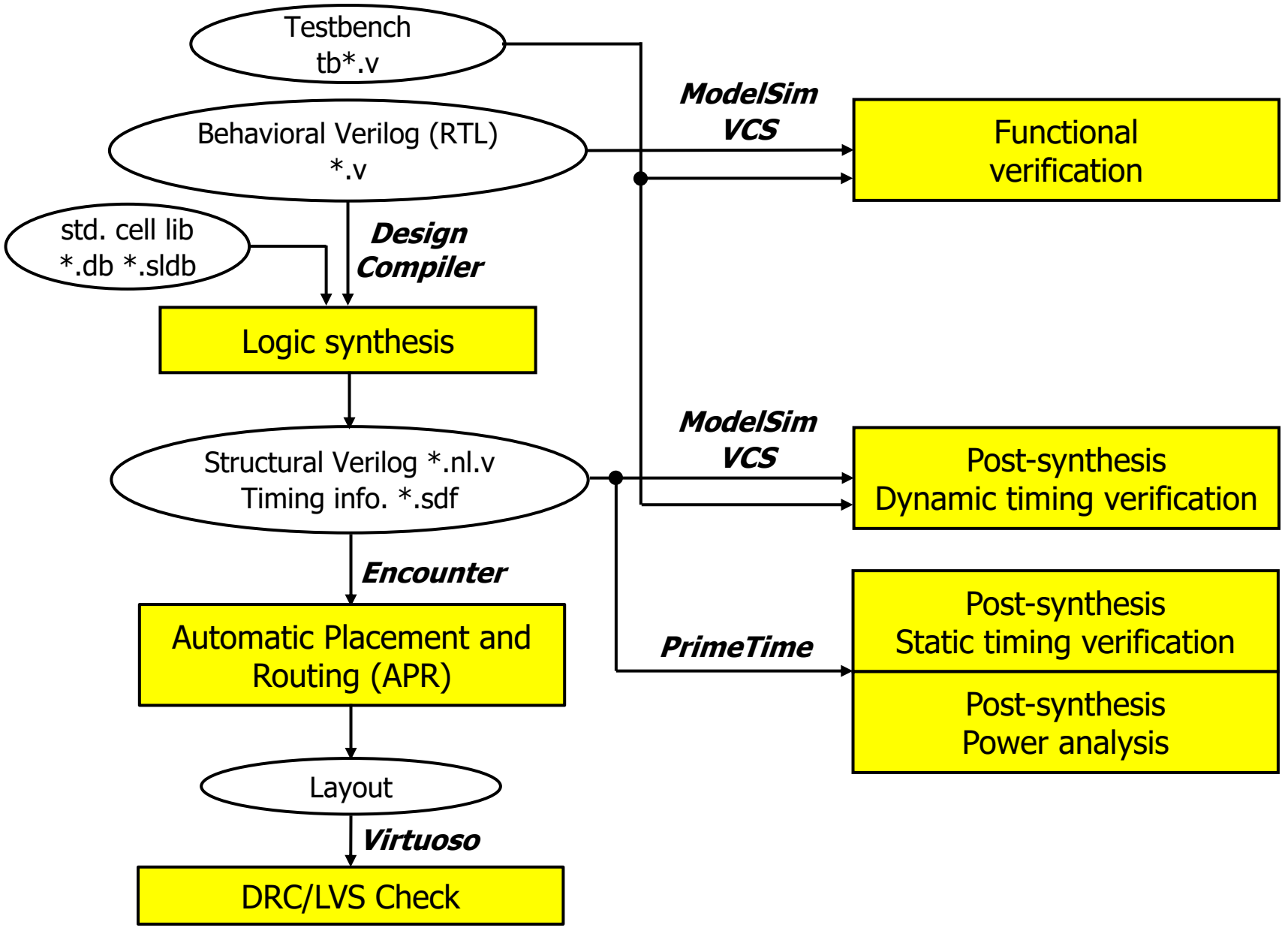
EECS E6321 Tutorial 05

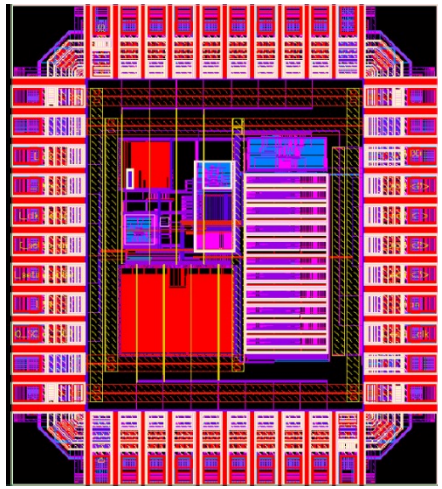
Top-level APR

Mingoo Seok & Previous TAs

Semi-Custom Design Flow: Block-level



Semi-Custom Design Flow – Top-level



Blocks with layout
- Semi-custom designed block
- Full-custom designed block
- I/O pads

