Lecture

15

ECE M216A

Physical Synthesis

Prof. Dejan Marković

ee216a@gmail.com

Routing is Everywhere in Your Daily Life

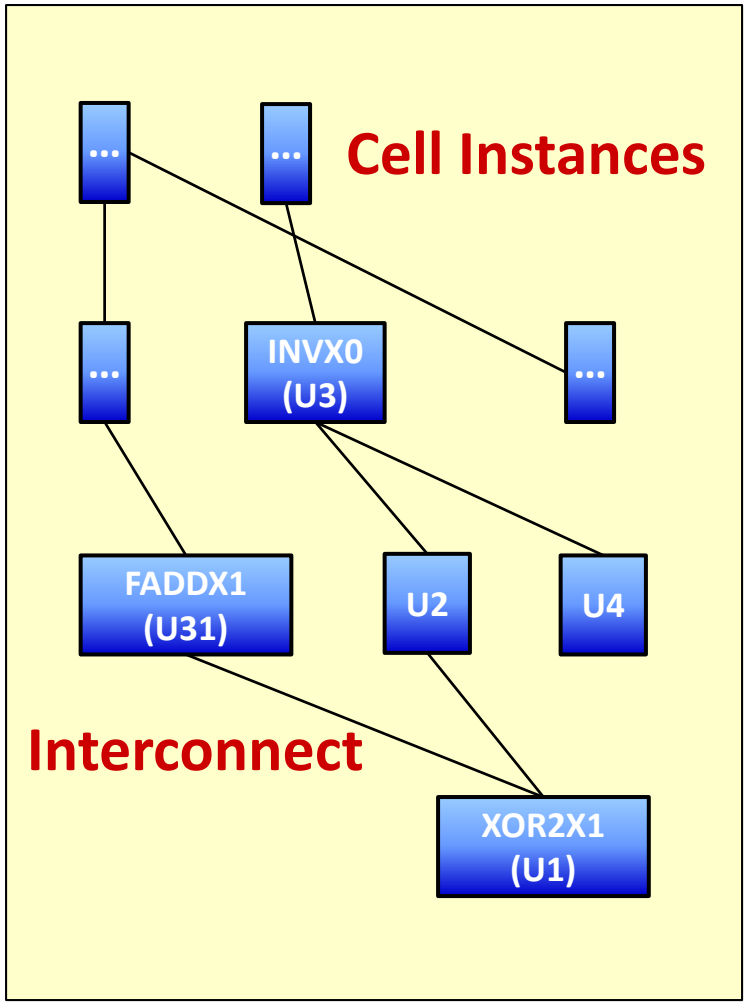
BAD Routing: Waste of Time + Gas!

BAD Routing: Slow Chip + Waste of Energy!

From Synthesis to Place-and-Route (PnR)

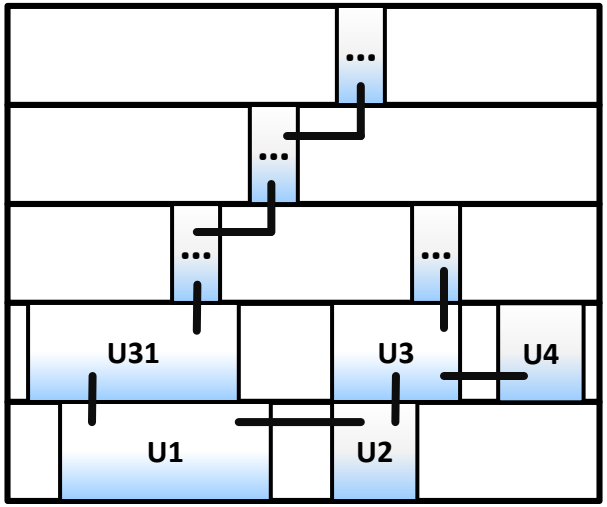
```
module ADD(DO, DI_A, DI_B) ;
  output [31:0] DO;
  input [31:0] DI_A;
  input [31:0] DI_B;
  wire ... ;
  ...
  FADDX1  U31 ( ... );
  FADDX1  U30 ( ... );
  ...
  XOR2X1  U1 ( ... ) ;
  XOR2X1  U2 ( ... ) ;
  INVX0  U3 ( ... ) ;
  NAND2X0  U4 ( ... ) ;
  OA21X1  U5 ( ... ) ;
endmodule
```

Synthesis Netlist



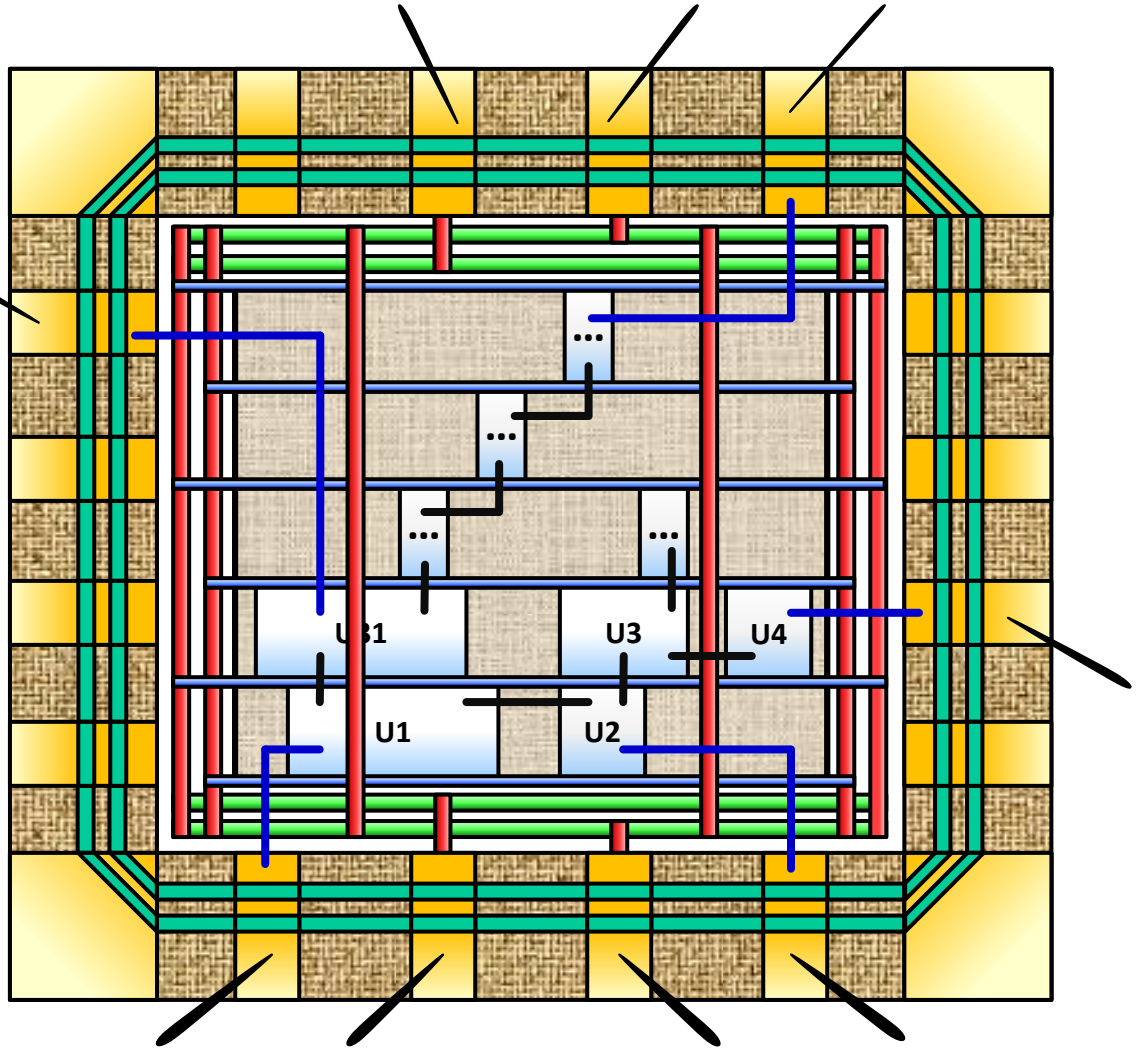
Visualization of Netlist

From Synthesis to Place-and-Route (PnR)

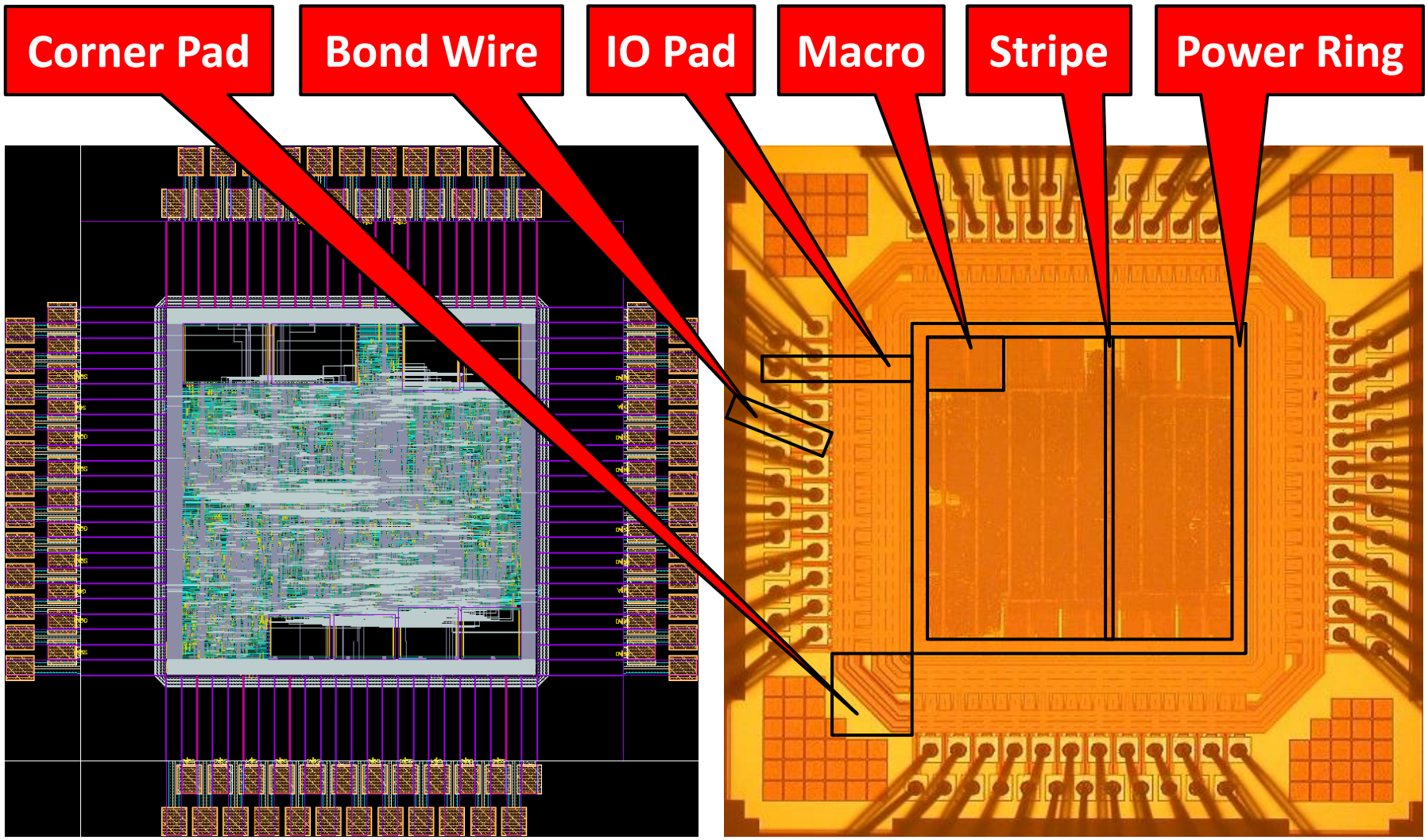


Wait! Of Course, Not That Easy…

- IO assignment
- Floor-planning & power-planning
- Cell placement
- Optimization, clock-tree syn. & detailed routing
- Add core filler
- Add IO filler
- DRC, LVS, tapeout
- Bonding (for chip measurements)



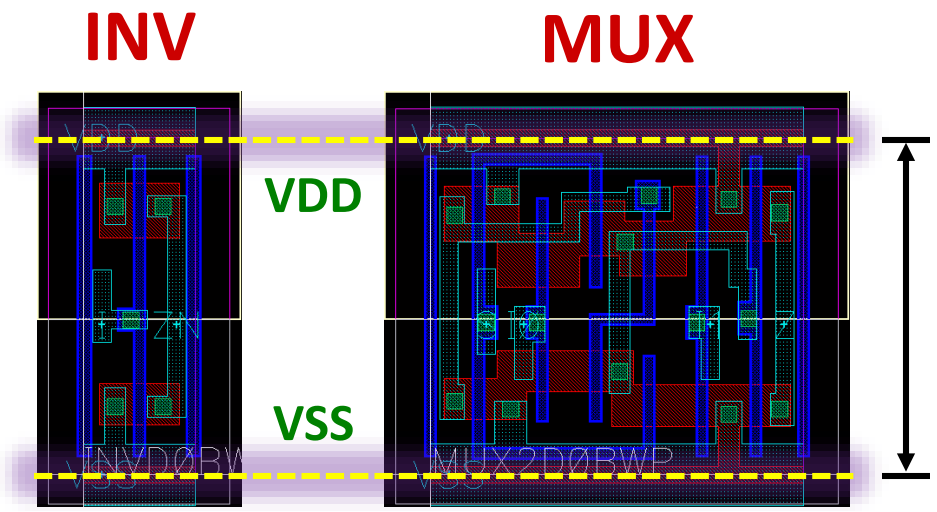
Chip Layout vs. Micrograph



How Tool Handles Large-Scale PnR (1)

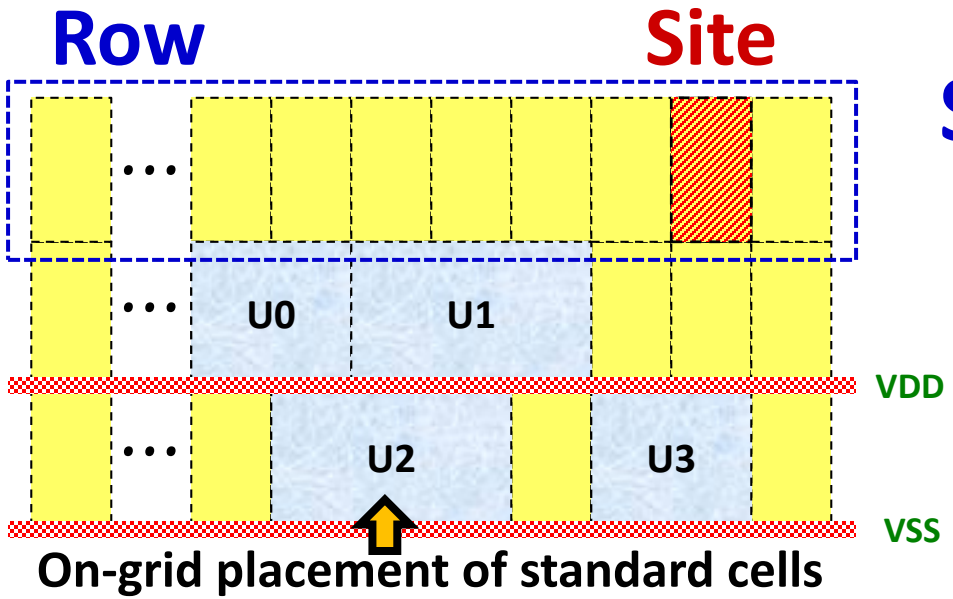
- **Today we're talking about billion-transistor designs**
- **Continuous design variables for circuit opt.: infeasible**
- **PnR technology: constrained, discrete optimization**
  - Library of standard cells
  - Site, Row-based placement
  - Fixed routing track & pitch
  - Layout exchange format (LEF)

PnR Technology (1)



Standard-Cell Library

- Same height
- Same power rail
- Discrete sizing (1×, 2×, 4×, …)

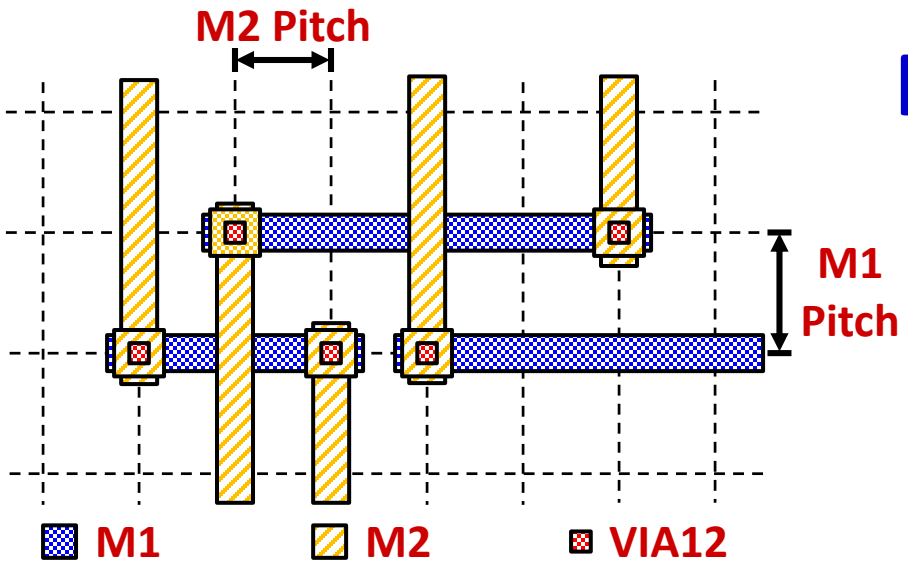


On-grid placement of standard cells

Site, Row-Based PnR

- Placement grid
- Easy VDD/VSS connection

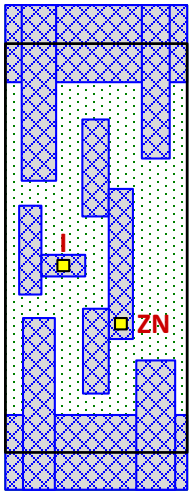
PnR Technology (2)



