog file and design database with connectivity modeled
saveNetlist final.v
saveDesign $init_top_cell.enc


#/clear/apps/cadence-2025/rhel8/sky130-data
```

5. Modified verilog test bench

```
`timescale 1ns/1ps
//------------------------------------------------------
// File    : testbench_top.v
// Purpose : Generate precise stimuli matching the
//           required timing, and export each input
//           as Spectre-readable PWL files (step style).
// Notes   : Verilog-2001; no DUT instantiated.
//           Logic '1' = 1.8 V, '0' = 0 V.
//------------------------------------------------------

module testbench_top;

  // -------- Stimulus signals --------
  reg       in_clka, in_clkb, in_restart, in_load;
  reg  [3:0] in_d1_in;
  reg  [3:0] in_d2_in;

  // -------- PWL export handles & state --------
  real VDD_V;
  integer f_clka, f_clkb, f_rst, f_load;
  integer f_d1_0, f_d1_1, f_d1_2, f_d1_3;
  integer f_d2_0, f_d2_1, f_d2_2, f_d2_3;

  real prev_clka, prev_clkb, prev_rst, prev_load;
  real prev_d1_0, prev_d1_1, prev_d1_2, prev_d1_3;
  real prev_d2_0, prev_d2_1, prev_d2_2, prev_d2_3;

  // ---- helpers: write initial point (t=0s) ----
  task write_init;
    input integer fd; input real v0;
    begin
      $fwrite(fd, "%0.6e %0.6f\n", 0.0, v0); // (time[s], volt[V])
    end
```

```
endtask

// ---- helpers: write a step at current time ----
// keep previous value up to t, then jump to new value at t+1ps
task write_step;
  input integer fd; input b; inout real prev;
  real vnew, t;
  begin
    vnew = b ? 1.8 : 0.0;
    t    = $realtime * 1e-9;        // ns -> s
    $fwrite(fd, "%0.6e %0.6f\n", t,           prev);
    $fwrite(fd, "%0.6e %0.6f\n", t+1.0e-12,   vnew);
    prev = vnew;
  end
endtask

// ---- open files, init values & t=0 point ----
initial begin
  VDD_V = 1.8;

  in_restart = 0; in_load = 0;
  in_d1_in   = 4'b0000; in_d2_in = 4'b0000;
  in_clka    = 0; in_clkb = 0;

  f_clka = $fopen("in_clka.pwl","w");
  f_clkb = $fopen("in_clkb.pwl","w");
  f_rst  = $fopen("in_restart.pwl","w");
  f_load = $fopen("in_load.pwl","w");
  f_d1_0 = $fopen("in_d1_in_0.pwl","w");
  f_d1_1 = $fopen("in_d1_in_1.pwl","w");
  f_d1_2 = $fopen("in_d1_in_2.pwl","w");
  f_d1_3 = $fopen("in_d1_in_3.pwl","w");
  f_d2_0 = $fopen("in_d2_in_0.pwl","w");
  f_d2_1 = $fopen("in_d2_in_1.pwl","w");
  f_d2_2 = $fopen("in_d2_in_2.pwl","w");
  f_d2_3 = $fopen("in_d2_in_3.pwl","w");

  prev_clka = 0.0; prev_clkb = 0.0;
  prev_rst  = 0.0; prev_load = 0.0;
  prev_d1_0 = 0.0; prev_d1_1 = 0.0; prev_d1_2 = 0.0; prev_d1_3 = 0.0;
  prev_d2_0 = 0.0; prev_d2_1 = 0.0; prev_d2_2 = 0.0; prev_d2_3 = 0.0;


end
```

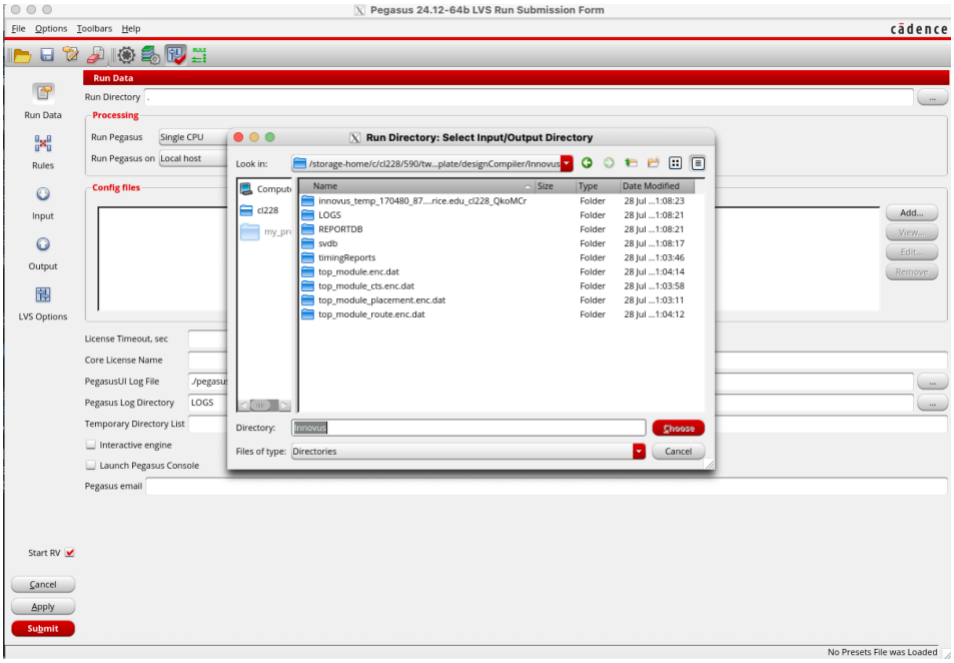
```
// ---- auto-log to PWL on any change ----
always @(in_clka)       write_step(f_clka, in_clka, prev_clka);
always @(in_clkb)       write_step(f_clkb, in_clkb, prev_clkb);
always @(in_restart)    write_step(f_rst,  in_restart, prev_rst);
always @(in_load)       write_step(f_load, in_load,    prev_load);

always @(in_d1_in[0])   write_step(f_d1_0, in_d1_in[0], prev_d1_0);
always @(in_d1_in[1])   write_step(f_d1_1, in_d1_in[1], prev_d1_1);
always @(in_d1_in[2])   write_step(f_d1_2, in_d1_in[2], prev_d1_2);
always @(in_d1_in[3])   write_step(f_d1_3, in_d1_in[3], prev_d1_3);

always @(in_d2_in[0])   write_step(f_d2_0, in_d2_in[0], prev_d2_0);
always @(in_d2_in[1])   write_step(f_d2_1, in_d2_in[1], prev_d2_1);
always @(in_d2_in[2])   write_step(f_d2_2, in_d2_in[2], prev_d2_2);
always @(in_d2_in[3])   write_step(f_d2_3, in_d2_in[3], prev_d2_3);

// -------- precise timing spec --------
// clocks: period=40ns, width=10ns; clka delay=10ns, clkb delay=30ns
initial begin : gen_clka
  integer k;
  in_clka = 0;
  #10; // delay 10ns
  for (k=0; k<100; k=k+1) begin
    in_clka = 1; #10;   // high 10ns
    in_clka = 0; #30;   // low 30ns  -> 40ns period
    if ($realtime >= 620.0) disable gen_clka; // safety stop around 620ns
  end
end

initial begin : gen_clkb
  integer k;
  in_clkb = 0;
  #30; // delay 30ns
  for (k=0; k<100; k=k+1) begin
    in_clkb = 1; #10;   // high 10ns
    in_clkb = 0; #30;   // low 30ns
    if ($realtime >= 620.0) disable gen_clkb;
  end
end

// control pulses
initial begin
  // in_restart: 40ns -> 80ns
```

```
    #40;  in_restart = 1;
    #40;  in_restart = 0;

    // in_load: 200ns -> 240ns, and 400ns -> 440ns
    #80;
    #40;  /* 200ns */ in_load = 1;
    #40;  /* 240ns */ in_load = 0;
    #160; /* 400ns */ in_load = 1;
    #40;  /* 440ns */ in_load = 0;
  end

  // data bus sequences
  initial begin
    // d1: 0 -> 3 (0011) @200–240ns, then 0 -> 4 (0100) @240–280ns, then 0
    #240; in_d1_in = 4'b0011;
    #40;  in_d1_in = 4'b0000;
    #160;  in_d1_in = 4'b0100;

  end

  initial begin
    // write d2 separately to avoid timing confusion
    // d2: 0 -> 1 -> 0 -> 2
    // start from t=0
    #240; in_d2_in = 4'b0001;
    #40;  in_d2_in = 4'b0000;
    #160;  in_d2_in = 4'b0010;
  end

  // end sim
  initial begin
    #620; // make sure we cover up to ~600ns+
    $fclose(f_clka); $fclose(f_clkb);
    $fclose(f_rst);  $fclose(f_load);
    $fclose(f_d1_0); $fclose(f_d1_1); $fclose(f_d1_2); $fclose(f_d1_3);
    $fclose(f_d2_0); $fclose(f_d2_1); $fclose(f_d2_2); $fclose(f_d2_3);
    $finish;
  end

