# Agenda

DAY 1

| | |
|---|---|
| i | Introduction & Overview |
| 1 | Data Setup & Basic Flow |
| 2 | Design Planning |

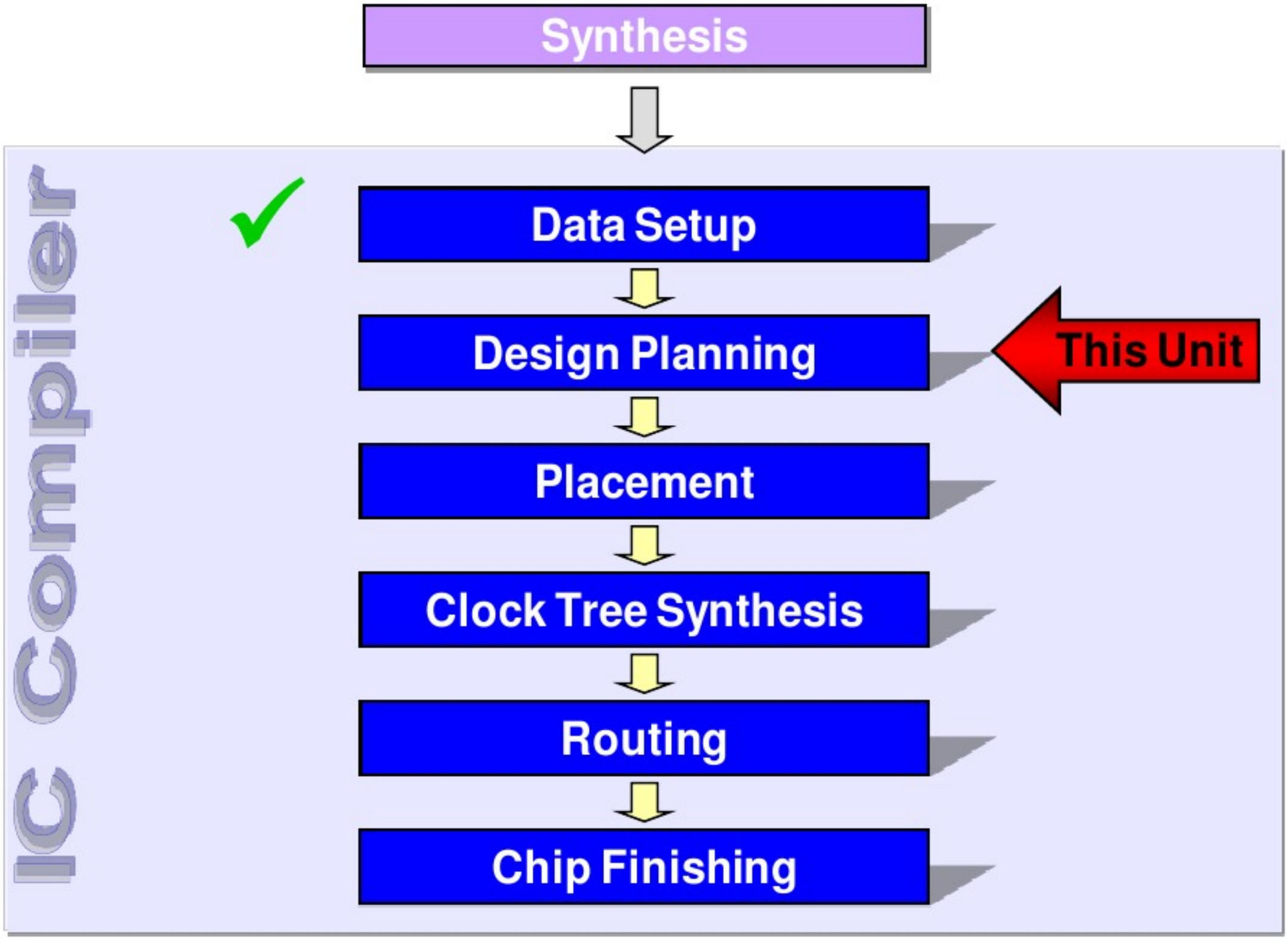

©2010 Synopsys, Inc. All Rights Reserved

# Unit Objectives

After completing this unit, you should be able to:

- Use IC Compiler to create a non-hierarchical chip-level floorplan
- Create a floorplan that is likely to be routable and achieve timing closure

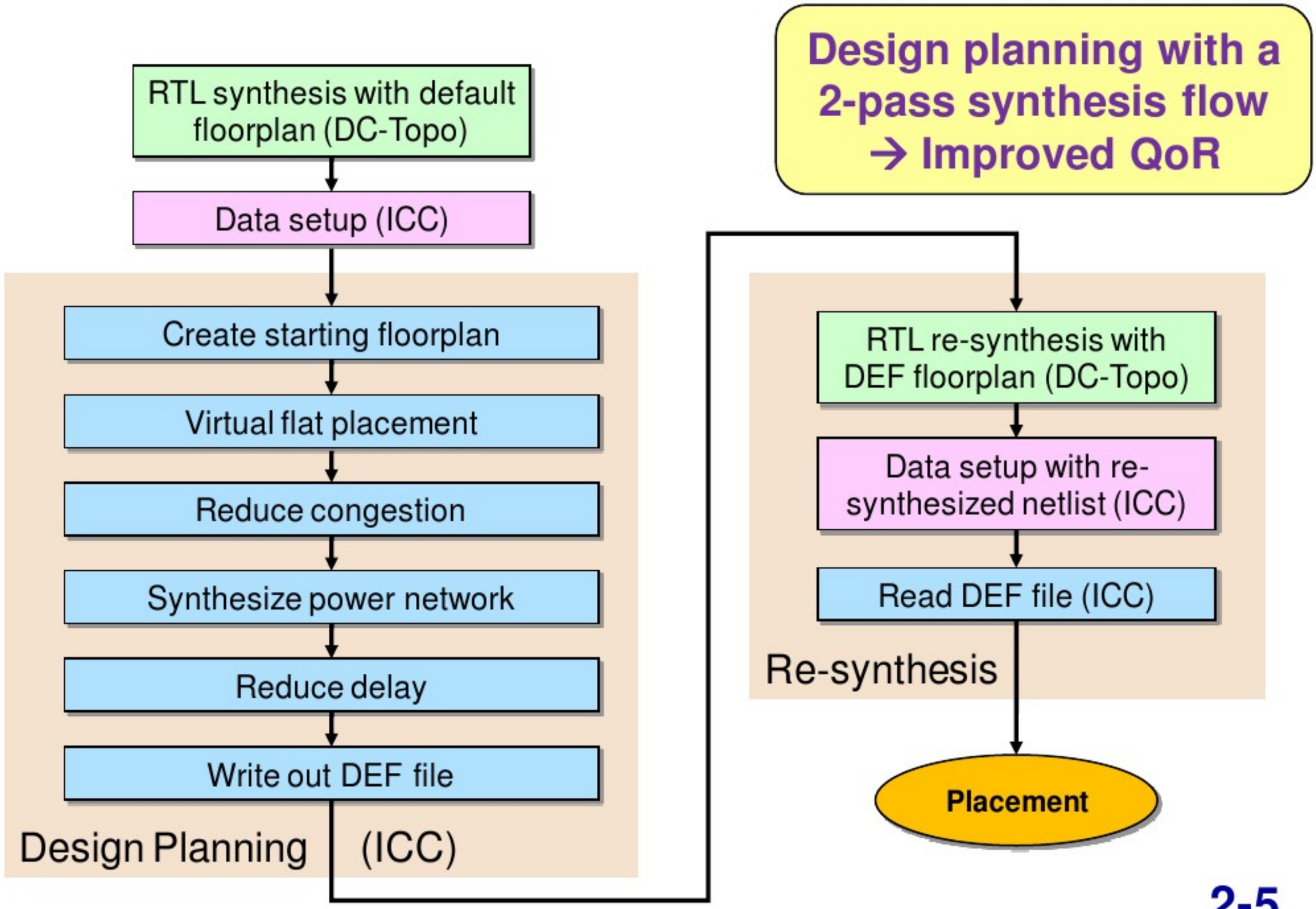
# General IC Compiler Flow



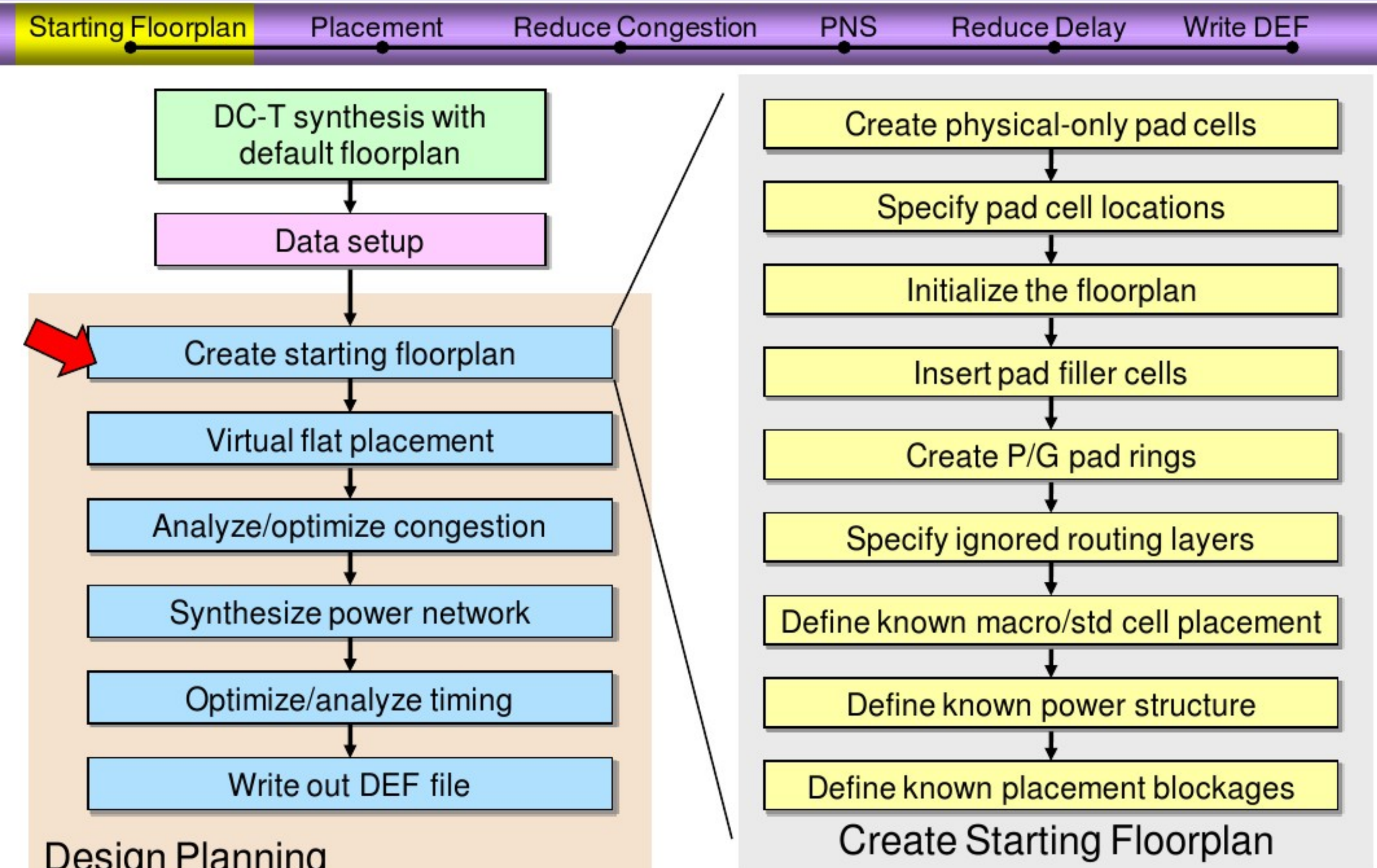
# Terminology

- ***Design planning*** **is the iterative process of creating a floorplan**
- **A chip-level *floorplan* entails defining:**
  - Core size, shape and placement rows
  - Periphery: IO, power, corner and filler pad cell locations
  - Macro cell placement
  - Standard cell placement constraints (blockages)
  - Power grid (rings, straps, rails)
- **A *physical design*, or *layout*, is the result of a synthesized netlist that has been *placed* and *routed***

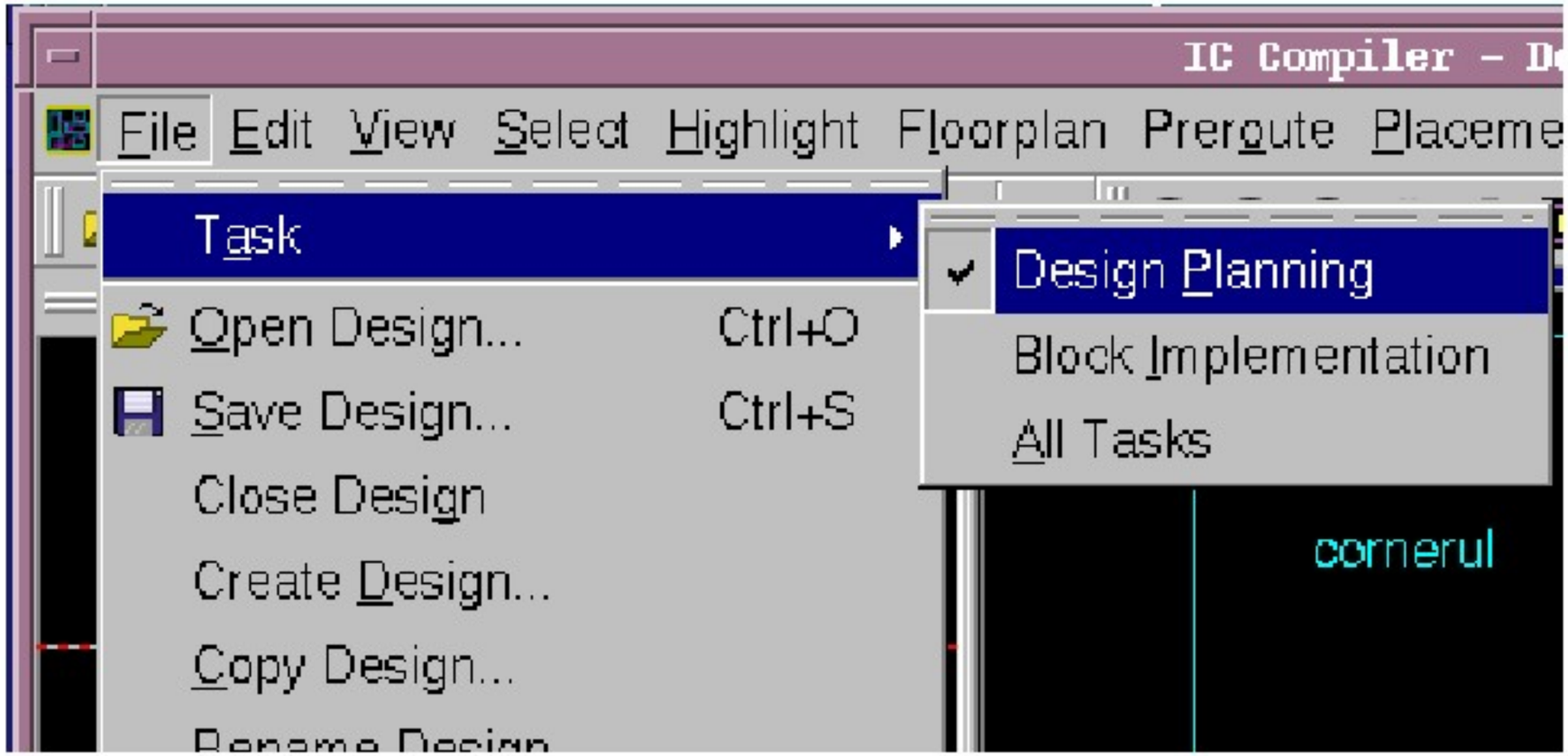
# ICC Design Planning and Re-Synthesis Flow



# Create the Starting Floorplan



# Select the *Design Planning Task* GUI




**This makes additional menus and forms available, which are otherwise hidden**

```
open_mw_cel DESIGN_data_setup
set_tlu_plus_files \
      -max_tluplus ./libs/abc_max.tlup \
      -min_tluplus ./libs/abc_min.tlup \
      -tech2itf_map  ./libs/abc.map
source tim_opt_ctrl.tcl
gui_set_current_task -name {Design Planning}
```

Re-apply TLUplus

Re-apply timing and optimization controls

# Create Physical-only Pad Cells



- Physical-only pad cells (VDD/GND, corner cells) are not part of the synthesized netlist
- Must be created prior to specifying the pad cell locations

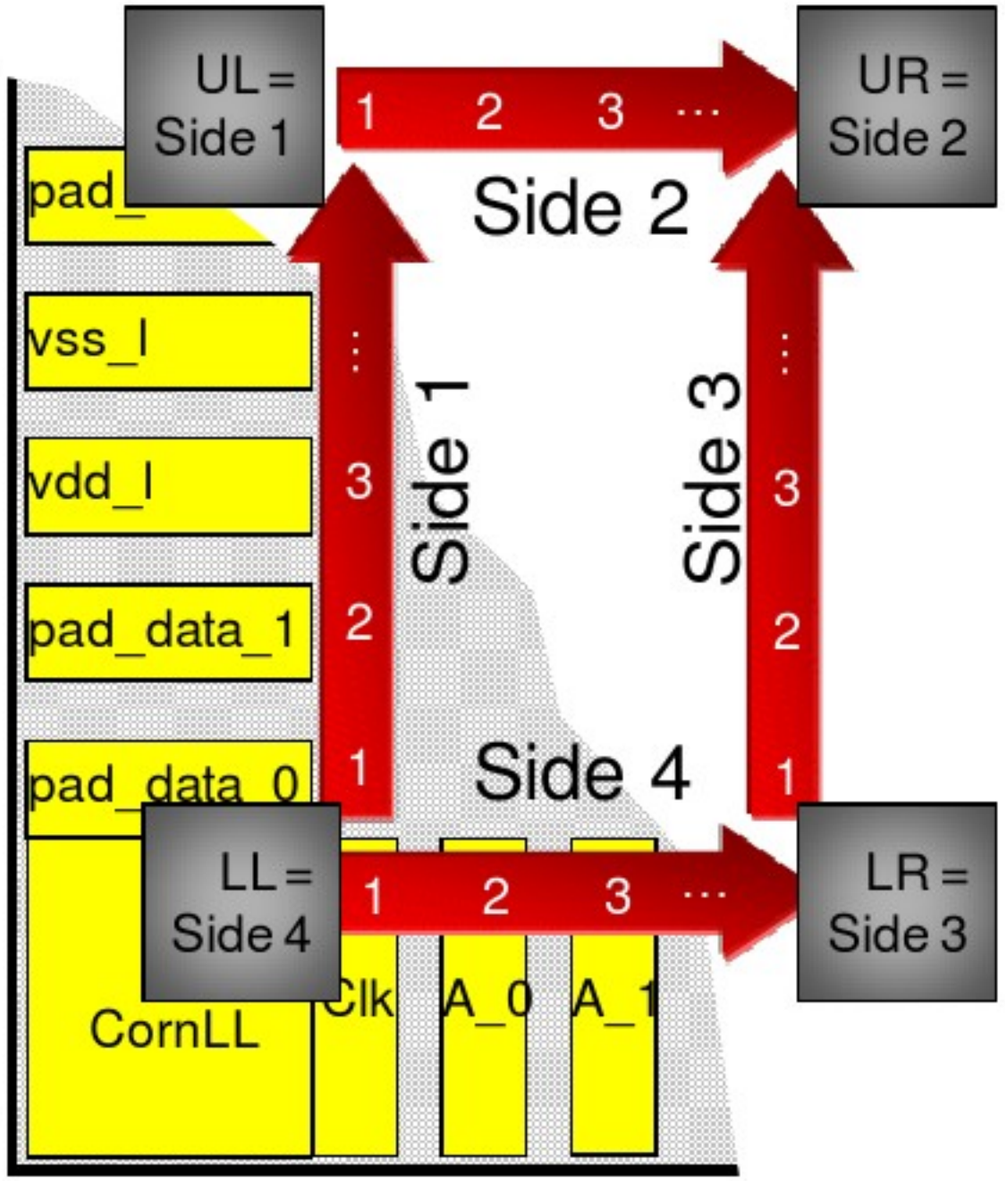
```
create_cell {vss_l vss_r vss_t vss_b} pv0i
create_cell {vdd_l vdd_r vdd_t vdd_b} pvdi
create_cell {CornLL CornLR CornTR CornTL} pfrelr
```

# Specify Pad Cell Locations

Starting Floorplan

```
# Place the corner cells
set_pad_physical_constraints -pad_name "CornUL" -side 1
set_pad_physical_constraints -pad_name "CornUR" -side 2
set_pad_physical_constraints -pad_name "CornLR" -side 3
set_pad_physical_constraints -pad_name "CornLL" -side 4

# Place io and power pads
# Left/Right sides start from bottom (excluding corner)
set_pad_physical_constraints -pad_name "pad_data_0" \
                             -side 1 -order 1
set_pad_physical_constraints -pad_name "pad_data_1" \
                             -side 1 -order 2
set_pad_physical_constraints -pad_name "vdd_l" \
                             -side 1 -order 3
....
# Bottom/Top sides start from left (excluding corner)
set_pad_physical_constraints -pad_name "Clk" \
                             -side 4 -order 1
set_pad_physical_constraints -pad_name "A_0" \
                             -side 4 -order 2
....
```




```
set_pad_physical_constraints \
-pad_name <name> -side <#> -order <#>
```

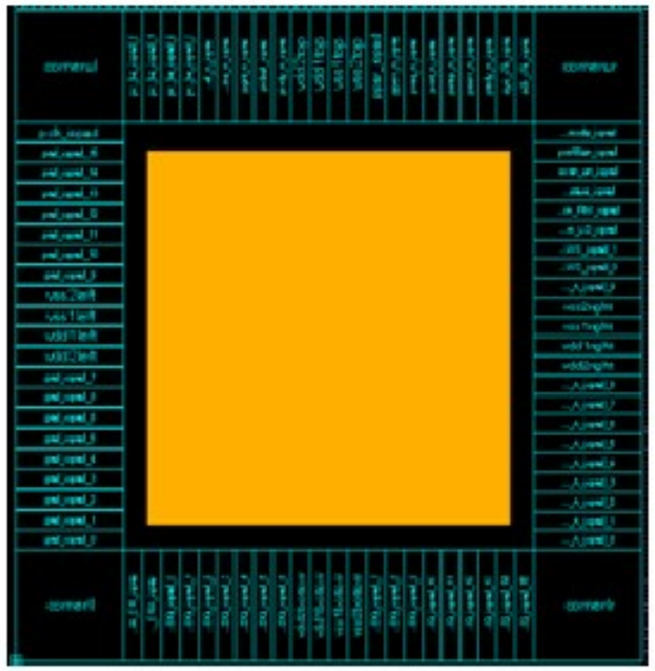
Constraints are honored when the floorplan is created, with even distribution by default

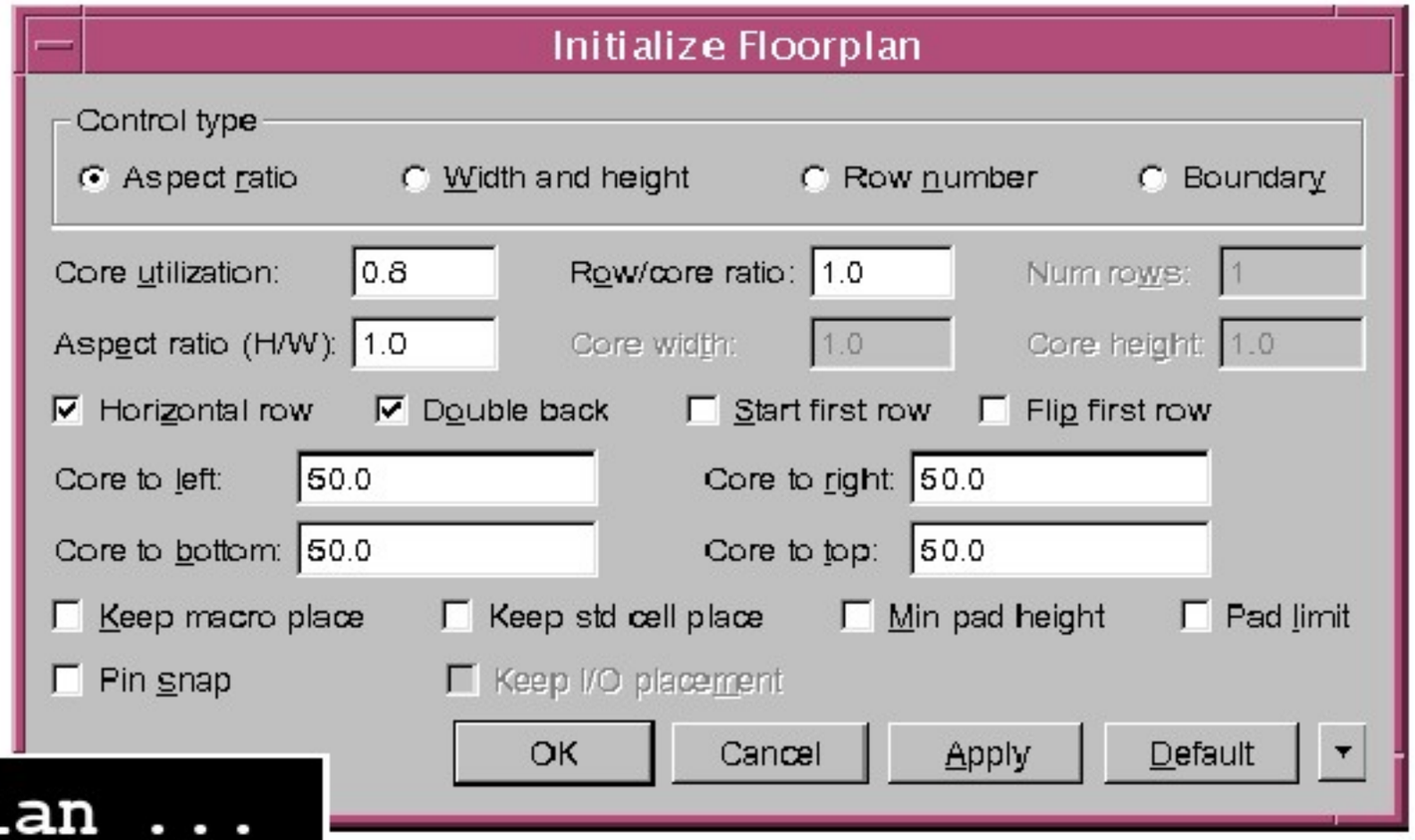
# Initialize the Floorplan



## Creates the core and periphery area

- Defines placement or site rows within the core area
- Defines the chip boundary or periphery area
- Places IO pads
  - Pads defined in netlist and by `create_cell`
  - Ordering defined by `set_pad_physical_constraints`





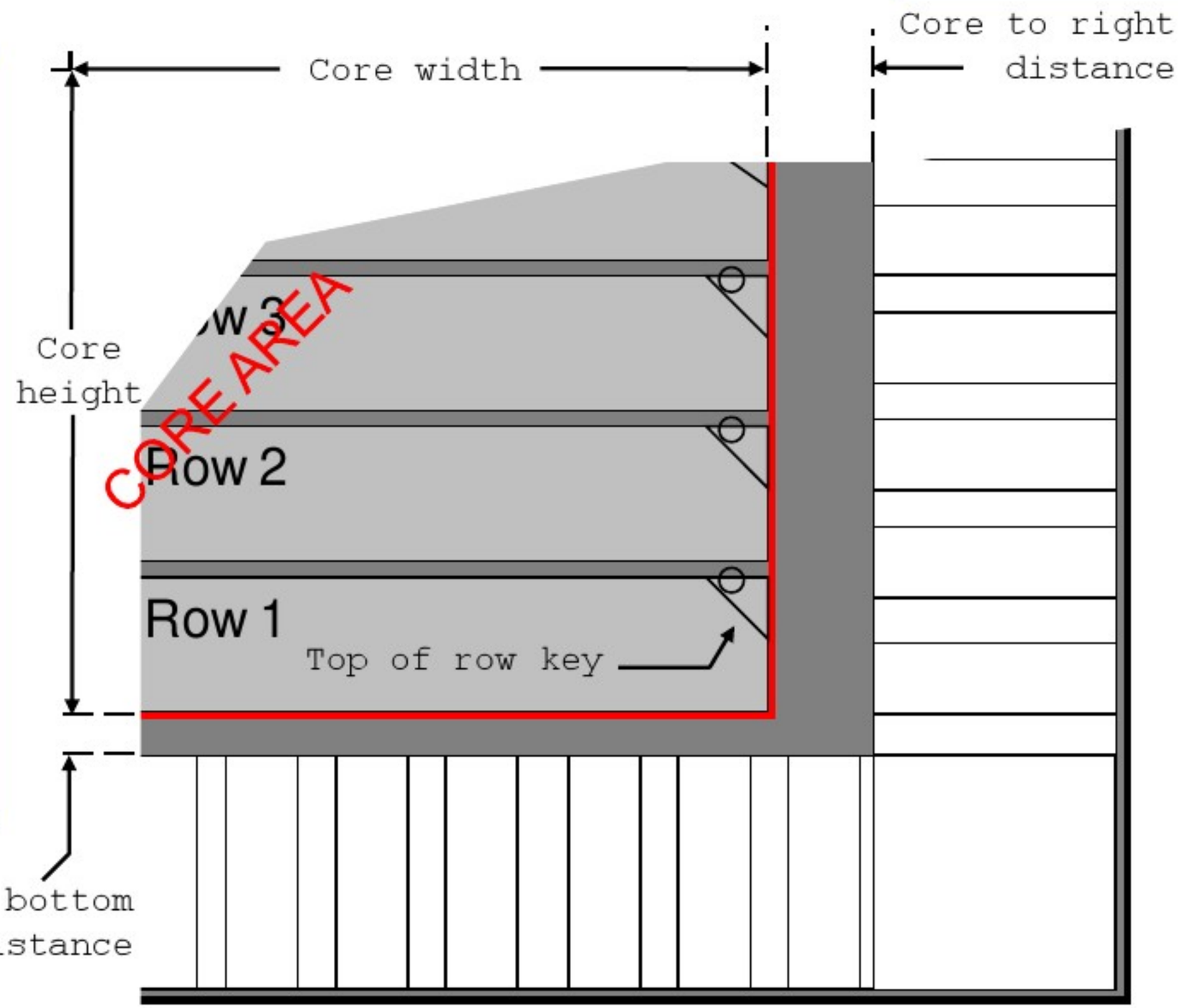

```
initialize_floorplan ...
```

# Core Area Parameters

Starting Floorplan

**Control Type**

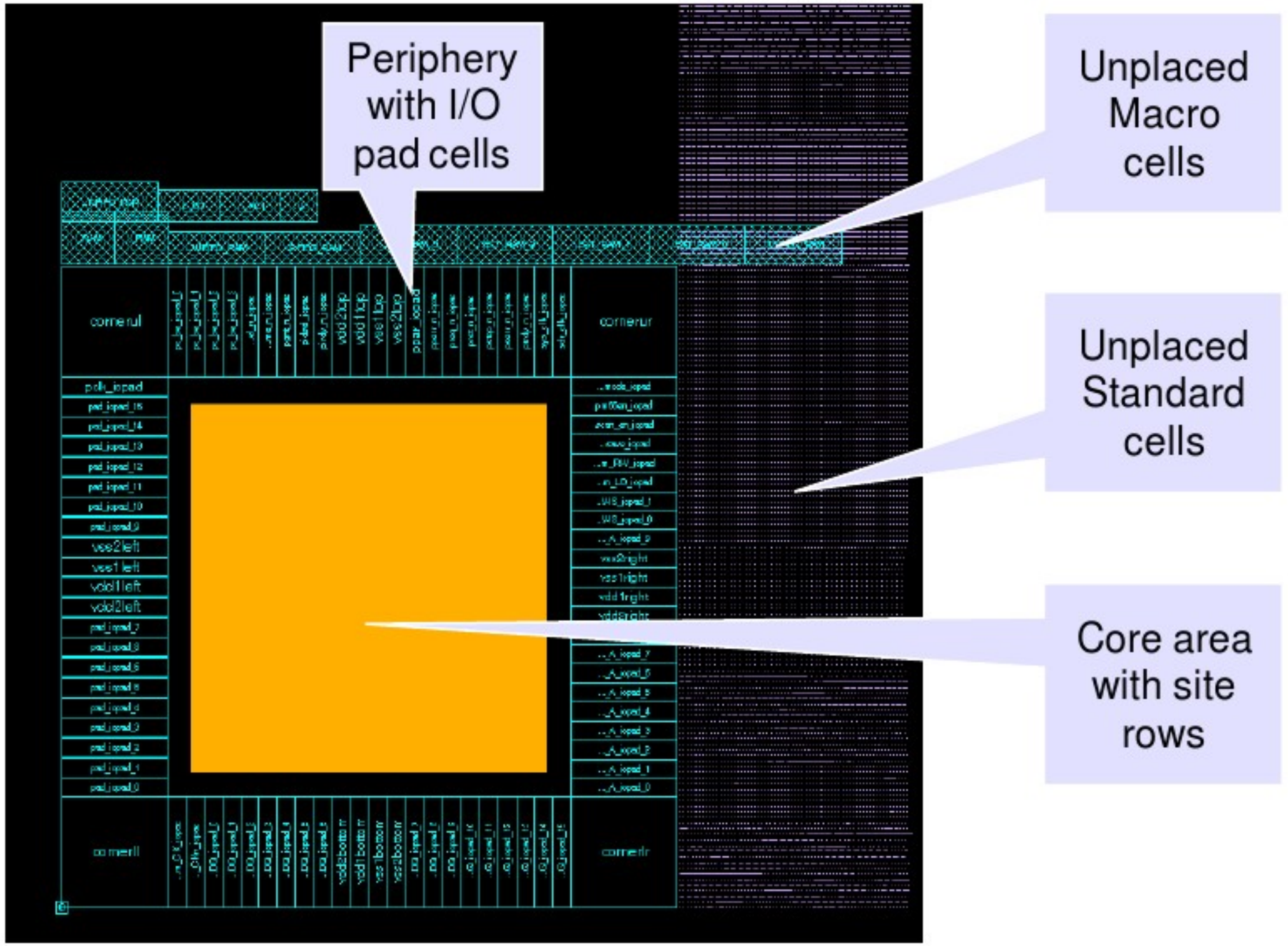
* Aspect ratio
  - →Core utilization
  - →Aspect ratio (H/W)
  - →Row/core ratio
* Width and height
  - →Core width
  - →Core height
  - →Row/core ratio
* .....
* .....




Example of a horizontal, no double back, no-flip first row and Row/core <1.0

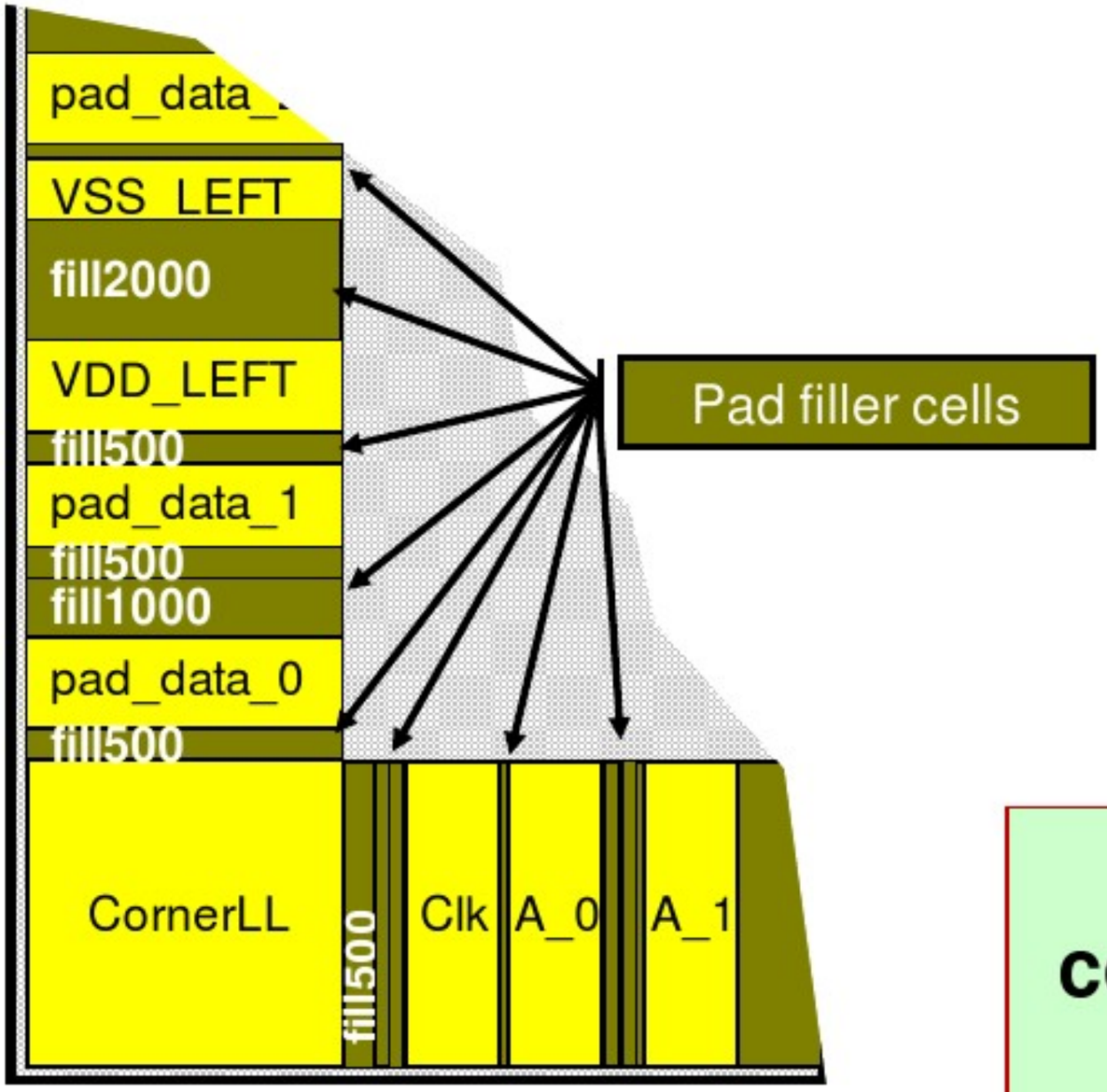
# Floorplan After Initialization

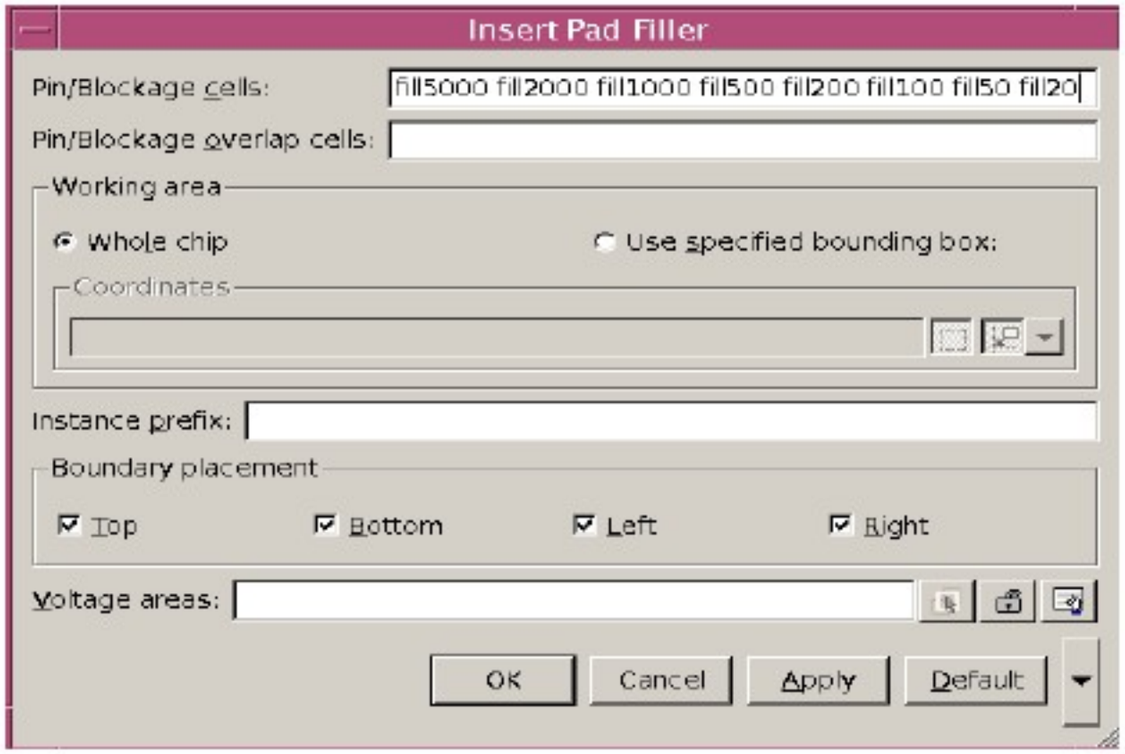




# Insert Pad Filler Cells

Starting Floorplan




**May be needed for continuity of N-well, P-well, and/or P/G routing**

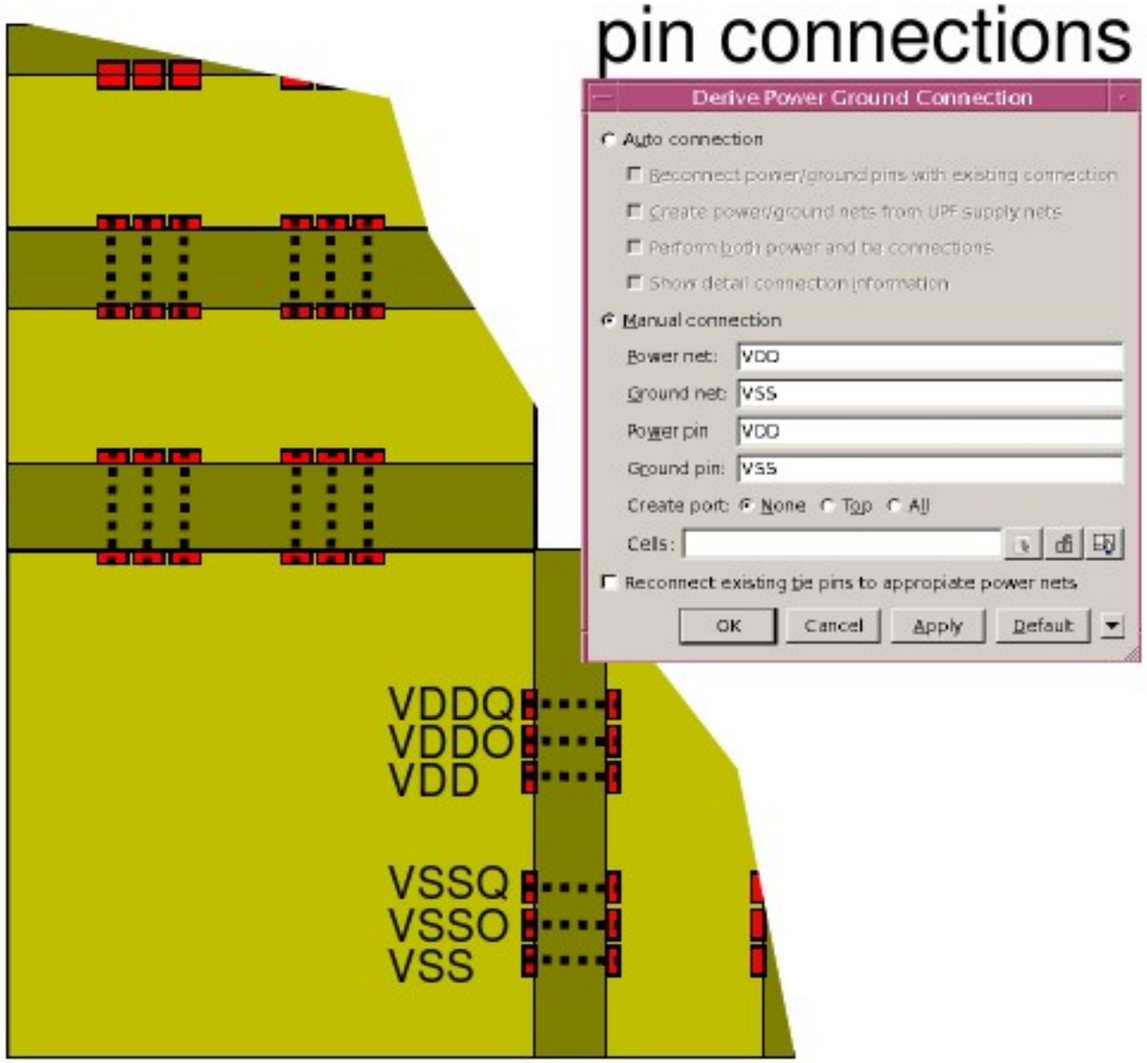
```
insert_pad_filler -cell "fill5000 fill2000 fill1000  ... "
```

