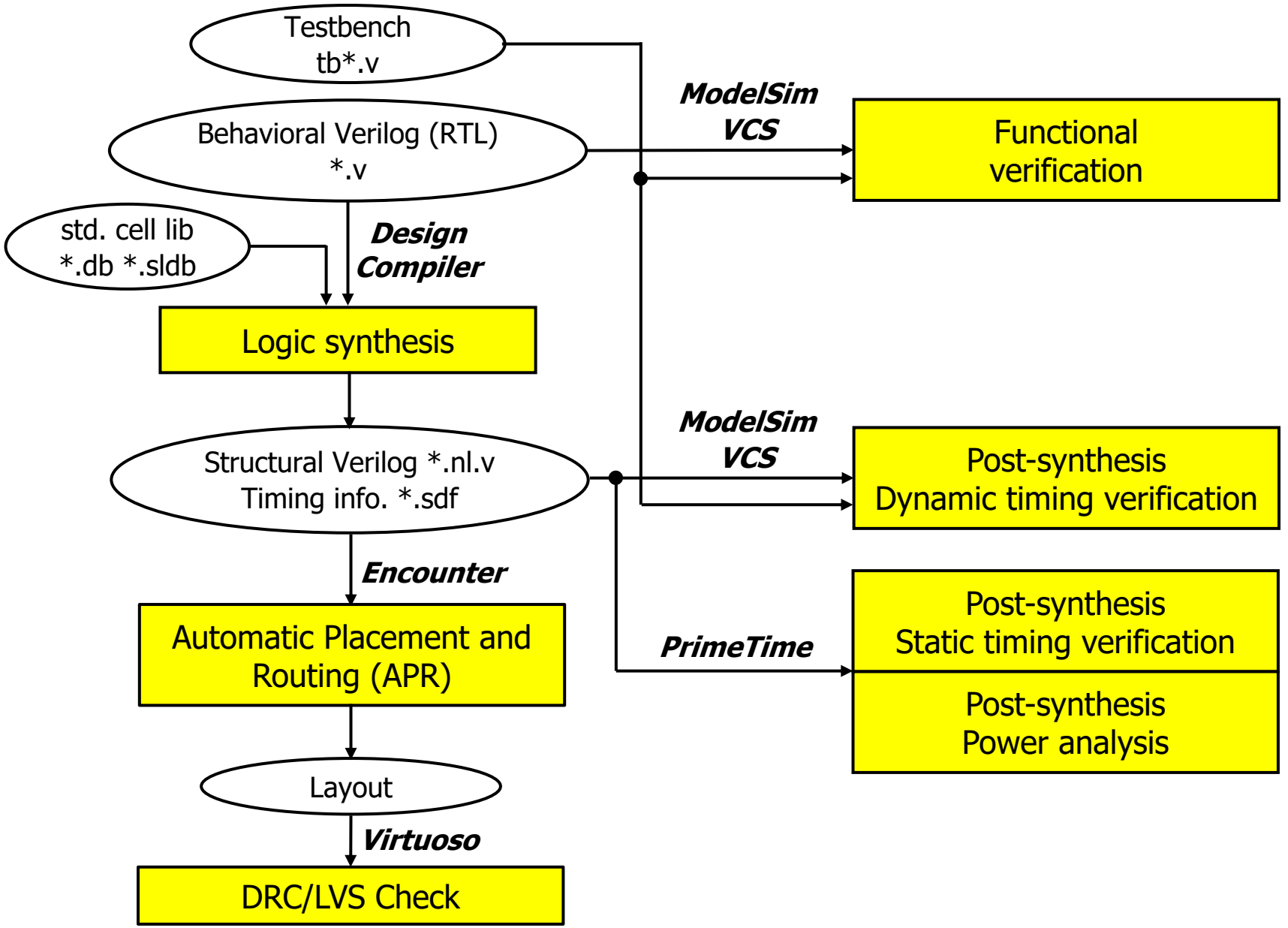
EECS E6321 Tutorial 03

Standard cells, Virtuoso, Abstract Gen, and Silicon Smart

Mingoo Seok & Previous TAs

Semi-Custom Design Flow: Block-level



Standard Cell Library

- Collection of low-level blocks
  - Fundamental logic blocks (INV, NAND, DFF…)
- Often Provided by the foundry and IP vendors

- Standard cell library includes
  - SPICE netlist (for circuit-level simulation)
  - Symbols (for schematics @ Virtuoso)
  - Layout information (for APR or custom physical design)
    - Physical layout (*.gds file) + abstract view (*.lef file)
  - Verilog modules (for dynamic simulation)
  - Timing/power information (for Synthesis or Timing/power analysis)
    - Library characterization (*.lib/*.db file)

- Visit - /courses/ee6350/pdk2025/tcbn65gplus

.lib and .db files

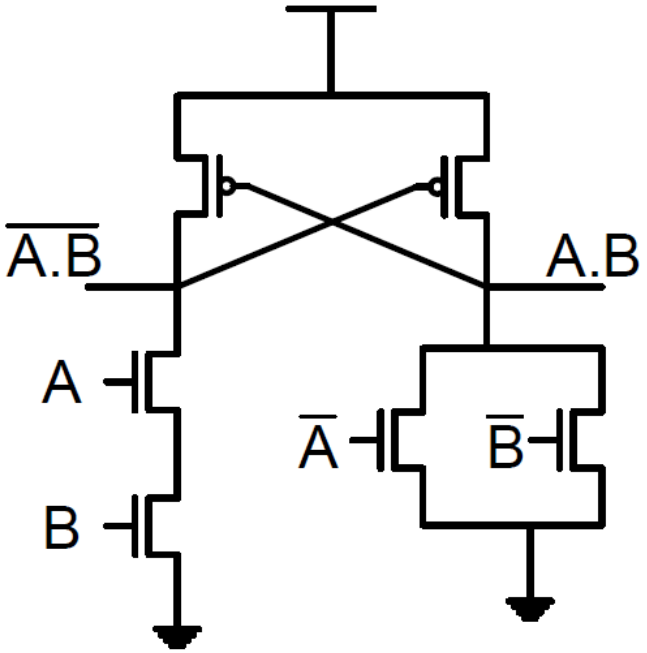
- Path: /courses/ee6350/pdk2025/tcbn65gplus/TSMCHOME/digital/Front_End/timing_power_noise/CCS/tcbn65gplus_200a
- Depending on conditions, different files are generated
- *.lib/*.db have same information
  - .lib is human readable
  - .db is not

```
ms4415@cadpc16 timing_power_noise]$ ls /courses/ee6350/pdk2025/tcbn65gplus/TSMCHOME/digital/Front_End/timing_power_noise/CCS/tcbn65gplus_200a
FIX_CCSN                        tcbn65gpluslt0d88_ccs.db        tcbn65gplusml1d11d1_ccs.lib     tcbn65gpluswc0d720d9_ccs.db     tcbn65gpluswcl0d90d72_ccs.lib
tcbn65gplusbc0d880d88_ccs.db    tcbn65gpluslt0d88_ccs.lib       tcbn65gplusml_ccs.db            tcbn65gpluswc0d720d9_ccs.lib    tcbn65gpluswcl0d90d9_ccs.db
tcbn65gplusbc0d880d88_ccs.lib   tcbn65gpluslt1d10d88_ccs.db     tcbn65gplusml_ccs.lib           tcbn65gpluswc0d72_ccs.db        tcbn65gpluswcl0d90d9_ccs.lib
tcbn65gplusbc0d881d1_ccs.db     tcbn65gpluslt1d10d88_ccs.lib    tcbn65gplustc0d80d8_ccs.db      tcbn65gpluswc0d72_ccs.lib       tcbn65gpluswcl_ccs.db
tcbn65gplusbc0d881d1_ccs.lib    tcbn65gpluslt1d11d1_ccs.db      tcbn65gplustc0d80d8_ccs.lib     tcbn65gpluswc0d90d72_ccs.db     tcbn65gpluswcl_ccs.lib
tcbn65gplusbc0d88_ccs.db        tcbn65gpluslt1d11d1_ccs.lib     tcbn65gplustc0d81d0_ccs.db      tcbn65gpluswc0d90d72_ccs.lib    tcbn65gpluswcz0d720d72_ccs.db
tcbn65gplusbc0d88_ccs.lib       tcbn65gpluslt_ccs.db            tcbn65gplustc0d81d0_ccs.lib     tcbn65gpluswc0d90d9_ccs.db      tcbn65gpluswcz0d720d72_ccs.lib
tcbn65gplusbc1d10d88_ccs.db     tcbn65gpluslt_ccs.lib           tcbn65gplustc0d8_ccs.db         tcbn65gpluswc0d90d9_ccs.lib     tcbn65gpluswcz0d720d9_ccs.db
tcbn65gplusbc1d10d88_ccs.lib    tcbn65gplusml0d880d88_ccs.db    tcbn65gplustc0d8_ccs.lib        tcbn65gpluswc_ccs.db            tcbn65gpluswcz0d720d9_ccs.lib
tcbn65gplusbc1d11d1_ccs.db      tcbn65gplusml0d880d88_ccs.lib   tcbn65gplustc1d00d8_ccs.db      tcbn65gpluswc_ccs.lib           tcbn65gpluswcz0d72_ccs.db
tcbn65gplusbc1d11d1_ccs.lib     tcbn65gplusml0d881d1_ccs.db     tcbn65gplustc1d00d8_ccs.lib     tcbn65gpluswcl0d720d72_ccs.db   tcbn65gpluswcz0d72_ccs.lib
tcbn65gplusbc_ccs.db            tcbn65gplusml0d881d1_ccs.lib    tcbn65gplustc1d01d0_ccs.db      tcbn65gpluswcl0d720d72_ccs.lib  tcbn65gpluswcz0d90d72_ccs.db
tcbn65gplusbc_ccs.lib           tcbn65gplusml0d88_ccs.db        tcbn65gplustc1d01d0_ccs.lib     tcbn65gpluswcl0d720d9_ccs.db    tcbn65gpluswcz0d90d72_ccs.lib
tcbn65gpluslt0d880d88_ccs.db    tcbn65gplusml0d88_ccs.lib       tcbn65gplustc_ccs.db            tcbn65gpluswcl0d720d9_ccs.lib   tcbn65gpluswcz0d90d9_ccs.db
tcbn65gpluslt0d880d88_ccs.lib   tcbn65gplusml1d10d88_ccs.db     tcbn65gplustc_ccs.lib           tcbn65gpluswcl0d72_ccs.db       tcbn65gpluswcz0d90d9_ccs.lib
tcbn65gpluslt0d881d1_ccs.db     tcbn65gplusml1d10d88_ccs.lib    tcbn65gpluswc0d720d72_ccs.db    tcbn65gpluswcl0d72_ccs.lib      tcbn65gpluswcz_ccs.db
tcbn65gpluslt0d881d1_ccs.lib    tcbn65gplusml1d11d1_ccs.db      tcbn65gpluswc0d720d72_ccs.lib   tcbn65gpluswcl0d90d72_ccs.db    tcbn65gpluswcz_ccs.lib
```

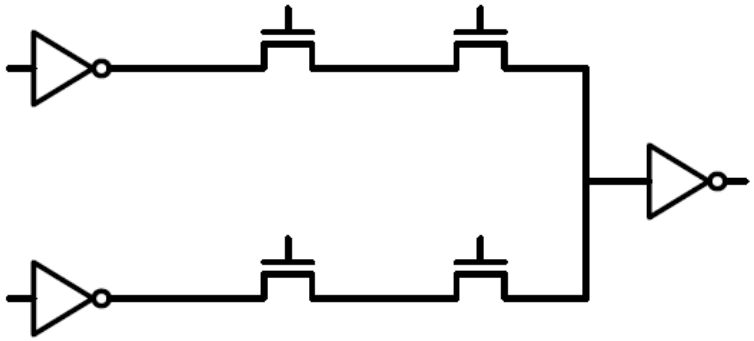
Logic Families

- How can we make digital circuits composed of different logic families?

