

Layout Synthesis using Automatic Place-and-Route Tools - Part 1

Jaeha Kim, Sung-Joon Lee, and Eunseo Kim
Mixed-Signal IC and System Group
Seoul National University
May 20. 2016

Place & Route Directory Overview

- ▶ First, let's look at the directory of the place & route
- ▶ Type at the terminal :

```
$ cd IDEC_CBDF/Place_Route  
$~/IDEC_CBDF/Place_Route> ls
```

- ▶ You'll see two folders
 - ▶ **lib** : folder containing milkyway libraries
 - ▶ **script** : folder containing ICC scripts, Makefile, etc.
 - ▶ **answer** : example P&R-done result

P&R Library

- ▶ “lib” folder contains three MilkyWay libraries :
 - ▶ scmos_harp : PLL cell libraries which we simulated in day 1
 - ▶ scmos_cello : Digital cell libraries (NAND, NOR, INV, ...)
 - ▶ scmos_viola : I/O PAD cell libraries
- ▶ What's missing is the **top library** that uses the above three libraries as reference libraries, and will contain the result of P&R

Starting IC Compiler

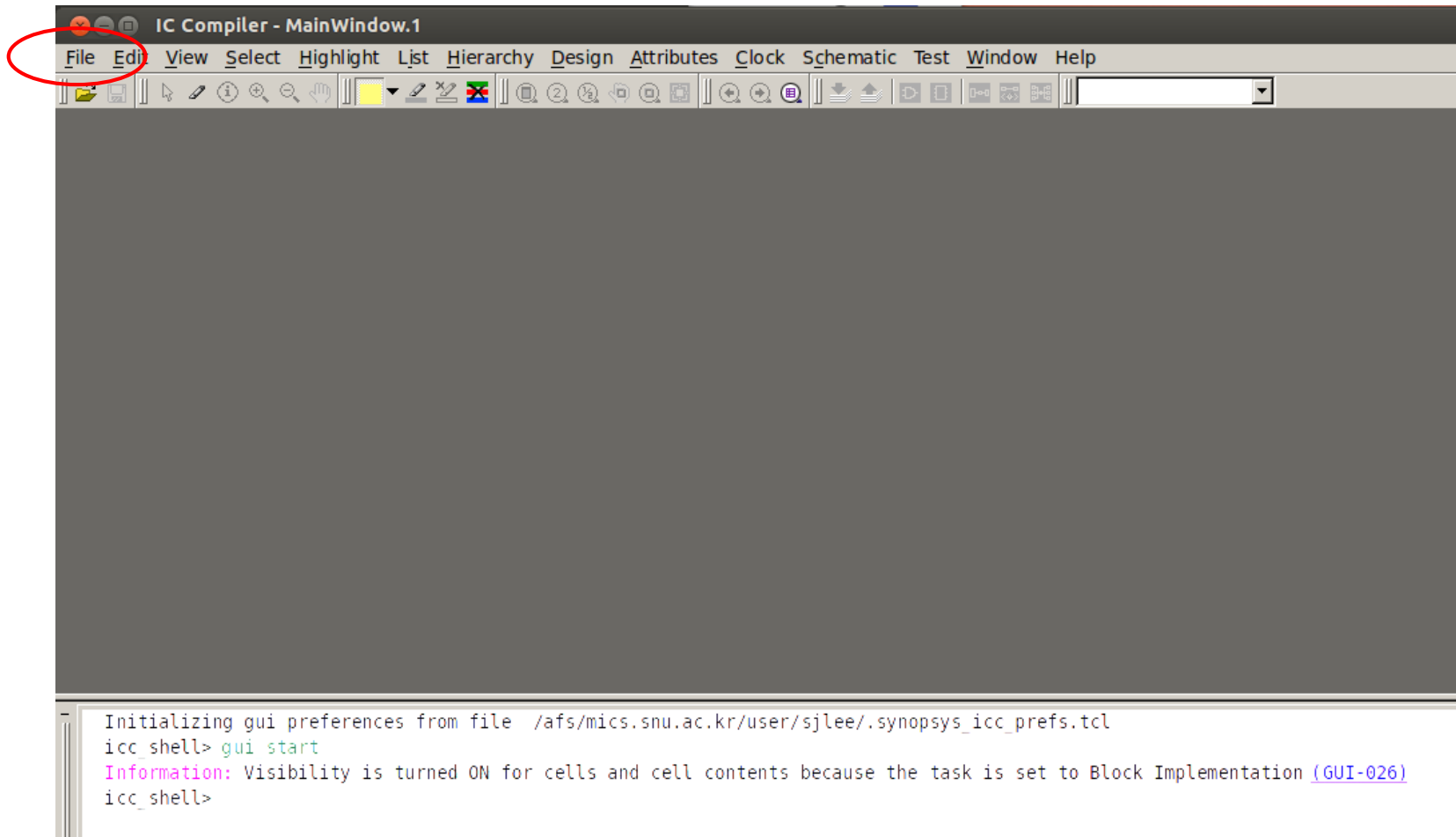
- ▶ You can see the contents of MilkyWay Libraries by starting IC Compiler
- ▶ Move to the directory “script”, Type “icc_shell -gui &”

```
$~/IDEC_CBDF/Place_Route> cd script
```

```
$~/IDEC_CBDF/Place_Route/script> icc_shell -gui &
```