**Order of cells is important – list largest to smallest from left to right to use minimum number of cells!**

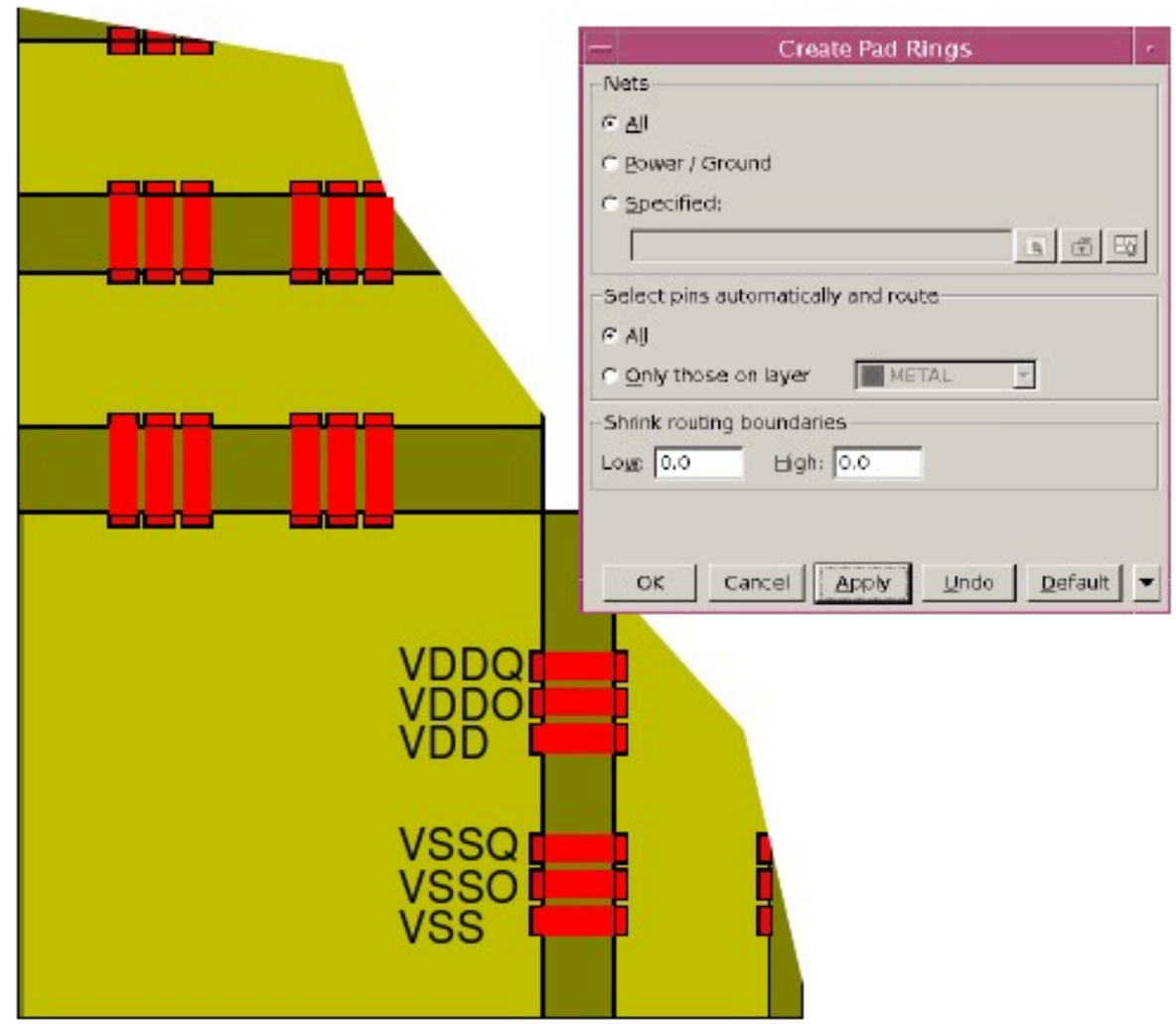
# Create P/G Pad Rings

Starting Floorplan

Make logical P/G pin connections




Route P/G rings




```
derive_pg_connection -power_net VDD  -power_pin VDD  -ground_net VSS  -ground_pin VSS
derive_pg_connection -power_net VDDO -power_pin VDDO -ground_net VSSO -ground_pin VSSO
derive_pg_connection -power_net VDDQ -power_pin VDDQ -ground_net VSSQ -ground_pin VSSQ
derive_pg_connection -power_net PWR  -ground_net GND -tie

create_pad_rings
```

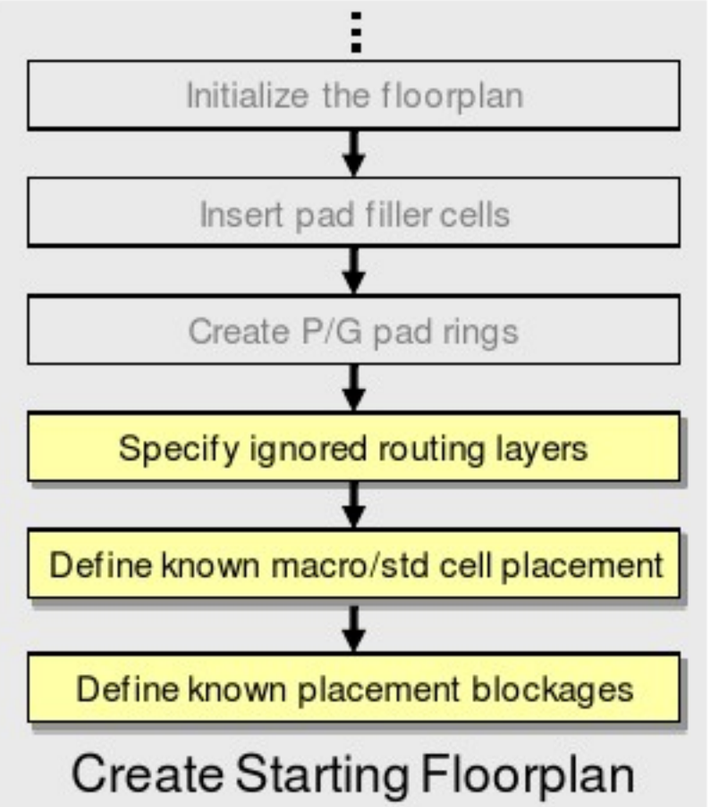
**May be needed for P/G continuity**

# Prior to Virtual Flat Placement



- **By default *virtual flat placement* (`create_fp_placement`):**
  - Places standard cells and non-fixed macros
  - Can place cells anywhere in the core
  - Assumes all layers defined in the technology file are used

- **Prior to placement specify any non-default constraints**
  - Routing layers that should not be used
  - Pre-defined macro/standard cell locations
  - Placement blockages



# Ignore Extra Routing Layers



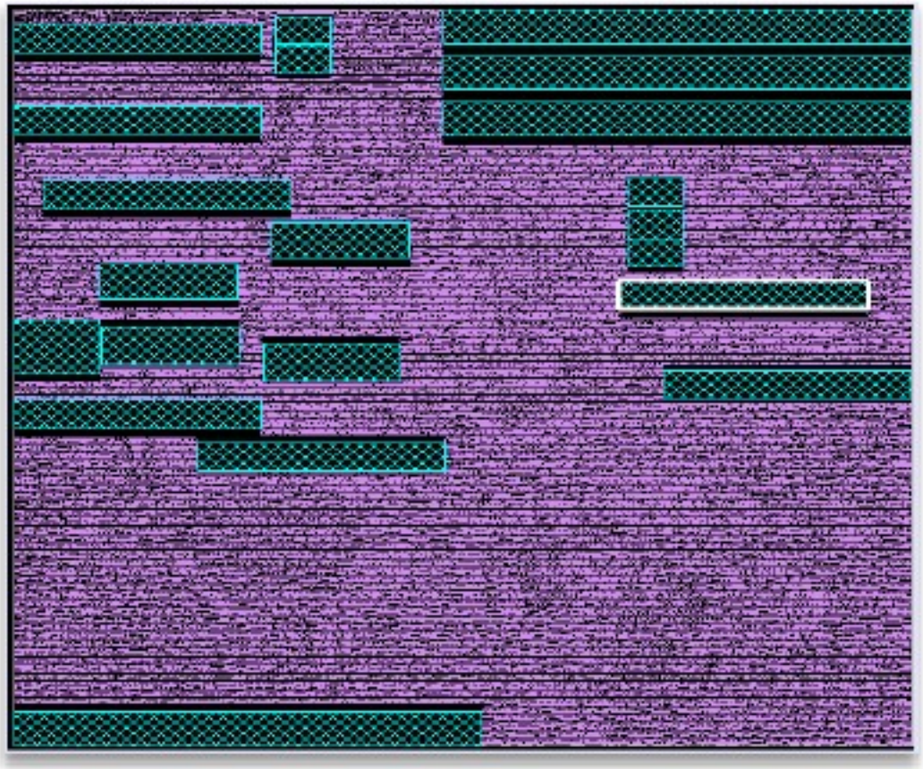
- **By default IC Compiler will use <u>all</u> metal layers defined in your technology**
- **If planning to use fewer metal layers can result in:**
  - Optimistic congestion analysis pre-route
  - Inaccurate delay calculations due to inaccurate parasitic RC calculations
- **Tell IC Compiler to ignore these unused layers**
  - Accurate congestion and timing analysis prior to routing
  - No additional “route guides” needed

```
set_ignored_layers -max_routing_layer M7
```

```
report_ignored_layers
remove_ignored_layers
```

# Constraining Macros

Starting Floorplan



- **After `create_fp_placement` the resulting macro placement can be "disorganized"**
  - May complicate the power structure
  - May use more routing resources for busses
- **You can define macro placement:**
  - Manually, using the GUI, prior to placement
  - With constraints, which guide placement

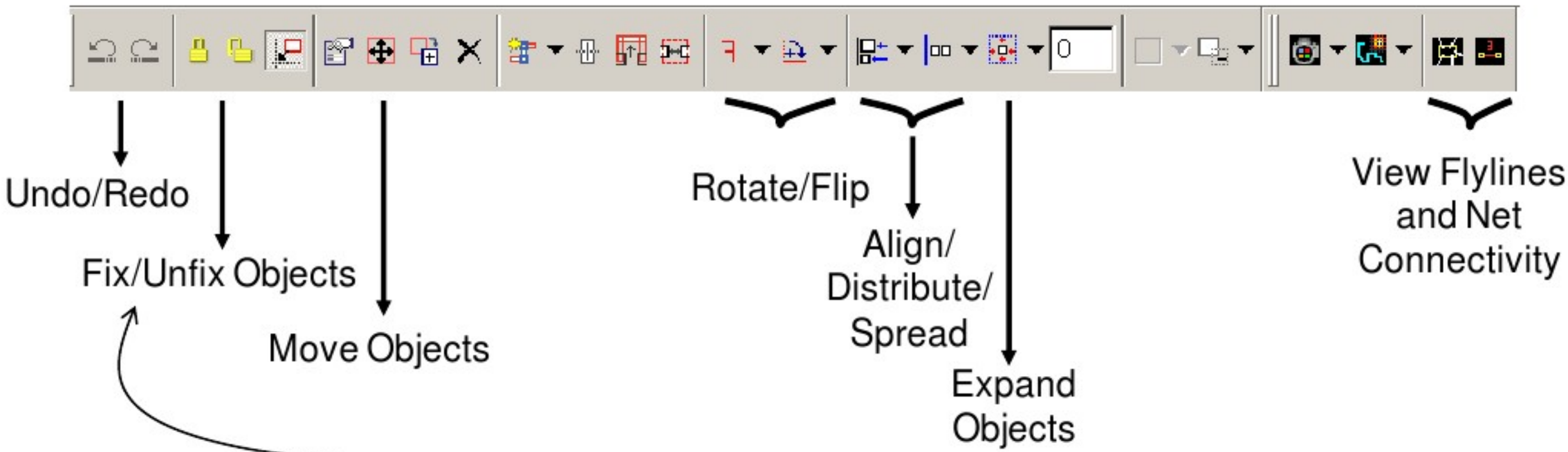
```
set_fp_macro_options ...
set_fp_macro_array ...
set_fp_relative_location ...
```

See Appendix A

# Manual Macro Placement



- **The GUI can be used to "manually" place macros in known locations and orientations (will be shown in lab)**




- **Placed macros , if not "fixed", can be moved by** `create_fp_placement`
  - To prevent this "fix" the placed macros using the toolbar, or:

```
set_dont_touch_placement [get_cells <list_of_cells>]
```

# Congestion Potential Around Macro Cells

Starting Floorplan

- **Routing to standard cells can often be difficult near edges or corners of macro cells**
- **Solution: Add placement blockages around macro cells**
  - *Hard* blockages prevent standard cells from <u>ever</u> being placed there
  - *Soft* blockages prevent standard cells from being placed during initial coarse placement, but are ignored by subsequent placement legalization or optimization

# Apply Global Placement Blockages

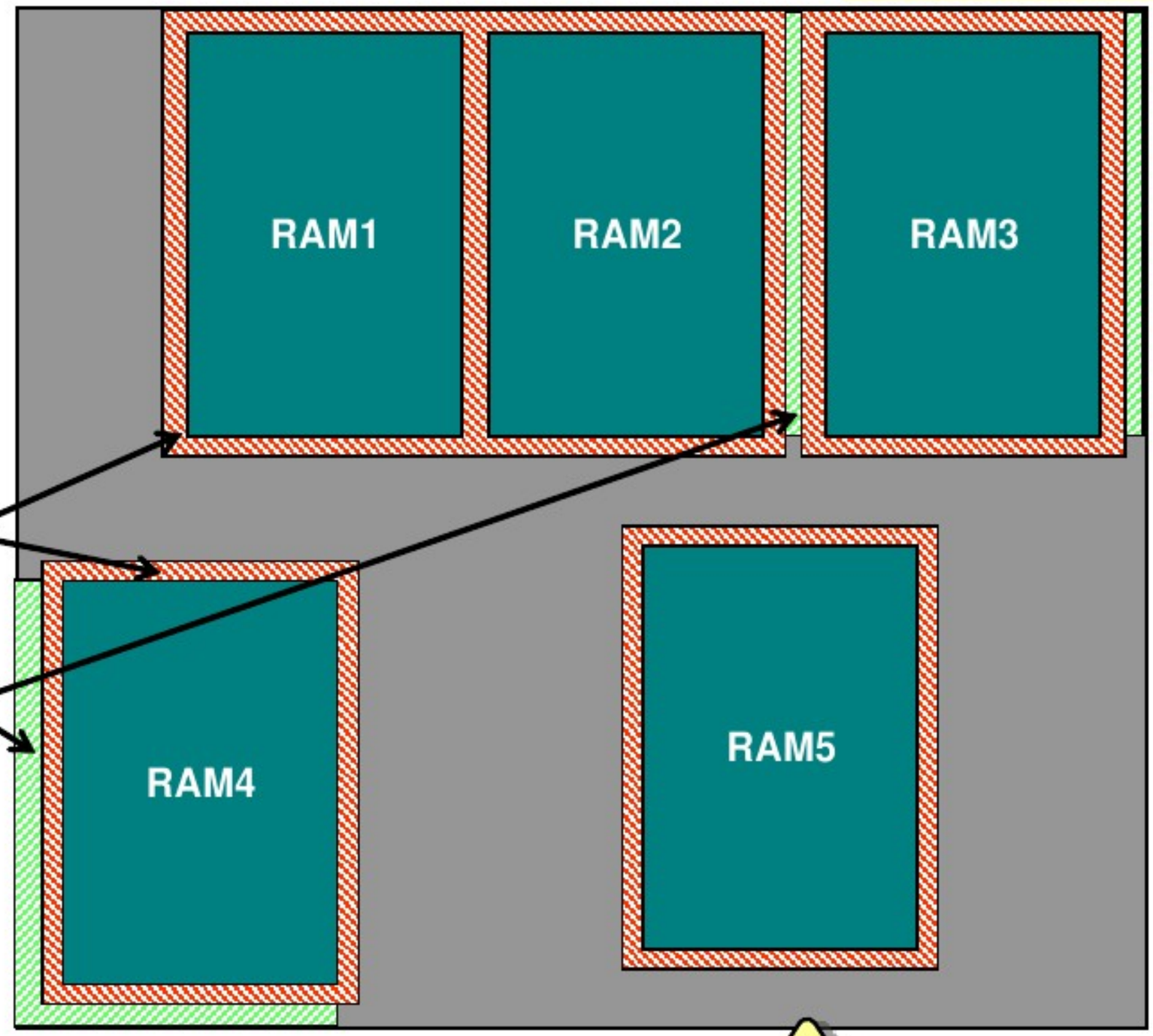
Starting Floorplan

## Global blockages

- Apply to all 'fixed' macros
- Consider all cell edges

Hard blockage always created on all four sides

Soft blockage created only for narrow channels between macros, or between a macro and the core boundary



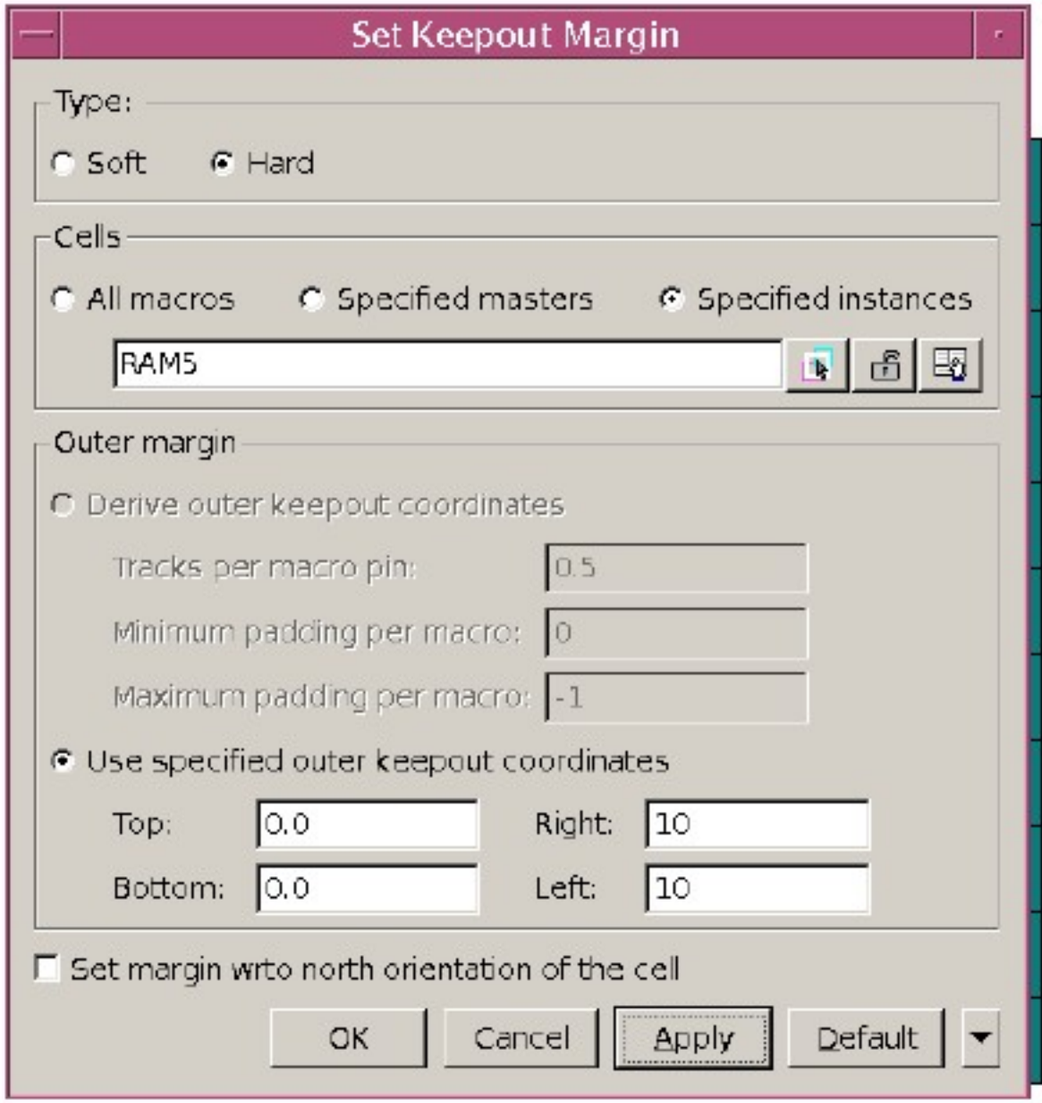

```
set_app_var physopt_hard_keepout_distance 10
set_app_var placer_soft_keepout_channel_width 25
```

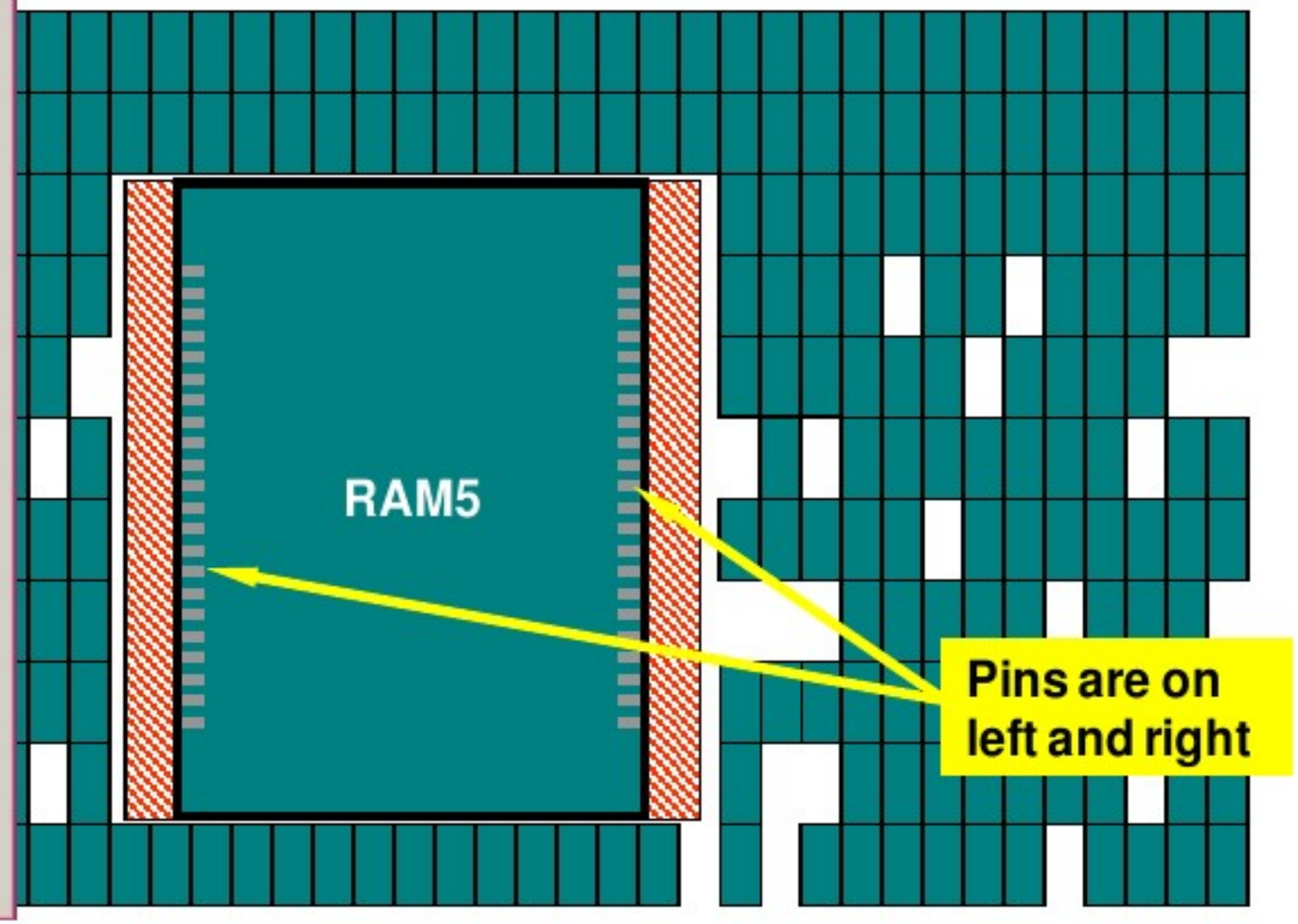
Also add to `.synopsys_dc.setup`

# Apply Specific Placement Blockages

Starting Floorplan

- **Pad the macro to reduce congestion around the pins**



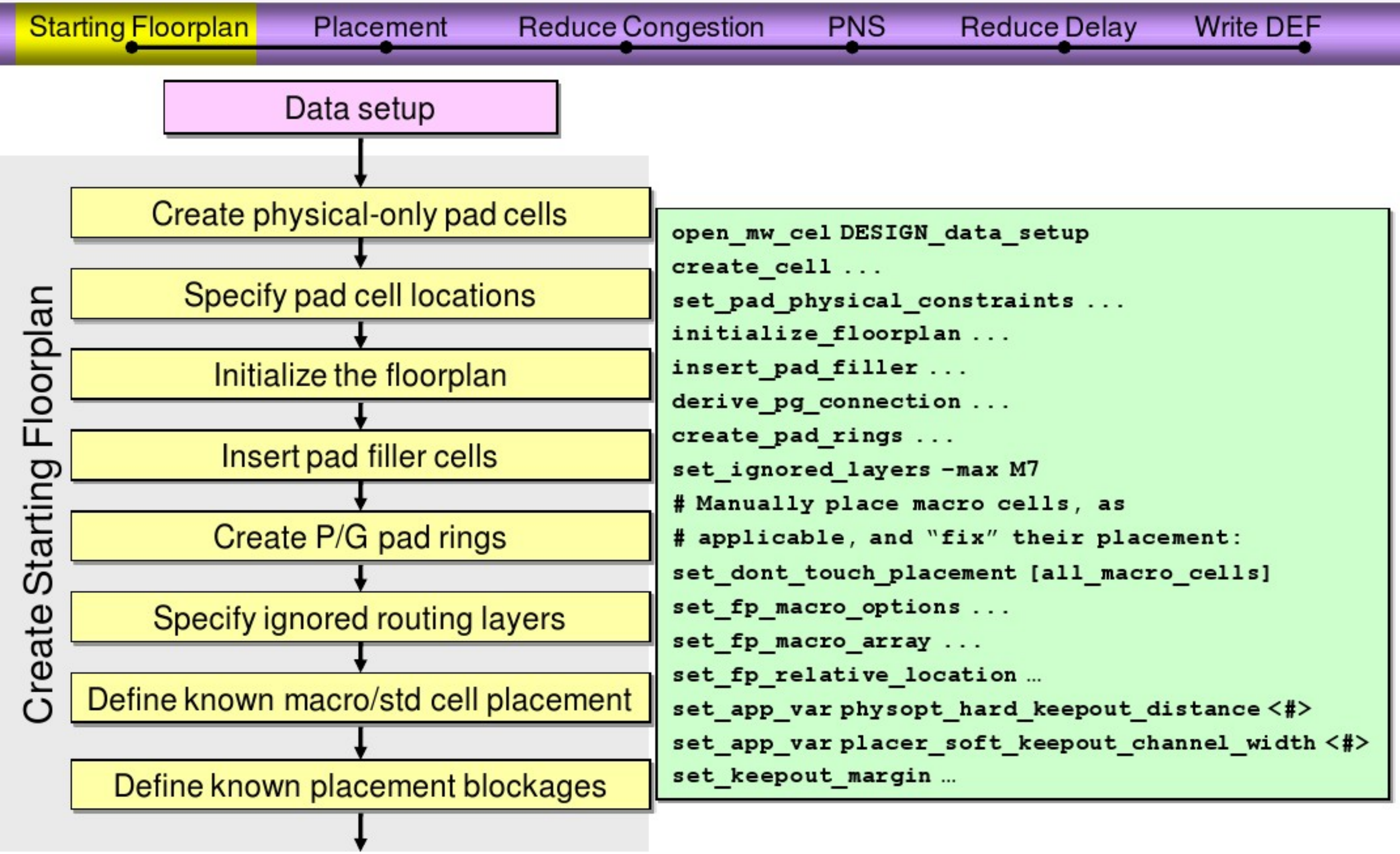




```
set_keepout_margin -type hard -outer {10 0 10 0} RAM5
```

hard or soft

{left bottom right top}

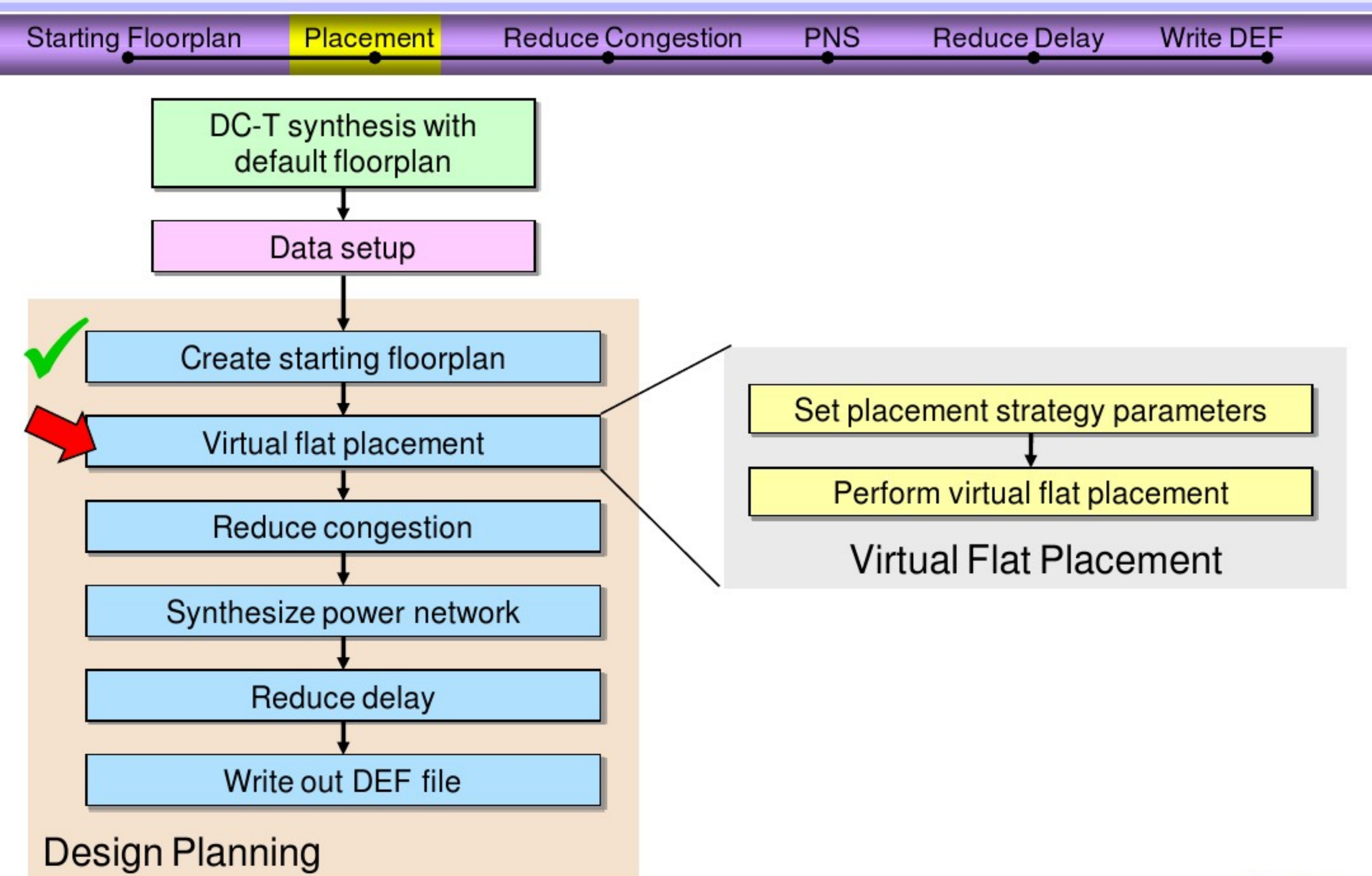
# Summary: Create the Starting Floorplan



# Test For Understanding

1. **What are "physical-only" cells? What are some examples of physical-only pad cells?**
2. **What does `initialize_floorplan` do? Circle all that apply:**
   a. Defines placement rows within the core area
   b. Defines the chip boundary or periphery area
   c. Places IO pad cells in their defined locations
   d. Places macro cells per their placement constraints
3. **A "soft" placement blockage will:**
   a. Allow only timing-critical cells to be placed
   b. Allow only non timing-critical cells to be placed
   c. Allow cell placement only during initial coarse placement
   d. Prevent cell placement during initial coarse placement

# Perform Virtual Flat Placement



# Set Placement Strategy Parameters



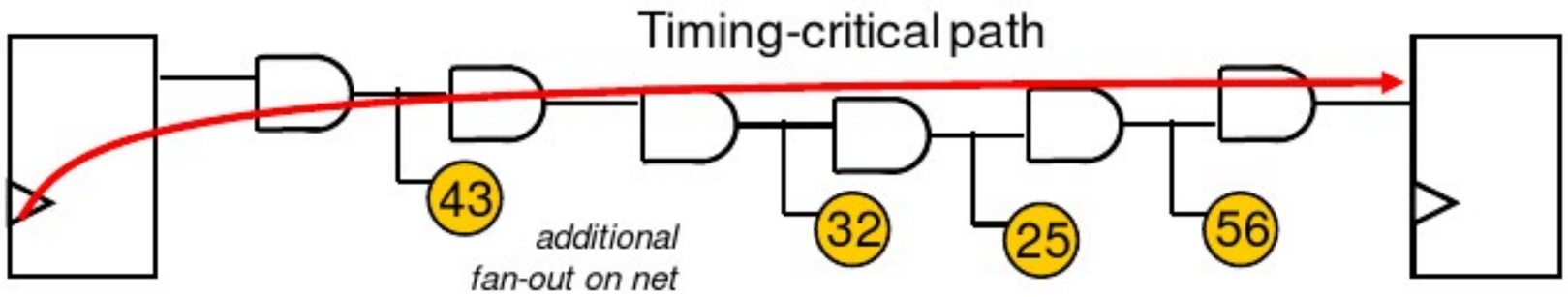
- **Placement strategy parameters are used to control the following during `create_fp_placement`:**
  - How macros are handled
  - Optimization algorithms and effort
- **Consider setting a "sliver size" and turning on "virtual IPO":**

```
report_fp_placement_strategy
set_fp_placement_strategy -sliver_size 10
                          -virtual_IPO on
```

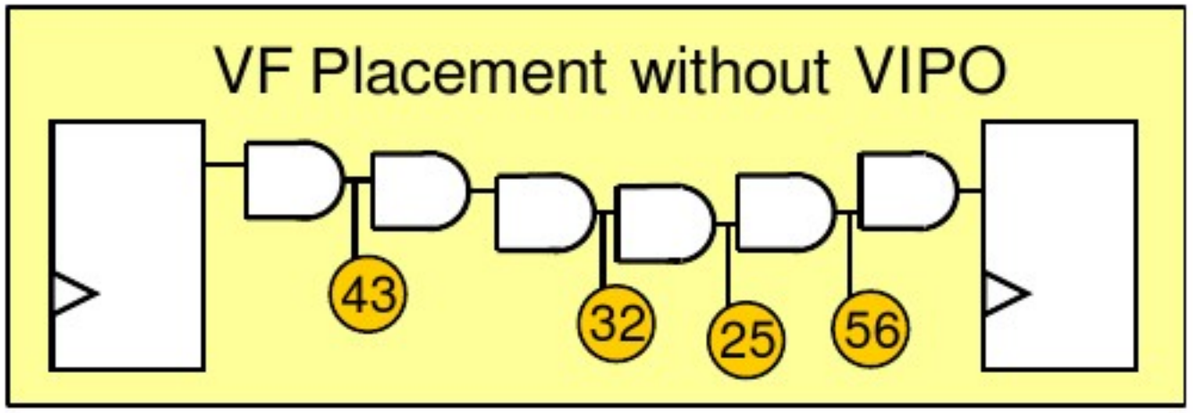
  - In the example above standard cells will not be placed in channels between macros (*slivers*) of less than 10 microns
    - Helps to reduce potential congestion between macros
  - Next slide: Explanation of *virtual in-place optimization* (*VIPO*)
  - Use default settings for the remaining options (see notes below) unless you have a specific need to modify them

# VF Placement with *Virtual IPO* (VIPO)

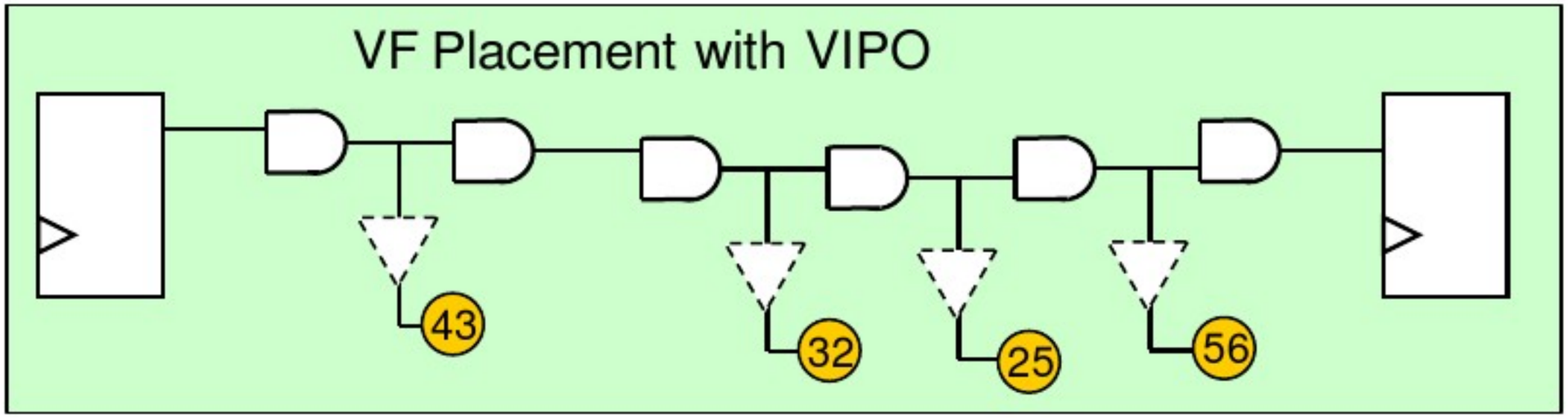





Original netlist contains large fan-outs along the critical path which can contribute significantly to delay




Without VIPO, cells along the critical path are placed close together to reduce delay – this can cause congestion and may not be necessary, since IPO during actual placement optimization can easily fix this



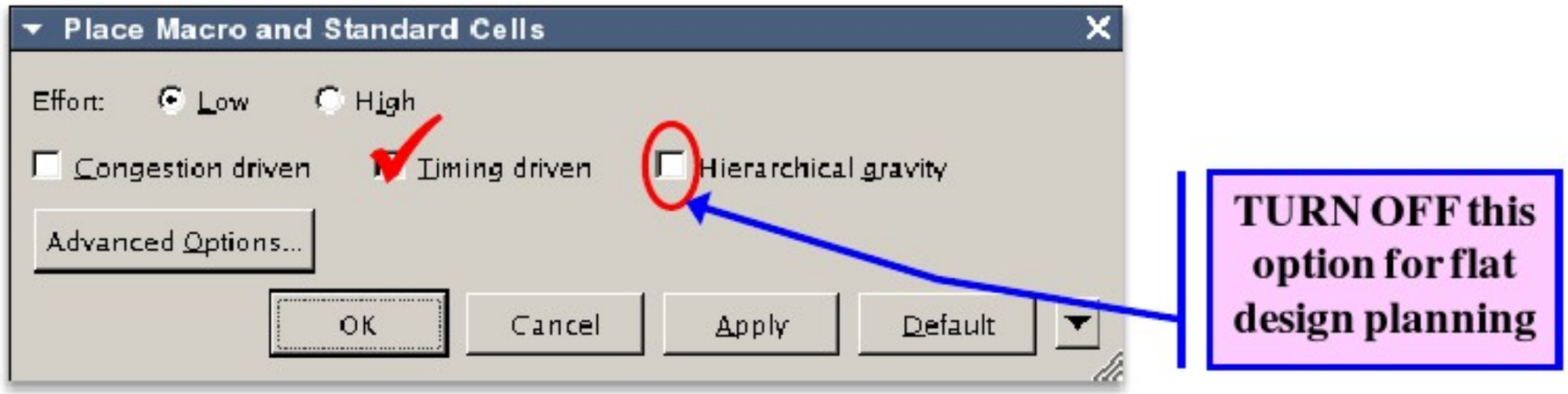

With VIPO turned ON, sizing and buffer insertion (which mimics more realistic placement optimization) is performed "in memory", thus relaxing the need to unnecessarily shorten the distance between the critical cells

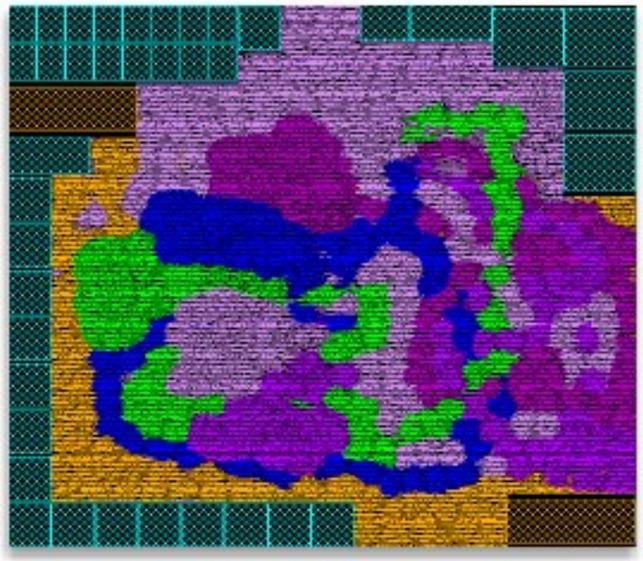
# Perform Virtual Flat Placement



## Perform quick placement, which will be used to:

- Improve floorplan's impact on congestion and timing
- Plan the power structure



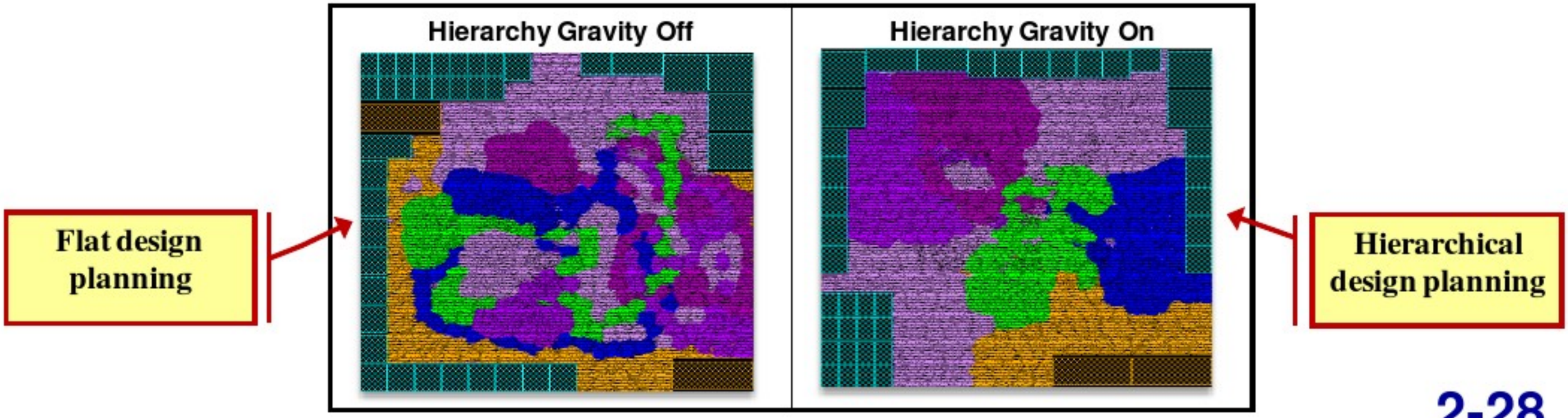
```
create_fp_placement -timing_driven -no_hierarchy_gravity
```

- Standard cells and non-fixed macros are legally placed
- By default placement is wirelength-driven → enable timing-driven
- No logic optimization is done
- By default clumps cells within same logical hierarchy → turn gravity off