Differential Cascode Voltage Switch (DCVS)

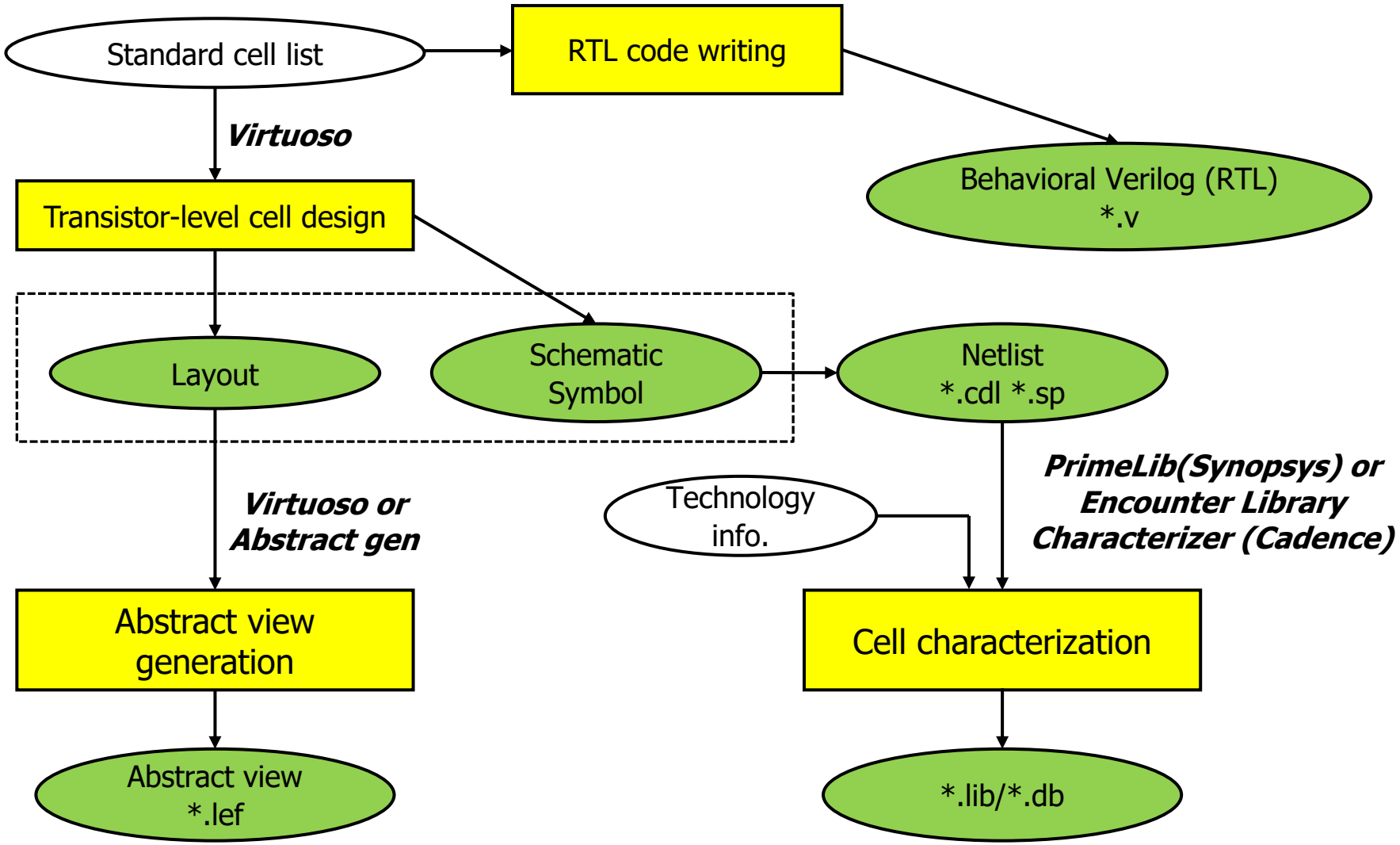


Pass Gate Logic



- We need to develop our own standard cells for logic synthesis and APR

Library Development Flow



EECS E6321 Tutorial 03

Standard cells, Virtuoso, Abstract Gen, and Silicon Smart

Mingoo Seok & Previous TAs

To Start with…

- Copy files from
  - /courses/ee6350/proj_2025Spring/ref/virtuoso
  - You don’t have to copy all subfolders, though

- Another tutorial is on the EECS4321 website
  - http://www.bioee.ee.columbia.edu/courses/cad/html

Cadence Virtuoso

- cds.lib includes the library

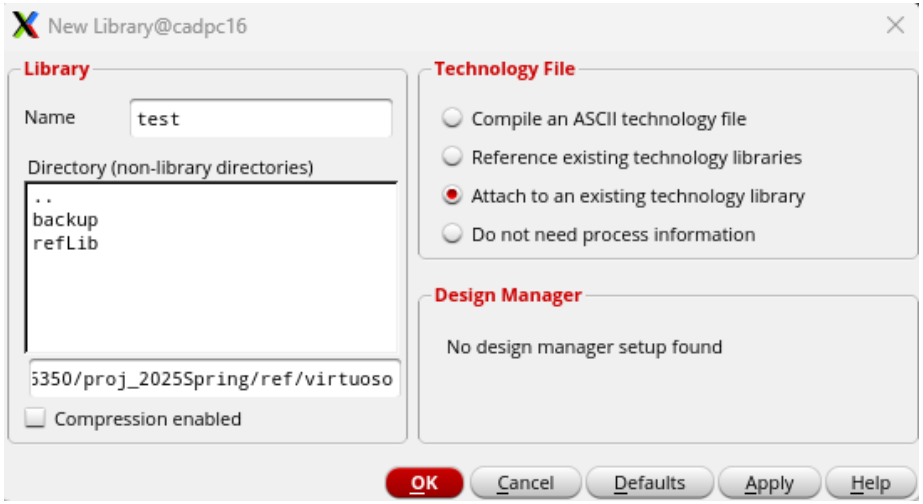
```
# Base libraries
INCLUDE /courses/ee6350/pdk2025/CMN65GP/T-N65-CM-SP-018-K3_RF_1p9m_6X1Z1U_ALRDL/cds.lib

# Standard cell library
DEFINE tcbn65gplus /courses/ee6350/pdk2025/tcbn65gplus/TSMCHOME/digital/Back_End/oa/tcbn65gplus

# Pad library
DEFINE tpfn65gpgv2od3_200d_mt_2_9lm /courses/ee6350/pdk2025/tpfn65gpgv2od3_200d/TSMCHOME/digital/Back_End/oa/tpfn65gpgv2od3_200d_mt_2_9lm

# Bonding pad library
# Note: this library is streamed in from the gds provided by the pdk because there is no oa (OpenAccess) for this in the pdk.
DEFINE tpbn65v_200a_cup_9M_6X1Z1U /courses/ee6350/proj_2025Spring/ref/virtuoso/tpbn65v_200a_cup_9M_6X1Z1U

# Seal ring library
DEFINE E6350_LARGE_SEALRING_2320X1110 /courses/ee6350/shared/E6350_LARGE_SEALRING_2320X1110

# Project libraries
DEFINE down_counter /courses/ee6350/proj_2025Spring/ref/virtuoso/down_counter
DEFINE sram_controller /courses/ee6350/proj_2025Spring/ref/virtuoso/sram_controller
DEFINE sram00 /courses/ee6350/proj_2025Spring/ref/virtuoso/sram00
DEFINE sram_wrapper /courses/ee6350/proj_2025Spring/ref/virtuoso/sram_wrapper
DEFINE dut /courses/ee6350/proj_2025Spring/ref/virtuoso/dut
DEFINE test_clkgen /courses/ee6350/proj_2025Spring/ref/virtuoso/test_clkgen
DEFINE scan_chain /courses/ee6350/proj_2025Spring/ref/virtuoso/scan_chain
DEFINE top /courses/ee6350/proj_2025Spring/ref/virtuoso/top
DEFINE chip /courses/ee6350/proj_2025Spring/ref/virtuoso/chip
DEFINE chip_streamed_in /courses/ee6350/proj_2025Spring/ref/virtuoso/chip_streamed_in
DEFINE chip_w_sealring_streamed_in /courses/ee6350/proj_2025Spring/ref/virtuoso/chip_w_sealring_streamed_in
DEFINE DM_chip_w_sealring /courses/ee6350/proj_2025Spring/ref/virtuoso/DM_chip_w_sealring
DEFINE DODPO_chip_w_sealring /courses/ee6350/proj_2025Spring/ref/virtuoso/DODPO_chip_w_sealring
DEFINE chip_w_sealring_w_dummy_streamed_in /courses/ee6350/proj_2025Spring/ref/virtuoso/chip_w_sealring_w_dummy_streamed_in
```

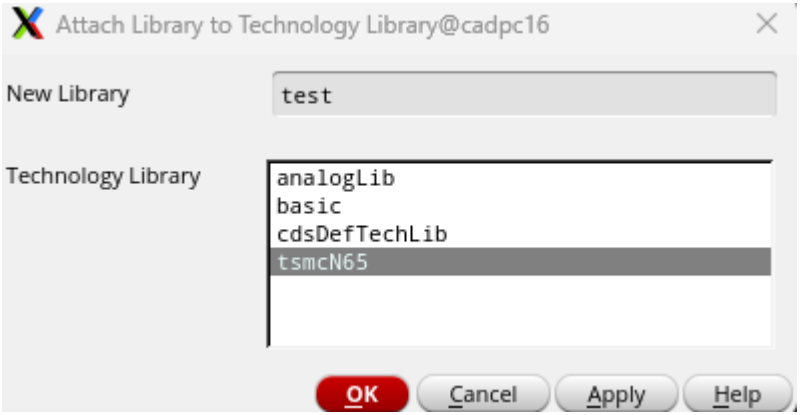
```
DEFINE cdsDefTechLib $CDSHOME/tools/dfII/etc/cdsDefTechLib
DEFINE basic $CDSHOME/tools/dfII/etc/cdslib/basic
DEFINE analogLib $CDSHOME/tools/dfII/etc/cdslib/artist/analogLib
DEFINE tsmcN65 ./tsmcN65
```

- .cdsenv, .cdsinit contain the environmental setting for the virtuoso tool
- run_virtuoso.source.sh

Make a New Library

1. Select CIW → File → New → Library
2. Put your own library name
3. Choose 'Attach to an existing technology library' and click 'OK' button
4. Choose 'tsmcN65' at 'Attach Library to Technology Library' window