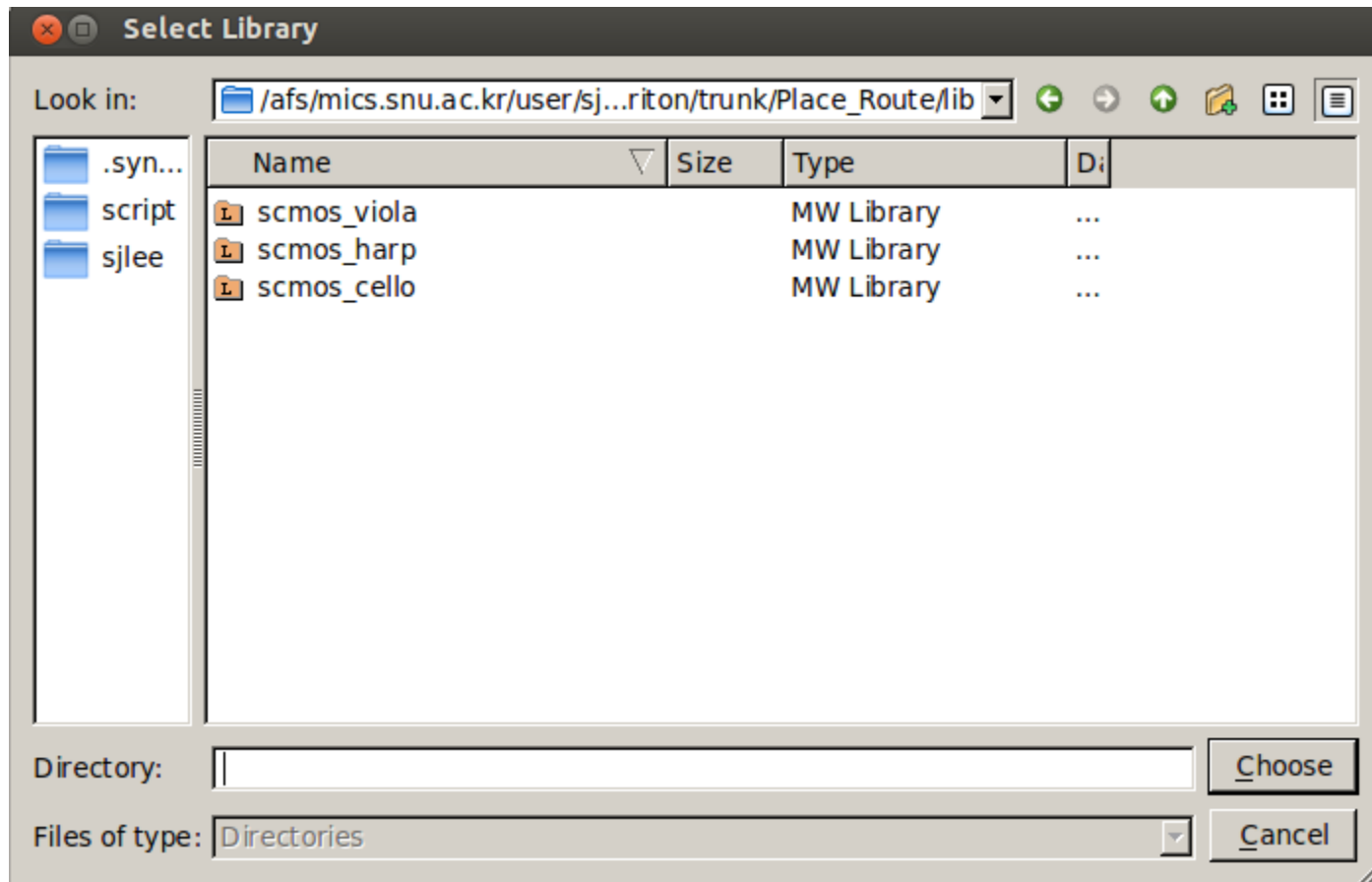
IC Compiler Interface

► Click File > Open Design...



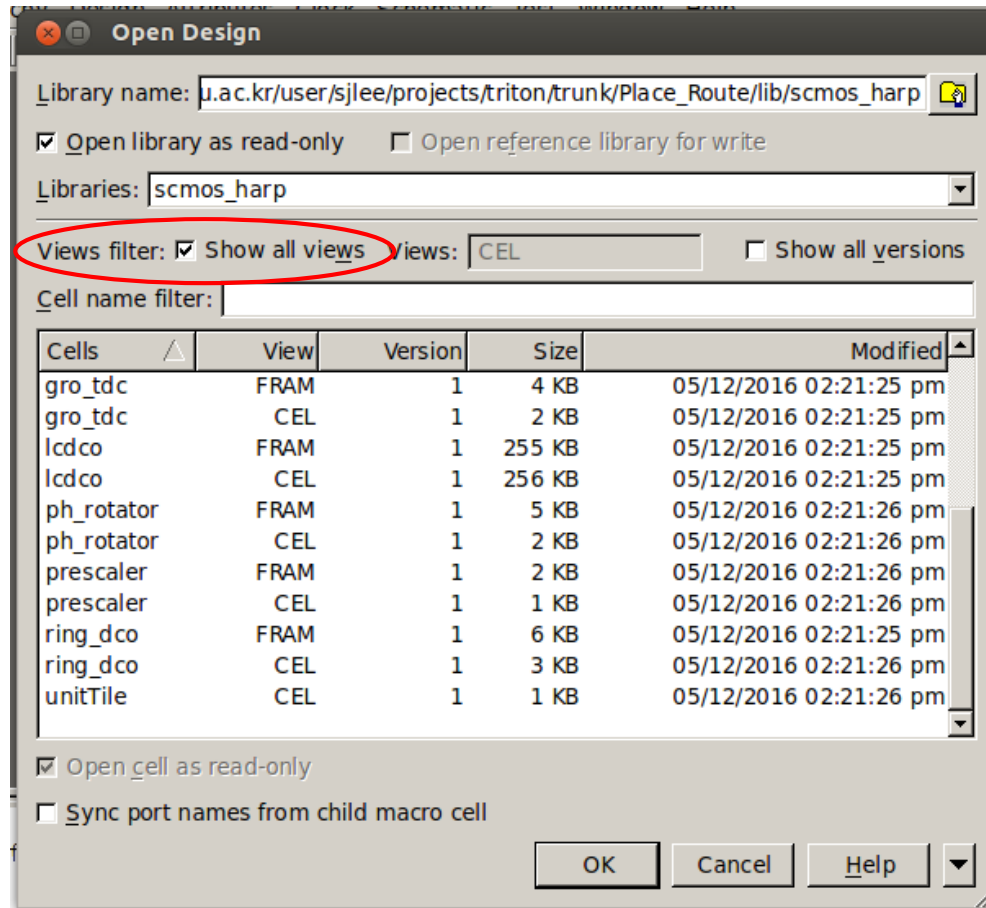
IC Compiler Interface

- Move to the directory ~/IDEC_CBDF/Place_Route/lib, and choose scmos_harp

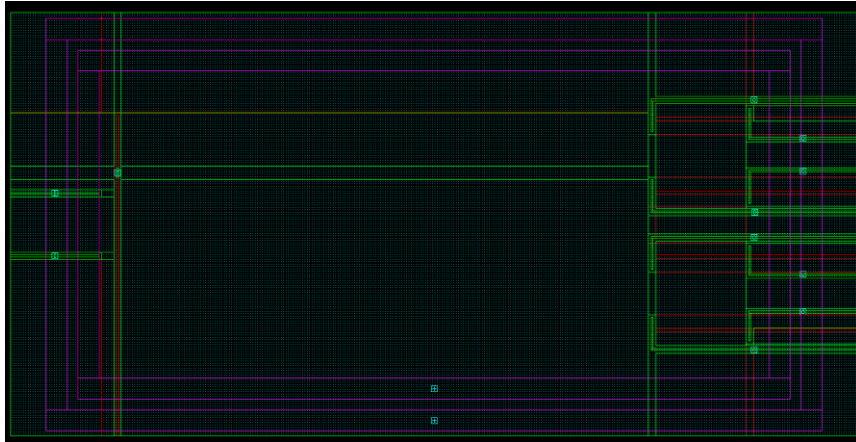


IC Compiler Interface

- ▶ Check “Show all views” and you can see the two types of view

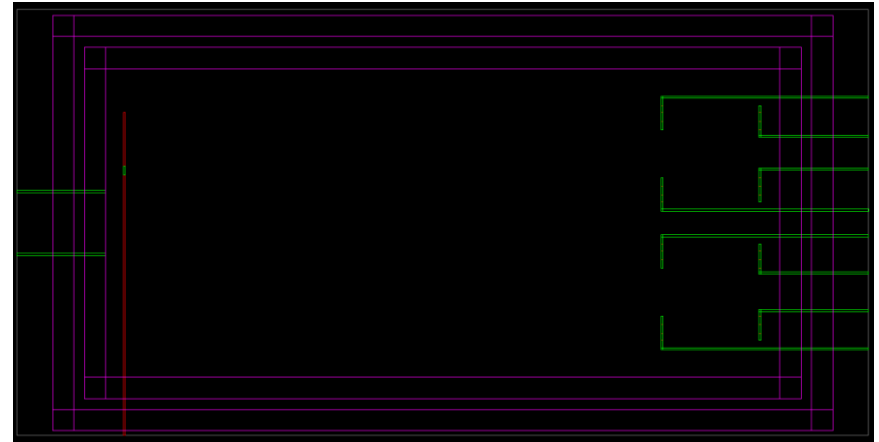


FRAM View and CEL View



FRAM view

An abstract representation of a cell used for placement and routing; contains only the metal blockages, allowed via areas, and pins of the cell