# Hierarchy Aware Placement or Gravity

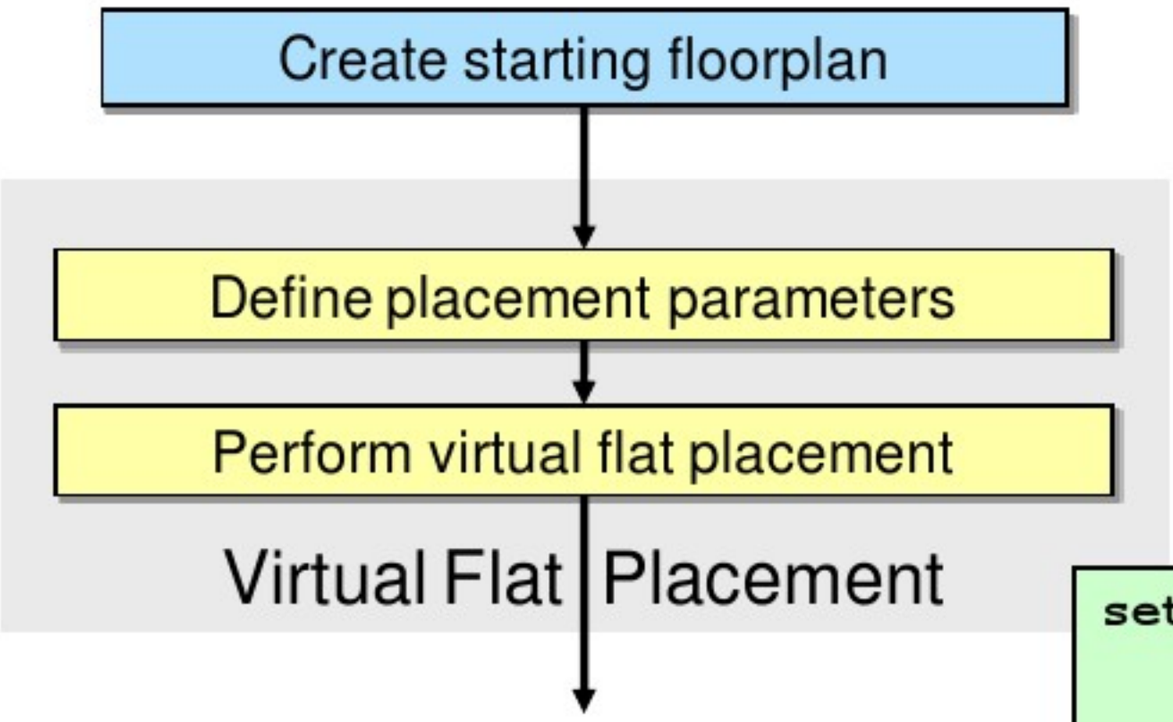


- **By default `create_fp_placement` performs *hierarchy-aware placement* (gravity = ON)**
  - Placement tries to keep cells in the same logical hierarchy blocks physically close
  - The resulting placement becomes a useful guide to form physical "partitions" in a hierarchical design flow (discussed in the "*IC Compiler 2: HDP*" workshop)
- **In a non-hierarchical design flow (this workshop) "gravity" should be turned off**



# Summary: Virtual Flat Placement

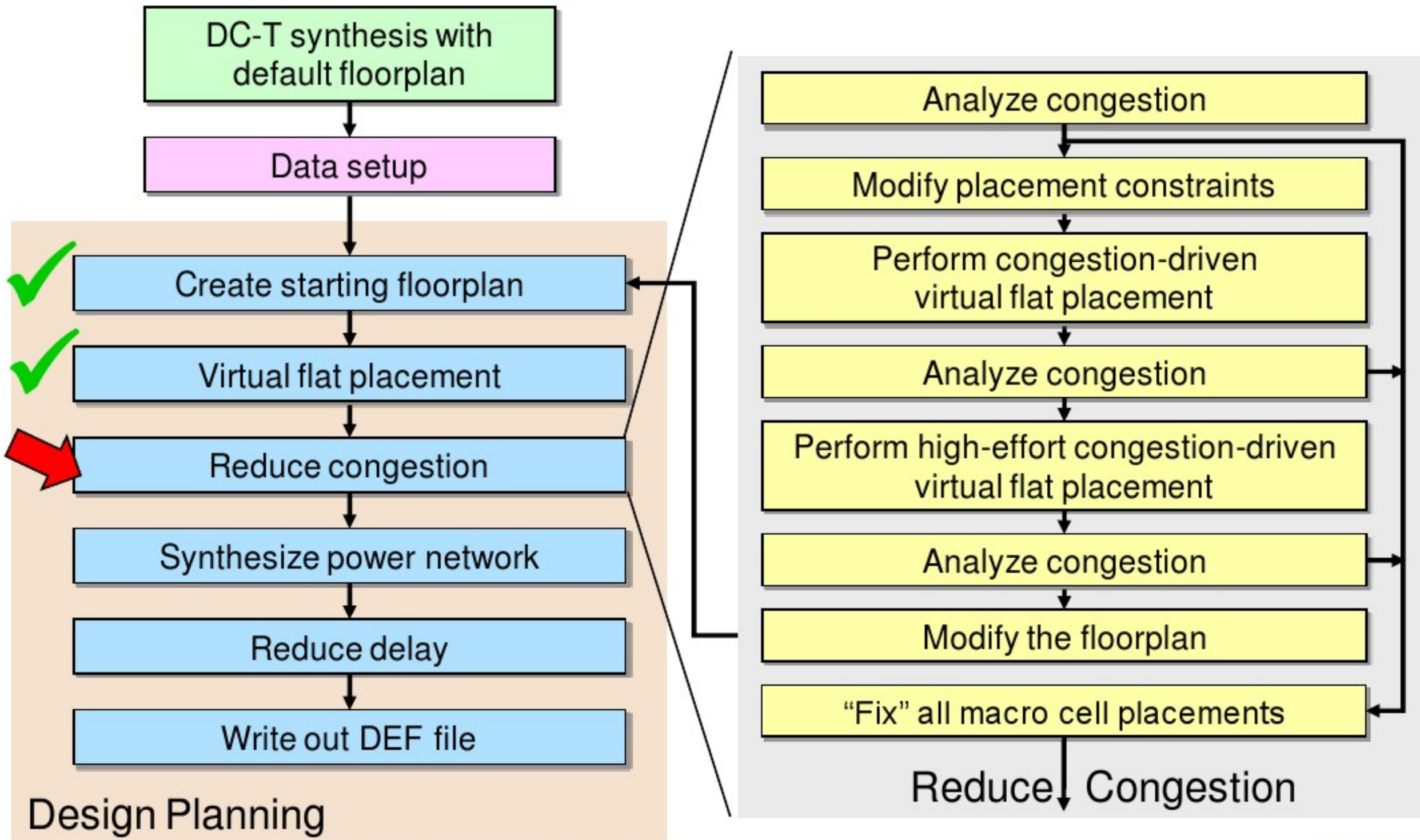





```
set_fp_placement_strategy -sliver_size 10 \
                          -virtual_IPO on
create_fp_placement -no_hierarchy -timing
```

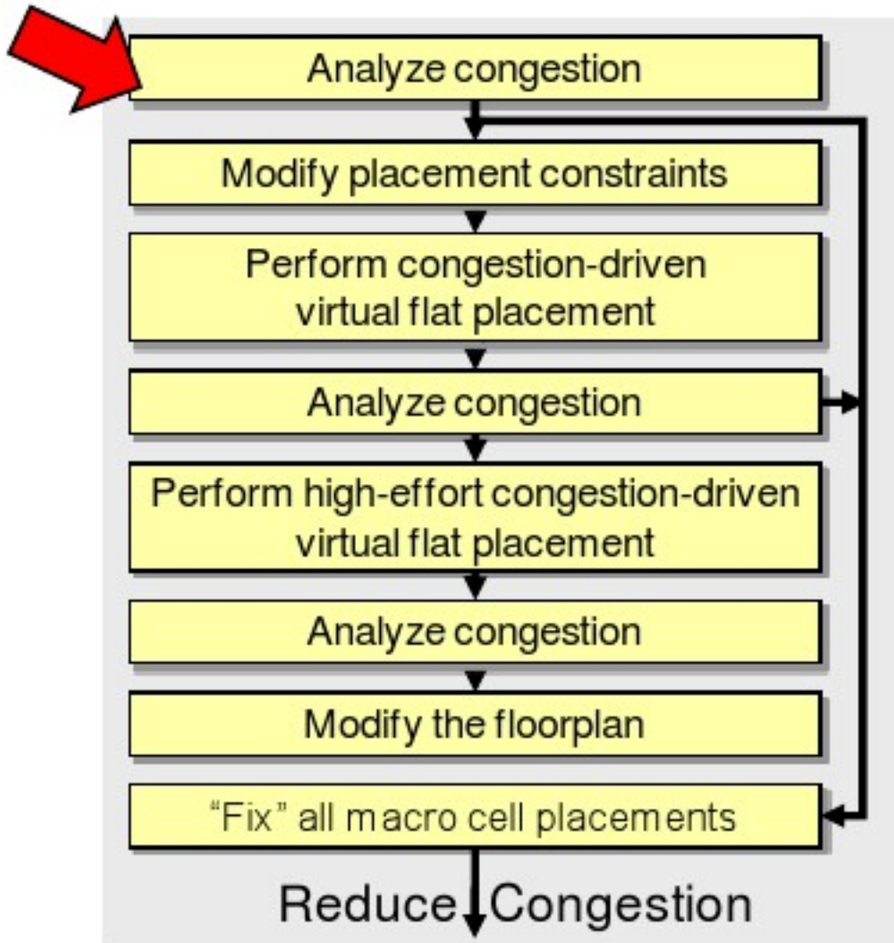
# Reduce Congestion

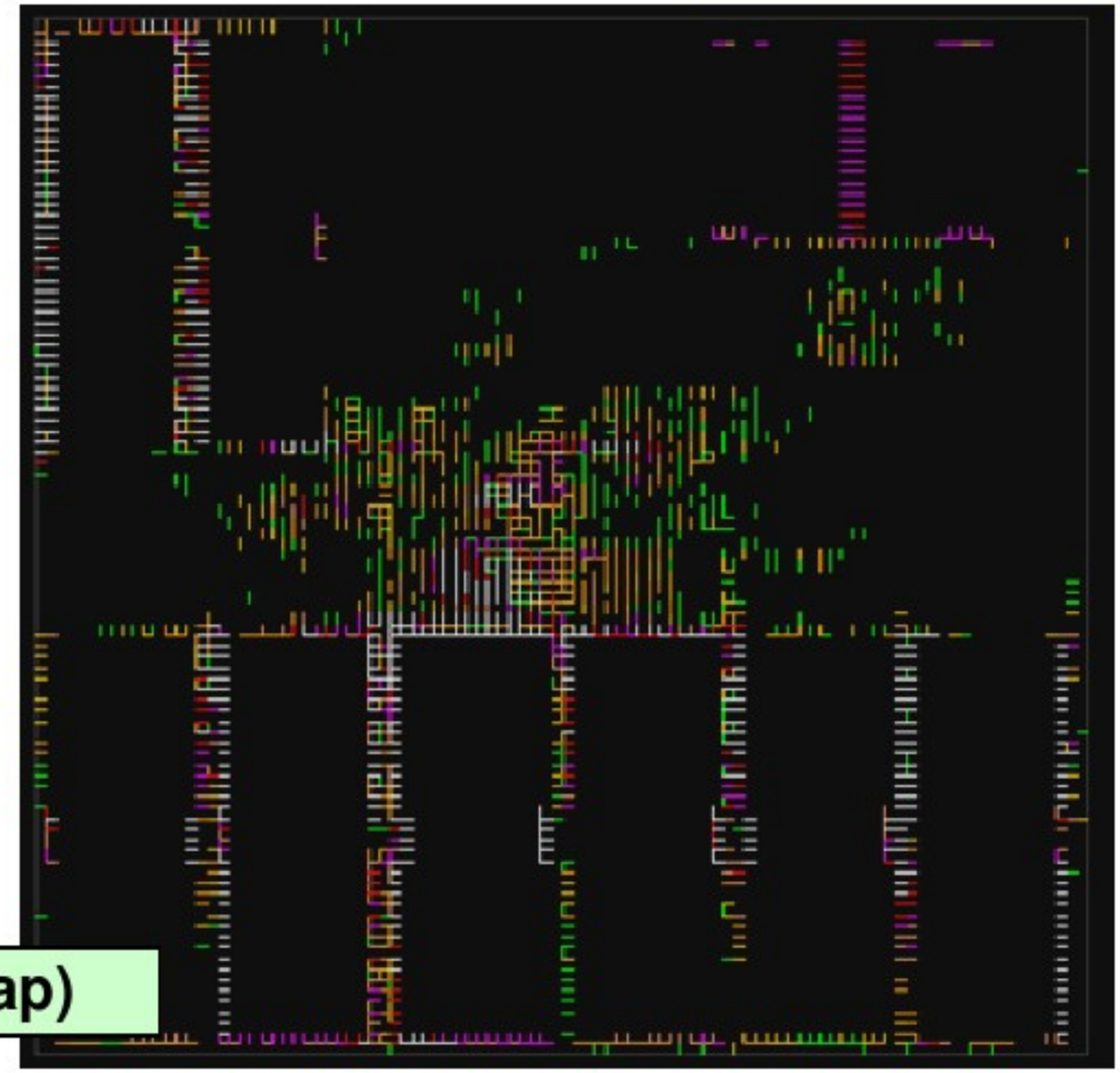




# Is the Design Congested?

Reduce congestion



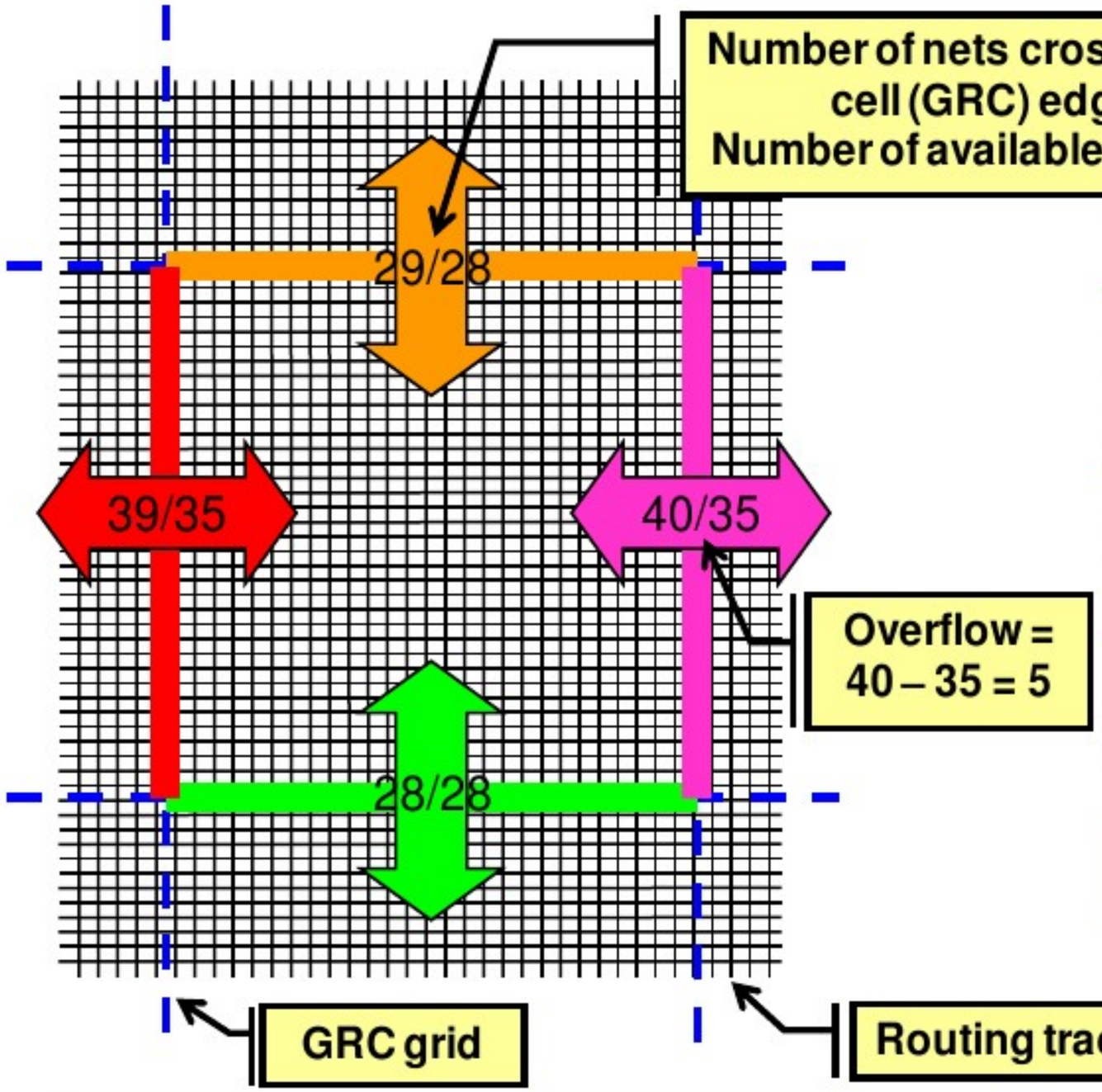



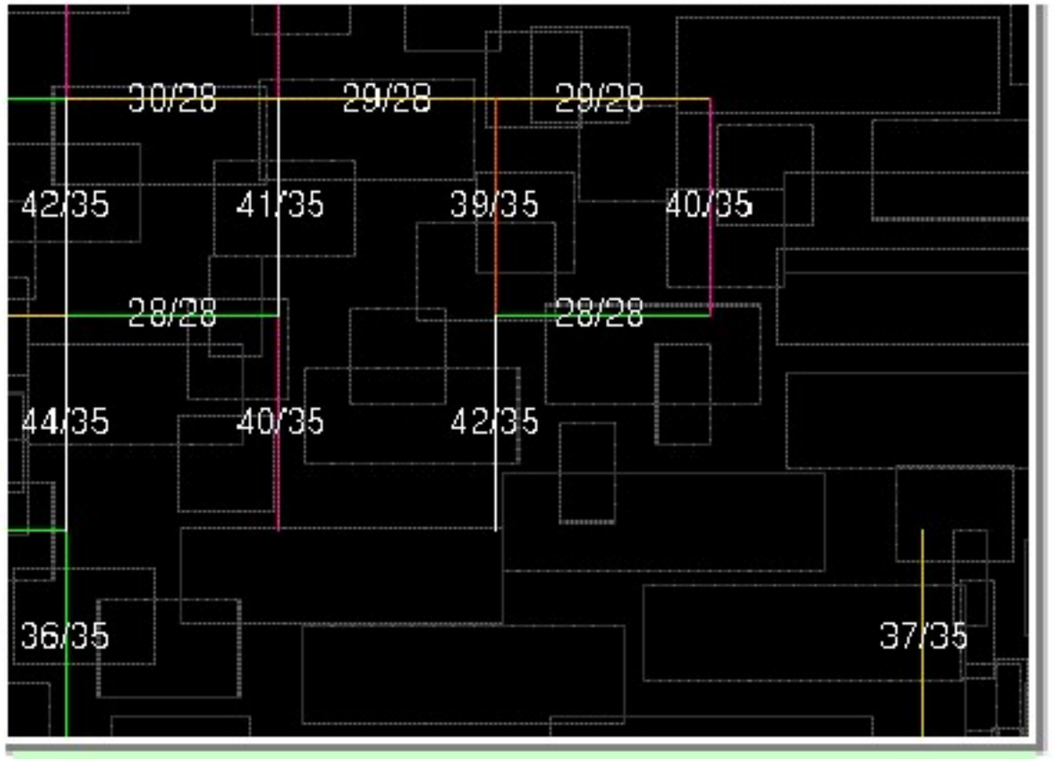
Congestion map (heat map)

```
report_congestion -grc_based -by_layer \
                  -routing_stage global
```

# Understanding the Congestion Calculation

Reduce congestion




Congestion map (heat map)

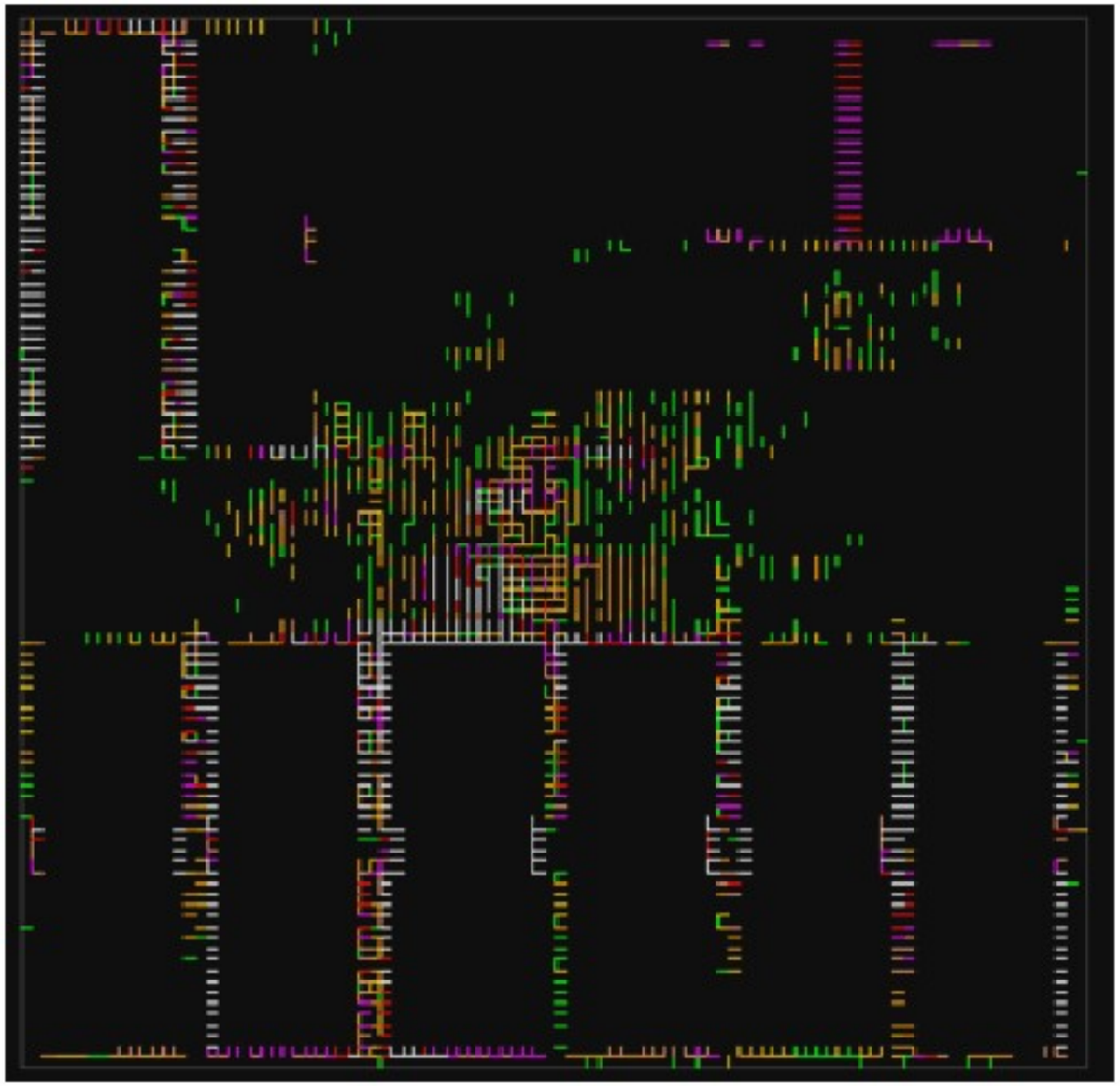
**The size of the 'overflow' determines the highlighting color – larger overflows are brighter or 'hotter'**

# Congestion Guidelines

Reduce congestion

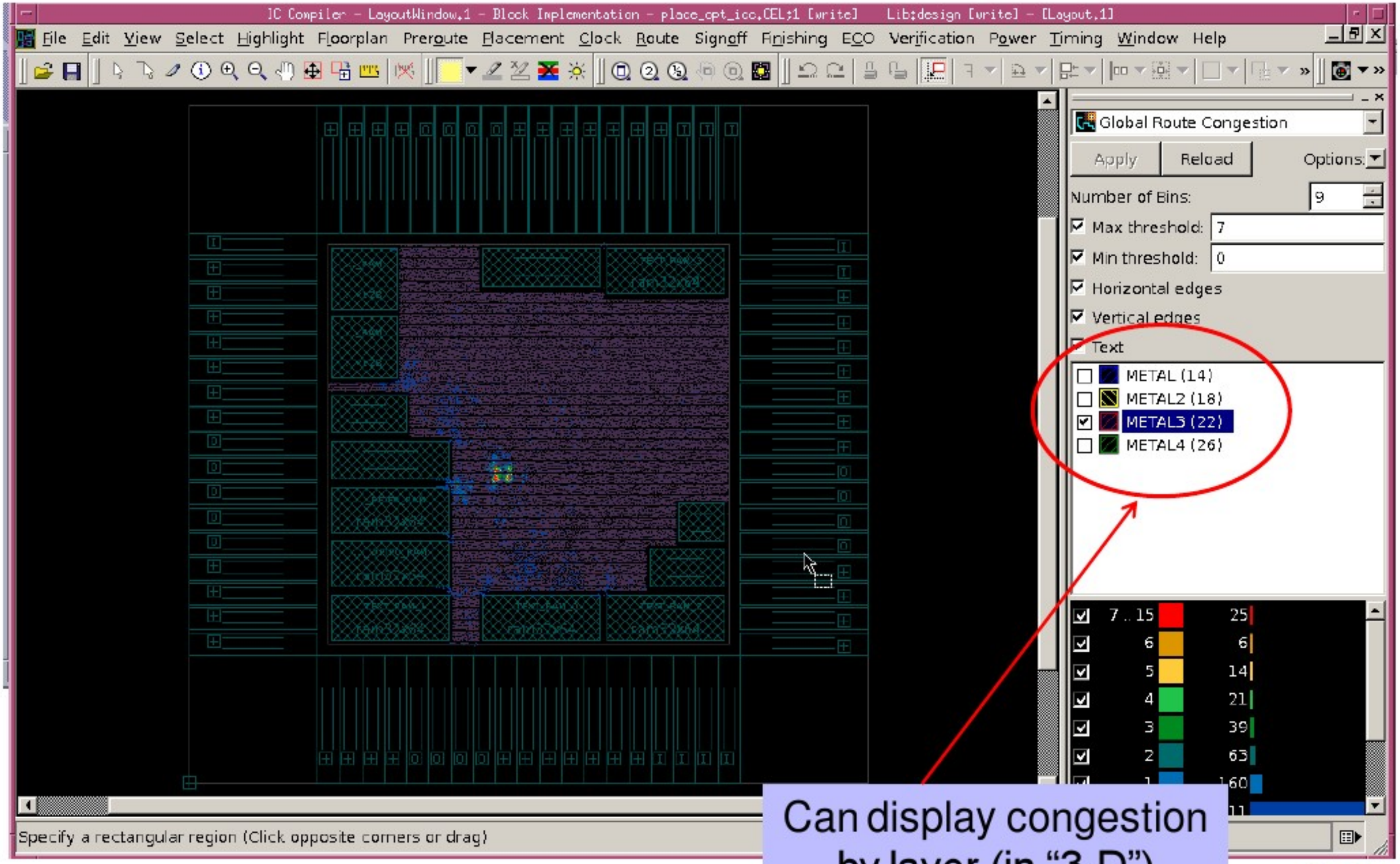
**Congestion can be serious or "un-acceptable" if one or more of the following are true:**

- **Large or many "hotspots" in the congestion map**
  - Serious routability challenge
- **Any GRC "overflow" larger than around 10**
  - Possible non-routable nets
- **About ~2%, or more, GRC edges have an overflow**
  - Possible signal integrity or timing degradation issues



- **If congestion is not a problem move on to "PNS", otherwise ….**

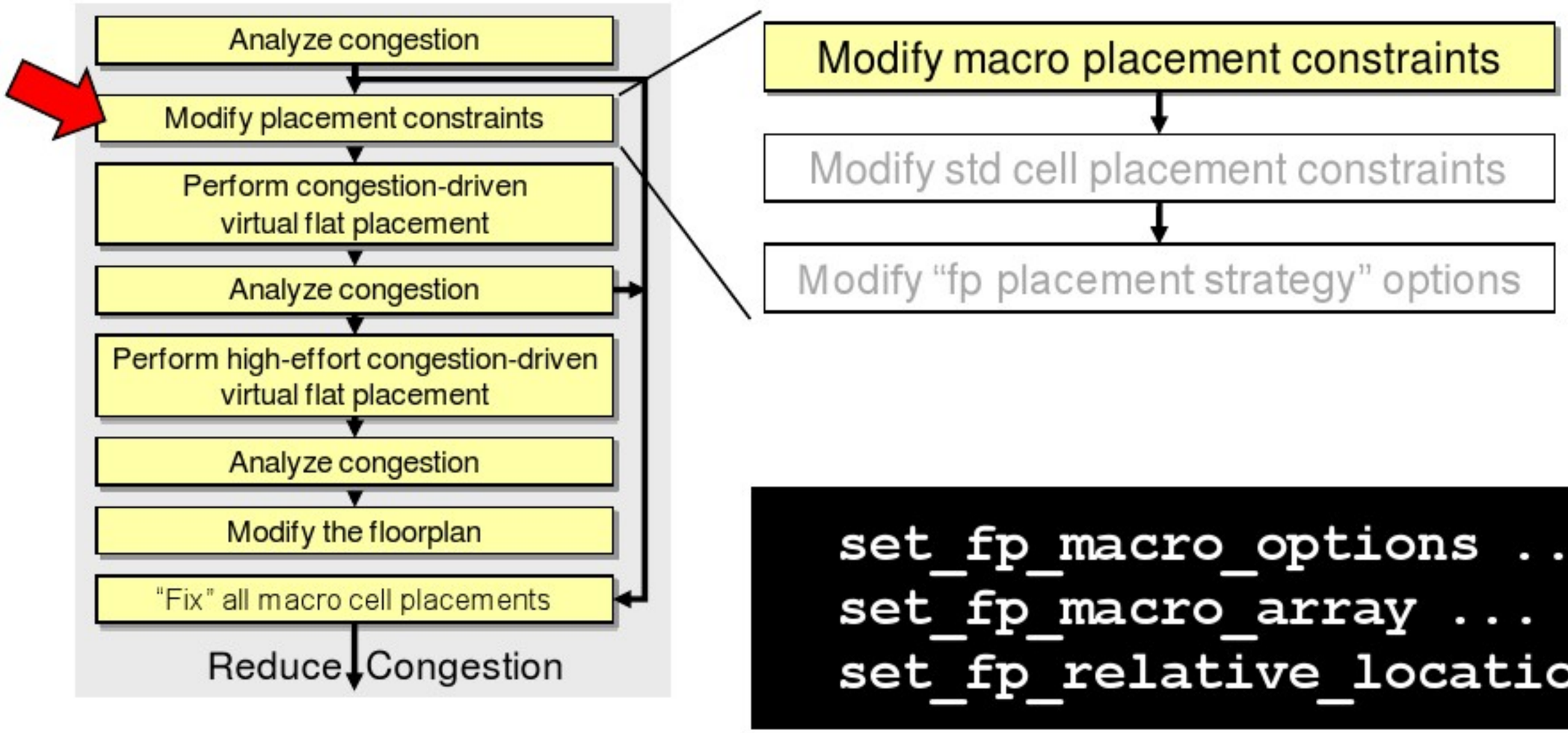
# Can Analyze Congestion Layer-by-Layer




Can display congestion by layer (in "3-D")

# Modify Macro Placement Constraints

Reduce congestion

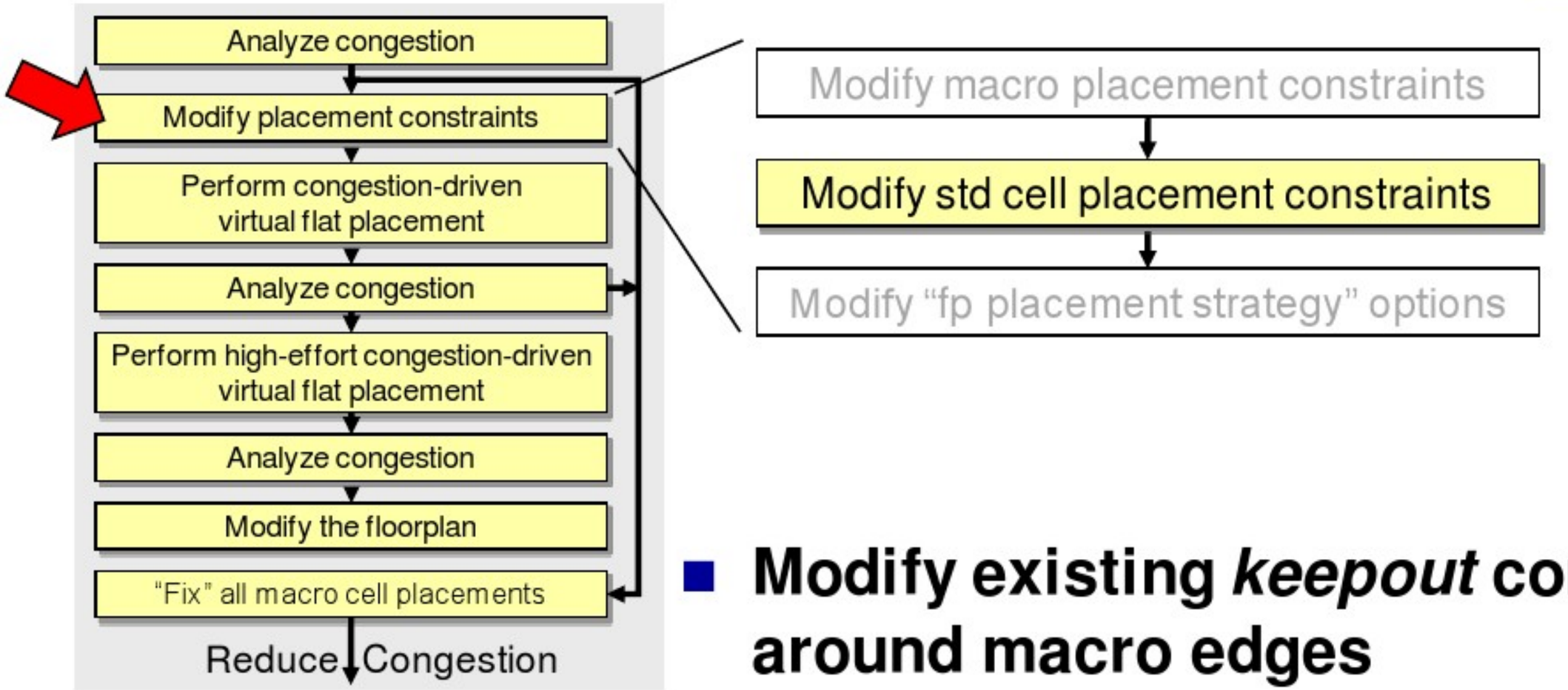



```
set_fp_macro_options ...
set_fp_macro_array ...
set_fp_relative_location ...
```

- Increase spacing between macros
- Align bus signal pins between macros
- Change orientation (rotate and/or mirror)

# Apply Standard Cell Placement Constraints

Reduce congestion




- **Modify existing *keepout* constraints around macro edges**

```
set_app_var physopt_hard_keepout_distance 10
set_app_var placer_soft_keepout_channel_width 25
set_keepout_margin -type hard -outer {10 0 10 0} RAM5
```

Discussed earlier

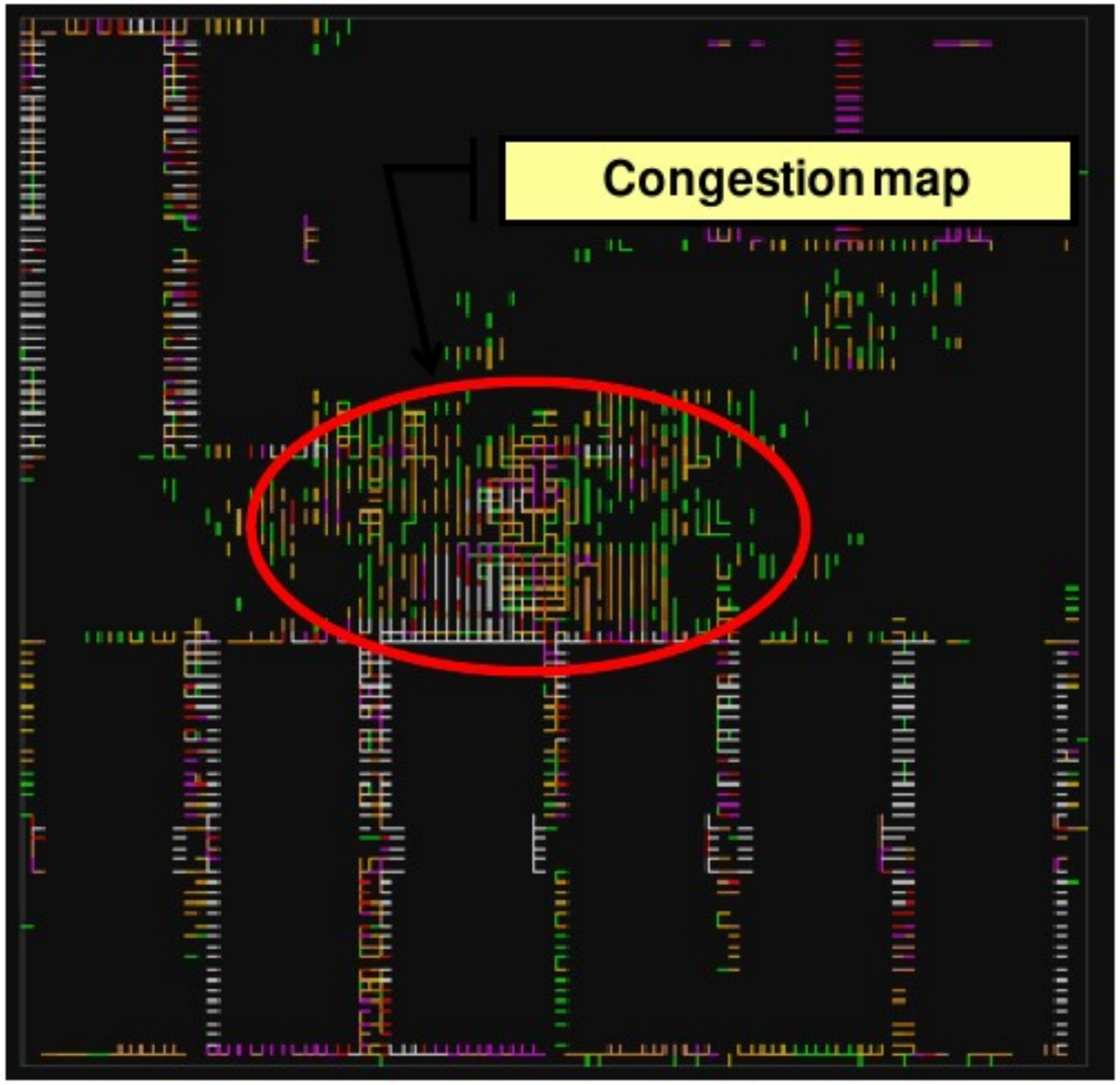
- **Apply additional placement constraints**

```
set_congestion_options ...
create_placement_blockage ...
```

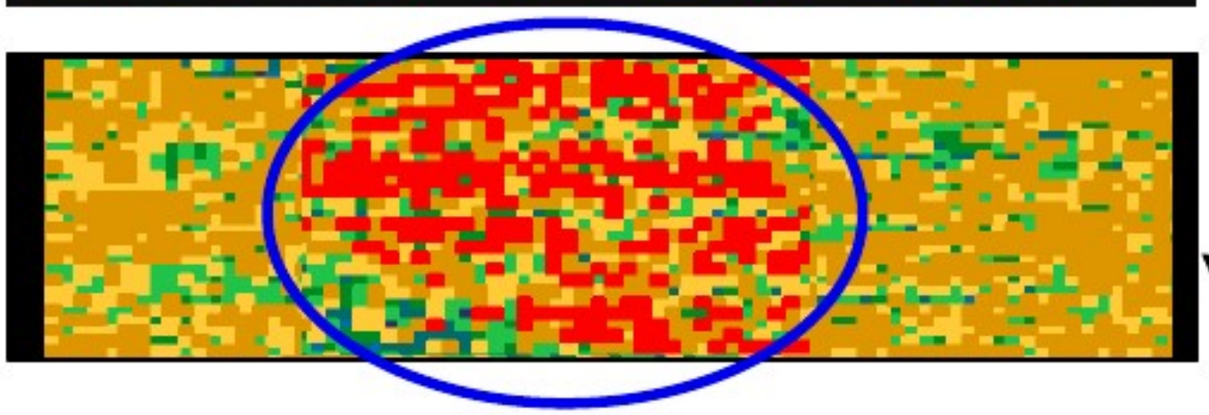
Shown next

# Is High Cell Density Causing Congestion?

Reduce congestion



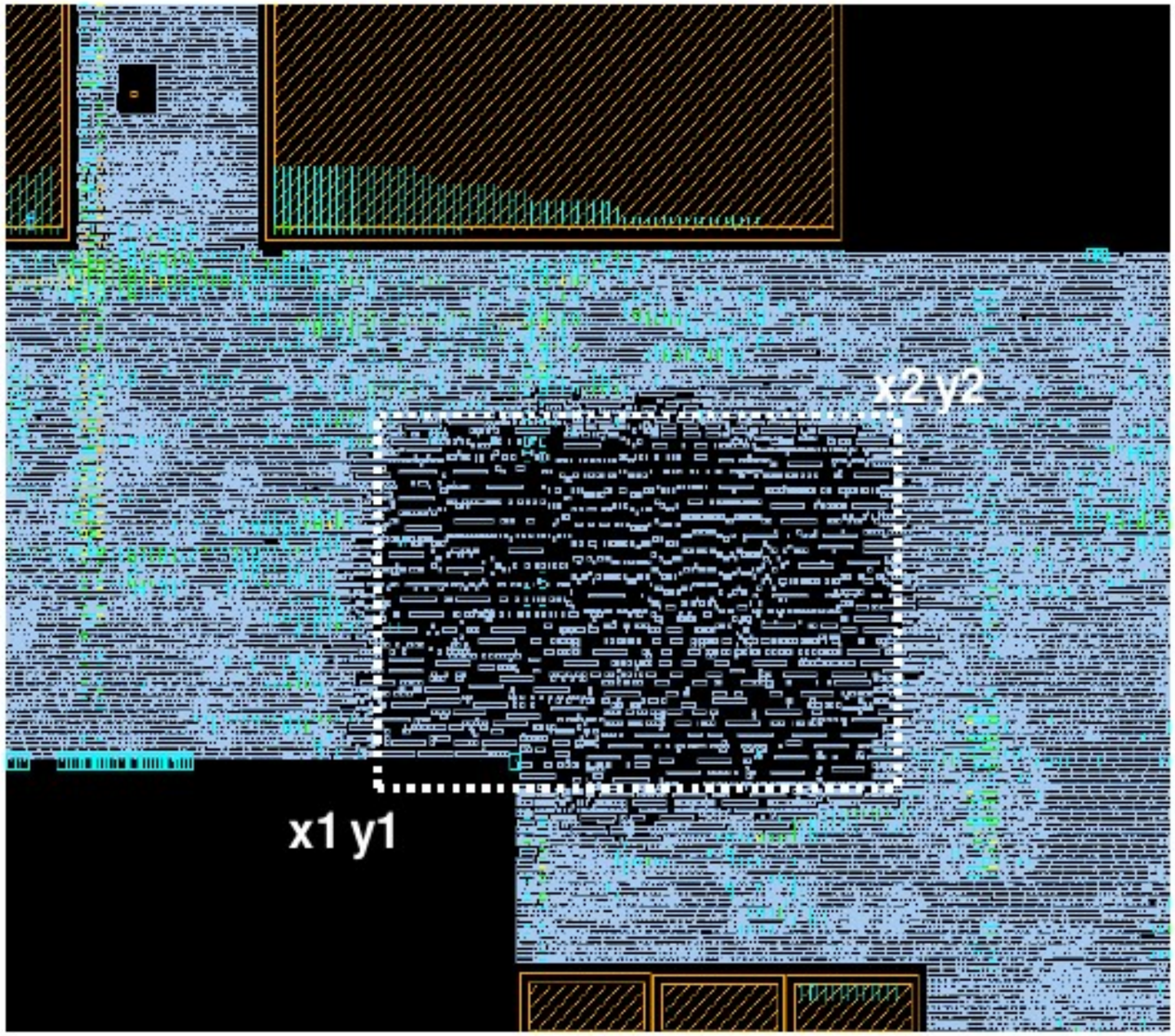

- **Cell density can be up to 95% in any given area of the chip or block, by default**
- **High cell density may be causing high congestion**
- **Look at a cell density map**
  - If cell density hotspots coincide with congestion hotspots → Lower cell density in that area