endmodule
```

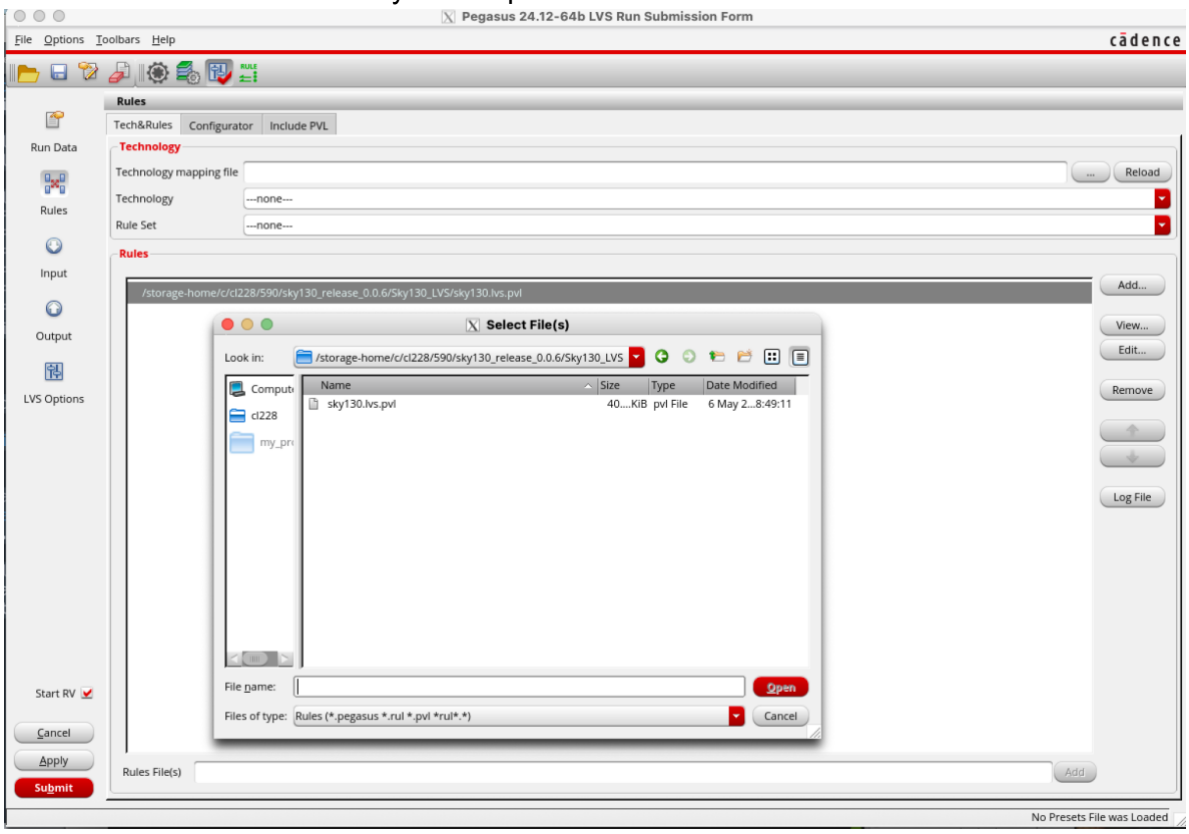
Appendix B

Tool path: /clear/apps/elec8/bin/virtuoso-2025

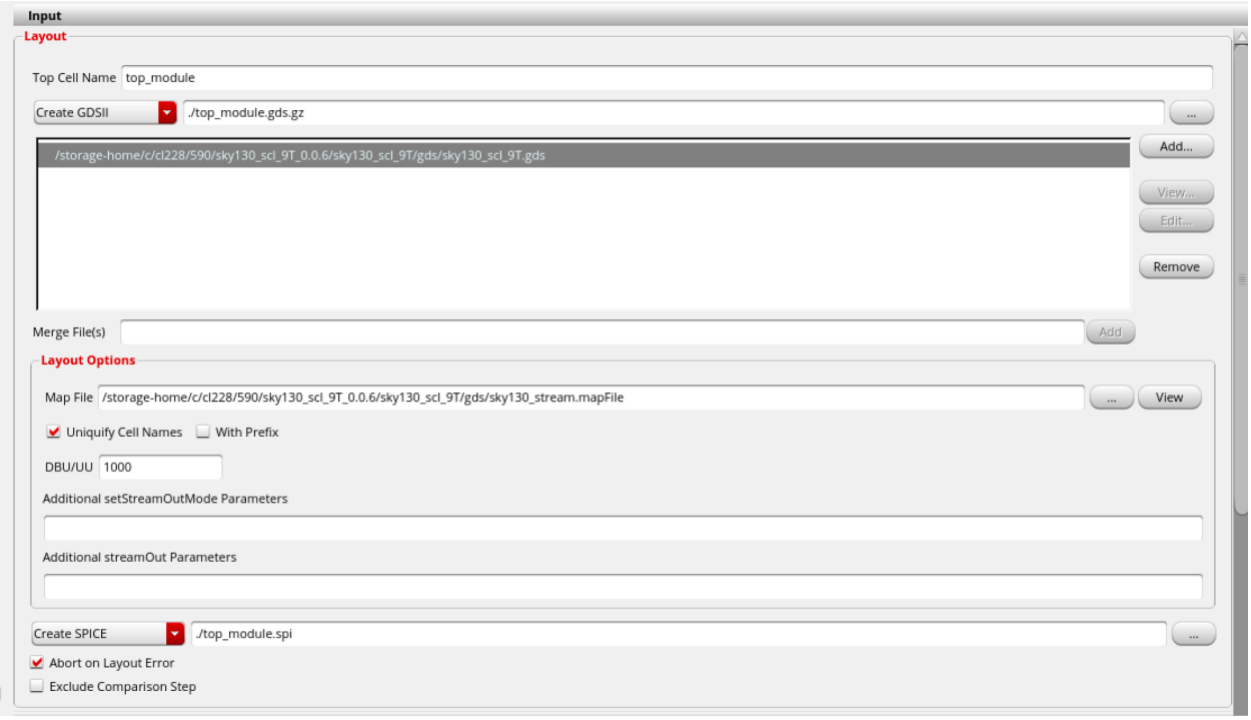
1. Set Run Directory



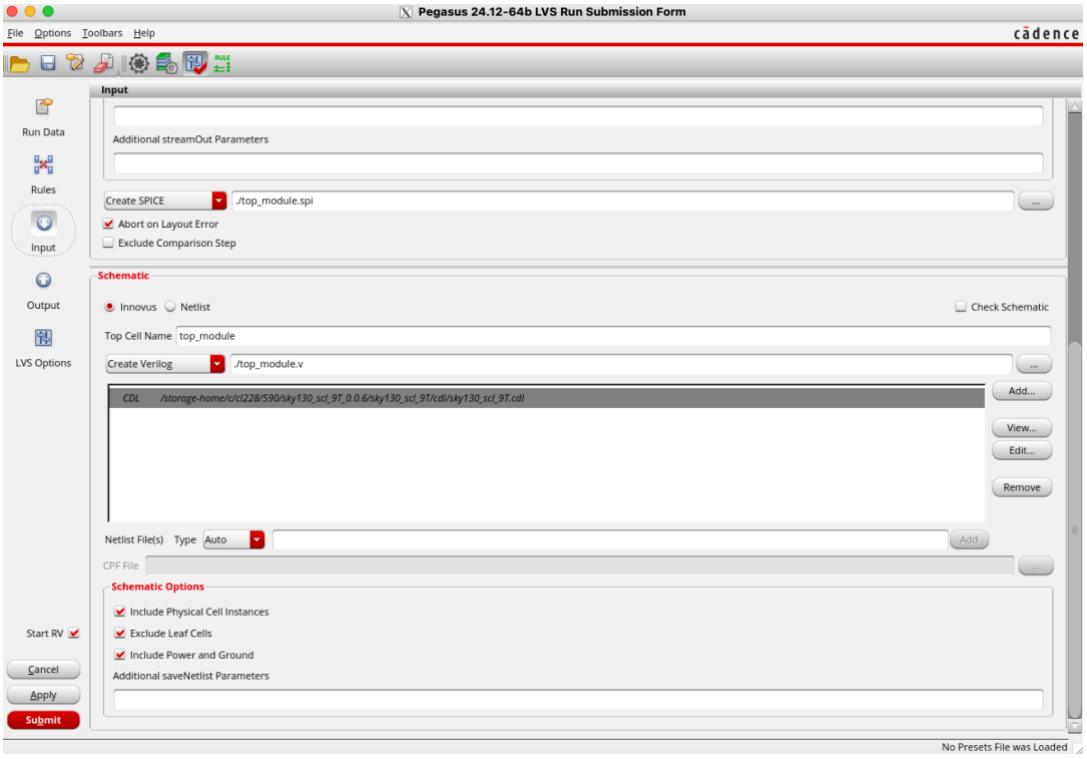
2. Choose “Rules”
   Click “Add…” and add sky130.lvs.pvl file



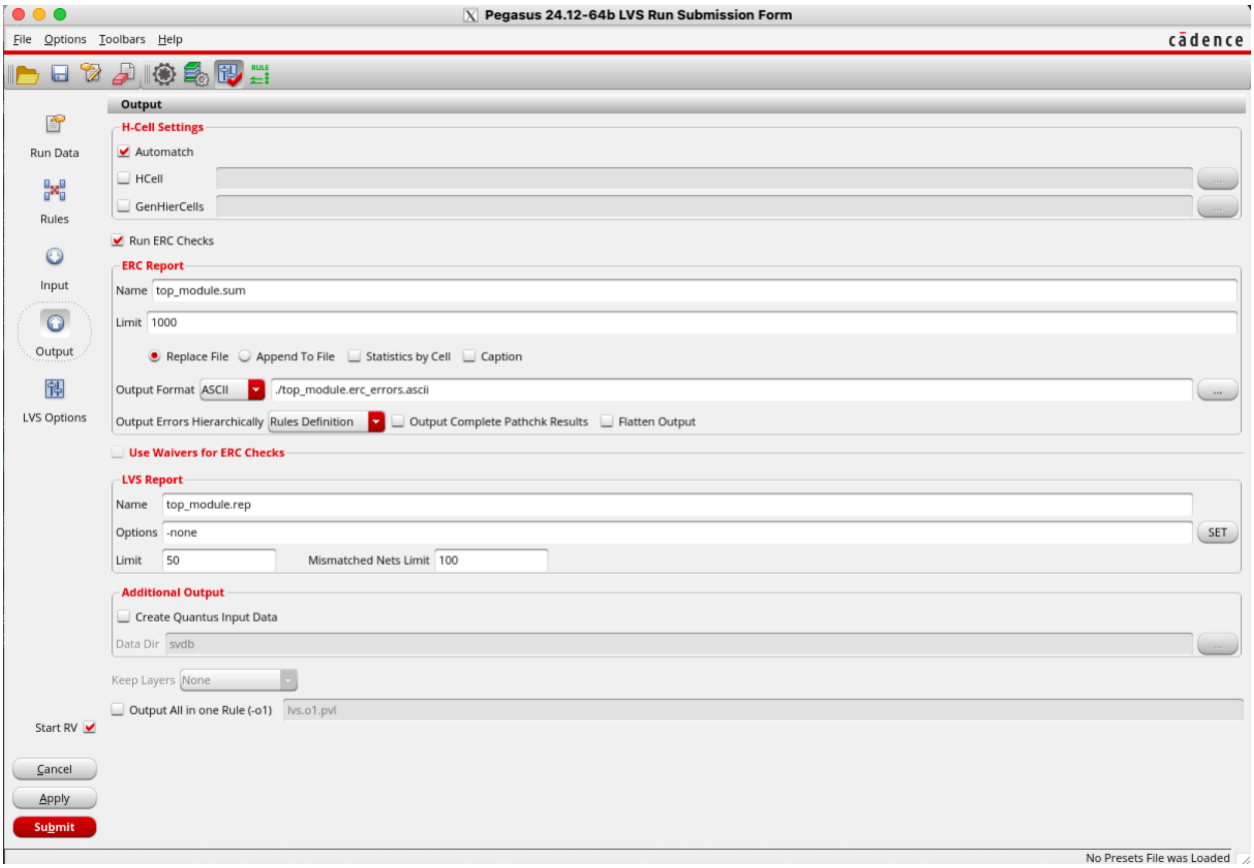
3. Choose Input
   Add gds file from PDK
   Then set Mapfile. It should be in the same directory with gds file
   Make sure it creates SPICE file



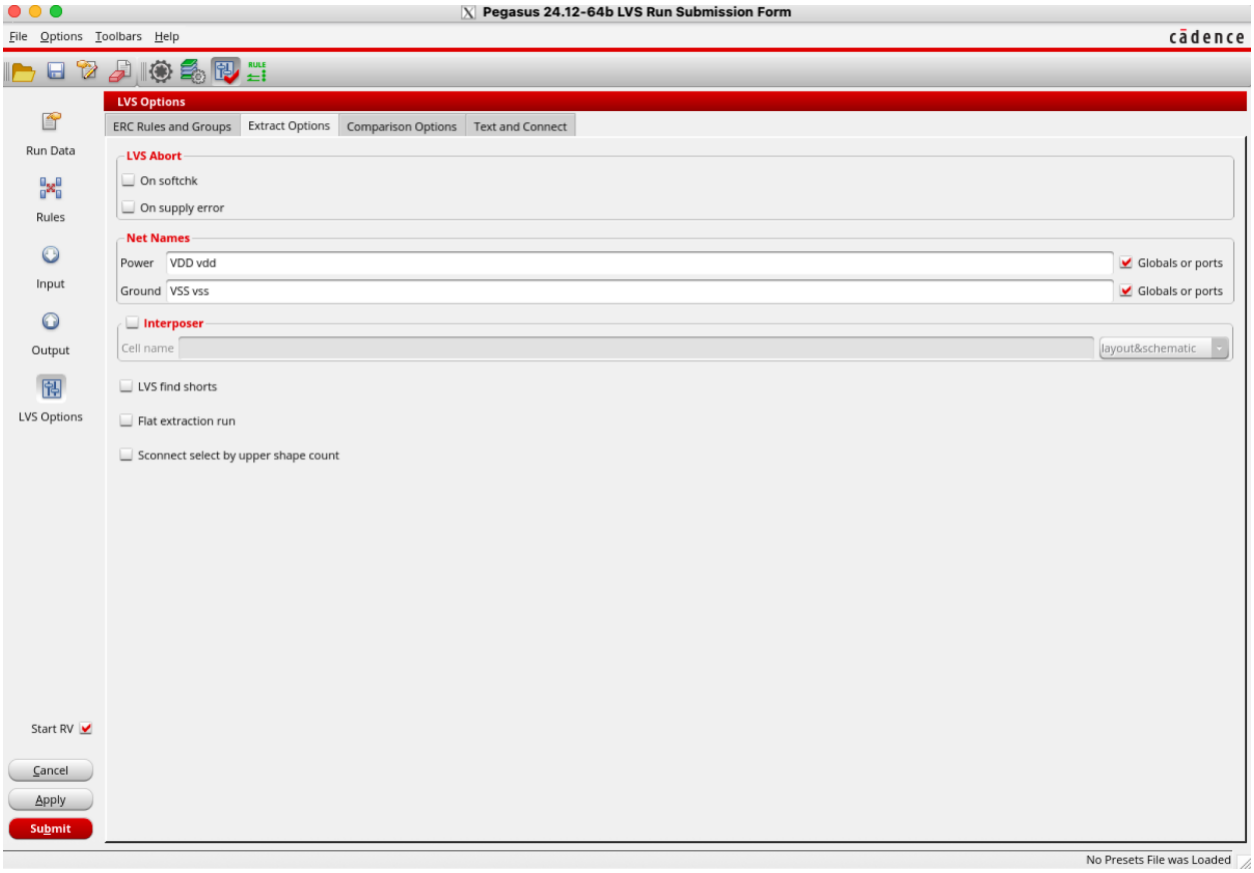
4. Scroll down to schematic
   Add cdl file



5. Choose Output
   Check “Automatch” in H-Cell Settings