CEL view

The full layout view of a physical structure such as a via, standard cell, macro, or whole chip; contains placement, routing, pin, and netlist information for the cell

Creating Top Library

- ▶ Close the library, and let's create top library to proceed our P&R
 - ▶ Click File > Exit
- ▶ Type "make create_top_library" in the terminal

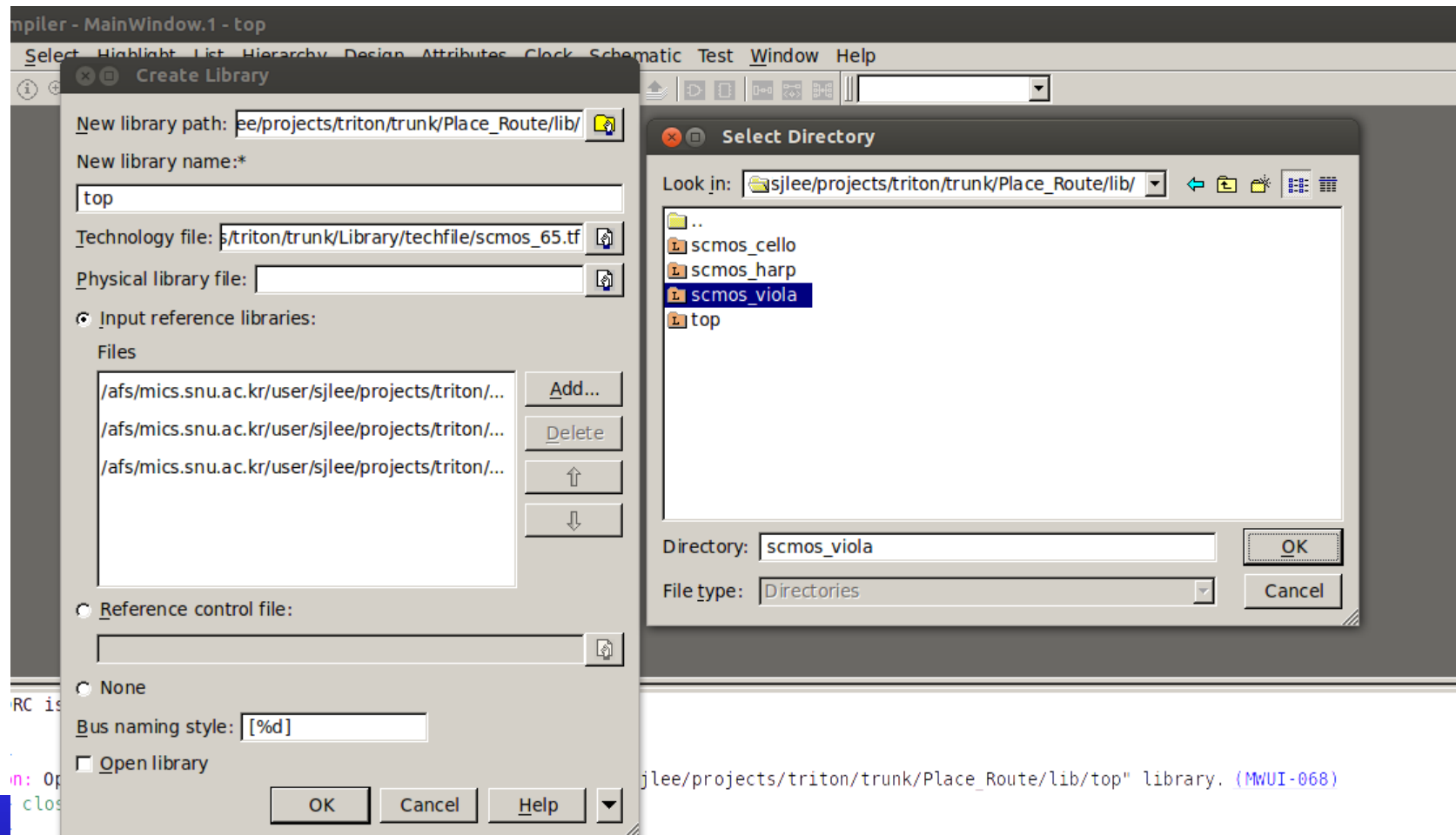
```
$ make create_top_library
```
- ▶ What happened?

Creating Top Library

- ▶ You have run IC compiler using the script `./prep_scripts/create_top_library.tcl` by Makefile
- ▶ You can find out that “top” directory in `~/IDEC_CBDF/Place_Route/lib/`

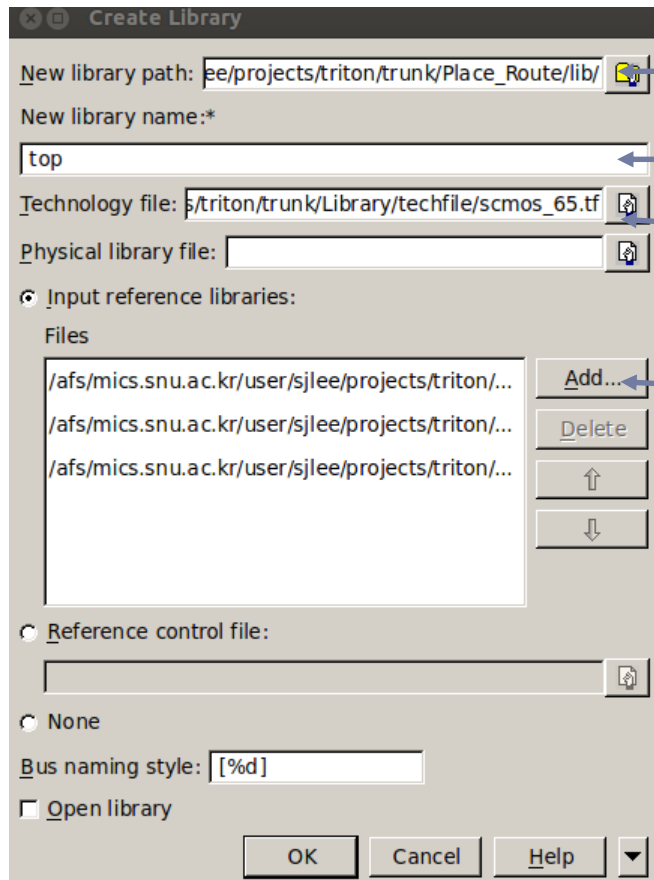
Creating Top Library

- ▶ You can also make top library in GUI, by clicking File > Create Library



Creating Top Library

- ▶ You can also make top library in GUI, by clicking File > Create Library



Top Library Path

Top Library Name

Attached Techfile

Reference Library to use

First look at the P&R script

- ▶ Before beginning the P&R, let's see what script do we have:

```
$~/IDEC_CBDF/Place_Route/script> cd icc_scripts  
$~/IDEC_CBDF/Place_Route/script/icc_scripts> ls
```

