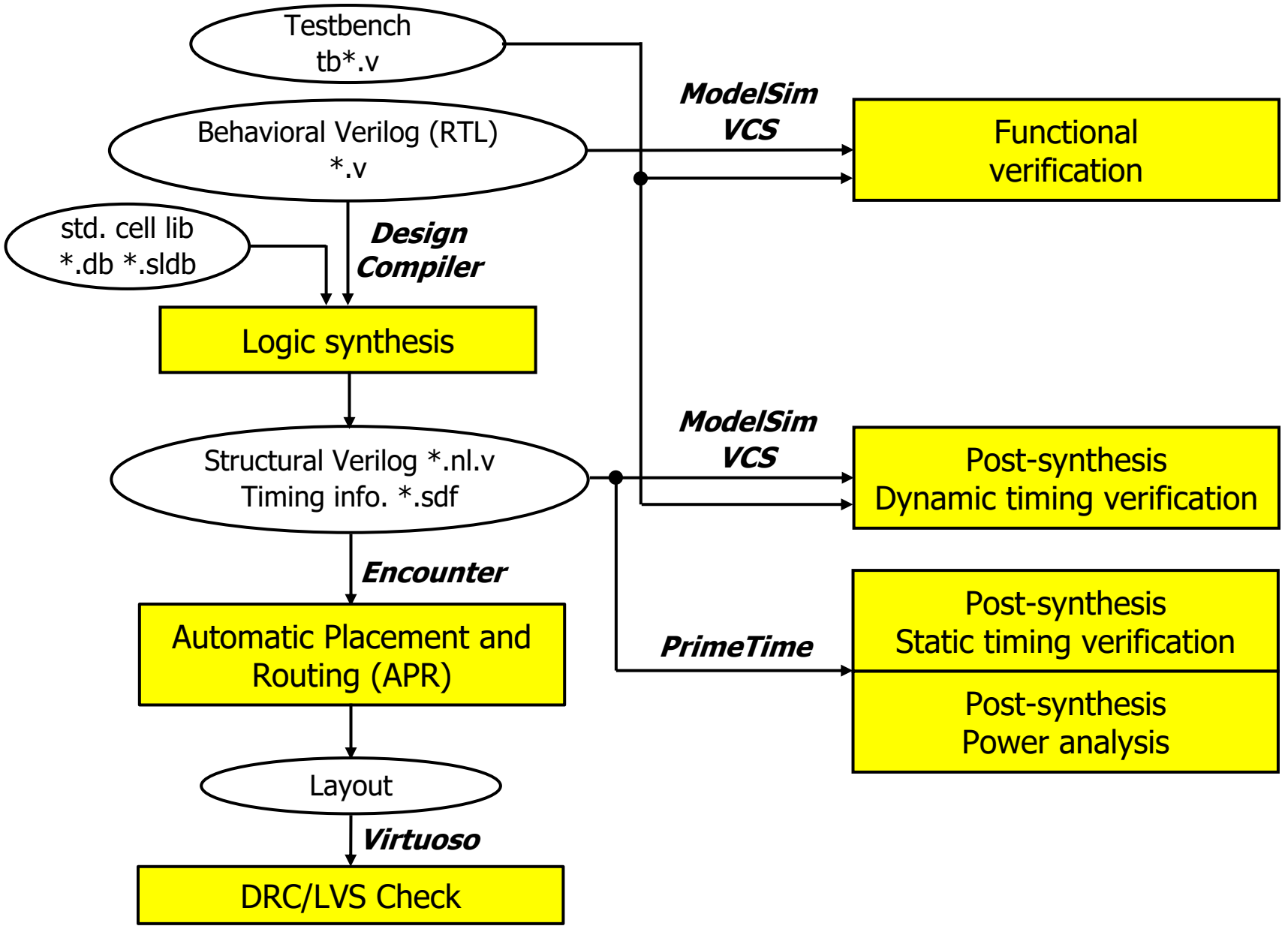
EECS E6321 Tutorial 04

Block-level APR and SRAM Compiler

Mingoo Seok & Previous TAs

Semi-Custom Design Flow: Block-level



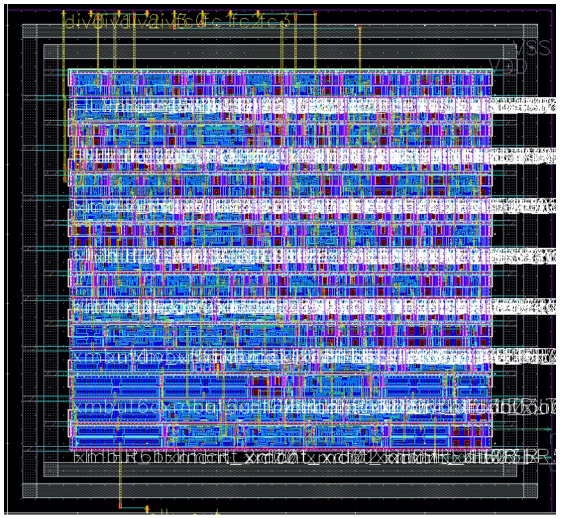
What is APR?

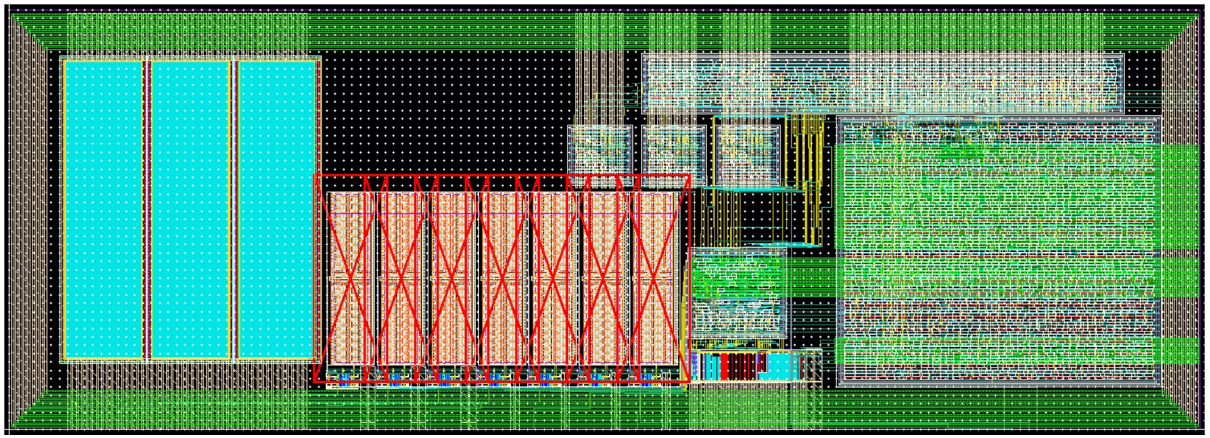
Auto placement & routing

Based on physical constraints (e.g., area, metal blockage), the tool places instances (modules or standard cells) and connects them using wires and vias

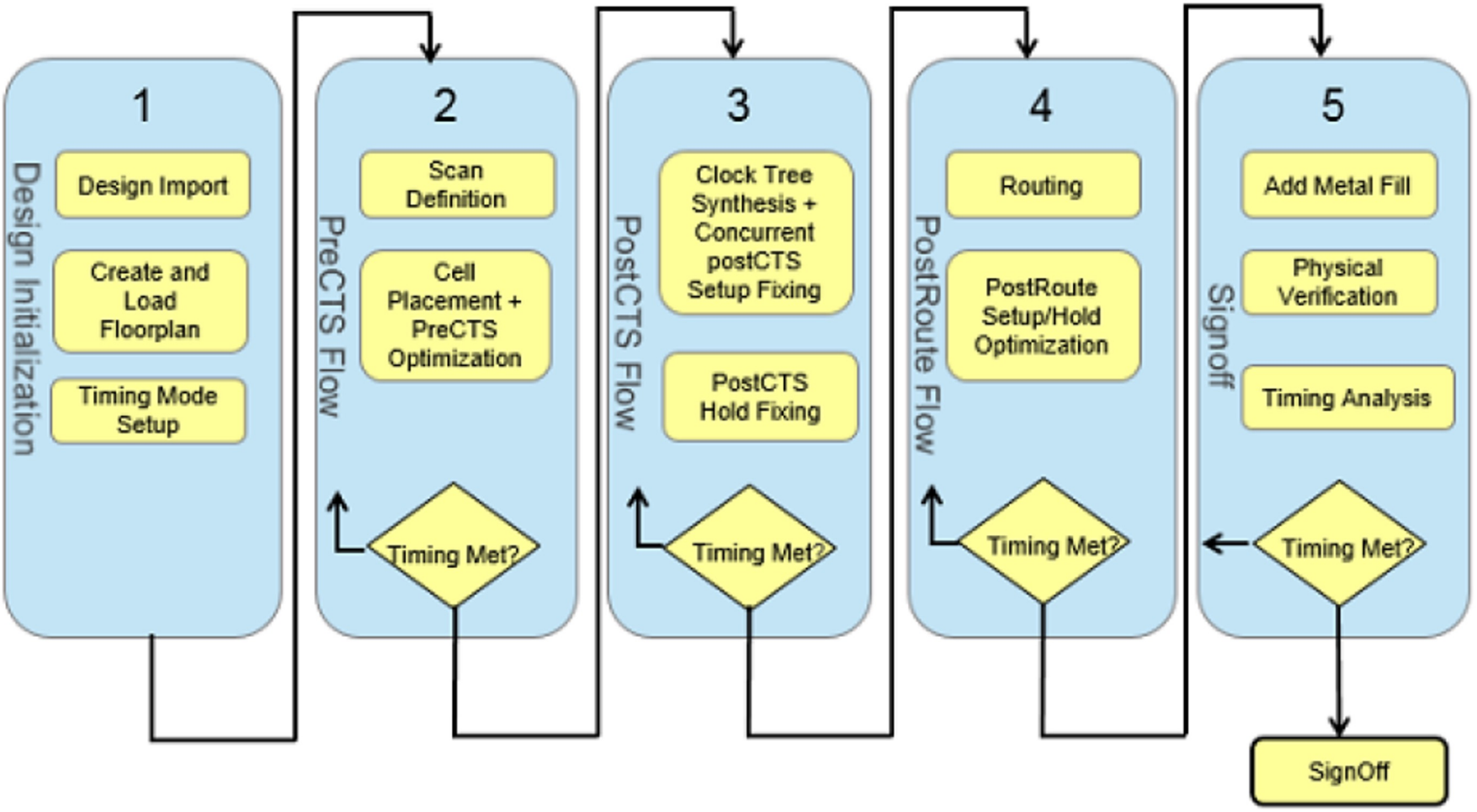
We use APR when

1. We implement the digital module, which is made of standard cells after synthesis
2. We want to connect all modules, including std cell-based and full custom, on the higher level (You will need abstract of all modules (.lef))