Cell density map

# Reducing Cell Density Hotspots

Reduce congestion

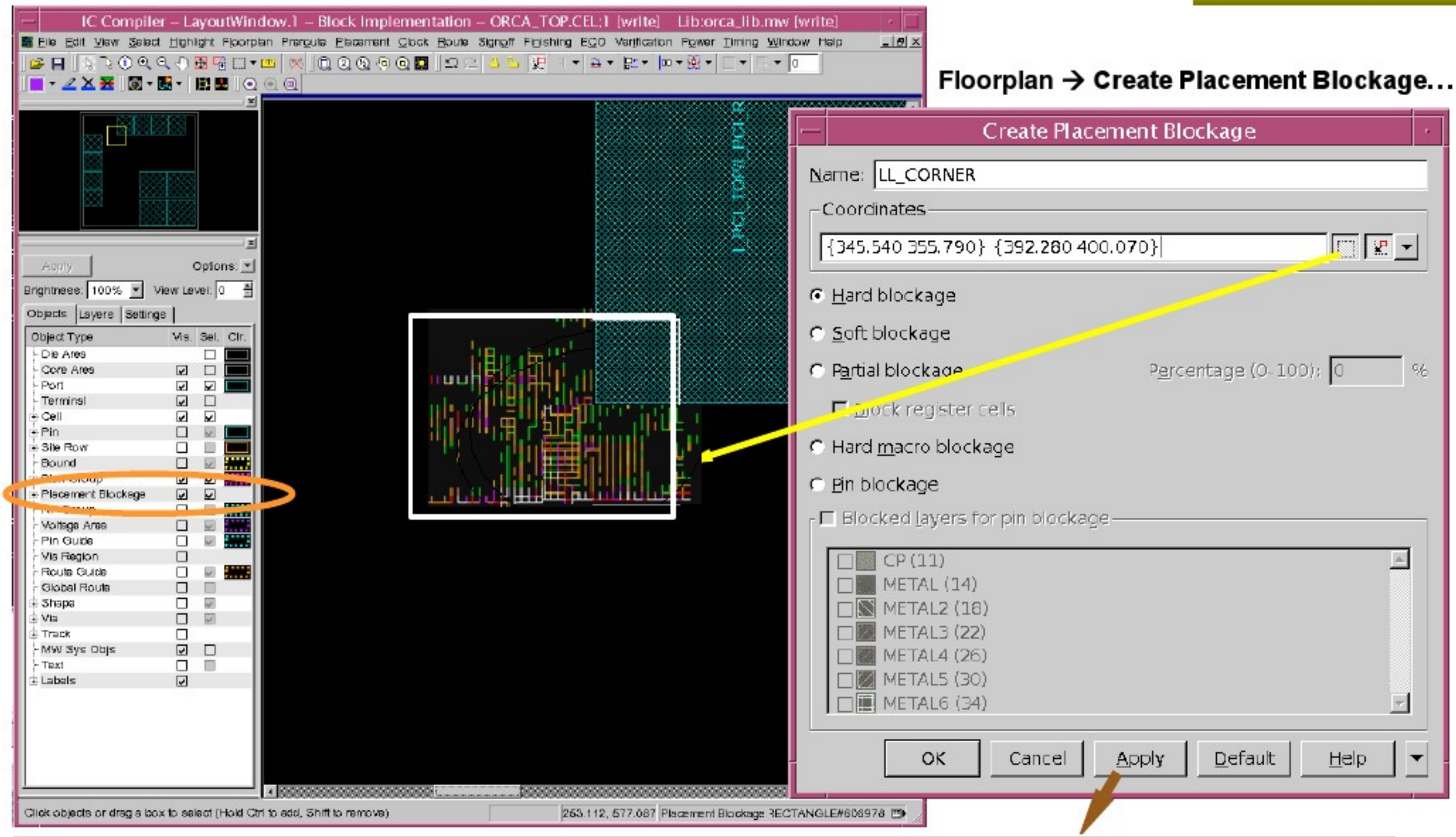



```
set_congestion_options -max_util 0.4 \
    -coordinate {x1 y1 x2 y2}
```

# Coordinate-based Placement Blockages

Reduce congestion

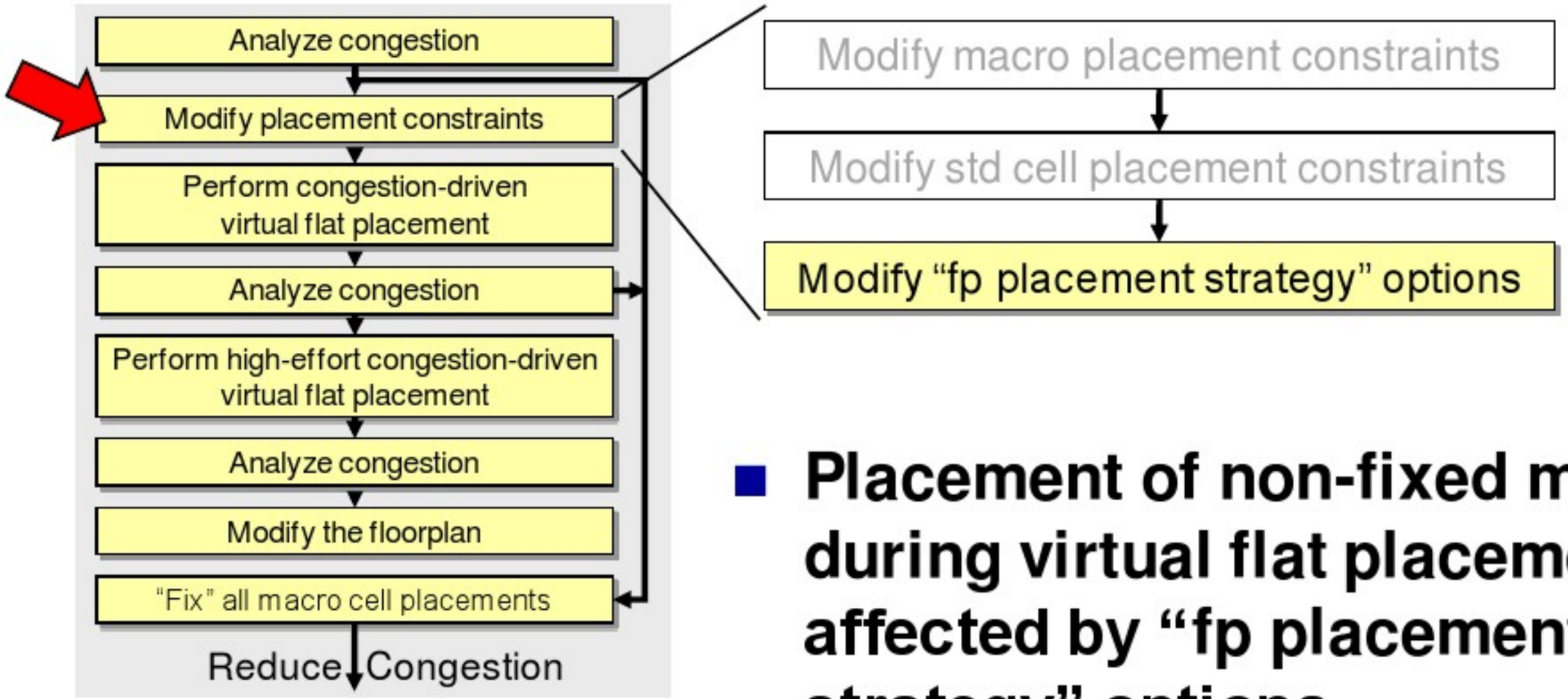
**Floorplan → Create Placement Blockage…**




```
create_placement_blockage -name LL_CORNER -type hard \
                -bbox {345.540 355.790 392.280 400.070}
```

# Modify "FP Placement Strategy" Options

Reduce congestion




- **Placement of non-fixed macros during virtual flat placement are affected by "fp placement strategy" options**
  - **Up to now default options were used, except for**

```
set_fp_placement_strategy -sliver_size 10 \
                          -virtual_IPO on
```

  - **Modify macro placement strategy options, as appropriate, to help reduce congestion**

# Placement Strategy Options and Defaults

Reduce congestion

```
set_fp_placement_strategy
   -macro_orientation <automatic | all | N>
   -auto_grouping <none | user_only | low | high>
   -macro_setup_only <on | off>
   -macros_on_edge <on | off>
   -sliver_size <0.00>
   -snap_macros_to_user_grid <on | off>
   -fix_macros <none | soft_macros_only | all>
   -congestion_effort <low | medium | high>
   -adjust_shapes <on | off>
   -IO_net_weight <1.0>
   -plan_group_interface_net_weight <1.0>
   -legalizer_effort <low | high>
   -spread_spare_cells <on | off>
   -legalizer_effort <low | high>
   -virtual_IPO <on | off>
```

Some examples shown next

# Macro Placement Strategy Examples

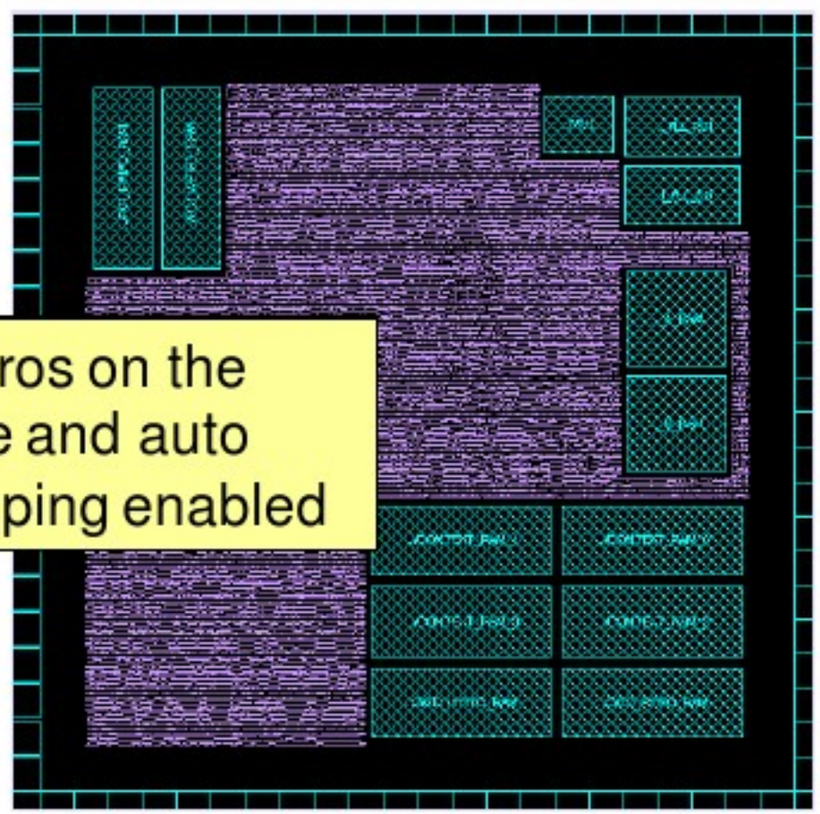
Reduce congestion

- Push macros to core edges for congestion control

```
set_fp_placement_strategy -macros_on_edge on
```

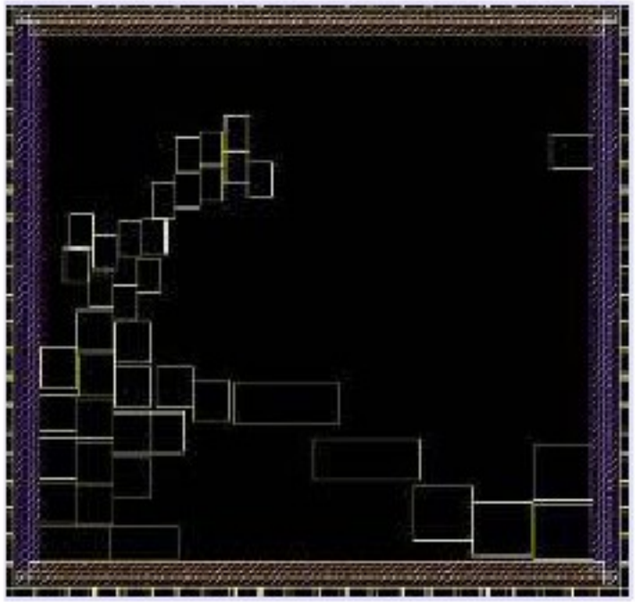
- Group related macros into an array to improve timing and congestion

```
set_fp_placement_strategy -auto_grouping high
```

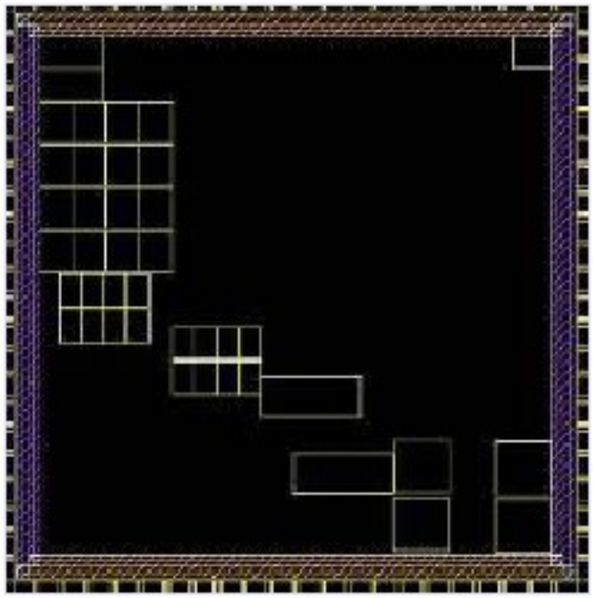


Macros on the edge and auto grouping enabled

-auto_grouping none

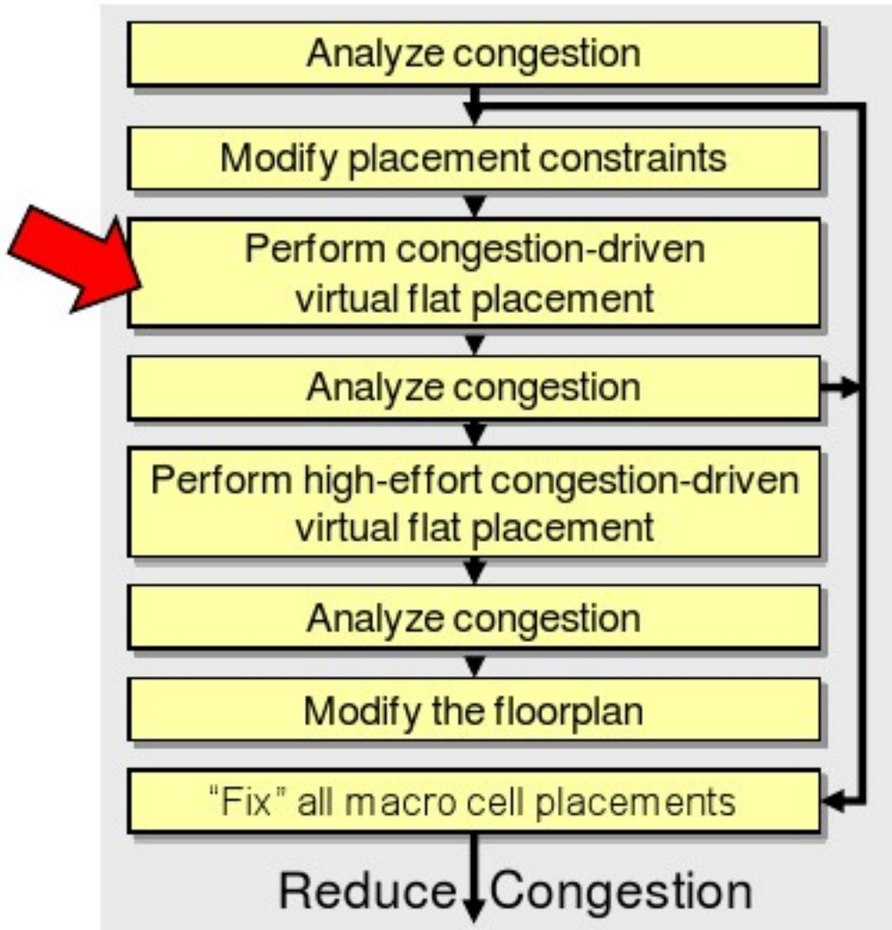


-auto_grouping high



# Perform *Congestion-driven* Placement

Reduce congestion



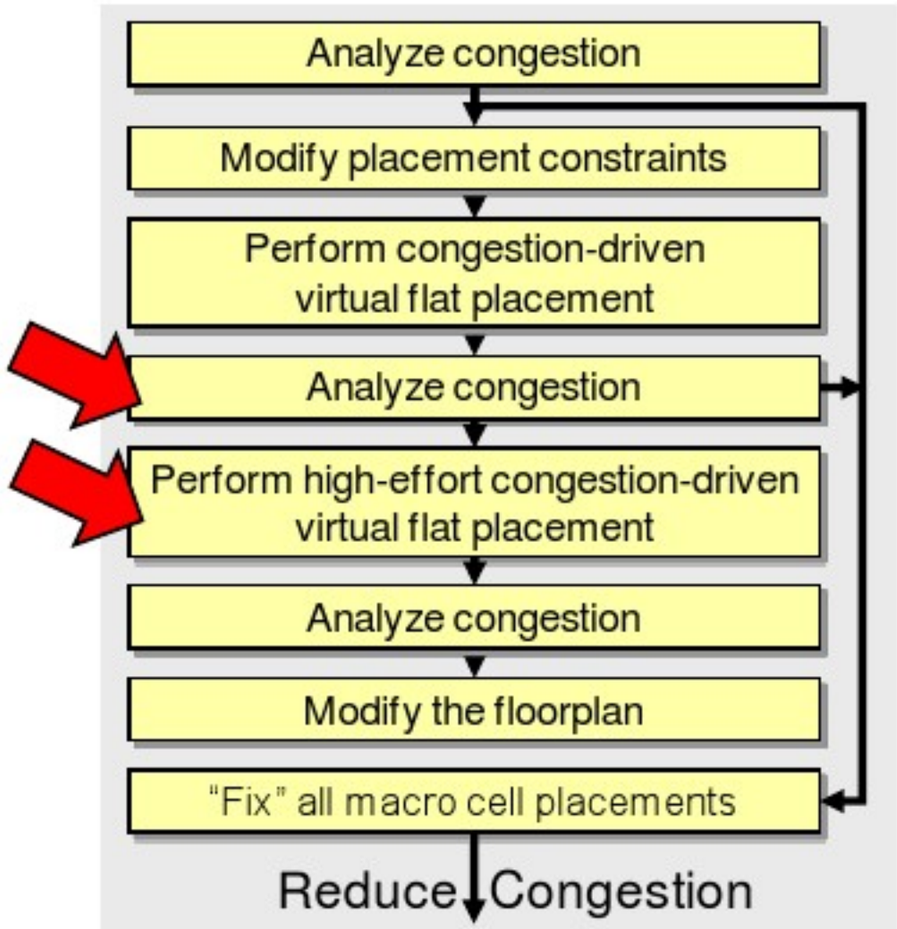

- **After modifying the various placement constraints perform another virtual flat placement**
- **Enable congestion-driven mode**

```
create_fp_placement -timing -no_hierarchy_gravity \
                    -congestion
```

**Do not use congestion-driven mode on designs with no congestion!**

# Invoke the *High Effort* Congestion Strategy

Reduce congestion



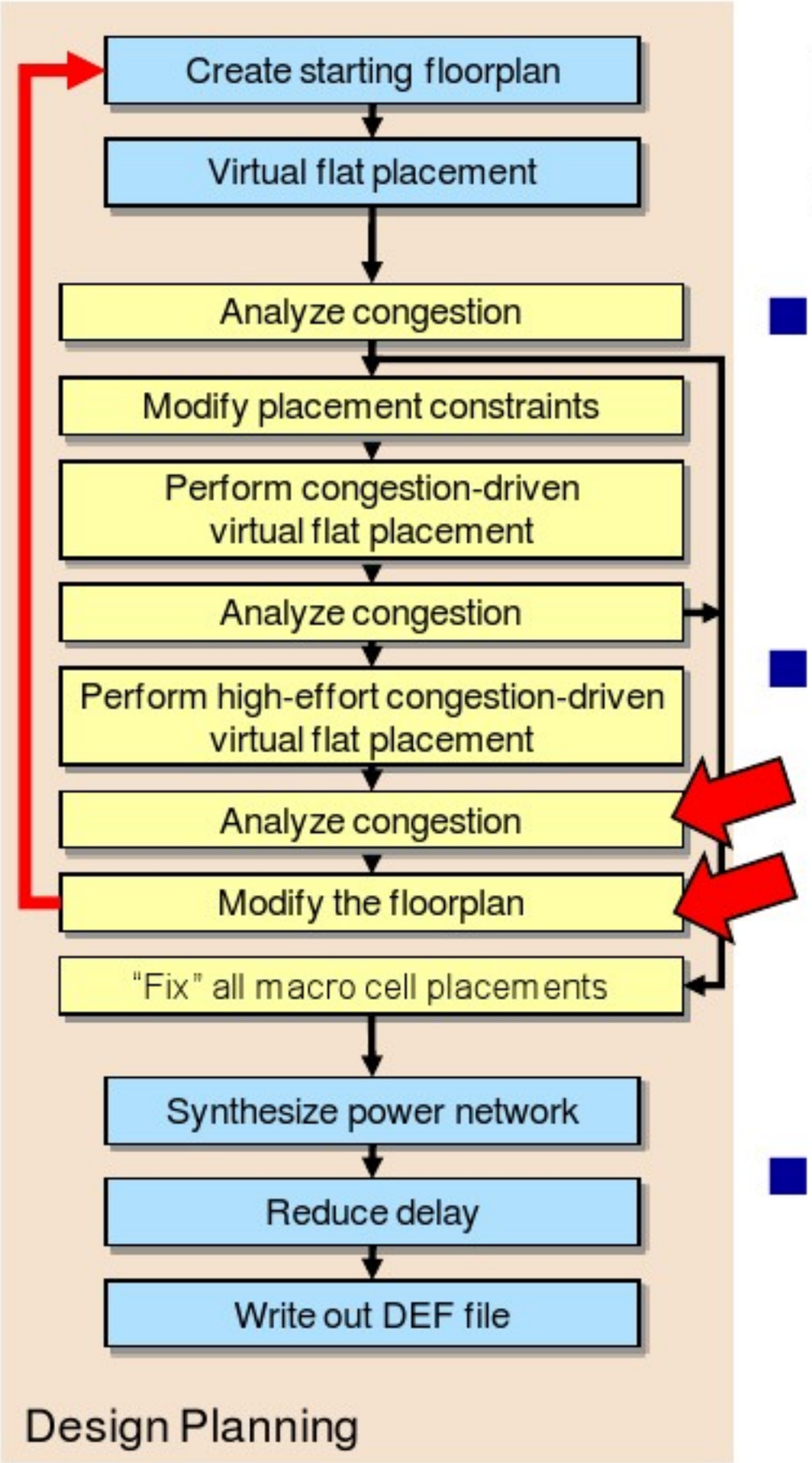

**If congestion is still not acceptable, perform *high-effort* congestion-driven placement:**

```
report_congestion -grc_based -by_layer \
                  -routing_stage global

set_fp_placement_strategy -congestion_effort high
create_fp_placement -timing -no_hierarchy_gravity \
                    -congestion
```

# Modify the Floorplan

Reduce congestion

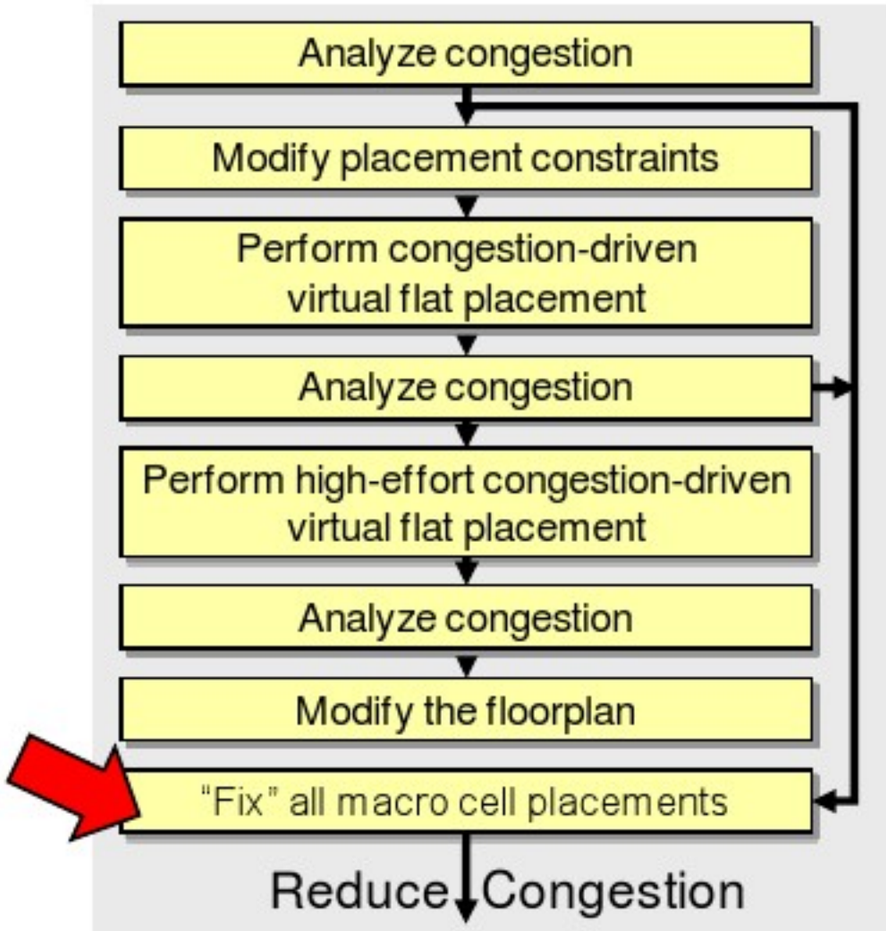



**If congestion is still a problem start over with a modified floorplan:**

- **Top-level pads or ports**
  - Spread, re-order or move to other sides
  - Change to a different metal layer
- **Core aspect ratio and size**
  - Make taller/wider to add more horizontal/vertical routing resources
  - Increase the core area to reduce utilization (cell density)
- **Modify power grid structure (if built)**
  - Use higher metal layers
  - Change width and/or spacing of straps

# "Fix" All Macro Cell Placement

Reduce congestion

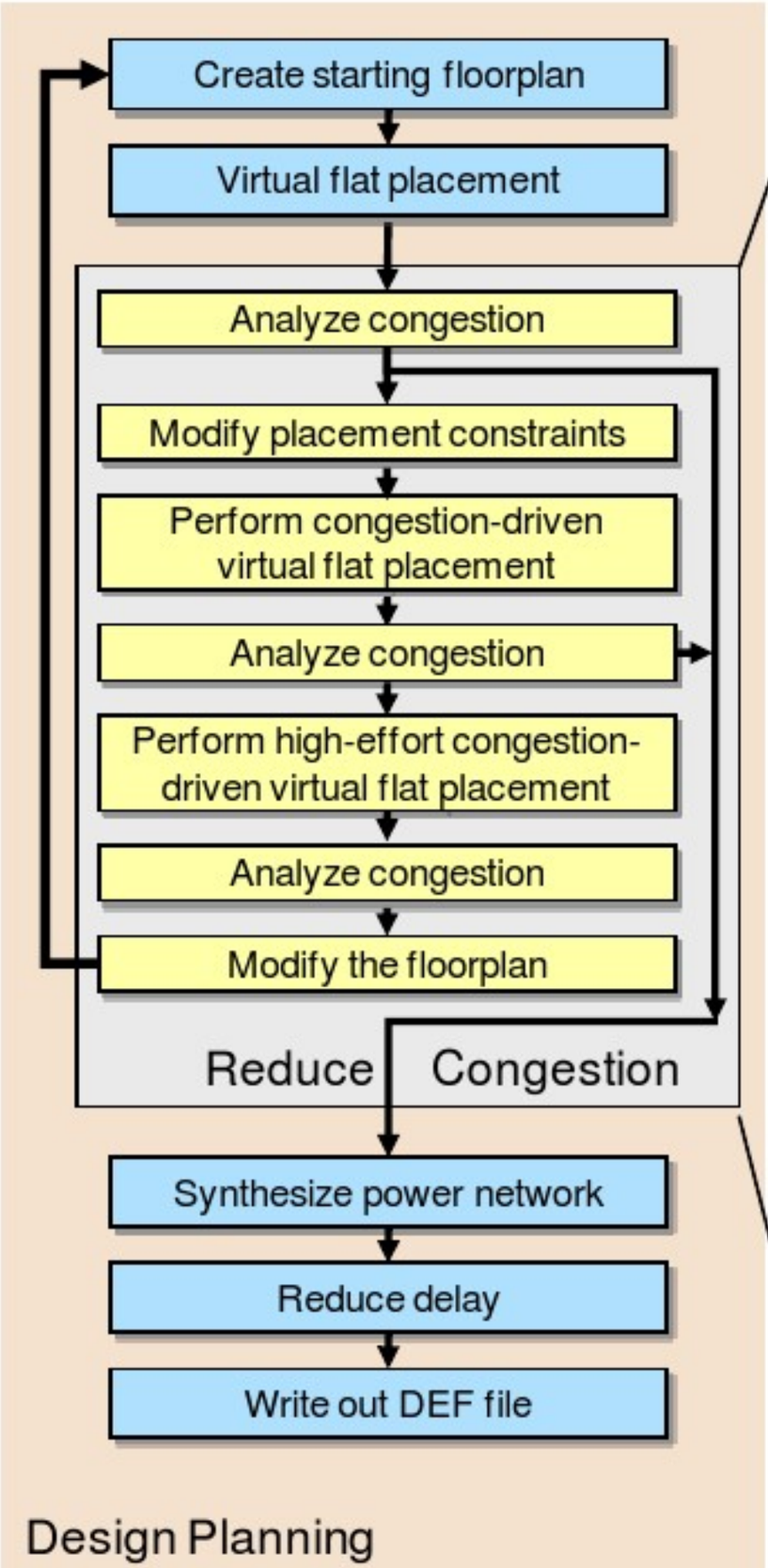



**After all congestion issues are resolved, "fix" the placement of all macro cells to prevent any subsequent placement steps from moving them**

```
set_dont_touch_placement [all_macro_cells]
```

# Summary: Reduce Congestion

Reduce congestion




```
# Analyze congestion:
report_congestion -grc_based -by_layer \
                  -routing_stage global
# Modify macro placement constraints:
set_fp_macro_options ...
set_fp_macro_array ...
set_fp_relative_location ...
# Modify existing std cell keepout constraints:
set_app_var physopt_hard_keepout_distance <#>
set_app_var placer_soft_keepout_channel_width <#>
set_keepout_margin ...
# Apply additional std cell placement constraints:
set_congestion_options ...
create_placement_blockage ...
# Modify fp placement strategy options:
set_fp_placement_strategy ...
# Perform congestion-driven placement:
create_fp_placement -timing -no_hier -congestion
# Analyze congestion, as above. If still congested
# enable high-effort congestion mode:
set_fp_placement_strategy -congestion_effort high
create_fp_placement -timing -no_hier -congestion
# Analyze congestion and if still a
# problem → re-floorplan.
# Fix the placement of all macro cells:
set_dont_touch_placement [all_macro_cells]
```

# Test For Understanding (1 of 2)

1. **Circle the correct statement(s) about what `create_fp_placement` does by default:**
   a. Optimizes logic (cell sizing, buffering) to improve timing
   b. Legally places std cells and non-fixed macros to minimize wire length
   c. Optimizes placement to improve timing
   d. Clumps cells from the same logical hierarchy together

2. **Comparing "9/1" and "20/12" in a congestion map:**
   a. They are equally bad
   b. 20/12 is worse because both numbers are much larger
   c. 9/1 is worse the ratio is much larger
   d. Can not tell without also comparing their "heat" colors

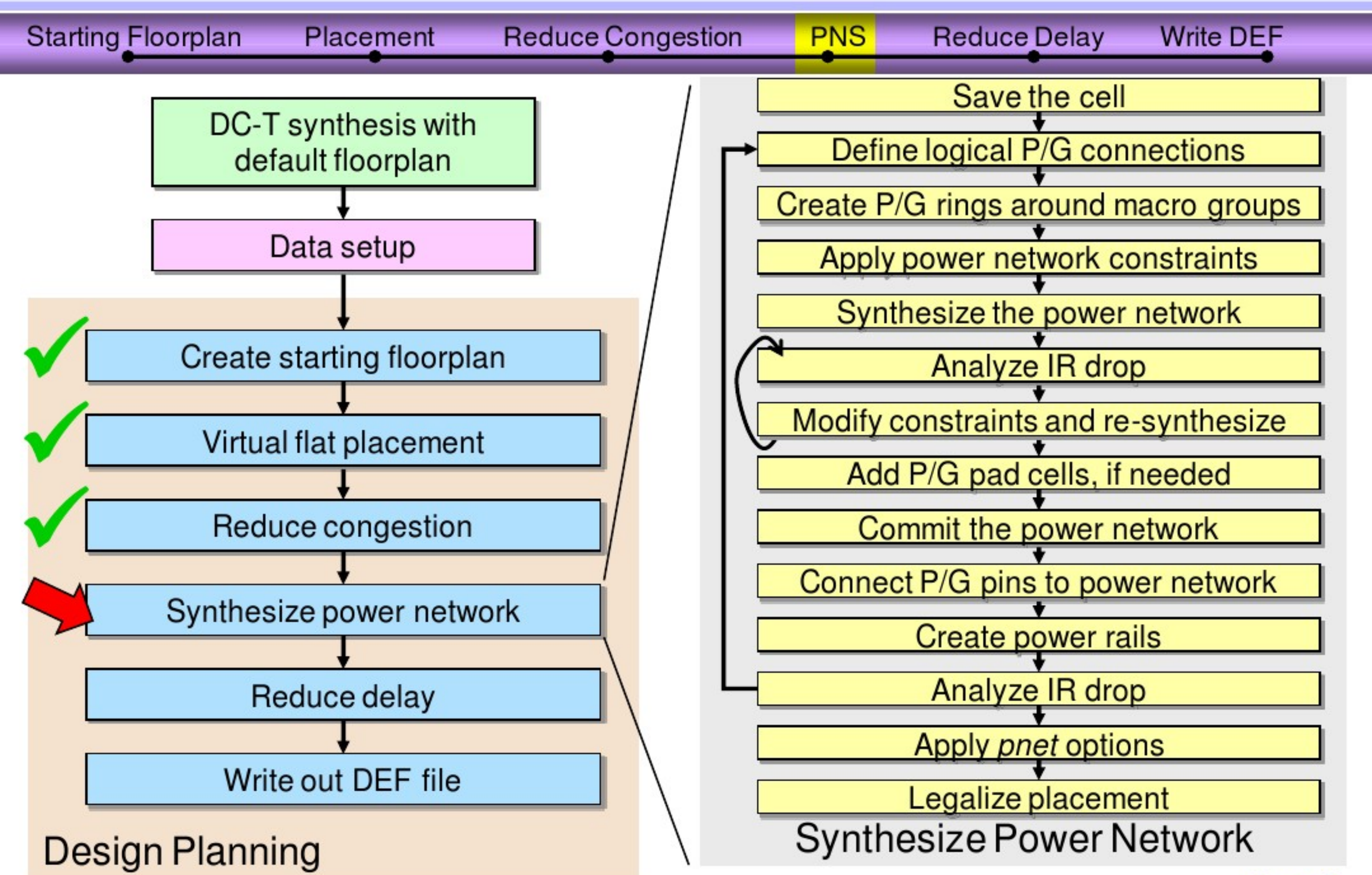
3. **What are some common areas for congestion to occur?**

# Test For Understanding (2 of 2)

4. **Macro placement can be controlled by performing the following prior to virtual flat placement:**
   a. Manually placing and "fixing" macros
   b. Specifying general "placement strategy parameters" (e.g. auto-grouping, macros on edge, macro orientation)
   c. Applying "macro placement constraints" (options, arrays, relative location)
   d. All of the above

5. **If you can not achieve acceptable congestion after modifying placement constraints and parameters, as well as performing high effort congestion-driven virtual flat placement, what should you do?**

# Synthesize the Power Network (PNS)



# Power Network Synthesis (PNS)



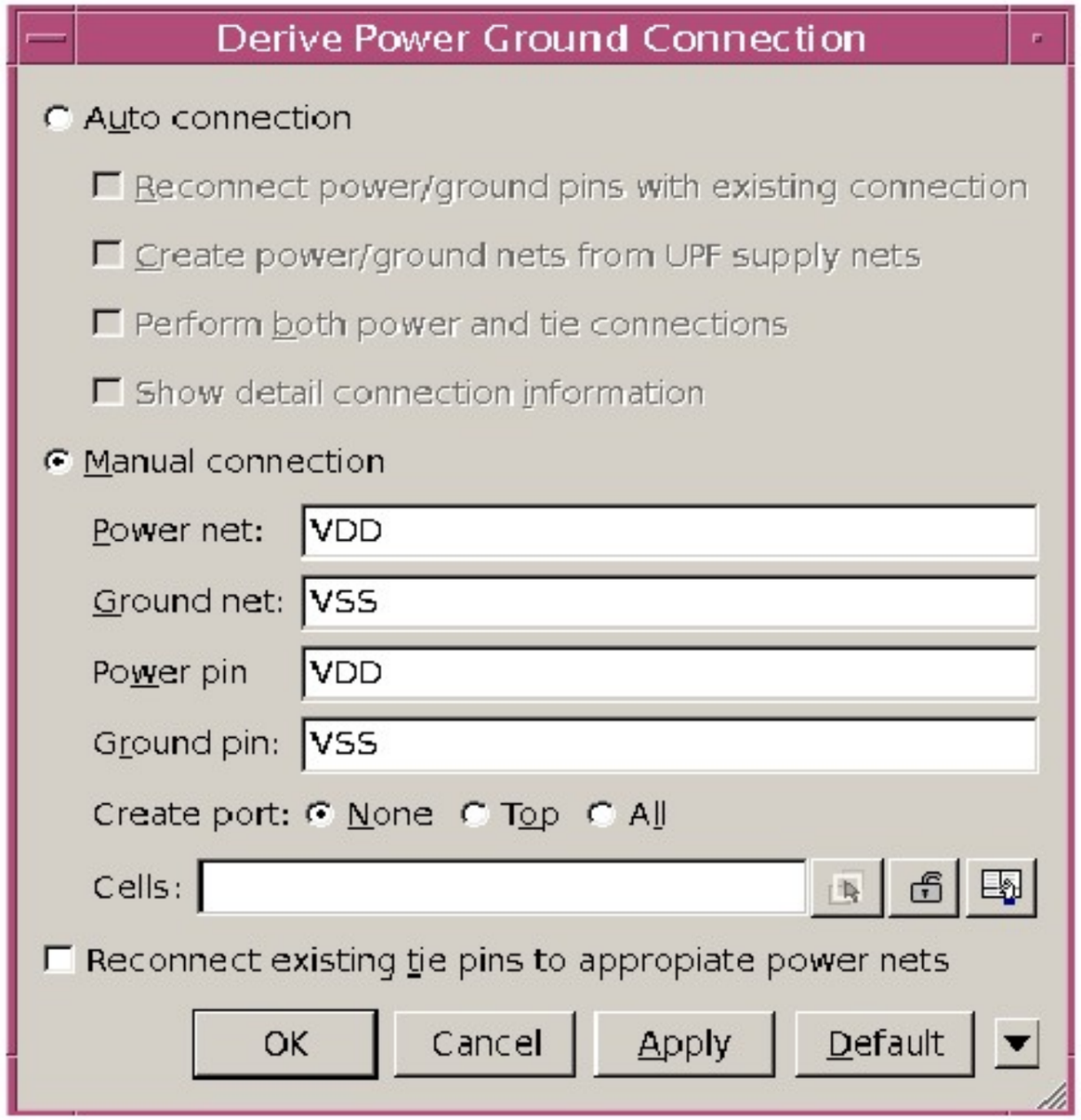
**Constraint-based PNS automates a typically tedious and iterative manual task**

- **Creates DRC/ERC-clean macro power rings and power straps**
- **Calculates the width and the number of power straps to meet power and max IR-drop settings**
- **Displays an IR-drop "heat map" for easy analysis**
- **Allows repeated "what-if" analyses before "committing" the power network structure**
- **After "committing" there is no easy way to "undo" the power network → Save the cell before PNS**

```
save_mw_cel -as DESIGN_pre_pns
```

# Define Logical Power/Ground Connections

PNS




- **Make sure that logical P/G connections are defined prior to PNS**
  - This may have already been done during *Data Setup* and/or creation of pad P/G rings
- **One command for each power/ground net**

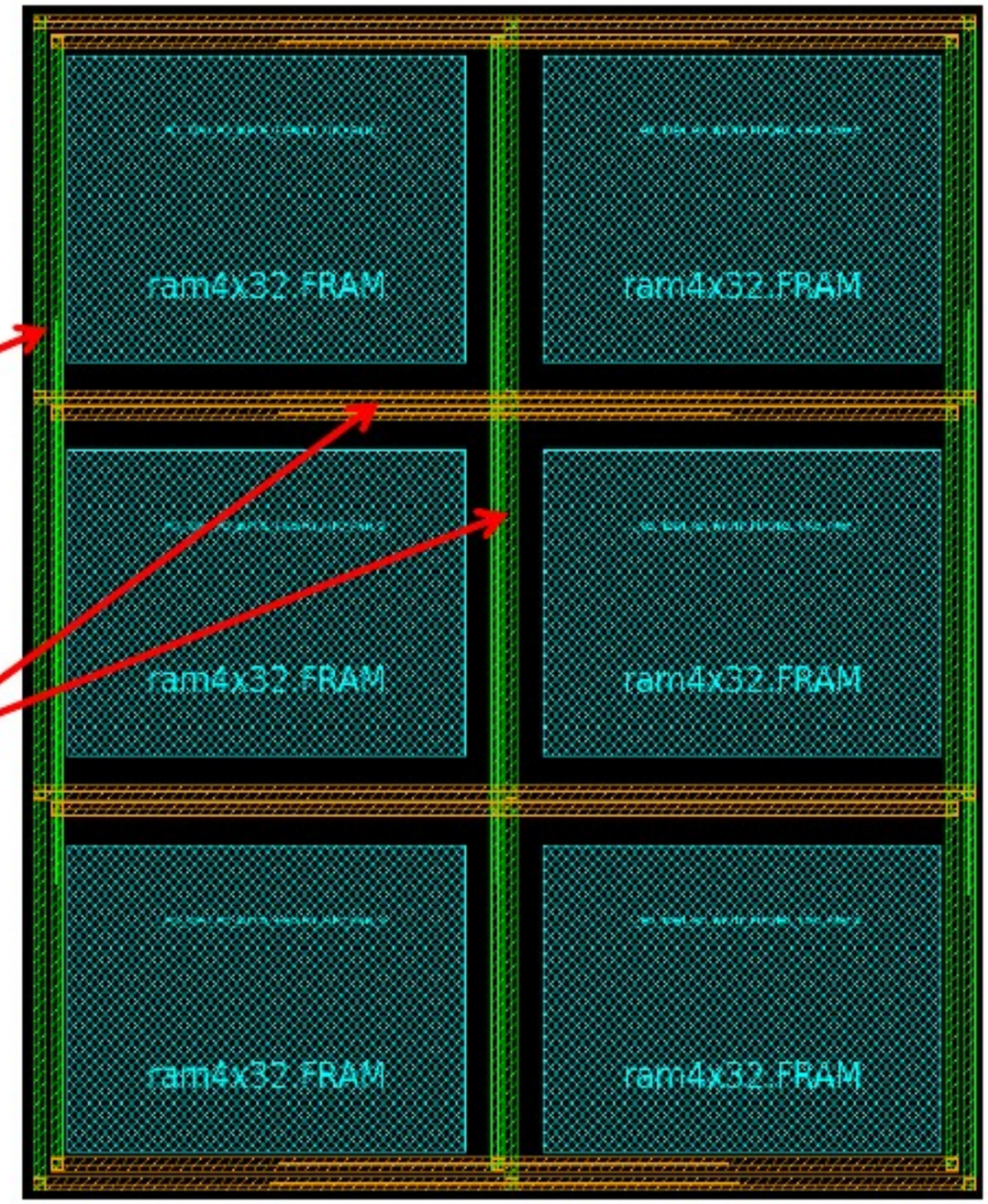
```
derive_pg_connection -power_net VDD  -power_pin VDD  -ground_net VSS  -ground_pin VSS
derive_pg_connection -power_net VDDO -power_pin VDDO -ground_net VSSO -ground_pin VSSO
derive_pg_connection -power_net VDDQ -power_pin VDDQ -ground_net VSSQ -ground_pin VSSQ
derive_pg_connection -power_net PWR  -ground_net GND -tie
```

# P/G Rings Around Groups of Macros

PNS

**If your design contains groups of macros placed in close proximity, or in a specific configuration:**

- **May prefer to "manually" create a P/G ring around each group**
  - Alternative is to use PNS to create rings around individual macros, or, no rings at all
  - Can include "straps" inside the group rings
- **Need to create 'group' rings prior to PNS**
  - Individual P/G macro rings will be created during PNS