Fixed routing pitch

- On-grid routing
- Lowering complexity of routing algorithm

```
LAYER VTH_P  ... END VTH_P
LAYER VTH_N  ... END VTH_N
LAYER PO     ... END PO
LAYER CO     ... END CO
LAYER M1     ... END M1
LAYER VIA1   ... END VIA1
...
SITE core    ... END core
SITE bcore   ... END bcore
...
MACRO INVD0BWP ... END INVD0BWP
MACRO DFQD2BWP ... END DFQD2BWP
```



Layout Ex. Format

- Process info. (layer, design rule, par. RC)
- Simplified view of std. cells & macros

LEF: Technology Information

<item> <unit> <precision>

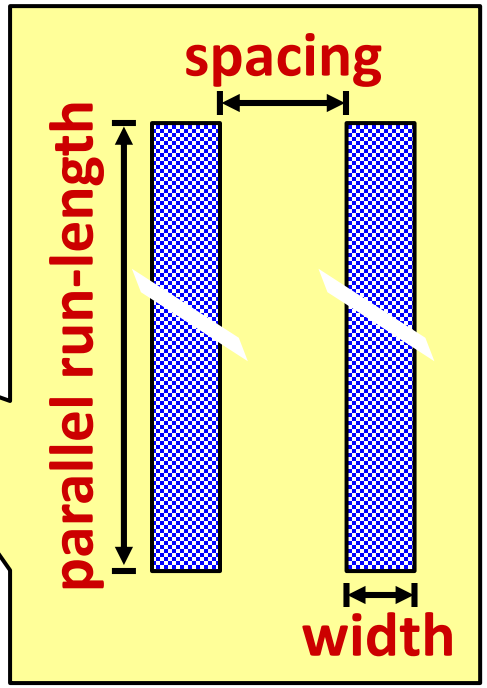
routing direction & pitch

3D geometry

```
UNITS
   CAPACITANCE PICOFARADS 10 ;
   CURRENT MILLIAMPS 10000 ;
   VOLTAGE VOLTS 1000 ;
   FREQUENCY MEGAHERTZ 10 ;
   DATABASE MICRONS 2000 ;
END UNITS

MANUFACTURINGGRID 0.005 ;

SITE bcore
  SIZE 0.140 BY 2.520 ;
  CLASS CORE ;
  SYMMETRY Y ;
END bcore

SITE core
  SIZE 0.140 BY 1.260 ;
  CLASS CORE ;
  SYMMETRY Y ;
END core
...
```

site def.

```
LAYER M2
  TYPE ROUTING ;
  DIRECTION VERTICAL ;
  PITCH 0.14 ;
  HEIGHT 0.535 ; THICKNESS 0.125 ;
  WIDTH 0.07 ; MAXWIDTH 4.50 ;
  SPACINGTABLE      PARALLELRUNLENGTH
                    0.00 0.27 0.40 0.62 1.50
  WIDTH        0.00 0.07 0.07 0.07 0.07 0.07
  WIDTH        0.17 0.07 0.08 0.08 0.08 0.08
  WIDTH        0.24 0.07 0.12 0.12 0.12 0.12
  WIDTH        0.31 0.07 0.12 0.14 0.14 0.14
  WIDTH        0.62 0.07 0.12 0.14 0.21 0.21
  WIDTH        1.50 0.07 0.12 0.14 0.21 0.50 ;
  ...
  MINENCLOSEDAREA 0.2 ;
  ANTENNAMODEL OXIDE1 ;
  ...
  RESISTANCE RPERSQ      0.3080000000 ;
  CAPACITANCE CPERSQDIST 0.0002317460 ;
END M2
...
```

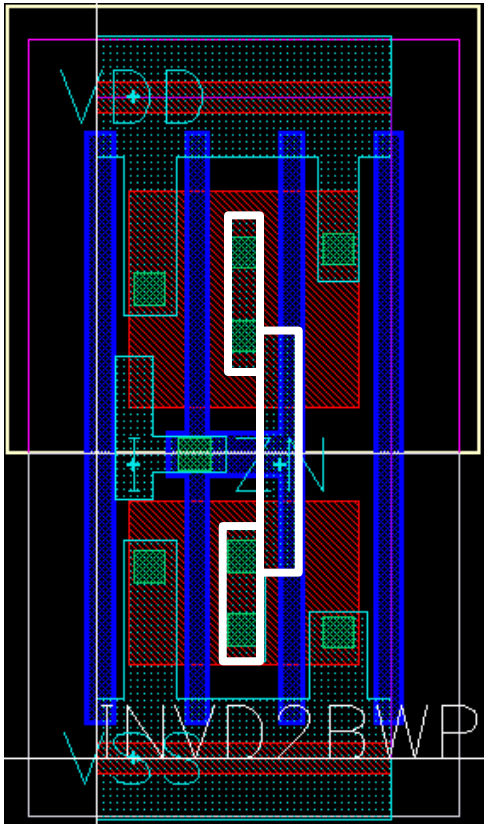


Q: Where to place it? A: Only in the core region.
Q: Can tool flip it? A: Yes, but only in Y direction.

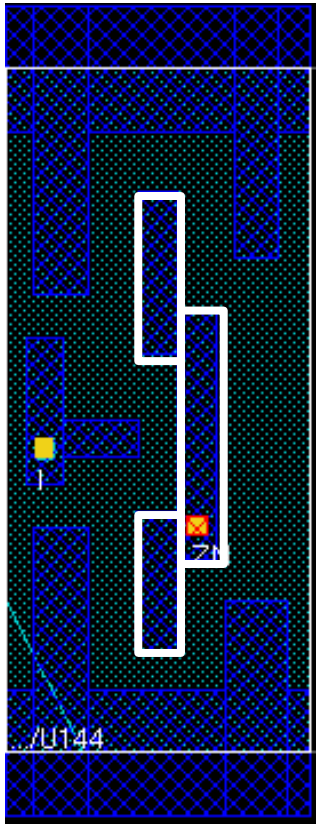
design-rule check info.
& parasitic RC values

LEF: Simplified Cell View (1)

PnR tool doesn't care about detailed layout within a cell, it only cares about the shapes of IO pins and VDD/VSS. Simplifying the layout of cells: improved tool runtime



Simplify



No more poly, N-well, contact, N/P doping…

Only IO and VDD/VSS left for tool to route

like building a road in front of your home

You don’t provide internal layout of your home to workers.

You only tell them where the doors are.

LEF: Simplified Cell View (2)

cell name

placed in the core region

can be flipped both vertically & horizontally

placed on the grid defined by SITE core

RECT x1 y1 x2 y2 to describe the shape

Antenna information for design-rule check (DRC) & SI analysis (discuss later)

USE POWER/GROUND tells the tool where the power pins are

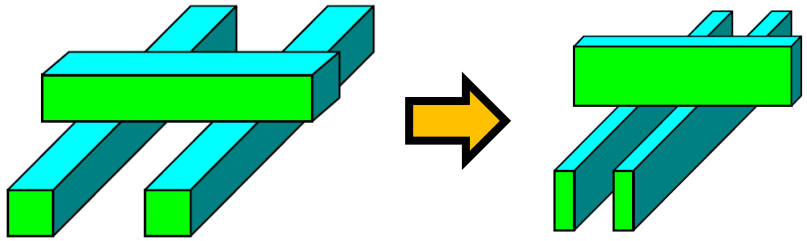
```
MACRO INVD2BWP
  CLASS CORE ;
  FOREIGN INVD2BWP 0.000 0.000  ;
  ORIGIN 0.000 0.000 ;
  SIZE 0.560 BY 1.260 ;
  SYMMETRY X Y ;
  SITE core ;
  PIN ZN
    ANTENNADIFFAREA 0.1008 ;
    DIRECTION OUTPUT ;
    PORT
    LAYER M1 ;
    RECT  0.320 0.355 0.385 0.815 ;
    RECT  0.315 0.185 0.320 0.815 ;
    RECT  0.240 0.185 0.315 0.445 ;
    RECT  0.245 0.735 0.315 1.035 ;
    END
  END ZN
  PIN I
    ANTENNAGATEAREA 0.0576 ;
    DIRECTION INPUT ;
    LAYER M1 ;
    RECT  0.105 0.545 0.245 0.615 ;
    RECT  0.035 0.495 0.105 0.765 ;
    END
  END I
  PIN VDD
      DIRECTION INOUT ;
      USE POWER ;
      SHAPE ABUTMENT ;
      PORT
      LAYER M1 ;
      RECT  0.500 1.145 0.560 1.375 ;
      RECT  0.420 0.910 0.500 1.375 ;
      RECT  0.150 1.145 0.420 1.375 ;
      RECT  0.050 0.845 0.150 1.375 ;
      RECT  0.000 1.145 0.050 1.375 ;
      END
    END VDD
  PIN VSS
      DIRECTION INOUT ;
      USE GROUND ;
      SHAPE ABUTMENT ;
      PORT
      LAYER M1 ;
      RECT  0.515 -0.115 0.560 0.115 ;
      RECT  0.405 -0.115 0.515 0.280 ;
      RECT  0.150 -0.115 0.405 0.115 ;
      RECT  0.050 -0.115 0.150 0.415 ;
      RECT  0.000 -0.115 0.050 0.115 ;
      END
    END VSS
  END INVD2BWP
```

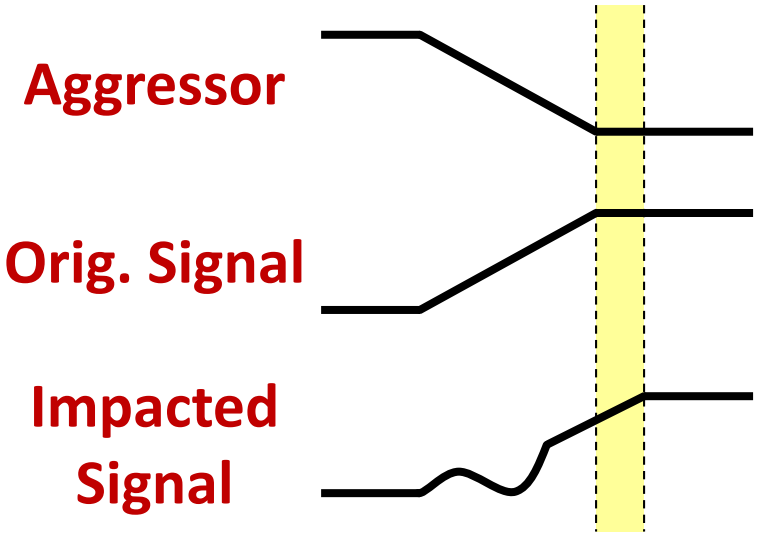
Timing & Noise Analysis

- **Static timing analysis (propagation delay)**
  - Cell timing library (.lib)
- **Parasitic extraction (wire delay)**
  - Capacitance table (.captab)
  - QX technology file (.cl)
- **Signal integrity (crosstalk, antenna effect)**
  - CeltIC library (.cdb)

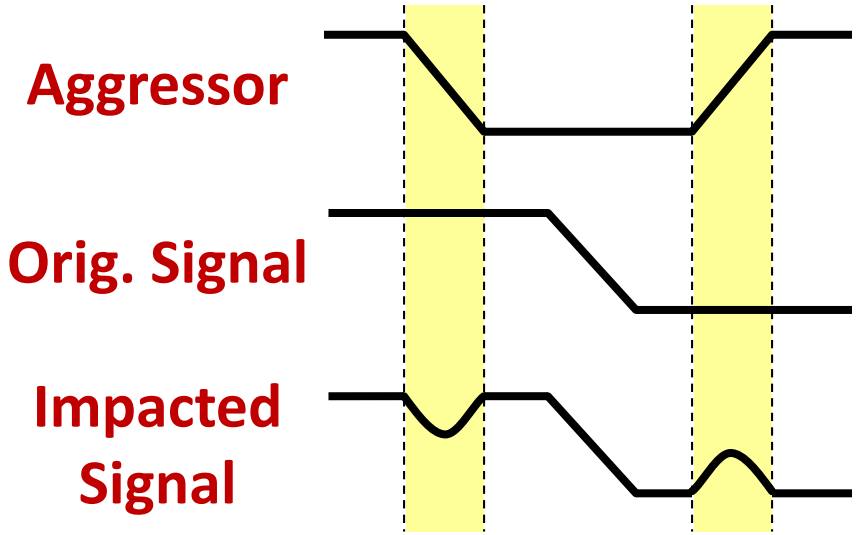
Crosstalk

- **Getting serious in deep-submicron era**
  - Smaller pitches
  - Greater height/width ratio
  - Higher clock frequency





Delay Problem



Noise Problem

Crosstalk Prevention

- **Placement solution**
  - Insert buffer in lines
  - Upsize driver
- **Routing solution**
  - Congestion optimization
  - Limit length of parallel nets
  - Wider wire spacing
  - Shield special nets (e.g. clock)

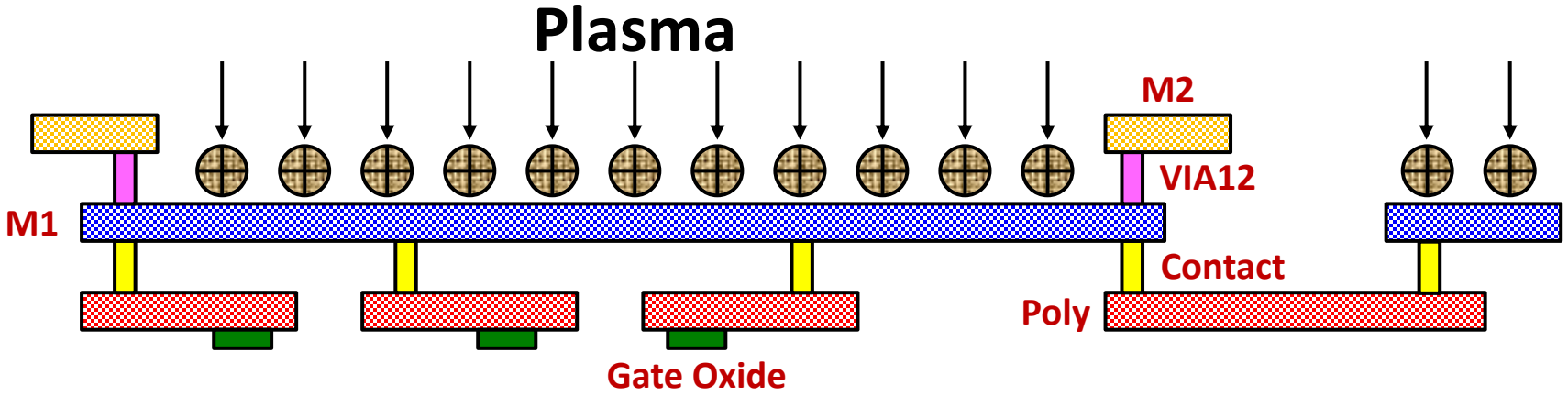
Antenna Effect

During chip fabrication, metals are initially deposited, covering the entire chip.

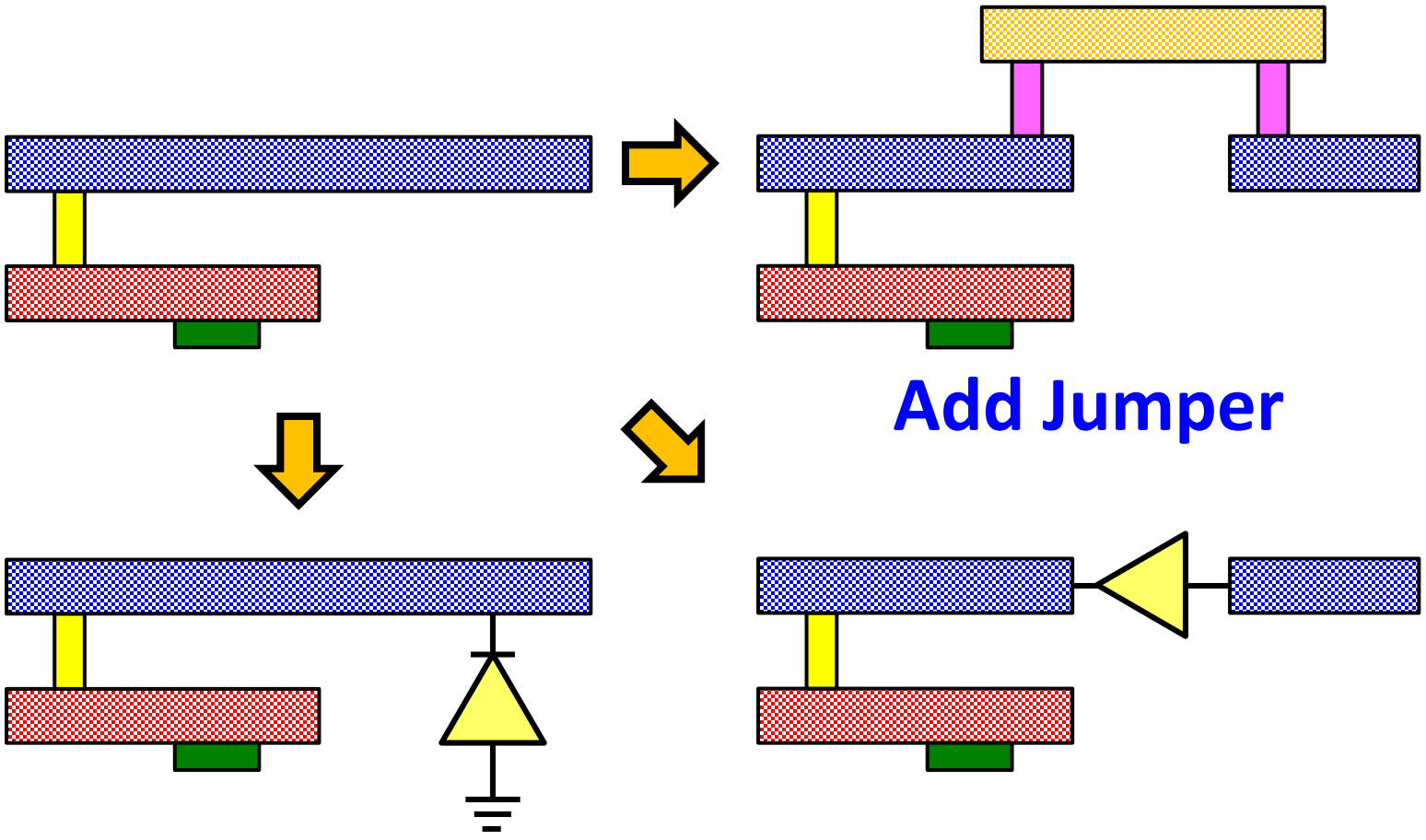
Unneeded portions of the metal are removed by etching, typically using plasma (charged particles).

