

high_level overview:

Q: xxx_opt命令是否可以分开运行, 观察运行结果? (mega commands拆分)

A: 商业工具(ICC, Innovus)等不会对用户公开xxx_opt这种mega commands拆分用法. 拆分command更多的是用于RD debug issue等.

Q: filler cell区别, pad filler, decap, filler

A: Cell Filler就是加在stdCell之间; I/O Filler是加在I/O PAD之间, 保证一定顺序的连接关系的; Metal Fill, 也叫dummy, 是区别于前两种的一种金属填充, 一般是在绕线之后为了满足metal density的要求而添加的。decap是一种特殊的filler cell, 主要用于去耦, 可以降低IR.

Q: 为什么先修setup, hold放在后续流程修复? (setup难, hold易?)

A:

Setup: 违反setup时间会导致数据未能正确锁存, 从而造成时序错误。

Hold: 违反hold时间会导致数据在时钟边沿后立即变化, 从而造成数据冲突。

所以由此可见, Setup违例通常影响功能正确性。如果不修复setup违例, 电路在特定时钟频率下可能无法正确工作。

如果先修复hold, 再修复setup会带来什么影响?

修复setup违例通常需要增加信号路径的延迟, 如通过插入缓冲器、重新布局布线等方法。这些调整会直接影响同一信号路径的hold时间。如果先修复hold违例, 再修复setup违例, 修复setup违例的操作可能会再次引入新的hold违例, 导致重复劳动。

design planning:

Q: AI driven design planning, 客户对于这个功能看法?

A: N/A

Q: AG fp是AE做的还是客户通过第三方工具做的fp?

A: 以WSH为例, AG走的是defin流程, 第三方工具(Innovus)做好fp并输出def, 给到AG来做placement及之后的步骤.

placement:

Q: place会用到一些solver, GIGA的place阶段是否也有对应的solution?

A: AG用到共轭梯度法来作数值优化的, EMIR用该算法来求解线性方程组(from 佳敏)

<http://10.30.200.21:8088/issues/30203>

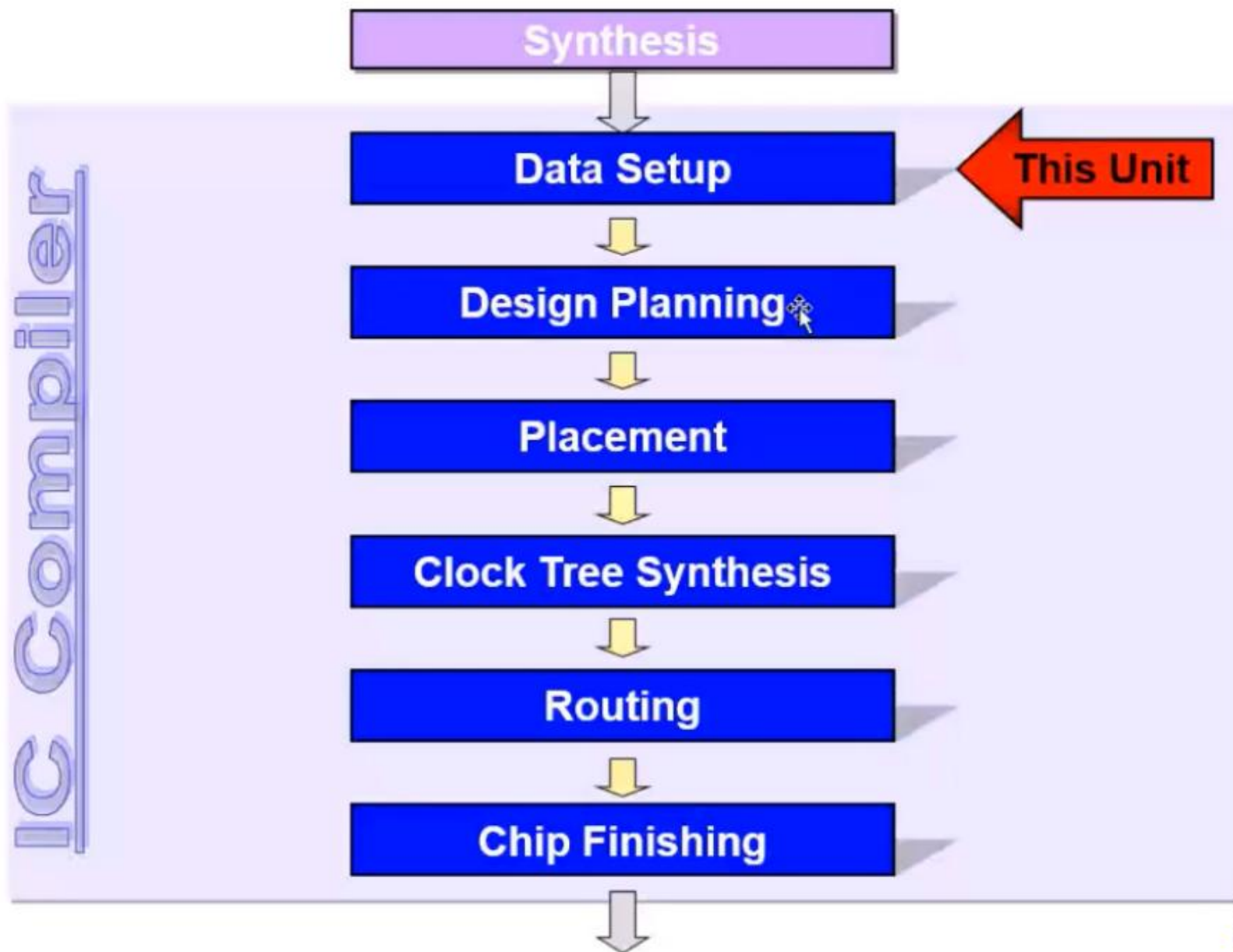
Q: EMIR IR violation客户会通过怎样的方式进行fix (place阶段的小方法?)