# Create P/G Rings Around Groups of Macros

PNS

**Must create each P/G ring one at a time :**

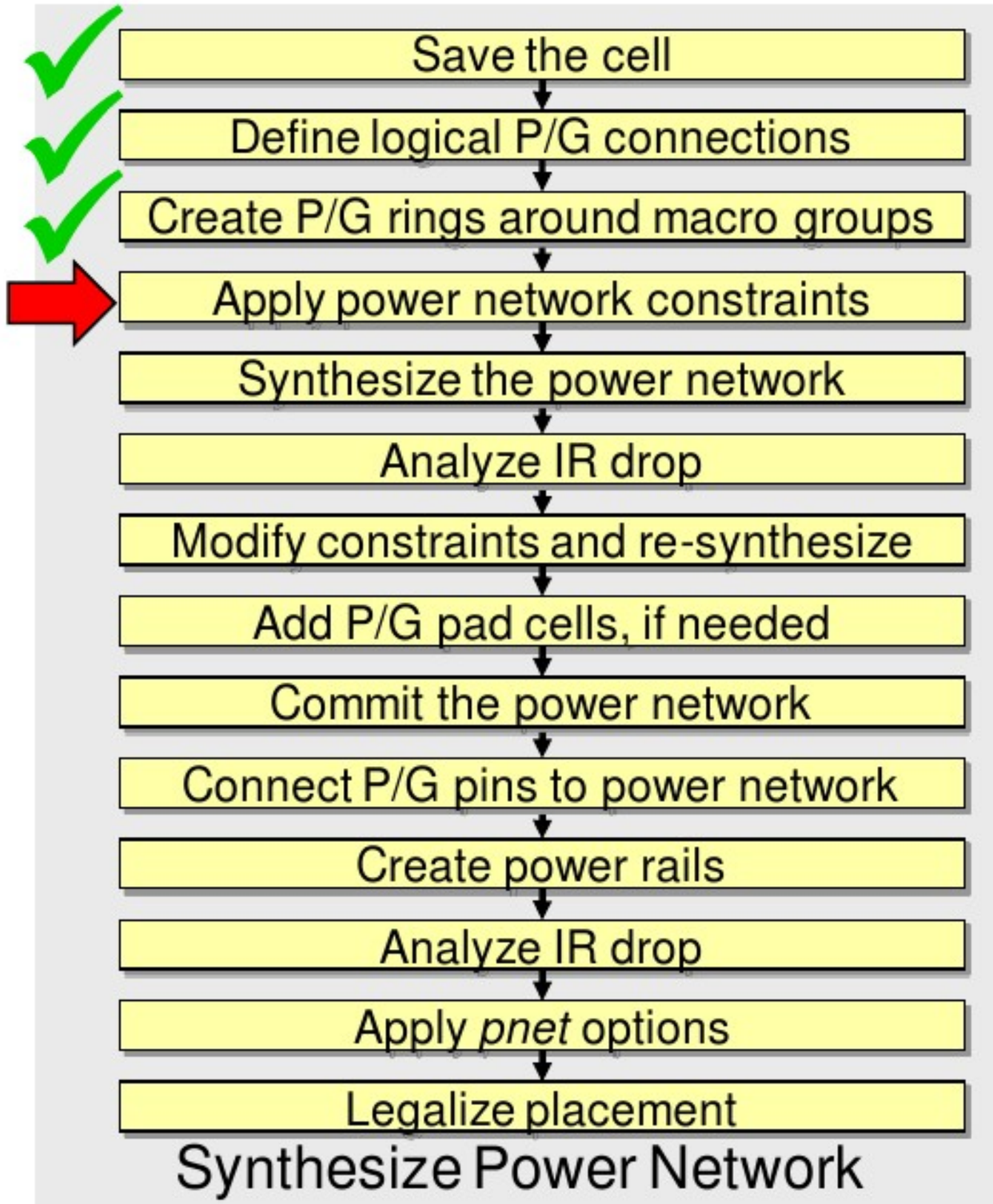
- **Define a "region" that encompasses a group of macros**
- **Define the group ring and strap layers, widths, offsets …**
- **"Commit" to create the actual routes**
- **Repeat for each group and remove the last region when done**

```
# Create ring #1
set_fp_rail_region_constraints  -polygon {{366 1304} {366 1680} {663 1680} {663 1304}}
create_fp_group_block_ring      -nets {VDD VSS} \
 -horizontal_ring_layer METAL5  -horizontal_ring_offset 0.78 -horizontal_ring_width 3 \
 -vertical_ring_layer METAL4    -vertical_ring_offset 0.73   -vertical_ring_width 3 \
 -horizontal_strap_layer METAL5 -horizontal_strap_width 3 \
 -vertical_strap_layer METAL4   -vertical_strap_width 3
commit_fp_group_block_ring

# Create ring #2
set_fp_rail_region_constraints  -polygon {{1426 726} {1426 1053} {1807 1053} {1807 726}}
...
set_fp_rail_region_constraints -remove
```

# Ready for PNS






**After any "manual" P/G pre-routes have been created, you are ready for PNS**

# Apply Power Network Constraints

PNS

```
set_fp_rail_constraints ...
```

Preroute Placement Partition Clock Rou

- Connect Power Ground Net...
- Disconnect Power Ground Net...
- Set Preroute DRC Options...
- Set Preroute Advanced Via Rule...
- Create Virtual Power Pads...
- Show Virtual Power Pads
- Power Network Constraints ▸
  - Layers Constraints...
  - Ring and Straps Constraints...
  - Global Constraints...
  - Region Constraints...
  - Block Rings Constraints...
  - Power Pads Constraints...
- Voltage Power Network Constraints ▸
- Synthesize Power Network...
- Analyze Power Network...
- Power Network Voltage Drop Map
- Power Network Resistance Map
- Power Network Electromigration Map

**Layers Power Network Constraints**

Layer: METAL4

Direction: (•) Vertical ( ) Horizontal

Density: (•) By strap number: Max: 128 Min: 16
( ) By pitch: Max: Min:

Width: Max: Min: 0.440

PG spacing: (•) Minimum ( ) Interleaving ( ) Microns:

**Rings and Straps Power Network Constraints**

[x] Generate rings

Power Ground nets: VDD VSS

Horizontal layers: METAL (14 / METAL2 (1 / METAL3 (2 / METAL4 (2

Vertical layers: METAL (14 / METAL2 (1 / METAL3 (2 / METAL4 (2

[ ] Ring width: ( ) Fixed: ( ) Variable: Min: Max:

Ring spacing: Ring offset to ID or PNS region:

Extend strap: (•) Core ring ( ) Top boundary ( ) Pad ring

Default

**Global Power Network Constraints**

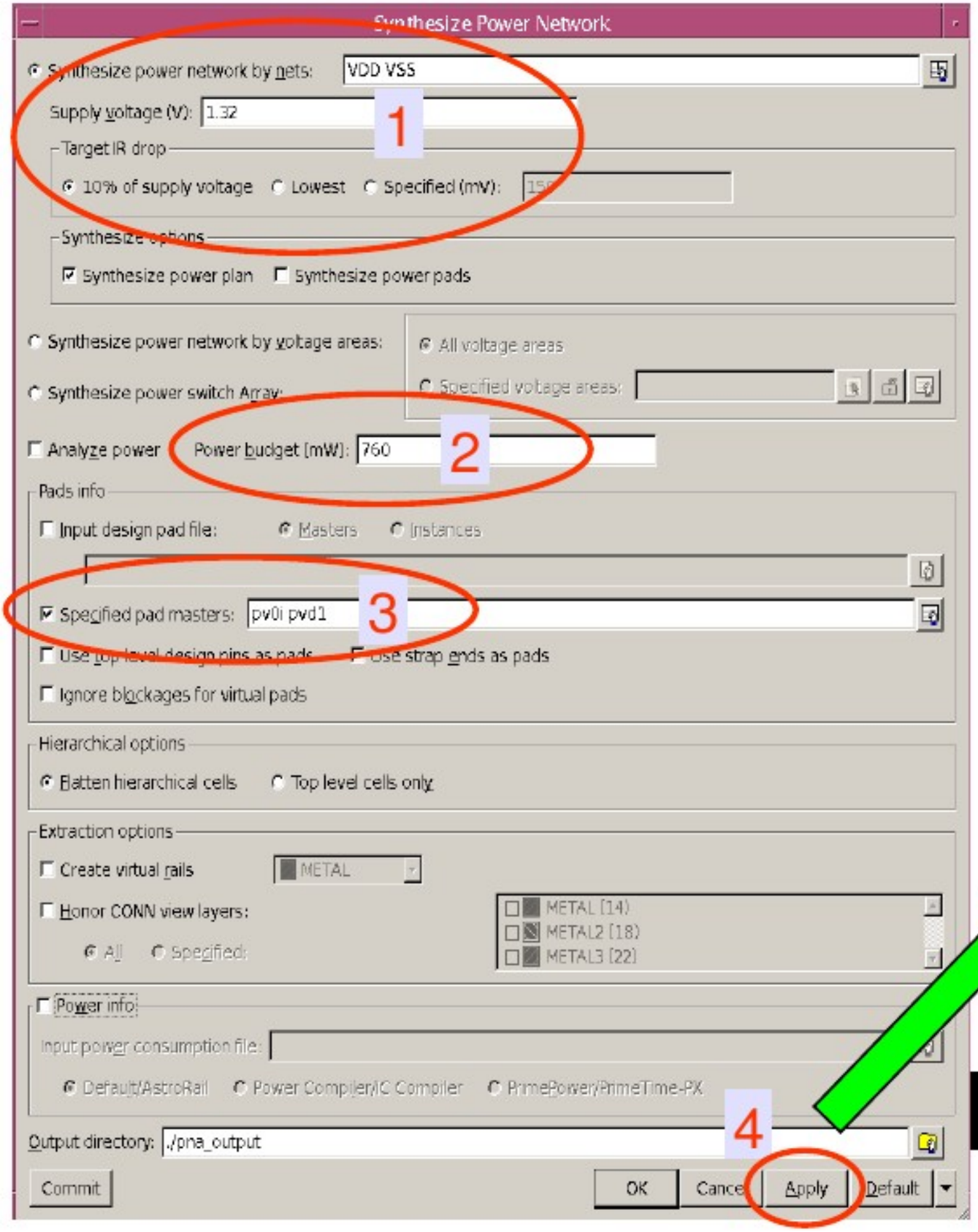
- [x] Remove floating segments
- [x] Force same width sizing
- [x] Generate stacked vias
- [ ] Optimize tracks usage
- [ ] Ignore all the blockages
- [ ] No routing over plan groups
- [ ] No routing over hard macros
- [ ] No routing over soft macros
- [ ] Keep rings outside of core area

Set Default

Close

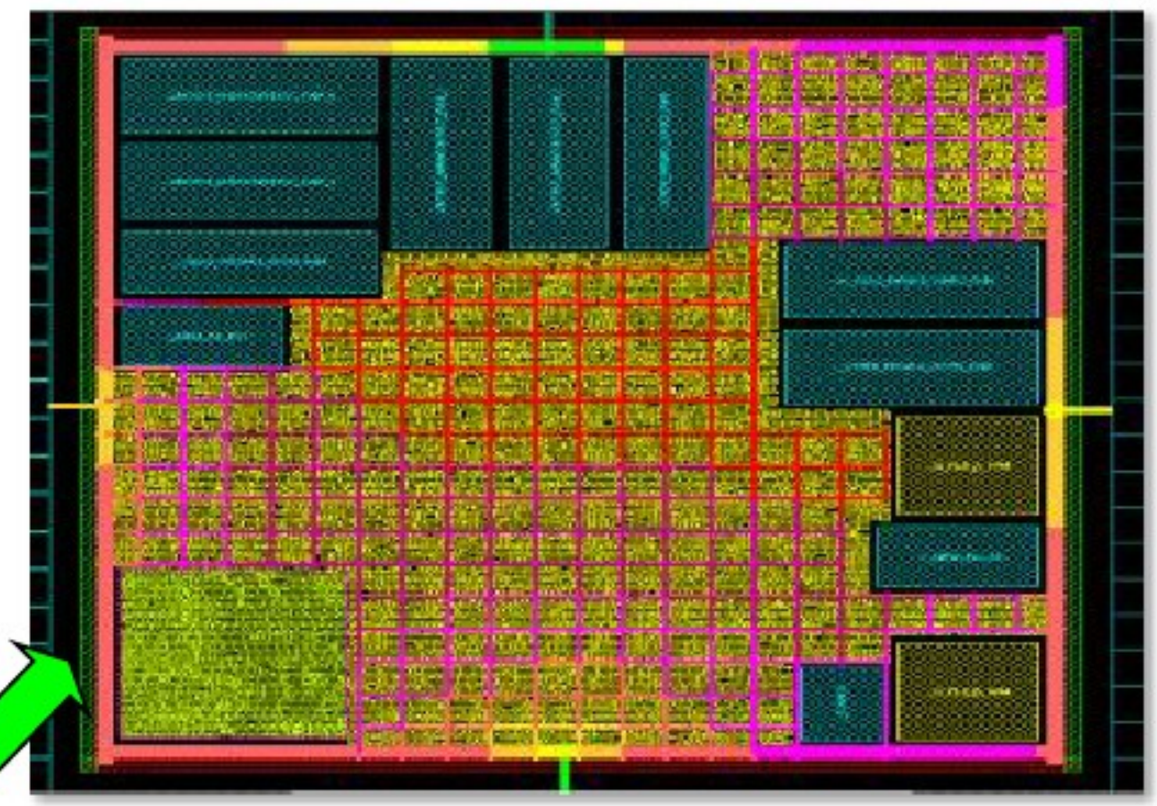
# Synthesize and Analyze the Power Network

PNS




PNS calculates the required number of straps based on provided constraints. Heat map aids the analysis

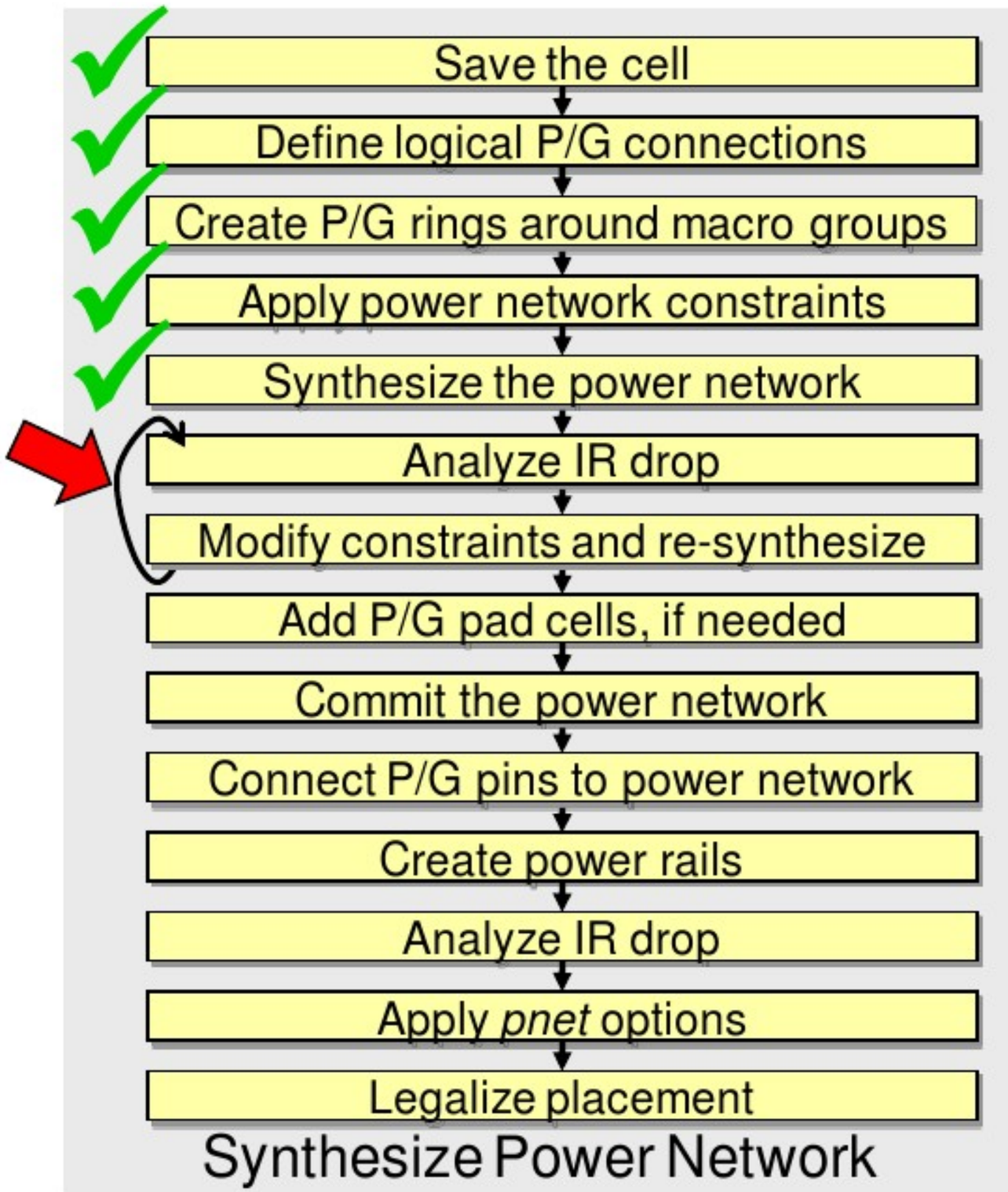
Preview of power plan with IR-drop heat map – no actual routes created yet



```
synthesize_fp_rail ...
```

# Modify Constraints and Re-synthesize

PNS

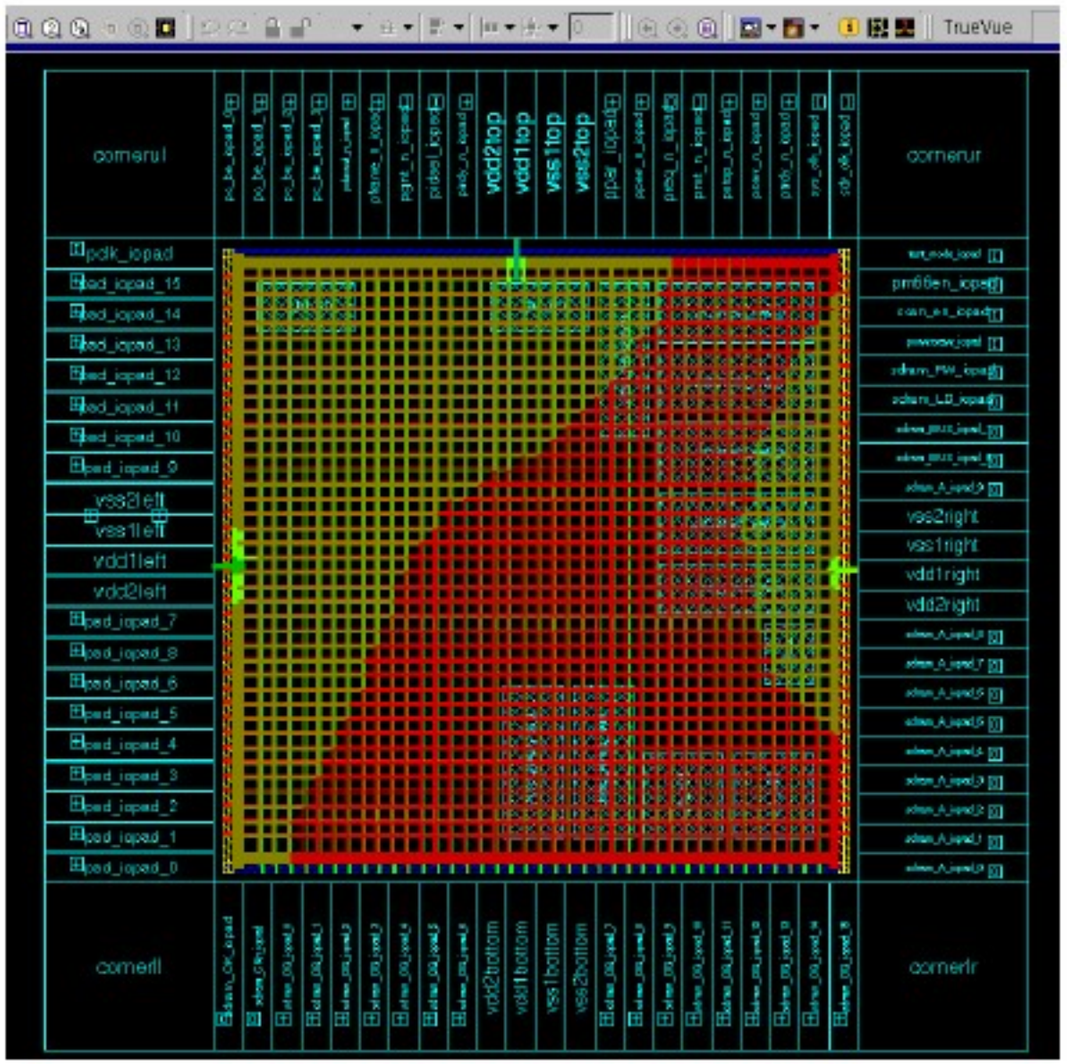


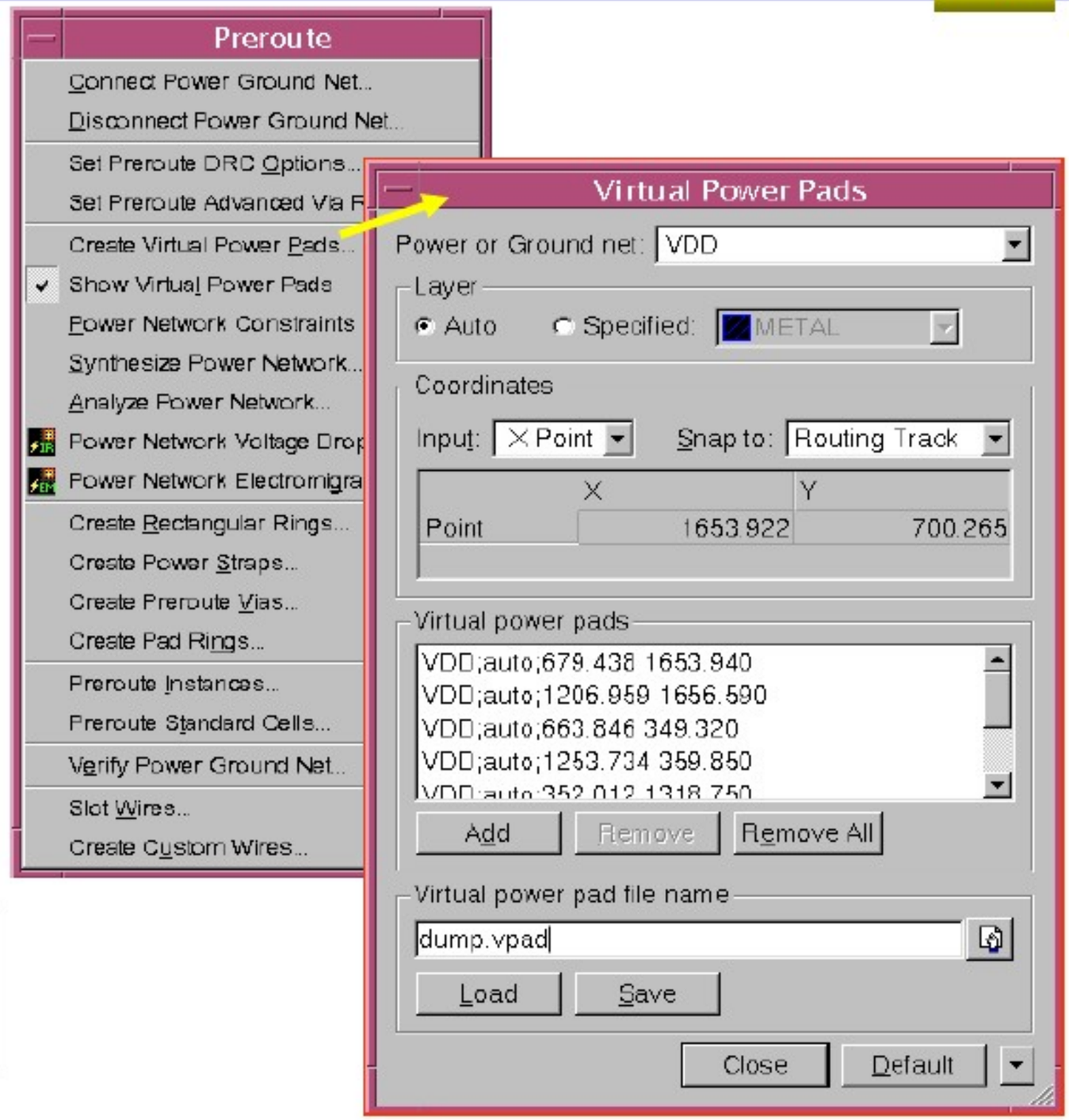

**If the maximum IR drop is not acceptable**

- Modify power network constraints
- Re-synthesize
- Repeat as needed

# Create Virtual Power/Ground Pads if Needed

PNS



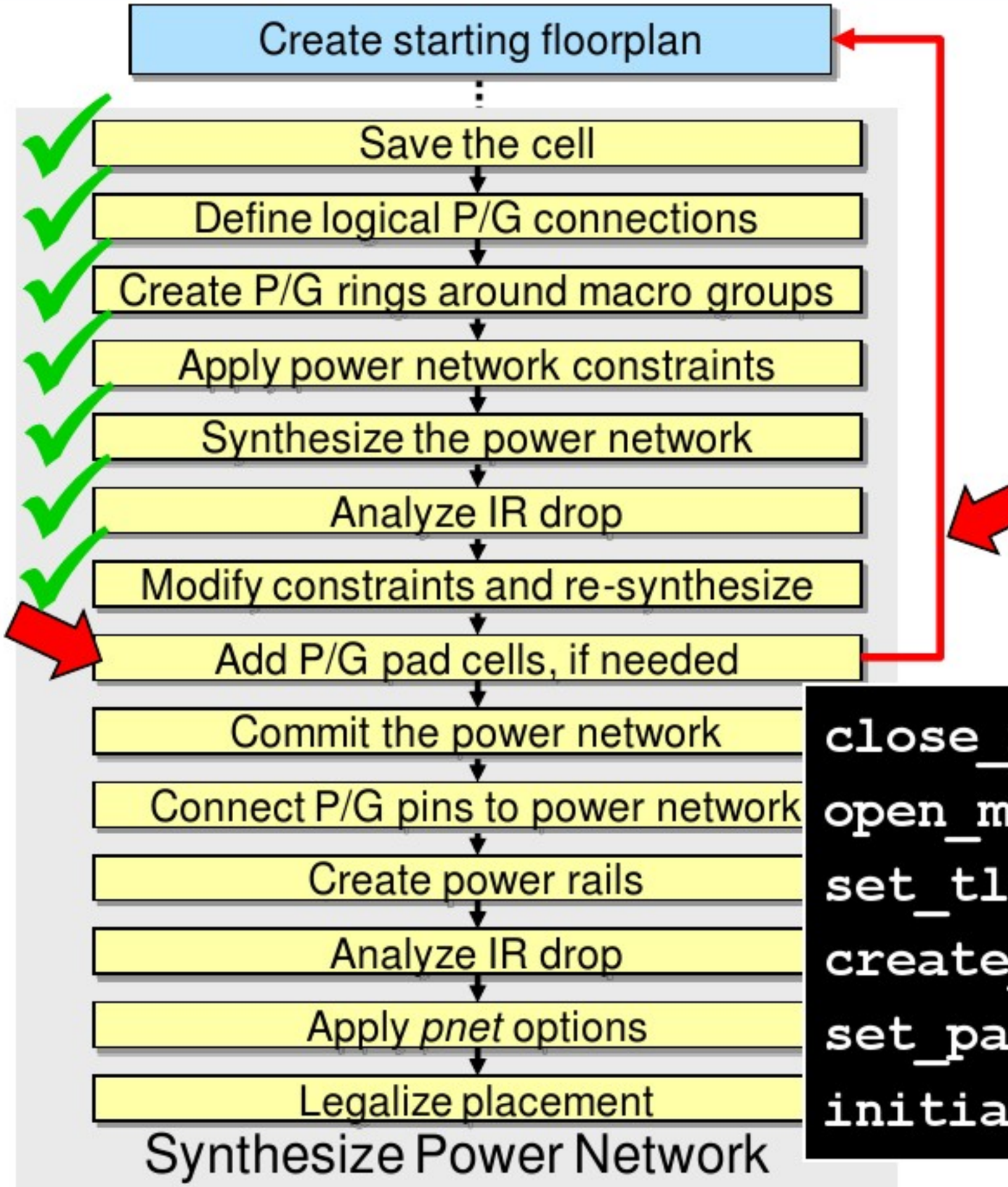




**Above example indicates that additional P/G pads may help**

```
create_fp_virtual_pad ...
# Then "Apply" PNS to update heat map
```

# Create New Floorplan with Added P/G Pads

PNS

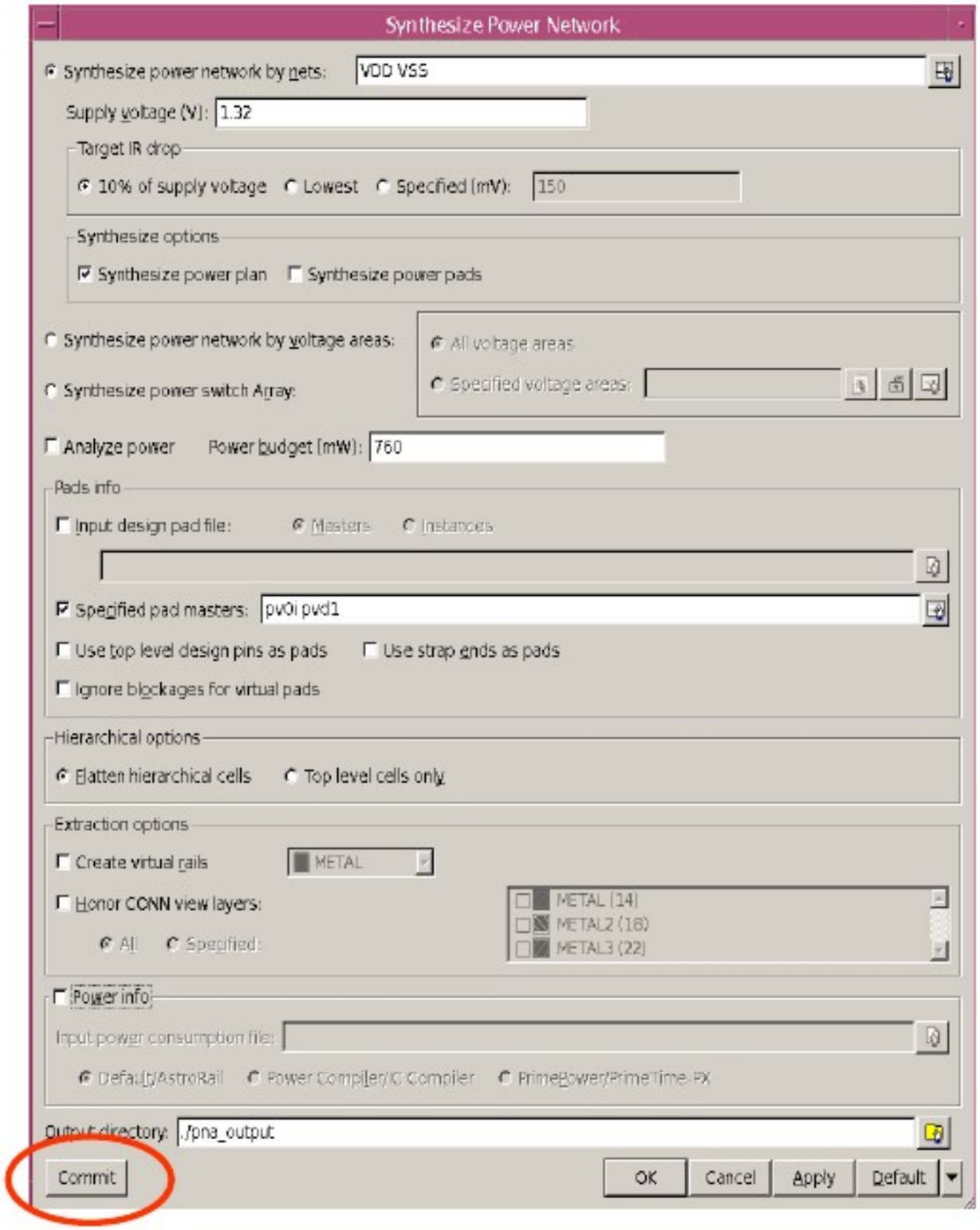



**If additional P/G pads are required you will need to create a new initial floorplan with the improved P/G pad cell constraints**

```
close_mw_cel
open_mw_cel DESIGN_data_setup
set_tlu_plus_files ...
create_cell ...
set_pad_physical_constraints ..
initialize_floorplan ...
```

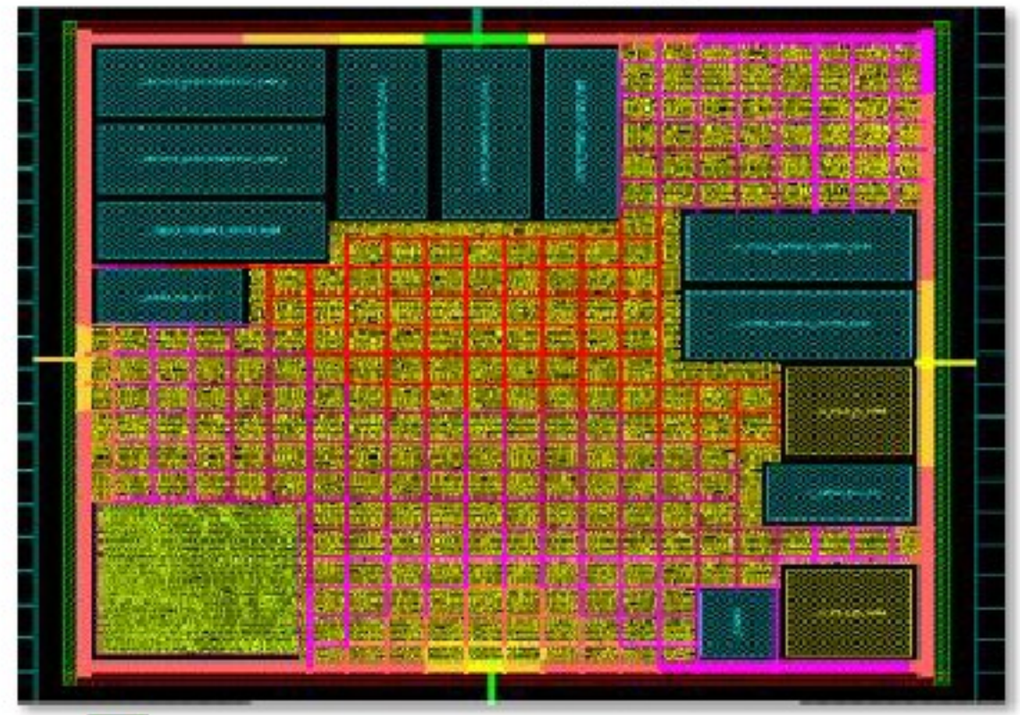
# Commit the Power Network

PNS

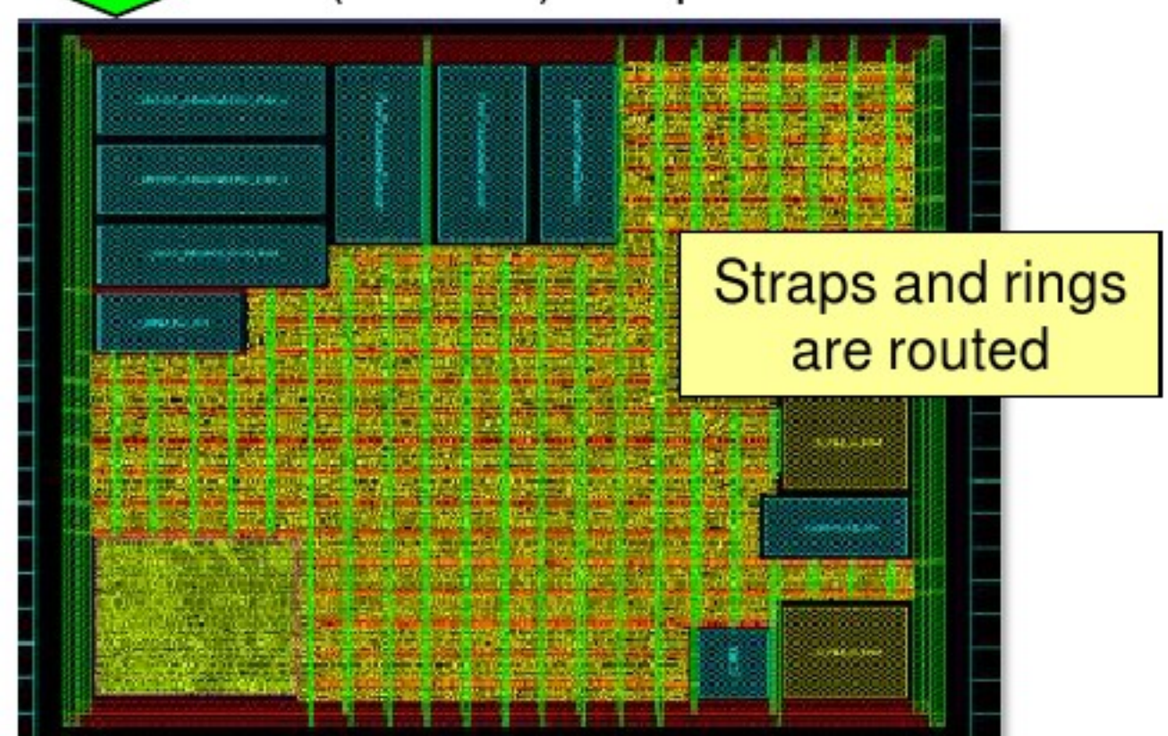



**commit_fp_rail**

Preview (Apply)

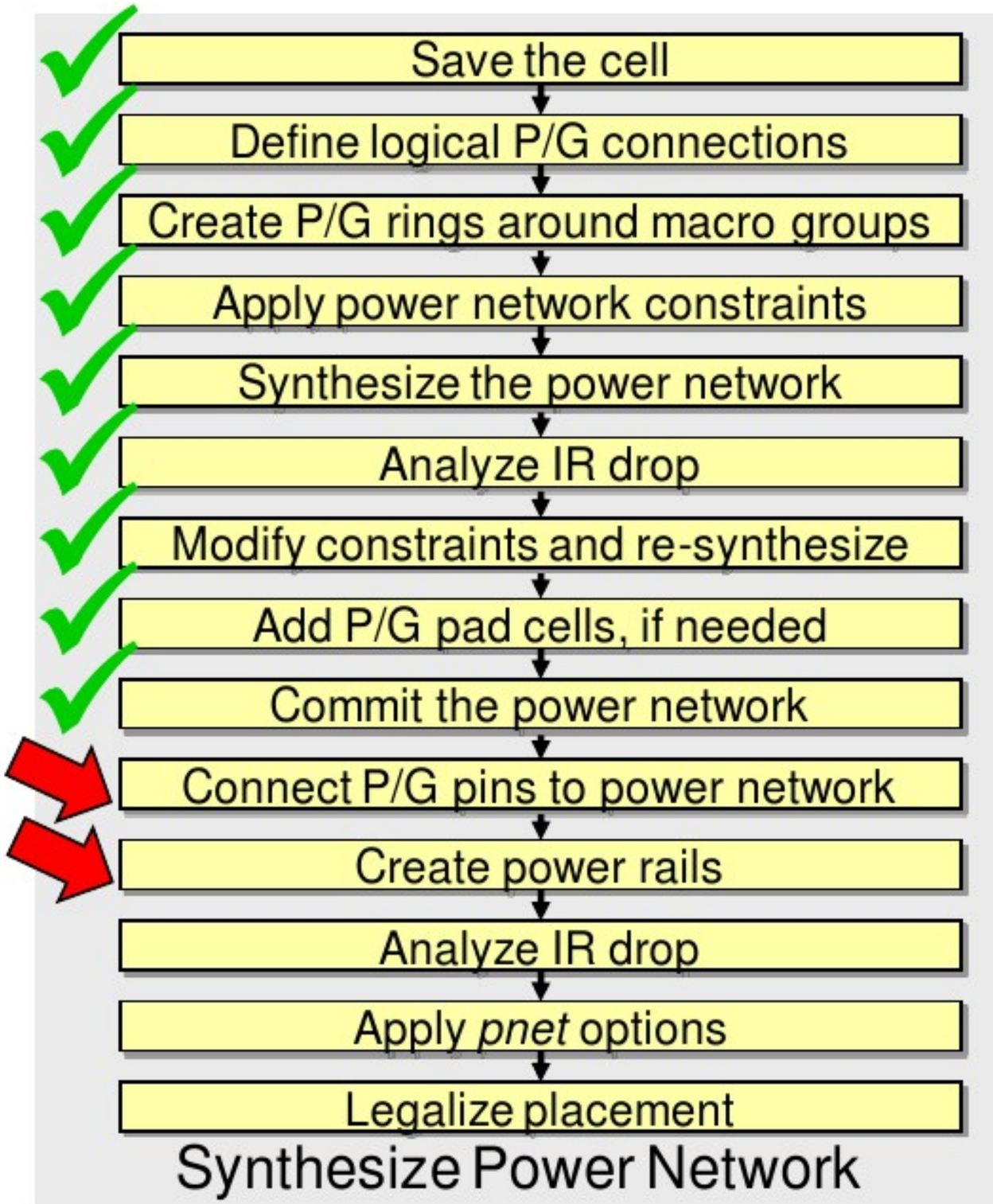


Build (Commit) the power network



# Connect P/G Pins and Create Power Rails

PNS

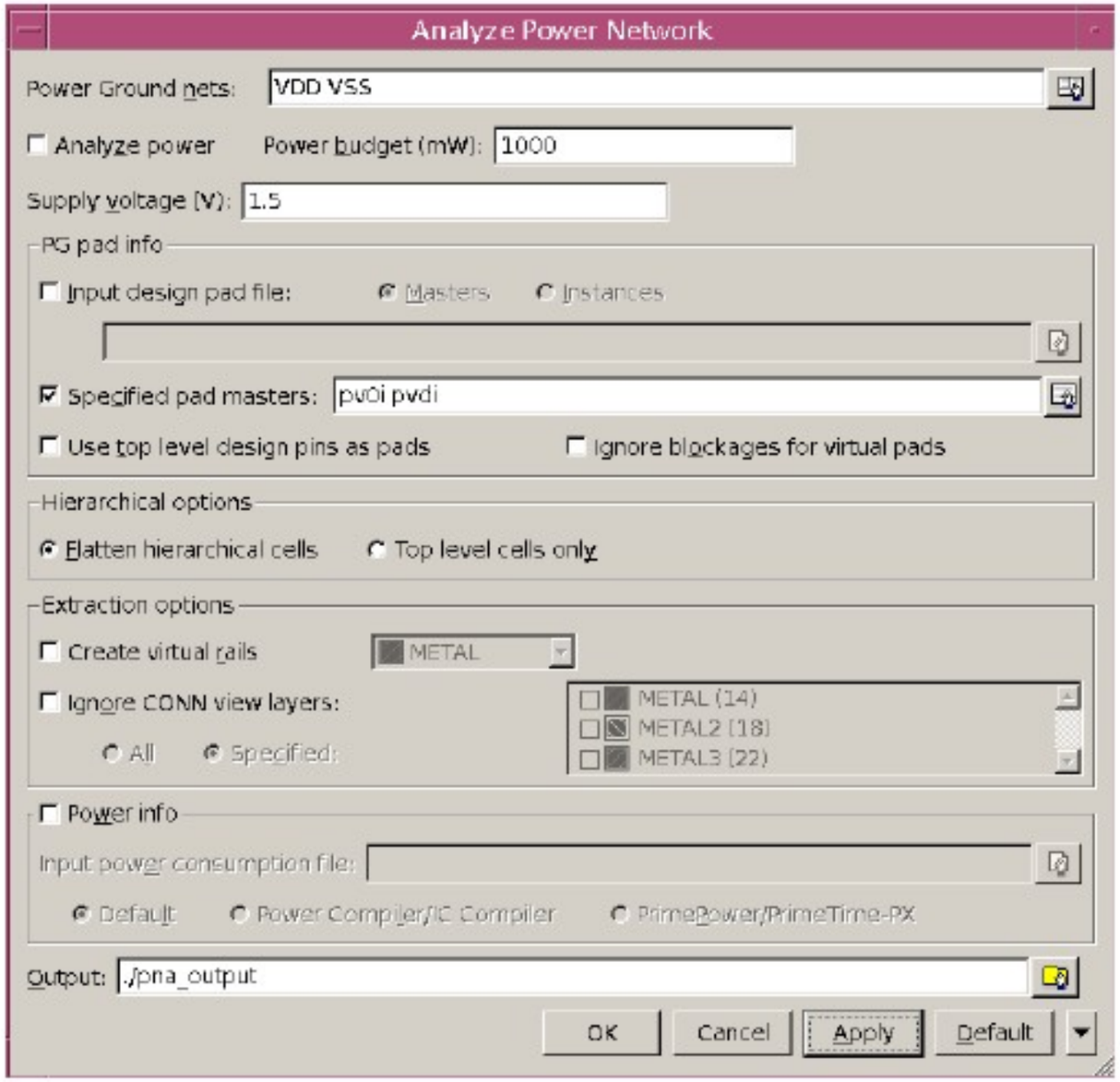



- **Connect P/G macro pins and pad pins to core rings and straps**
- **Create power rails along the standard cell placement rows**