Design Implementation Flow



Cadence Innovus

- Cadence Innovus provides an integrated solution for an RTL-to-GDSII design flow
  - The previous Cadence APR tool was called Encounter, but they make a new platform called Innovus
  - We will use Innovus

- Some Encounter commands don't work in Innovus

- You can find the Innovus manuals at
  - /tools/cadence/INNOVUS191/doc/innovus*…

To Start with…

- Copy files from
  - /courses/ee6350/proj_2025Spring/ref/innovus/
    - down_counter: example
  - /courses/ee6350/proj_2025Spring/ref/qsim_apr
    - down_counter: example

Important files

- down_counter.tcl
  - Includes general settings and commands for the Innovus design flow
- config.globals
  - Input configuration file
  - Sets the design, I/O file, and .lef file
- mmmc.view
  - Multi-Mode Multi-Corner (MMMC) setup file
  - It will be used to implement the design that can operate across different conditions
  - We will consider only one condition: TTTT
  - Typical NMOS, typical PMOS, typical temperature, and typical voltage
- down_counter.io
  - An I/O assignment file
- makecdl.sh
  - A script converting the .v file (Verilog) to the .cdl file

How to Run Innovus

- Command: $innovus
- In this tutorial, it is better to run the Innovus in interactive mode using the GUI to actually see what each line of the script is doing
  - Start Innovus in a GUI
  - Open up the .tcl file in another terminal
  - Copy & paste each line of code into the encounter terminal
- If you are not sure about a command/script line, recommend to use "**man**" commands
- Once you become familiar, use the Innovus in the shell script mode as usual
  - Please see the Makefile or run_apr.sh

down_counter.tcl

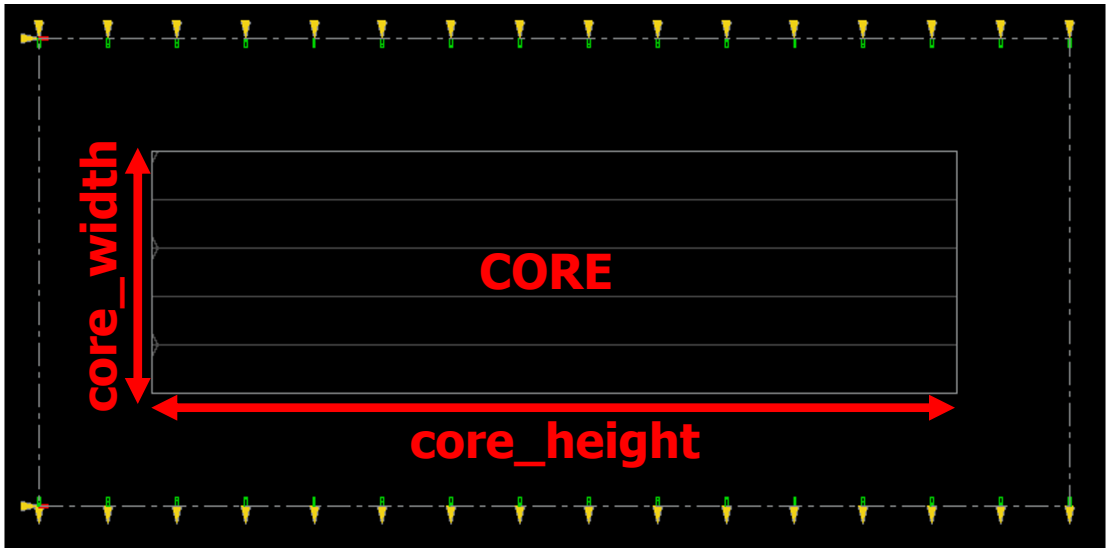
- Step 1) Set up design and set parameters for the floorplan

```
#####################################################
# Setup design
#####################################################

set design_name down_counter
source ./config.globals
set_message -no_limit
set_message -id {TECHLIB-1467} -limit 10
setMultiCpuUsage -localCpu max -acquireLicense 8

# Initialize a design using the Tcl globals listed in the Related Global section
# Notes: At this script, all the settings are included in the file 'config.globals'
init_design

#####################################################
# Set parameters for the floorplan
#####################################################

# Need modification
set std_cell_height 1.8
# core_height should be a multiple of the std_cell_height
set core_width 20
set core_height [expr 10*$std_cell_height]
set ring_left_width 1
set ring_right_width 1
set ring_top_width 1
set ring_bottom_width 1
set ring_left_space 0.5
set ring_right_space 0.5
set ring_top_space 0.5
set ring_bottom_space 0.5
```

down_counter.tcl

- Step 2) Create the floorplan

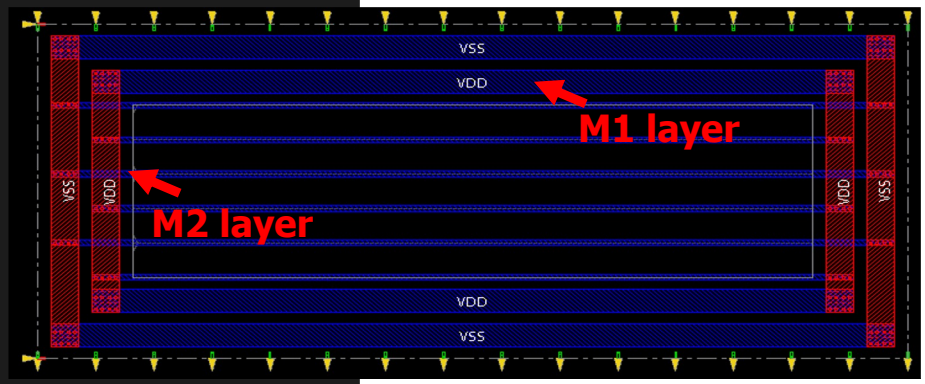
```
###################################################
# Create the floorplan
###################################################

# Specify the floorplan dimensions by size; or by die, I/O, or core coordinates (Need modification)
floorPlan -s $core_width $core_height [expr 2*$ring_left_width + $ring_left_space + 2] \
[expr 2*$ring_bottom_width + $ring_bottom_space +2] [expr 2*$ring_right_width + $ring_right_space +2] \
[expr 2*$ring_top_width + $ring_top_space + 2]

# GUI only
redraw
fit
```



down_counter.tcl

- Step 3) Generate power rails

```
################################################
# Generate power rails
################################################

# Adds a new global net connection to the specified global net
proc connect_std_cells_to_power { } {

    # Connect the tiehi, tiolo nets in instances to the global nets 'VDD' & 'VSS'
    globalNetConnect VDD -type tiehi -inst * -verbose
    globalNetConnect VSS -type tielo -inst * -verbose

    # Connect the power pins in instances to the global nets 'VDD' & 'VSS'
    globalNetConnect VDD -type pgpin -pin VDD -inst * -verbose
    globalNetConnect VSS -type pgpin -pin VSS -inst * -verbose
}

Puts "####################"
Puts "###"
Puts "### Power Routing ..."
Puts "###"
Puts "####################"

connect_std_cells_to_power
# Apply and restore the global net connectivity rules to the the design
# and creates the necessary connecktions between instances and these global nets
applyGlobalNets


# Create rings for specified nets round the core boundary or selected blocks and groups of core rows
# Notes: This command should be used after creating an initial floorplan
#        Need to use different metal layers for horizontal and vertical lines (e.g. even: vertical, odd: horizontal)
addRing -nets {VDD VSS} -type core_rings -layer {top M5 bottom M5 left M6 right M6} -width $ring_top_width \
        -spacing $ring_top_space -center 1 \
            -extend_corner {lt lb rt rb}

# GUI only
redraw
fit

# Add vertical power rails (Need modification)
addStripe -block_ring_top_layer_limit M6 \
          -block_ring_bottom_layer_limit M6 \
          -padcore_ring_top_layer_limit M6 \
          -padcore_ring_bottom_layer_limit M6 \
          -max_same_layer_jog_length 6 \
          -merge_stripes_value 4 \
          -layer M6 \
          -set_to_set_distance 8 \
          -direction vertical \
          -nets {VDD VSS} \
          -width 1 \
          -spacing 0.5 -area {8 0 16 100}

# Route power nets
sroute -nets {VDD VSS} -allowJogging 0 -allowLayerChange 0
```



down_counter.tcl

- Step 4) Set all the input/output ports

```
################################################
# Set all the input/output ports
################################################

# Load an I/O assignment file
loadIoFile "./$design_name.io"

redraw

# Save the flooprlan information to a file (intermediate saving)
saveDesign "$design_name.floorplan.enc"
```

down_counter.io

```
Offset: 5
Spacing: 1.5
Pin: clk W 3 0.1800 0.6000
Pin: rstn W 3 0.1800 0.6000
Pin: start W 3 0.1800 0.6000
Pin: reset W 3 0.1800 0.6000

Offset: 5
Spacing: 1.5
Pin: counter_out[0] E 3 0.1800 0.6000
Pin: counter_out[1] E 3 0.1800 0.6000
Pin: counter_out[2] E 3 0.1800 0.6000
Pin: counter_out[3] E 3 0.1800 0.6000
Pin: counter_out[4] E 3 0.1800 0.6000
Pin: counter_out[5] E 3 0.1800 0.6000
Pin: counter_out[6] E 3 0.1800 0.6000
Pin: counter_out[7] E 3 0.1800 0.6000
```