   Check “Create Quantus Input Data” in Additional Output if need to run Quantus later

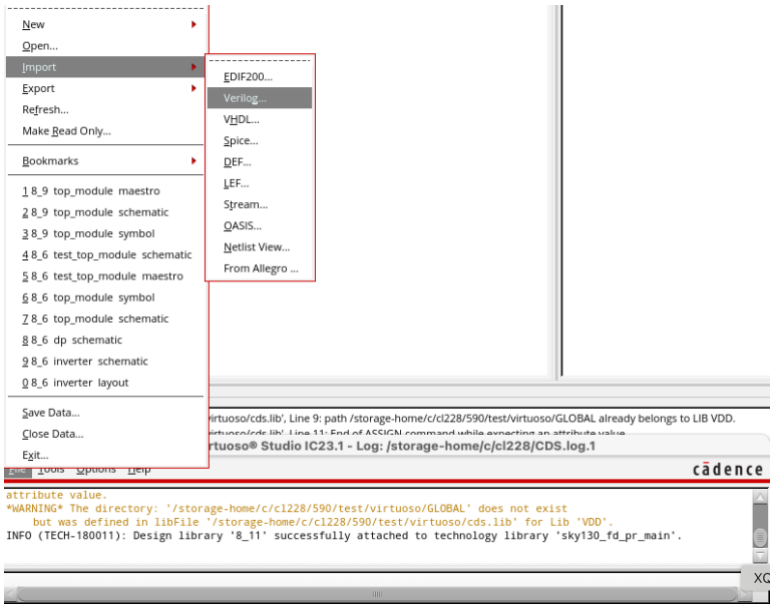


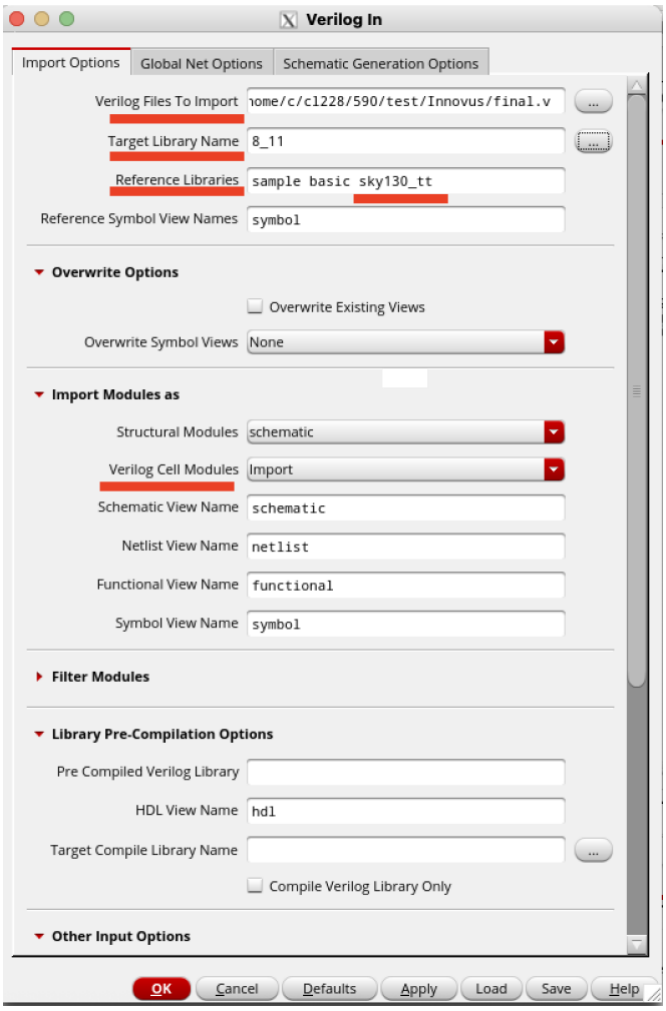
6. LVS options->Extract Options
   Define Power and Ground. (I'm not sure what this step does, but the tutorial I read does it this way.)



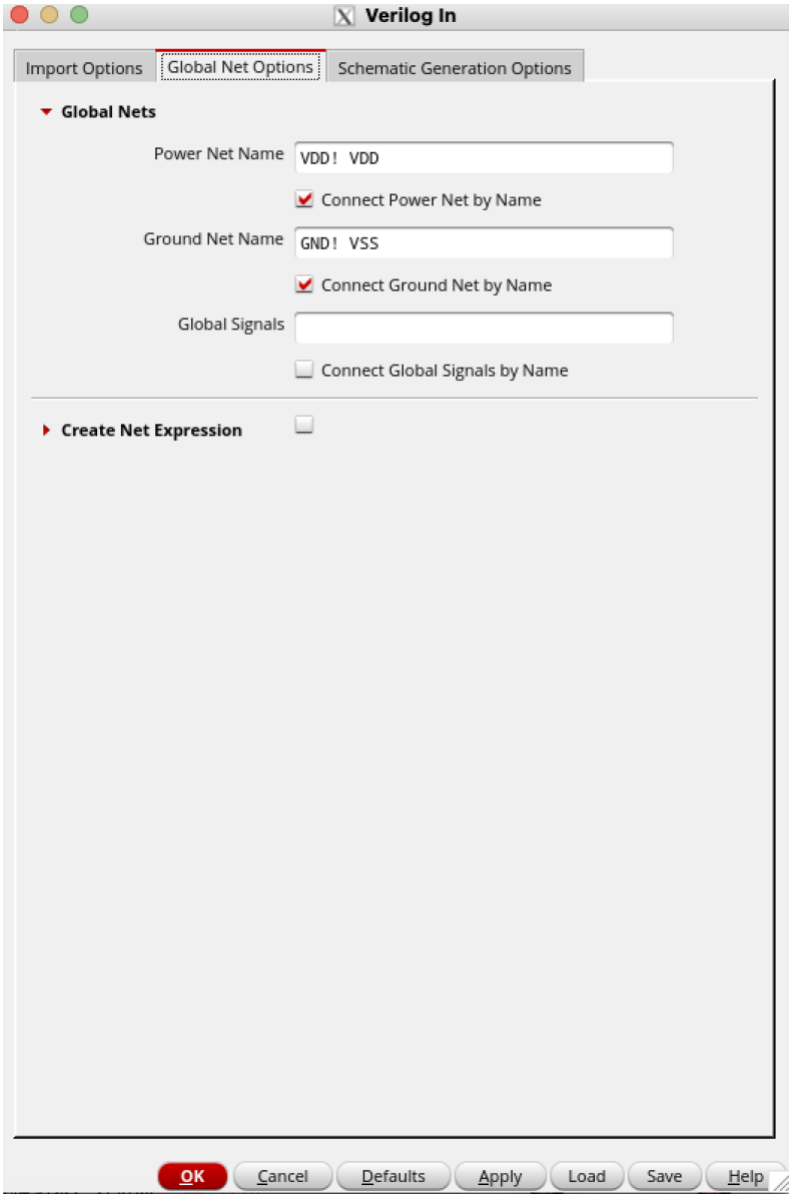
Appendix C

1. Import verilog file to virtuoso

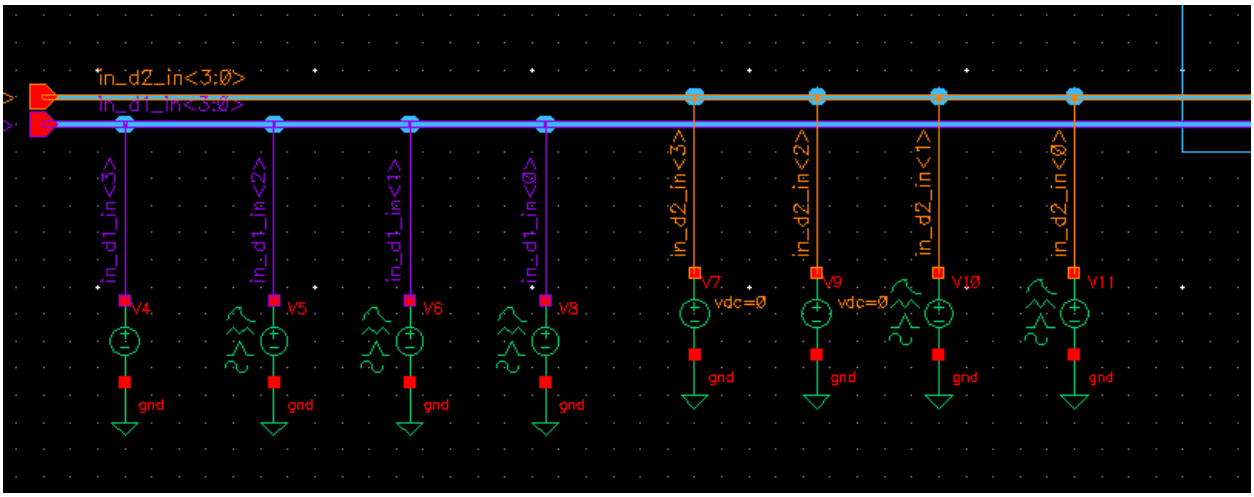




Set global nets

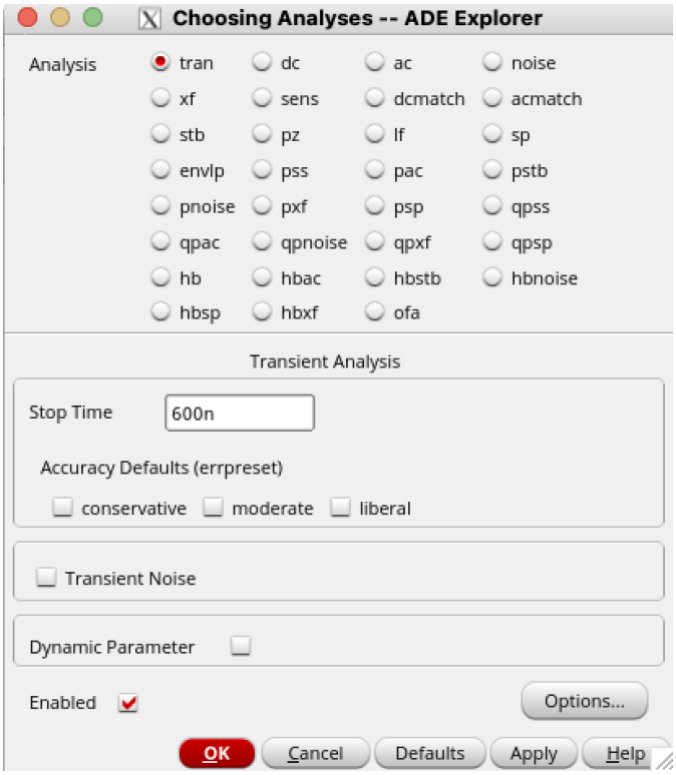


Then added input source in schematic



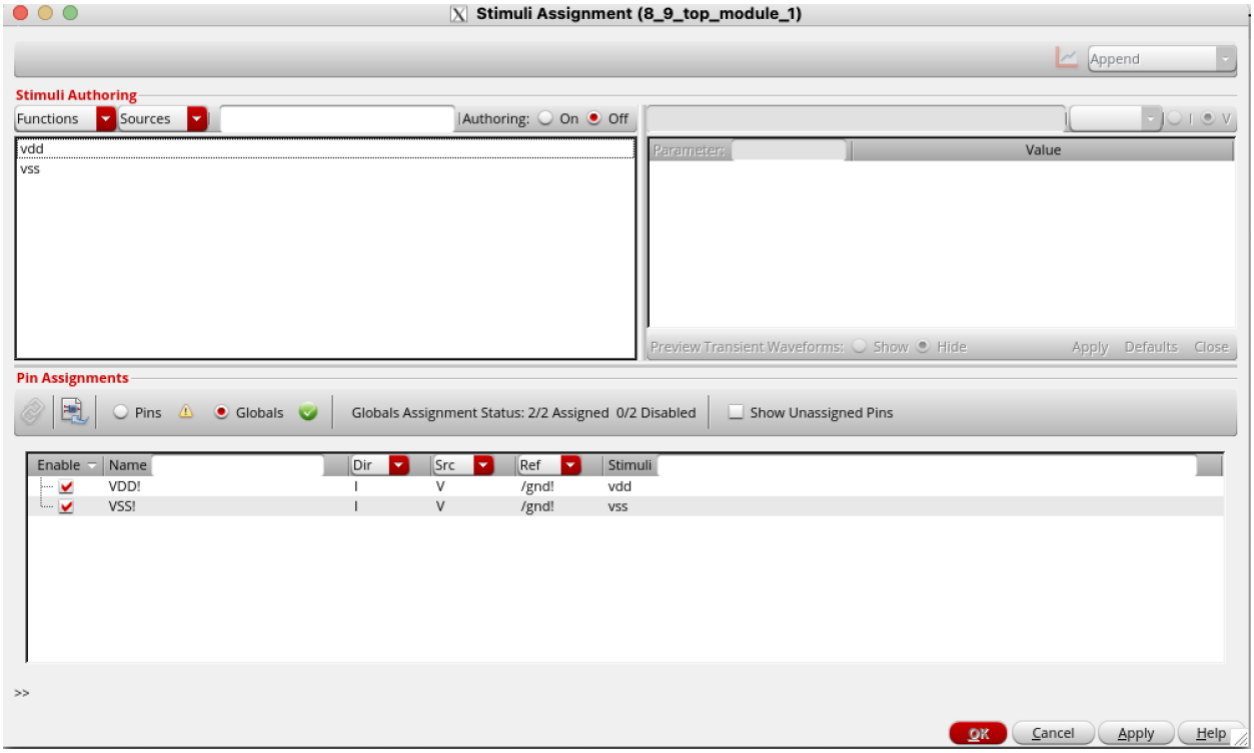