Encounter

Top-level APR

Layout

Virtuoso

Final post-PEX verification

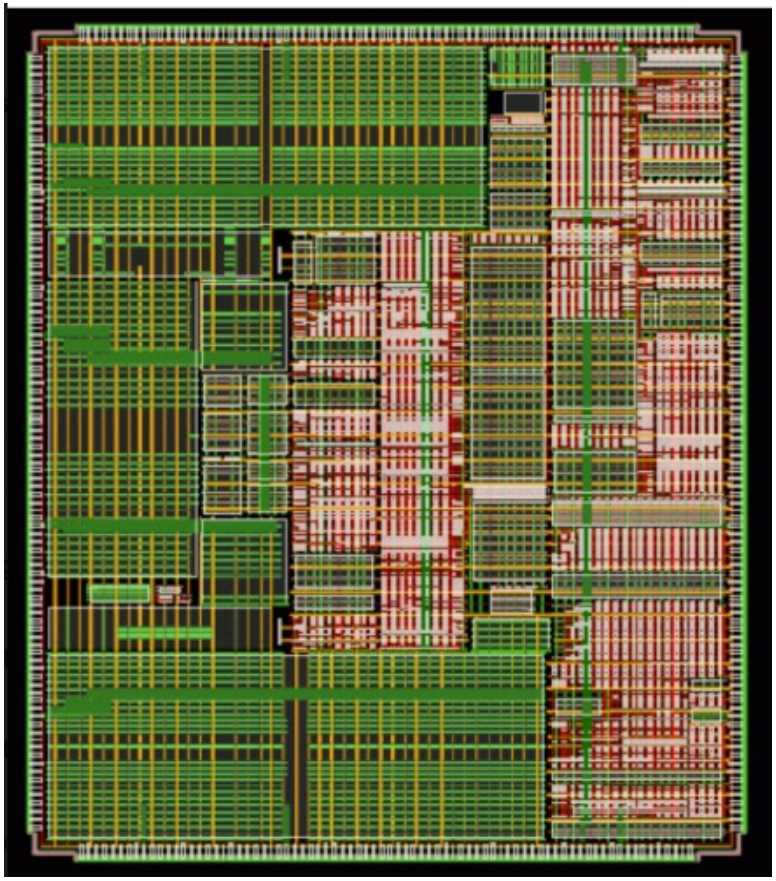
Virtuoso

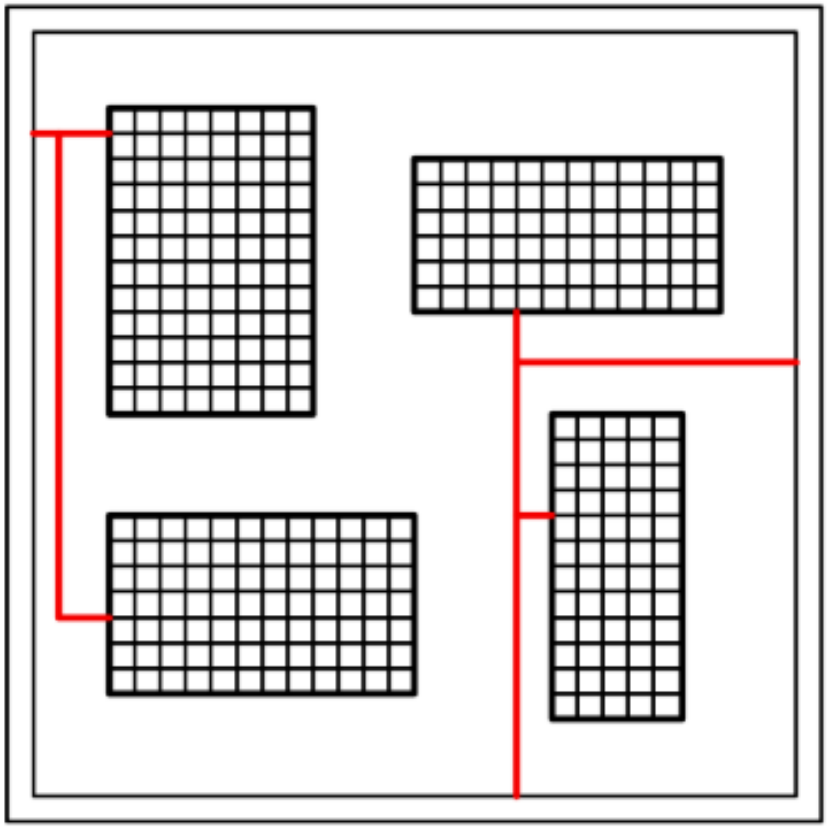
DRC/LVS Check

TAPE OUT

Top-Level APR

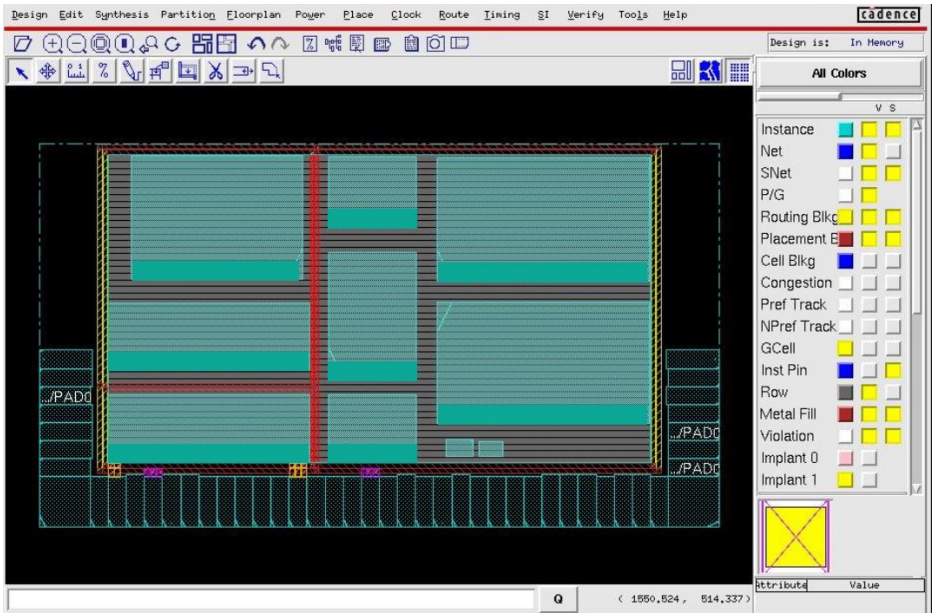
What should We Do?

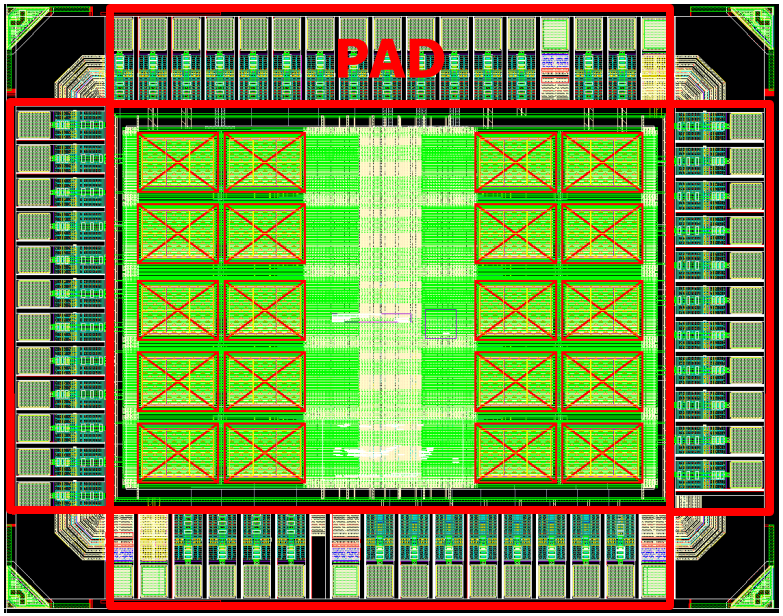




- Draw power grids
- Place blocks and std cells
- Create clock distribution networks
- Route signals among them

What should We Do?





- Place I/O cells and bonding pads (or bumps if it is a flip chip)
  - We will not do this
  - However, the ref project has the sample script that you can refer

Important Files

- top.v (In 'rtl' directory)
  - Top-level **structural** model
- A tool uses the **.v file** and the **.lef file** to perform routing

```
module top (
    inout VDD,
    inout VDD_TEST,
    input VSS,
    input clk_ext,
    input rstn,
    input reset,
    input start,

    input phi,
    input phib,
    input scan_in,
    input scan_i0o1,
    input load,
    output scan_out,
    output clk_div_1k,
    output [7:0] counter_out
);

    wire w_clk;
    wire [7:0] w_sram_dout;

    dut dut_inst (
        .VDD(VDD),
        .VSS(VSS),