Exposed metal collects charges, forming voltage potential.

Might be large enough to damage the gate oxide.

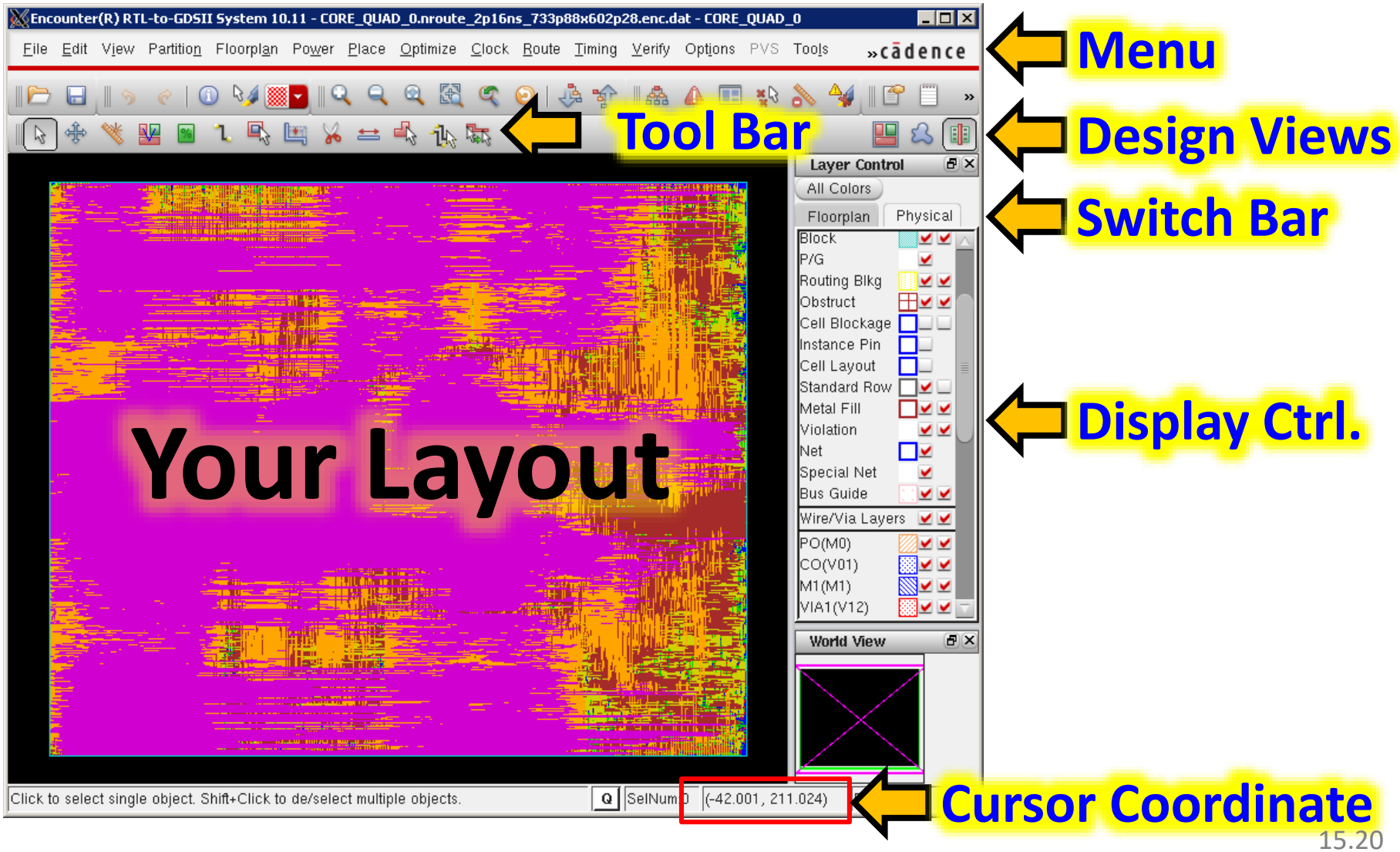


Fixing Antenna Problem

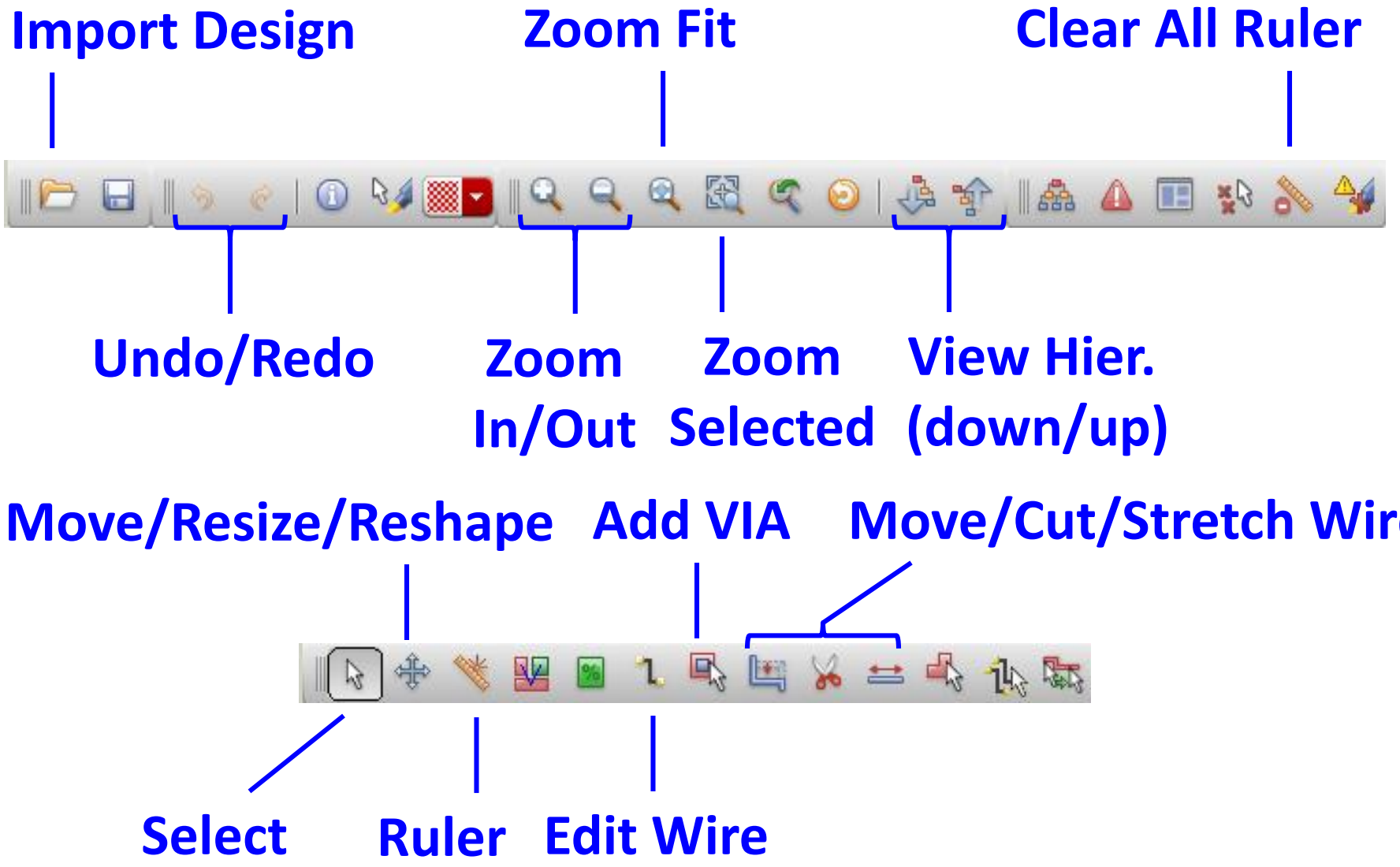


Commercial PnR Tool: SoC Encounter

Hybrid GUI-Script Interface



Tool Bar


Import Design
Zoom Fit
Clear All Ruler
Undo/Redo
Zoom In/Out
Zoom Selected
View Hier. (down/up)
Move/Resize/Reshape
Add VIA
Move/Cut/Stretch Wire
Select
Ruler
Edit Wire

Design Views

Amoeba View

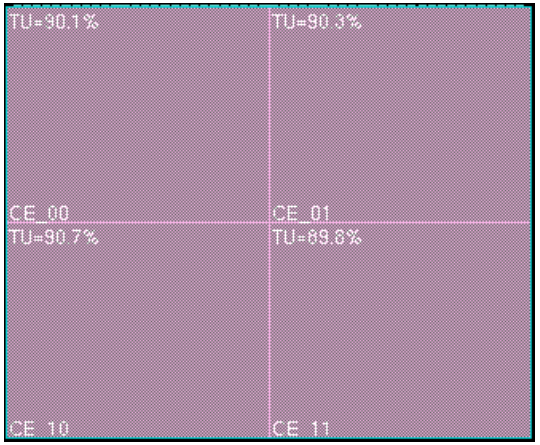
Display the outline of modules after placement

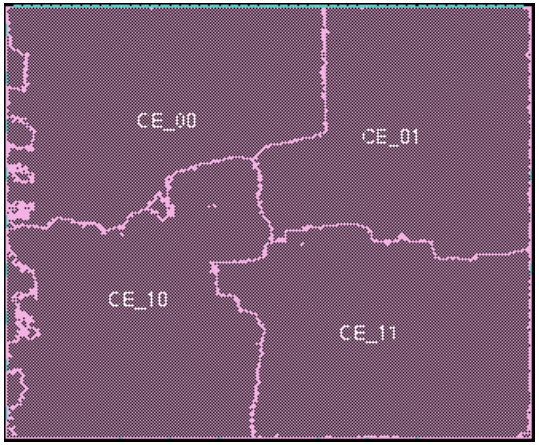
Floorplan View

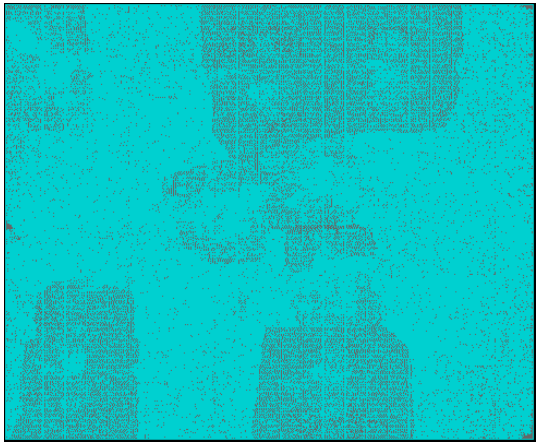
Show hierarchical module block guides, connection lines & floorplan objects

Placement View

Display the detailed placement of cells and macro blocks



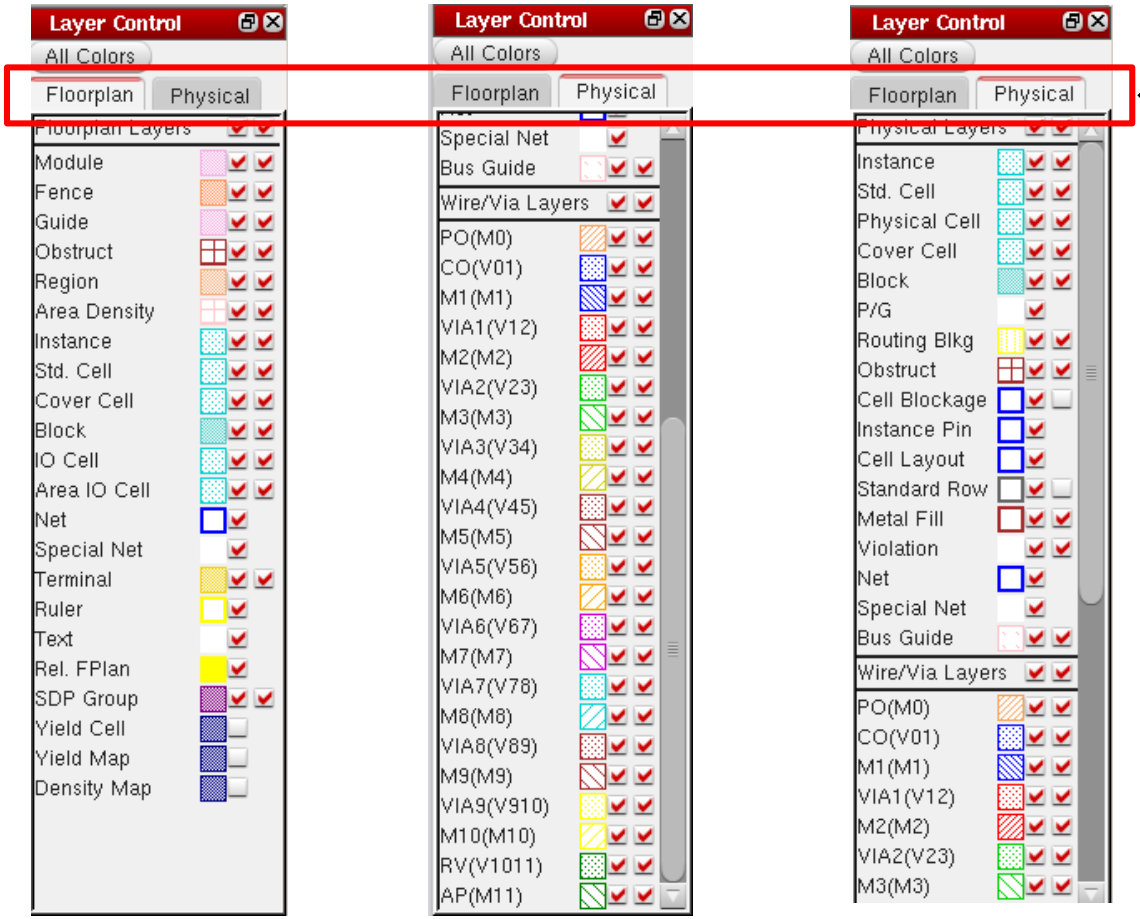




Display Control

Color Display: Floorplan Objects & Physical Layers

(can be customized by yourself)



Switch Bar b/w F-plan & Physical

Useful Bindkeys

Key	Action
q	Edit Attribute
f	Zoom Fit
z	Zoom In
Z	Zoom Out
k	Create Ruler
Shift + k	Delete All Ruler

Key	Action
space	Select Next
e	Popup Edit
T	Edit Trim
0-9	Toggle layer [0-9]
h/H	View Hier. (Up/Down)
x	Clear DRC

More bindkeys:

Options → Set Preference → Binding Key

SoC Encounter: General PnR Steps

- **File preparation & initial setup**
- **Floor-planning**
  - Specify layout size & power rail names
  - Manually place hard macros (e.g. SRAM blocks, if any)
  - Add power ring & power stripes
  - Special route (connect cells' VDD/VSS to power ring)
- **Placement & pre-CTS optimization**
- **Clock-tree synthesis (CTS) & post-CTS optimization**
  - Generate clock specs. & synthesize clock tree
  - Display clock and optimize design
- **Detailed routing & post-route opt. (timing & SI driven)**
- **Layout verification & stream out**

SoC Encounter: General PnR Steps

- **File preparation & initial setup**
- **Floor-planning**
  - Specify layout size & power rail names
  - Manually place hard macros (e.g. SRAM blocks, if any)
  - Add power ring & power stripes
  - Special route (connect cells' VDD/VSS to power ring)
- **Placement & pre-CTS optimization**
- **Clock-tree synthesis (CTS) & post-CTS optimization**
  - Generate clock specs. & synthesize clock tree
  - Display clock and optimize design
- **Detailed routing & post-route opt. (timing & SI driven)**
- **Layout verification & stream out**

File Preparation

You Generate

- Gate-level netlist (.vg) — from synthesis result
- Design constraints (.sdc) — from synthesis result
- Multi-Mode Multi-Corner (MMMC) view (.view) — let SoCE generate template (only the 1st time), then manually edit
- IO assignment file (.io) — Same ways as above
- Configuration file (.conf) — Same ways as above
- Clock specs (.clocktch) — Same ways as above

You Find from Foundry Design Kit

- Timing library (.lib)
- Layout exchange format (.lef)
- Capacitance table (.captab)
- Parasitic extraction tech file (.cl)
- CeltIC noise analysis model (.cdb)
- Stream-out layer map (.map)

Slight Modification of .vg

- **When we're at the final chip-level integration stage, we need to connect top-level module IO pins to the chip IO pads to communicate with outside environment**
  - Create a new module (e.g. **CHIP**)
  - Instantiate your top-level module in module **CHIP**
  - Find names of IO pad modules in process verilog model
  - Manually instantiate the IO pad modules, connect it with top-level modules IO

Find IO pads' names in process verilog file

```
module FullAdd(S, A, B) ;
  output S;
  input A;
  input B;
  FADDX1  U00 (S, A, B);
endmodule
```

+

```
module CHIP (CHIP_S,
            CHIP_A, CHIP_B) ;
  output CHIP_S ;
  input CHIP_A ;
  input CHIP_B ;
  wire A, B ;
```

```
  wire A, B ;
  FullAdd TopLvl(S, A, B) ;
  PDO08CDG  O1(.Z(CHIP_S), .I(S)) ;
  PDIDGZ  I0(.Z(A), .I(CHIP_A)) ;
  PDIDGZ  I1(.Z(B), .I(CHIP_B)) ;
endmodule
```

Slight Modification of .sdc

- **Synthesis only considers propagation delay of cells. It doesn't include accurate wire delay information.**
- **We can't use same clock period as in synthesis for PnR.**
  - Suggest: increase 0.2~0.4ns to accommodate parasitic
- **I/O delay and the three major design rules (max-fanout, max-transition, max-cap.) might need modification too.**

IO Assignment File

Method 1: Follow the format below, store as **.io** file, then do **File → Load → I/O File**

Method 2: Do **Edit → Pin Editor** to assign pins after your design is completely loaded. Save it (**File → Save → I/O File**) for future use

```
(globals
   version = 3
   io_order = default
)
(iopin
   (top
      (pin name="xx" offset=xx layer=xx width=xx depth=xx )
      ...
   )
   (bottom
   )
 (left
   )
 (right
   )
```

pin location (in µm) r.p.t the origin of the edge

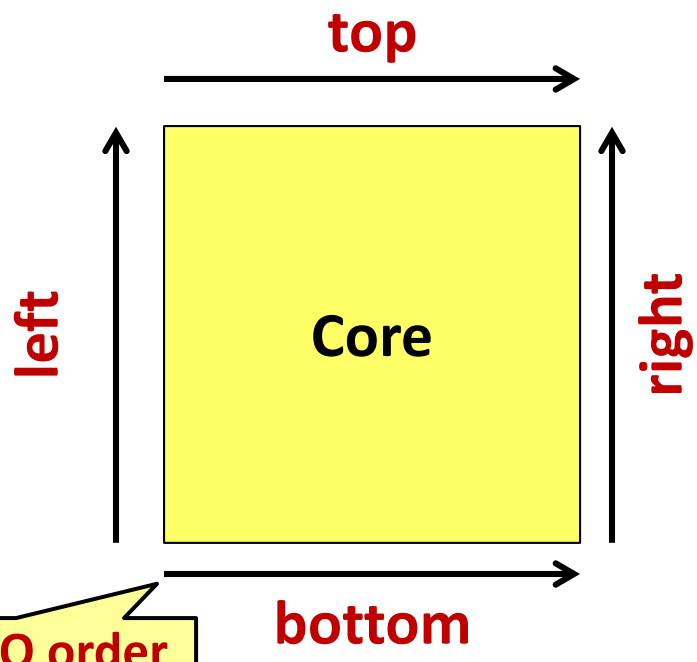
pin name

metal layer (e.g. 5) to place this pin

pin width (in µm)

pin length (in µm)

default I/O order



Getting Started!