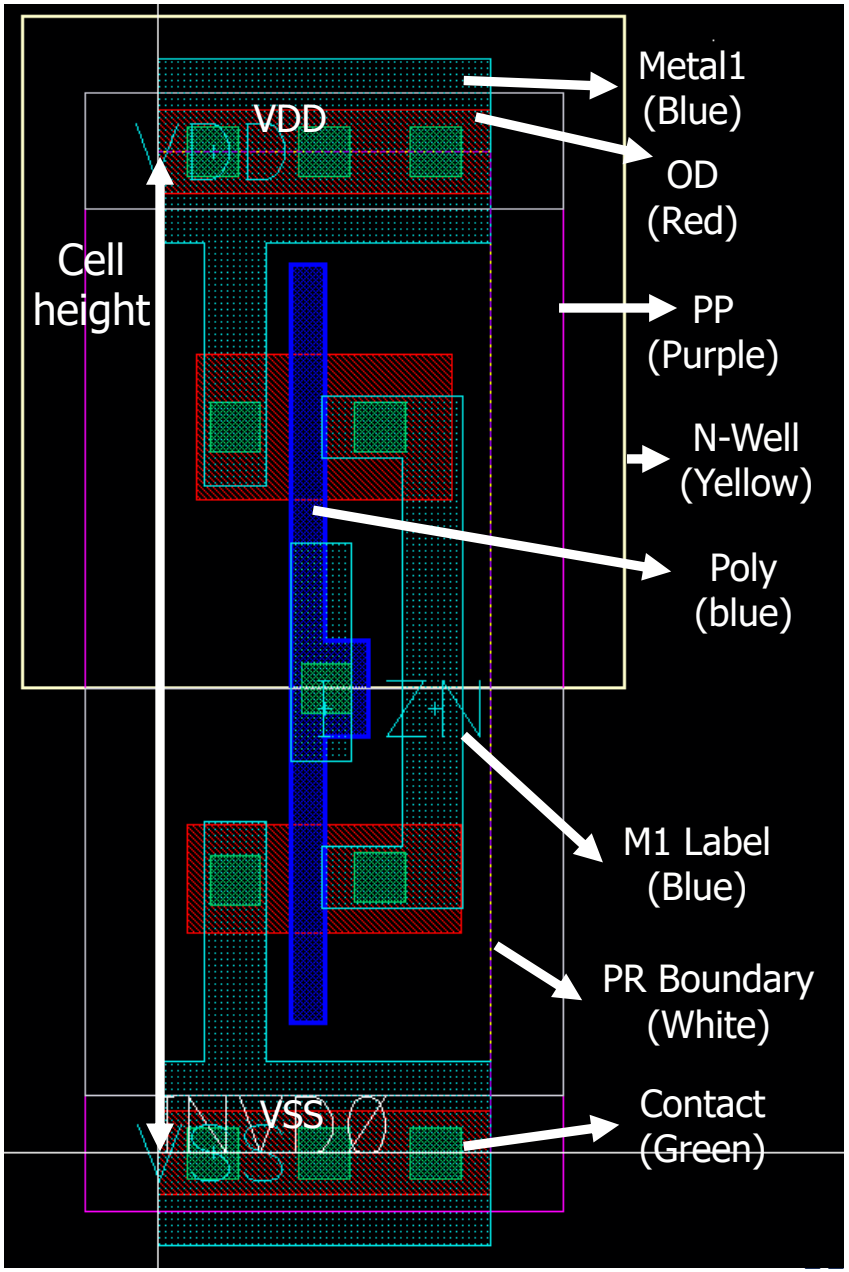




Inverter Layout

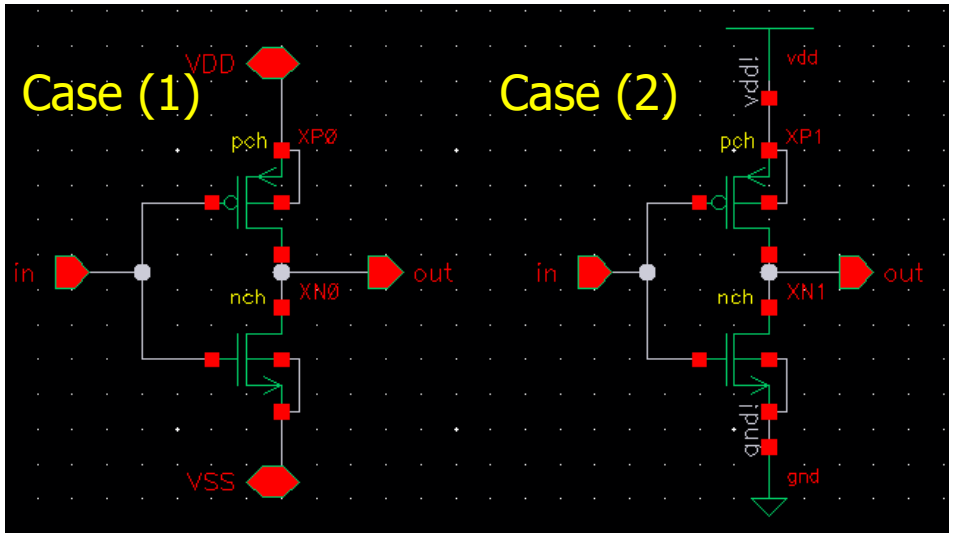
- Please copy the 'INVD0' cell to your library
  - Cell position: /courses/ee6350/pdk2025/tcbn65gplus/TSMCHOME/digital/Back_End/oa/tcbn65gplus
- pch: 260n/60n, nch: 195n/60n
- Layer description

Layer	Color	Description
OD	Red	Active area (gate oxide and N+/P+ diffusion regions)
PO	Blue	Polysilicon line
NW	Yellow	N-Well
PP	Purple	P+ select (defines PMOS and p+ substrate contacts)
CO	Green	Metal contact
Mx	-	Metal layer (x = 1 - 9)
Vx	-	Via (x = 1 - 8)
PR Boundary	White	Boundary for placement & route



When Drawing Schematic

- When you set the power/gnd pin your schematic, you can use (1)inout pins or (2)vdd/gnd cells in 'analogLib'
  - In case (2), the net names are '**vdd!**', and '**gnd!**'
  - If the exclamation mark(**!**) is used as the part of the net name, this net is regarded as the global net
- If you want to know about global nets in detail, please see 'Virtuoso_Inherited_Connections_Tutorial' in CourseWorks



When Drawing Layout

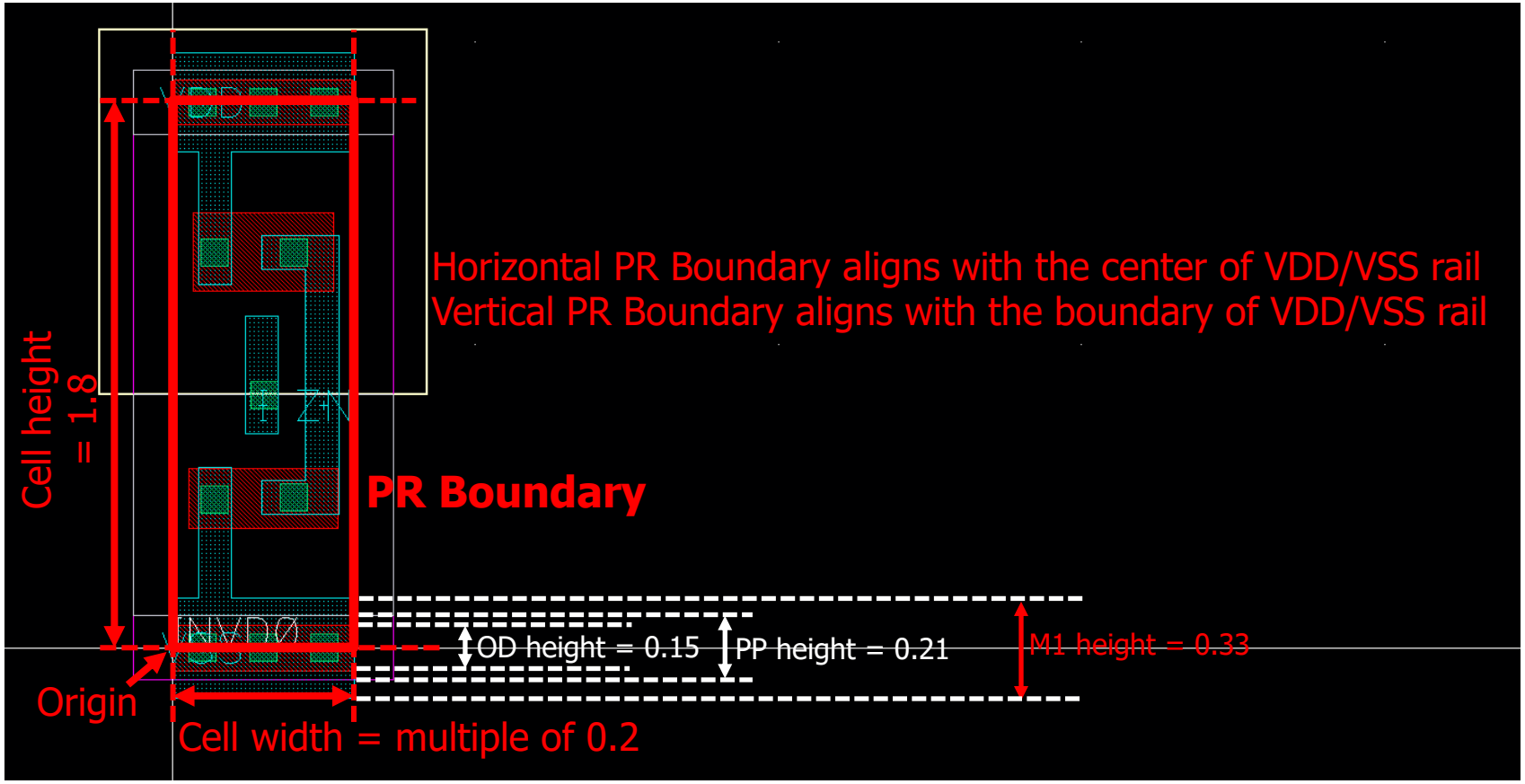
- If you draw the layout of the cell which is not used as a standard cell, you don't need to draw 'PR Boundary' layer

- For LVS clean, you should create a label for the in/out nets
  - 'Label' and 'Pin' are different
  - 'Pin's used for **creating abstract view**

- Recommend that pin boundary should be included in layer drawing

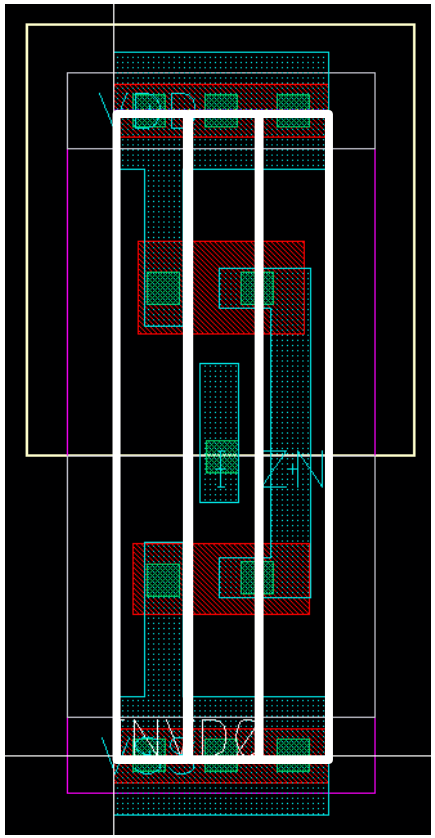
Because We Draw the Standard Cell…

- There exists more strict rules for our layout
  - Cell width/height, VDD/VSS rail
  - In addition, it is better to consider the position of other layers
- Need to draw 'PR Boundary' layer precisely