        .clk(w_clk),
        .rstn(rstn),
        .reset(reset),
        .start(start),
        .sram_dout(w_sram_dout),
        .counter_out(counter_out)
    );

    wire w_en_cnt;
    wire w_en_int;
    wire [3:0] w_div;
    wire [4:0] w_fc;
```

```
    test_clkgen test_clkgen_inst (
        .VDD(VDD_TEST),
        .VSS(VSS),
        .clk_ext(clk_ext),
        .en_cnt(w_en_cnt),
        .en_int(w_en_int),
        .div0(w_div[0]),
        .div1(w_div[1]),
        .div2(w_div[2]),
        .div3(w_div[3]),
        .fc0(w_fc[0]),
        .fc1(w_fc[1]),
        .fc2(w_fc[2]),
        .fc3(w_fc[3]),
        .fc4(w_fc[4]),
        .clk_out(w_clk),
        .cnt_out(clk_div_1k)
    );

    scan_chain scan_chain_inst (
        .VDD(VDD_TEST),
        .VSS(VSS),
        .sram_dout(w_sram_dout),
        .en_int(w_en_int),
        .en_cnt(w_en_cnt),
        .fc(w_fc),
        .div(w_div),
        .phi(phi),
        .phib(phib),
        .scan_i0o1(scan_i0o1),
        .load(load),
        .scan_in(scan_in),
        .scan_out(scan_out),
        .phi_out(),
        .phib_out(),
        .scan_i0o1_out(),
        .load_out()
    );

endmodule
```