Login *eeapps.seas.ucla.edu*

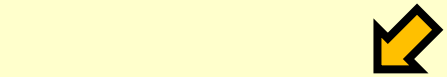
Enter *source /usr/apps/cadence/SETUP.EDI10.11*

Enter *encounter* (32-bit version) to open
(note: don't enter *encounter &*)

MMMC View Files (1)

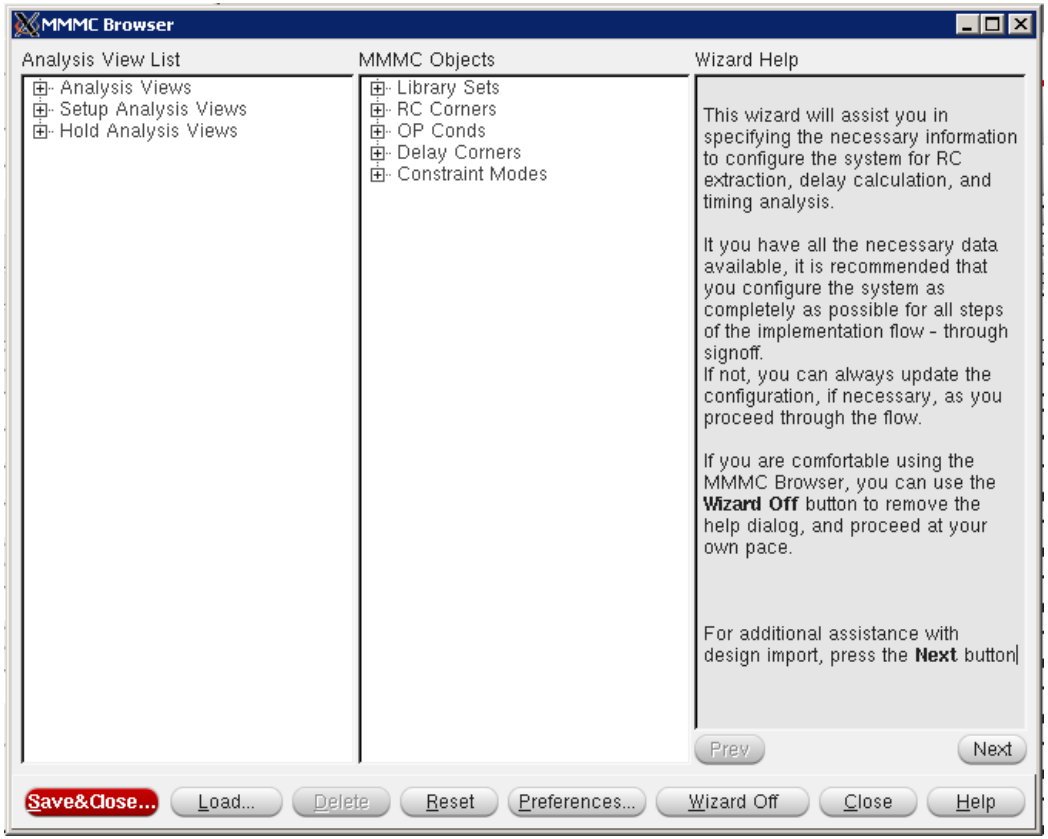
Method 1: Follow the format below, store as **.view** file, then source it in **.conf** file

```
# Version:1.0 MMMC View Definition File
# Do Not Remove Above Line
create_rc_corner   -name rc_corner_setup -cap_table {.captab file} -qx_tech_file {.cl file}
create_rc_corner   -name rc_corner_hold  -cap_table {.captab file} -qx_tech_file {.cl file}
create_op_cond     -name op_cond_setup   -library_file {.lib file}
create_op_cond     -name op_cond_hold    -library_file {.lib file}
create_library_set -name library_setup   -timing {.lib file} -si {.cdb file}
create_library_set -name library_hold    -timing {.lib file} -si {.cdb file}
create_constraint_mode -name SDC_setup   -sdc_files ".sdc file name"
create_constraint_mode -name SDC_hold    -sdc_files ".sdc file name"
create_delay_corner -name setup -library_set {library_setup} -rc_corner {rc_corner_setup}
create_delay_corner -name hold  -library_set {library_hold}  -rc_corner {rc_corner_hold}
create_analysis_view -name setup -constraint_mode {SDC_setup} -delay_corner {setup}
create_analysis_view -name hold  -constraint_mode {SDC_hold } -delay_corner {hold}
set_analysis_view -setup {setup} -hold {hold}
```



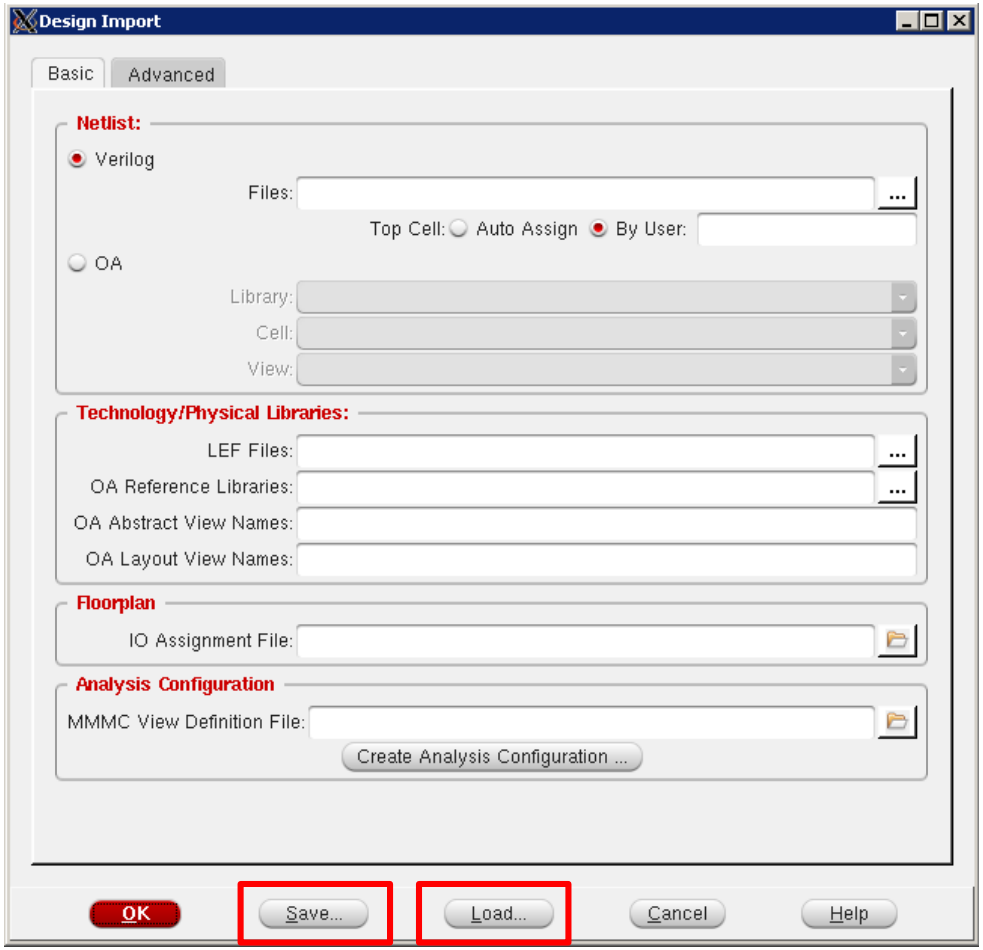
MMMC View Files (2)

Method 2: Do **File → Import Design → Create Analysis Configuration** and follow the wizard. Save it as **.view** file for future use



Initial Setup (1)

File → Import Design



For the first time, press **Save** to create an empty .conf file

↓

Edit .conf to include required files (.vg, .sdc, .lef, .view)

↓

Also specify core utilization and CTS buffer list in .conf

↓

Save it, then press **Load** to import the modified .conf

Initial Setup (2)

```
global rda_Input
set cwd your_working_directory        ⇦ Modify
set rda_Input(import_mode)
    {-treatUndefinedCellAsBbox 0 -keepEmptyModule 1 }
set rda_Input(ui_netlist) ".vg file"
set rda_Input(ui_netlisttype) {Verilog}
...
set rda_Input(ui_settop) {1}
set rda_Input(ui_topcell) "top_module_of_your_design"
...
set rda_Input(ui_view_definition_file) ".view file"
set rda_Input(ui_timingcon_file,full) ".sdc file"
set rda_Input(ui_leffile) ".lef file"
set rda_Input(ui_cts_cell_list)
{check_standard_cell_library_manual_to_list_the_buffers
_you_want_for_PnR_tool_to_do_clock_tree_synthesis}
set rda_Input(ui_core_cntl) {aspect}
set rda_Input(ui_aspect_ratio) {aspect_ratio_you_want}
set rda_Input(ui_core_util) {utilization_you_want}
...
set rda_Input(ui_row_height)
    {please_check_lef_file_for_the_height_of_std_cell}
set rda_Input(ui_isHorTrackHalfPitch) {0}
set rda_Input(ui_isVerTrackHalfPitch) {1}
set rda_Input(ui_ioOri) {R0}
```

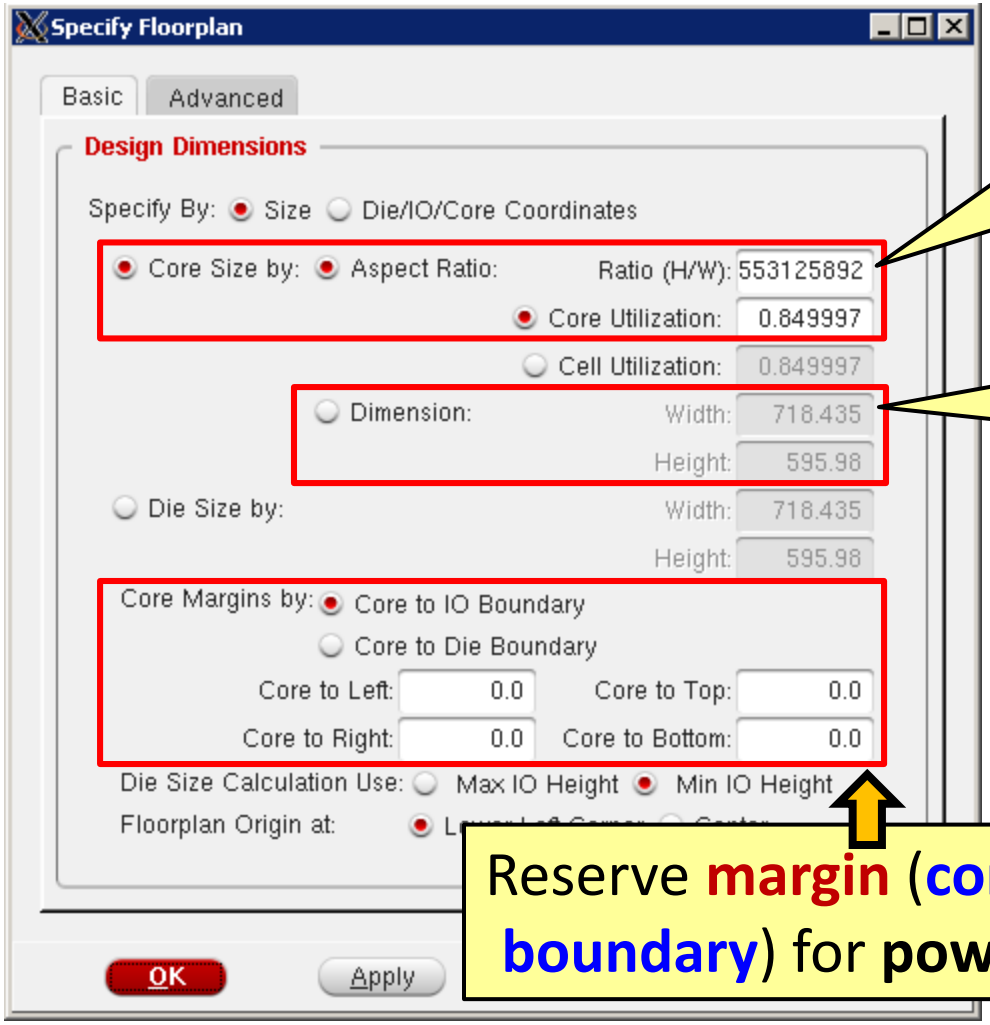
```
set rda_Input(ui_isOrigCenter) {0}
set rda_Input(ui_isVerticalRow) {0}
set rda_Input(ui_delay_limit) {1000}
set rda_Input(ui_net_delay) {1000.0ps}
set rda_Input(ui_net_load) {0.5pf}
set rda_Input(ui_in_tran_delay) {0.0ps}
...
set rda_Input(ui_preRoute_cap) {1}
set rda_Input(ui_postRoute_cap) {1}
set rda_Input(ui_postRoute_xcap) {1}
set rda_Input(ui_preRoute_res) {1}
set rda_Input(ui_postRoute_res) {1}
set rda_Input(ui_shr_scale) {1.0}
set rda_Input(ui_rel_c_thresh) {0.03}
set rda_Input(ui_tot_c_thresh) {5.0}
set rda_Input(ui_cpl_c_thresh) {3.0}
set rda_Input(ui_time_unit) {none}
...
set rda_Input(ui_pwrnet) {VDD}
set rda_Input(ui_gndnet) {VSS}
set rda_Input(flip_first) {1}
set rda_Input(double_back) {1}
set rda_Input(assign_buffer)   {1}
set rda_Input(use_io_row_flow) {0}
set rda_Input(ui_gen_footprint) {0}
```

SoC Encounter: General PnR Steps

- File preparation & initial setup
- **Floor-planning**
  - Specify layout size & power rail names
  - Manually place hard macros (e.g. SRAM blocks, if any)
  - Add power ring & power stripes
  - Special route (connect cells' VDD/VSS to power ring)
- Placement & pre-CTS optimization
- Clock-tree synthesis (CTS) & post-CTS optimization
  - Generate clock specs. & synthesize clock tree
  - Display clock and optimize design
- Detailed routing & post-route opt. (timing & SI driven)
- Layout verification & stream out

Floor-planning

Floorplan → Specify Floorplan

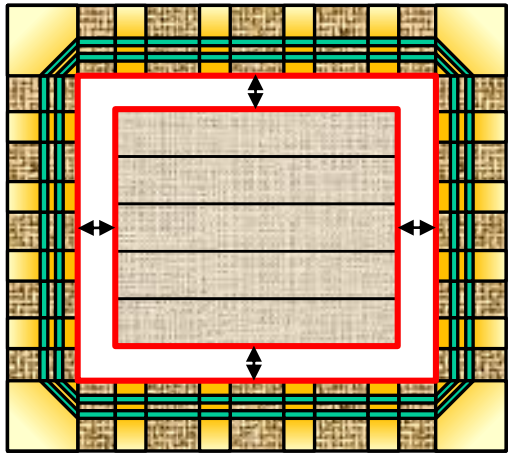


Specify core size *indirectly* using **aspect ratio** (**height by width**) & **utilization**

or

Specify core size *directly* using **dimension** (**H & W**)