- I/O file format:
  - Pin: net_name location (NESW) metal# pin_width pin_length
  - Use the Offset & Spacing commands to place the pins at the locations you want (Origin is at the bottom-left corner)
  - You can automate making an io file using Python

down_counter.tcl

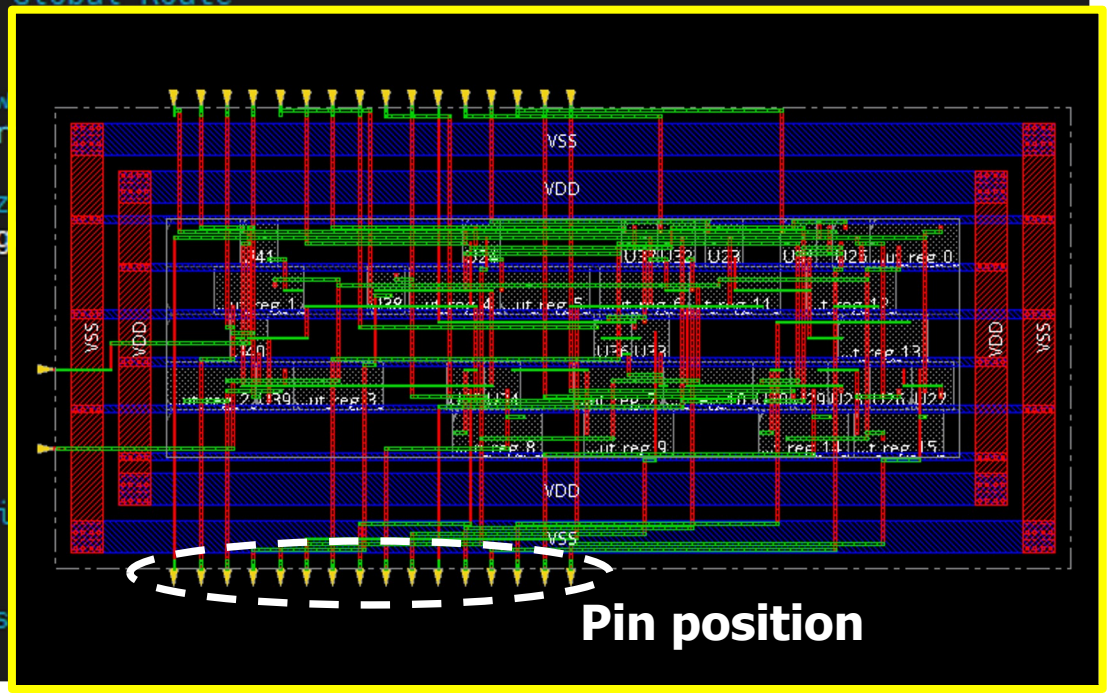
- Step 5) Place design
  - set(Design/Route/Place)Mode → place_design

```
Puts "####################"
Puts "###"
Puts "### Place Design ..."
Puts "###"
Puts "####################"

# Defines the process technology value
setDesignMode -process 65 -flowEffort standard
# Specify a routing layer limit for Early Global Route
setMaxRouteLayer 5

# Control certain aspects of how the software places cells
setPlaceMode -timingDriven true -congEffort high

# Set global parameters for timing optimization
setOptMode -fixFanoutLoad true -effort high -moveInst true -reclaimArea true

# Place stanard cells based on the global settings for placement, RC extraction, and timing analysis
place_design -noPrePlaceOpt
setDrawView place
connect_std_cells_to_power
redraw

# Check FIXED and PLACED cells for violations, add violation markers to the design display area
checkPlace

# Build the static timing model of the design
buildTimingGraph
```

down_counter.tcl

- Step 5) Place design
  - set(Design/Route/Place)Mode → place_design

```
Puts "####################"
Puts "###"
Puts "### Place Design ..."
Puts "###"
Puts "####################"

# Defines the process technology value
setDesignMode -process 65 -flowEffort standard
# Specify a routing layer limit for Early Global Route
setMaxRouteLayer 5

# Control certain aspects of how the softw
setPlaceMode -timingDriven true -congEffor

# Set global parameters for timing optimiz
setOptMode -fixFanoutLoad true -effort hig

# Place stanard cells based on the global
place_design -noPrePlaceOpt
setDrawView place
connect_std_cells_to_power
redraw

# Check FIXED and PLACED cells for violati
checkPlace

# Build the static timing model of the des
buildTimingGraph
```



down_counter.tcl

- Step 6) Do preCTS optimization

```
################################################
# Do preCTS optimization
################################################

Puts "############################"
Puts "###"
Puts "### PreCTS Optimization ..."
Puts "###"
Puts "############################"

#set_dont_touch clkloc_inst
setOptMode -yieldEffort none ; # Default is none
setOptMode -effort high ; # man setOptMode and see the effort table
setOptMode -maxDensity 0.95 ; # Default is 0.95 (netlist won't grow above this value)
setOptMode -drcMargin 0.0 ; # Default is 0
#setOptMode -holdTargetSlack 0.0
setOptMode -holdTargetSlack 0.2
setOptMode -setupTargetSlack 0.0
setOptMode -SimplifyNetlist false ; # When true, simplifies the netlist to decrease congestion, area, and improve runtime
clearClockDomains
setOptMode -usefulSkew false


# Perform timing optimization before or after the clock tree is built, or after routing and generate timing reports
# Notes: '-preCTS' option performs timing optimization on the placed design, before the clock tree is built
 optDesign -preCTS
connect_std_cells_to_power
applyGlobalNets
# Save design
redraw
saveDesign "$design_name.placed.enc"
```

down_counter.tcl

```
################################################
### Do clock tree synthesis
################################################

Puts "############################"
Puts "###"
Puts "### Clock Tree Synthesis ..."
Puts "###"
Puts "############################"

# Set global analysis modes for timing analysis
# Notes: '-cppr' option removes pessimism from clock paths
#        '-analysisType bcwc' option checks the design for two extreme conditions
#setAnalysisMode -cppr both
setAnalysisMode -analysisType bcwc


update_constraint_mode -name typical_constraint -sdc_file "../../dc/$design_name/$design_name.syn.mod.sdc"
# Control certain aspects of how the NanoRoute router routes the design (uncomment if necessary)
# setNanoRouteMode -quiet -routeTopRoutingLayer 3

#Declare the analysis views to be used during ccopt_design
set_analysis_view -setup {typical} -hold {typical}
# Define route types
create_route_type -name leaf_rule -top_preferred_layer M5 -bottom_preferred_layer M4
create_route_type -name trunk_rule -top_preferred_layer M5 -bottom_preferred_layer M4
create_route_type -name top_rule -top_preferred_layer M5 -bottom_preferred_layer M4
# Specify route types
set_ccopt_property -net_type leaf route_type leaf_rule
set_ccopt_property -net_type trunk route_type trunk_rule
set_ccopt_property -net_type top route_type top_rule
set_ccopt_property routing_top_min_fanout 10000
#
# Specify the buffer and inverter cells for CTS
set_ccopt_property inverter_cells [list CKND1 CKND12 CKND16 CKND2 CKND20 CKND24 CKND3 CKND4 CKND6 CKND8]
set_ccopt_property buffer_cells [list CKBD1 CKBD12 CKBD16 CKBD2 CKBD20 CKBD24 CKBD3 CKBD4 CKBD6 CKBD8]

#use inverteres instead of buffers
set_ccopt_property use_inverters true

# Specify the target skew for clock tree balancing (Change skew !!!!!!!!)
#set_ccopt_property target_max_trans 35ps
set_ccopt_property target_skew 20ps

# Create a clock tree network with associated skew groups and other CTS configuration settings
create_ccopt_clock_tree_spec -file ccopt_clock_tree.spec
source "./ccopt_clock_tree.spec"

# Perform clock concurrent optimization (CCOpt) on the current loaded design in Innovus
# CCOpt optimizes both the clock tree and the datapath to meet global timing constraints
ccopt_design -cts

# Generate skew and clock tree report
report_ccopt_skew_groups -file "$design_name.ccopt.skew.rpt"
report_ccopt_clock_trees -file "$design_name.ccopt.clk_tree.rpt"

# Save design
connect_std_cells_to_power
redraw
saveDesign "$design_name.clock.enc"
savePlace "$design_name.place"
```

- Step 7) Do clock tree synthesis (CTS)
  - set_ccopt_property → ccopt_design

down_counter.tcl

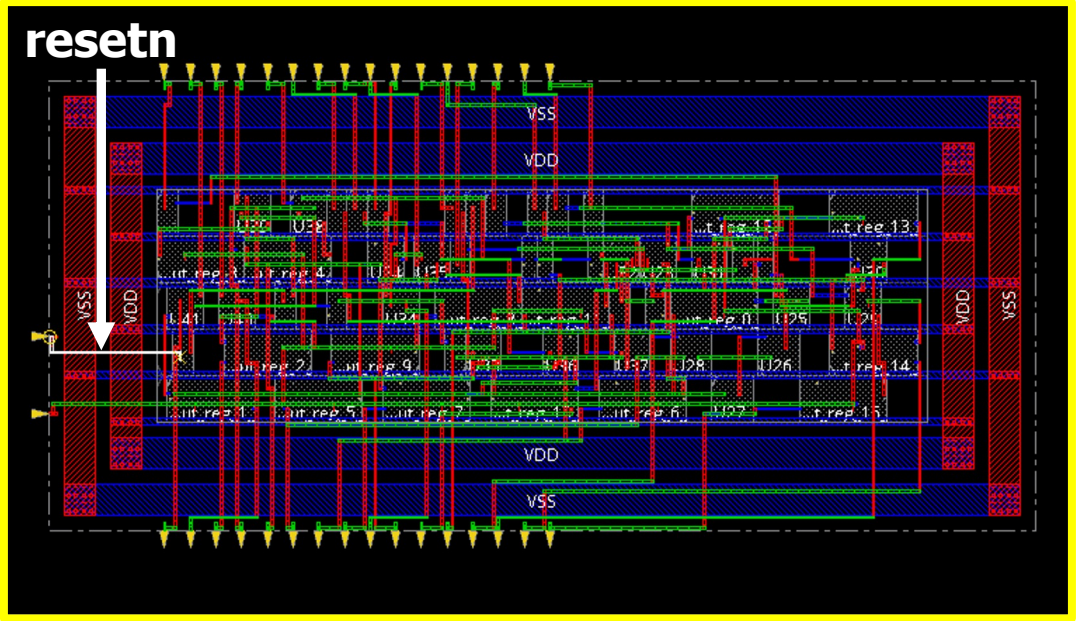
- Step 8) Route signals
  - Route the critical signal first
  - setNanoRouteMode → globalDetailRoute
  - setNanoRouteMode –quiet –routeSelectedNetOnly **true**/false

```
################################################
# Route signals
################################################

# Unfix the clock nets to avoid routing problems
changeUseClockNetStatus -noFixedNetWires

############### Route resetn first ###############
# Attach attributes to nets / Attaching the attributes alloows the NanoRoute routing commands
setAttribute -net resetn -weight 5 -avoid_detour true -bottom_preferred_routing_layer 2 \
             -top_preferred_routing_layer 3 -preferred_extra_space 2

# Select a net and highlight it in the design display window
selectNet resetn

# Control certain aspects of how the NanoRoute router routes the design
setNanoRouteMode -quiet -routeWithTimingDriven true
setNanoRouteMode -quiet -routeSelectedNetOnly true
setNanoRouteMode -quiet -routeTopRoutingLayer 3
setNanoRouteMode -quiet -routeBottomRoutingLayer 1

# Use the NanoRoute router to perform both global and detailed routing with one command
globalDetailRoute

redraw
################################################
```