Global VDD vs. Local VDD

- In this lab, we used two VDD: VDD and VDD_TEST.
- You need to specify VDD & VSS as ports in module description and instantiation

```
module top (
    inout VDD,
    inout VDD_TEST,
    input VSS,
    input clk_ext,
    input rstn,
    input reset,
    input start,

    input phi,
    input phib,
    input scan_in,
    input scan_i0o1,
    input load,
    output scan_out,
    output clk_div_1k,
    output [7:0] counter_out
);

    wire w_clk;
    wire [7:0] w_sram_dout;

    dut dut_inst (
        .VDD(VDD),
        .VSS(VSS),
        .clk(w_clk),
        .rstn(rstn),
        .reset(reset),
        .start(start),
        .sram_dout(w_sram_dout),
        .counter_out(counter_out)
    );

    wire w_en_cnt;
    wire w_en_int;
    wire [3:0] w_div;
    wire [4:0] w_fc;
```

```
    test_clkgen test_clkgen_inst (
        .VDD(VDD_TEST),
        .VSS(VSS),
        .clk_ext(clk_ext),
        .en_cnt(w_en_cnt),
        .en_int(w_en_int),
        .div0(w_div[0]),
        .div1(w_div[1]),
        .div2(w_div[2]),
        .div3(w_div[3]),
        .fc0(w_fc[0]),
        .fc1(w_fc[1]),
        .fc2(w_fc[2]),
        .fc3(w_fc[3]),
        .fc4(w_fc[4]),
        .clk_out(w_clk),
        .cnt_out(clk_div_1k)
    );

    scan_chain scan_chain_inst (
        .VDD(VDD_TEST),
        .VSS(VSS),
        .sram_dout(w_sram_dout),
        .en_int(w_en_int),
        .en_cnt(w_en_cnt),
        .fc(w_fc),
        .div(w_div),
        .phi(phi),
        .phib(phib),
        .scan_i0o1(scan_i0o1),
        .load(load),
        .scan_in(scan_in),
        .scan_out(scan_out),
        .phi_out(),
        .phib_out(),
        .scan_i0o1_out(),
        .load_out()
    );

endmodule
```

Important Files

- In the innovus directory
- top.tcl
  - Includes general setting and commands for Innovus design flow
- config.globals
  - Input configuration file
  - Sets the design, I/O file, .lef file
- top.io
  - I/O assignment
- No mmmc.view file (No timing constraint)

- run_apr: run innovus w/o GUI
- makecdl.sh: A Perl script converting the .PG.v file (Verilog) to the .cdl file

top.tcl

- Step 1) Setup the target design and APR environments

```
set design_name top
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
##########################################################

set core_width 800
set core_height 540
```

top.tcl

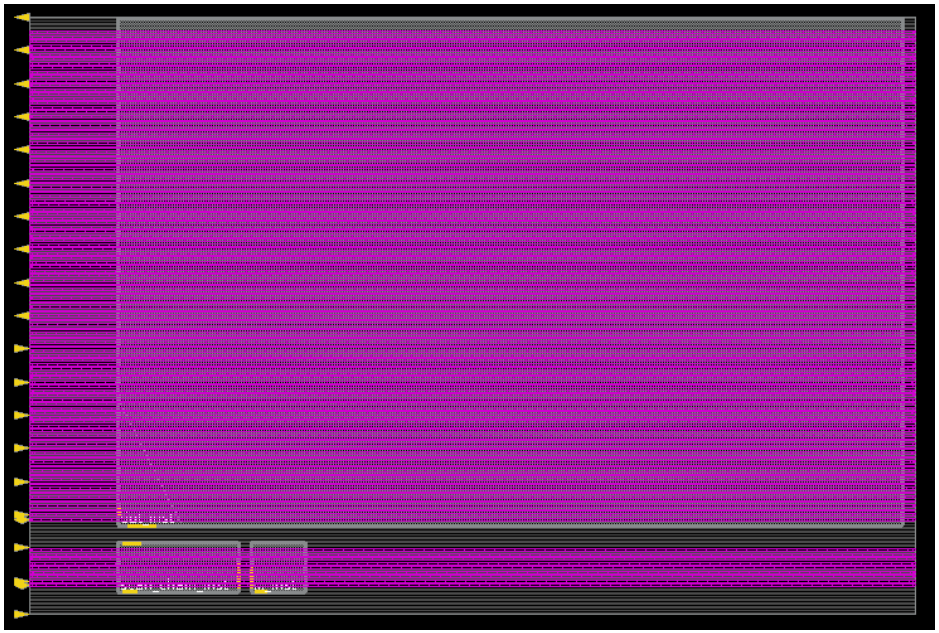
- Step 2) Create the floorplan and place blocks

```
floorPlan -s $core_width $core_height 0 0 0 0
redraw
fit

##################################################
# Place Blocks
##################################################

placeInstance dut_inst 80 80 R0 -fixed
placeInstance test_clkgen_inst 200 20 R0 -fixed
placeInstance scan_chain_inst 80 20 R0 -fixed

checkPlace
redraw
fit
```