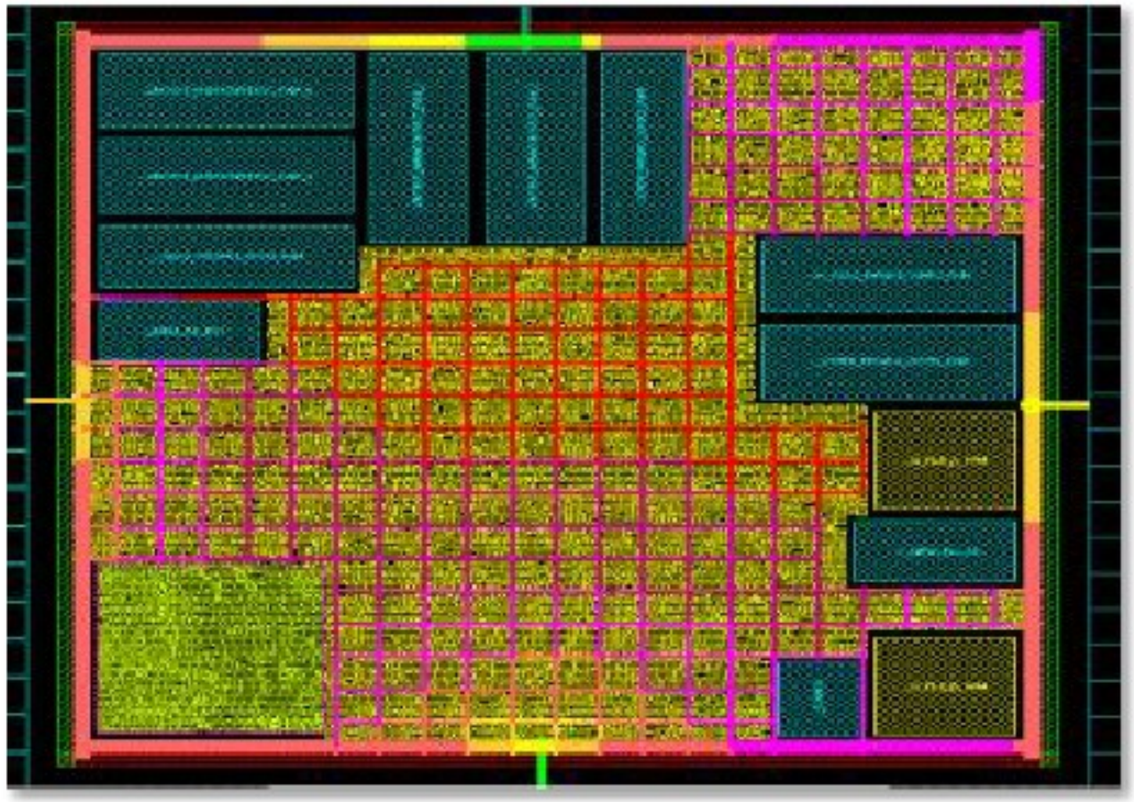
```
preroute_instances
preroute_standard_cells \
    -fill_empty_rows      \
    -remove_floating_pieces
```

# Analyze the Power Network

PNS




After "commit" and after P/G pins and rails have been pre-routed → Analyze the power network for a more accurate IR-drop map

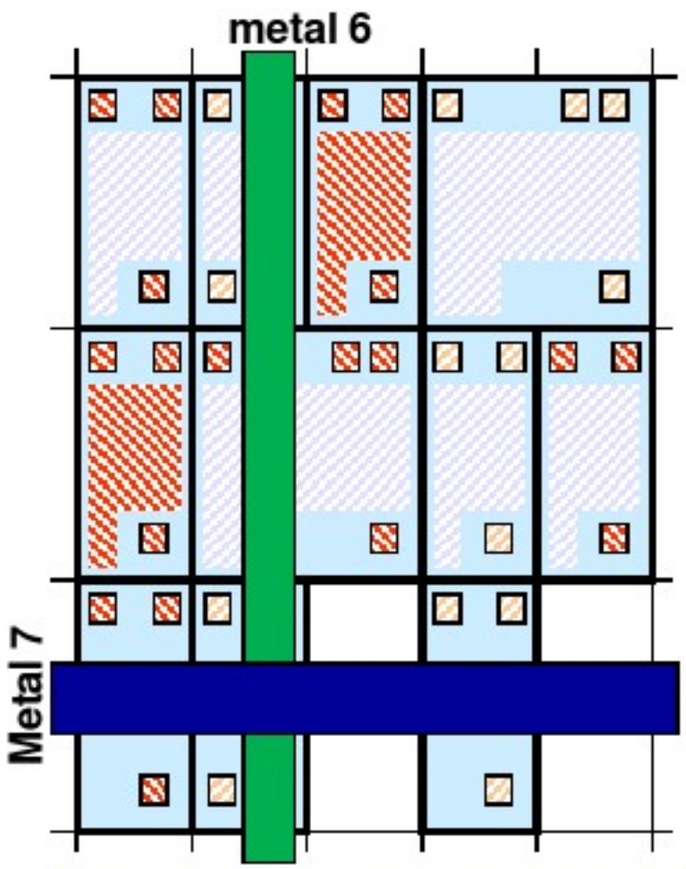


```
analyze_fp_rail ...
# If IR-drop is not acceptable:
close_mw_cel
open_mw_cel DESIGN_pre_pns
# Begin a new PNS flow
```

# Apply Power Net Placement Blockages

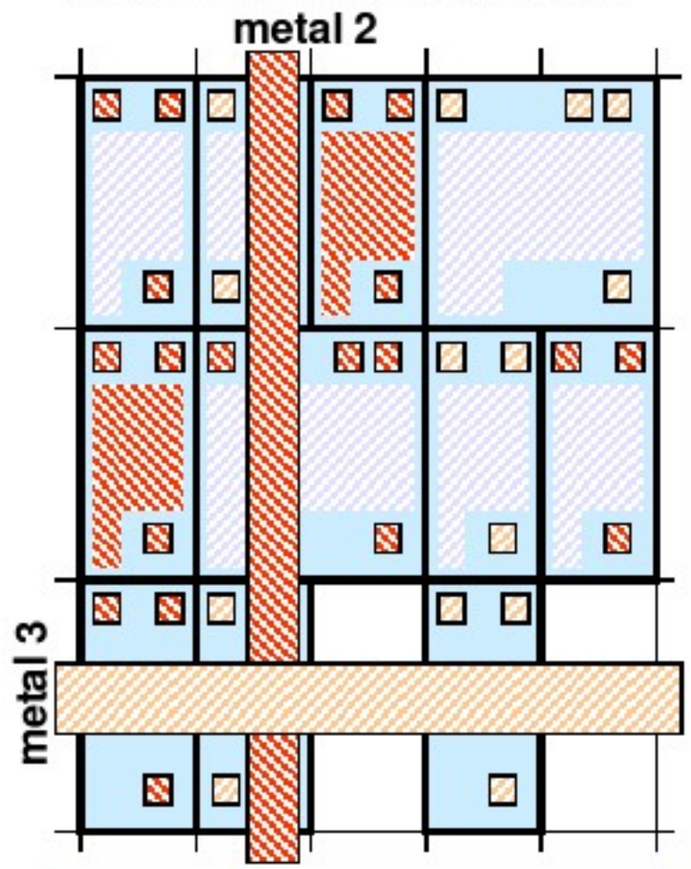
PNS

## No *pnet* blockage



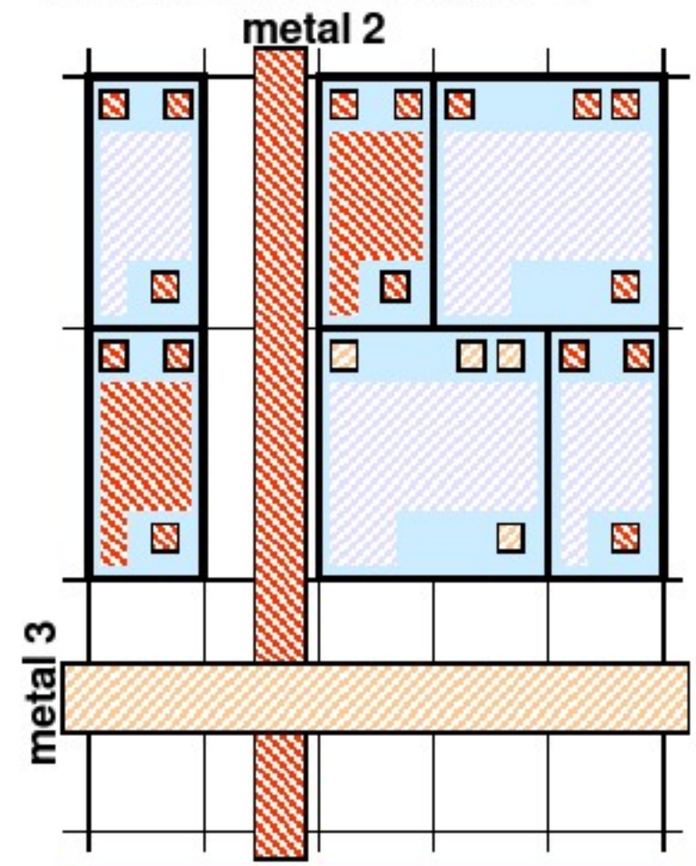

All cells allowed under P/G straps

## Partial *pnet* blockage on metal 2 and 3




Cells allowed under P/G nets, with min pin-to-strap spacing

## Complete *pnet* blockage on metal 2 and 3




No cells allowed under P/G straps

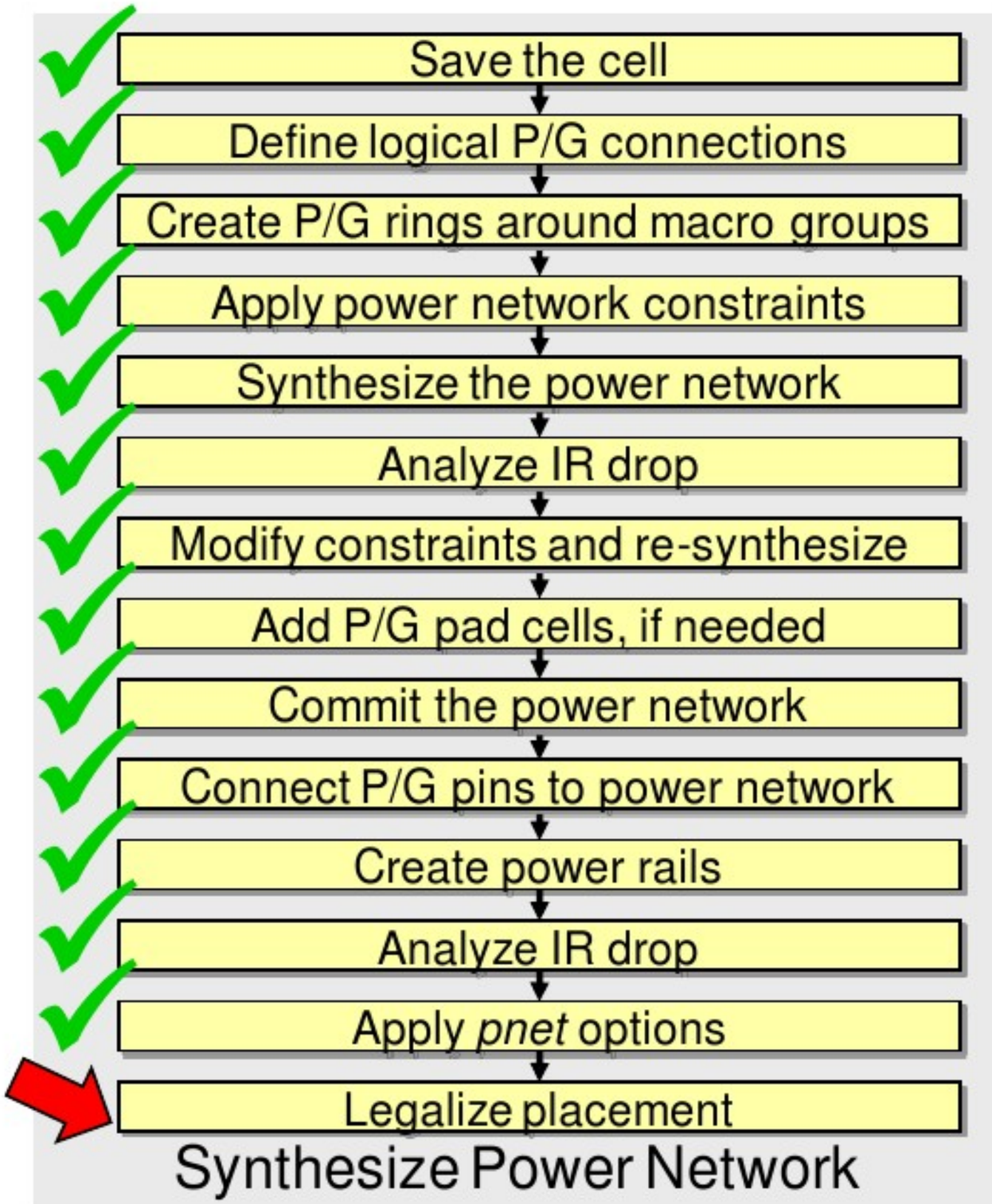
More uniform cell density → Less overall congestion

Less congestion along P/G nets

```
set_pnet_options -partial {metal2 metal3}
set_pnet_options -complete {metal2 metal3}
```

# Perform Placement Legalization

PNS

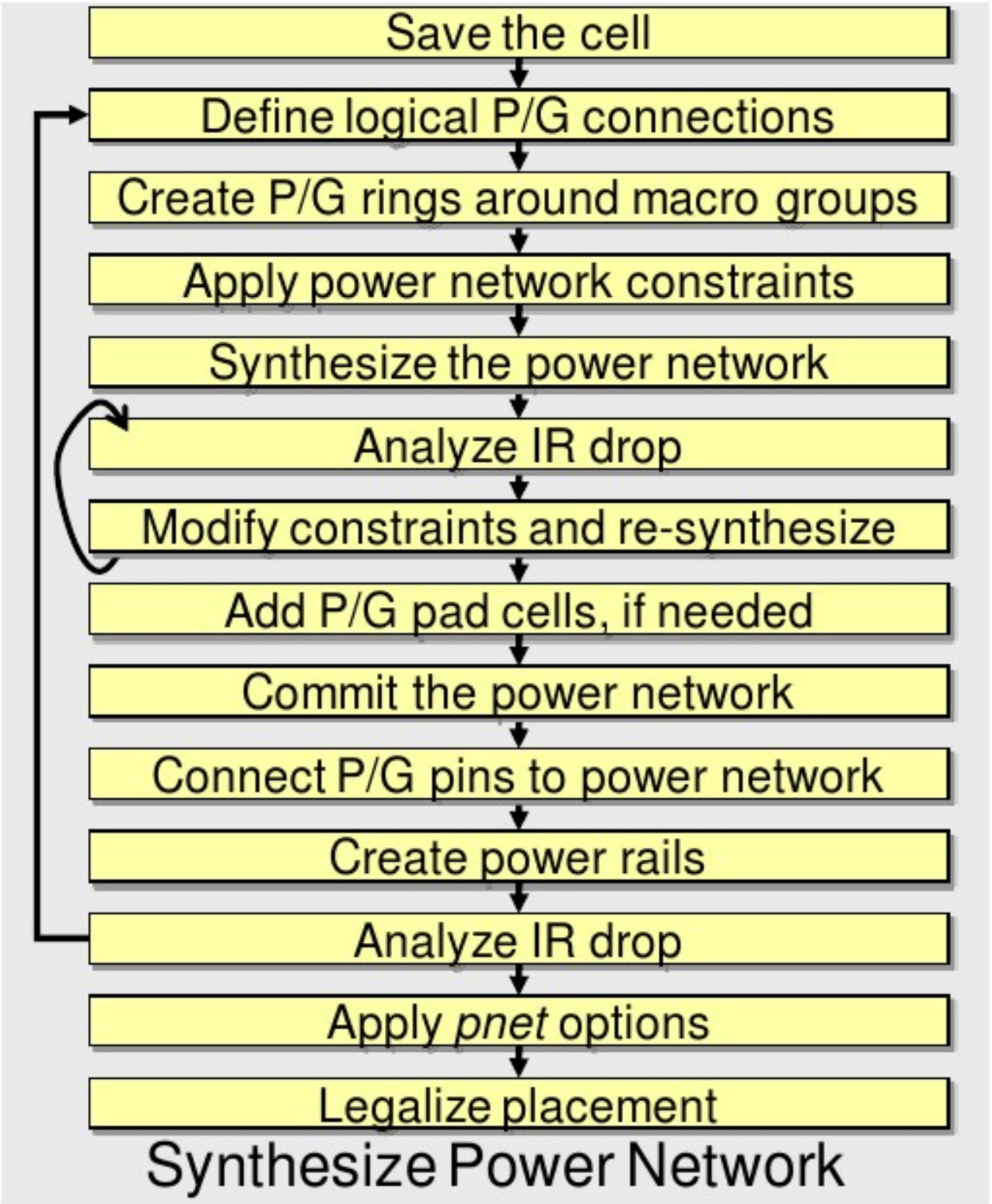



**If you have applied *partial* or *complete* power net placement blockages you will need to legalize placement to move violating standard cells away from the power straps**

```
legalize_fp_placement
```

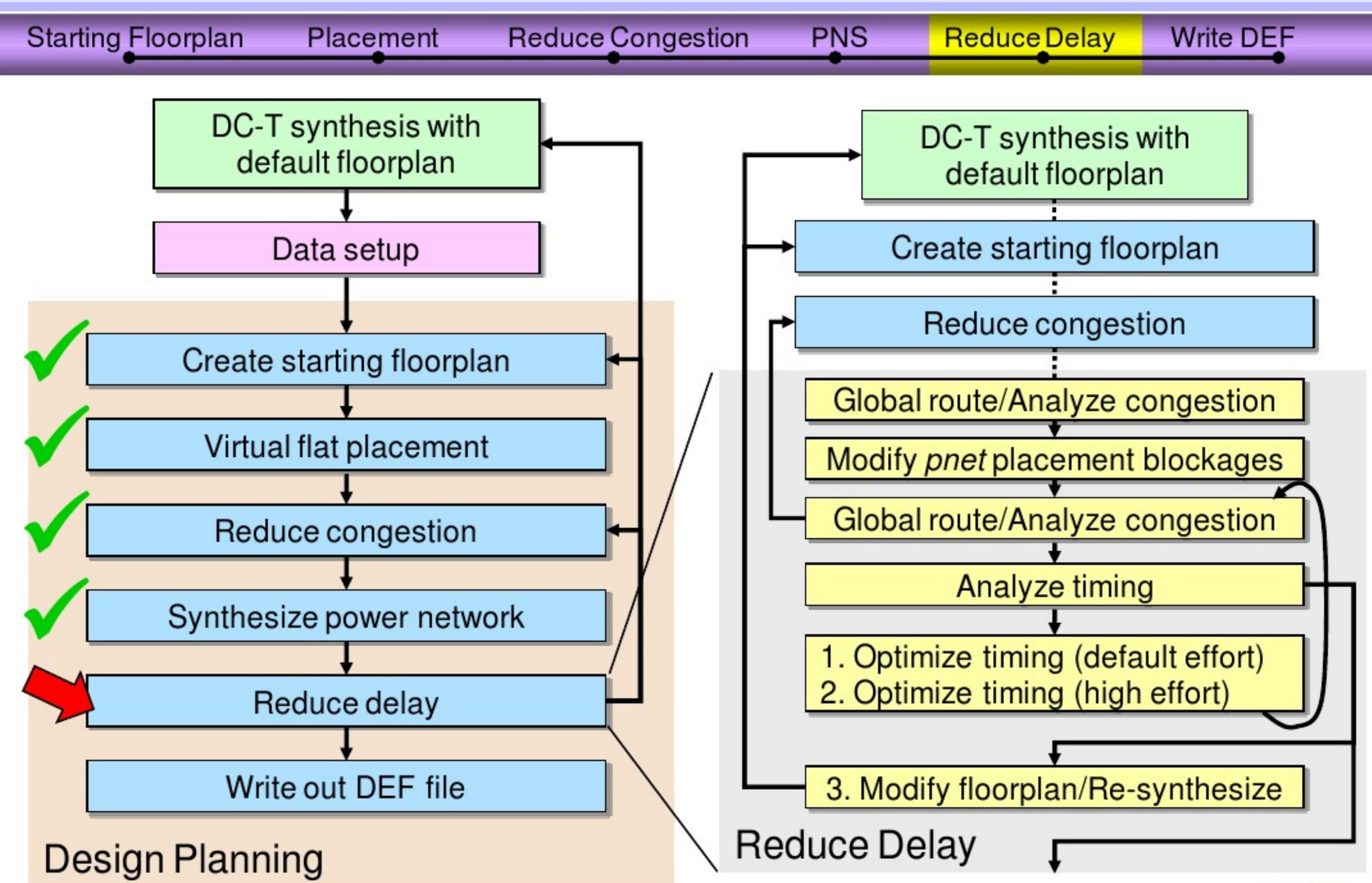
# Summary: Synthesize the Power Network





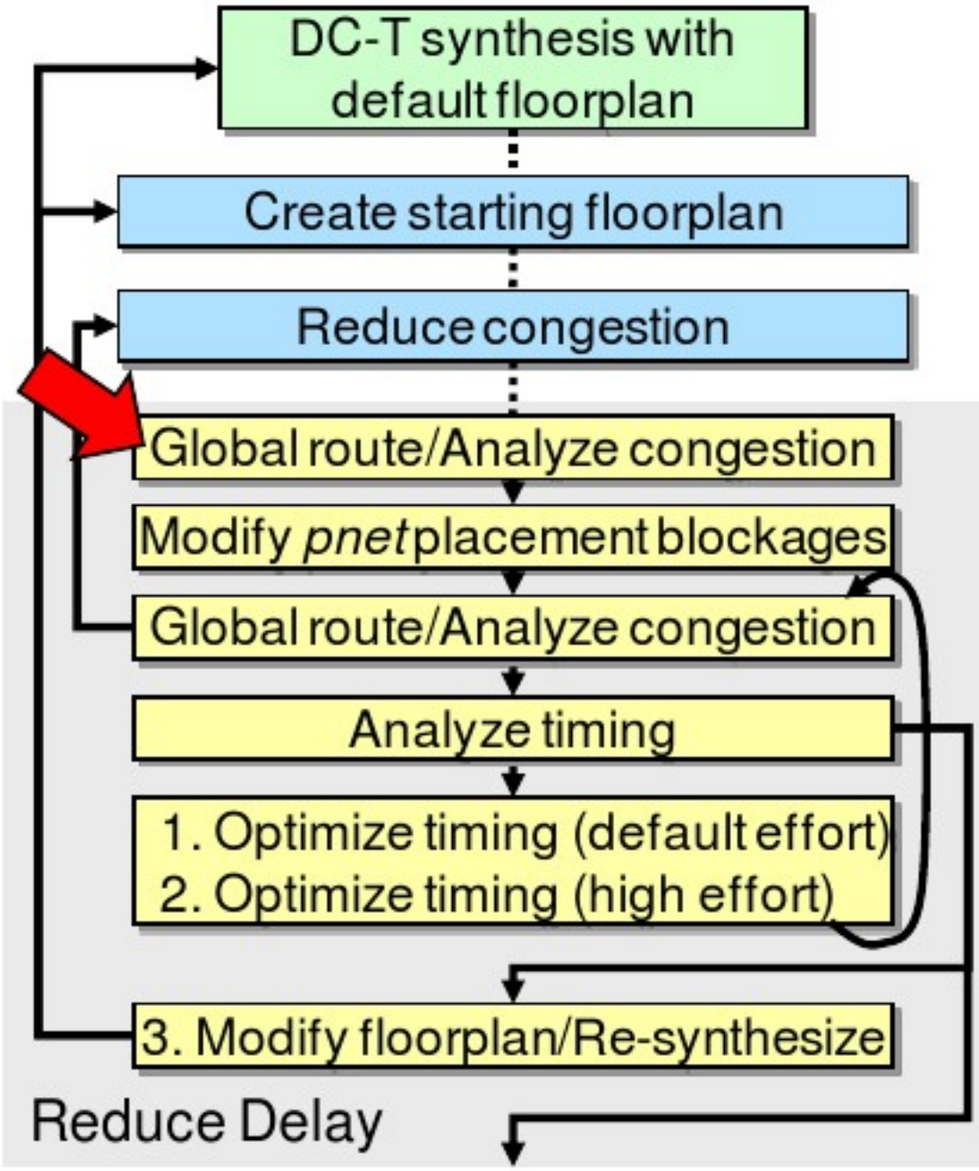

```
save_mw_cel -as DESIGN_pre_pns
derive_pg_connection ...
set_fp_rail_region_constraints  ...
create_fp_group_block_ring ...
commit_fp_group_block_ring
set_fp_rail_region_constraints -remove
set_fp_rail_constraints ...
synthesize_fp_rail ...
# Modify power network constraints
# and re-synthesize, as needed.
create_fp_virtual_pad; # If needed, re-
# initialize the floorplan with the
# added P/G pads.
commit_fp_rail
preroute_instances
preroute_standard_cells ...
analyze_fp_rail ...
# If IR-drop is not acceptable start
# over with the "pre_pns" cell.
set_pnet_options ...
legalize_fp_placement
```

# Reduce Delay



# Global Route and Analyze Congestion

Reduce Delay




- **Before performing timing analysis perform an <u>actual</u> global route**

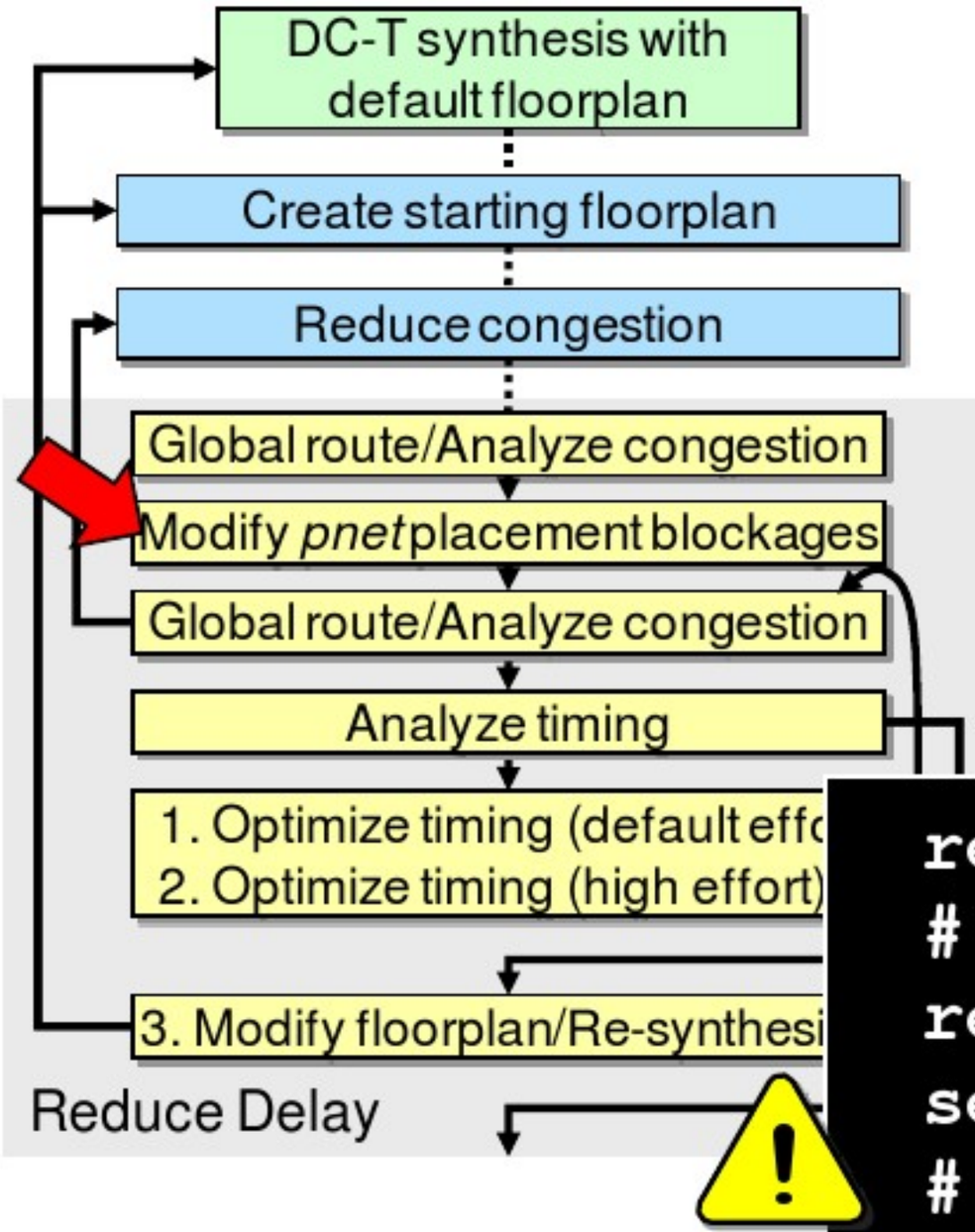
```
route_zrt_global
```

  - By default timing analysis is based on "virtual route"
  - Global route allows more accurate timing analysis

- **Analyze the congestion map**

# Modify Power Net Placement Blockages

Reduce Delay

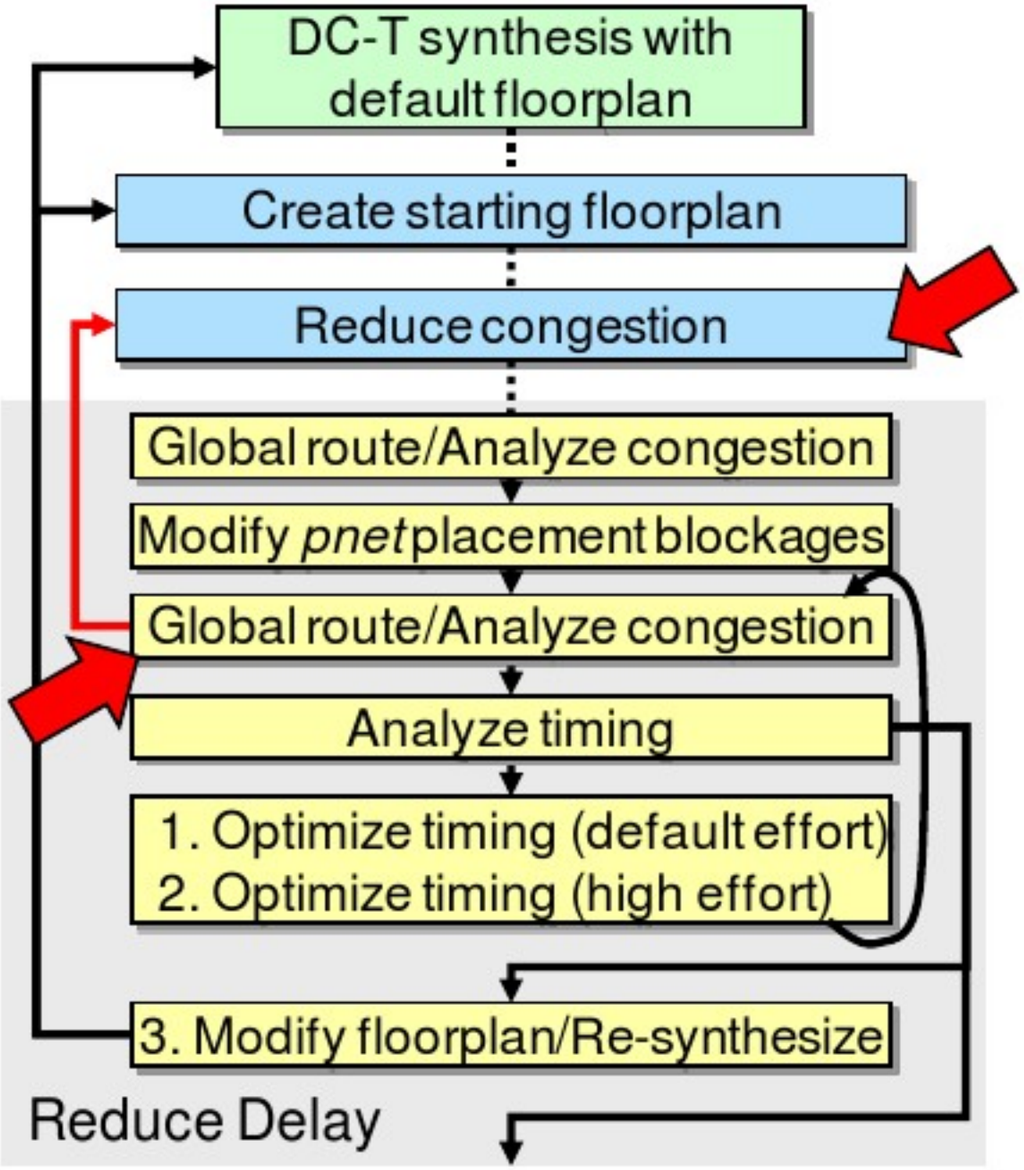



- **If congested, check and modify power net (*pnet*) placement blockage options, as needed:**
  - If *complete* blockages exist, remove or change to *partial*
- **Legalize placement**

```
report_pnet_options
# Remove blockages:
remove_pnet_options OR
set_pnet_options -none {M6 M7}
# Modify existing options:
remove_pnet_options
set_pnet_options -partial {M2 M3}
legalize_fp_placement
```

# Return to "Reduce Congestion", if Needed

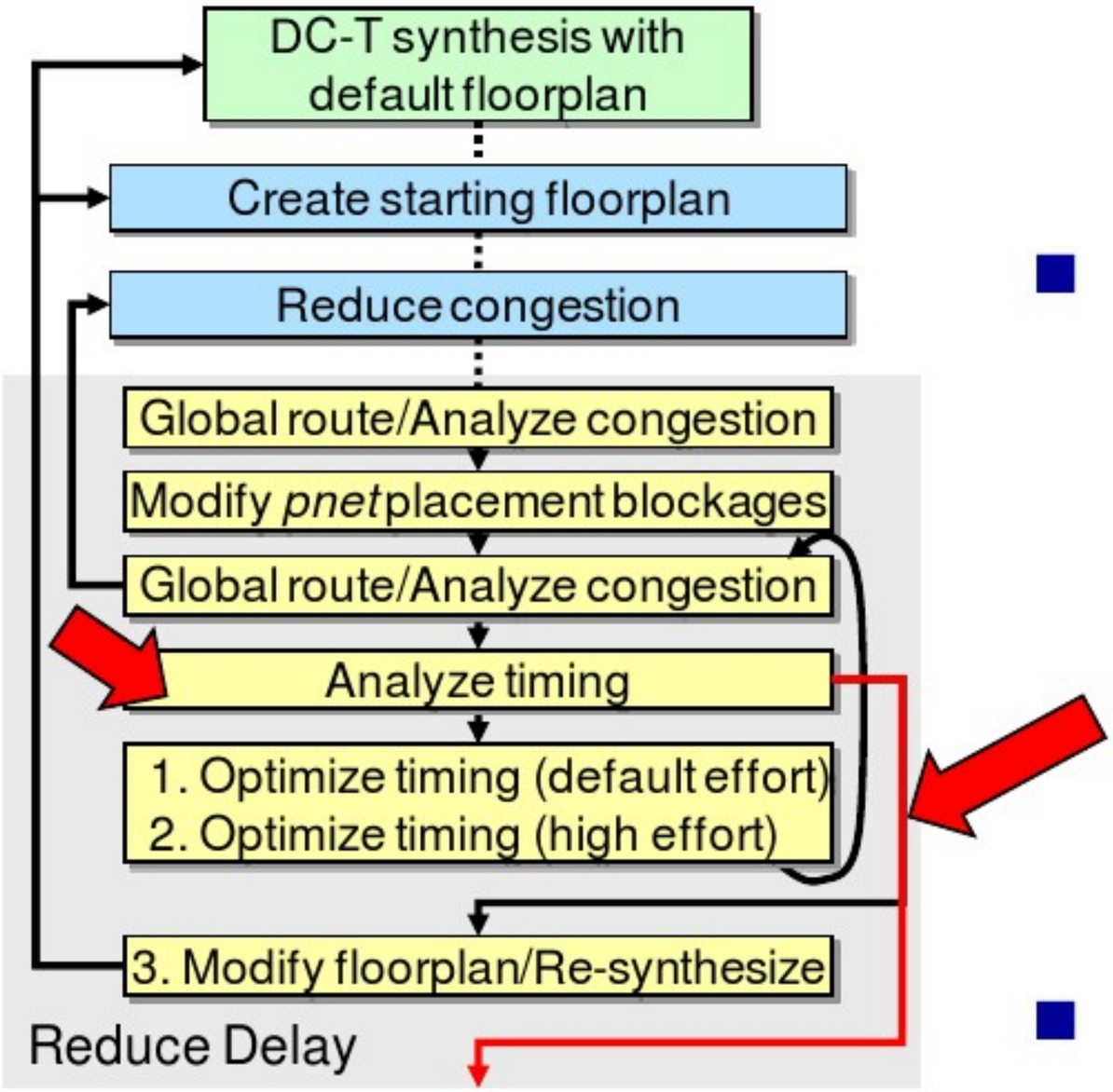
Reduce Delay




- **Perform another global route and analyze congestion**
- **If modifying *pnet* options is not applicable or does not fix the congestion issues, return to the "Reduce Congestion" step**
- **Once congestion is under control continue with timing analysis**

# Analyze Timing

Reduce Delay



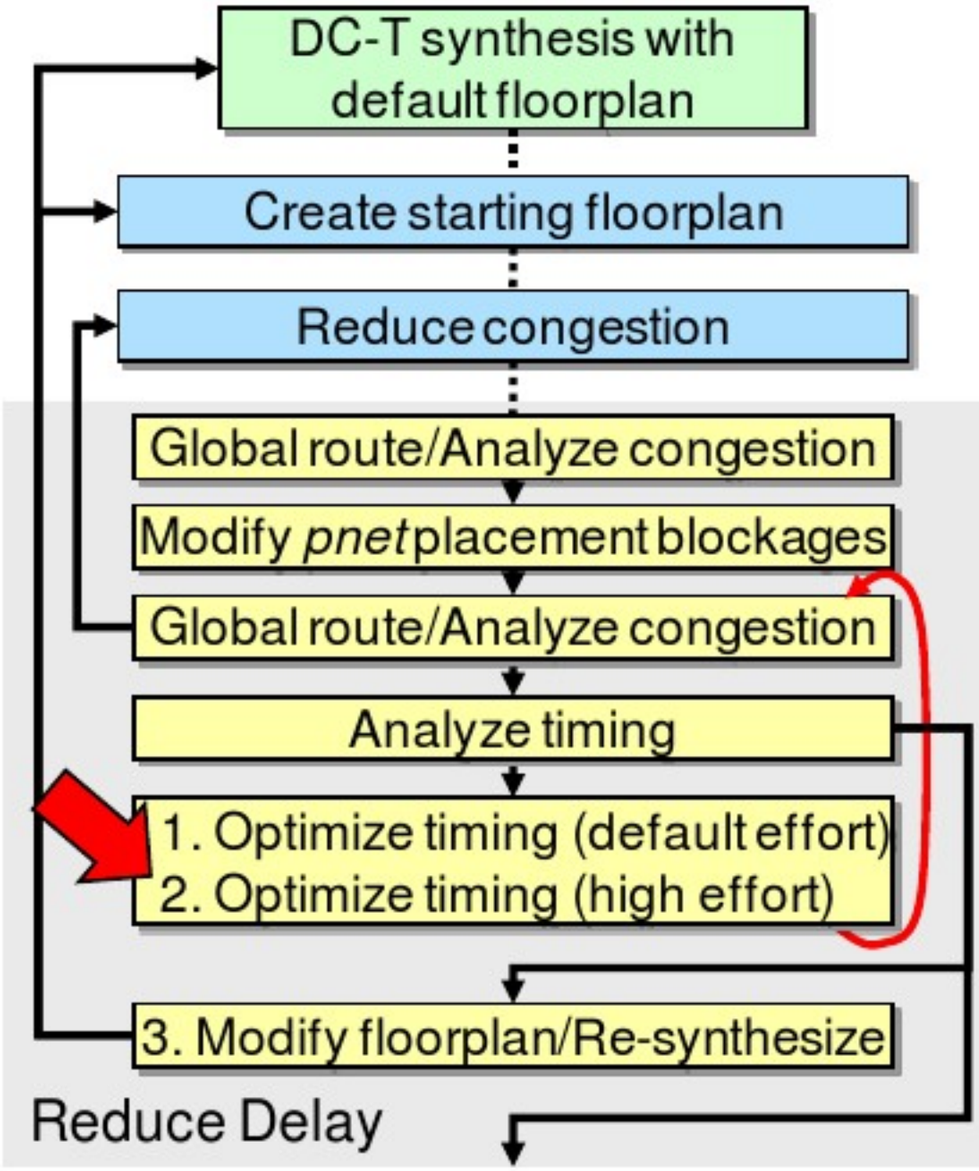

- **Generate a timing report**
  - Parasitic RC extraction is automatically invoked (`extract_rc`)
  - Acceptable timing: *WNS* < 15-20% of required delay

```
report_timing
```

- **If timing is acceptable skip to "Write DEF" step**

# Perform In-Place Optimization

Reduce Delay




- **If the WNS > 15-20% perform *in-place optimization***

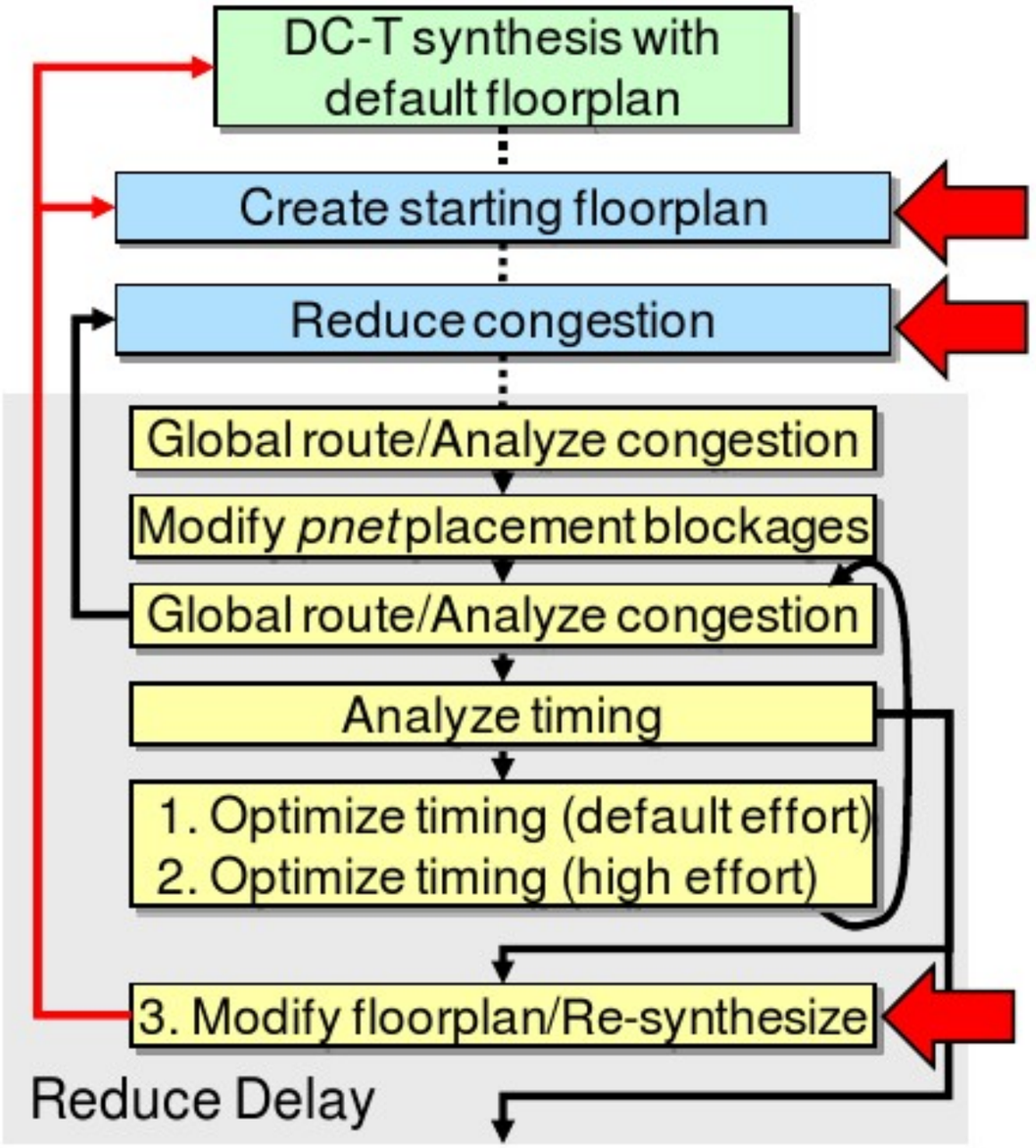
```
optimize_fp_timing -fix_design_rule
                   <-effort high>
```