down_counter.tcl

- Step 8) Route signals
  - Route all other nets

```
Puts "############################"
Puts "###"
Puts "### Route Other Signals ..."
Puts "###"
Puts "############################"

############## Route all other nets ##############
setNanoRouteMode -quiet -routeSelectedNetOnly false
setNanoRouteMode -quiet -routeWithTimingDriven true
setNanoRouteMode -quiet -routeTdrEffort 10
setNanoRouteMode -quiet -drouteFixAntenna true
setNanoRouteMode -quiet -routeWithSiDriven true
setNanoRouteMode -quiet -routeSiLengthLimit 200
setNanoRouteMode -quiet -routeSiEffort high
setNanoRouteMode -quiet -routeWithViaInPin false
setNanoRouteMode -quiet -routeWithViaOnlyForStandardCellPin false
setNanoRouteMode -quiet -droutePostRouteSwapVia none
setNanoRouteMode -quiet -drouteUseMultiCutViaEffort low
setNanoRouteMode -routeTopRoutingLayer 5
setNanoRouteMode -routeBottomRoutingLayer 1
setNanoRouteMode -drouteElapsedTimeLimit 0
```



down_counter.tcl

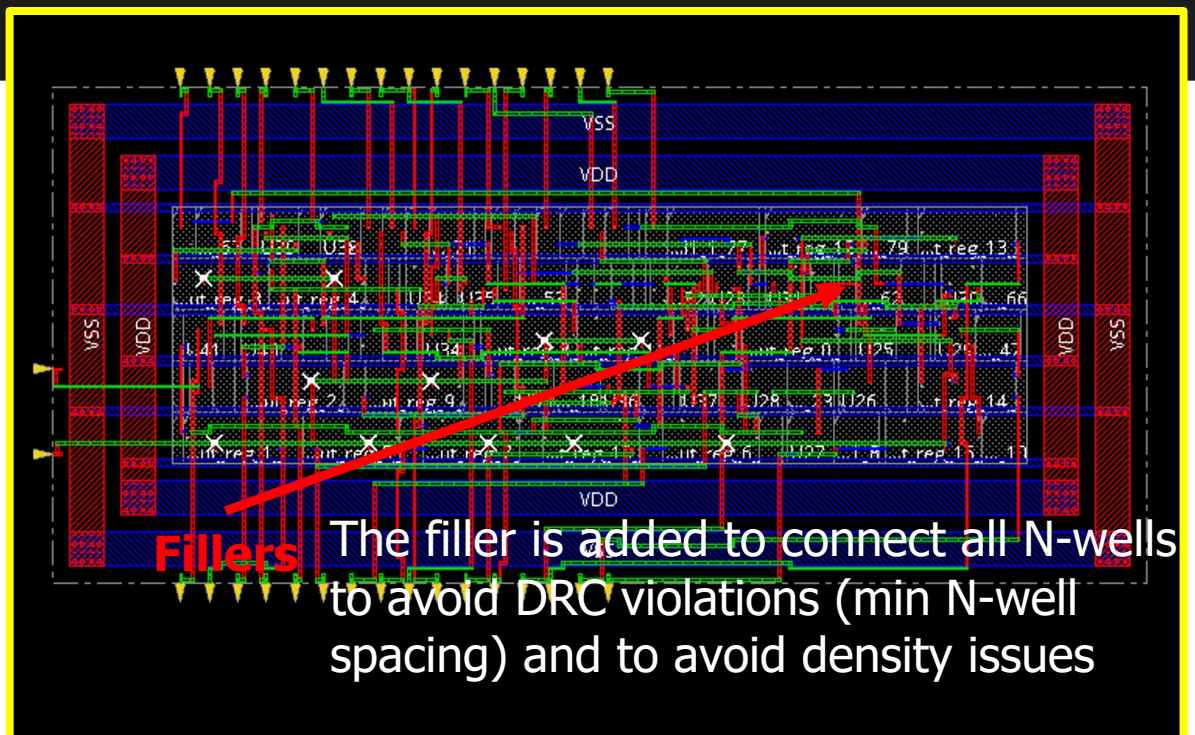
- Step 9) Extract and optimize
- setOptMode
- optDesign

```
Puts "#####################################"
Puts "###"
Puts "### RC Extraction and Optimization ..."
Puts "###"
Puts "#####################################"

# Set the native RC extraction mode
# Notes: This command should be used before using the extractRC command
setExtractRCMode -engine postRoute -effortLevel low

# Extract resistance and capacitance for the interconnects and store the results in an RC database
extractRC

# Set global analysis modes for timing analysis
# Notes: '-analysisType onChipVaration' option calculates the delay for one path based on maximum operation
#        condition while calculating the delay for another path based on minimum operating condition
#        for setup or hold checks
setAnalysisMode -analysisType onChipVariation

# Set global parameters for timing optimization
setOptMode -yieldEffort none
setOptMode -effort high
setOptMode -drcMargin 0.0
setOptMode -holdTargetSlack 0.2 -setupTargetSlack 0.0
setOptMode -holdFixingEffort high
setOptMode -simplifyNetlist false
setOptMode -usefulSkew false
setOptMode -moveInst true
setOptMode -reclaimArea true
setOptMode -fixDRC true
setOptMode -fixCap true

# Perform timing optimization before or after the clock tree is built, or after routing
# Notes: '-hold' option corrects hold violations
#        '-postRoute' option performs timing optimization on a design whose routing is complete
optDesign -postRoute -setup -hold

# Connect all new cells to VDD/GND
#globalNetConnect VDD -type pgpin -pin {VDD} -override
#globalNetConnect VSS -type pgpin -pin {VSS} -override
#globalNetConnect VDD -type tiehi
#globalNetConnect VSS -type tielo
connect_std_cells_to_power
applyGlobalNets

# Save design
saveDesign "$design_name.routed.enc"
```

down_counter.tcl

Step 10) Add fillers

```
Puts "############################"
Puts "###"
Puts "### Add Decap or Fillers ..."
Puts "###"
Puts "############################"

# Check width, spacing, and internal geometry of objects and the wiring between them
# Create and save violation markers in the design database
verify_drc

# Add filler cells
# Notes: '-cell' option specifies the list of filler cells to add
# addFiller -cell FILL16TS FILL1TS FILL2TS FILL32TS FILL4TS FILL64TS FILL8TS -prefix IBM13RFLPVT_FILLER
#addFiller -cell FILL1 FILL2 FILL4 FILL8 FILL16 FILL32 FILL64 FILL1_LL FILL_NW_FA_LL FILL_NW_HH FILL_NW_LL -fillBoundary
#addDeCap -cell DCAP64 DCAP32 DCAP16 DCAP8 DCAP4
addFiller -cell DCAP64 DCAP32 DCAP16 DCAP8 DCAP4 FILL64 FILL32 FILL16 FILL8 FILL4 FILL2 FILL1 -prefix FILLER
verifyGeometry
verify_drc
redraw
```



down_counter.tcl

- Step 11) Verify
- Step 12) Generate outputs
  - There should be no violations from all the reports
    - final.cap.tarpt (**Violation on 'clk' node is allowed**)
    - $(design_name).antenna.rpt
    - $(design_name).conn.rpt
    - $(design_name).critnet.rpt
    - $(design_name).geom.rpt
    - $(design_name).hold/setup.rpt
  - Output files
    - $(design_name).power.rpt
    - $(design_name).summary.rpt
    - $(design_name).final.def
    - $(design_name).gds & lef
    - $(design_name).(phy/nophy).v
    - $(design_name).(verilog).sdf
    - $(design_name).spef

Two .sdf files are generated
- $(design_name).sdf is for top-level APR
- $(design_name).verilog.sdf is for post-APR Verilog simulation

down_counter.tcl

- Step 11) Verify
- Step 12) Generate outputs
  - There should be no violations fr
    - final.cap.tarpt (**Violation on 'c**
    - $(design_name).antenna.rpt
    - $(design_name).conn.rpt
    - $(design_name).critnet.rpt
    - $(design_name).geom.rpt
    - $(design_name).hold/setup.rpt
  - Output files
    - $(design_name).power.rpt
    - $(design_name).summary.rpt
    - $(design_name).final.def
    - $(design_name).gds & lef
    - $(design_name).(phy/nophy).v
    - $(design_name).(verilog).sdf
    - $(design_name).spef

lfsr1.setup.rpt

```
Path 1: MET Setup Check with Pin lfsr_out_reg_3_/CK
Endpoint:   lfsr_out_reg_3_/D (v) checked with  leading edge of 'clk'
Beginpoint: resetn            (^) triggered by  leading edge of 'clk'
Path Groups: {clk}
Analysis View: typical
Other End Arrival Time          0.000
- Setup                         0.345
+ Phase Shift                   4.000
+ CPPR Adjustment               0.000
- Uncertainty                   0.010
= Required Time                 3.645
- Arrival Time                  2.677
= Slack Time                    0.968
     Clock Rise Edge                     0.000
     + Input Delay                       0.100
     + Drive Adjustment                  0.041
     = Beginpoint Arrival Time           0.141
     +-------------------------------------------------------------------------------+
     |         Instance          |     Arc      |    Cell    | Delay | Arrival | Required |
     |                           |              |            |       |  Time   |   Time   |
     |---------------------------+--------------+------------+-------+---------+----------|
     |                           | resetn ^     |            |       |   0.141 |    1.109 |
     | FE_OFC36_resetn           | A ^ -> Y ^   | CLKBUFX2TS | 0.182 |   0.323 |    1.290 |
     | FE_OFC15_resetn           | A ^ -> Y ^   | CLKBUFX2TS | 0.202 |   0.524 |    1.492 |
     | FE_OFC16_resetn           | A ^ -> Y ^   | CLKBUFX2TS | 0.214 |   0.738 |    1.706 |
     | FE_OFC17_resetn           | A ^ -> Y v   | INVX2TS    | 0.095 |   0.833 |    1.801 |
     | FE_OFC19_resetn           | A v -> Y ^   | INVX2TS    | 0.081 |   0.914 |    1.881 |
     | FE_OFC20_resetn           | A ^ -> Y ^   | CLKBUFX2TS | 0.197 |   1.111 |    2.078 |
     | FE_DBTC0_resetn           | A ^ -> Y v   | INVX2TS    | 0.091 |   1.202 |    2.170 |
     | FE_OFC21_FE_DBTN0_resetn  | A v -> Y v   | CLKBUFX2TS | 0.217 |   1.419 |    2.386 |
     | FE_OFC22_FE_DBTN0_resetn  | A v -> Y v   | CLKBUFX2TS | 0.245 |   1.664 |    2.632 |
     | FE_OFC23_FE_DBTN0_resetn  | A v -> Y v   | CLKBUFX2TS | 0.249 |   1.913 |    2.881 |
     | FE_OFC24_FE_DBTN0_resetn  | A v -> Y v   | CLKBUFX2TS | 0.252 |   2.165 |    3.133 |
     | U39                       | B0 v -> Y v  | AO22XLTS   | 0.512 |   2.677 |    3.645 |
     | lfsr_out_reg_3_           | D v          | DFFQX1TS   | 0.000 |   2.677 |    3.645 |
     +-------------------------------------------------------------------------------+
```