top.tcl

- Step 3) Generate power rails

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

addStripe -block_ring_top_layer_limit M7 \
          -block_ring_bottom_layer_limit M7 \
          -padcore_ring_top_layer_limit M7 \
          -padcore_ring_bottom_layer_limit M7 \
          -max_same_layer_jog_length 6 \
          -merge_stripes_value 25 \
          -layer M7 \
          -set_to_set_distance 8 \
          -direction horizontal \
          -nets {VDD_TEST VSS} \
          -width 2 \
          -spacing 2 -area {0 25 10000 60}

addStripe -block_ring_top_layer_limit M7 \
          -block_ring_bottom_layer_limit M7 \
          -padcore_ring_top_layer_limit M7 \
          -padcore_ring_bottom_layer_limit M7 \
          -max_same_layer_jog_length 6 \
          -merge_stripes_value 25 \
          -layer M7 \
          -set_to_set_distance 8 \
          -direction horizontal \
          -nets {VDD VSS} \
          -width 2 \
          -spacing 2 -area {0 85 10000 530}
```

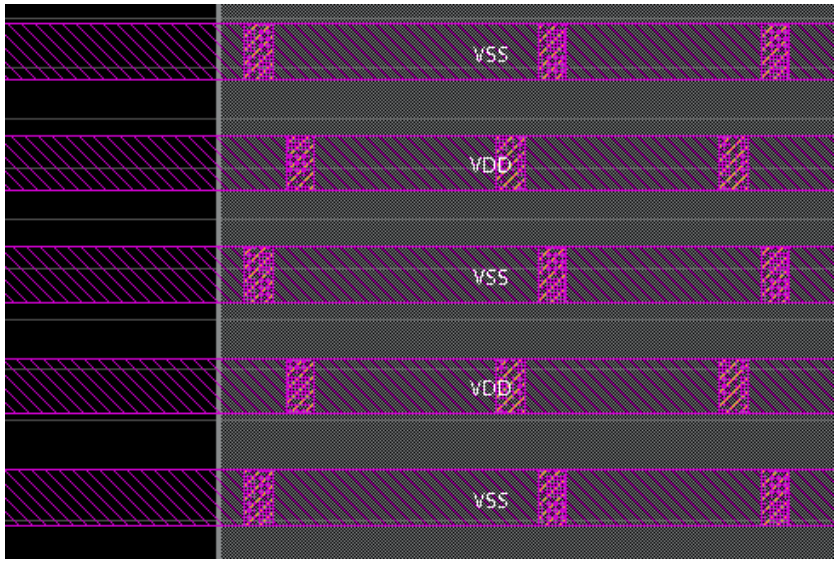


top.tcl

- Step 3) Generate power rails

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

addStripe -block_ring_top_layer_limit M7 \
          -block_ring_bottom_layer_limit M7 \
          -padcore_ring_top_layer_limit M7 \
          -padcore_ring_bottom_layer_limit M7 \
          -max_same_layer_jog_length 6 \
          -merge_stripes_value 25 \
          -layer M7 \
          -set_to_set_distance 8 \
          -direction horizontal \
          -nets {VDD_TEST VSS} \
          -width 2 \
          -spacing 2 -area {0 25 10000 60}

addStripe -block_ring_top_layer_limit M7 \
          -block_ring_bottom_layer_limit M7 \
          -padcore_ring_top_layer_limit M7 \
          -padcore_ring_bottom_layer_limit M7 \
          -max_same_layer_jog_length 6 \
          -merge_stripes_value 25 \
          -layer M7 \
          -set_to_set_distance 8 \
          -direction horizontal \
          -nets {VDD VSS} \
          -width 2 \
          -spacing 2 -area {0 85 10000 500}
```

Draw VDD and VSS alternately
They are connected to blocks' M3 rails



MQ layer (M7)