- **Repeat global routing and timing analyses**
- **If timing is still not acceptable, repeat timing optimization with *high effort***
- **Repeat timing analyses steps**

- ❖ **Performs cell sizing, buffer insertion and AHFS**
- ❖ **Improves timing and DRC violations**
- ❖ **Legalizes placement**

# Modify Floorplan or Re-Synthesize, if Needed

Reduce Delay

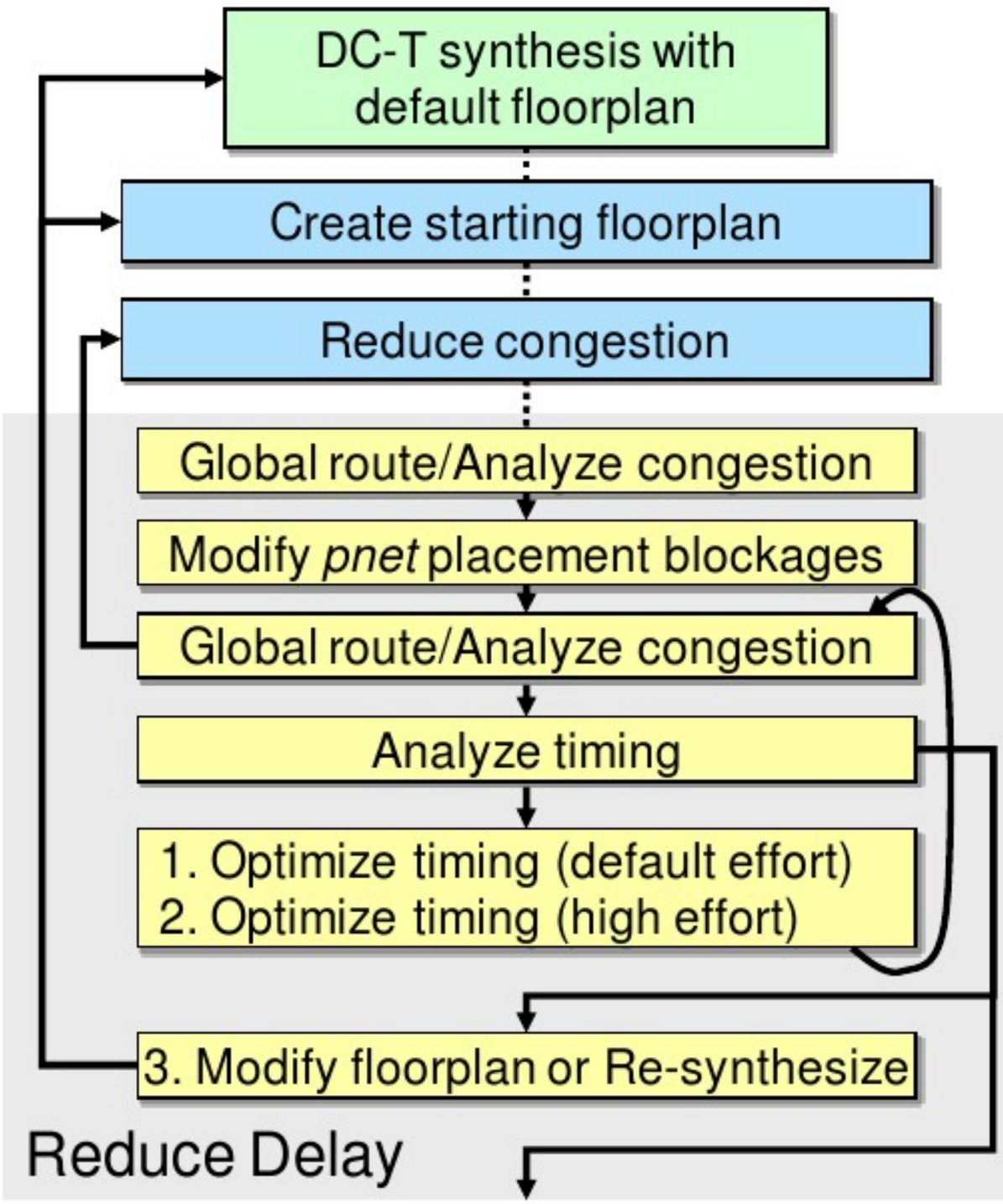



**If the timing violations are still unacceptably large after *high-effort optimization*:**

- **Modify the floorplan, or…**
- **Re-synthesize the design with**
  - Better constraints
  - Better partitioning

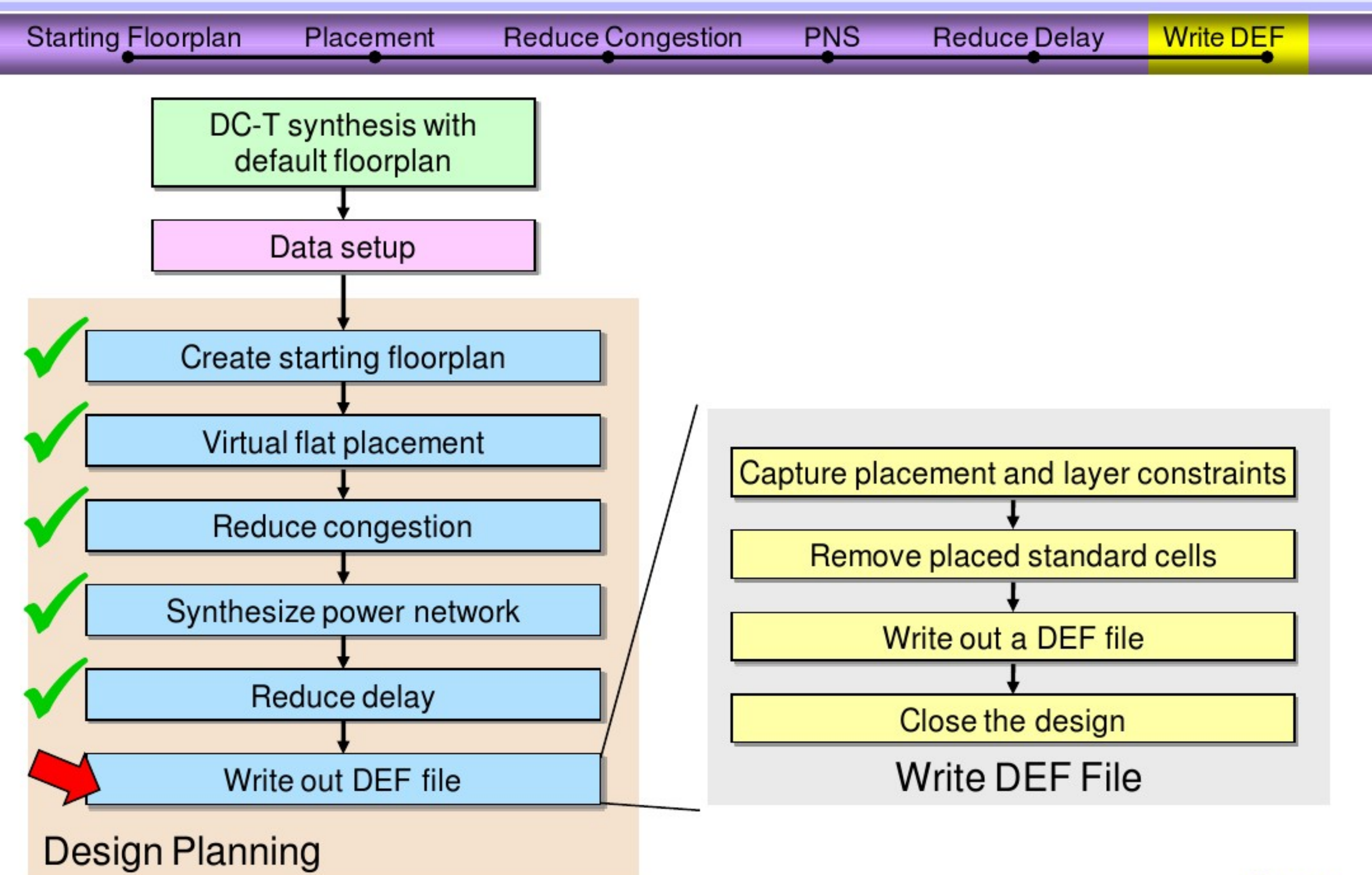
# Summary: Reduce Delay






```
route_zrt_global
# If congested modify pnet blockages
# and perform inc'l placem't, as needed:
report_pnet_options
remove_pnet_options; # OR
set_pnet_options -none {M2 M3}
set_pnet_options -partial {M2 M3}
legalize_fp_placement
route_zrt_global
# If congested goto "Reduce Congestion"
report_timing
# If timing OK skip to "Write DEF"
optimize_fp_timing -fix_design_rule
                   <-effort high>
# If timing still not OK after
# high-effort optimization then need to
# re-floorplan or re-synthesize
```

# Write Out the DEF File



# Floorplan Settings Not Saved in *DEF*

Write DEF

- **Everything about the floorplan will be captured using `write_def`, with the exception of the settings below**
- **These *commands* must be re-applied once after loading *DEF***
  - Settings are saved with the cell

```
set_ignored_layers -max M7
set_pnet_options -partial | -complete …
```

Apply after loading *DEF* and save the cell

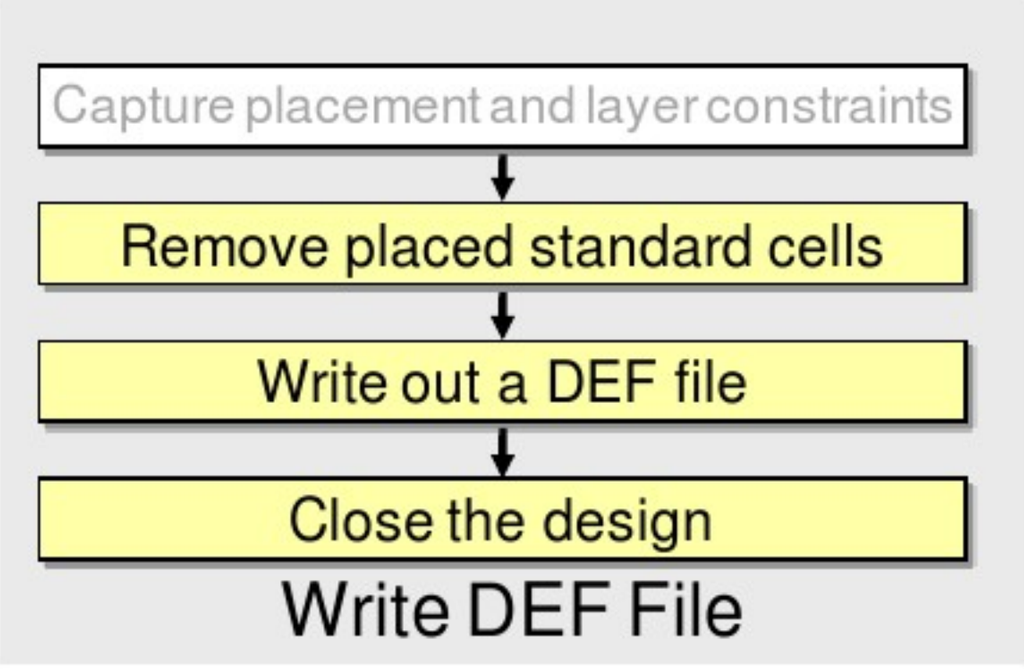
- **These *variable* settings must be re-applied whenever a new IC Compiler session is invoked**
  - Variable settings are not saved with the cell

```
set_app_var physopt_hard_keepout_distance 10
set_app_var placer_soft_keepout_channel_width 25
```

Re-apply at the beginning of each new ICC session or put in `.synopsys_dc.setup`

# Write out *DEF* File



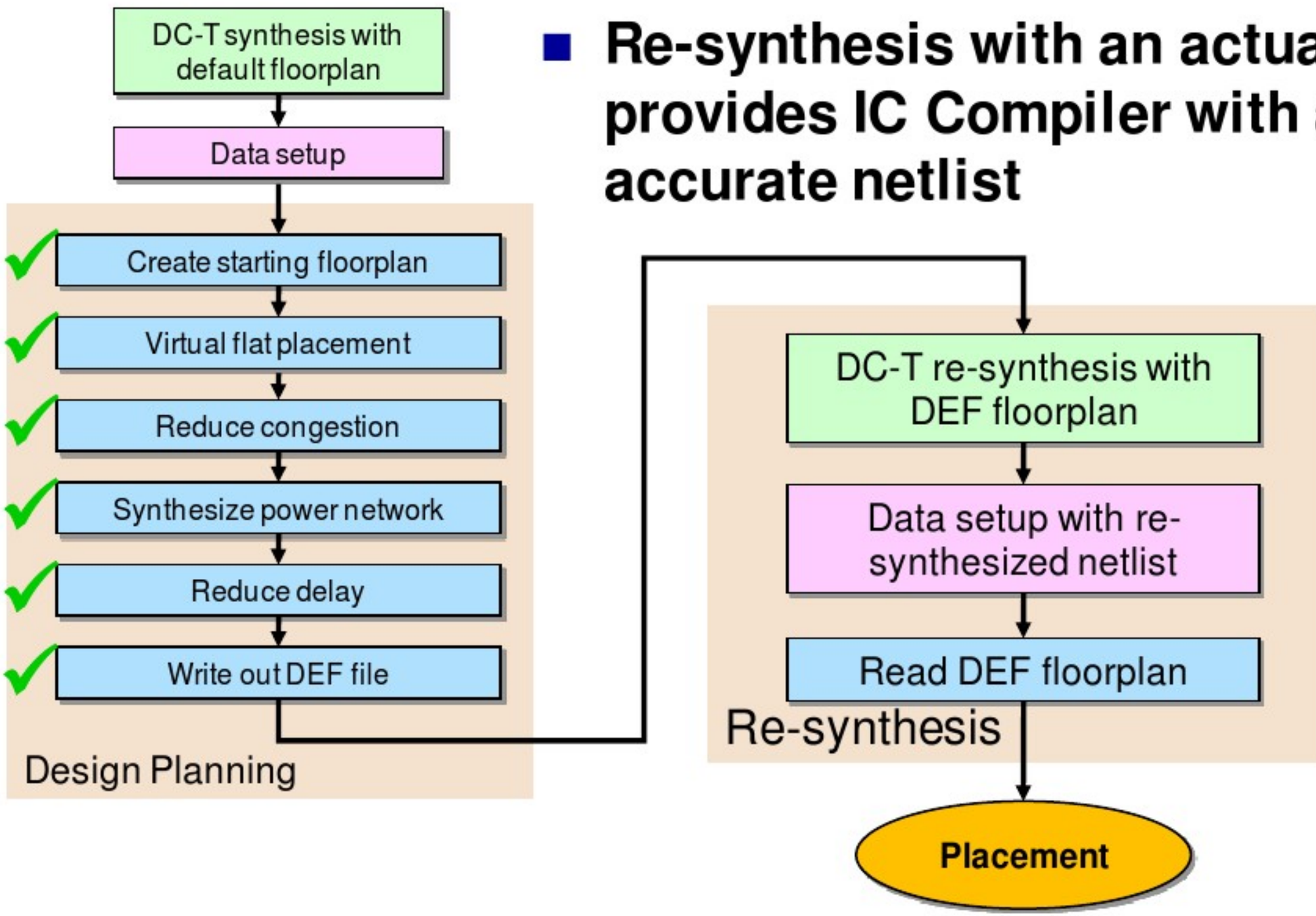



- **Write out a *DEF* file:**
  - Will be used by *Design Compiler Topographical* for re-synthesis
  - Will be loaded into IC Compiler after re-synthesis and data setup
- **Close the design cell without saving it**
  - Only the above files are needed

```
remove_placement -object_type standard_cells
write_def -version 5.6 -placed -blockages -all_vias \
      -rows_tracks_gcells -routed_nets -specialnets \
                                -output FLOORPLAN.def
close_mw_cel
```

# Re-Synthesize Before Placement

- Initial synthesis used a default DC-T floorplan
- Re-synthesis with an actual floorplan provides IC Compiler with a more accurate netlist



# Re-Synthesize Before Placement

Design Planning

```
UNIX% dc_shell -topographical
dc_shell-topo> read_verilog ORIGINAL_RTL.v
dc_shell-topo> source ORIGINAL_CONSTRAINTS.cons
dc_shell-topo> extract_physical_constraints FLOORPLAN.def
dc_shell-topo> ...
dc_shell-topo> compile_ultra ...
dc_shell-topo> ...
dc_shell-topo> write -format ddc -output DESIGN_2.ddc
```

Load the floorplan into DC-T

Write out the 2nd pass netlist

```
UNIX % icc_shell
create_mw_lib design_lib_orca_2 ...
set_tlu_plus_files ...
import_designs DESIGN_2.ddc -format ddc -top DESIGN_TOP
derive_pg_connection ...
source tim_opt_ctrl.tcl
```

Data Setup: Create a 2nd pass starting cell

```
read_def FLOORPLAN.def
set_ignored_layers -max M7
set_pnet_options -partial | -complete …
save_mw_cel -as DESIGN_ready_for_placement
set_app_var physopt_hard_keepout_distance 10
set_app_var placer_soft_keepout_channel_width 25
```

Load the floorplan

Re-apply *routing layer* and *pnet* constraints

Re-apply global keepouts*

Placement

* Can also be placed in `.synopsys_dc.setup`

# Test For Understanding (1 of 2)

1. **Circle the correct statement(s) about *PNS*:**
   a. Allows what-if analysis based on different user constraints
   b. Creates DRC/ERC clean routes with `synthesize_fp_rail`
   c. Can automatically calculate the width and number of straps based on power and IR-drop constraints
   d. Allows what-if analysis with "virtual" IO signal pads

2. **A "partial" *pnet* blockage allows cells to be placed only under the specified metal layers (DRC permitting) – all other layers are completely blocked.**

   ***True* or *False?***

# Test For Understanding (2 of 2)

3. **Circle the correct statement(s) regarding `optimize_fp_timing -fix_design_rule`:**
   a. Optimizes placement to improve DRC violations
   b. Optimizes the logic to improve timing violations
   c. Optimizes placement to reduce congestion
   d. Optimizes placement to improve timing violations

4. **Why is it recommended to execute `route_zrt_global` prior to timing analysis?**

5. **What is the *DEF* file used for?**

6. **What is the point of doing DC-T re-synthesis <u>after</u> the floorplan is completely defined?**

# Summary

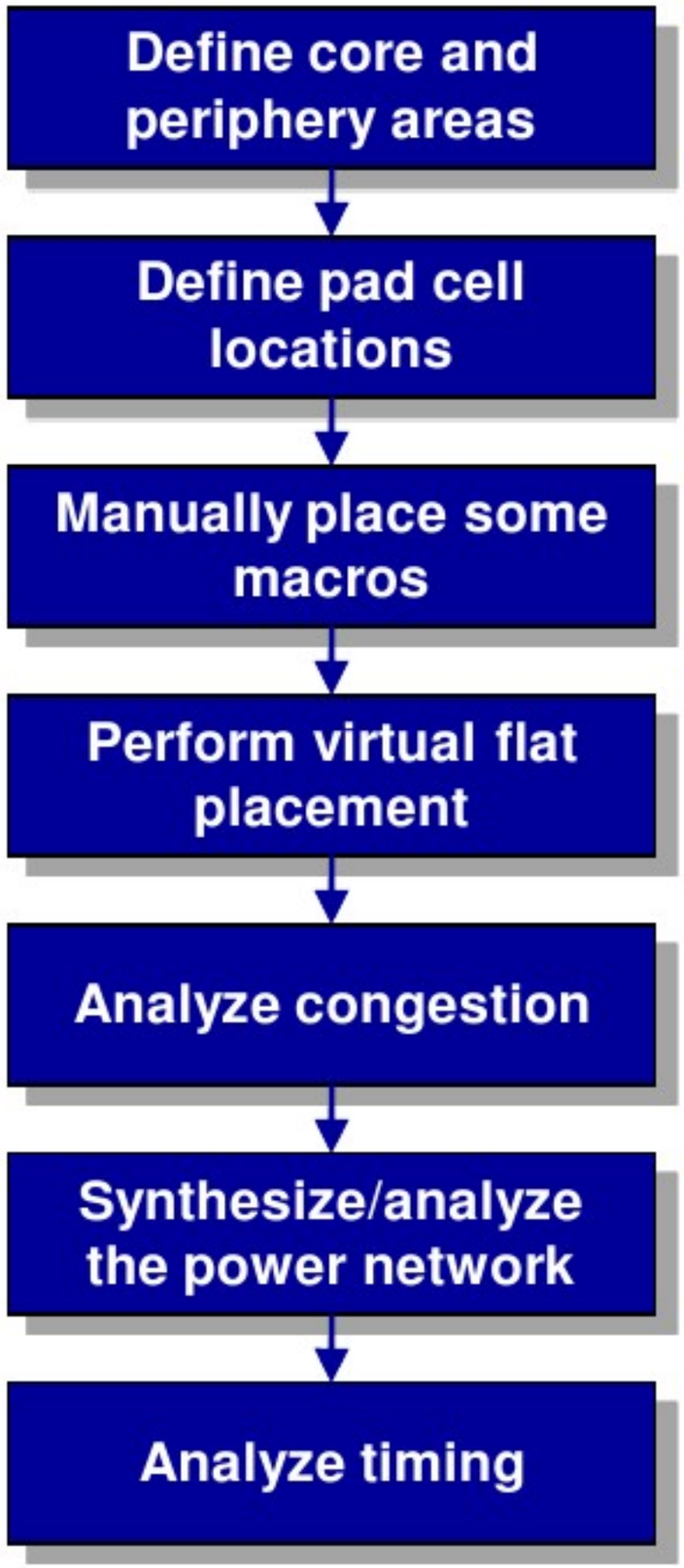
**You should now be able to:**

- **Use IC Compiler to create a non-hierarchical chip-level floorplan**
- **Create a floorplan that is likely to be routable and achieve timing closure**

**Sign up for the "ICC2: HDP" workshop to learn about hierarchical design planning!**

# Lab 2: Design Planning

100 minutes



# Appendix A

## Macro placement constraints:

```
set_fp_macro_options ...
set_fp_macro_array ...
set_fp_relative_location ...
```

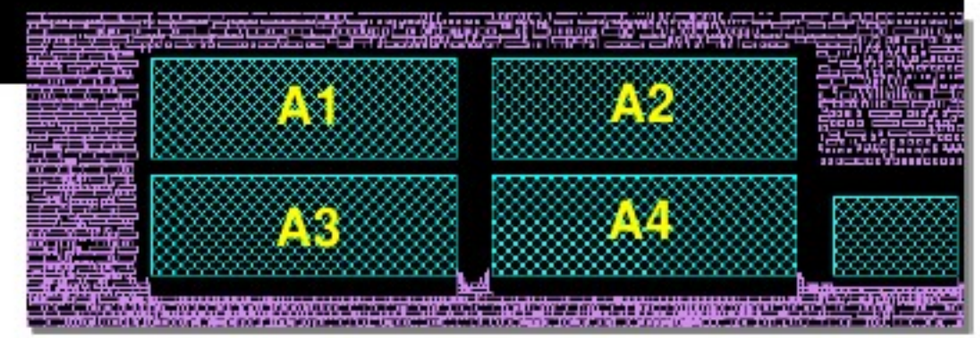
## Placement constraints are "soft"

- Placer will *try* to honor them but is not required to

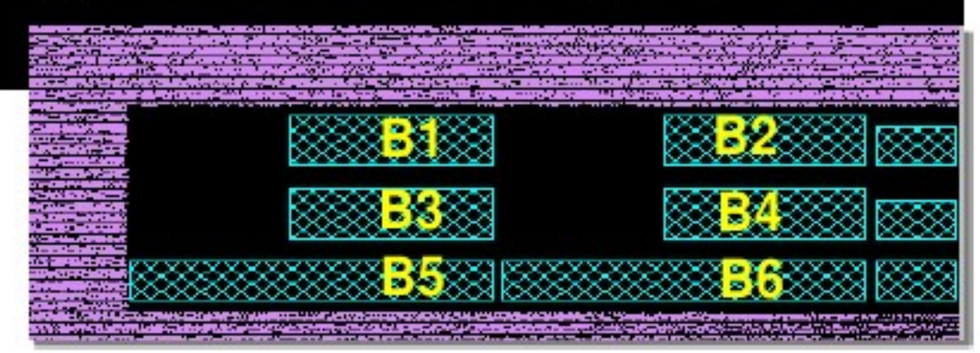
# Macro Constraints: Arrays

Starting Floorplan

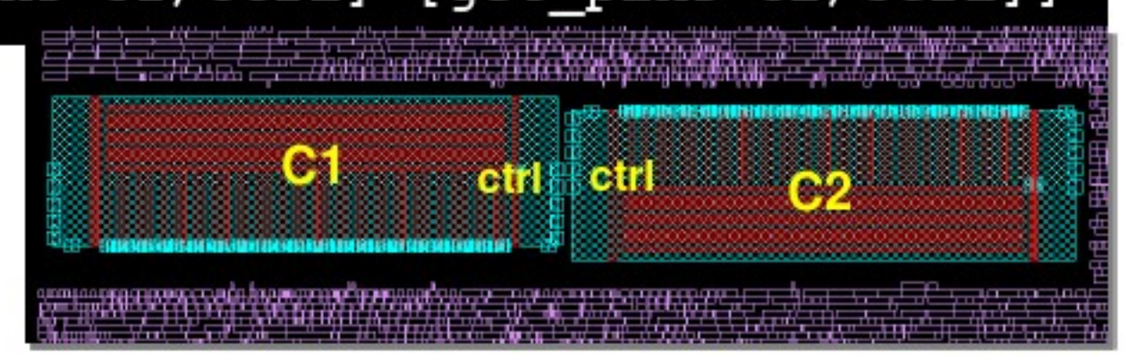
```
set_fp_macro_array -name A_array -elements  \
 [list [get_cells A1 A2] [get_cells A3 A4]] \
 -x_offset 15 -y_offset 5
```




```
set_fp_macro_array -name B_array -elements \
 [list [get_cells B1 B2] [get_cells B3 B4] [get_cells B5 B6]] \
 -x_offset 4 -y_offset 12 -align_2d rb
```



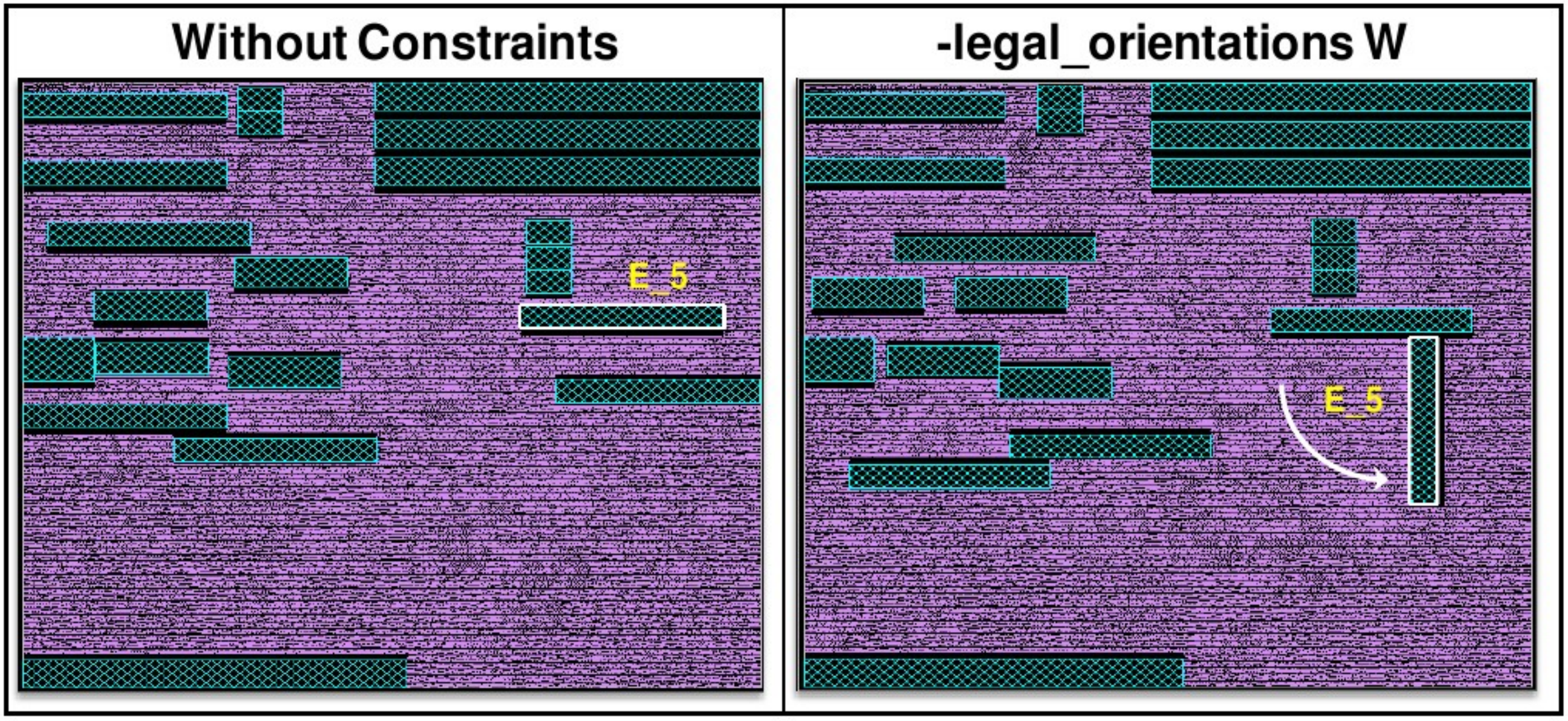

```
set_fp_macro_array -name C_array -elements [get_cells C1 C2] \
 -x_offset 4 -align_pins [list [get_pins C1/ctrl] [get_pins C2/ctrl]]
```



# Macro Constraints: Legal Orientation Option

Starting Floorplan

```
set_fp_macro_options [get_cells E_5] -legal_orientations W
```

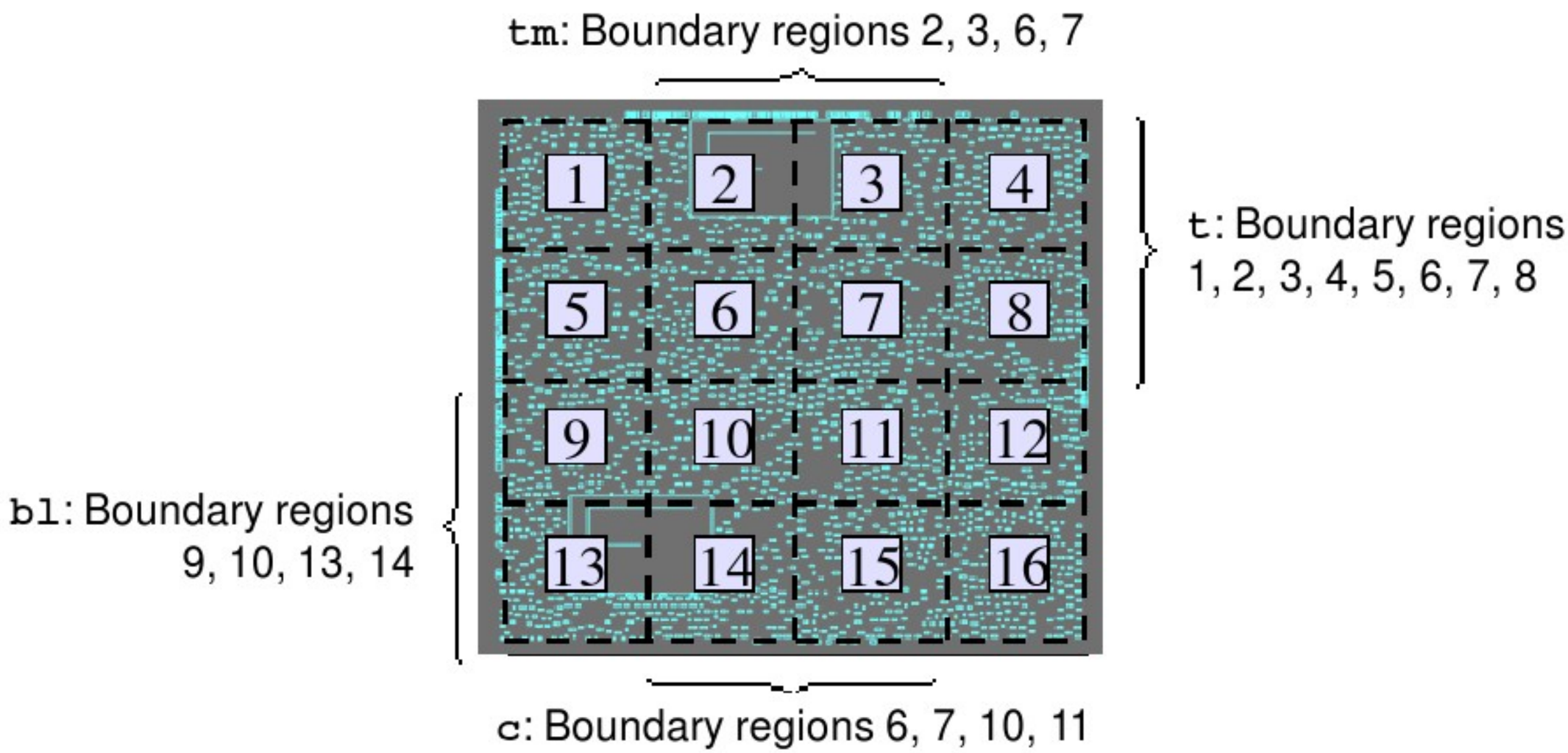


# Macro Constraints: Anchor Bound Option

Starting Floorplan

```
set_fp_macro_options collection_of_cells \
 -anchor_bound <l, r, t, b, bl, tl, br, tr, bm, tm, lm, rm, c>
```

**Cells are placed in specified ½ or ¼ core areas**

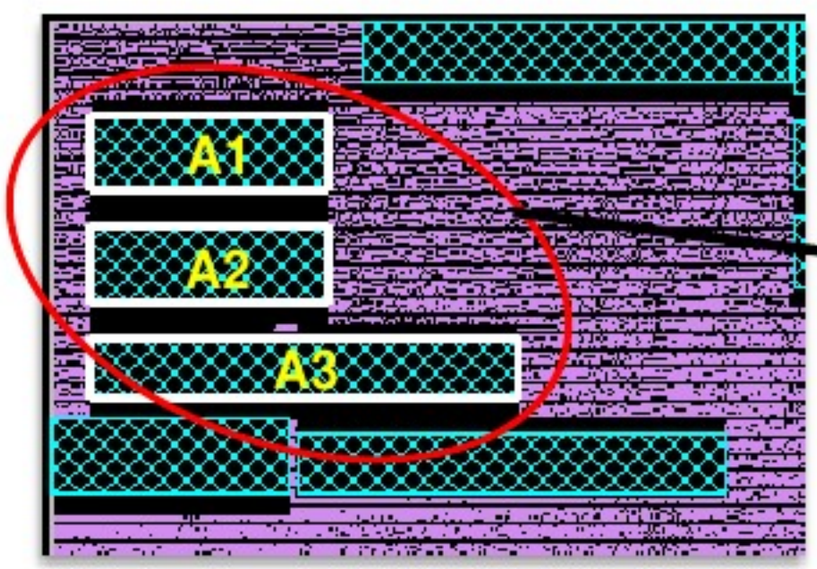


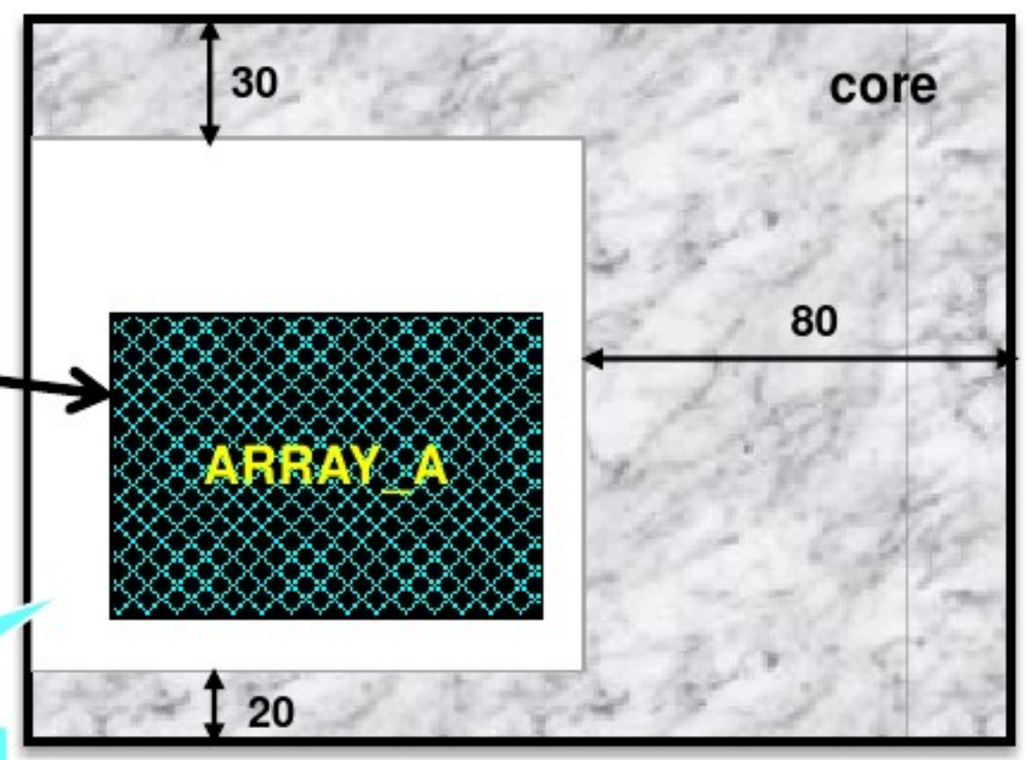
# Macro Constraints: Side Channel Option

Starting Floorplan

```
set_fp_macro_array -name ARRAY_A -vertical \
              -elements [get_cells "A1 A2 A3"]
set_fp_macro_options ARRAY_A \
              -side_channel "0 80 30 20"
```

**Cells or arrays are spaced <u>minimum</u> distances from the core edges**




The macro can be placed anywhere in this "open" area

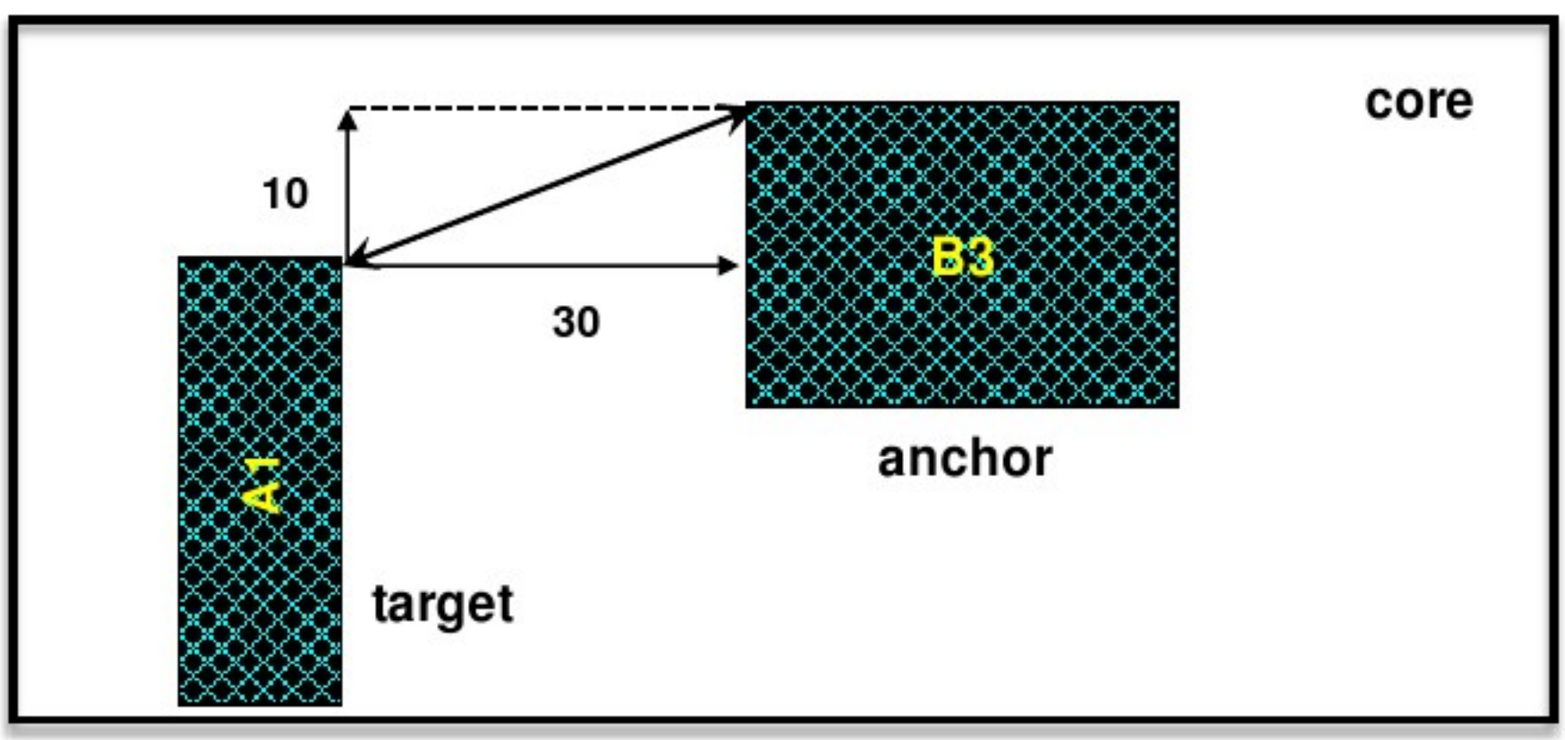
# Macro Constraints: Relative Location

Starting Floorplan

`set_fp_relative_location` **sets a relative location constraint on a target cell with respect to an anchor object**

- Anchor: A fixed cell, another cell with a relative-location, or a corner of the core area

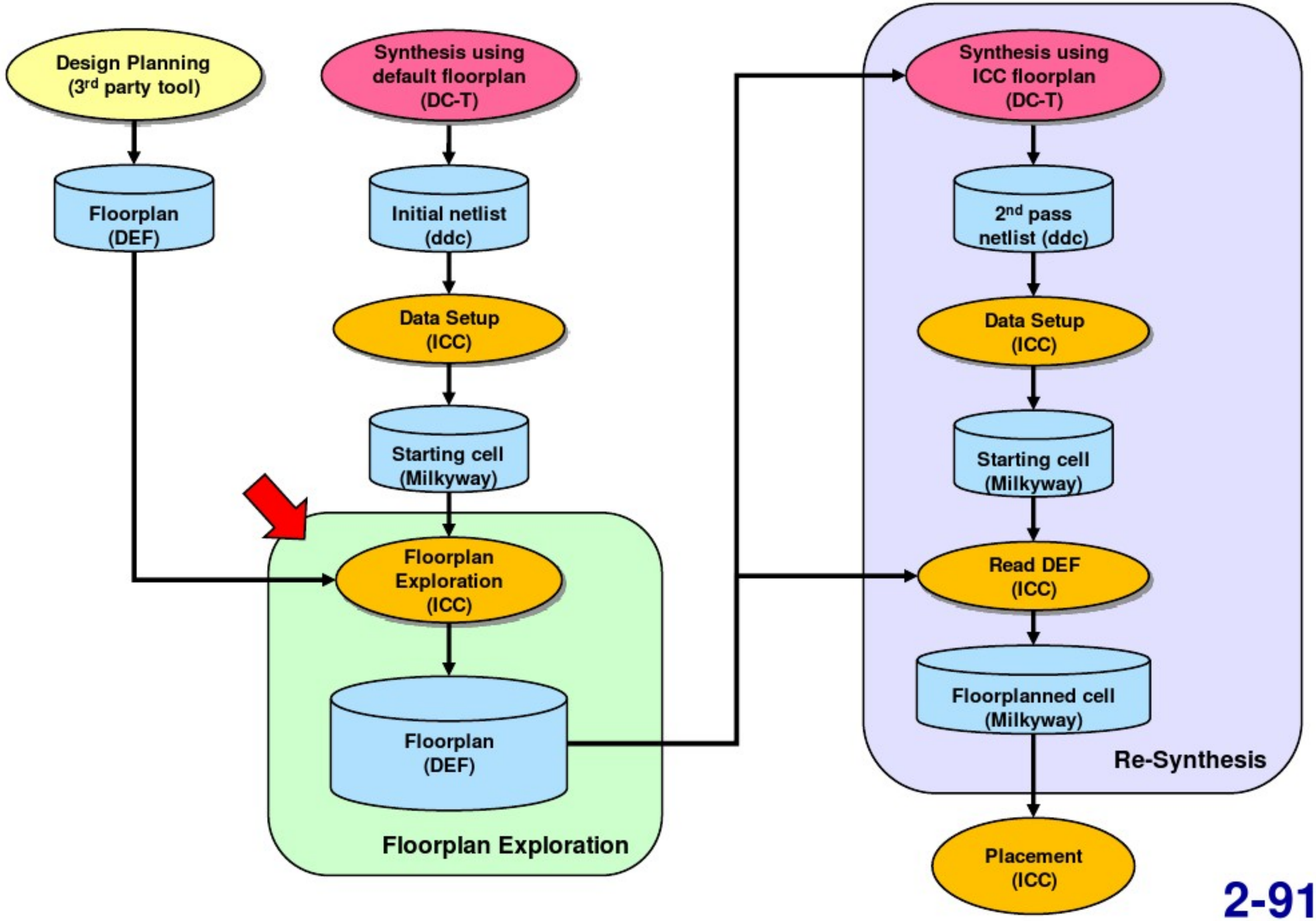
```
set_fp_relative_location -name RP1 -target_cell "A1"
      -target_orientation "S" -target_corner "tr" \
      -anchor_object "B3" -anchor_corner "tl"
      -x_offset 30 -y_offset 10
```