lfsr1.summary.rpt

```
============================
Floorplan/Placement Information
============================
Total area of Standard cells: 1080.000 um^2
Total area of Standard cells(Subtracting Physical Cells): 708.480 um^2
Total area of Macros: 0.000 um^2
Total area of Blockages: 0.000 um^2
Total area of Pad cells: 0.000 um^2
Total area of Core: 1080.000 um^2
Total area of Chip: 2672.640 um^2
Effective Utilization: 1.0000e+00
Number of Cell Rows: 5
% Pure Gate Density #1 (Subtracting BLOCKAGES): 100.000%
% Pure Gate Density #2 (Subtracting BLOCKAGES and Physical Cells): 65.600%
% Pure Gate Density #3 (Subtracting MACROS): 100.000%
% Pure Gate Density #4 (Subtracting MACROS and Physical Cells): 65.600%
% Pure Gate Density #5 (Subtracting MACROS and BLOCKAGES): 100.000%
% Pure Gate Density #6 (Subtracting MACROS and BLOCKAGES for insts are not placed): 65.600%
% Core Density (Counting Std Cells and MACROs): 100.000%
% Core Density #2(Subtracting Physical Cells): 65.600%
% Chip Density (Counting Std Cells and MACROs and IOs): 40.409%
% Chip Density #2(Subtracting Physical Cells): 26.509%
# Macros within 5 sites of IO pad: No
Macro halo defined?: No
```

Pure gate density = 65%

.cdl File

- Circuit Description Language (CDL) is a kind of netlist, a description of an electronic circuit

- After generating the .v file from APR, we need to generate .cdl file for circuit-level simulation and LVS check
  - .cdl netlist will be compared to the layout instead of .sp netlist

- Command: $perl v2lvs.pl $(design_name)
  - This command runs the Perl script
  - You can use the 'makecdl.sh' shell script or Makefile

Post-APR Verilog Simulation

- We need to do a Verilog dynamic simulation to verify functionality
- We can do SPICE/Spectre simulation. But since it takes much more time, it should be used for timing and power consumption, rather than functionality

runsim.do

```
# include standard cell verilog model
vlog +acc -incr /courses/ee6350/pdk2025/tcbn65gplus/TSMCHOME/digital/Front_End/verilog/tcbn65gplus_140b/tcbn65gplus_pwr.v

# include the testbench file
vlog +acc -incr tb_down_counter.v

# include verilog modules
vlog +acc -incr ../../innovus/down_counter/down_counter.PG.v

# run simulation with sdf annotations and check waveforms

vsim -voptargs=+acc -t ps -lib work \
-sdfmax testbench/down_counter_inst=../../innovus/down_counter/down_counter.verilog.sdf \
testbench

do waveformat.do
run -all
```

Gate-level netlist from APR & std. cell library

Delay info (sdf) annotation

tb_down_counter.v

```
down_counter down_counter_inst (
    .clk(clk),
    .rstn(rstn),
    .start(start),
    .reset(reset),
    .counter_out(counter_out),
    .VDD(VDD),
    .VSS(VSS)
);

//initial $sdf_annotate("../../innovus/down_counter/down_counter.verilog.sdf", down_counter_inst, , , "maximum")

always begin
    `HALF_CLK_CYCLE clk = ~clk;
end
```

It can be done in the tb

GDS Import into Cadence Virtuoso

.gds File

- After running the APR, you will obtain a GDS file.
- You can import the GDS file into Cadence Virtuoso to create the layout view
- In the Cadence Virtuoso environment, you can run LVS and DRC using Siemens/MentorGraphics Calibre

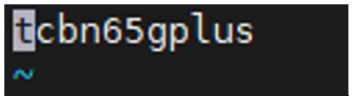
- You will do various jobs (e.g., parasitic extraction) on the layout view in the Cadence Virtuoso environment, and eventually export the layout to a GDS file
- We will submit this GDS file to a foundry for the actual tapeout

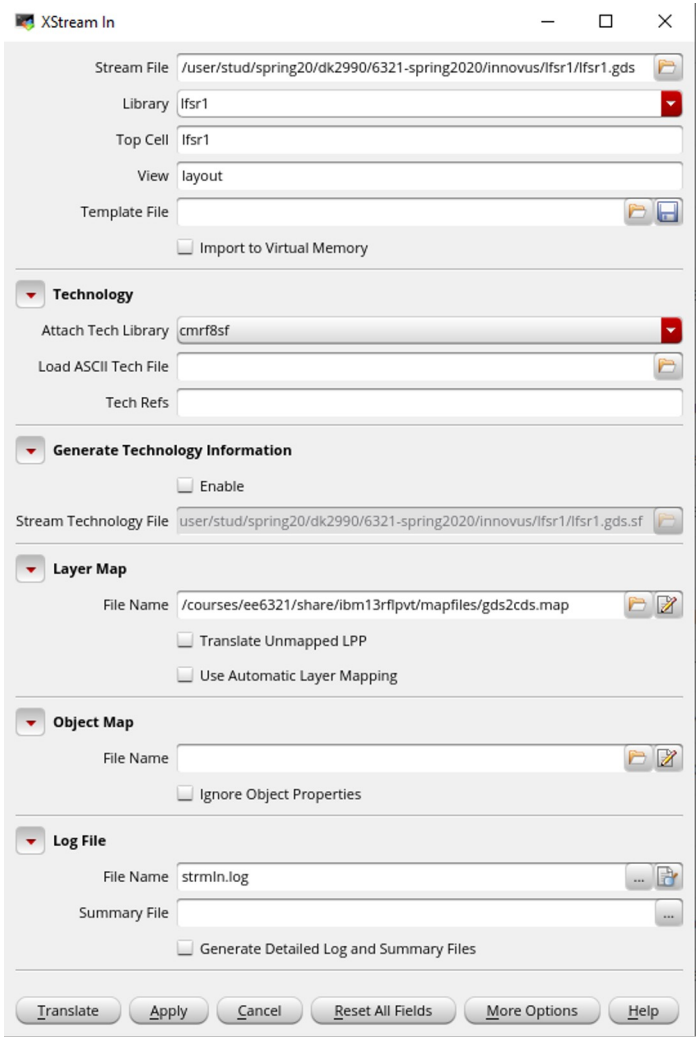
- Please refer to or load /virtuoso/gds_import.template

How to Import the .gds File

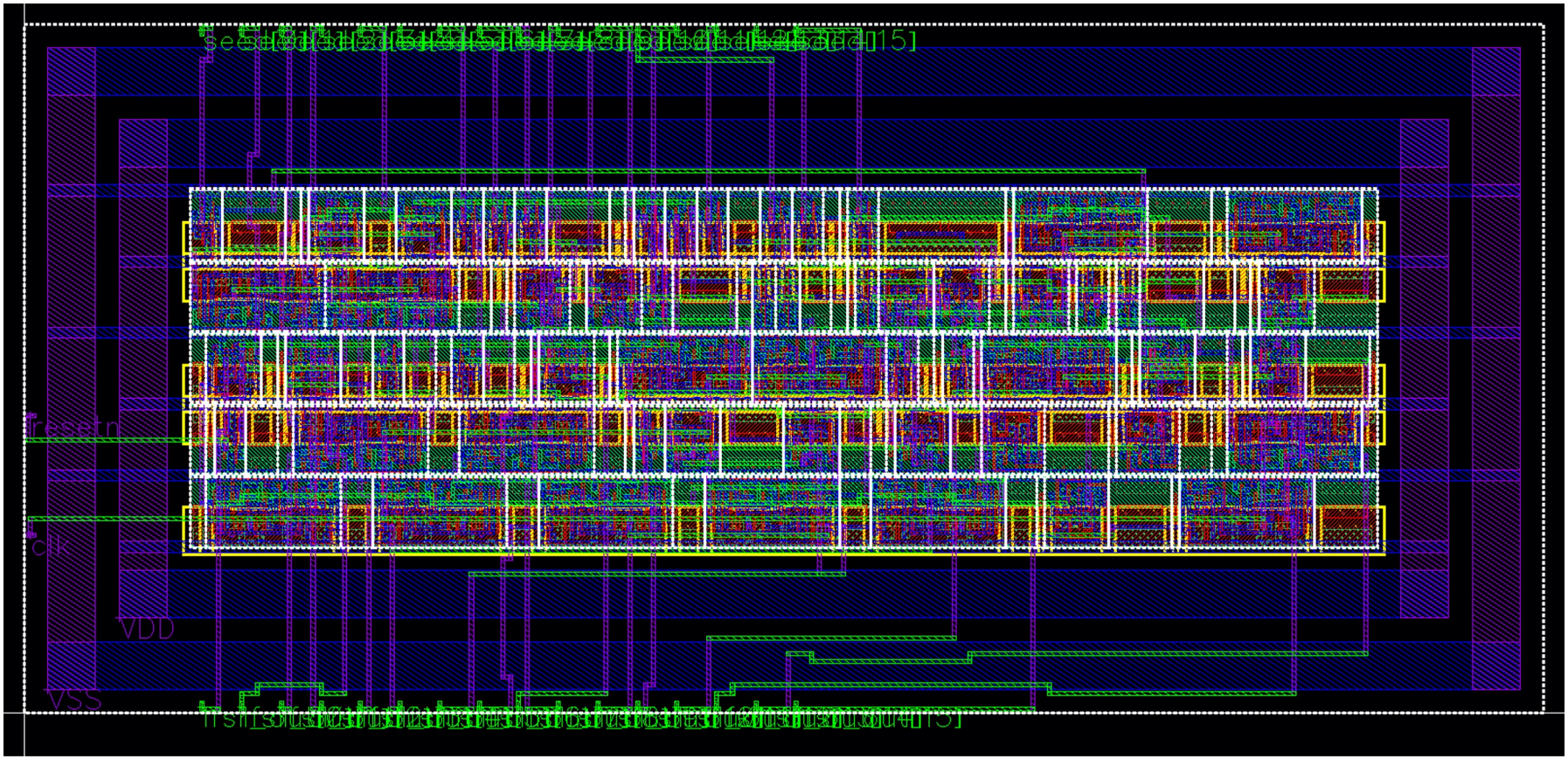
1. Generate 'refLib/refLib.list' at your Virtuoso directory, like the picture on the right
2. Run Virtuoso and make the library (Need to attach the technology 'tsmcN65')
3. Select CIW → File → Import → Stream…
4. Change the settings as below
   - Stream File: .gds file generated from APR
   - Library: The name of lib. made from (2)
   - Top Cell: Block name
   - View: layout
   - Attach Tech Library: tsmcN65
   - Layer Map – File Name: /courses/ee6350/pdk2025/CMN65GP/T-N65-CM-SP-018-K3_RF_1p9m_6X1Z1U_ALRDL/tsmcN65/tsmcN65.layermap
5. Click the 'More Options' button and move to the 'Mapping' tab
6. Load 'refLib.list' file in 'Ref Lib File Name.'
7. Click the 'OK' button, and 'Translate.'