- ▶ You can see 10 tcl files

P&R Script Overview

| Flow | Script name | Descriptions |
|-------------------|--------------------------------------|--|
| Setup Environment | init_setup.tcl | <ul style="list-style-type: none"> • Initial parameter setup for P&R • chip size, filler location, power domain/strap/name |
| | init_design.tcl | <ul style="list-style-type: none"> • model, pad location, SDC load • Initial floorplan |
| Place | place_optimizer.tcl | <ul style="list-style-type: none"> • Custom cells and digital blocks placement, set blockage area • Place optimization |
| | clock_tree_syn.tcl | <ul style="list-style-type: none"> • Clock tree synthesis |
| | pg_imprinting.tcl, insert_filler.tcl | <ul style="list-style-type: none"> • Power/ground allocation, insert filler |
| Routing | power_syn.tcl, route_optimizer.tcl | <ul style="list-style-type: none"> • Power strap generation, clock tree and wire routing |
| Output | output.tcl | <ul style="list-style-type: none"> • Finishing works & making outputs |
| Etc. | open_lib.tcl | <ul style="list-style-type: none"> • Opening Top library with discretized grids in GUI (For convenience) |

Path/Control Input Setting – init_setup.tcl

init_setup.tcl

- ▶ Before running automation scripts, you have to set variables in init_setup.tcl suitable for your design
 - ▶ Path/Design definition
 - ▶ Control inputs/variables
 - ▶ Library settings

```
#Path/Design definition ---  
set PROJPATH ...  
...  
Set DESIGN "TOP"  
...  
#Defining Lambda based grid ---  
Set LAMBDA 0.035  
...  
#Defining Chip & Pad Size ---  
Set CHIP_WIDTH ...
```

Path/Control Input Setting – init_setup.tcl

init_setup.tcl

- ▶ Before running automation scripts, you have to set variables in init_setup.tcl suitable for your design
 - ▶ Path/Design definition
 - ▶ Control inputs/variables
 - ▶ Library settings

```
#Information for filler insertion ---
set FILLER_ARRAY ...
...
#Power-domain/strap information ---
Set NUM_PG ...
Set DPLL_POWER_NET "vdda"
...
#Control inputs ---
Set MW_TOP_EXIST "1"
...
```


Initializing Design

- ▶ Initializing design covers :
 - ▶ Floorplanning
 - ▶ Conducting coarse placement
 - ▶ Creating plan-group
 - ▶ Reading SDC

Initializing Design

► \$~/IDEC_CBDF/Place_Route/scripts> vi icc_scripts/init_design.tcl

```
#Read hierarchical design -----
```

```
read_verilog -top top -verbose "\
```

```
$SYNPATH/digital_lf/digital_lf_mapped.v \
```

Reading model hierarchically

```
$MODEL_PATH/dpll/dpll.sv \
```

```
$MODEL_PATH/top/top.sv \
```

```
"
```

```
...
```

```
#Creation of floorplan
```

Creating floorplan

```
create_floorplan -control_type width_and_height -core_width $CORE_WIDTH
```

```
-core_height $CORE_HEIGHT ...
```

Initializing Design

#Creation & Visualization of plan groups -----

remove_plan_groups -all

create_plan_groups {dpllo/digital_lf} -cycle_color

Creating plan group for
digital_lf

#Coarse Placement &Resizing of plan groups

create_fp_placement

Coarse Placement

...

#Reading SDC-format timing-constraint for each cell

current_instance /top/dpllo/digital_lf

Read SDC of digital_lf_mapped

read_sdc \${SYNPATH}/digital_lf/digital_lf_mapped.cell.sdc

...

save_mw_cel \${DESIGN}

Plan Group

- ▶ **A plan group** is a partition with a physical boundary. Each plan group represents a module in the logic hierarchy that you want to implement as a physical block and contains cells that belong to that module. A plan group can have many logical modules.

Initializing Design

- ▶ To initialize a design, type “make init_design”

```
$~/IDEC_CBDF/Place_Route/script> make init_design
```

- ▶ You can find the detail of process log at ‘script/logs/init_design.log’ after initializing design is done

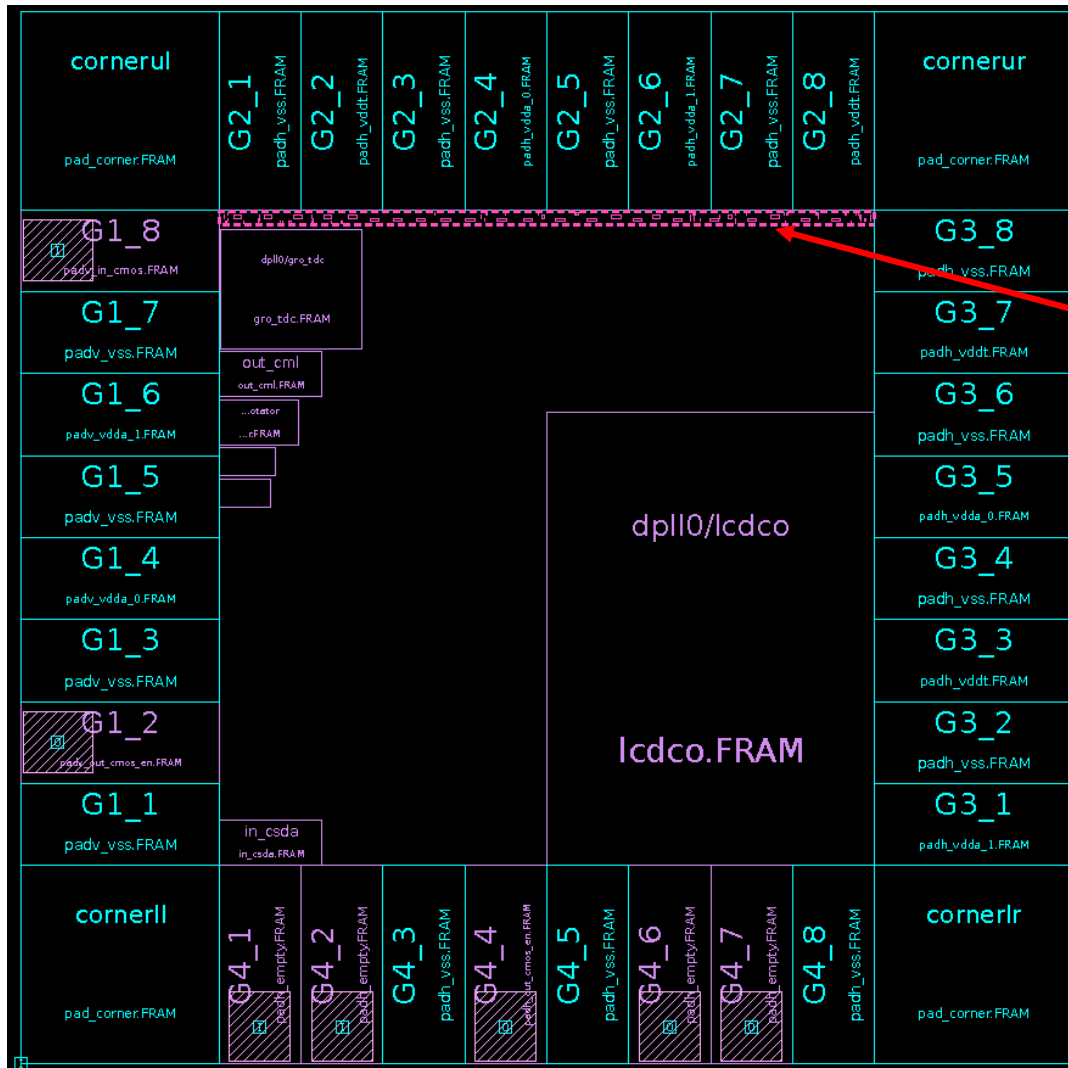
- ▶ Let’s find out the result. Type “make open_lib”

```
$~/IDEC_CBDF/Place_Route/script> make open_lib
```