# Appendix B

***Floorplan exploration and re-synthesis*** **for floorplans created by 3rd party tools**

# Floorplan Exploration and Re-Synthesis Flow

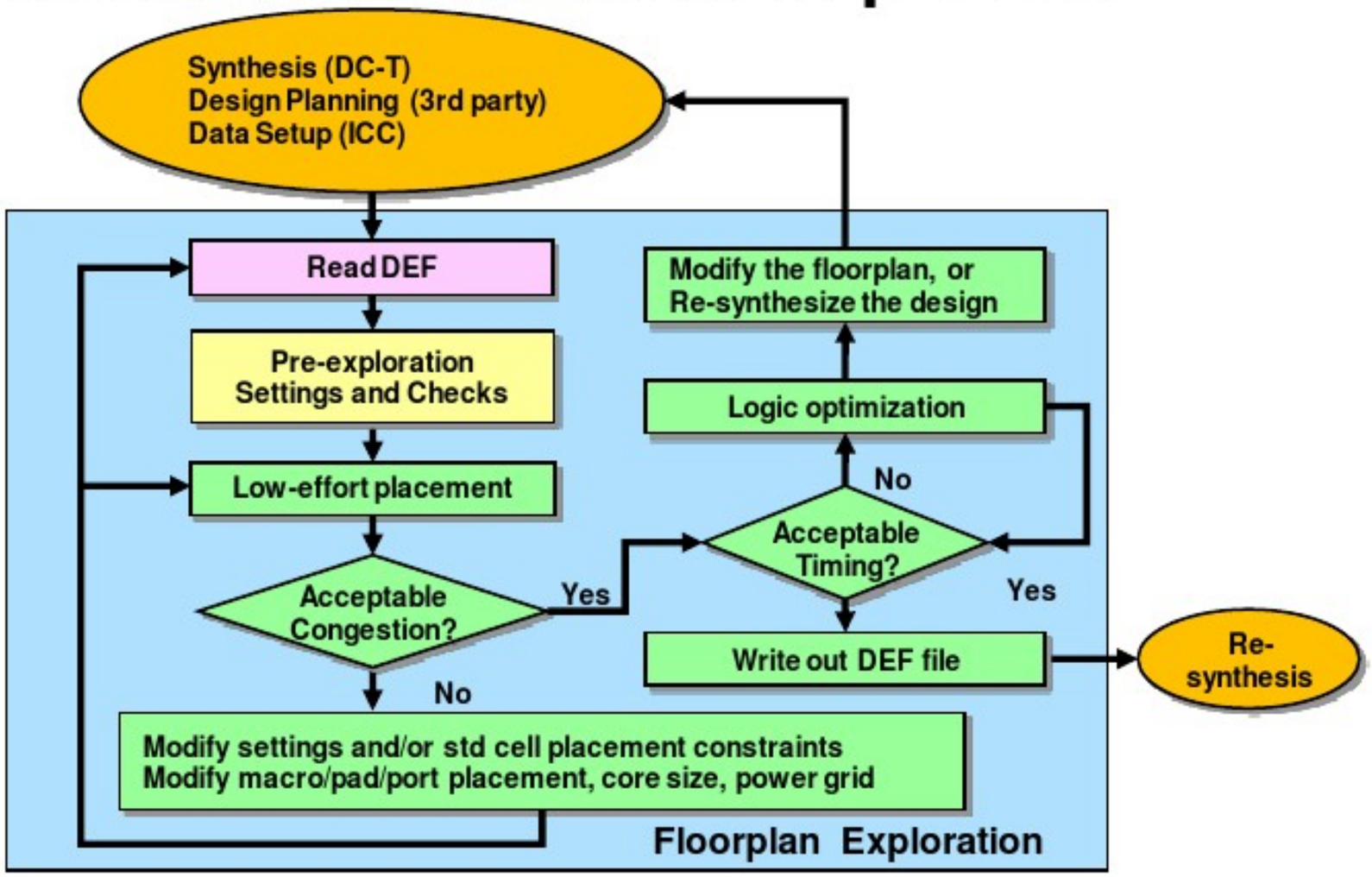


# Floorplan Exploration Objectives

**Quickly determine, before placement, if the final layout may run into routability or timing issues**

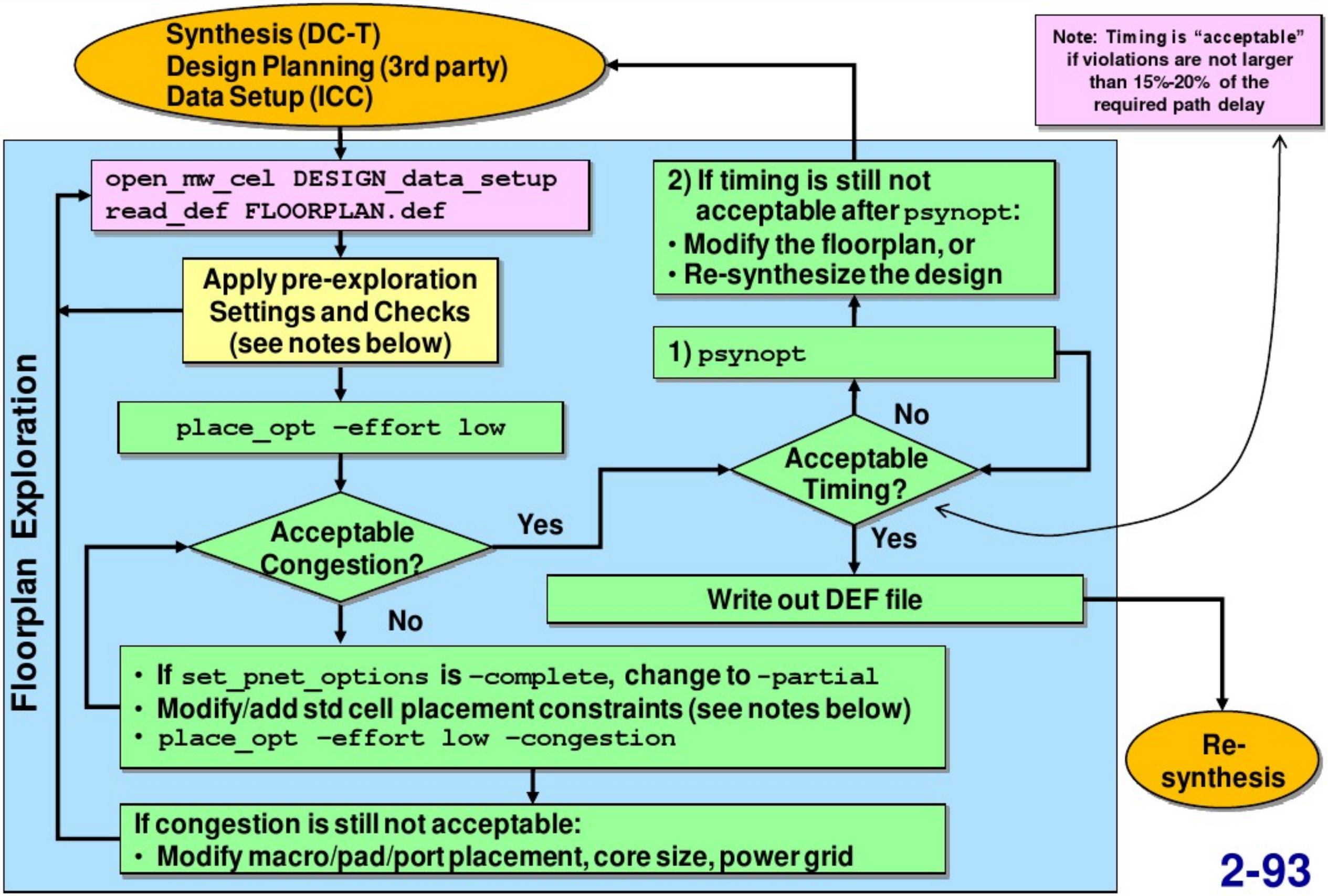
**If so, determine what should be modified to prevent these issues:**

- **Standard cell placement constraints (e.g. blockages)**
- **Floorplan**
- **Synthesized netlist**

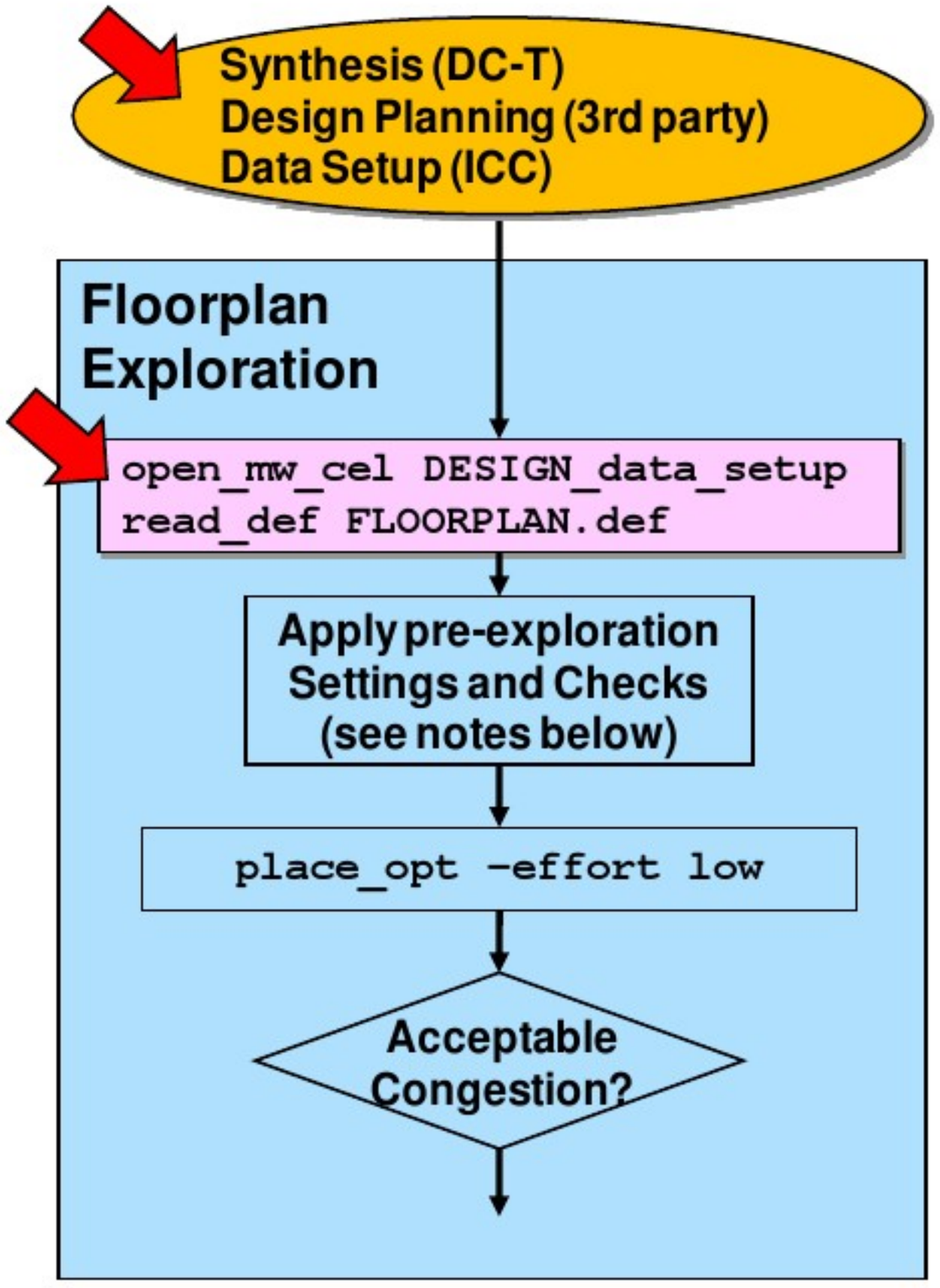



**Floorplan exploration is recommended for designs that were floor-planned using a 3rd party tool**

# Overview: Floorplan Exploration

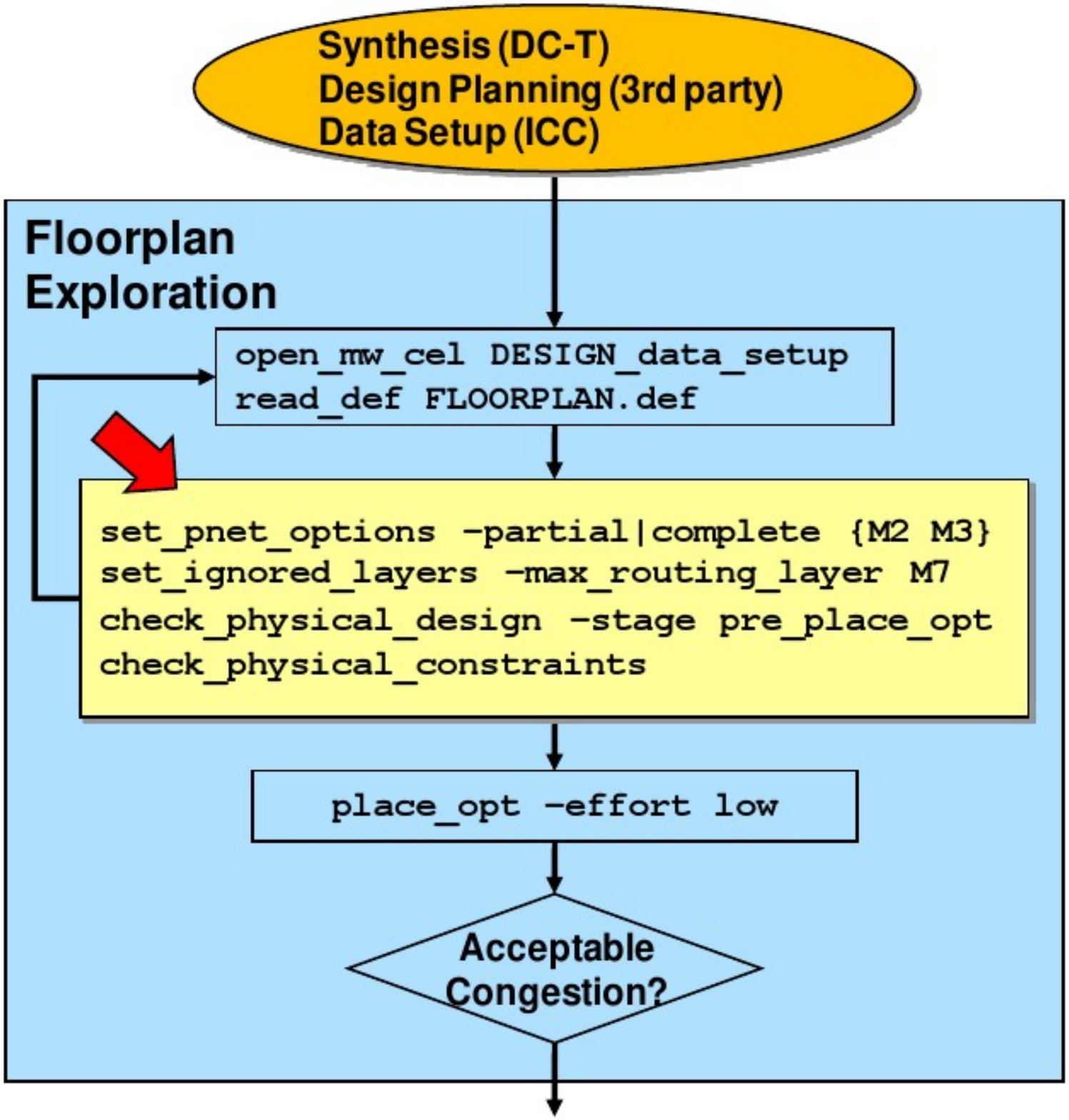


# Load the Floorplan Definition (DEF) File




- Perform *synthesis* with DC-Topographical
- Perform *Data Setup* in IC Compiler with the netlist from synthesis
- Perform *design planning* with a 3rd party tool
  - Write out the floorplan definition (*DEF*) file
- Open the *starting cell* and load the *DEF* file
  - The graphical view is updated to reflect the floorplan

# Pre-exploration Settings and Checks




- **Define the *maximum routing layer* and apply *pnet* options, as needed**
  - See 2-16 and 2-60
- **Perform two checks before placement**
  - Modify the floorplan to correct problems (see next slide)

# Check Placement Readiness

`check_physical_design -stage pre_place_opt` **checks the readiness for placement of:**

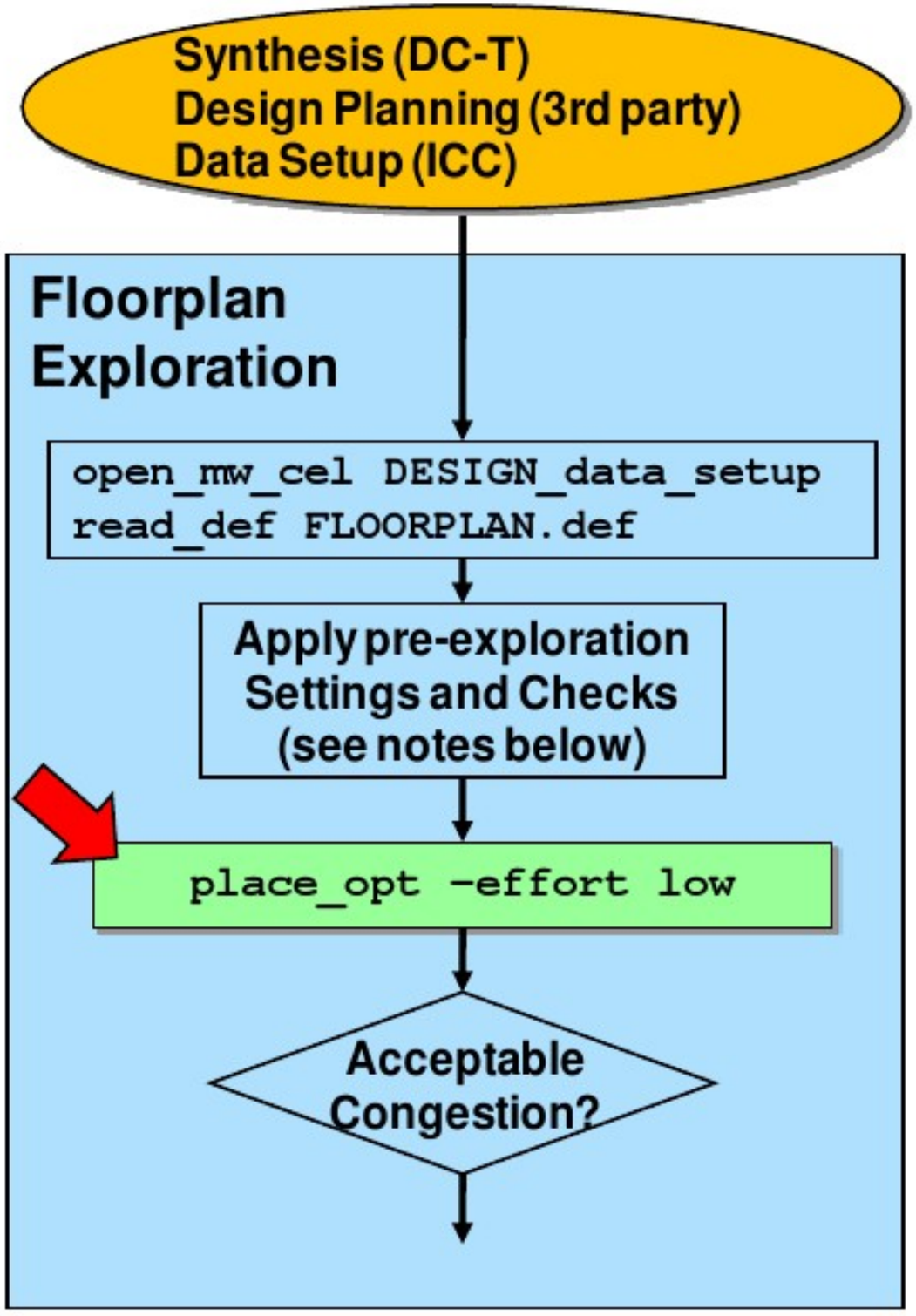
- Floorplan
- Netlist
- Design constraints

`check_physical_constraints` **reports:**

- Cells placed in “hard placement blockage” areas
- Metal layer inconsistencies against the library
- R/Cs for routing layers
- Narrow placement regions (“chimneys”)
- Legal sites for cell placement
- Large RC variations between metal layers

**Modify the floorplan, constraints or libraries as needed**

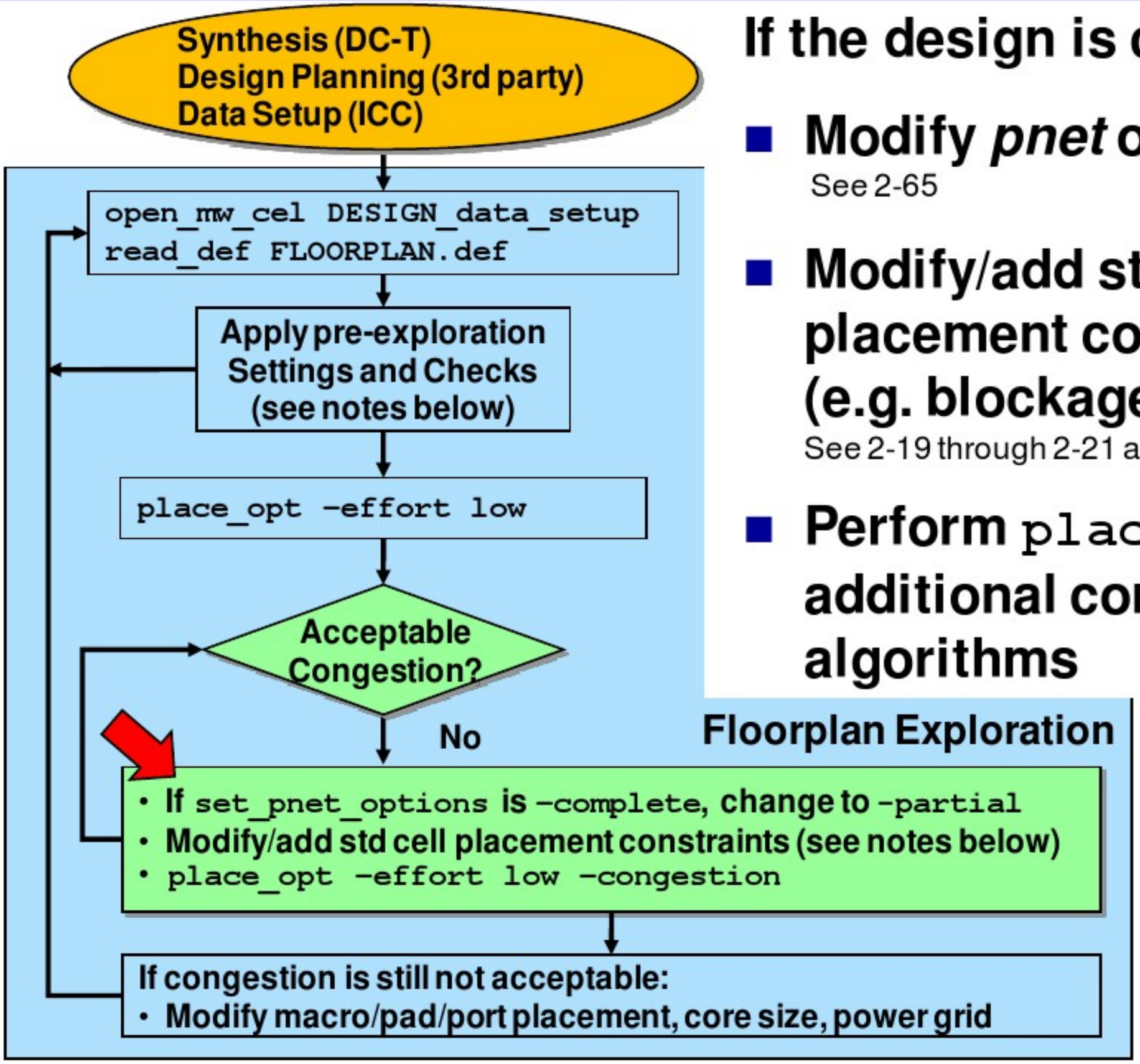
# Perform Low-effort Placement Optimization




```
place_opt -effort low
```

- **Performs concurrent timing- and congestion-driven *placement* and *logic optimization***
- **The *low effort* minimizes logic optimization runtime**
  - Ideal for "exploration" purposes

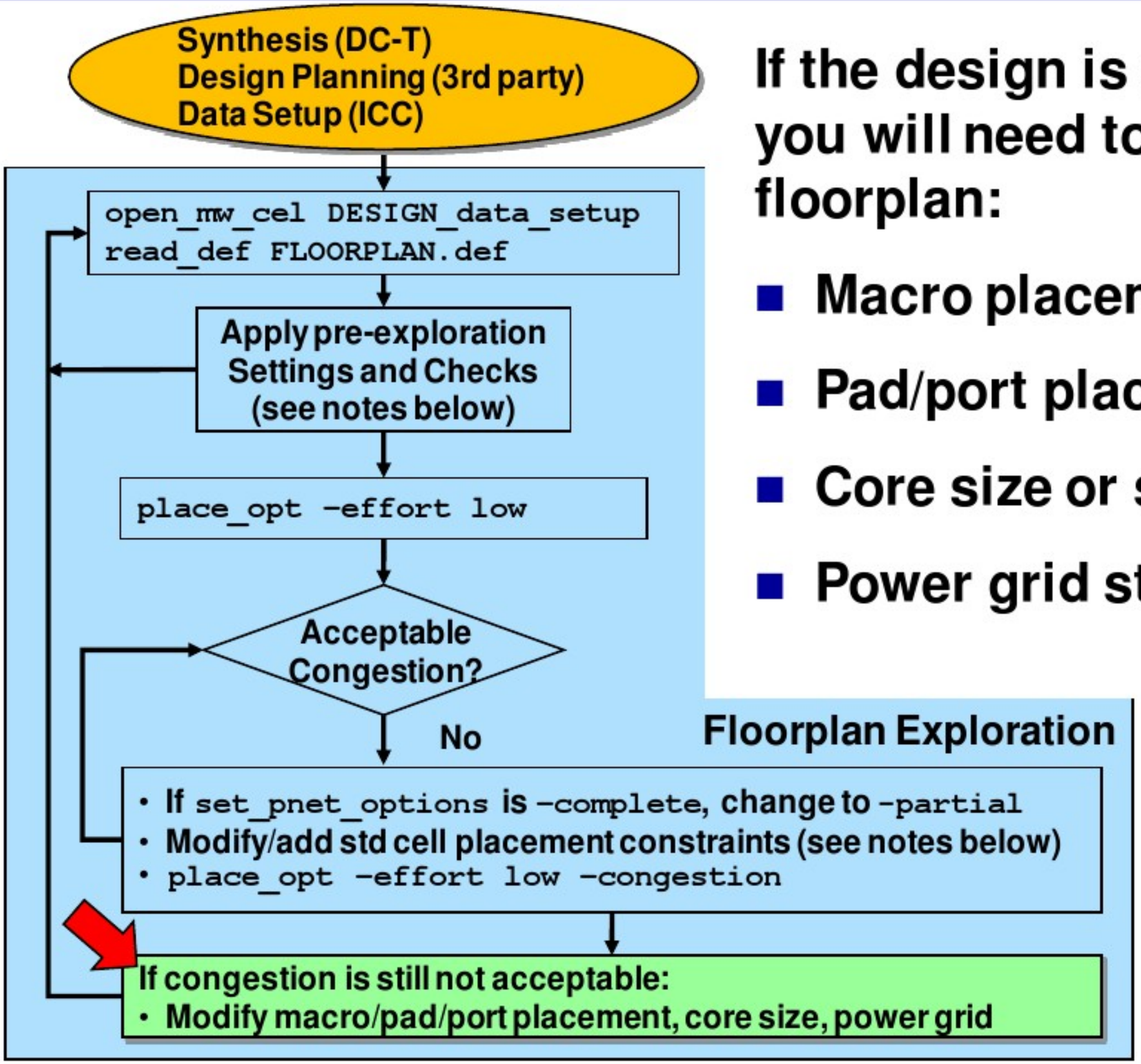
# Check and Fix Congestion




**If the design is congested:**

- **Modify *pnet* options**
  See 2-65
- **Modify/add standard cell placement constraints (e.g. blockage, utilization %)**
  See 2-19 through 2-21 and 2-36 though 2-39
- **Perform `place_opt` with additional congestion-driven algorithms**

# Modify the Floorplan




**If the design is still congested you will need to modify the floorplan:**

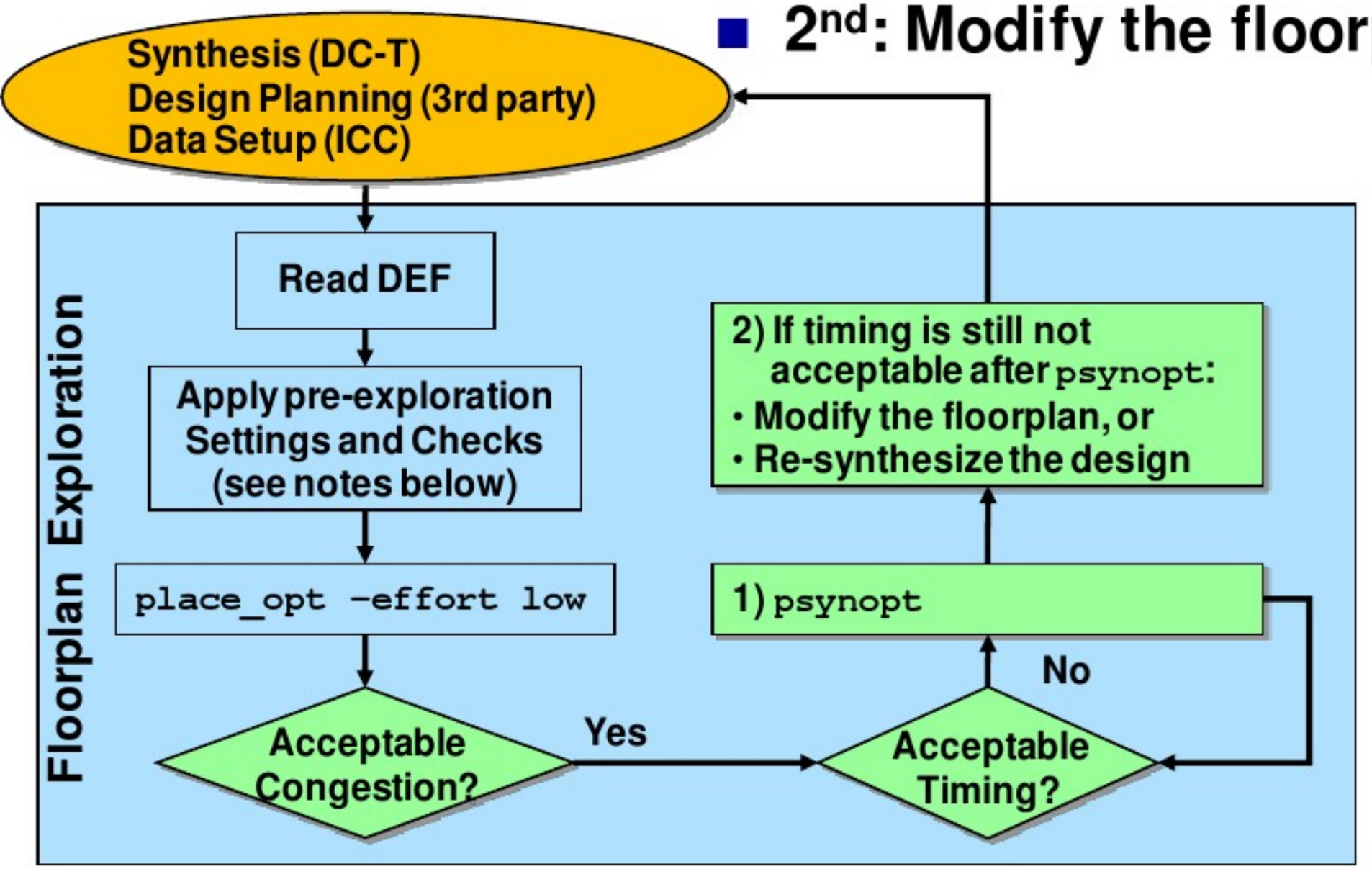
- **Macro placement**
- **Pad/port placement**
- **Core size or shape**
- **Power grid structure**

**Load the original cell from *Data Setup* before loading the new *DEF* file**

# Analyze and Fix Timing

**Once congestion is OK, analyze timing and fix if the worst violation is larger than 15-20% of the required path delay:**

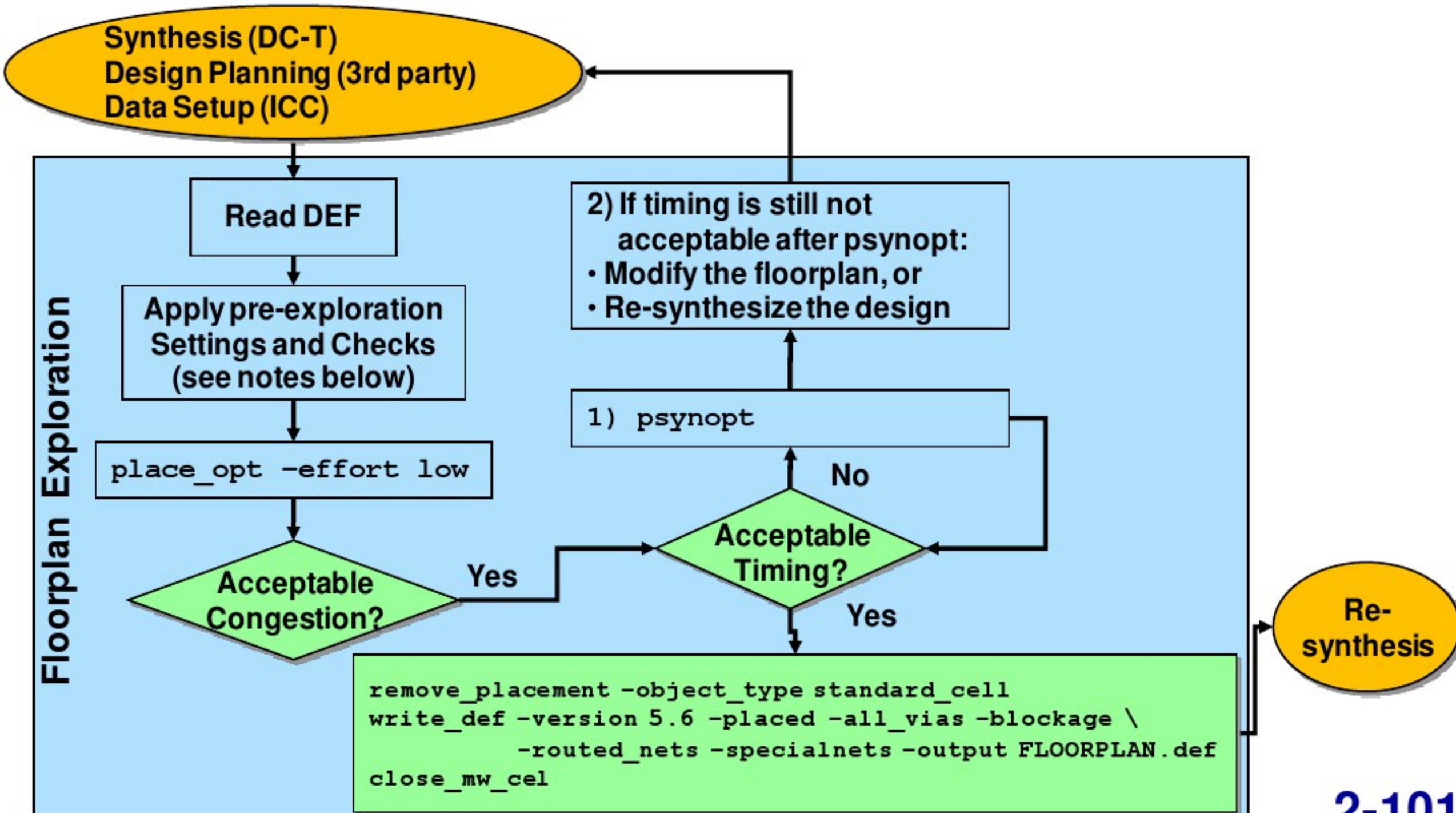
- **1st: Incremental logic optimization**
- **2nd: Modify the floorplan or netlist**



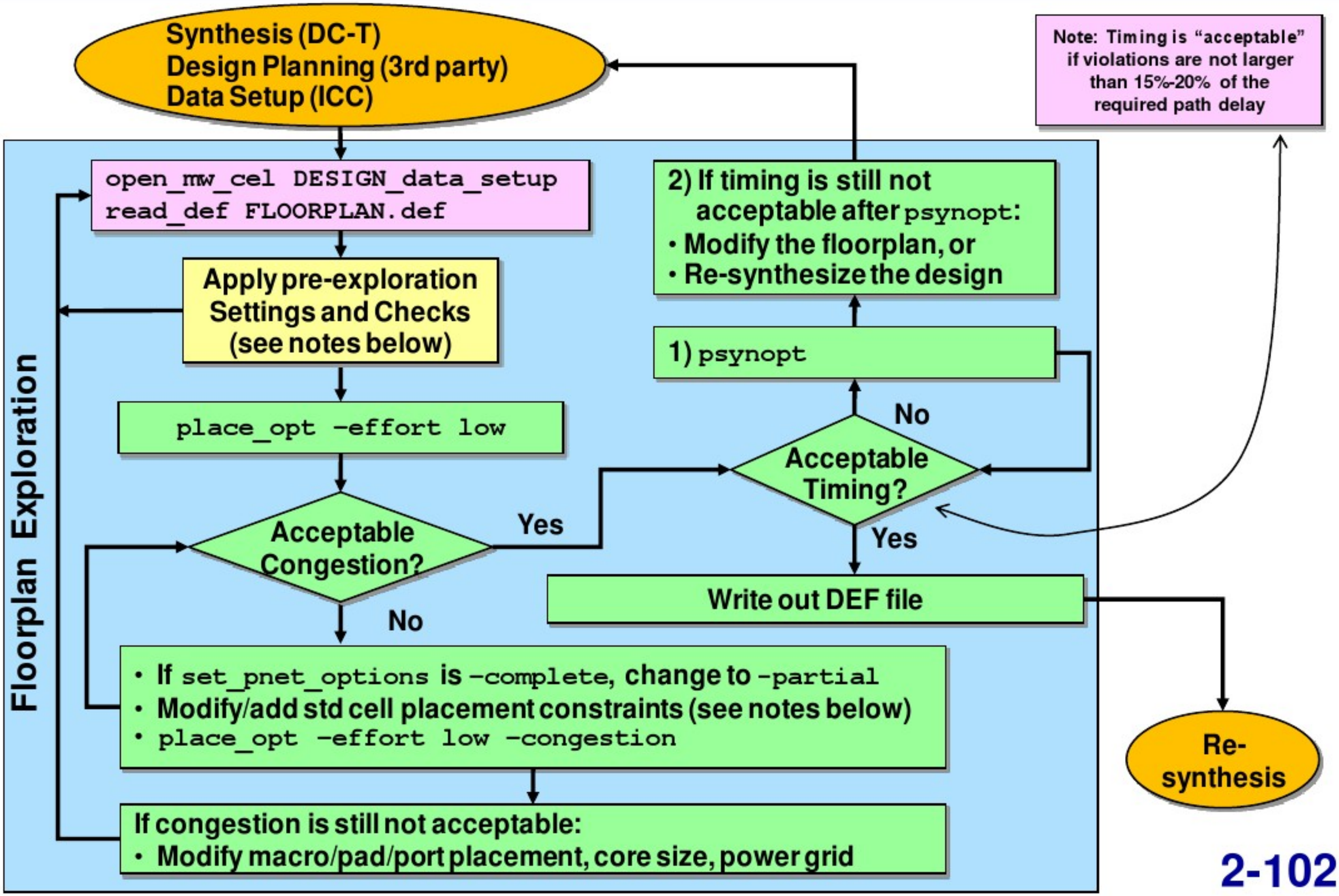
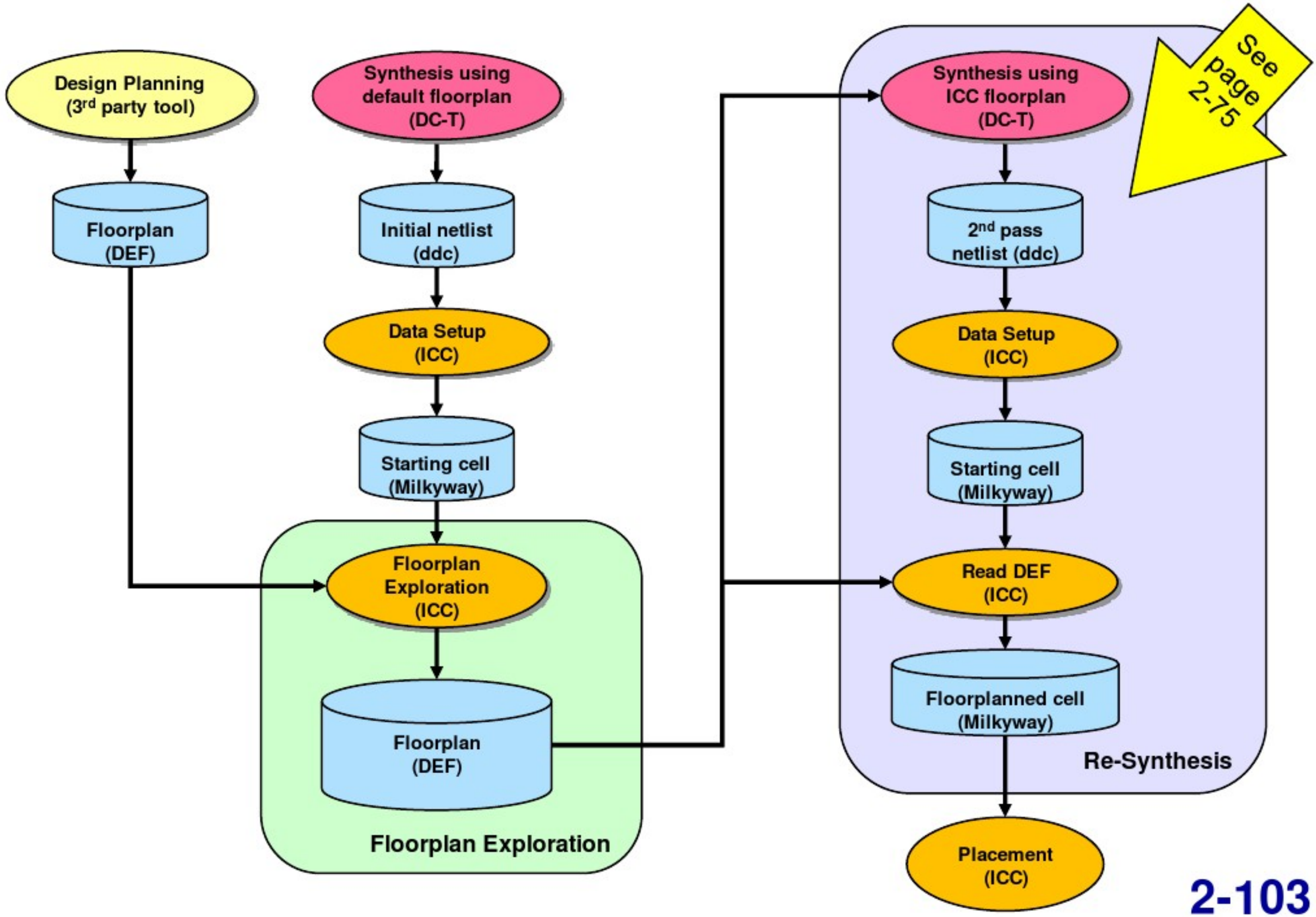

Note: Timing is "acceptable" if violations are not larger than 15%-20% of the required path delay

# Write Out the Floorplan → DEF File

**Once timing is OK, the floorplan is ready for re-synthesis and placement → Write out the floorplan file as *DEF***



# Summary: Floorplan Exploration



# Summary: 3rd Party Design Planning Flow