A: IC设计流程中EMIR工具进行signoff时IR和EM violation结果一般来说会以文本等方式反馈给PR流程进行优化, 因为主流EMIR工具与主流APR工具不互通, 无法做到无缝衔接.

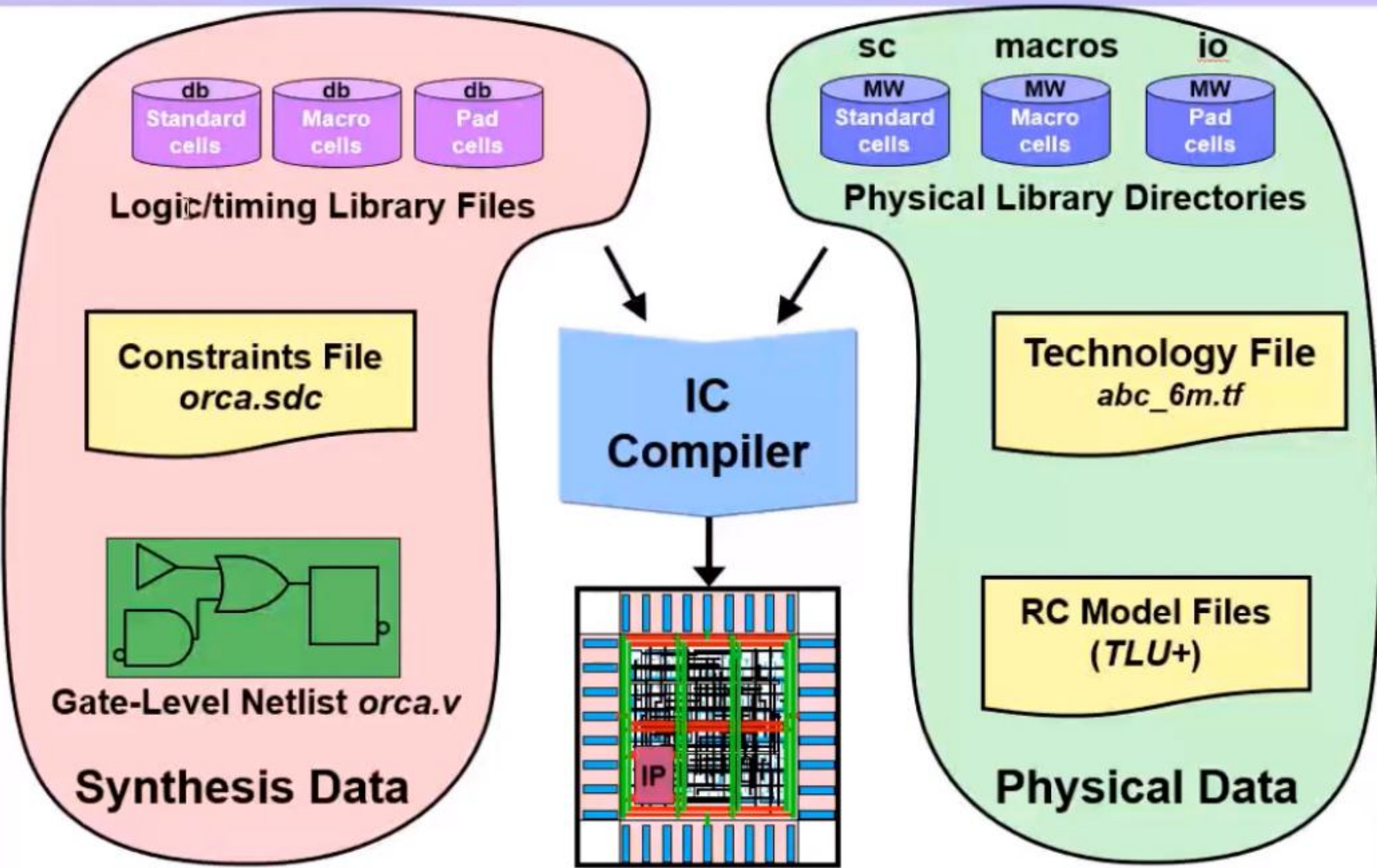
C家Innovus与Voltus database互通, Voltus可以生成供Innovus读入的database.