Initializing Design Result

- ▶ In IC Compiler GUI, click File > Open Design
- ▶ Move to folder “lib” and choose top, click OK

Initializing Design Result



Plan Group of digital_If

* Make sure that all of your cells are placed correctly

Custom Placement

- ▶ Unlike digital cells, custom cells (such as cells of PLL) need to be decided where to be placed
- ▶ And also, we have to decide the boundary (plangroup) of digital cells to be placed
- ▶ Two ways of doing it :
 - ▶ 1) By script : giving coordinate of cells to the IC Compiler
 - ▶ 2) By tool : using Custom Designer
- ▶ We rather prefer 1). Why? (think about iteration)

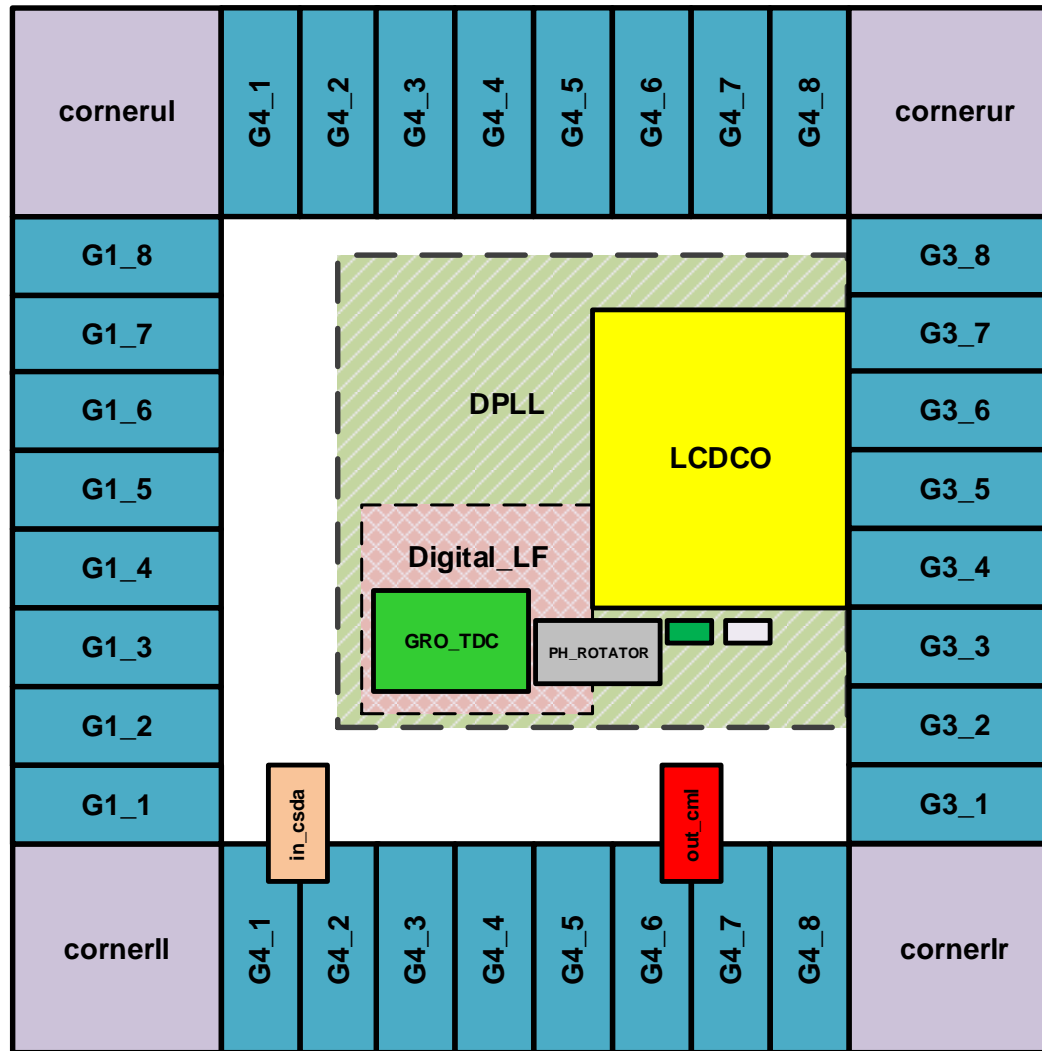
Dividing Chip Area by Voltage Domain

- ▶ We have two voltage domain in our chip,
 - ▶ vddt : voltage for I/O pad circuit
 - ▶ vdda : voltage for DPLL
- ▶ IC compiler do not consider the voltage domain when placing and adding digital cells, if we do not give constraints
- ▶ Giving constraints can be done by creating **voltage area, placement blockage**

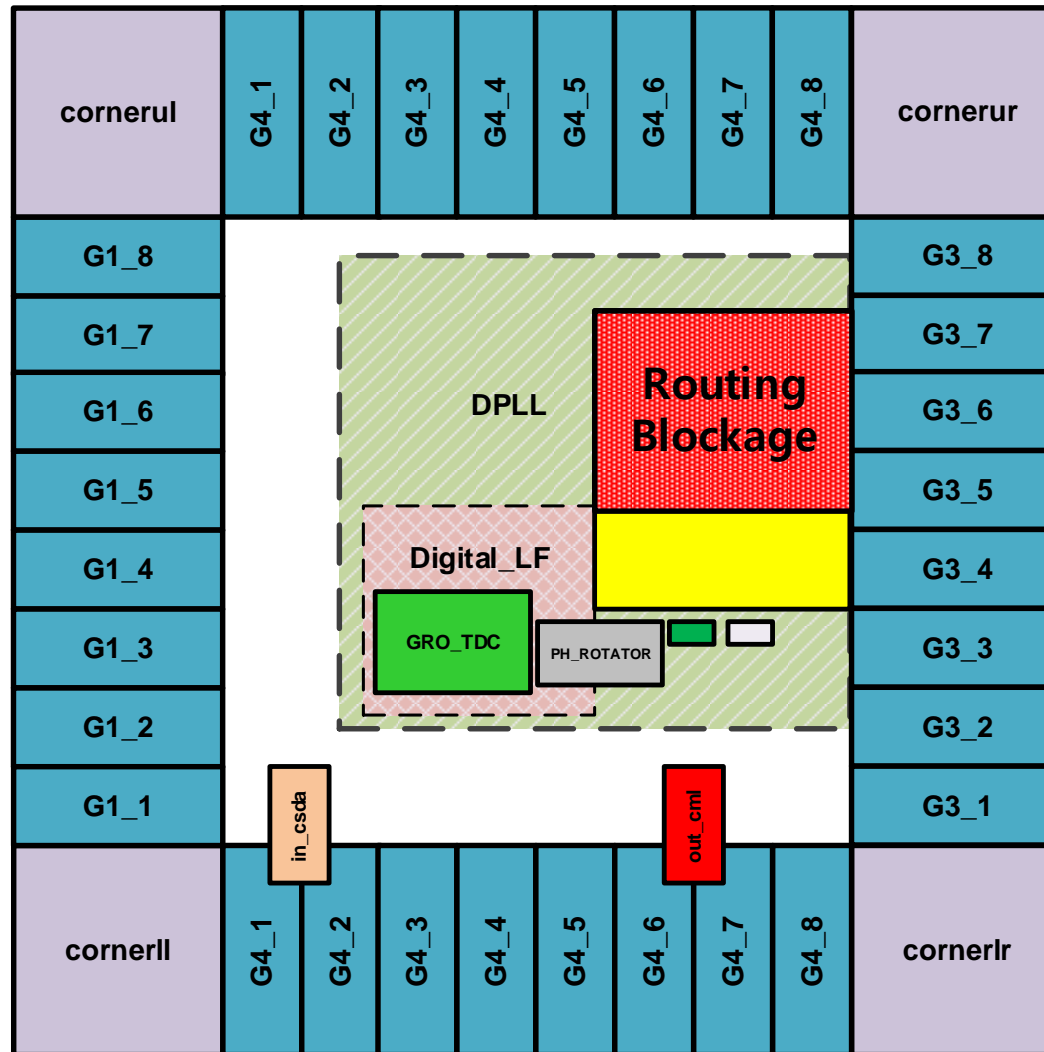
Routing Blockage (optional)