Run ADE explore and change simulator to spectre

Set tran analysis



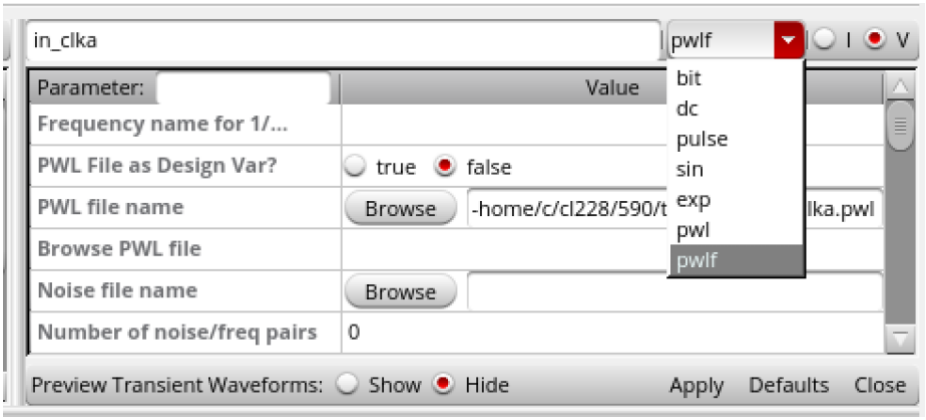
Choose setup->stimuli

Assign global nets VDD! and VSS! to stimuli sources.

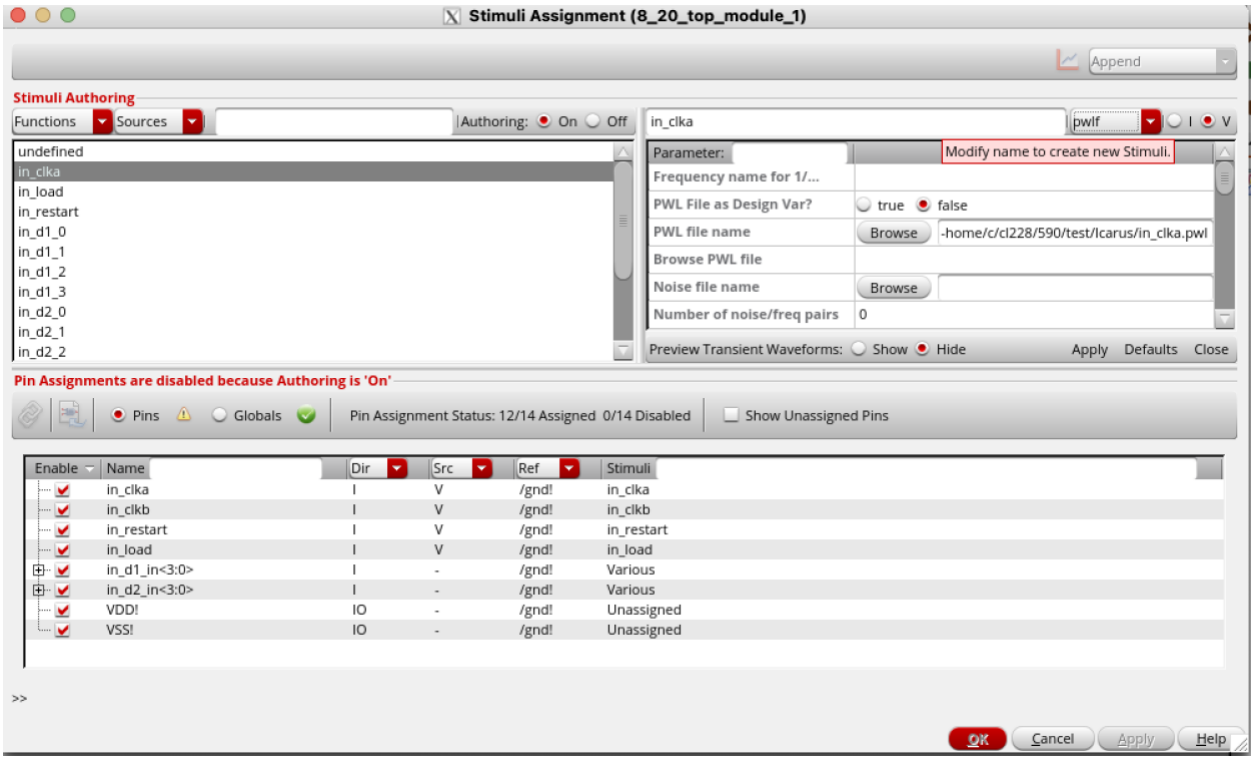


Another simulation way is using pwl files generated from verilog testbench

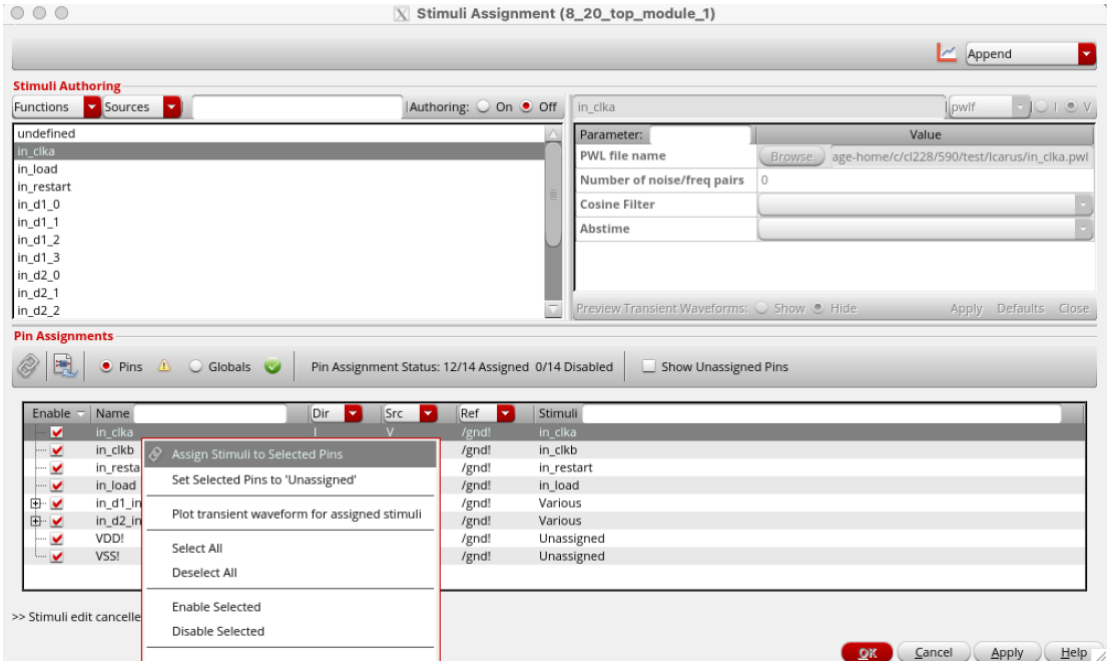
Open stimuli, then change it to pwlf. Set the PWL file name to the pwl file you want to use. Click Apply



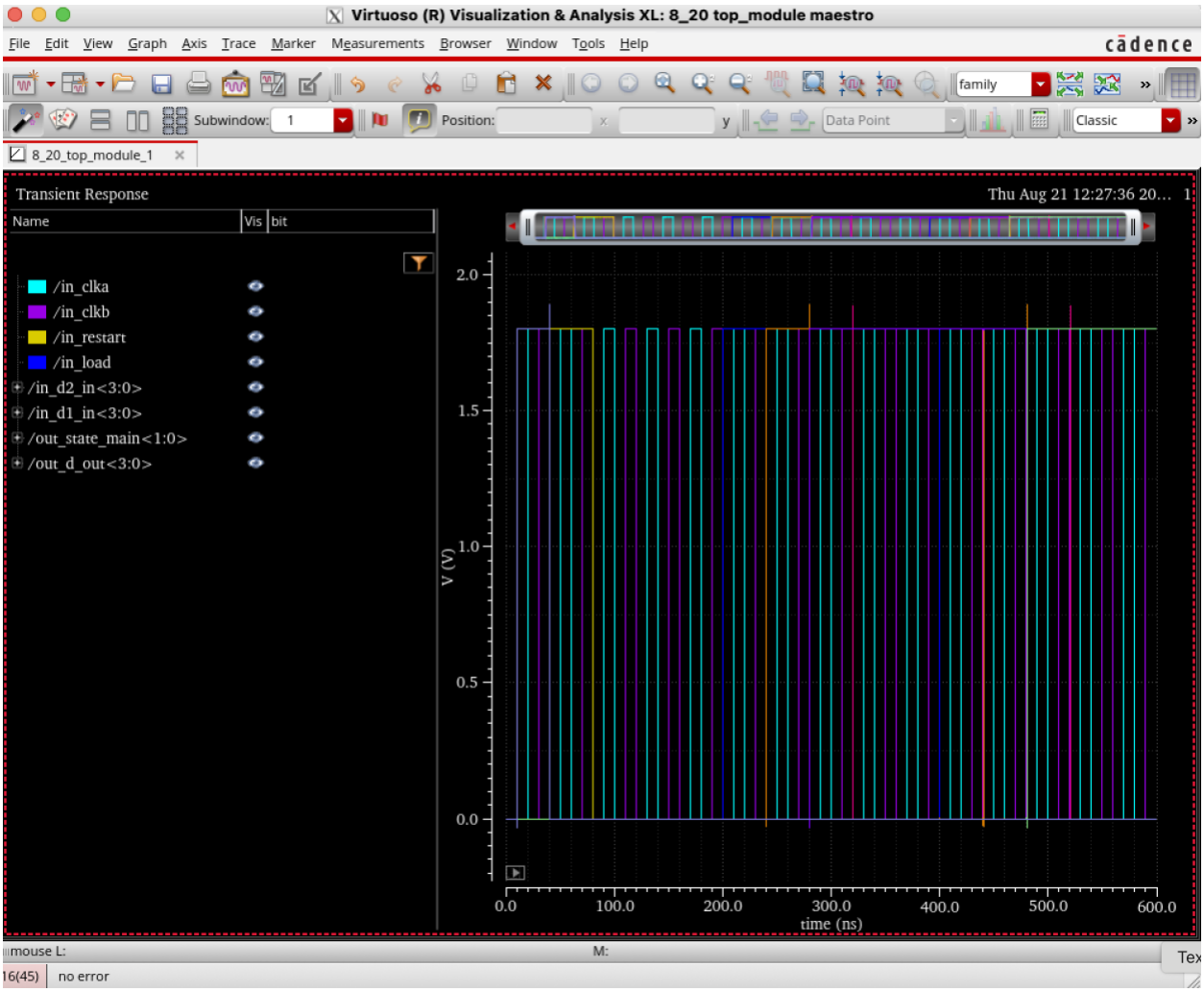
It will show up on left window



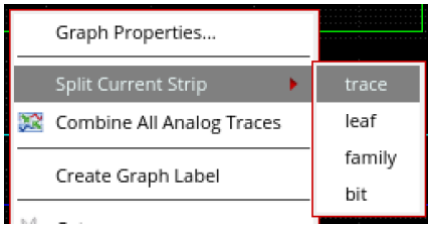