Reserve **margin** (**core to IO boundary**) for **power ring**



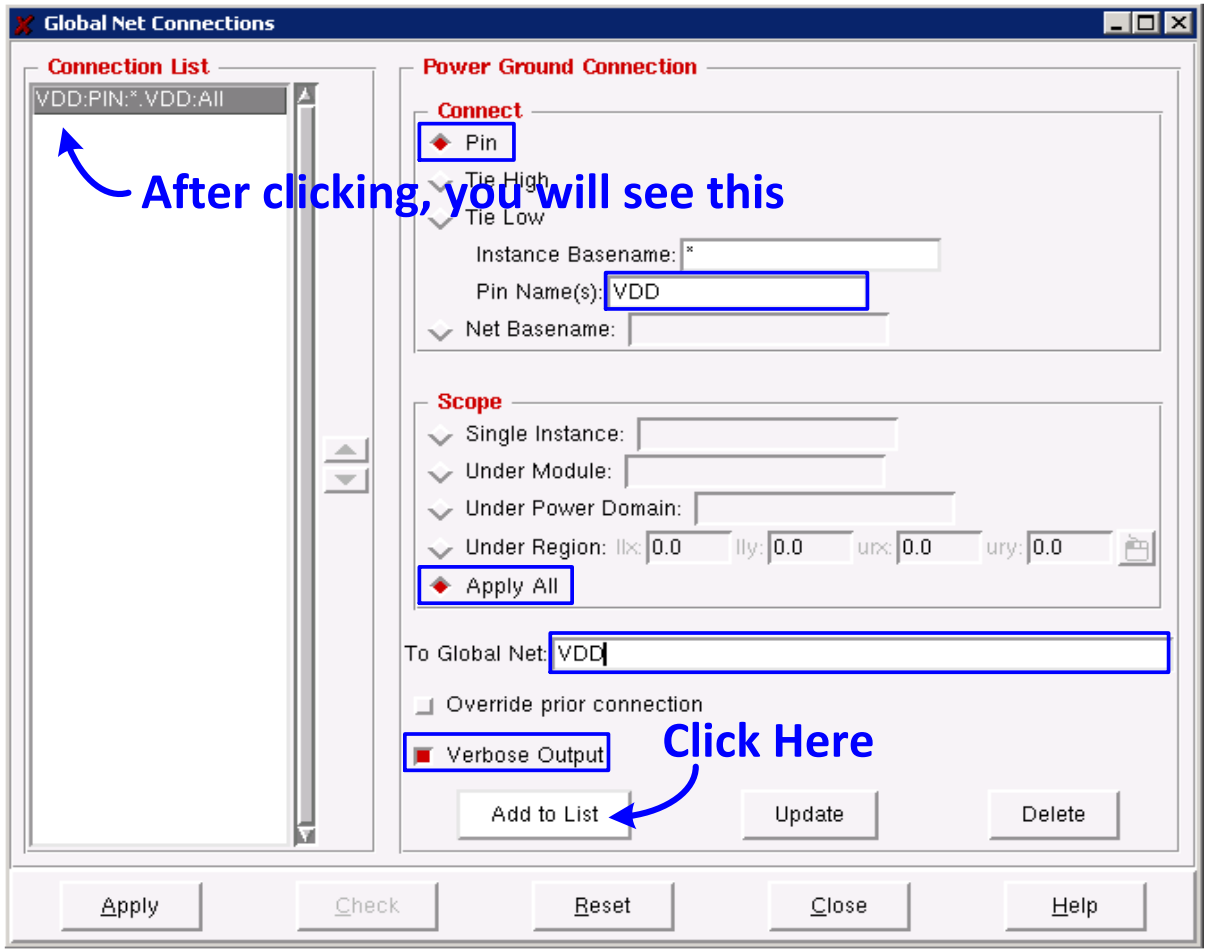
Mapping Cell Power Pins to Global Rail

Floorplan → Connect Global Nets

Connect **VDD pin** to global **VDD** net

Do same thing to connect **VSS**

You can see the message about # of **VDD/VSS pins** connect to global **VDD/VSS** in the terminal window



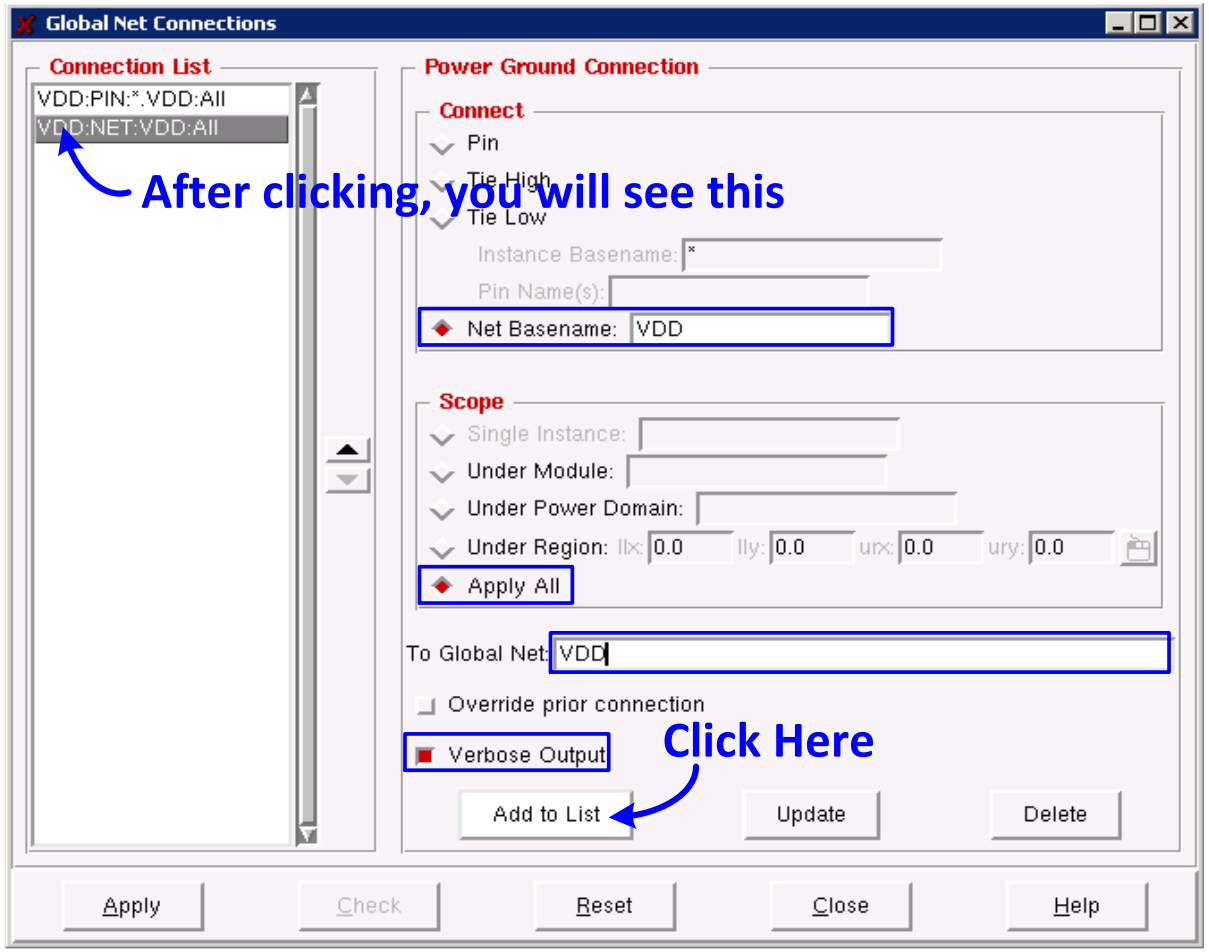
Mapping Power Nets to Global Rail

Floorplan → Connect Global Nets

Connect **VDD net** to global **VDD** net

Do same thing to connect **VSS**

You can see the message about # of **VDD/VSS nets** connect to global **VDD/VSS** in the terminal window



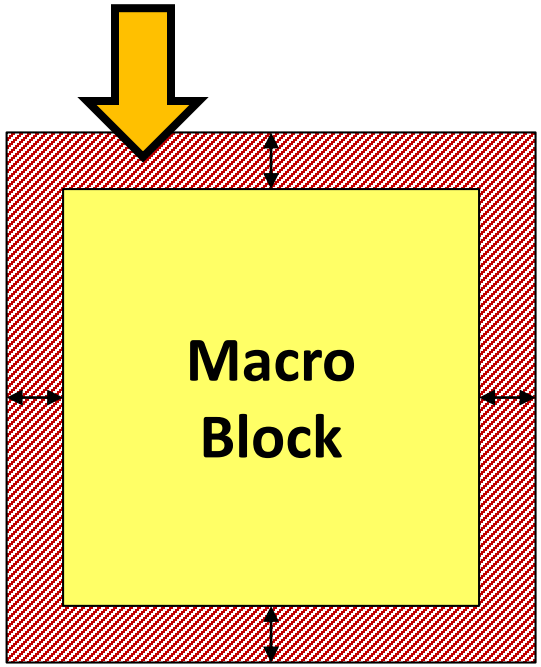
Manually Place/Edit Hard Macro (if wanted)

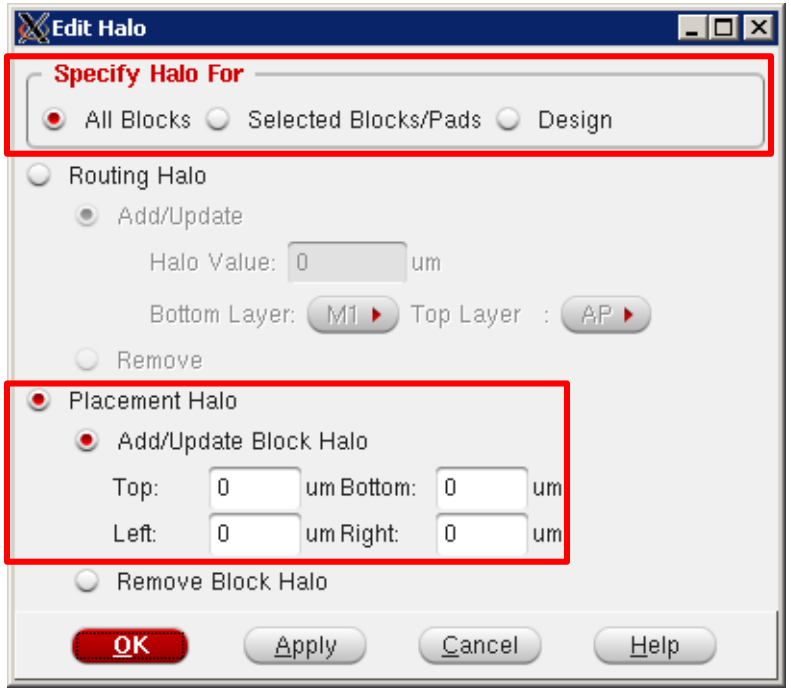
Use  to move objects to desired locations

Edit HALO:

Floorplan → Edit Floorplan → Edit HALO

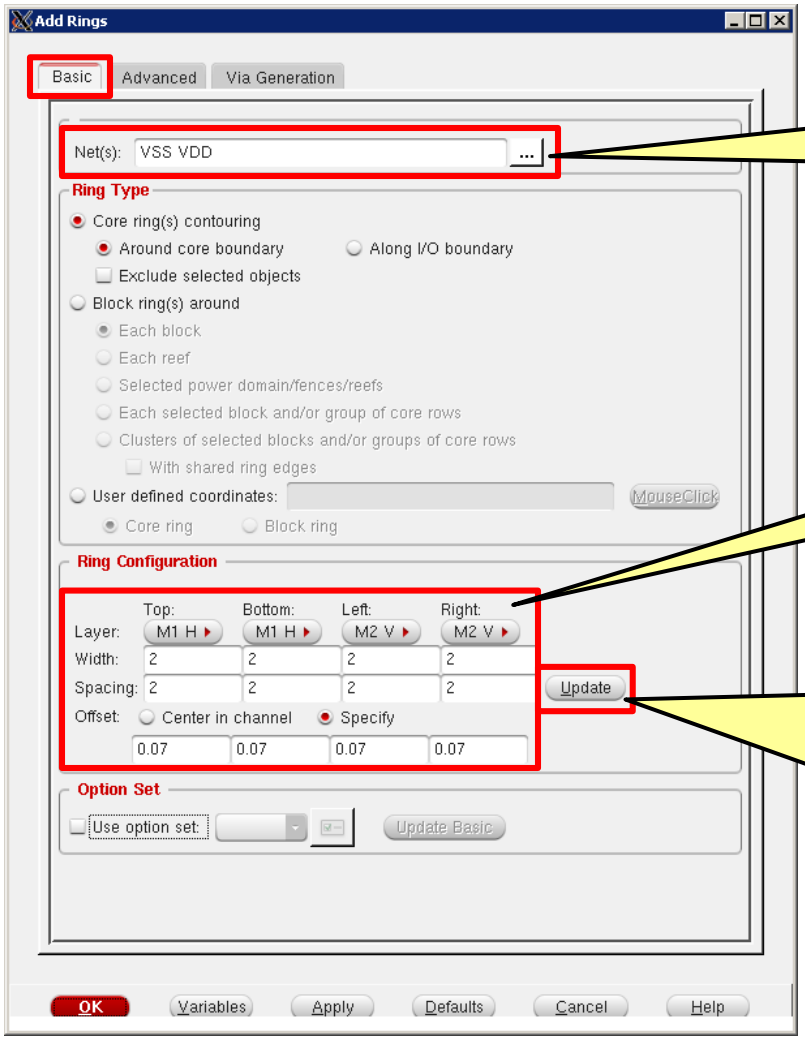
Prevent the placement of blocks & std. cells within specified **HALO** region in order to reduce congestion around a block.





Add Power Ring (1)

Power → Power Planning → Add Rings



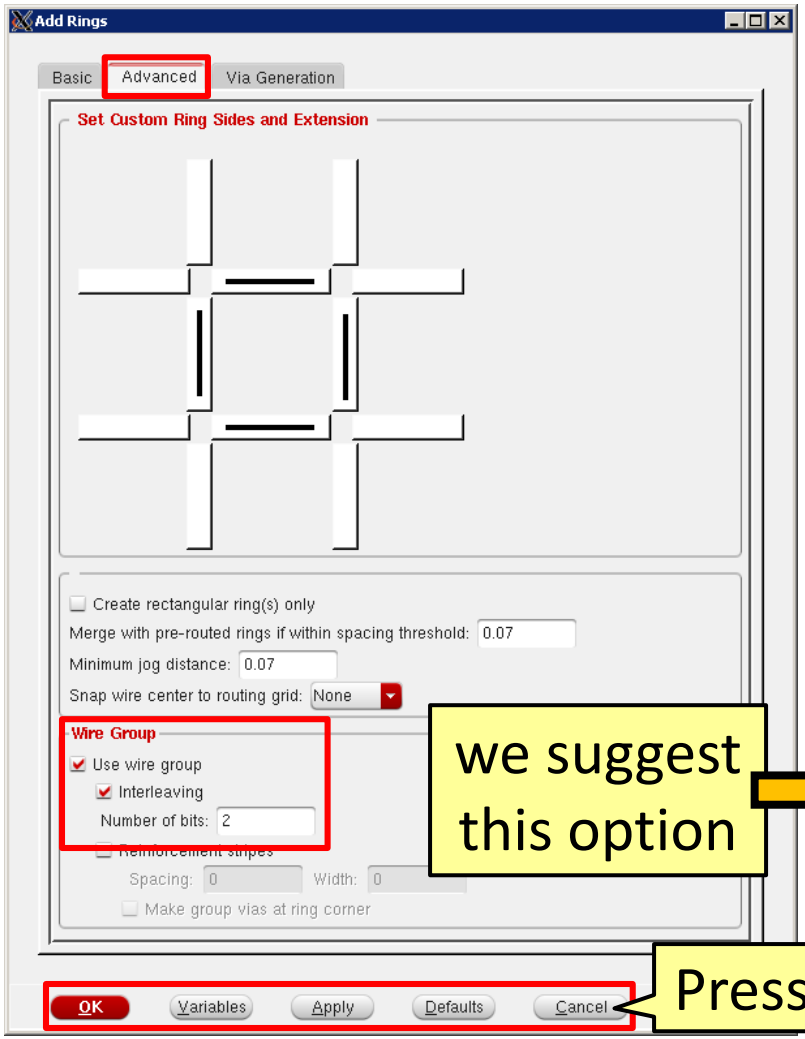
Specify the names of **global nets** to deliver **core power**

Specify **layer**, metal **width** & **spacing** of power ring

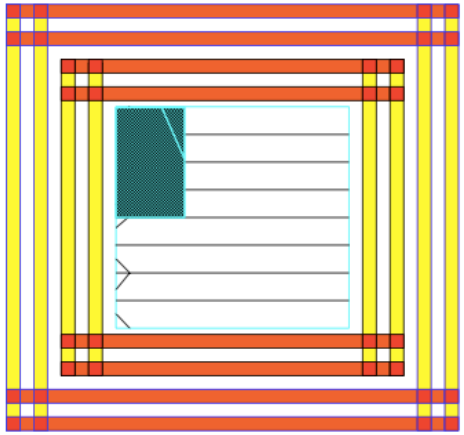
(note: after setting up a new value, press **Update** to have the tool adjust the values to the nearest legal ones that can pass DRC for you)

Add Power Ring (2)

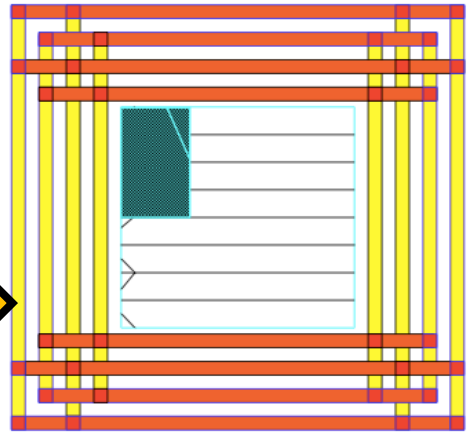
Power → Power Planning → Add Rings



we suggest this option



using wire group *w/o* interleaving, number of bits = 2



using wire group *with* interleaving, number of bits = 2

Press **OK**, or **Apply→Cancel** to add ring

How to Decide the Width of Power Ring?