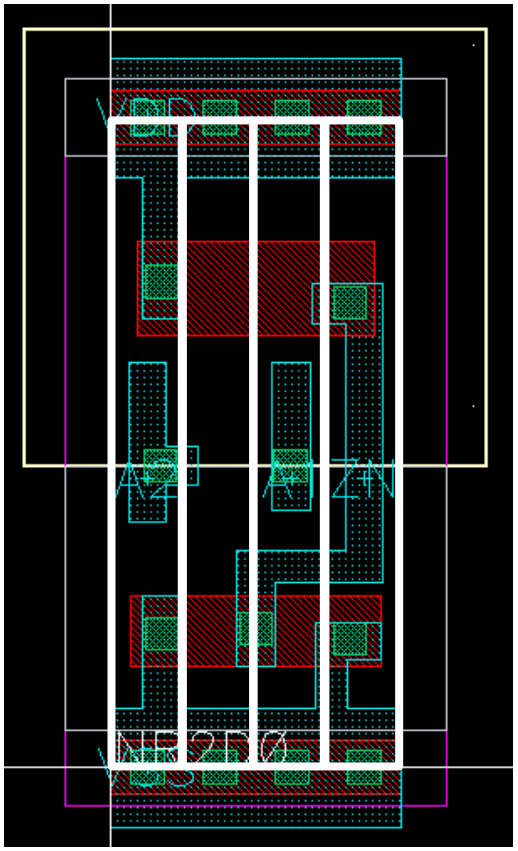


Standard Cell Layout

- We need to change the contact position to avoid DRC violation
- (TSMC 65nm) The contact should be located at center of every cell unit (0.2 μm × 1.8 μm)'s power rails



INVD0

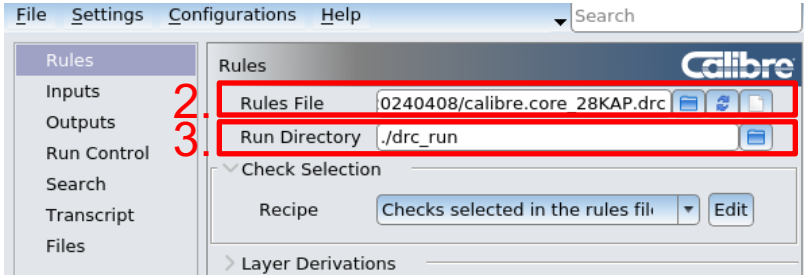


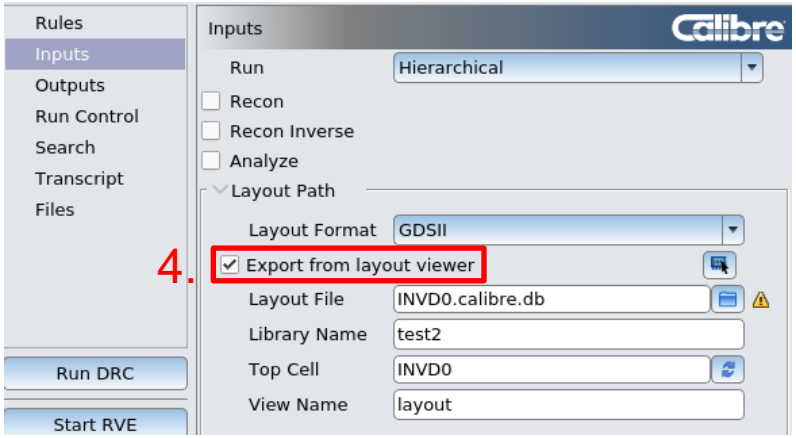
NR2D0

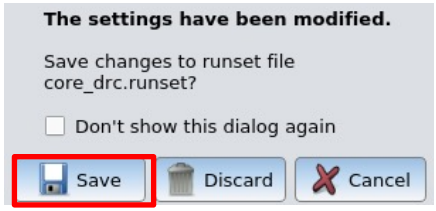
DRC Check

1. In the layout view, select Calibre → nmDRC
2. When the DRC window opens up, choose the right 'Rules file'
   - Sample located at /courses/ee6350/shared/calibre/DRC_20240408 /calibre.core_28KAP.drc
3. Then, choose 'Run Directory' as (Virtuoso directory)/drc_run @ 'Rules' tab
   - If you leave it as your Virtuoso directory, the directory will be dirty due to DRC dump files
4. At 'Inputs' → 'Layout Path' tab, check 'Export from layout viewer'
5. Click 'Run DRC' button

*After running whole process, we can save this runset file (don't need to repeat 2-4)





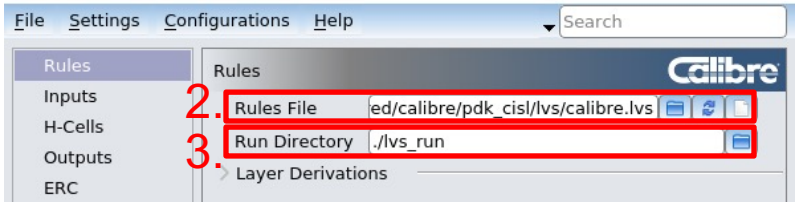


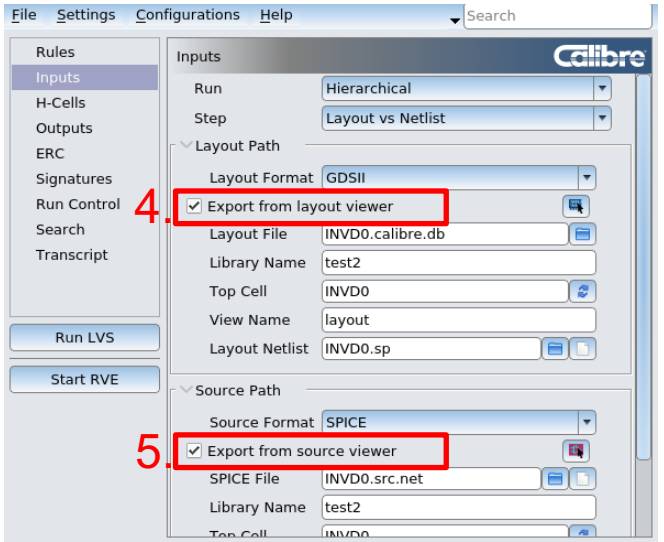
LVS Check

1. In the layout view, select Calibre → nmLVS
2. When the LVS window opens up, choose the right 'Rules file'
   - Sample located at:

   /courses/ee6350/shared/calibre/pdk_cisl/lvs/calibre.lvs
3. Then, choose 'Run Directory' as (Virtuoso directory)/lvs_run @ 'Rules' tab
4. At 'Inputs' → 'Layout Path' tab, check 'Export from layout viewer'
5. At 'Inputs' → 'Source Path' tab, check 'Export from source viewer'
6. Click 'Run LVS' button

*After running whole process, we can save this runset file (don't need to repeat 2-5)

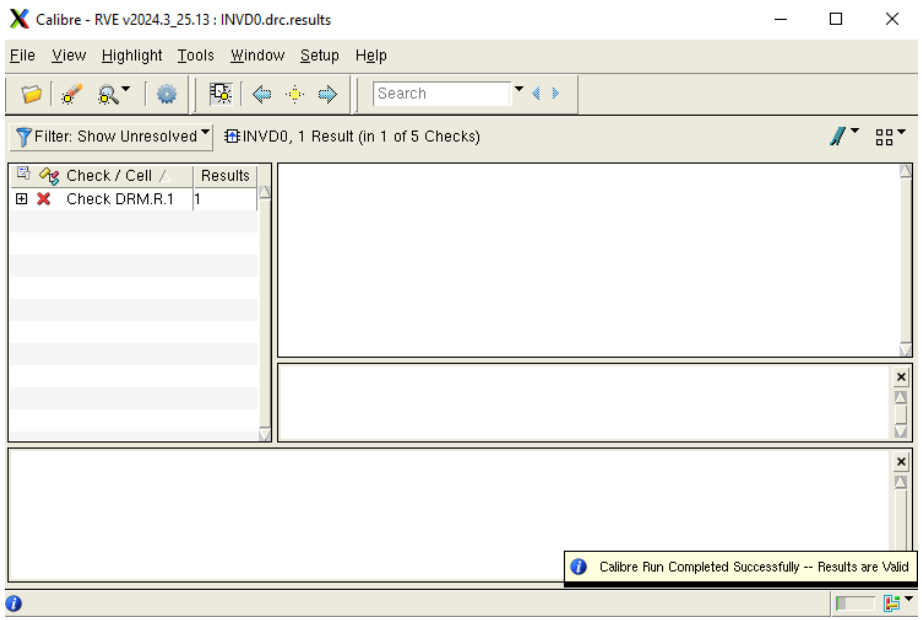




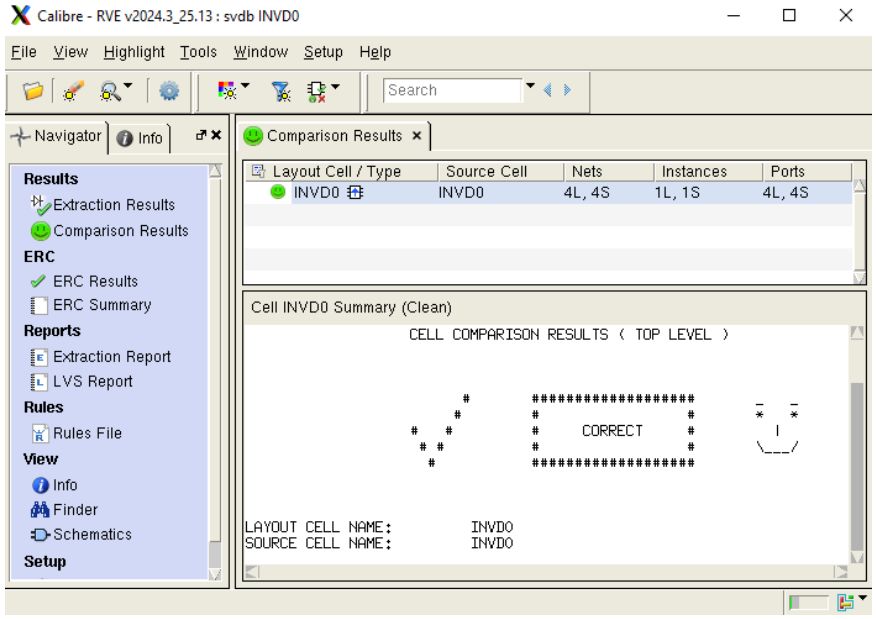
DRC/LVS Notes

- Please save your DRC/LVS runset files in your Virtuoso directory (e.g. runset.block, runset.top)
- When you re-run DRC/LVS, you can select Calibre → Run nmDRC/LVS, and use the saved runset files
- However, if you newly open Virtuoso, you have to configure the settings using the GUI