Then change authoring to off, choose signal from stimuli authoring and pin from pin assignments.
Then right click on the chosen pin. Click Assign Stimuli to Selected Pins.
Repeat this step until all of pins were assigned,



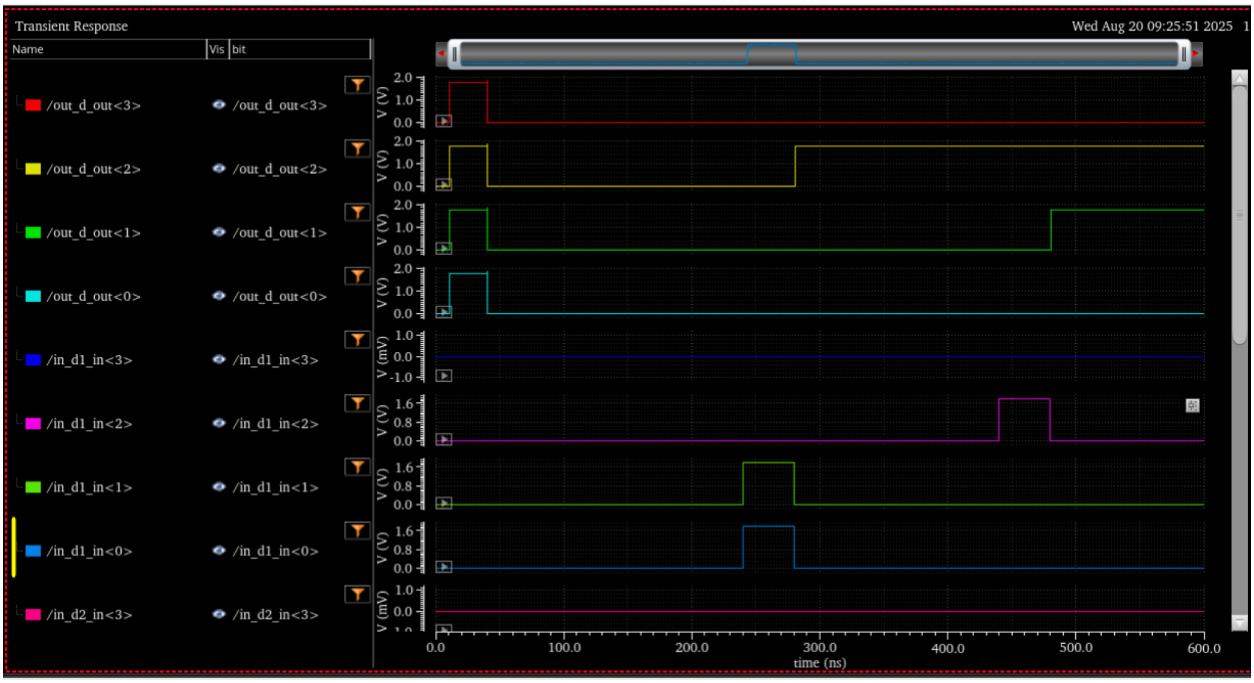
Now you can run simulation
It will show like this automatically.



Right click then choose split current strip



Final result:



Appendix D

Add cds.lib, setup.bash and .cdsinit file (three in total) to the current directory. Download sky130_release_0.0.6 and sky130_scl_9T_0.0.6 and unzip them.

Edit cds.lib, change directory path and define the following four libraries.