Formula for metal width

$$\text{Width} = \frac{\text{Total Power}}{\text{Supply Voltage}} \times \frac{1}{4} \times \frac{1}{\text{Metal Current Density}}$$

- Rule-of-thumb current density ~= **4 mA/µm** (if >0.5µm & below 100°C). For exact value please check **process design-rule manual**.
- The ¼ is by assuming each edge delivers same amount of current

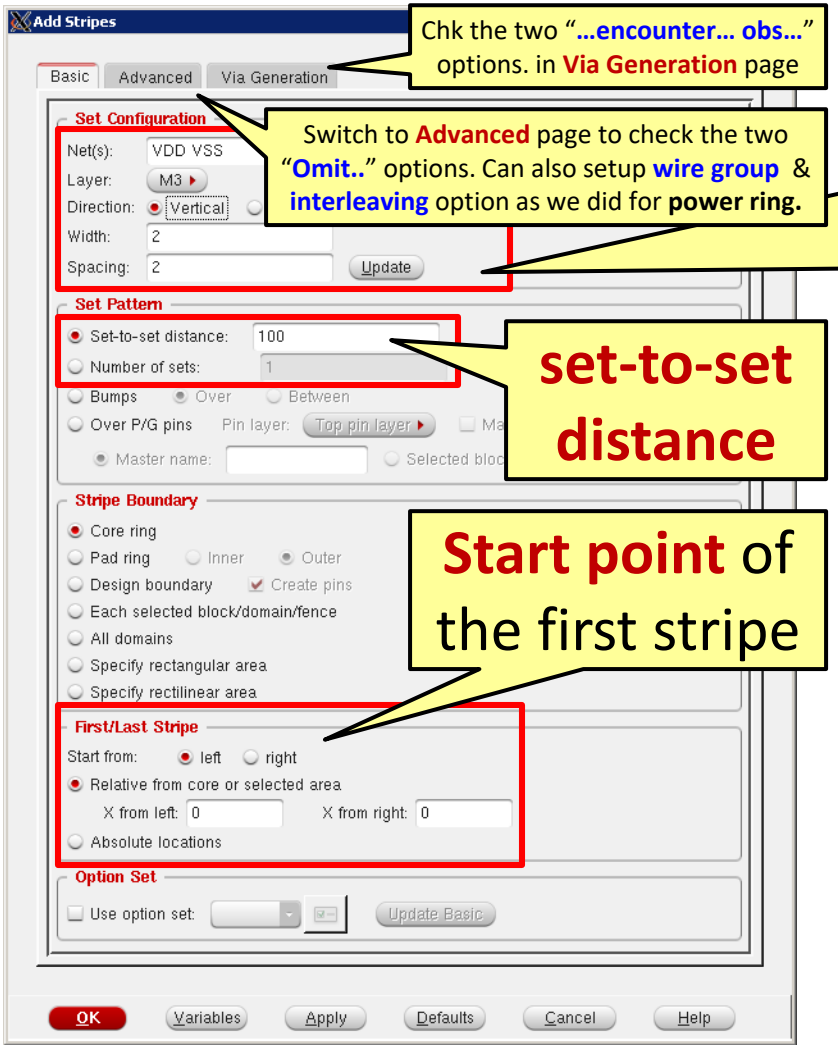
(Note: Above is only for V_{DD} or V_{SS}, should times 2 to get both. Also need to plus metal spacing to pass DRC)

Example: A post-layout core consumes **80 mW@1 Volt**. It uses M3 (top & bottom) and M4 (left & right) for power ring, each having density of **4mA/µm**. Design rule asks the spacing b/w metals to be **>0.5 µm**. We want a wire-group interleaved by **2**.

Solution: Total width of V_{DD} or V_{SS} = 0.25×(80mW/1Volt)/(4mA/µm) = 5µm. Total power ring width = 5µm×2(for V_{DD}+V_{SS}) + 3×0.5µm(spacing inside ring, as shown in previous page) + 2×0.5µm(spacing b/w ring & core+IO pads) = 12.5µm. This also tells you should specify 12.5µm spacing on each side of the core during floorplan. (Note: Due to 2-bit wire group, the width for each V_{DD} or V_{SS} is in fact 5µm/2=2.5µm)

Add Stripe to Enhance Power Delivery

Power → Power Planning → Add Stripes



Chk the two "**...encounter... obs...**" options. in **Via Generation** page

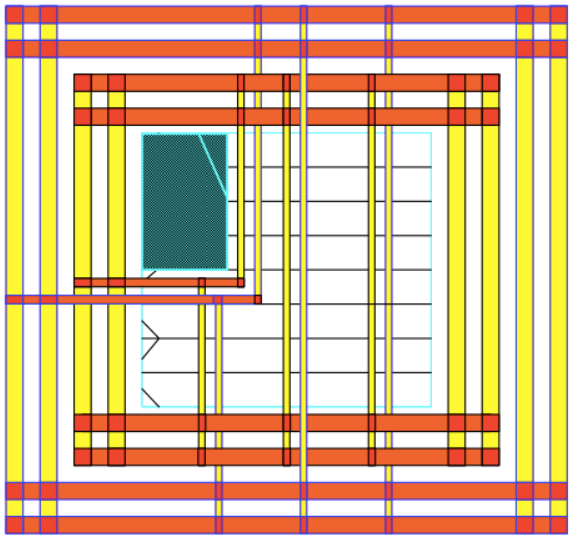
Switch to **Advanced** page to check the two "**Omit..**" options. Can also setup **wire group** & **interleaving** option as we did for **power ring.**

Specify the stripe information (**metal layer** it uses, **direction**, **width** and **spacing**). **Update** to get DRC-legal values.

set-to-set distance

Start point of the first stripe

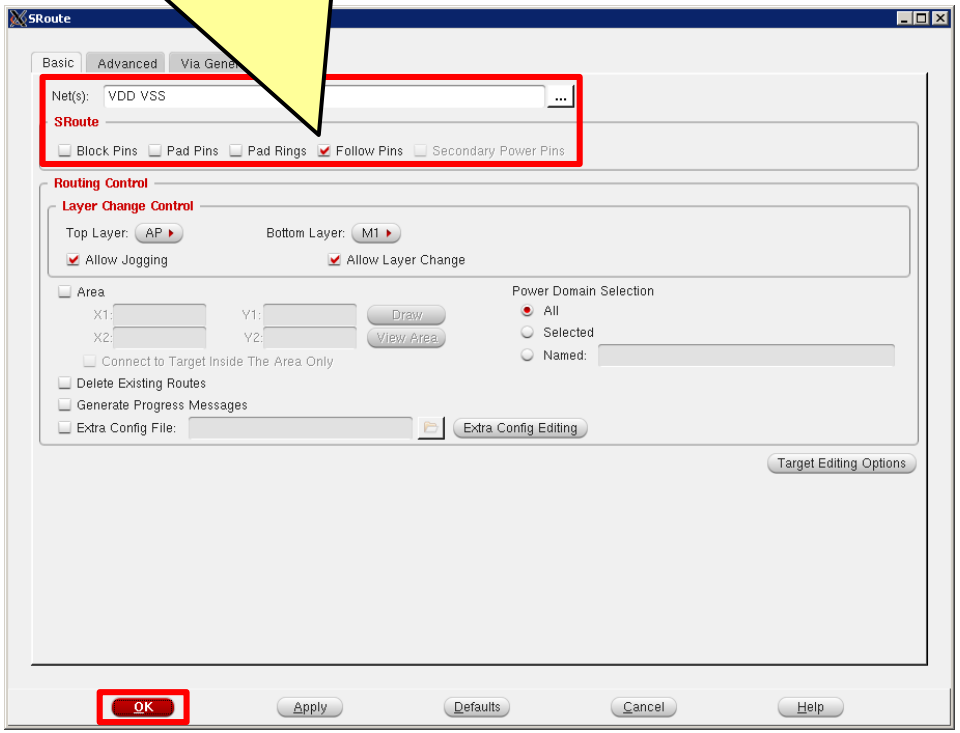
Tip: We usually only put **vertical stripes** using **M2** or **M4** because std cells' **VDD/VSS** rail naturally serve as **horizontal stripes.**

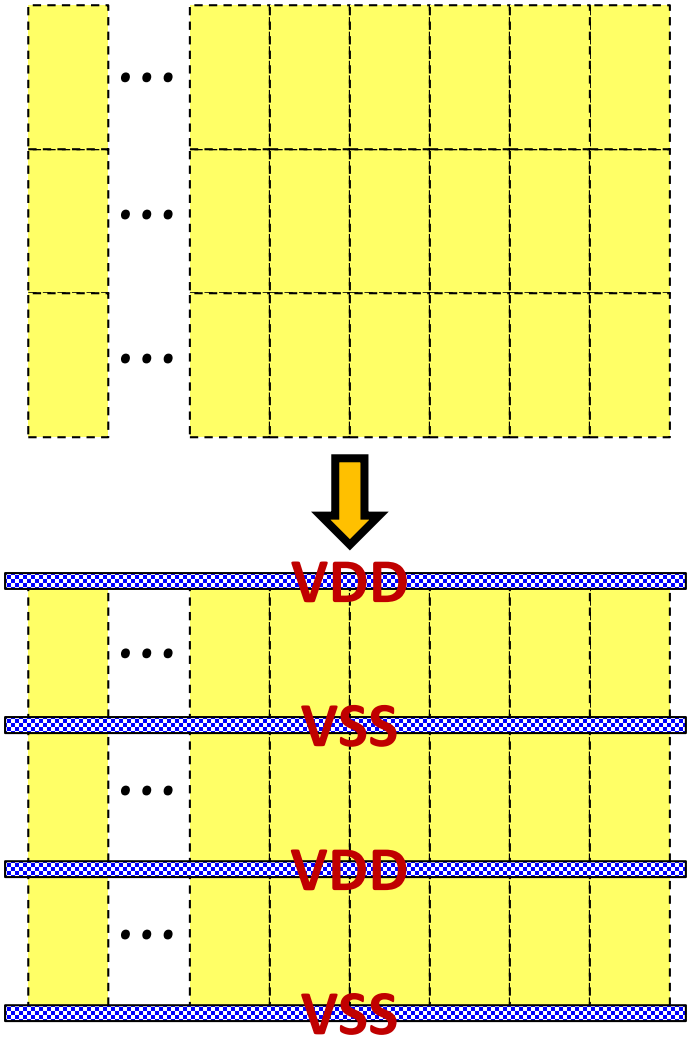


Special Route: Connect Std. Cell Power Pins

Route → Special Route

Put **VDD VSS** in **Net**, check **Follow Pins**, then press **OK**.
(change **SRoute** options to connect **Pad Pins**, **Block Pins** and **Pad Rings**)



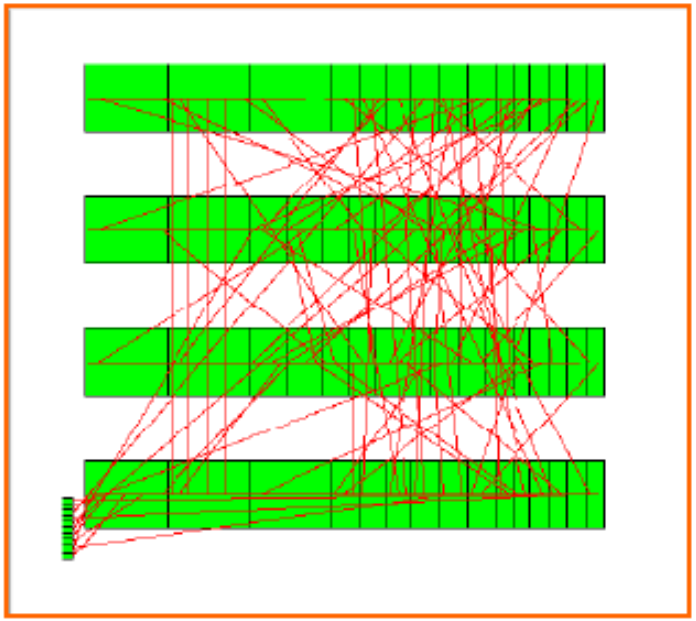


SoC Encounter: General PnR Steps

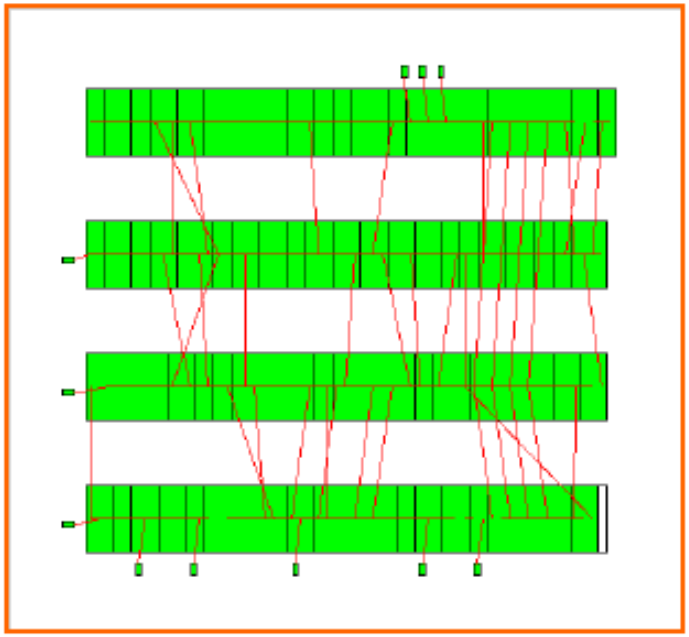
- File preparation & initial setup
- Floor-planning
  - Specify layout size & power rail names
  - Manually place hard macros (e.g. SRAM blocks, if any)
  - Add power ring & power stripes
  - Special route (connect cells' VDD/VSS to power ring)
- **Placement & pre-CTS optimization**
- Clock-tree synthesis (CTS) & post-CTS optimization
  - Generate clock specs. & synthesize clock tree
  - Display clock and optimize design
- Detailed routing & post-route opt. (timing & SI driven)
- Layout verification & stream out

Placement (1)

Placement: highly dependent on IO locations & floorplan, and can affect circuit performance



A bad placement



A good placement

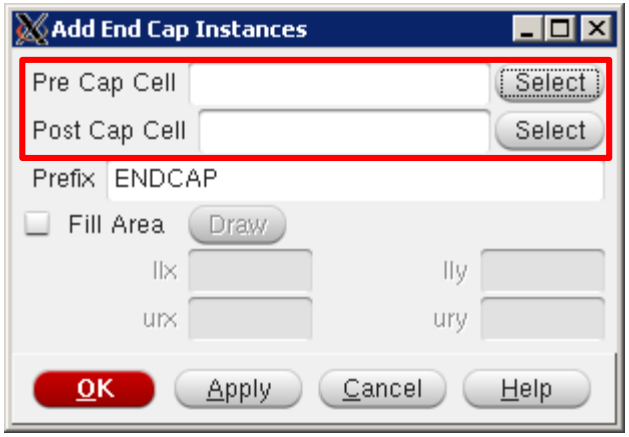
Courtesy: Andrew Kahng, UCSD

Placement (2)

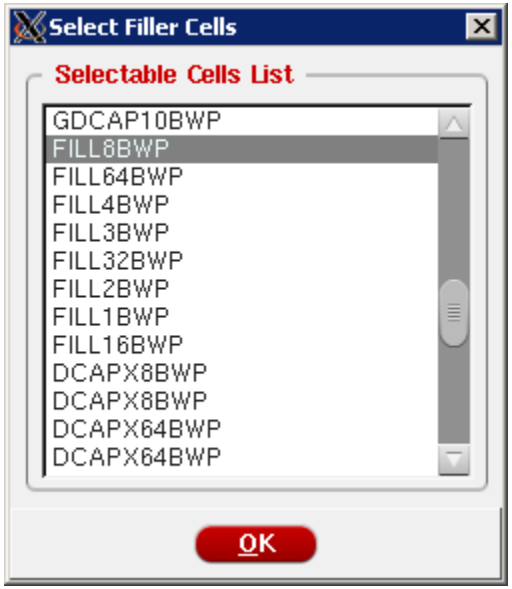
Place → Specify → Placement Blockage

Don't place standard cells with specified metal layers under stripes

Place → Physical Cell → Add End Cap

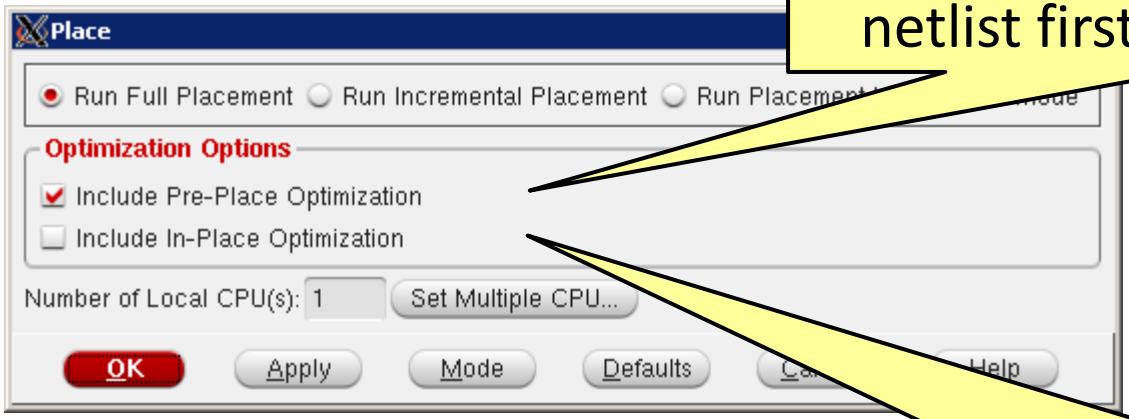


Press **Select**



Placement (3)

Place → Place Standard Cells and Blocks



Pre-place opt. (PPO): tool **will** delete all the buffers in gate-level netlist first, then do placement

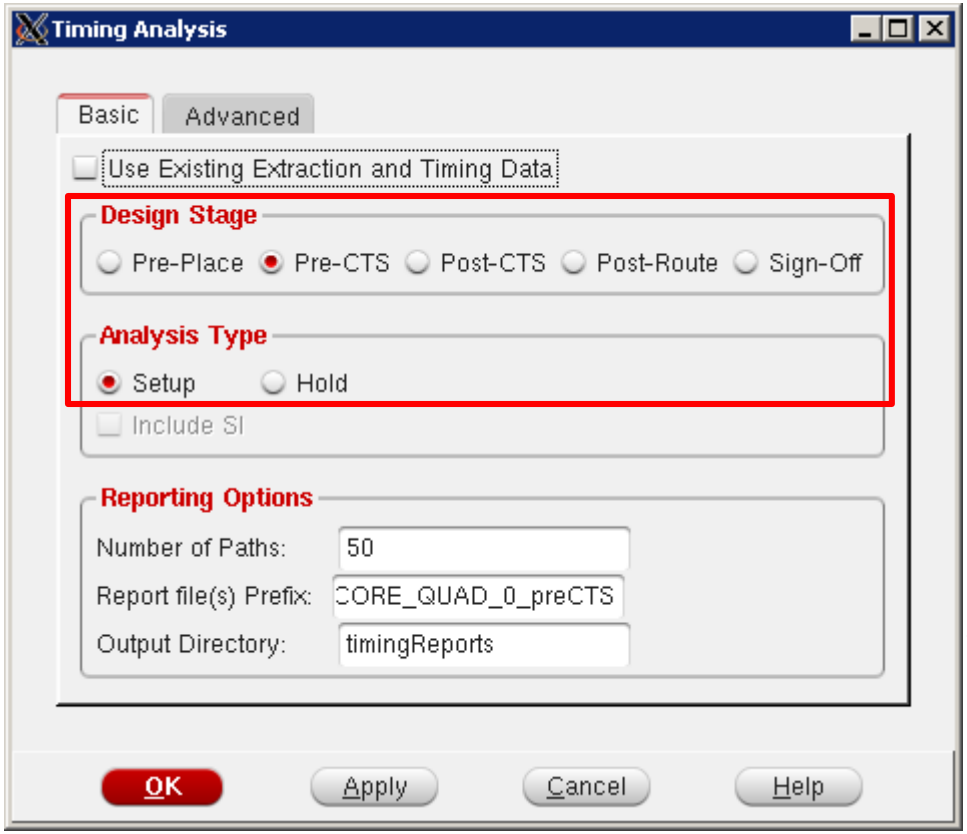
In-place opt. (IPO): tool **won't** delete any buffer in .vg. It does placement & pre-CTS optimization

Note: **IPO** will soon become obsolete, replaced by **pre-CTS optimization**. Therefore we don't suggest to check that option. Only check **PPO** if needed.