DRC Clean



LVS Clean



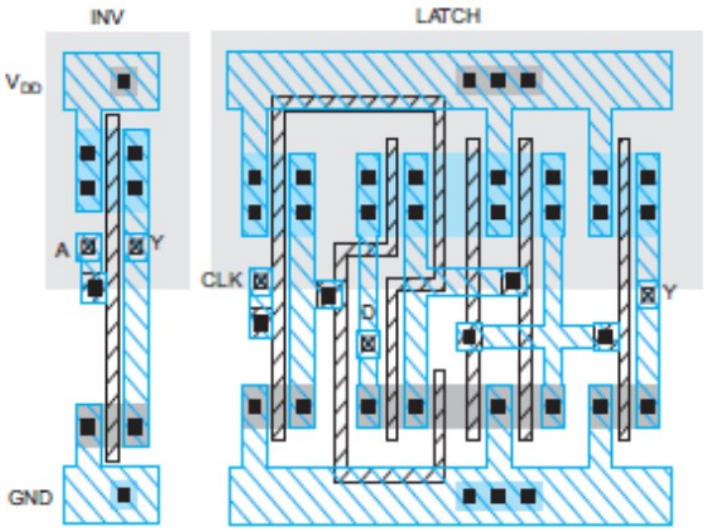
EECS E6321 Tutorial 03

Standard cells, Virtuoso, Abstract Gen, and Silicon Smart

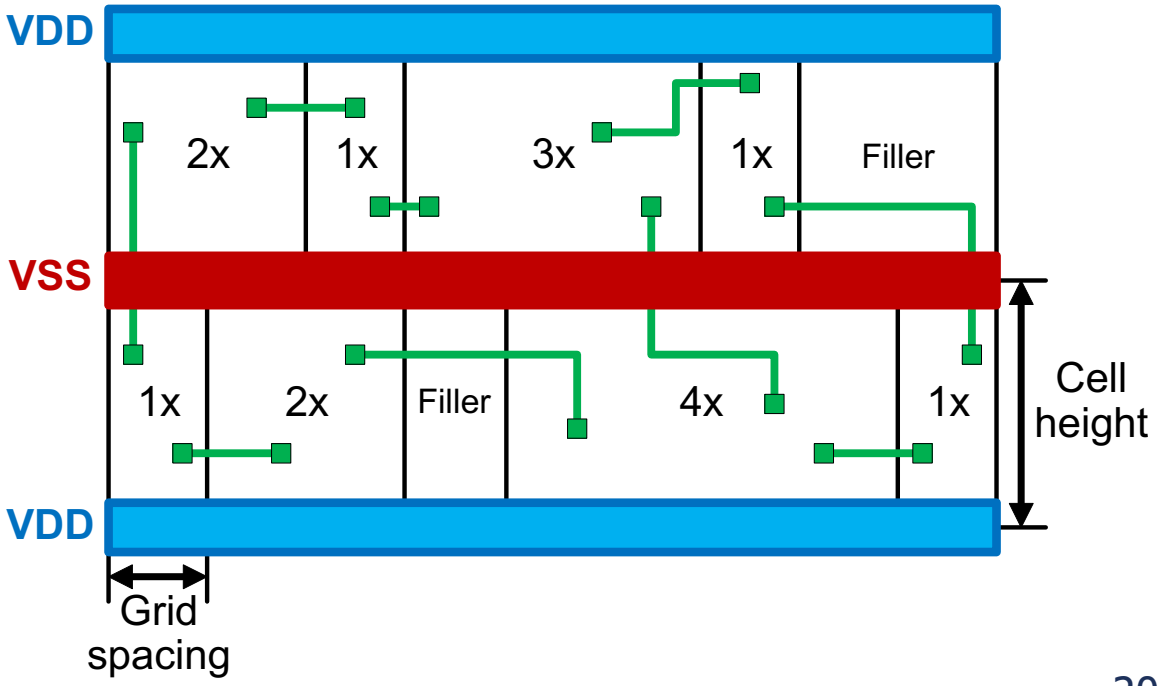
Mingoo Seok & Previous TAs

Standard Cell Design Rule

- All cells have the same height
  - Horizontal VDD/VSS rails' height should be the same as well
- The cell width must be a multiple of the grid spacing
- Filler cells should be included in your standard cell library
  - They provide continuity for your VDD/VSS rails, as well as for N-Well



Placing and Routing



Abstract Generator

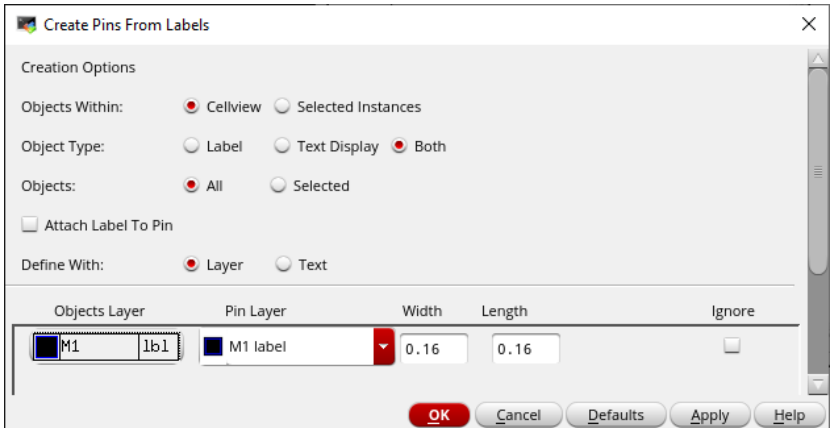
- Abstracts are simpler representations of the standard cells
- Abstracts include information that is pertinent to the place-and-route-tools only
  - i.e., **metal** and **via** layers
- After creating abstract view, the *.lef file should be produced for the following steps
  - The output *.lef file contains the physical description of cells (dimension, location of pins and metals)
  - It can be fed into Automatic Place and Route (APR) such as Cadence Innovus

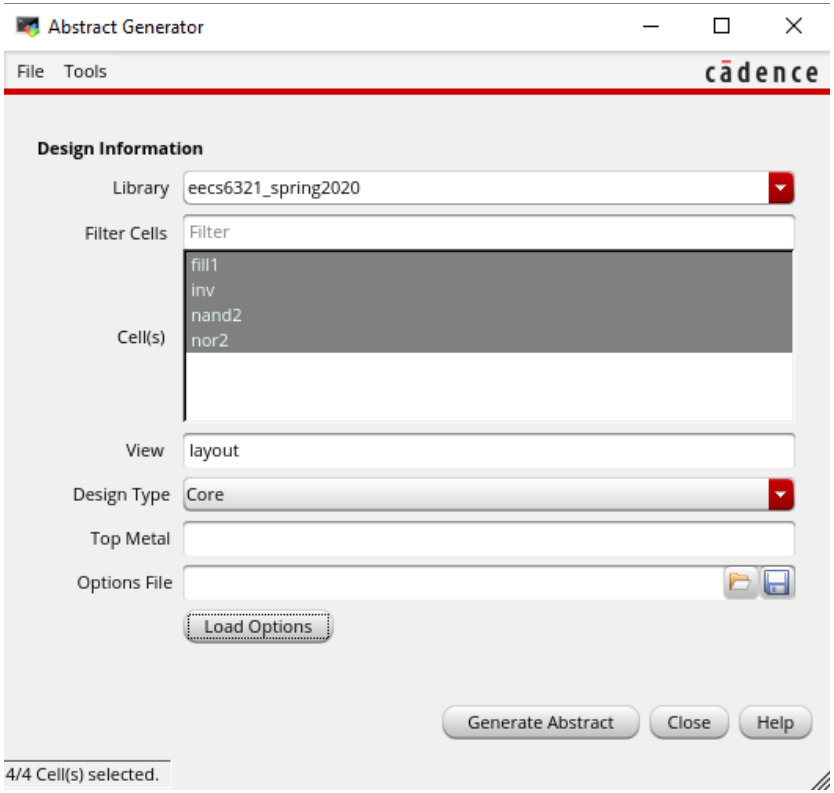
- **Before creating abstract views** @ Layout window, select Edit → Advanced → Move origin, and change the origin as the bottom-left corner of the **OUTLINE**

How to Create Abstract Views

@ Virtuoso

1. At layout window, select Tools → Create Pins From Labels
   - If you already created the pin while drawing layout, skip step 1)
   - **Create pins only from 'Metal label' layers / ignore other layers**
2. Set pins' I/O type (Select the pin and press 'Q')
3. Select CIW → Tools → Abstract Generator
4. Change the settings as below
   - Library: Your library
   - Cell: Your cell
   - View: layout
   - Design Type: Core
5. Click 'Load Options' button
6. Click 'Generate Abstract' button

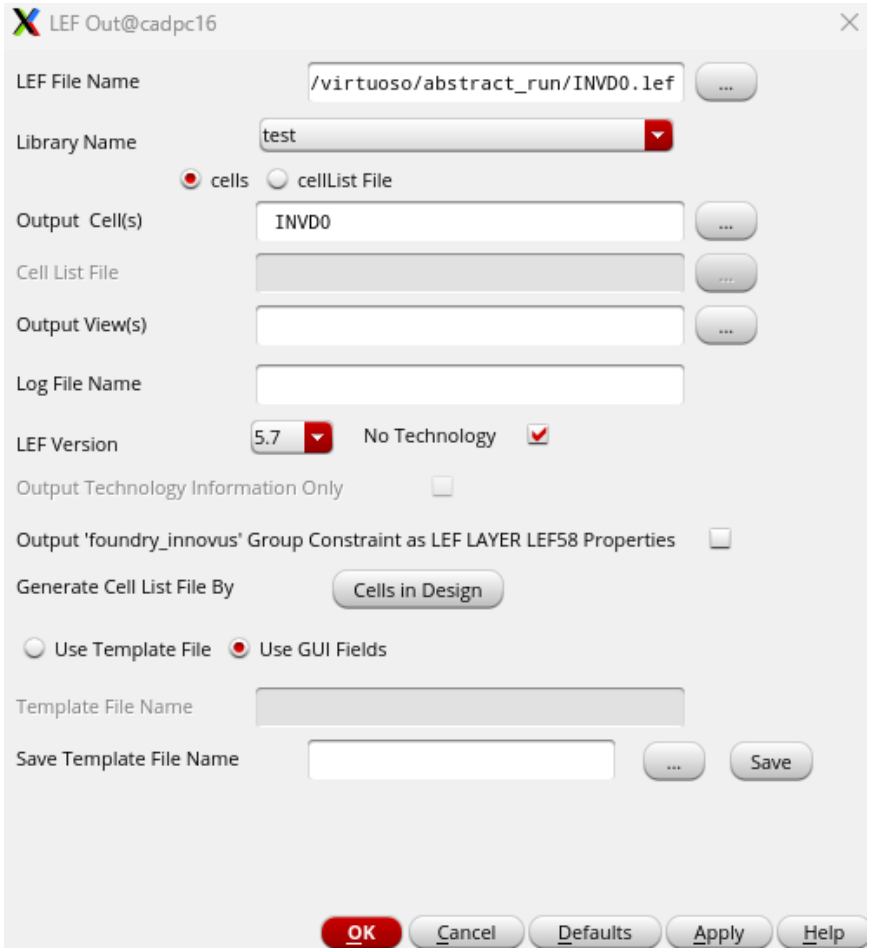




How to Export LEF File

1. Select CIW → File → Export → LEF
2. Change the settings as below
   - LEF File Name: Up to you
   - Better to make a sub-directory (e.g., abstract_run) and save *.lef files in this directory
   - Library Name: Your library
   - Output Cell: Your cell (You can select multiple cells)
   - Output View: abstract
3. Check 'No Technology'
4. Click 'OK' button

- With our 'abstract.options' setting, layer OUTLINE defines the cell area
- <u>When you do the assignment, please select multiple output cells and generate one .lef file including the information of all cells</u>