- ▶ Some custom cells are sensitive to crosstalk due to the wires crossing above them
- ▶ In our example, inductor in LCDCO should be isolated to other signals
- ▶ **Routing Blockage** literally blocks the specified layer of routing in the specified area

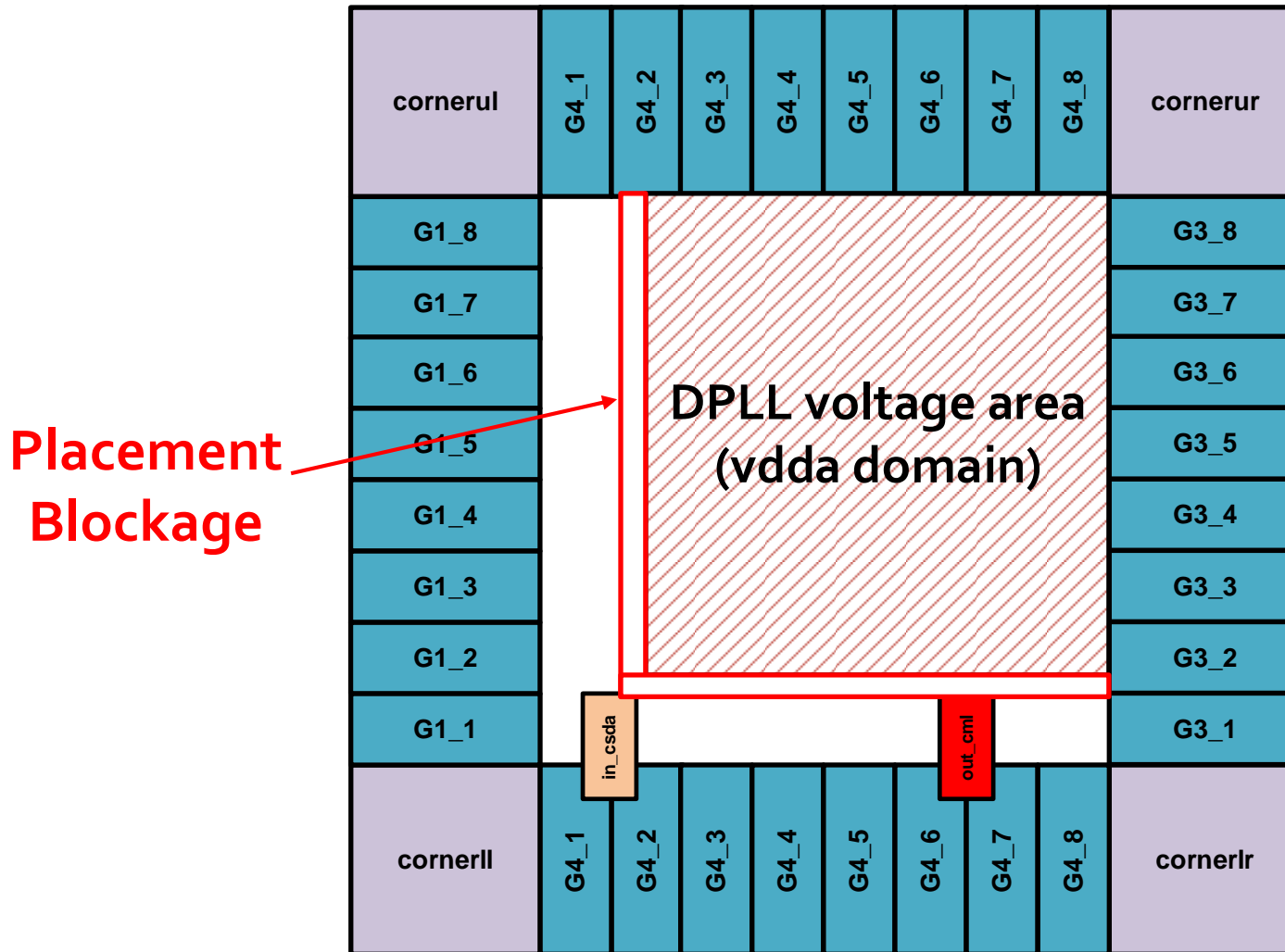
Concept of Voltage Area & Blockage



Concept of Voltage Area & Blockage



Concept of Voltage Area & Blockage



Script for Custom Placement

- ▶ Script for custom placement, voltage area, placement blockage and routing blockage are in 'floorplan/top.tcl'
- ▶ Let's see, at terminal :

```
$~/IDEC_CBDF/Place_Route/script> vi floorplan/top.tcl
```

Script for Custom Placement

```
#CELL out_cml
```

```
move_object -to {663.6 271.6} {out_cml}
```

```
rotate_objects -to N {out_cml}
```

Move, rotate and fix the cell

```
rotate_objects -by 270 -pivot {663.6 271.6} {out_cml}
```

```
set_undoable_attribute [get_cells -all {out_cml}] is_fixed {1}
```

```
...
```

```
#Plan Groups
```

```
set_object_boundary dpllo/digital_lf \
```

```
-bbox {{260.4 296.8} {520.8 470.4}}
```

Resize Plan group

```
#voltage area for dpll
```

```
create_voltage_area -coordinate {246.4 296.8 845.6 845.6} -name voltage_area_dpll {dpll}
```

```
-cycle_color
```

Create voltage area &
placement blockage

```
#placement blockage for filler cutting
```

```
create_placement_blockage -coordinate {{246.4 296.8} {253.4 845.6}} -name placement_blockage_o
```

```
-type hard
```