MV layer (M7)
(in the top.tcl, we use horizontal strips)

top.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"
```

- We skip
  - Step 5) Place design
  - Step 6) Do preCTS optimization
  - Step 7) Do clock tree synthesis (CTS)
  - Feel free to try later yourself

top.tcl

- Step 8) Route signals
  - Set routing options through ‘setNanoRouteMode’ command first
  - Do ‘globalDetailRoute’

```
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


globalDetailRoute
deleteAllRouteBlks
redraw
##################################################
```

top.tcl

- Step 9) Extract

```
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

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


##################################################
# Verify the design
##################################################

Puts "###############"
Puts "###"
Puts "### Verify ..."
Puts "###"
Puts "###############"

# Clear all design rule checking (DRC) markers in your design
clearDrc
verify_drc
verifyGeometry

# Detect conditions such as opens, unconnected wires, unconnected pins, loops, partial routing, and unrouted nets
# Generate violation markers in the design window
verifyConnectivity -type regular -error 1000 -warning 50

# Verify process antenna effect (PAE) and maximum floating area violations
verifyProcessAntenna
```

- We skip
  - Step 10) Add fillers. → but you must do it; w/o it, you cannot pass DRC

top.tcl

- Step 11) Verify
- Step 12) Generate outputs

```
# Report
report_power -leakage -cap -nworst -pg_pin -outfile "$design_name.power.rpt"

# Generate hierarchical design abstract (LEF) information for the current routed block-level design
#write_lef_abstract "$design_name.lef" -5.7 -PgpinLayers {5 6} -specifyTopLayer 6 -stripePin
write_lef_abstract "$design_name.lef" -5.7 -PgpinLayers 7 -specifyTopLayer 7 -stripePin

# Write the specified information to a DEF file
defOut -floorplan -netlist -routing "$design_name.final.def"

# Create a GDSII file of the current database
# Notes: '-mapFile' option specifies the file used for layer mapping
#        '-libName' option specifies the library to convert to GDSII Stream format
streamOut "$design_name.gds" -mapFile "../layermap/tsmc65_6350_spring24.layermap" -libName tcbn65gplus -structureName $design_name -units 1000 -mode ALL

# Write a netlist file of the design
# Notes: '-phys' option writes out physical cell instances, and inserts power and ground nets in the netlist
saveNetlist -phys -excludeLeafCell -excludeCellInst "FILL64 FILL32 FILL16 FILL8 FILL4 FILL2 FILL1" "$design_name.phy.v"
saveNetlist "$design_name.nophy.v"
saveNetlist -includePhysicalCell "DCAP64 DCAP32 DCAP16 DCAP8 DCAP4" -includePowerGround -excludeLeafCell -excludeCellInst "FILL64 FILL32 FILL16 FILL8 FILL4 FILL2 FI
LL1" "$design_name.PG.v"

# Extract RC information
extractRC -outfile "$design_name.cap"
rcOut -spef "$design_name.spef"

# Write delays to a Standard Delay Format (SDF) file
write_sdf -version 2.1 "$design_name.sdf"
write_sdf -version 2.1 -target_application verilog "$design_name.verilog.sdf"

# Report hold/setup violation
setAnalysisMode -checkType hold -useDetailRC true
report_timing -check_type hold -nworst 5 > "$design_name.hold.rpt"
setAnalysisMode -checkType setup -useDetailRC true
report_timing -check_type setup -nworst 5 > "$design_name.setup.rpt"
reportCapViolation -outfile final_cap.tarpt

# Run DRC and connection checks
verifyGeometry
verifyConnectivity -type all

# Report statistics for the entire design
summaryReport -outfile "$design_name.summary.rpt"

# Generate a file containing a list of nets which have critical slack of the currently specified timing analysis mode
reportCritNet -outfile "$design_name.critnet.rpt"

do_extract_model "$design_name.lib" -view typical
```

.cdl Generation

- When we generated the .cdl file after block-level APR, we use the standard cell .cdl file

makecdl.sh

```
#!/bin/bash

FN=top

v2lvs -i -v $FN.PG.v -o $FN.cdl \
-s /courses/ee6350/proj_2025Spring/ref/innovus/dut/dut.cdl \
-s /courses/ee6350/proj_2025Spring/ref/innovus/test_clkgen/test_clkgen.cdl \
-s /courses/ee6350/proj_2025Spring/ref/innovus/scan_chain/scan_chain.cdl \
-lsr /courses/ee6350/proj_2025Spring/ref/innovus/dut/dut.cdl \
-lsr /courses/ee6350/proj_2025Spring/ref/innovus/test_clkgen/test_clkgen.cdl \
-lsr /courses/ee6350/proj_2025Spring/ref/innovus/scan_chain/scan_chain.cdl