Generated LEF File

- MACRO includes cell descriptions, cell dimensions, layout of pins and blockages
- **When you draw 'PR boundry' layer properly, you can achieve the correct ORIGIN and SIZE information**
- If needed, you can always make some edits on the file

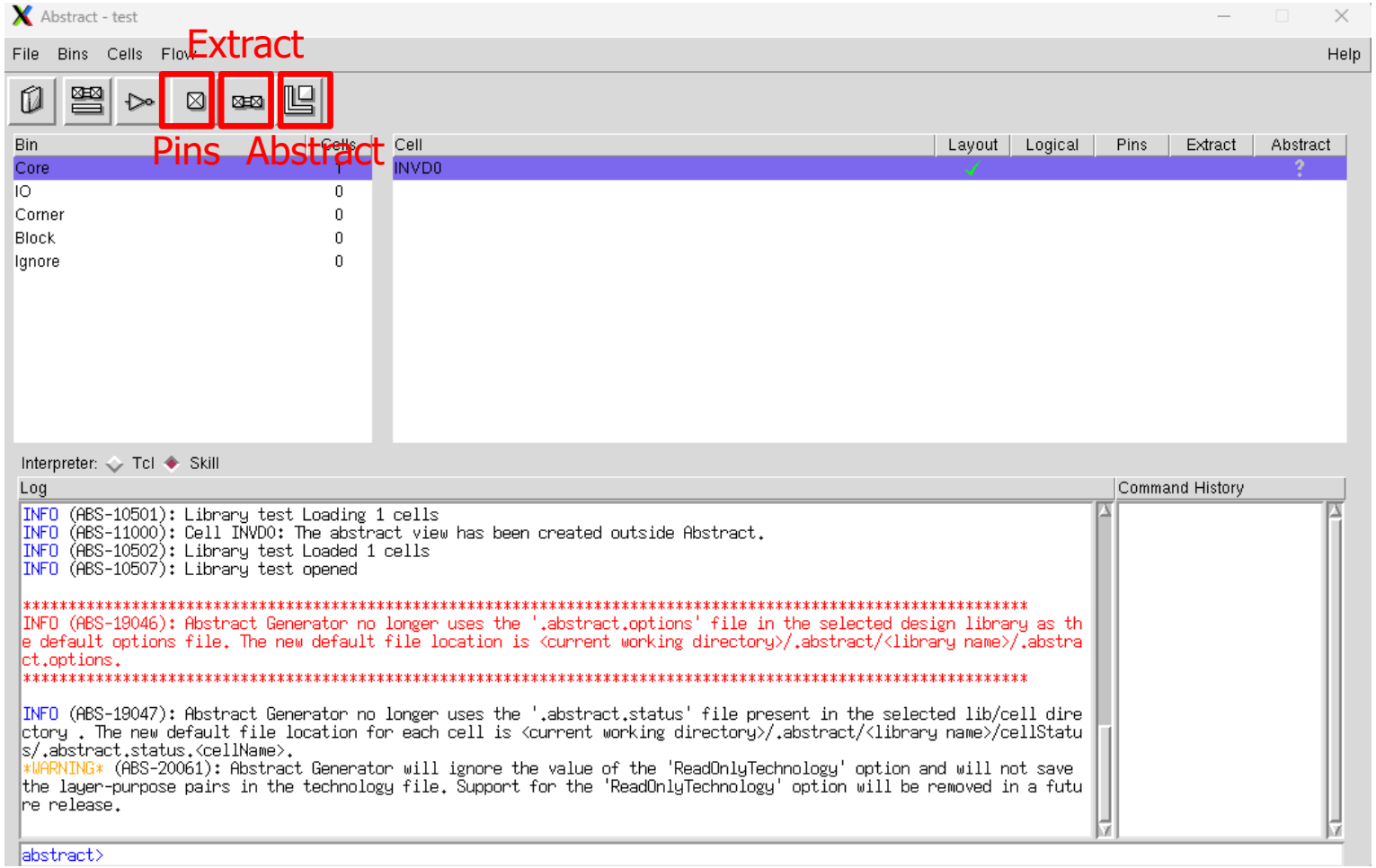
```
VERSION 5.7 ;
BUSBITCHARS "[]" ;
DIVIDERCHAR "/" ;

MACRO INVD0                                  Size
  CLASS CORE ;
  ORIGIN 0 0 ;
  FOREIGN INVD0 0 0 ;
  SIZE 0.6 BY 1.8 ;
  SYMMETRY X Y ;
  SITE CoreSite ;
  PIN VSS                                    Pin info.
    DIRECTION INOUT ;
    USE GROUND ;
    SHAPE ABUTMENT ;
    PORT
      LAYER M1 ;
        RECT 0 -0.165 0.6 0.165 ;
        RECT 0.085 -0.165 0.195 0.595 ;
    END
  END VSS
  PIN VDD
```

```
  PIN VDD
    DIRECTION INOUT ;
    USE POWER ;
    SHAPE ABUTMENT ;
    PORT
      LAYER M1 ;
        RECT 0 1.635 0.6 1.965 ;
        RECT 0.085 1.2 0.195 1.965 ;
    END
  END VDD
  PIN ZN
    DIRECTION OUTPUT ;
    USE SIGNAL ;
    PORT
      LAYER M1 ;
        RECT 0.295 1.25 0.55 1.36 ;
        RECT 0.44 0.44 0.55 1.36 ;
        RECT 0.295 0.44 0.55 0.55 ;
    END
  END ZN
  PIN I
    DIRECTION INPUT ;
    USE SIGNAL ;
    PORT
      LAYER M1 ;
        RECT 0.24 0.705 0.35 1.095 ;
    END
  END I
END INVD0

END LIBRARY        No OBS layers
```

How to Create Abstract Views - Alternative

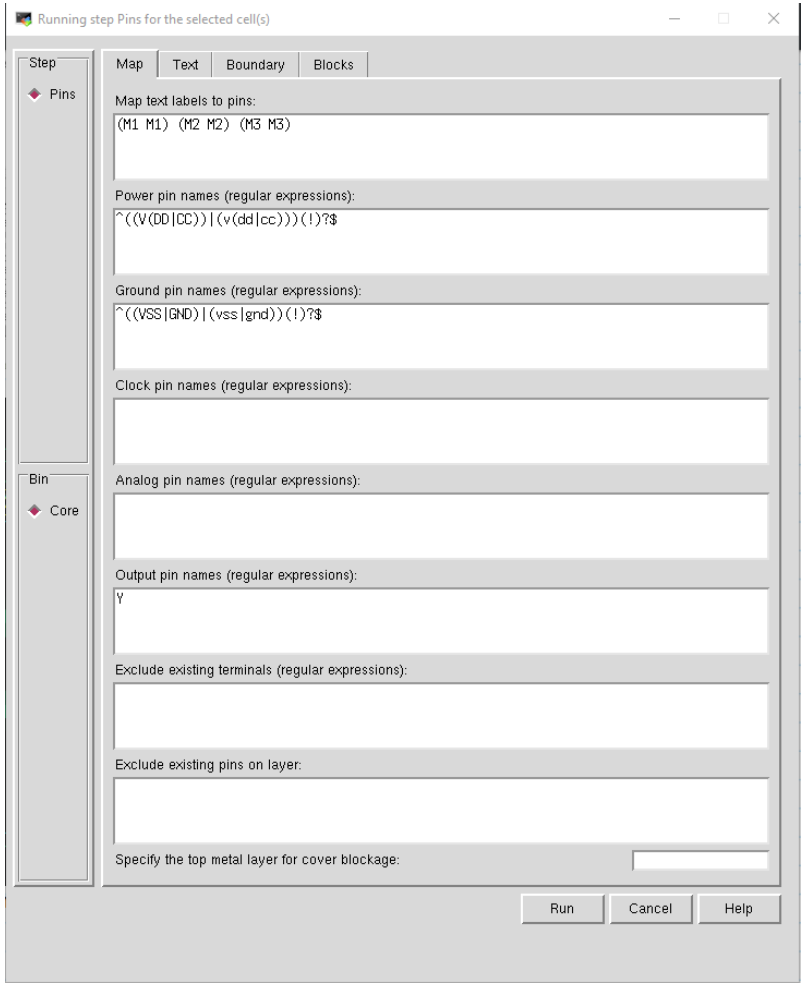
@ Abstract Gen

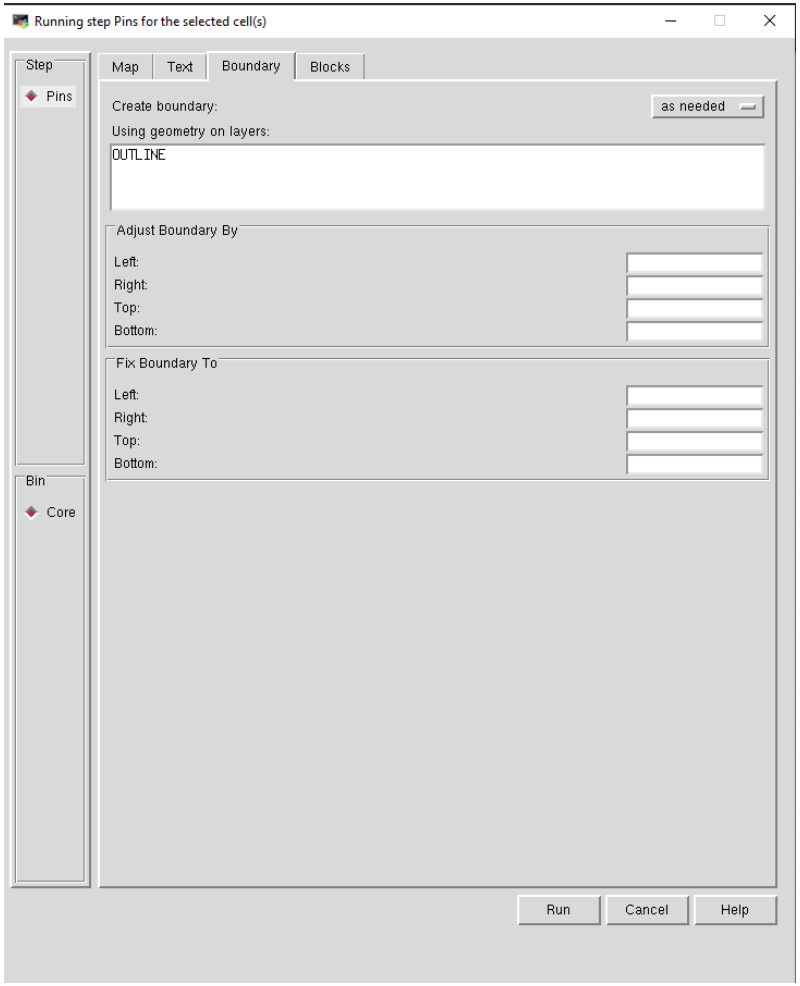
1. Type command $abstract at the terminal
2. Select File → Library → Open, and choose your library
3. Choose cells, select Cells → Move, and move then to 'Core'