fix IR方法, 主要以IR结果进行判定, 如果电源比较weak, 可以添加相应的电源; 如果从heatmap来看, 某条row上IR drop值较大, cell比较密集, 就会对cell进行上下挪动, 以减小IR drop的发生.

General IC Compiler Flow



Data Setup



Placed, Routed & Optimized Layout with Clock Trees

Logical Libraries

- Provide timing and functionality information for all standard cells (and, or, flipflop, ...)
- Provide timing information for hard macros (IP, ROM, RAM, ...)
- Define drive/load design rules:
 - Max fanout
 - Max transition
 - Max/Min capacitance
- Are usually the same ones used by Design Compiler during synthesis
- Are specified with variables:
 - target_library
 - link_library



Physical Reference Libraries



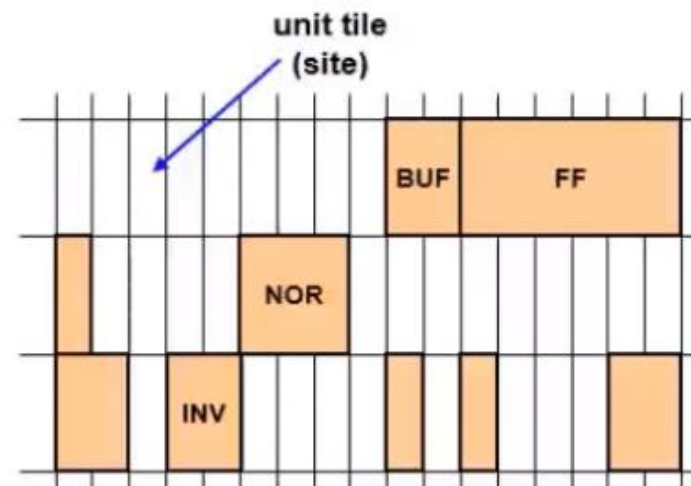
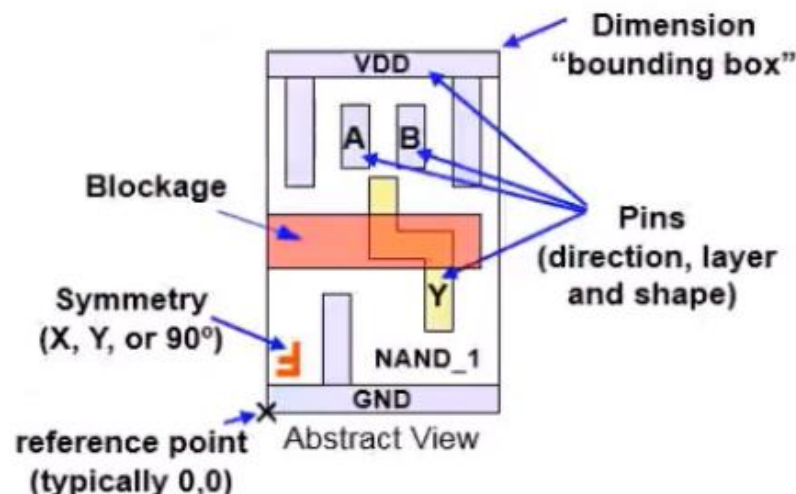
- Contain physical information of *standard*, *macro* and *pad* cells, necessary for placement and routing

- Define placement *unit tile*

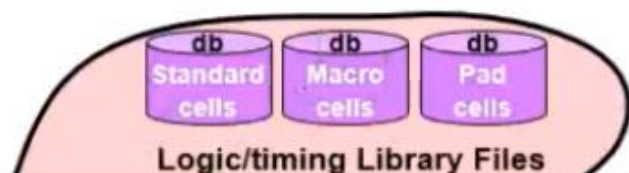
- Height of placement rows
- Minimum width resolution
- Preferred routing directions
- Pitch of routing tracks
- ...