cp ${FN}.cdl ${FN}.sp
```

- Then, we use **the .cdl file including sub-block netlists** to generate the output .cdl file after top-level APR
- Now, we have top.cdl and top.sp (same)

Post-APR Verilog Simulation

- We need to do Verilog simulation to verify functionality
- qsim_apr/tb_chip

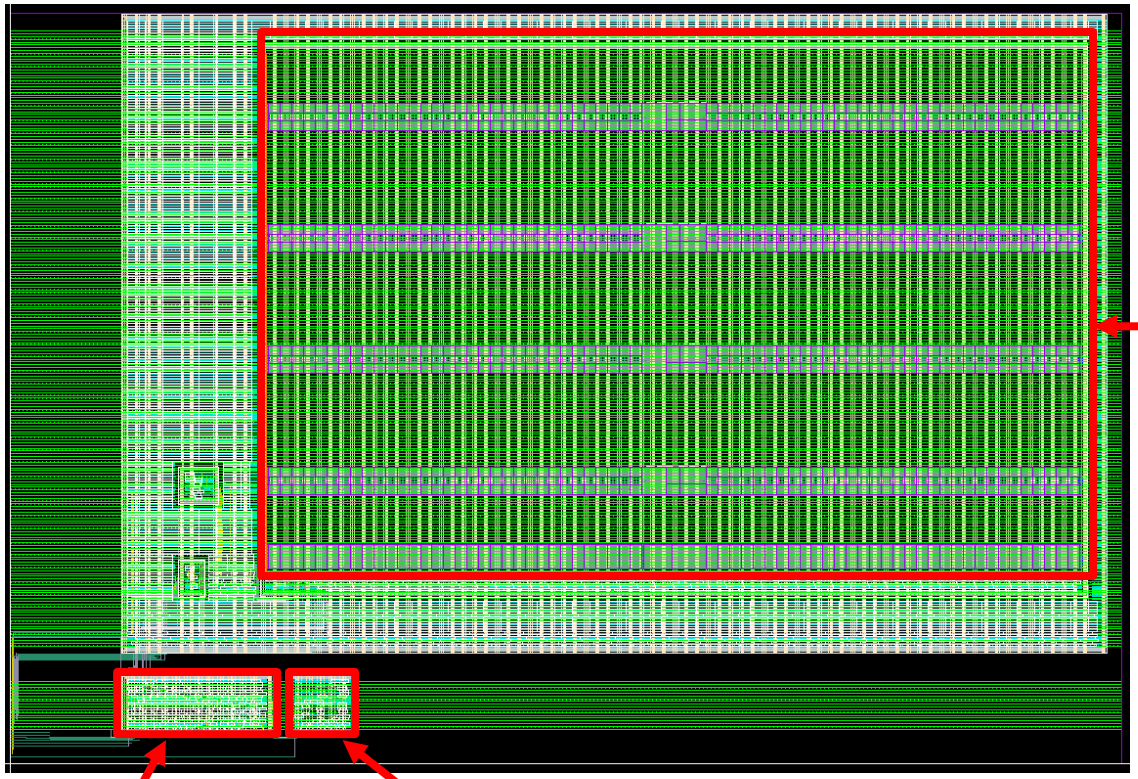
runsim.do

```
vlib work
vmap work work

# include standard cell verilog model
vlog +acc -incr /courses/ee6350/pdk2025/tcbn65gplus/TSMCHOME/digital/Front_End/verilog/tcbn65gplus_140b/tcbn65gplus_pwr.v
vlog +acc -incr ./verilog/tpfn65gpgv2od3_pwr.v

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

# include verilog modules
vlog +acc -incr ../../innovus/chip/chip.apr.v
    vlog +acc -incr ../../innovus/top/top.PG.v
        vlog +acc -incr ../../innovus/scan_chain/scan_chain.PG.v
        vlog +acc -incr ../../innovus/test_clkgen/test_clkgen.PG.v
        vlog +acc -incr ../../innovus/dut/dut.PG.v
            vlog +acc -incr ../../innovus/down_counter/down_counter.PG.v
            vlog +acc -incr ../../innovus/sram_controller/sram_controller.PG.v
            vlog +acc -incr ../../innovus/sram_wrapper/sram_wrapper.PG.v
            vlog +acc -incr +define+POWER_PINS ../../memory_compiler/sram00/sram00.v

# run simulation with sdf annotations and check waveforms
# here we suppress some timing checks in dut level sdf

vsim -voptargs=+acc -t ps -lib work \
-suppress 12088,12090,3262 \
-sdfmax testbench/chip_inst=../../innovus/chip/chip.verilog.sdf \
-sdfmax testbench/chip_inst/top_inst=../../innovus/top/top.verilog.sdf \
-sdfmax testbench/chip_inst/top_inst/scan_chain_inst=../../innovus/scan_chain/scan_chain.verilog.sdf \
-sdfmax testbench/chip_inst/top_inst/test_clkgen_inst=../../innovus/test_clkgen/test_clkgen.verilog.sdf \
-sdfmax testbench/chip_inst/top_inst/dut_inst=../../innovus/dut/dut.verilog.sdf \
-sdfmax testbench/chip_inst/top_inst/dut_inst/down_counter_inst=../../innovus/down_counter/down_counter.verilog.sdf \
-sdfmax testbench/chip_inst/top_inst/dut_inst/sram_controller_inst=../../innovus/sram_controller/sram_controller.verilog.sdf \
-sdfmax testbench/chip_inst/top_inst/dut_inst/sram_wrapper_inst=../../innovus/sram_wrapper/sram_wrapper.verilog.sdf \
testbench