refLib.list





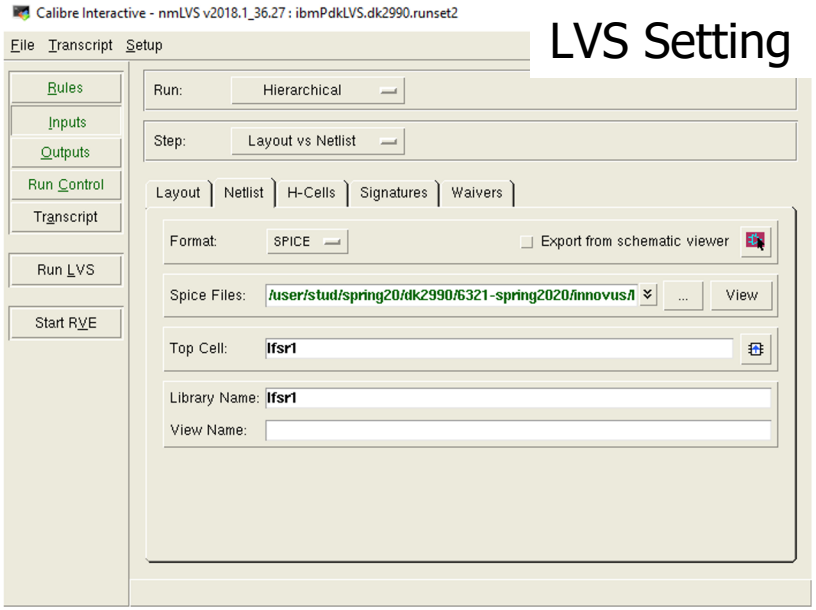
Imported Layout



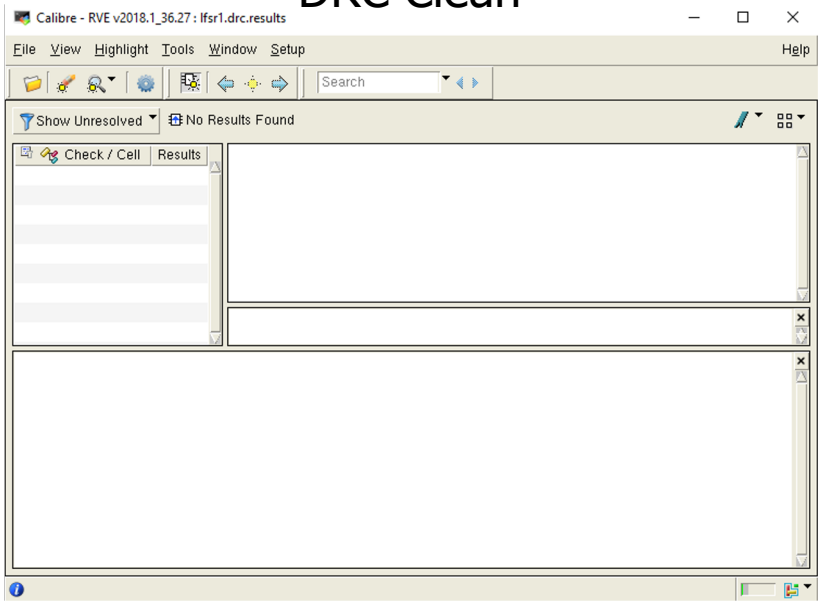
DRC/LVS Check

- DRC/LVS check can be done in the same way as introduced in EECS 4321
  - However, you need to change the input netlist file (Spice Files) as .cdl generated from APR in **LVS**
  - Also, uncheck 'Export from schematic viewer.'

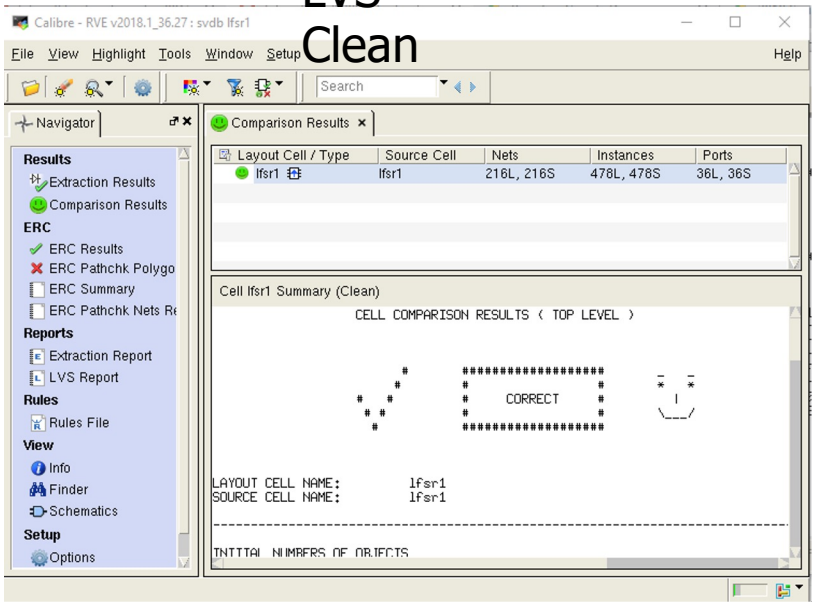
LVS Setting



DRC Clean



LVS Clean

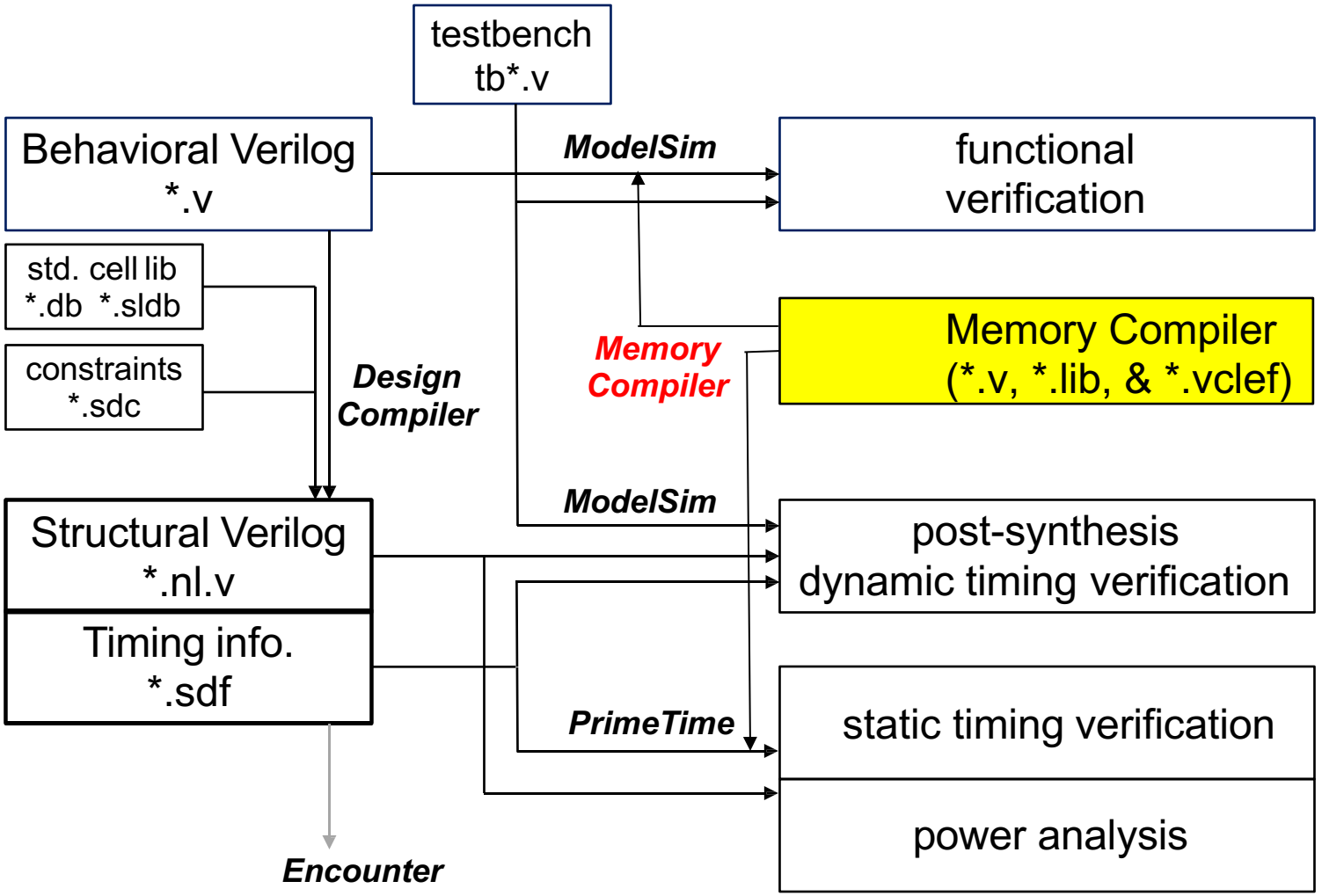


EECS E6321 Tutorial 04

Block-level APR and SRAM Compiler

Mingoo Seok & Previous TAs

Overview



Automatic Placement and Routing

Memory Compiler

- Using the SRAM memory compiler, one can generate single and dual-port, high speed and high density SRAM.