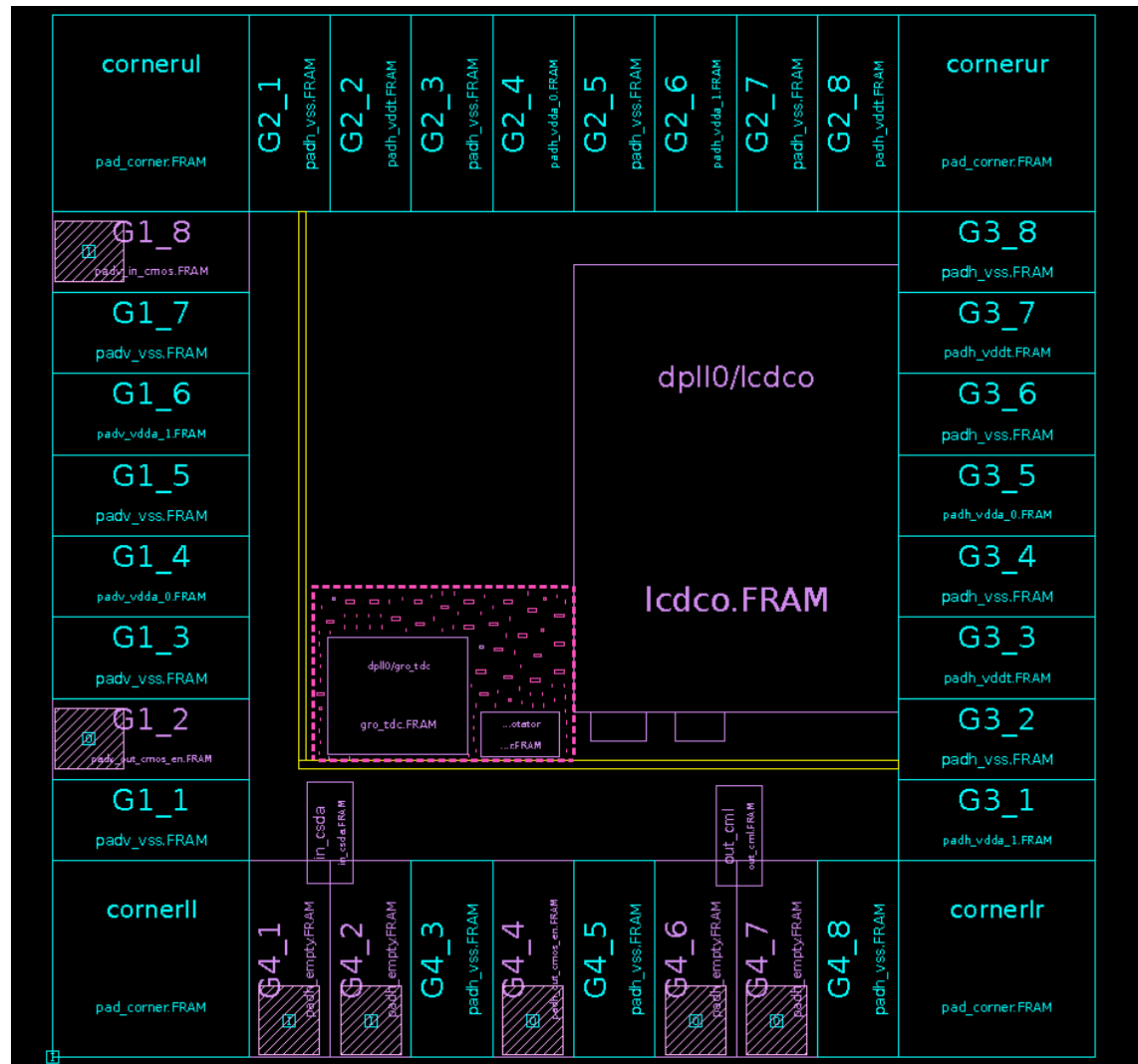
Optimizing Placement

- ▶ To optimize placement, close library in IC Compiler GUI, and run as following,

```
$~/IDEC_CBDF/Place_Route/script>make place_optimizer
```

- ▶ Process of optimizing placement includes
 - ▶ Loading floorplan/top.tcl
 - ▶ Getting custom cell coordinates and placing
 - ▶ Creating voltage area, placement blockage, routing blockage
 - ▶ Resizing/placing plangroups
 - ▶ Re-allocate digital cells & add digital cells if necessary

Optimizing Placement Result

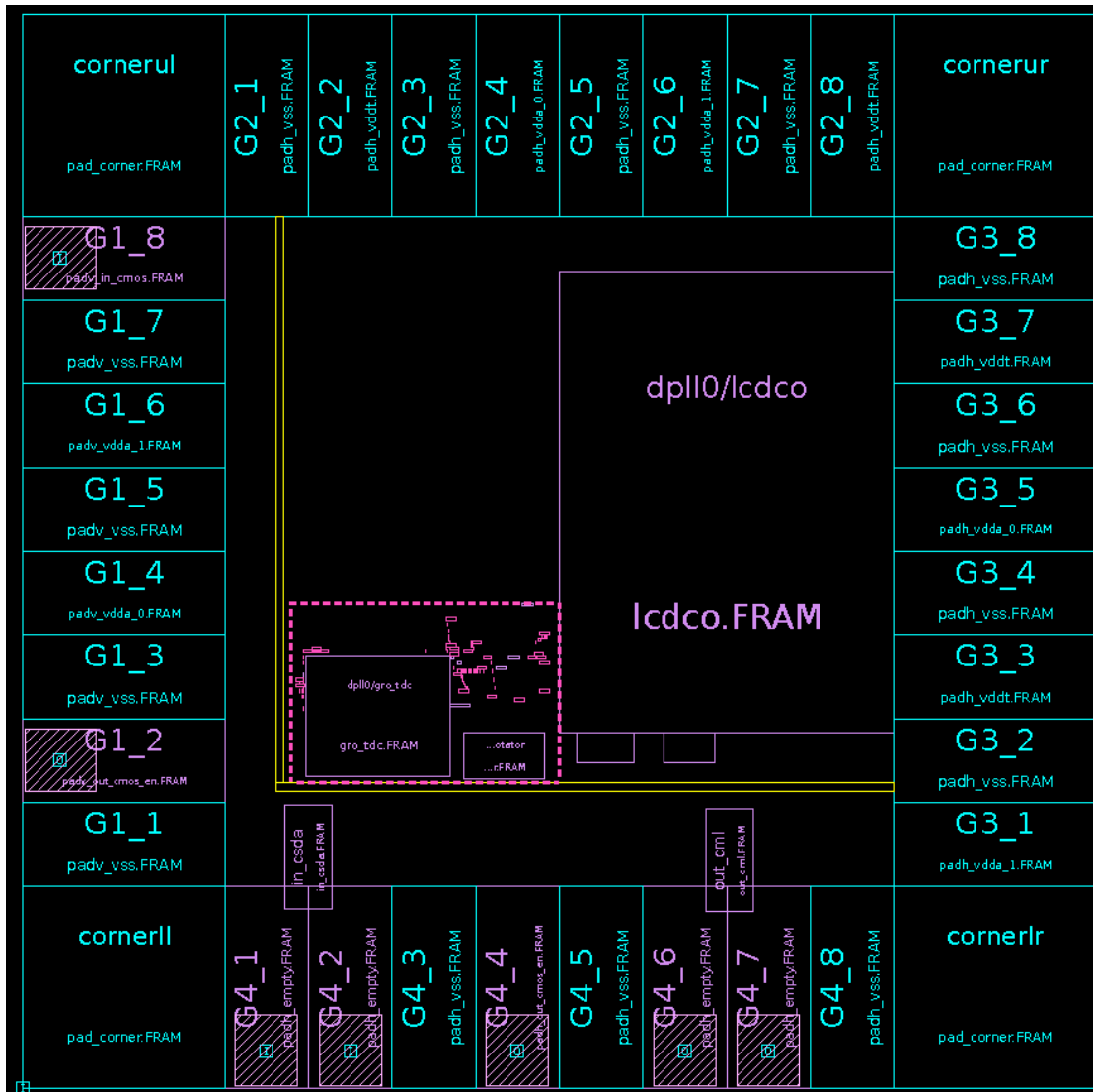


Clock Tree (pre)Synthesis

- ▶ Clock Tree (pre)synthesis includes re-allocating digital cells (especially buffer) and adding buffers if necessary
- ▶ *This procedure does not include routing
- ▶ For clock tree synthesis, run:

```
$~/IDEC_CBDF/Place_Route/script>make clock_tree_syn
```

Clock Tree Synthesis Result



Can you tell
the difference
between
previous step?