Pre-CTS Optimization

Timing → Report Timing

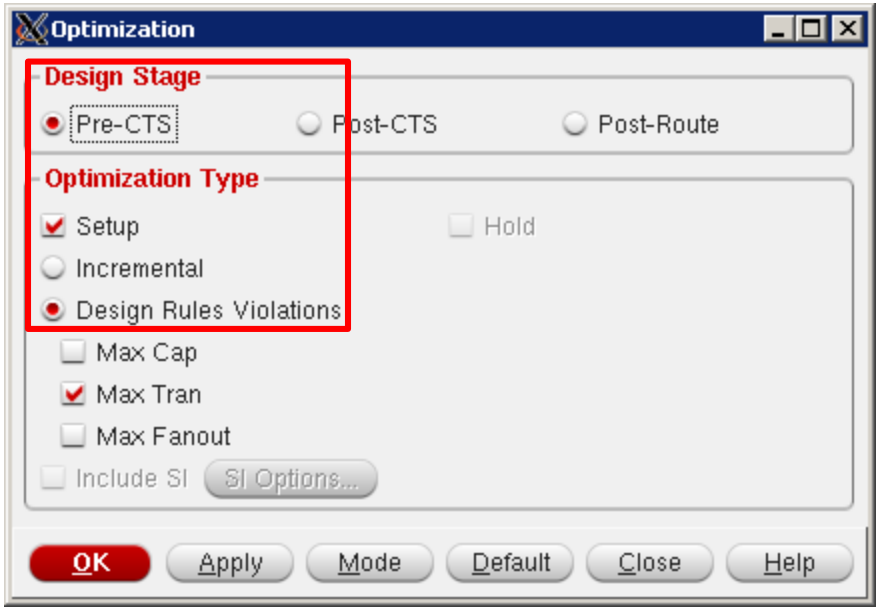
(check **setup** or **hold** to see timing info.)



Tip: **Timing → Debug Timing** allows you to see detailed delay info. of all paths, including **slack statistics**, **node parasitics**, **wire length** from one node to another, etc.

Optimize → Opt. Design

(check **Pre-CTS, setup** and **DRV**)



Tip: We suggest to fix all **DRV** during **pre-CTS opt.** stage, and check only **Max Cap** and **Max Tran** in **post-CTS** and **post-Route**. Also we can change to **Incremental** option if the timing violation is almost fixed.

Can try opt. for **multiple times** to pass the timing

SoC Encounter: General PnR Steps

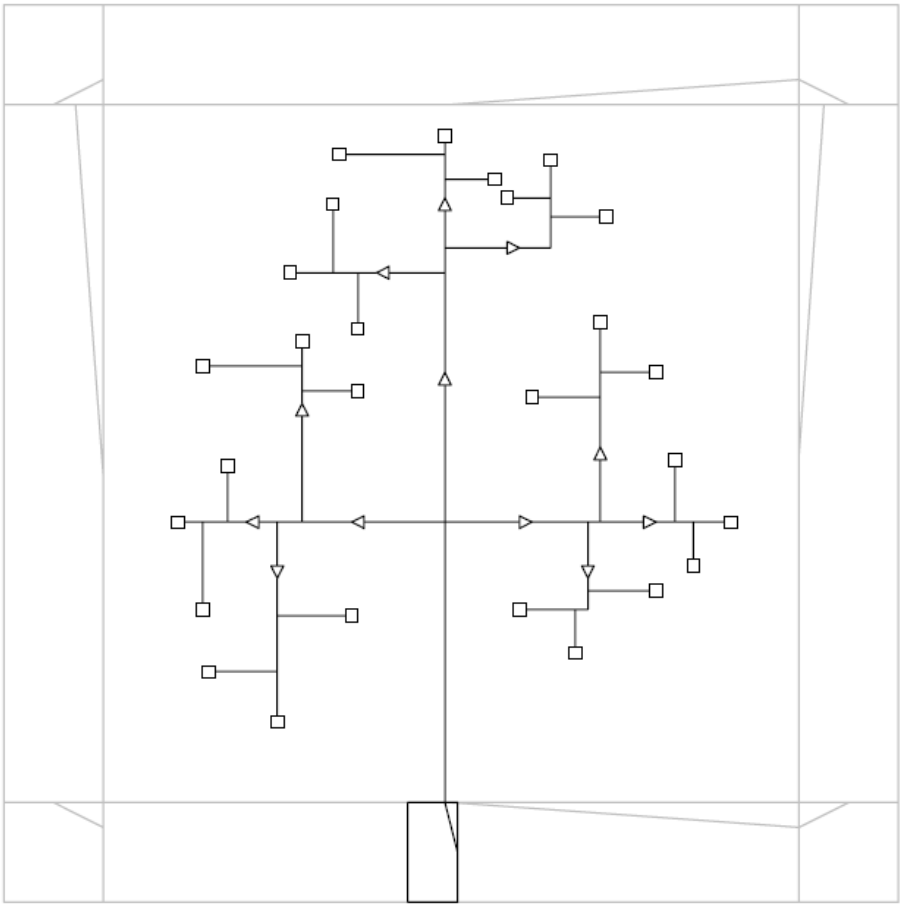
- File preparation & initial setup
- Floor-planning
  - Specify layout size & power rail names
  - Manually place hard macros (e.g. SRAM blocks, if any)
  - Add power ring & power stripes
  - Special route (connect cells' VDD/VSS to power ring)
- Placement & pre-CTS optimization
- **Clock-tree synthesis (CTS) & post-CTS optimization**
  - Generate clock specs. & synthesize clock tree
  - Display clock and optimize design
- Detailed routing & post-route opt. (timing & SI driven)
- Layout verification & stream out

Clock Tree Synthesis (CTS) (1)

Clock tree propagate clock signals, through clock buffers, to each of the registers

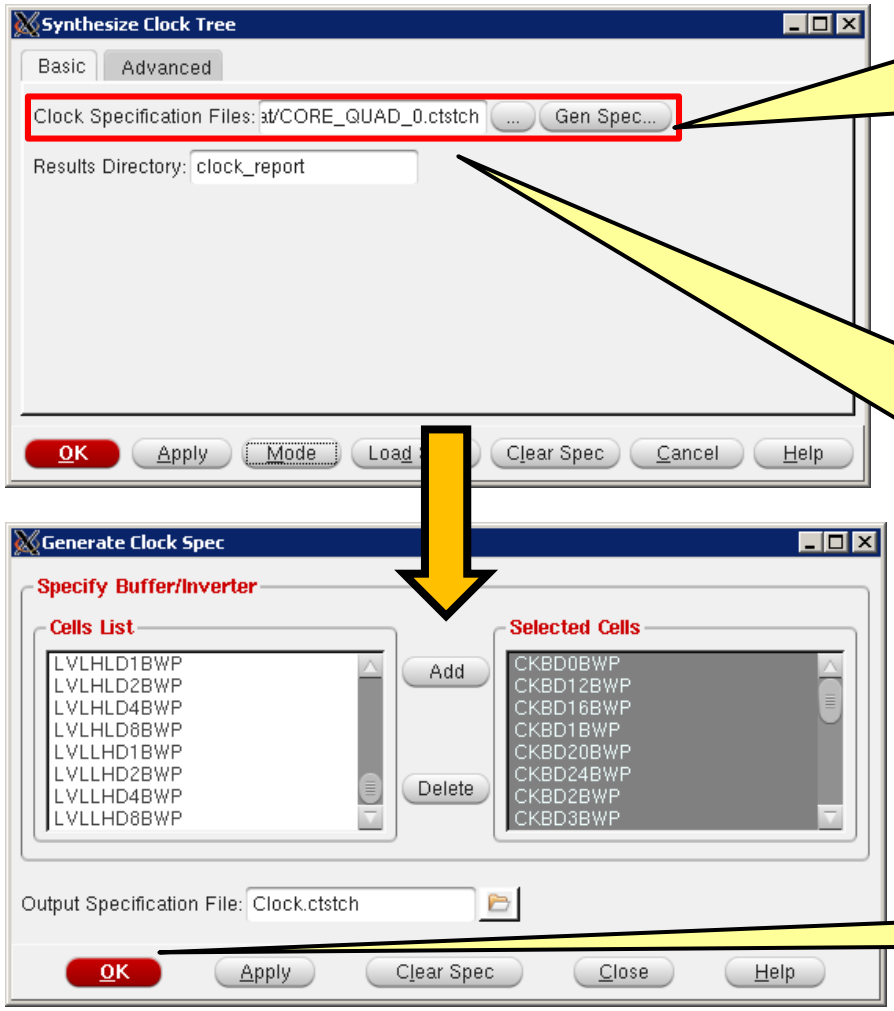
Clock problem

- Heavy net loading
- Long insertion delay
- Skew
- Coupling
- Power consumption
- Electromigration



Clock Tree Synthesis (CTS) (2)

Clock → Synthesize Clock Tree



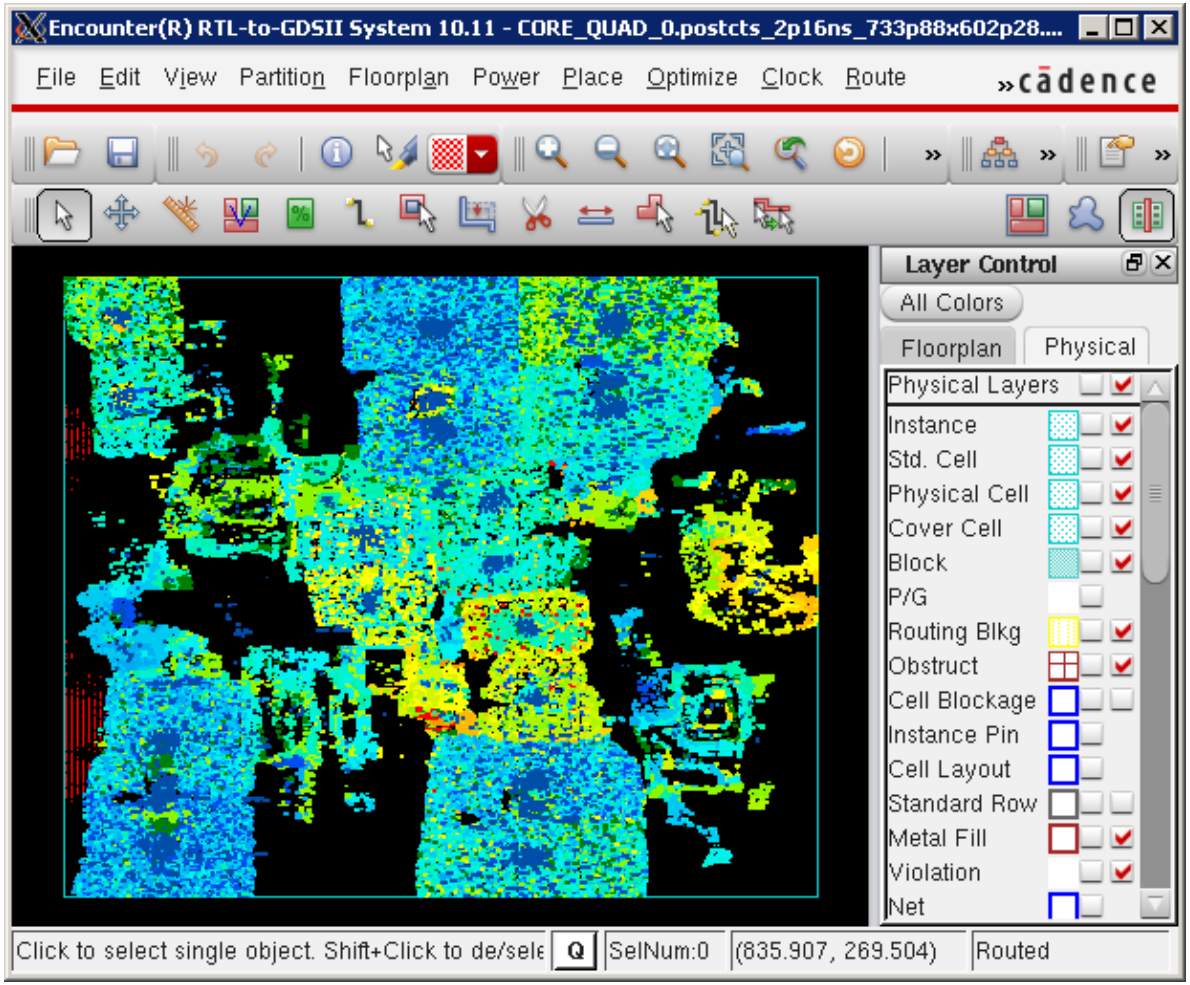
Press **Gen Spec** for the first time. Select all **clock buffers** and **clock inverters** for CTS.

If already have **Clock.ctstch**, just put that file in the **Clock Specification Files** and press **OK** to synthesize clock tree.

Press **OK** to get **Clock.ctstch**

Display Clock Tree to See Clock Distribution

Clock → Display → Display Clock Tree

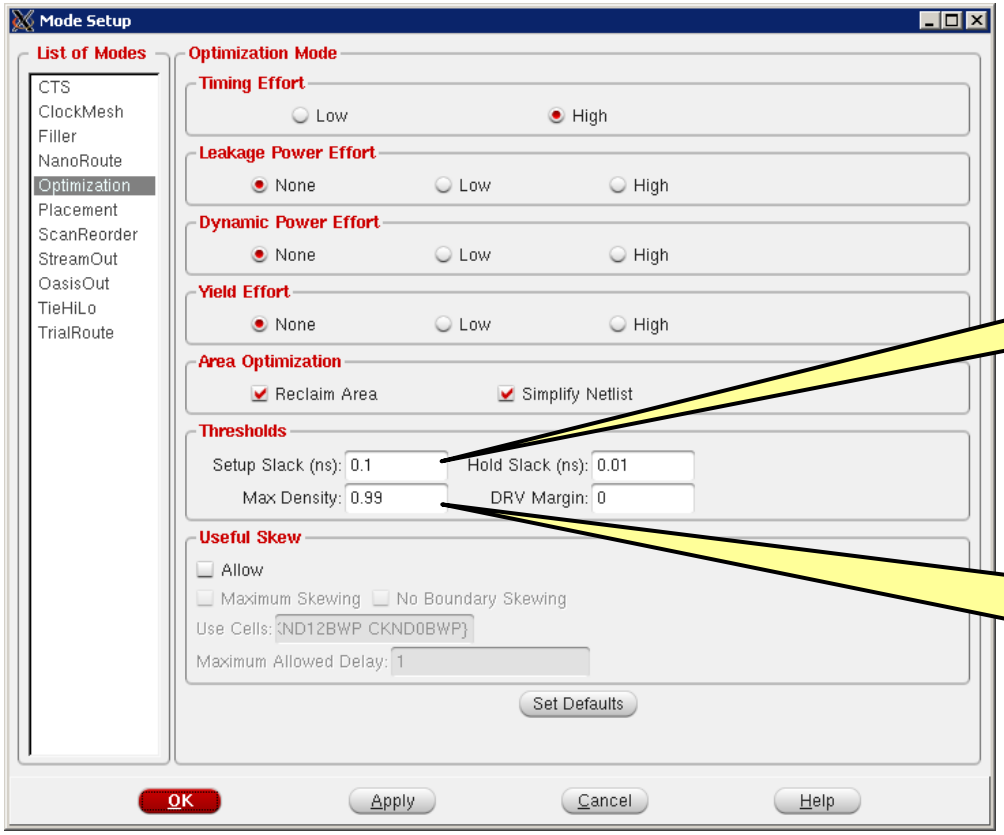


Post-CTS Optimization

Optimize → Optimize Design

(check **Post-CTS, setup** and **DRV** to fix **setup time**. Change to **hold** to fix **hold time**)

press Mode to enable advanced opt. features



Set **setup slack threshold** to be **>0** helps hold-time fixing

Orig. **max. utilization=0.95**, can change to higher value

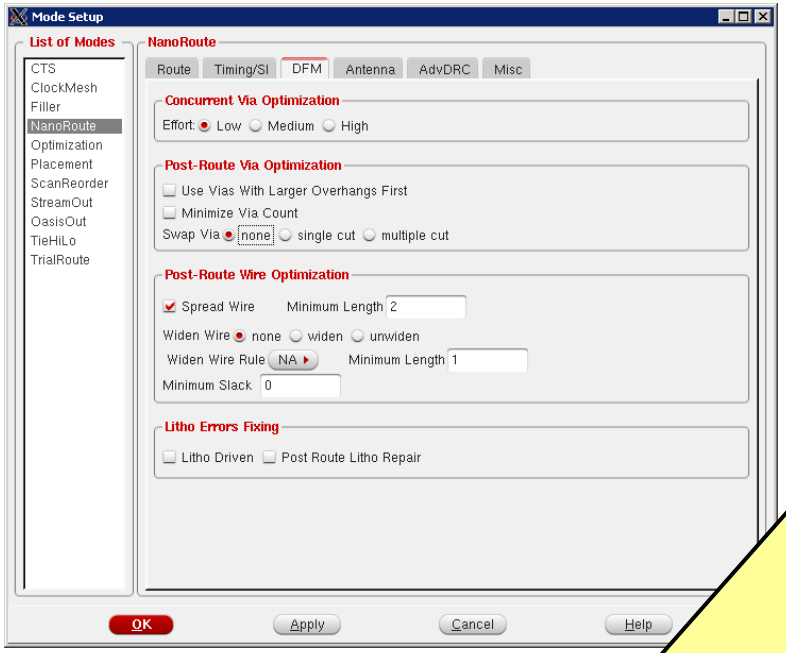
SoC Encounter: General PnR Steps

- File preparation & initial setup
- Floor-planning
  - Specify layout size & power rail names
  - Manually place hard macros (e.g. SRAM blocks, if any)
  - Add power ring & power stripes
  - Special route (connect cells' VDD/VSS to power ring)
- Placement & pre-CTS optimization
- Clock-tree synthesis (CTS) & post-CTS optimization
  - Generate clock specs. & synthesize clock tree
  - Display clock and optimize design
- **Detailed routing & post-route opt. (timing & SI driven)**
- Layout verification & stream out