- Located at:
  /tools/misc/EE6321/arm/tsmc/cln65gplus/

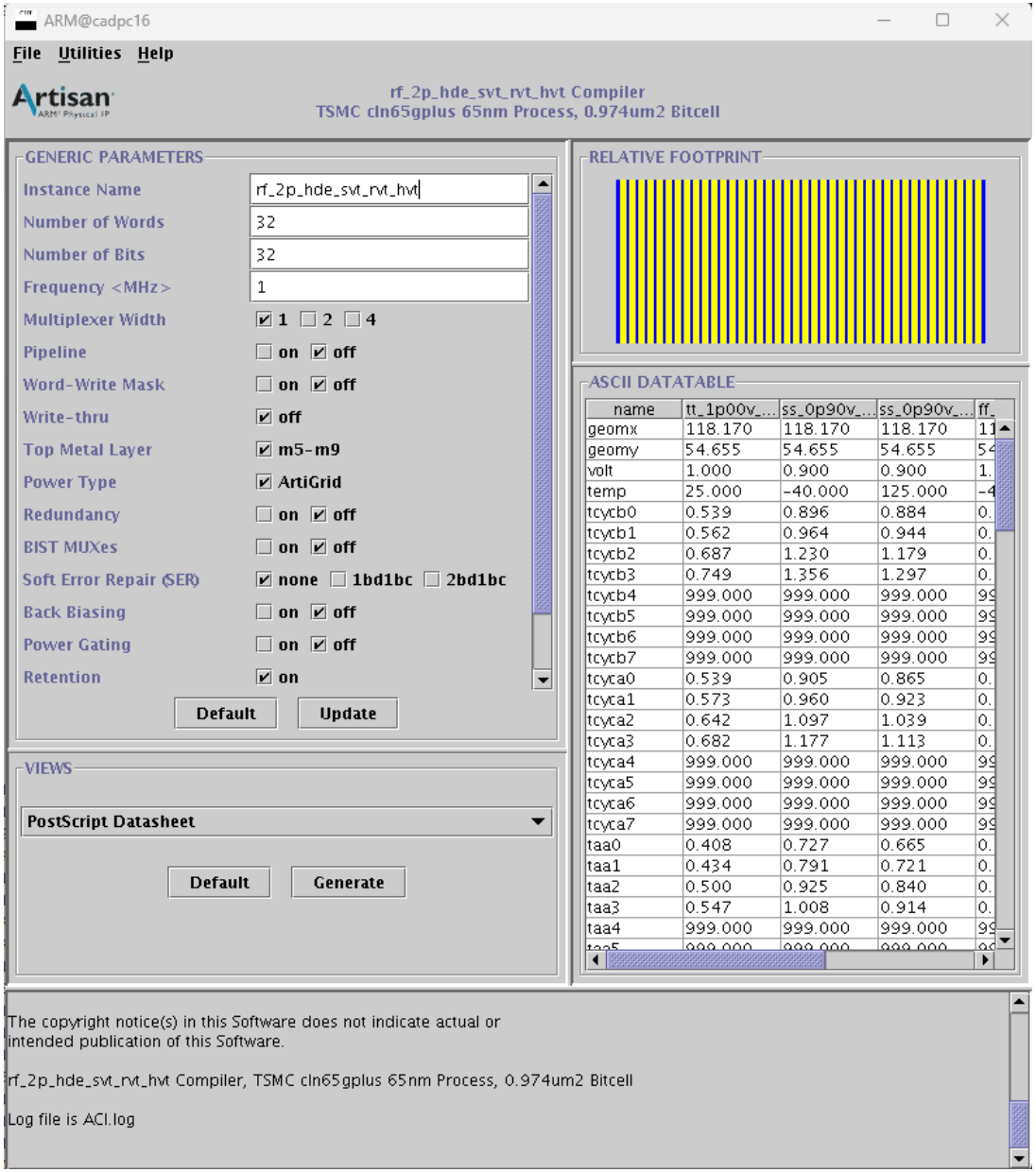
- Flavors
  - rf_2p_hde_svt_rvt_hvt : High Density Two-Port Register File Compiler
  - rf_sp_hdd_svt_rvt_hvt : High Density Single-Port Register File Compiler
  - rom_via_hde_hvt_rvt_hvt : High Density ROM Complier
  - sram_dp_hdc_svt_rvt_hvt : High Density Dual-Port SRAM Compiler
  - sram_sp_hdc_svt_rvt_hvt : High Density Single-Port SRAM Compiler

- Documents are available

SRAM Generator Feature

- Optimized for High Speed/High Density or Low-Power or High Density
- Aspect Ratio Control for Efficient Floor Planning
- Memory Operation and Retention at Low Frequency
- Low Active Power and Leakage-Only Standby Power
- Timing and Power Models for Industry-Leading Design Tools
- Configurable Word-Write Mask Option

GUI

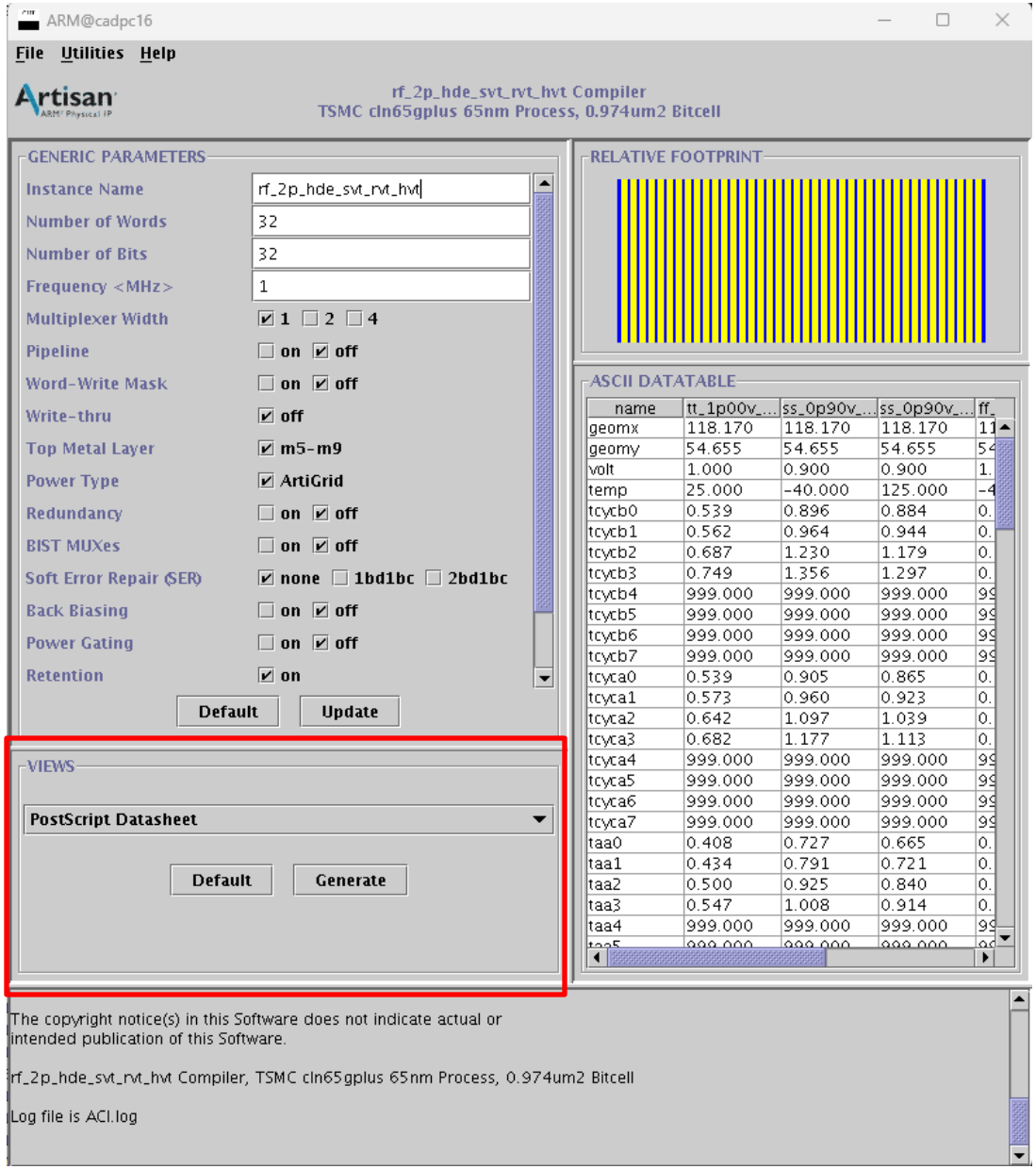


Executable: [ms4415@cadpc16 ~]$ /tools/misc/EE6321/arm/tsmc/cln65gplus/rf_2p_hde_svt_rvt_hvt/r0p2-00eac0/bin/rf_2p_hde_svt_rvt_hvt