Power/Ground Imprinting

- ▶ Power/Ground imprinting is to create logical connections between power/ground pins on standard cells & macros and the power/ground nets in the design
- ▶ If the power and ground nets are not explicitly defined in your netlist, this procedure creates the power and ground connection
- ▶ Let's see the script, type as below

```
$~/IDEC_CBDF/Place_Route/script>vi icc_scripts/pg_imprinting.tcl
```

Power/Ground Imprinting

```
derive_pg_connection\
```

```
-power_net "$DPLL_POWER_NET" -ground_net "GROUND_NET" \
```

```
-power_pin "DPLL_ANALOG_POWER_PIN" -ground_pin "$DPLL_GROUND_PIN" \
```

```
-cells {dp1lo/*} -reconnect
```

Connect DPLL_ANALOG_POWER_PIN to
DPLL_POWER_NET

```
...
```

```
derive_pg_connection -power_net "$DPLL_POWER_NET" -ground_net
```

```
"$GROUND_NET" -tie -cells {dp1lo*}
```

Connect Logic 1/o to
DPLL_POWER_NET/GROUND_NET

```
...
```

```
save_mw_cel ${DESIGN}
```

Power/Ground Imprinting

- For power/ground imprinting, run:

```
$~/IDEC_CBDF/Place_Route/script> vi icc_scripts/pg_imprinting.tcl
```

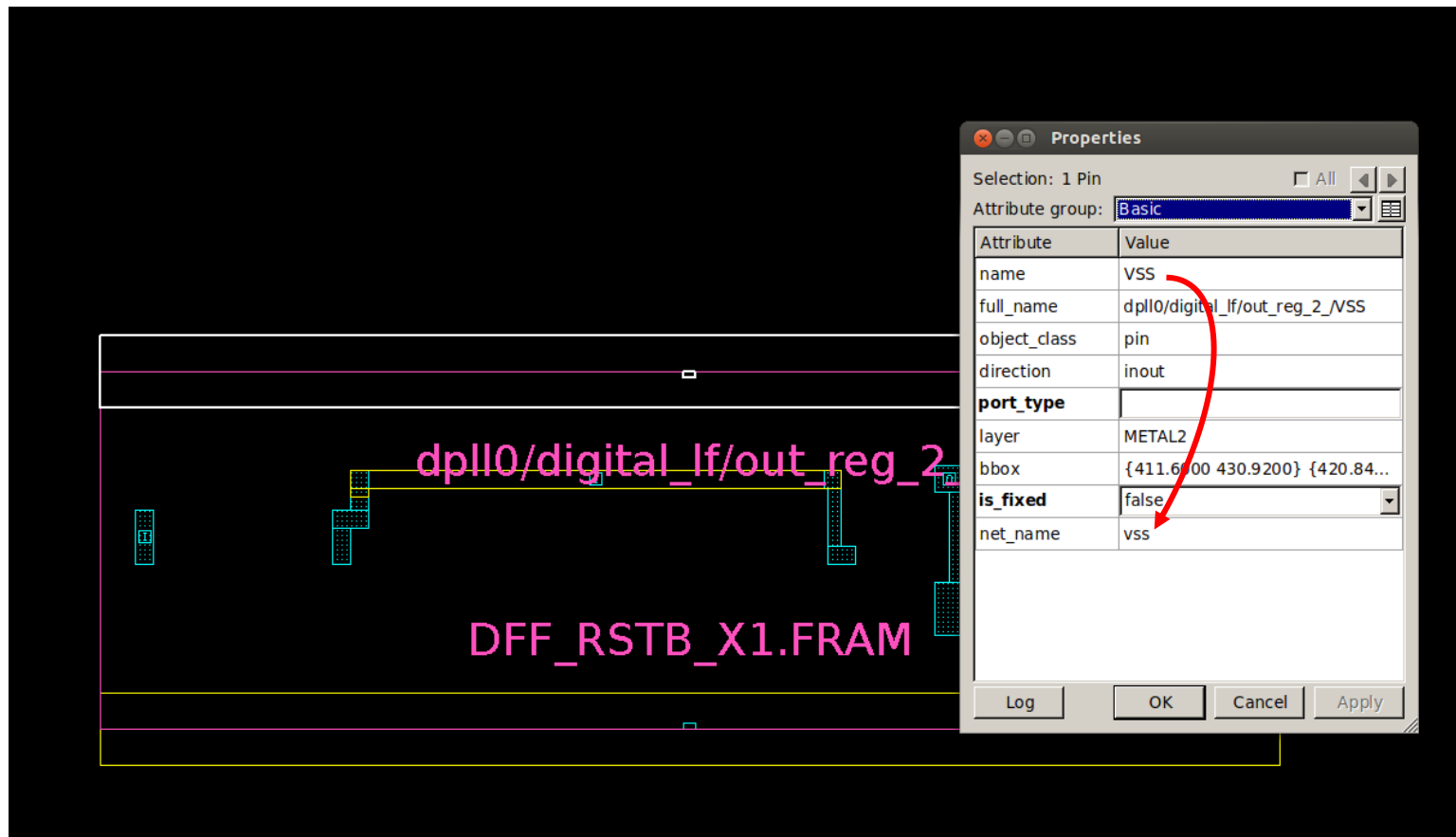
Power/Ground Imprinting Result

- In ICC GUI, check the Pin's 'Vis.' tap and click 'Apply'



Power/Ground Imprinting Result

- By clicking the left button to highlight the pin, and click right button and click “Properties”



Inserting Filler Cells

- ▶ After power imprinting, it requires filler cell insertion
- ▶ Filler cells fill gaps in the design to ensure that all power nets are connected and the spacing requirements are met
- ▶ In this example, the filler cells connected to vddt and the ones connected to vdda will be inserted and placed separately

Inserting Filler Cells

► \$~/IDEC_CBDF/Place_Route/script> vi icc_scripts/insert_filler.tcl

```
#vddt filler insertion
```

```
insert_stdcell_filler \
```

```
-cell_without_metal "$FILLER_ARRAY"
```

```
-bounding_box $FILLER_TOT_AREA \
```

```
-connect_to_power "$TEST_POWER_NET" \
```

```
-connect_to_ground "$GROUND_NET" \
```

```
-randomize
```

```
remove_stdcell_filler -bounding_box "$DPLL_FILLER_AREA" -stdcell
```

```
#vdda filler insertion
```

```
insert_stdcell_filler \
```

```
...
```

Insert vddt filler cells in whole chip

Remove vddt filler cells in DPLL area

Insert vdda filler cells in DPLL area

Inserting Filler Cells

- For inserting filler cells, run:

```
$~/IDEC_CBDF/Place_Route/script> make insert_filler
```

Inserting Filler Cells Result