```
DEFINE     analogLib    $CDSHOME/tools/dfII/etc/cdslib/artist/analogLib
DEFINE     basic        $CDSHOME/tools/dfII/etc/cdslib/basic
DEFINE sky130_fd_pr_main $PDK_DIR/libs/sky130_fd_pr_main
DEFINE sky130_tt /storage-home/c/cl228/590/sky130_scl_9T_0.0.6/sky130_scl_9T/oa/sky130_scl_9T
```

Then change the directory path in setup.bash. The PEGASUS_DRC has not been used, just set it up following Cadence guidance.

```
# Set up DRC file
export PEGASUS_DRC=/storage-home/c/cl228/590/sky130_release_0.0.6/Sky130_DRC

# Set the PDK_DIR variable to the root directory of the FreePDK distribution
# export PDK_DIR="/clear/courses/elec423/PDK/FreePDK45"
export PDK_DIR="/storage-home/c/cl228/590/sky130_release_0.0.6"
export PEGASUS_DRC="/storage-home/c/cl228/590/sky130_release_0.0.6/Sky130_DRC/Sky130_DRC_rev_0.0_2.2"

# Set CDSHOME to the root directory of the Cadence ICOA installatio
# export CDSHOME="/clear/apps/cadence/IC617"
export CDSHOME="/clear/apps/cadence-2023/IC231"
export OA_HOME='/clear/apps/cadence-2023/IC231/oa_v22.61.006'


# Source global definitions. Check if global default files exists
# if [ -f /etc/bashrc ]; then
#       . /etc/bashrc
# fi