How to Create Abstract Views - Alternative

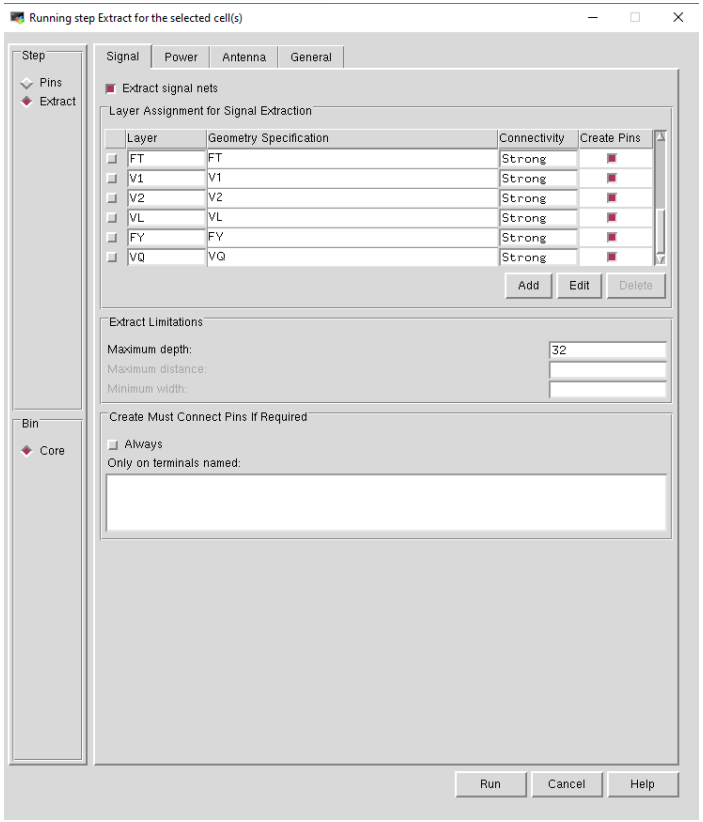
4. Click ‘Pins’ button, and change the settings at ‘Map’ tab
5. At ‘Boundary’ tab, put ‘OUTLINE’ in ‘Using geometry on layers’
6. Click ‘Run’ button

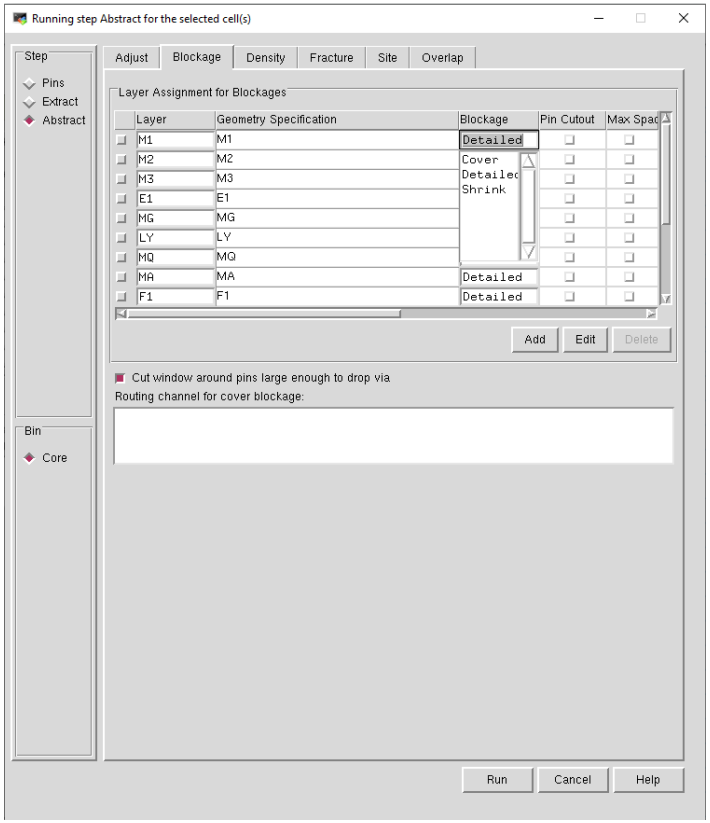




How to Create Abstract Views - Alternative

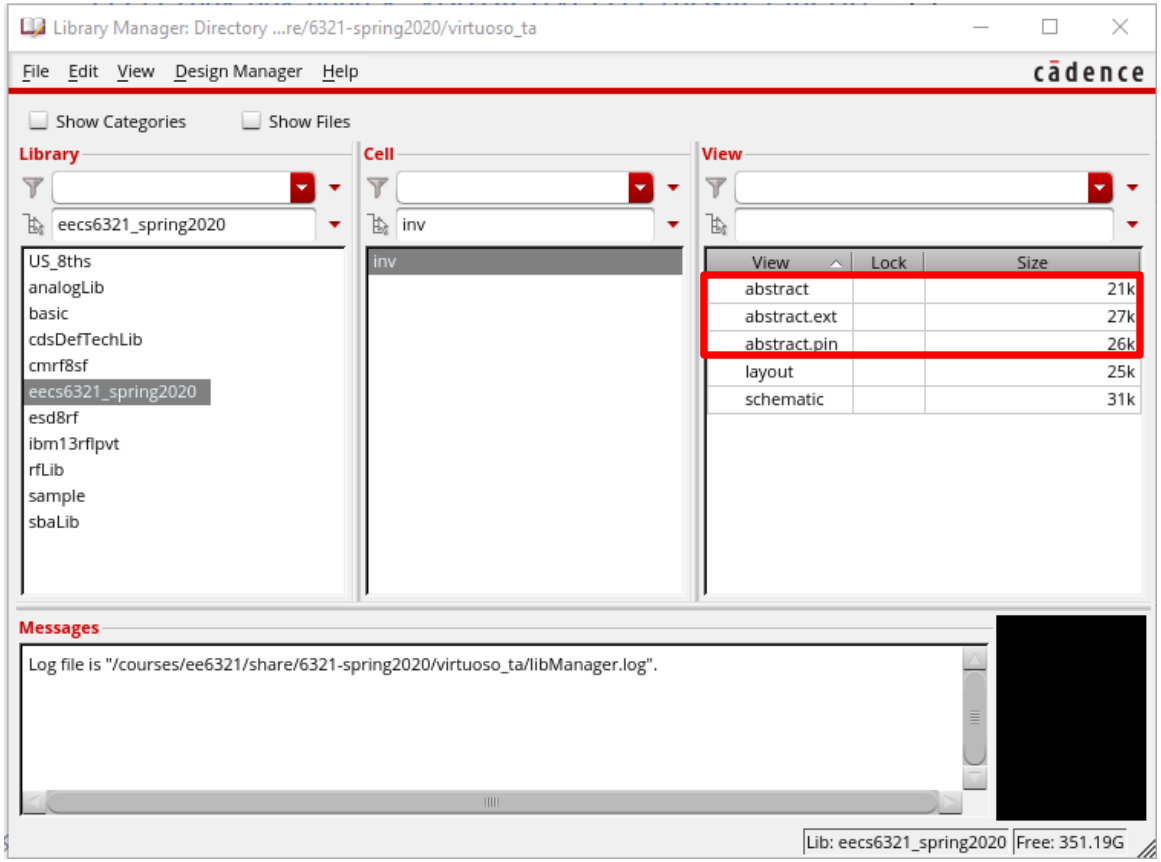
7. Click ‘Extract’ button, and ‘Run’ button again in the new window
8. Click ‘Abstract’ button, and you can use either ‘Detailed’ or ‘Cover’ option at ‘Blockage’ tab
   - Detailed: Abstract knows about each metal paths in a block so they detour them in detail
   - Cover: Metal paths detour entire block
9. Click ‘Run’ button

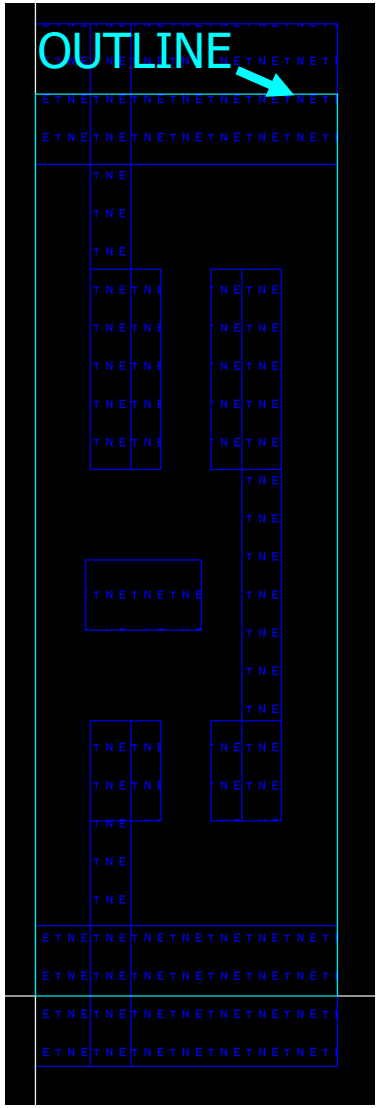




How to Create Abstract Views - Alternative

- Abstract view only includes the metal information





EECS E6321 Tutorial 03

Standard cells, Virtuoso, Abstract Gen, and Silicon Smart

Mingoo Seok & Previous TAs

To Start with…

- Copy files from
  - /courses/ee6350/proj_2025Spring/ref/siliconsmart/example

Synopsys SiliconSmart (SIS)

- Synopsys SiliconSmart (SIS) is a characterization and modeling engine
- The tool characterizes the timing, power, noise, and other critical characteristics of logic/sequential gates
- Given a netlist for a cell, SIS runs the simulations and generates the library file (.lib) for the cell
- The generated library can be used in Synthesis, PrimeTime, and compatible tools

Library Timing Model

- Concurrent Current Source (CCS)
  - Maintains accuracy by capturing the full output waveforms under varying slew and load conditions
  - Uses a current source for driver modeling
  - Consists of current samples as a function of time
  - More accurate, but larger
- Effective Current Source Model (ECSM)
  - Consists of voltage samples as a function of time
  - More efficient but less accurate / We will use this model
- Non-Linear Delay Model (NLDM)
  - Uses a voltage source for driver modeling
  - It is simpler than the other two models

Important files

- run.tcl
  - Includes general setting and commands for SIS characterization flow
  - Defines list of cells to characterize
- configure.tcl
  - Sets the operating condition: VDD, VSS, temperature, process
  - Defines parameters: Input slew, loading capacitance …
- netlists/
  - You need to **put** your netlist file (.sp) and instance file (.inst) here to be characterized
- lc.command
  - A script running Library Compiler to generate a .db file from the .lib file
- You can find some sample .tcl files at '/tools/synopsys/silicon-smart/P-2019.06/etc/examples/'
- SiliconSmart command: $siliconsmart run.tcl