- Are specified with the command:

- `create_mw_lib -mw_reference_library ...`



1. Specify the Logical Libraries



icc -gcl

.synopsys_dc.setup

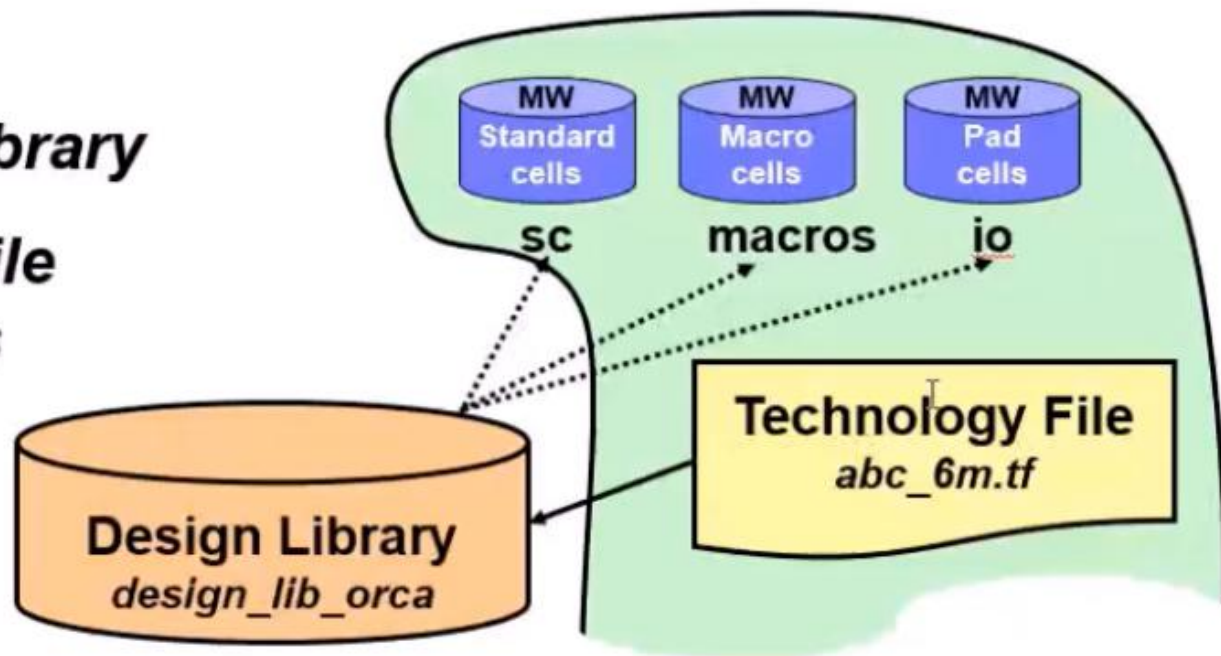
```
lappend search_path [glob ./libs/*/LM]
set_app_var target_library "sc_max.db"
set_app_var link_library "* sc_max.db io_max.db \
                           macros_max.db"
set_min_library sc_max.db -min_version sc_min.db
set_min_library io_max.db -min_version io_min.db
set_min_library macros_max.db -min_version macros_min.db
set_app_var symbol_library "sc.sdb io.sdb macros.sdb"
```

These settings can be re-applied in each new IC Compiler session, or more conveniently, entered once in the .synopsys_dc.setup file, which is automatically read by the tool when ICC is invoked

TCL: glob returns files/directories that match the specified pattern

2. Create a “Container”: The *Design Library*

- Create a *design library*
- Specify the *tech file* and *reference libs*



```
create_mw_lib design_lib_orca -open \  
-technology ./libs/abc_6m.tf \  
-mw_reference_library \  
"./libs/sc ./libs/macros ./libs/io"
```