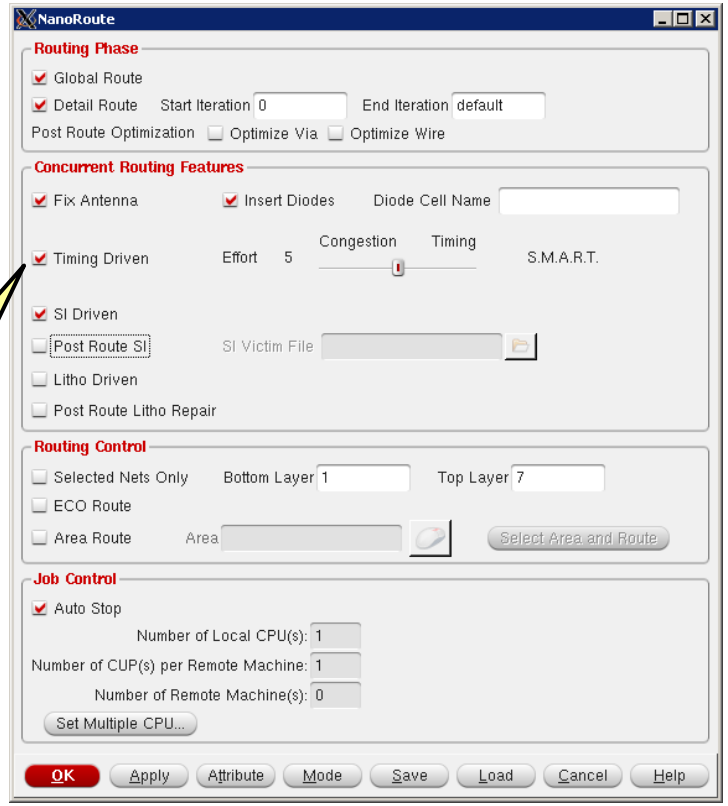
Detailed Route

Route → Nano Route → Route

Step 1: Press **Mode**, go to **DFM**, check **Spread Wire**



Step 2: Check following options
(can set **multi-CPU** to speed up)

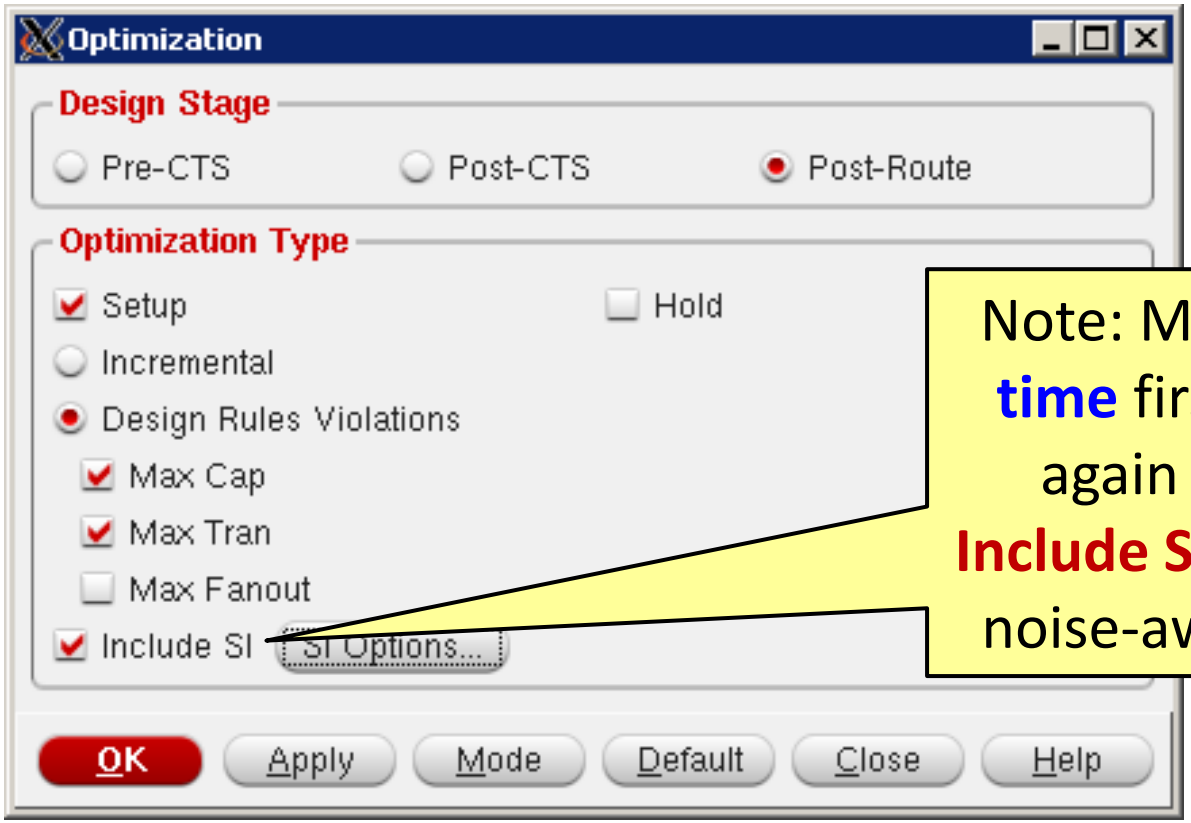


For optimized timing, set **Effort=10**; if design highly congested, set **Effort=1**

Post-Route Optimization

Optimize → Optimize Design

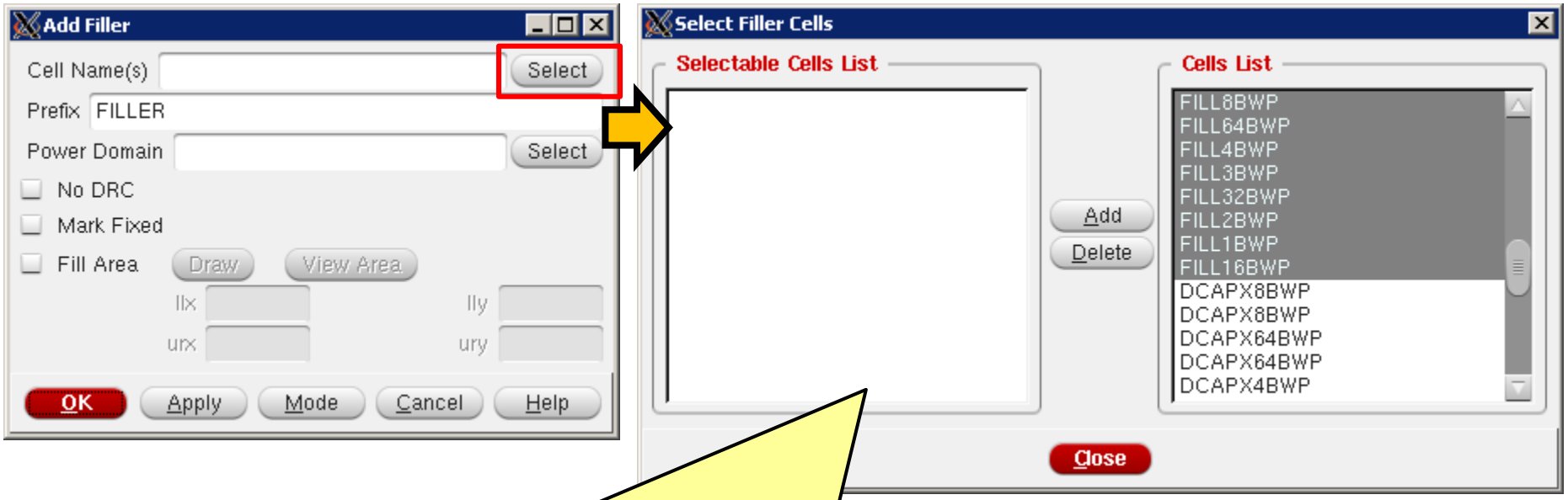
(check **Post-Route, setup** and **DRV** to fix **setup time**.
Change to **hold** to fix **hold time**)



Note: Must fix **setup/hold time** first, then fix timing again by checking the **Include SI** option for further noise-aware optimization.

Adding Fillers to Fill Unplaced Core Area

Place → Physical Cell → Add Filler



Do **Select → Add → Close** to list the fillers you want to use. **Remember**: add from **wider** filler to **narrower** filler.

tip: use left click + drag to select multiple fillers at once.

SoC Encounter: General PnR Steps

- File preparation & initial setup
- Floor-planning
  - Specify layout size & power rail names
  - Manually place hard macros (e.g. SRAM blocks, if any)
  - Add power ring & power stripes
  - Special route (connect cells' VDD/VSS to power ring)
- Placement & pre-CTS optimization
- Clock-tree synthesis (CTS) & post-CTS optimization
  - Generate clock specs. & synthesize clock tree
  - Display clock and optimize design
- Detailed routing & post-route opt. (timing & SI driven)
- **Layout verification & stream out**

Verify Layout

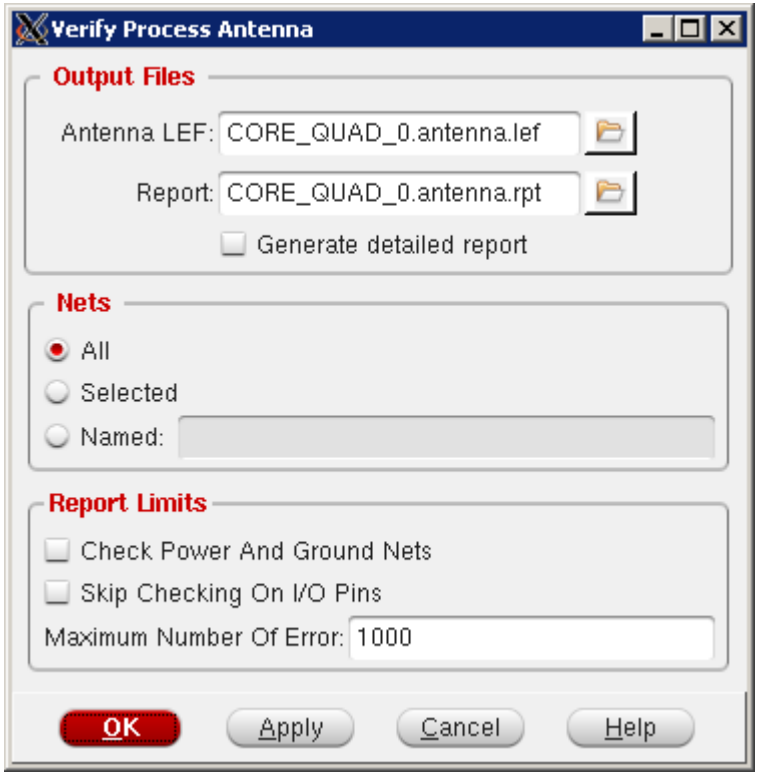
- DRC: **Verify → Verify Geometry**
  (should get zero error, otherwise gotta fix them)

```
Begin Summary ...
  Cells       : 0
  SameNet     : 0
  Wiring      : 0
  Antenna     : 0
  Short       : 0
  Overlap     : 0
End Summary

  Verification Complete : 0 Viols.  0 Wrngs.
```

- LVS: **Verify → Verify Connectivity**

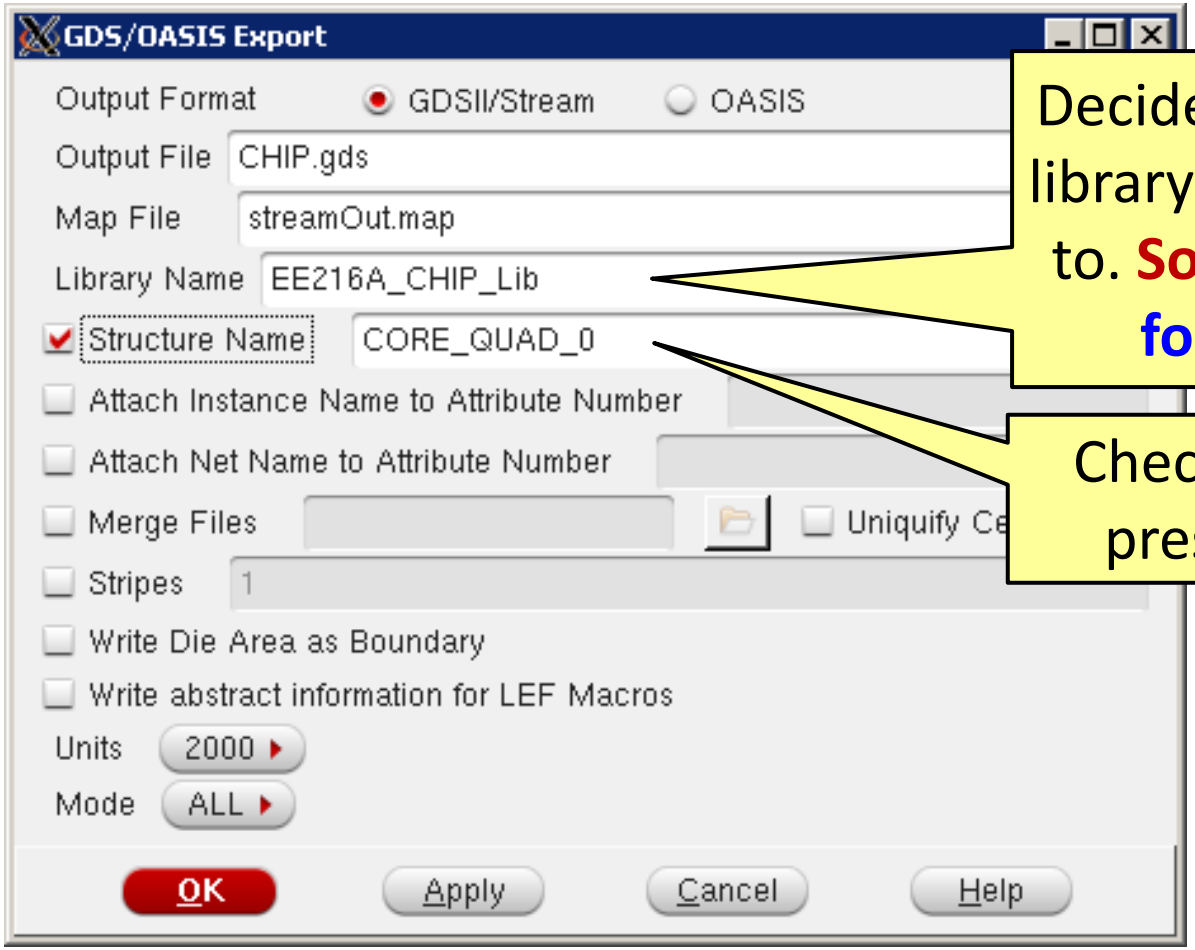
```
Begin Summary
  Found no problems or warnings.
End Summary
```

- Antenna violation: **Verify → Verify Process Antenna**



Stream Out GDS File

Design → Save → GDS/OASIS

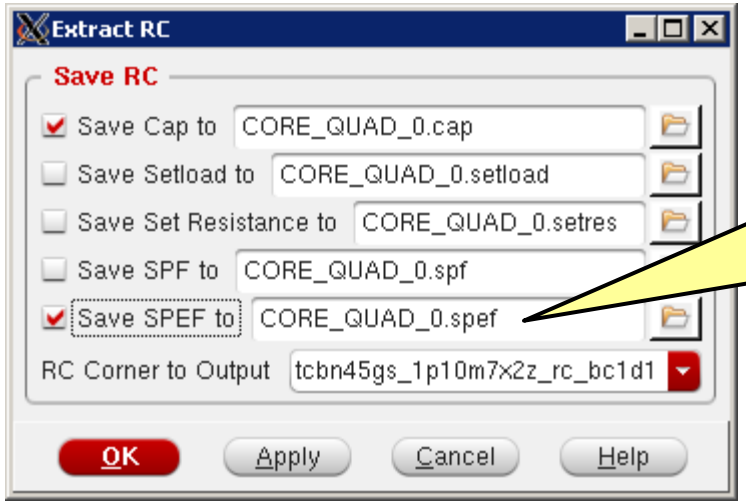


Decide the name of **.gds** & library it's gonna be belong to. **Source the .map** from **foundry design kit**

Check **Structure Name**, press **OK** to generate

Post-Layout Parasitic/LEF/Netlist/SDF/LIB

Parasitic: **Timing → Extract RC**



Check **Save SPEF to** and name the **.spef** file. Don't forget to generate **.spef** for both best- and worse-case RC corners. We use **.spef** in post-layout simulation

LEF, Netlist, SDF: **Using commands**

saveNetlist XXX_sim.vg (for post-sim.)
saveNetlist -flattenBus -includePowerGround XXX_lvs.vg (for Calibre LVS check)
lefOut XXX.lef –stripePin
write_sdf XXX.sdf

LIB: **Using commands**

setAnalysisMode -checkType setup
do_extract_model XXX_setup.lib
setAnalysisMode -checkType hold
do_extract_mode XXX_hold.lib

Post-Layout Simulation & Bottom-Up Design

- **Post-layout simulation**
  - Prepare **.vg**, and **.sdf** from SoC Encounter
  - Do the same thing as in pre-layout gate-level simulation

- **Bottom-up hierarchical design flow**
  - Once having **.lef** and **.lib**, your design can be treated as a **"big" standard cell**. We call it as a **macro block**.
  - At higher-level integration, just source that **.lef** and **.lib** and do the same thing as in bottom-level PnR.

Summary

- **Physical synthesis tool processes the design from gate to transistor level**
- **Slightly modify .vg and .sdc if necessary**
- **Take some time to carefully think about your floorplan & I/O locations to get optimized circuit performance**
  - Can affect wiring congestion
  - Suggested core dimension: as square as possible
- **Optimize timing in each design stage**
  - pre-cts: setup
  - post-cts: setup + hold
  - post-route: setup + hold + SI
- **Make sure to pass build-in DRC & LVS before finish**