do waveformat.do
run -all
```

.gds Import

- When importing .gds file at the Virtuoso, we need to use different 'Reference library list' (a.k.a. refLib.list)
- /ref/virtuoso/refLib/refLib_top.list →

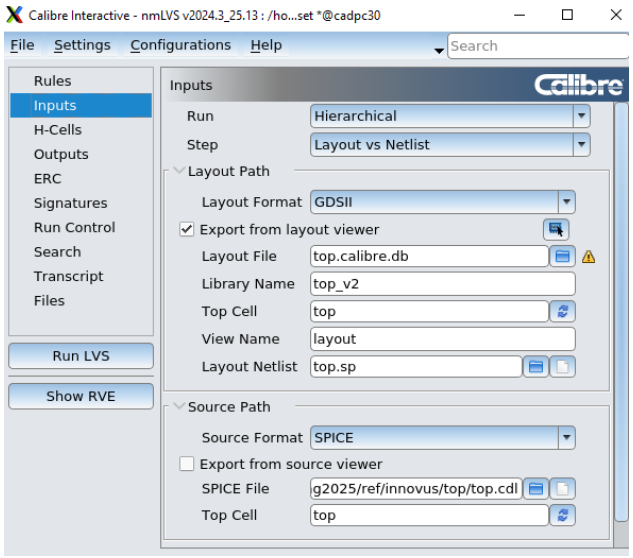
```
dut
test_clkgen
scan_chain
```



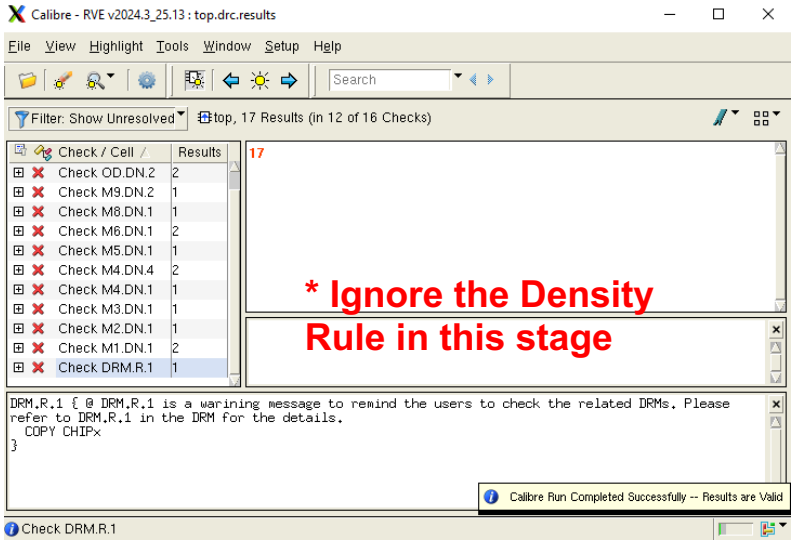
DRC/LVS Check

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

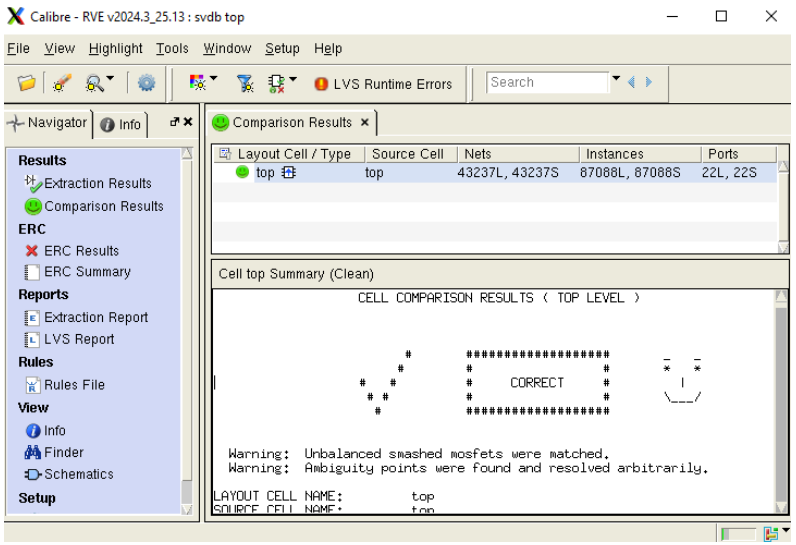
LVS Setting



DRC Clean



LVS Clean



End of the Slides