if ! [ -f ${PWD}/.cdsinit ]; then
  cp ${PDK_DIR}/.cdsinit ${PWD}/.cdsinit
fi

if ! [ -f ${PWD}/cds.lib ]; then
  cp ${PDK_DIR}/cds.lib ${PWD}/cds.lib
fi
```

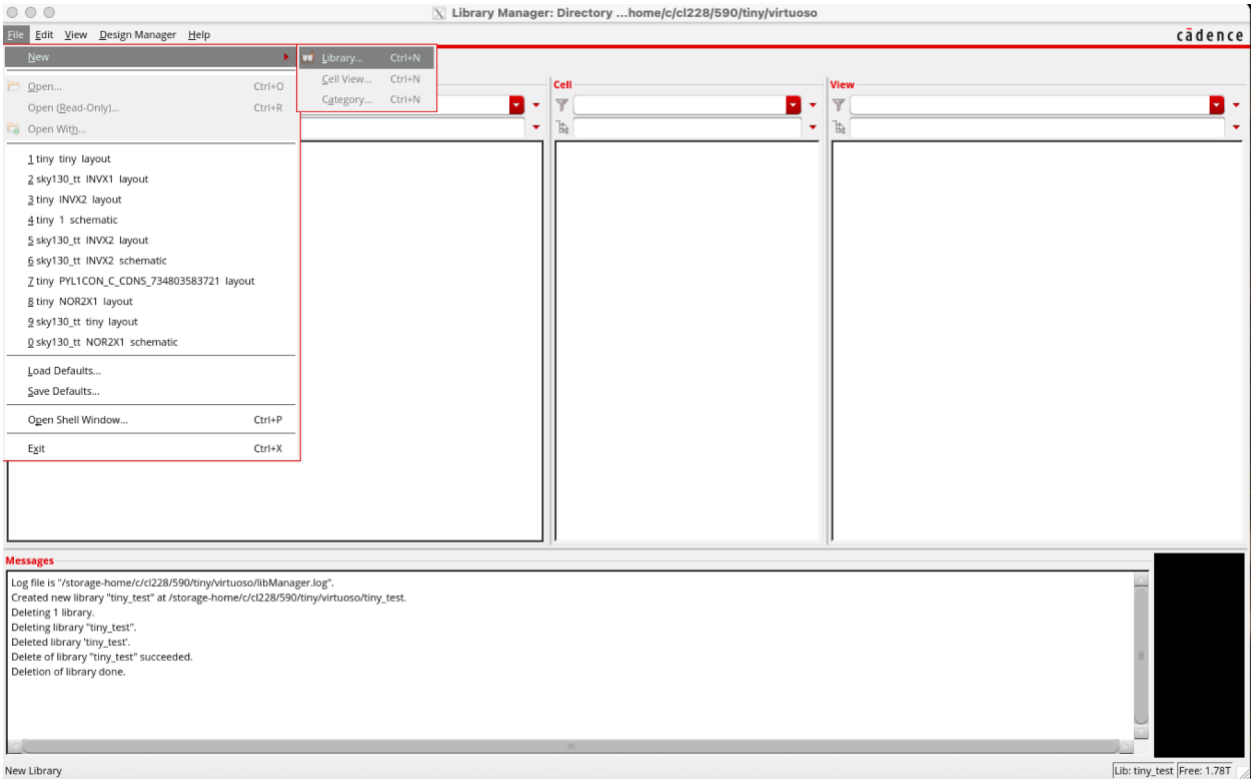
The .cdsinit file should stay the same as it uses the path defined in setup.bash file. But I'm not 100% sure the commands worked.

Clear original .cdsinit and cds.lib file

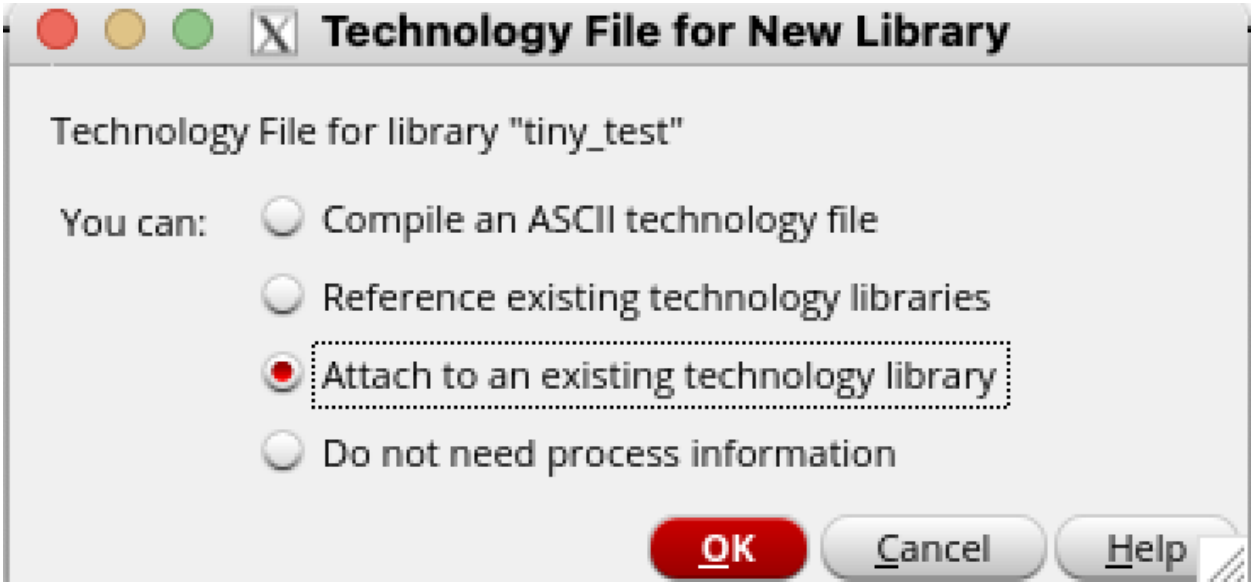
Then run: source setup.bash

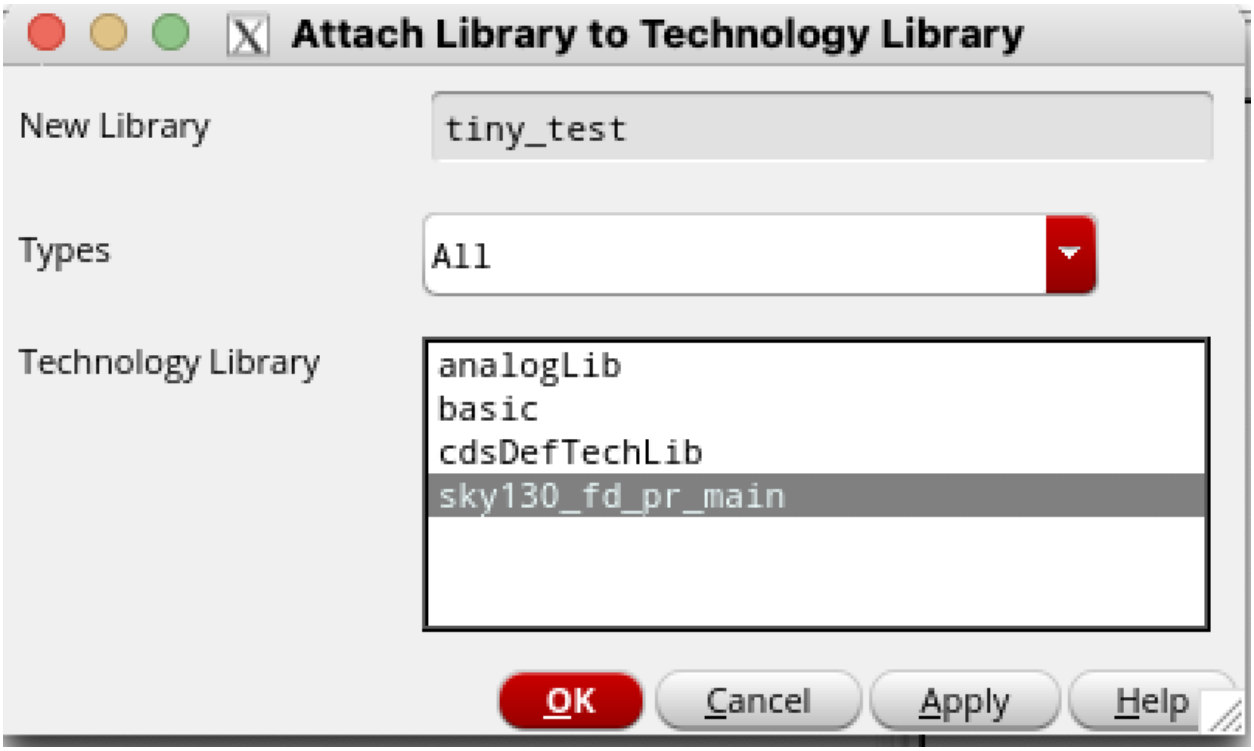
Open virtuoso it should have four libraries: analogLib, basic, sky130_fd_pr_main and sky130_tt. If not, then something is set up wrong.

In the library manager, add a new library to isolate the test file and original library files.

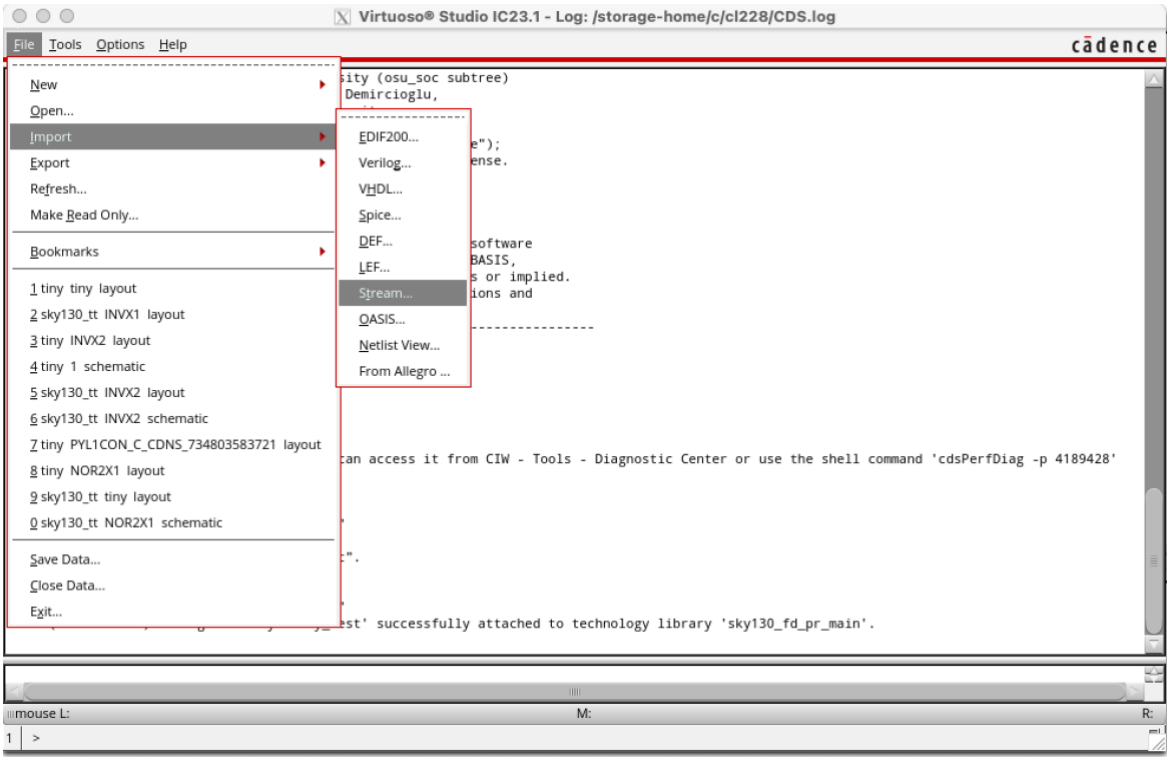


Select attach to an existing technology library