How to Generate .sp File

- First method
  - You can copy '(Cellname).src.net' file from your 'lvs_run' directory and rename it to '(Cellname).sp'
  - I recommend to use *.src.net file instead of *.sp file in 'lvs_run' directory
- Second method
  1. At cell's schematic window, select Launch → ADE L
  2. At ADE window, select Setup → Simulator, and change Simulator as 'hspiceD'
  3. Select Simulation → Netlist → Create
  4. You can find 'netlist file in ~/simulation/(Cellname)/hspiceD/schemetic /netlist directory
  5. Make this netlist as subckt, and rename it
- At this PDK, the transistor instance name should be started with '**x**' (Consider this when drawing the schematic in Virtuoso)

run.tcl

```
1  ######################################################################
2  # Create Characterization Directory
3  ######################################################################
4  set char_dir ibm13_cmrf8sf_ecsm_tt_25_1p0
5  create ${char_dir}
6
7  # Copy common scripts in the specific directory for library characterization
8  exec cp configure.tcl ${char_dir}/config/configure.tcl
9  set_log_file ${char_dir}/sis.log
10 set_location ${char_dir}
11
12 # Set cells to characterize and their netlist
13 set netlist_dir ${char_dir}/netlists/
14 set cells       {inv}
15
16
17 ######################################################################
18 # Import a reference Liberty model
19 # SIS tool allows the user to pick and choose the amount of
20 # information to be extraced from an existing/reference model
21 #
22 # Use ONE of the cases below
23 ######################################################################
24 #set import_lib /courses/ee6321/share/ibm13rflpvt/synopsys/scx3_cmos8rf_lpvt_tt_1p2v_25c.lib
25 #exec cp netlists ${char_dir}/ -ru
26
27 # Case 1: Extract all information
28 #import -fast -liberty $import_lib -extension .sp -netlist_dir $netlist_dir
29
30 # Case 2: Extract cell/pin/function/when information but no loads/slews
31 #         load/slews for char will be used from SIS parameter settings
32 #import -fast -liberty $import_lib -extension .sp -netlist_dir $netlist_dir -use_default_loads -use_default_slews $cells
33
34 # Case 3: Extract only the port directions and functional information of the cells
35 #import -fast -liberty $import_lib -extension .sp -netlist_dir $netlist_dir -use_default_whens $cells
36
37 # Case 4: Import just netlist and cells to be characterized
38 #import -fast -extension .sp -netlist_dir $netlist_dir $cells
```

Characterization directory

Put your .sp file in the netlist directory, also put .inst file if needed

- If you want to recharacterize the library in different conditions or add more cells at the library, please use the red box part

run.tcl

```
41 ##################################################################
42 # Prepare the cells by copying the instance files and netlists into
43 # the characterization directory
44 # Becuase this is not recharacterization process, we need to define
45 # our cell functionality and netlist manually
46 ##################################################################
47 foreach c $cells {
48     exec cp netlists/${c}.sp [get_location]/netlists/
49     exec cp netlists/${c}.inst [get_location]/control/
50 }
51
52
53 ##################################################################
54 # Generate a characterization plan for each cell and prepares it for
55 # characterization
56 ##################################################################
57 configure -fast -timing -ecsm -power
58
59
60 ##################################################################
61 # Run simulations and characterize
62 ##################################################################
63 characterize $cells
64
65
66 ##################################################################
67 # Model the specific data
68 # Arguments
69 # -skeletion: For use with recharacterization, dicards all timing,
70 #             constraint, internal_power, and CCS timing/power/noise
71 #             tables from the reference liberty
72 # -output: Specifies a prefix for the model output file
73 # The generated model includes timing especially ECSM voltage waveform,
74 # power, noise data
75 ##################################################################
76 model -skeleton -lib_name ibm13_cmrf8sf_ecsm -timing -ecsm -power -si -output ibm13_cmrf8sf_ecsm $cells
77
78
79 ##################################################################
80 # Any steps for model post-processing
81 ##################################################################
82 log_info "IAMDONE"
```

2) Configure (based on configure.tcl)

3) Characterize

4) Model

1) In this case, because there is no reference library, we need to define our cell functionality

inv.inst

```
1 set_netlist_file ../netlists/inv.sp
2 add_pin in default -input
3 add_pin out default -output
4 add_function out !in
```

If you want to know how to define the cell functionality, please see 'SiliconSmart_User_Guide'

Define the prefix of library name and output file name

configure.tcl

```
 3 ################################
 4 # OPERATING CONDITIONS DEFINITION
 5 ################################
 6 create_operating_condition tt_25_1p0
 7 add_opc_supplies     tt_25_1p0 vdd 1.0
 8 add_opc_grounds      tt_25_1p0 vss 0
 9 set_opc_temperature tt_25_1p0 25
10 set_opc_process      tt_25_1p0 [subst {
11         { .lib "/courses/ee6321/share/IBM_PDK/cmrf8sf/relDM/HSPICE/models/allModels.inc" tt }
12         { .inc "/courses/ee6321/share/IBM_PDK/cmrf8sf/relDM/HSPICE/models/design.inc" }
13 }]
```

Define the operation condition

Technology model

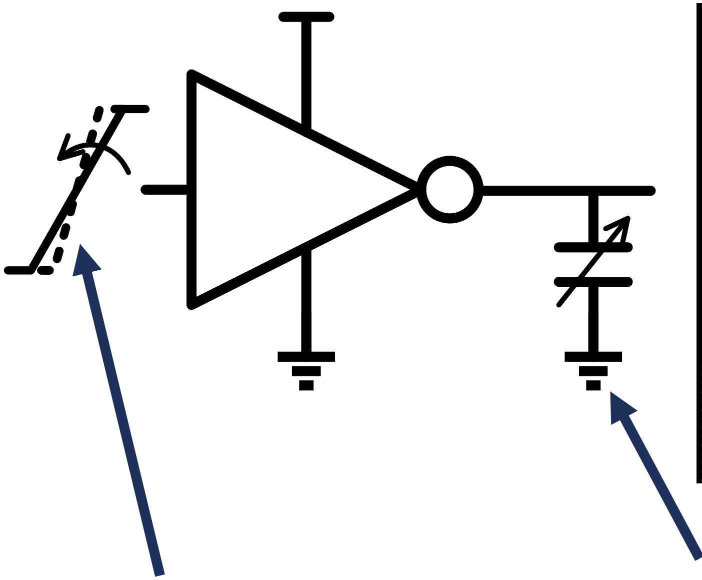
```
64     # Simulation resolution
65     set time_res_high 1e-13
66     set time_res_low 1e-12
67     #set gate_leakage_time_scaling_factor 100
```

```
144    # Operating load ranges
145    set default_load 0
146    set smallest_load 1e-15
147    set largest_load 200e-15
148    set autorange_load off
149
150    # Operating slew ranges
151    set smallest_slew 1e-11
152    set largest_slew 200e-11
```

Set the time resolution and slew rate range based on your FO4 delay

configure.tcl

- When SiliconSmart generates the .lib file, it sweeps various parameters (e.g. slew rate, output capacitance…)
- We should determine the reasonable range based on basic HSPICE simulation




```
# Operating slew ranges
set smallest_slew 0.49e-11
set largest_slew 490e-11
```

```
# Operating load ranges
set default_load 0
set smallest_load 1e-15
set largest_load 200e-15
set autorange_load off
```

Library Compiler

- After executing SiliconSmart, all the detailed model, result, reports are generated in the results directory (char_dir in run.tcl)
  - Any naming is allowed, but recommend to use your technology, PVT setting and library timing model (e.g. ibm13_cmrf8sf_ecsm_tt_25_1p0)
- **Library Generator** generates the .db file from .lib file
  - .db file is used for synthesis
- Command: $lc_shell –xg_mode –f lc.command | tee lc.log
- lc.command

```
2 read_lib ./lib/ecsm_tt_25_1p0.lib
3 write_lib ecsm_tt_25_1p0 -format db -output ./lib/ecsm_tt_25_1p0.db
4
5 exit
```

  - At this lab session, you have to use this script after copying the generated *.lib file in ./lib directory
- Please see run.sh or Makefile

Timing Library Information (.lib)


```
library(ibm13_cmrf8sf_ecsm_tt_25_1p0) {
  delay_model : table_lookup ;
  library_features(report_delay_calculation, report_power_calculation, report_noise_calculation);
  cell(inv) {
    ecsm_vtp : 0.425 ;
    ecsm_vtn : 0.3989 ;

    leakage_power() {
      related_pg_pin : "vdd" ;
      value : "0.0001104" ;
    }

    pg_pin(vdd) {
      voltage_name : vdd ;
      pg_type : primary_power ;
    }

    pg_pin(vss) {
      voltage_name : vss ;
      pg_type : primary_ground ;
    }

    pin(in) {
      capacitance : 0.001577 ;
      direction : input ;
      driver_waveform_rise : "driver_waveform_default_rise" ;
      driver_waveform_fall : "driver_waveform_default_fall" ;
      fall_capacitance : 0.001571 ;
      fall_capacitance_range(0.001122, 0.001571);
      input_voltage : default ;
      max_transition : 4.9 ;
      related_ground_pin : vss ;
      related_power_pin : vdd ;
      rise_capacitance : 0.001577 ;
      rise_capacitance_range(0.001178, 0.001577);
    }

    pin(out) {
      direction : output ;
      function : "(!in)" ;
      max_capacitance : 0.6563 ;
      max_transition : 3.921 ;
      min_capacitance : 0.0001 ;
      output_voltage : default ;
      related_ground_pin : vss ;
      related_power_pin : vdd ;

      internal_power() {
        related_pin : "in" ;

        fall_power(pwr_tin_oload_9x9) {
          index_1("0.0049, 0.04959, 0.2065, 0.506, 0.9727, 1.628, 2.489, 3.575, 4.9");
          index_2("0.0001, 0.006091, 0.02712, 0.06728, 0.1298, 0.2176, 0.3332, 0.4787, 0.6563");
          values("-0.0002858, -0.0001689, -0.0001562, -0.000154, -0.0001528, -0.0001525, -0.0001524, -0.0001523, -0.0001522",\
                 "-0.0003407, -0.0002517, -0.0001855, -0.0001676, -0.0001608, -0.0001581, -0.0001567, -0.000156, -0.0001556",\
                 "-3.188e-05, -0.0001848, -0.000188, -0.0001722, -0.0001636, -0.0001602, -0.0001574, -0.000157, -0.0001568",\
                 "0.0006978, 0.000263, 9.489e-06, -6.915e-05, -0.0001028, -0.0001197, -0.0001294, -0.0001362, -0.0001405",\
                 "0.001902, 0.00119, 0.0005531, 0.0002601, 0.0001091, 3.018e-05, -2.113e-05, -5.392e-05, -7.819e-05",\
                 "0.003629, 0.002653, 0.001562, 0.0009344, 0.0005751, 0.000369, 0.0002339, 0.0001381, 7.271e-05",\
                 "0.005922, 0.004711, 0.003117, 0.002056, 0.001399, 0.0009728, 0.0007071, 0.0005156, 0.0003698",\
                 "0.008824, 0.007398, 0.005305, 0.003718, 0.002671, 0.001956, 0.001464, 0.00112, 0.0008734",\
                 "0.01238, 0.01077, 0.008171, 0.006002, 0.004459, 0.003383, 0.002614, 0.002061, 0.001635");
        }

        rise_power(pwr_tin_oload_9x9) {
          index_1("0.0049, 0.04959, 0.2065, 0.506, 0.9727, 1.628, 2.489, 3.575, 4.9");
          index_2("0.0001, 0.006091, 0.02712, 0.06728, 0.1298, 0.2176, 0.3332, 0.4787, 0.6563");
          values("0.001264, 0.001374, 0.001383, 0.001325, 0.001222, 0.001225, 0.0009967, 0.0007334, 0.0005668",\
                 "0.001273, 0.00127, 0.00132, 0.001263, 0.001168, 0.001073, 0.0008361, 0.0009322, 0.0004124",\
                 "0.001591, 0.00142, 0.001237, 0.001158, 0.001148, 0.001149, 0.000945, 0.000677, 0.0004506",\
```

Library name

Leakage power info.

Power pin info.

Input specifications

Functional definition

Transition-specific energy info.

You can see transition-specific delay info. below

End of Slides