Different executable for each flavor

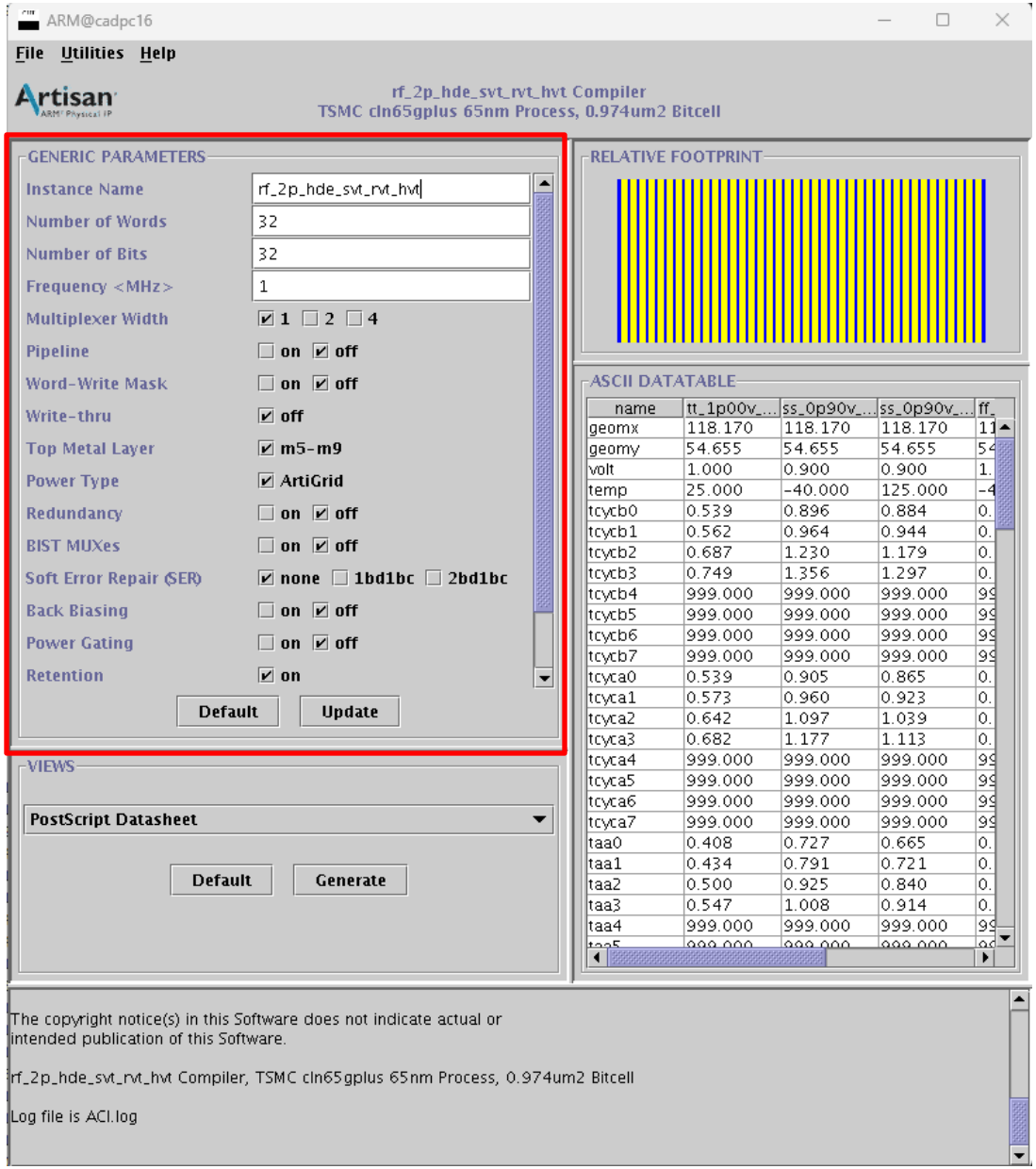
You can generate memory circuits in the command line mode (later slides)

Views (Output Files)



- PostScript Datasheet
- ASCII Datatable
- Verilog Model
- Synopsys Model
- VCLEF Footprint
- GDSII Layout
- LVS Netlist
- TetraMax Model
- PrimeTime Model
- Repair & SER Verilog

Circuits Parameters



- Number of words
- Number of bits
- Frequency
- Multiplexer width [um]
- …
- …
- Retention
- Extra margin adjustment
- Advanced test features

Please check the userguide

Generating SRAM Example (1)

Create “mem_comp” folder in your directory

Copy the contents from /courses/ee6350/proj_2025Spring/ref/memory_compiler to your local folder

Take a look at: /memory_compiler/sram00/run.sh

- It generates Verilog, synopsys, lvs, vclef-fp, and gds2
- Check the userguide.pdf for parameter settings

```
/tools/misc/EE6321/arm/tsmc/cln65gplus/sram_sp_hdc_svt_rvt_hvt/r0p0-00eac0/bin/sram_sp_hdc_svt_rvt_hvt verilog -instname sram00 -words 8192 -bits 32 -frequency 100 -mux 16 -drive 6 -write_mask on -write_thru on -wp_size 8 -top_layer m5-m9 -redundancy off -bmux off -ser none -back_biasing off -power_gating off -atf off -cust_comment "First Attempt" -left_bus_delim "[" -right_bus_delim "]" -prefix "" -name_case upper -check_instname on -diodes on -pwr_gnd_rename VDDPE:VDD,VDDCE:VDD,VSSE:VSS

/tools/misc/EE6321/arm/tsmc/cln65gplus/sram_sp_hdc_svt_rvt_hvt/r0p0-00eac0/bin/sram_sp_hdc_svt_rvt_hvt synopsys -instname sram00 -words 8192 -bits 32 -frequency 100 -mux 16 -drive 6 -write_mask on -write_thru on -wp_size 8 -top_layer m5-m9 -redundancy off -bmux off -ser none -back_biasing off -power_gating off -atf off -cust_comment "First Attempt" -left_bus_delim "[" -right_bus_delim "]" -prefix "" -name_case upper -check_instname on -diodes on -pwr_gnd_rename VDDPE:VDD,VDDCE:VDD,VSSE:VSS

/tools/misc/EE6321/arm/tsmc/cln65gplus/sram_sp_hdc_svt_rvt_hvt/r0p0-00eac0/bin/sram_sp_hdc_svt_rvt_hvt lvs -instname sram00 -words 8192 -bits 32 -frequency 100 -mux 16 -drive 6 -write_mask on -write_thru on -wp_size 8 -top_layer m5-m9 -redundancy off -bmux off -ser none -back_biasing off -power_gating off -atf off -cust_comment "First Attempt" -left_bus_delim "[" -right_bus_delim "]" -prefix "" -name_case upper -check_instname on -diodes on -pwr_gnd_rename VDDPE:VDD,VDDCE:VDD,VSSE:VSS

/tools/misc/EE6321/arm/tsmc/cln65gplus/sram_sp_hdc_svt_rvt_hvt/r0p0-00eac0/bin/sram_sp_hdc_svt_rvt_hvt vclef-fp -instname sram00 -words 8192 -bits 32 -frequency 100 -mux 16 -drive 6 -write_mask on -write_thru on -wp_size 8 -top_layer m5-m9 -redundancy off -bmux off -ser none -back_biasing off -power_gating off -atf off -cust_comment "First Attempt" -left_bus_delim "[" -right_bus_delim "]" -prefix "" -name_case upper -check_instname on -diodes on -pwr_gnd_rename VDDPE:VDD,VDDCE:VDD,VSSE:VSS

/tools/misc/EE6321/arm/tsmc/cln65gplus/sram_sp_hdc_svt_rvt_hvt/r0p0-00eac0/bin/sram_sp_hdc_svt_rvt_hvt gds2 -instname sram00 -words 8192 -bits 32 -frequency 100 -mux 16 -drive 6 -write_mask on -write_thru on -wp_size 8 -top_layer m5-m9 -redundancy off -bmux off -ser none -back_biasing off -power_gating off -atf off -cust_comment "First Attempt" -left_bus_delim "[" -right_bus_delim "]" -prefix "" -name_case upper -check_instname on -diodes on -pwr_gnd_rename VDDPE:VDD,VDDCE:VDD,VSSE:VSS
```

Generating SRAM Example (2)

```
-rw-r-----. 1 ms4415 ee6350-all     2233 Dec 12 10:49 ACI.log
-rwxr-----. 1 ms4415 ee6350-all       45 Dec 12 10:49 gen_db.sh
-rw-r-----. 1 ms4415 ee6350-all      183 Dec 12 10:49 lc.command
-rw-r-----. 1 ms4415 ee6350-all      182 Feb  6 09:41 lc_command.log
-rw-r-----. 1 ms4415 ee6350-all     2568 Feb  6 09:41 lc.log
-rw-r-----. 1 ms4415 ee6350-all     2562 Feb  6 09:41 lc_output.txt
-rwxr-----. 1 ms4415 ee6350-all       45 Dec 20 09:15 nuke.sh
-rwxr-----. 1 ms4415 ee6350-all     2469 Dec 20 09:14 run.sh
-rw-r-----. 1 ms4415 ee6350-all     5170 Dec 12 10:49 sram00_ant.clf
-rw-r-----. 1 ms4415 ee6350-all  4740077 Dec 12 10:49 sram00.cdl
-rw-r-----. 1 ms4415 ee6350-all 22007808 Dec 12 10:49 sram00.gds2
drwxr-x---. 2 ms4415 ee6350-all     4096 Dec 14 13:07 sram00_libs
-rw-r--r--. 1 ms4415 ee6350-all    78395 Feb  6 09:39 sram00.v
-rw-r-----. 1 ms4415 ee6350-all    52085 Dec 12 10:49 sram00.vclef
```

- Run run.sh will generates the following files:
  - .cdl – for LVS
  - .gds2 – for Virtuoso
  - .v – RTL model
  - .vclef – for P&R; pin locations and layout size in the APR tools
  - .clf - for P&R antenna (?)

Generating SRAM Example (3)

```
[ms4415@cadpc16 sram00_libs]$ pwd
/courses/ee6350/proj_2025Spring/ref/memory_compiler/sram00/sram00_libs
[ms4415@cadpc16 sram00_libs]$ ll
total 572
-rwxr-x---. 1 ms4415 ee6350-all 85881 Dec 12 10:49 sram00_nldm_ff_1p10v_1p10v_0c_syn.lib
-rwxr-x---. 1 ms4415 ee6350-all 85893 Dec 12 10:49 sram00_nldm_ff_1p10v_1p10v_125c_syn.lib
-rwxr-x---. 1 ms4415 ee6350-all 85890 Dec 12 10:49 sram00_nldm_ff_1p10v_1p10v_m40c_syn.lib
-rwxr-x---. 1 ms4415 ee6350-all 85891 Dec 12 10:49 sram00_nldm_ss_0p90v_0p90v_125c_syn.lib
-rwxr-x---. 1 ms4415 ee6350-all 85890 Dec 12 10:49 sram00_nldm_ss_0p90v_0p90v_m40c_syn.lib
-rwxr-x---. 1 ms4415 ee6350-all 40960 Feb  6 09:41 sram00_nldm_tt_1p00v_1p00v_25c_syn.db
-rwxr-x---. 1 ms4415 ee6350-all 85884 Dec 12 10:49 sram00_nldm_tt_1p00v_1p00v_25c_syn.lib
[ms4415@cadpc16 sram00_libs]$
```

Inspect and run gen_db.sh

- gen_db.sh: lc_shell -xg_mode -f lc.command | tee lc.log

It will generate a set of .db files (timing and power lib files)

- For logic synthesis, static timing analysis
- For P&R

Dynamic Verilog Simulation

Using Questrasim

Inspect the following location:
/courses/ee6350/proj_2025Spring/ref/qsim_rtl/tb_sram_wrapper

Files

- tb_sram_wrapper.v → testbench
- ../../rtl/sram_wrapper/sram_wrapper.v → sram_wrapper, particularly needed if you integrate multiple macros into a single memory module
- ../../memory_compiler/sram00/sram00.v → the macro, generated by the memory complier

Static Timing Analysis

Using Primetime

Inspect the following location:
/courses/ee6350/proj_2025Spring/ref/pt_dc/dut

Files

pt.tcl → should refer the .db file