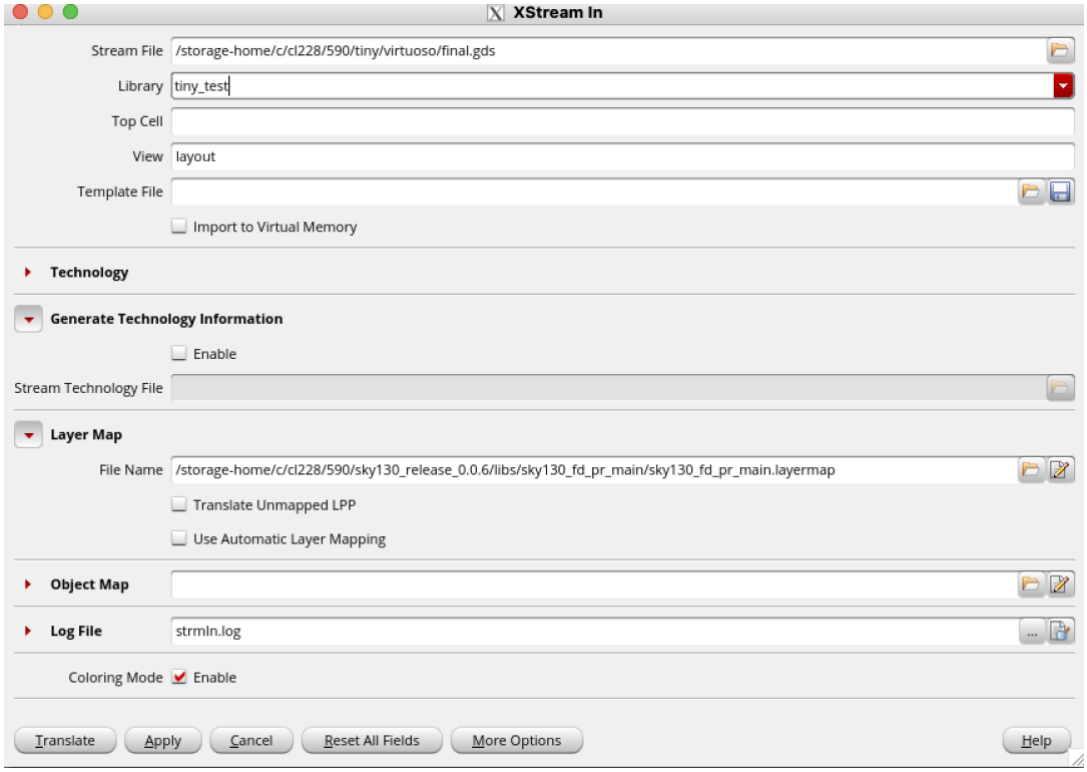




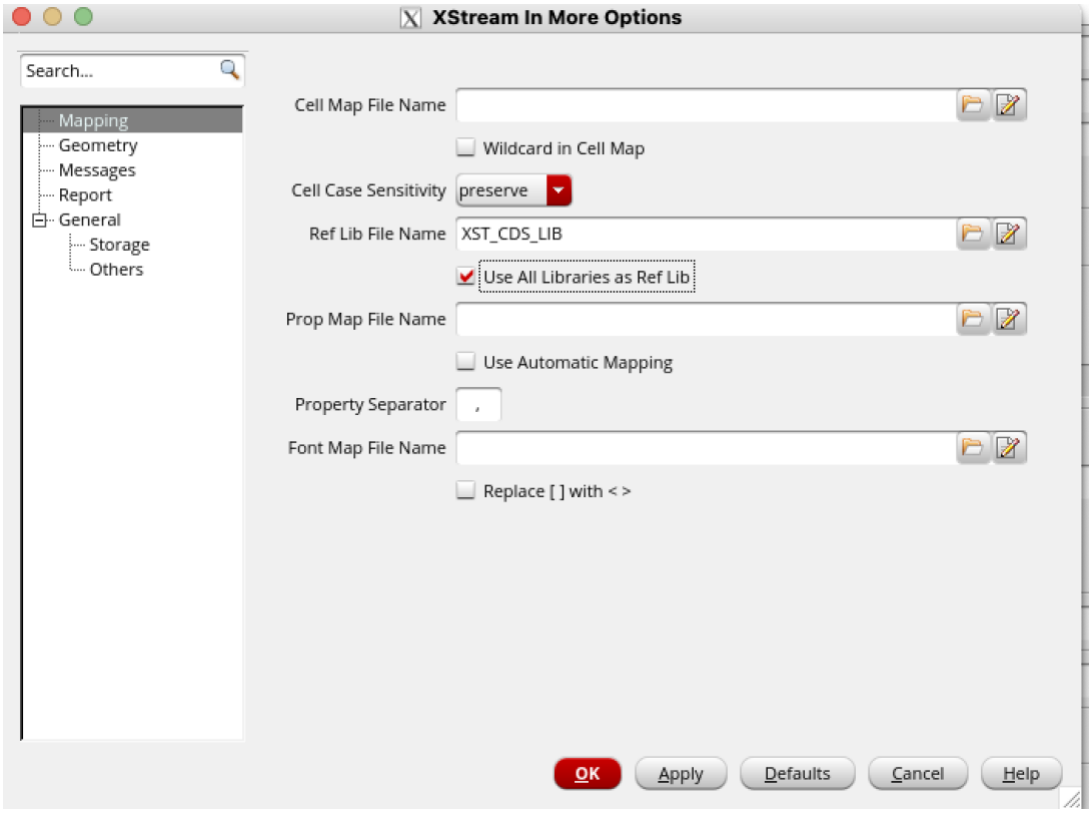
Go to log window, select import->Stream



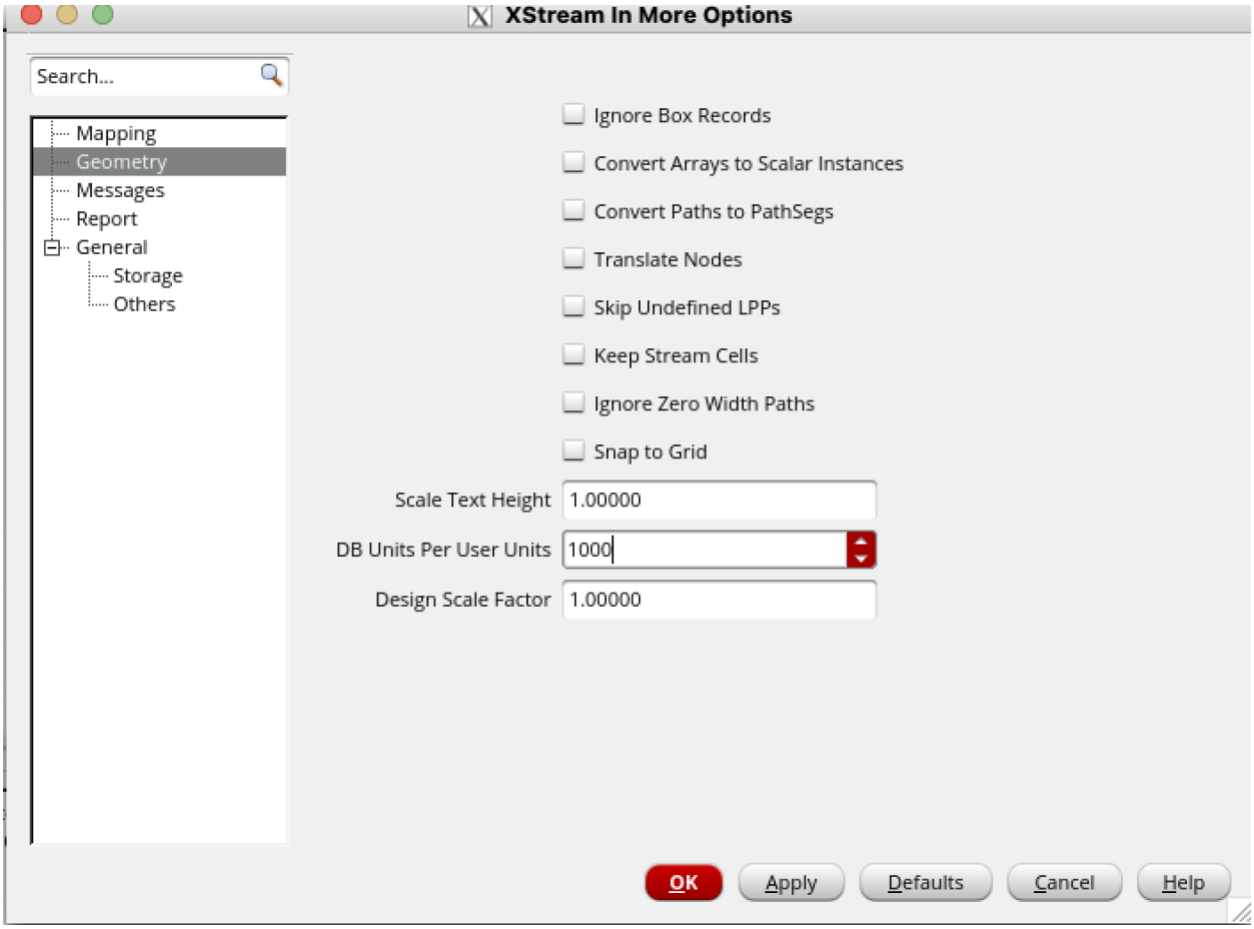
Select Stream file and library. Set up the layer map, which is sky130_release_0.0.6/libs/sky130_fd_pr_main/sky130_fd_pr_main.layermap



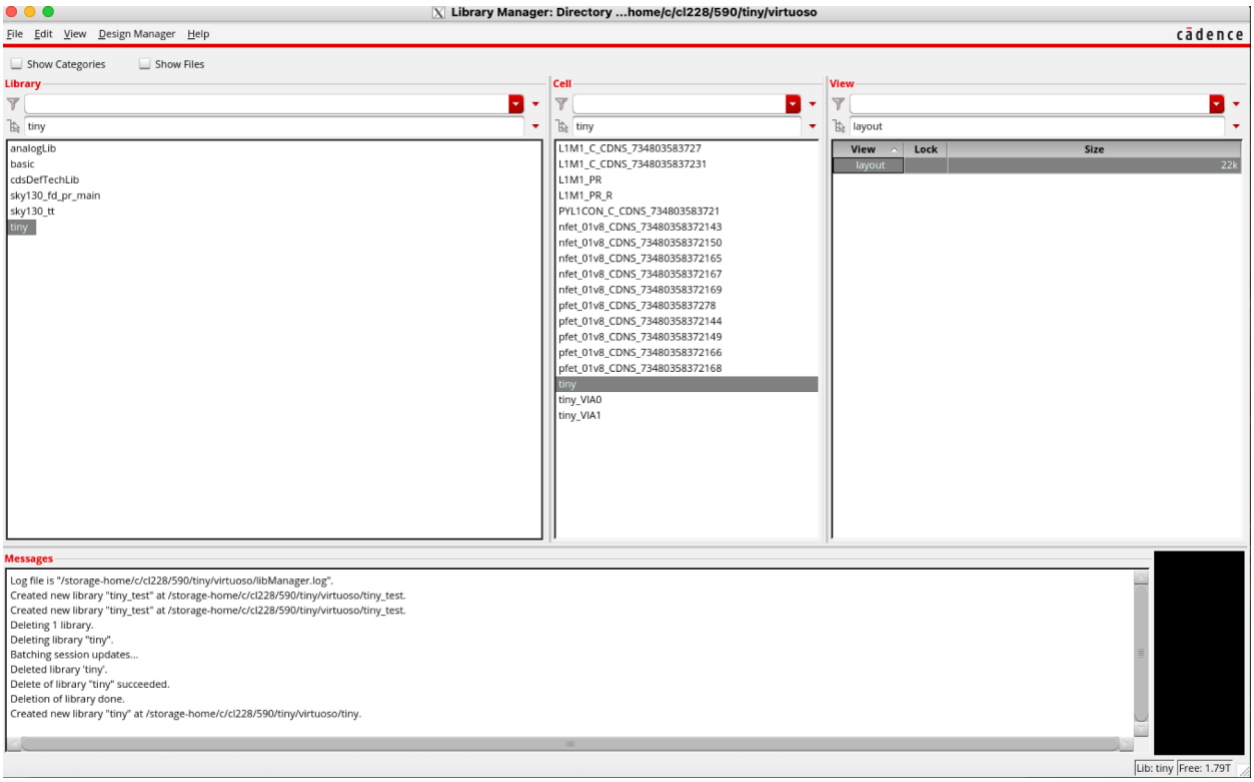
Click More Options, then check Use All Libraries as Ref Lib. Click ok then click translate



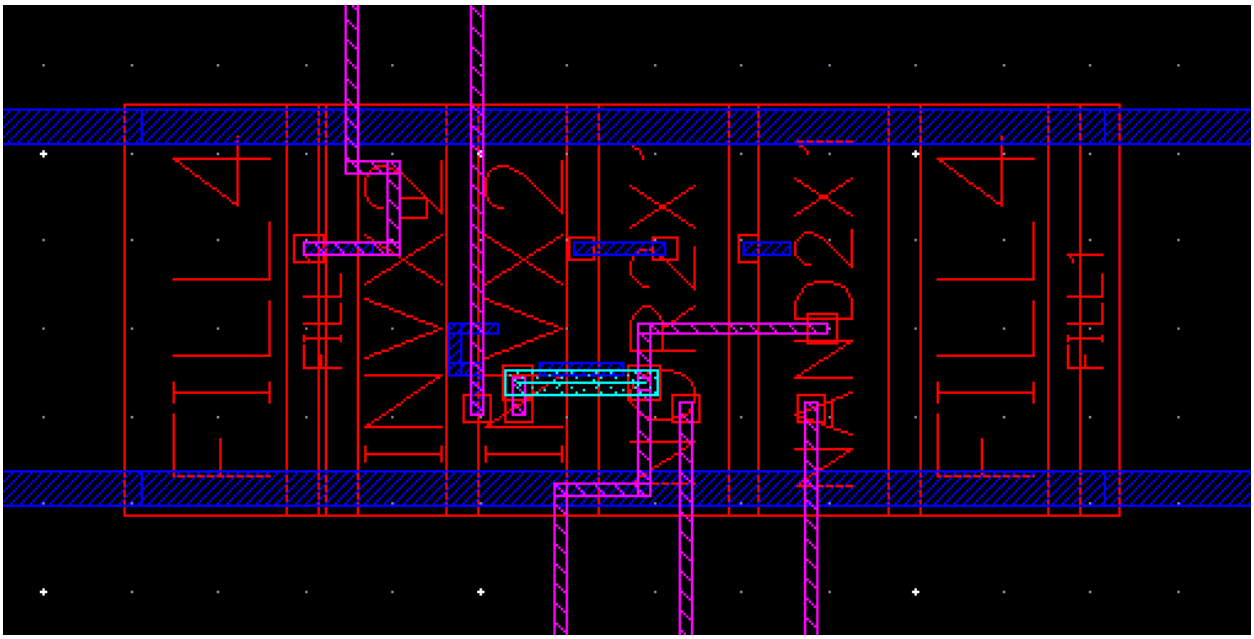
Set DB unit to 1000



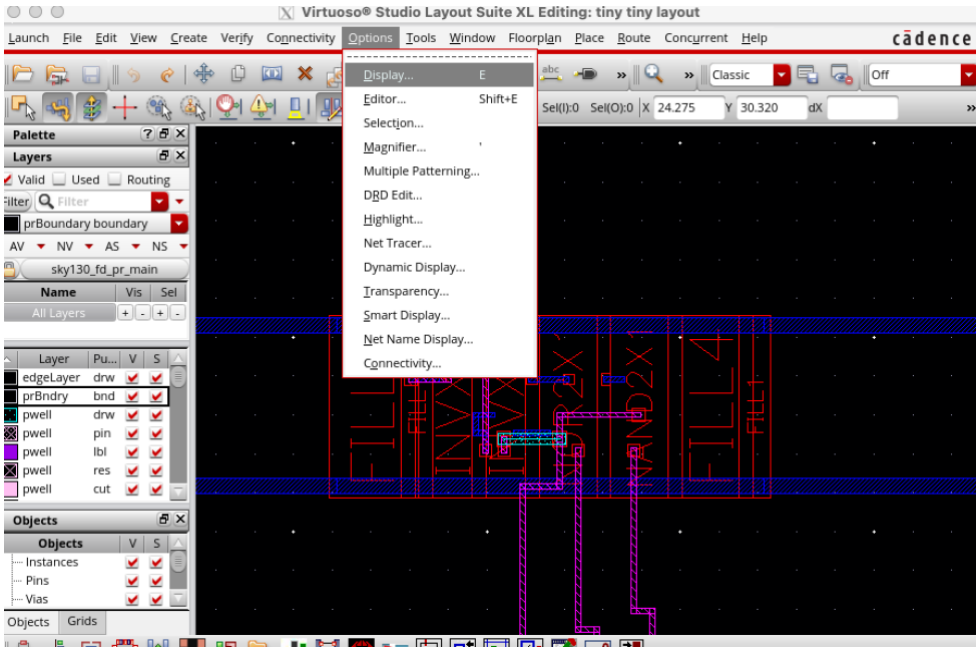
The library manager should look like this



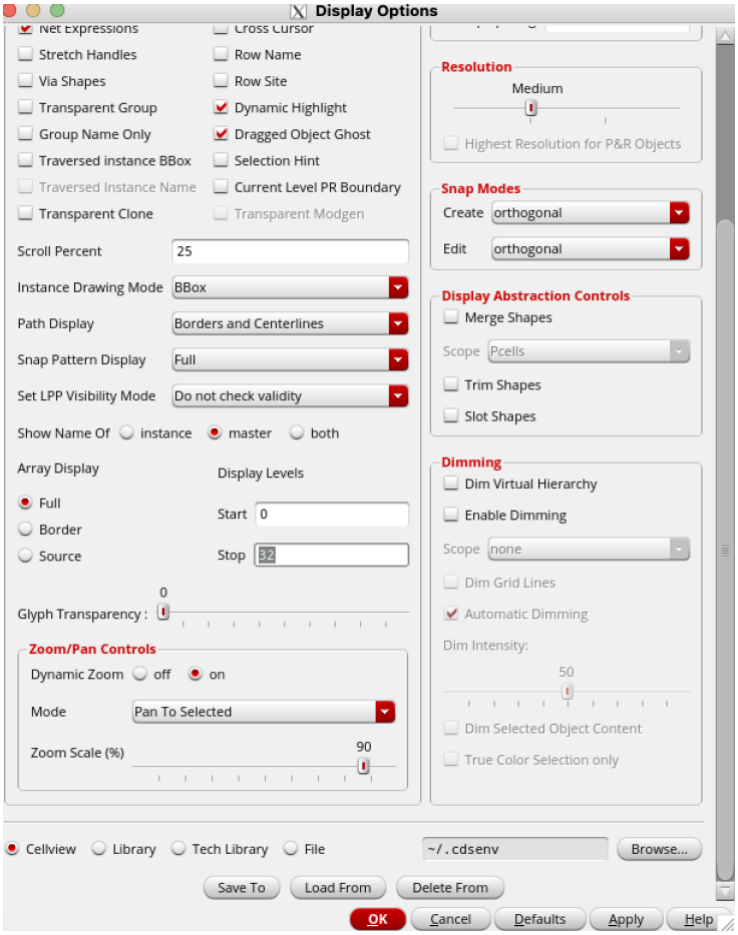
Open tiny layout, which only shows top view



Click Options->Display



Change Stop Display levels from 0 to 32



Full layers show up