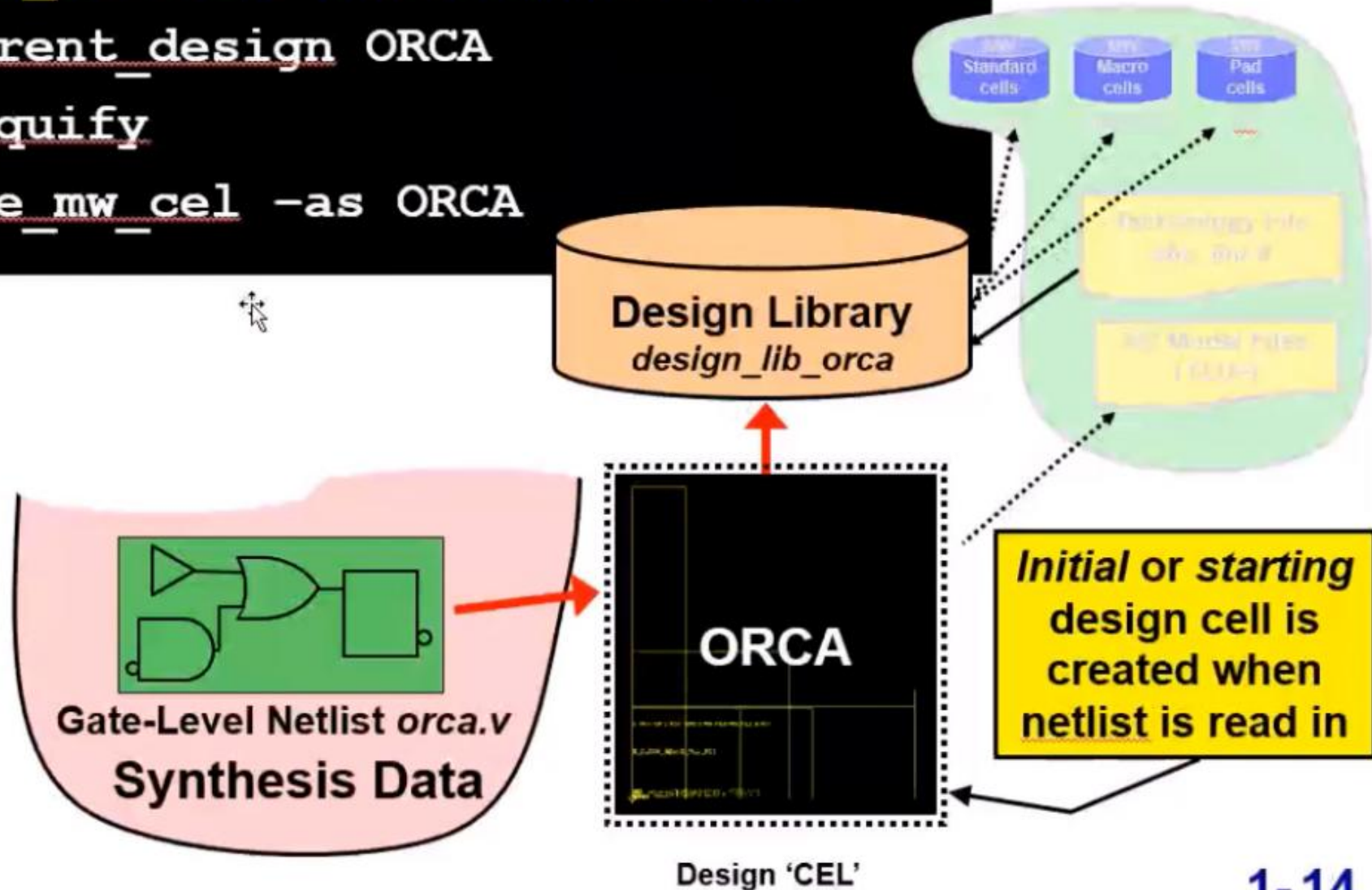

3a. Read the Netlist and Create a Design *CEL*

```
read_verilog ./netlist/orca.v
```

```
current_design ORCA
```

```
uniquify
```

```
save_mw_cel -as ORCA
```



- The *technology file* is unique to each technology
- Contains metal layer technology parameters:
 - Number and name designations for each layer/via
 - Physical and electrical characteristics of each layer/via
 - Design rules for each layer/Via (Minimum wire widths and wire-to-wire spacing, etc.)
 - Units and precision for electrical units
 - Colors and patterns of layers for display
 - ...

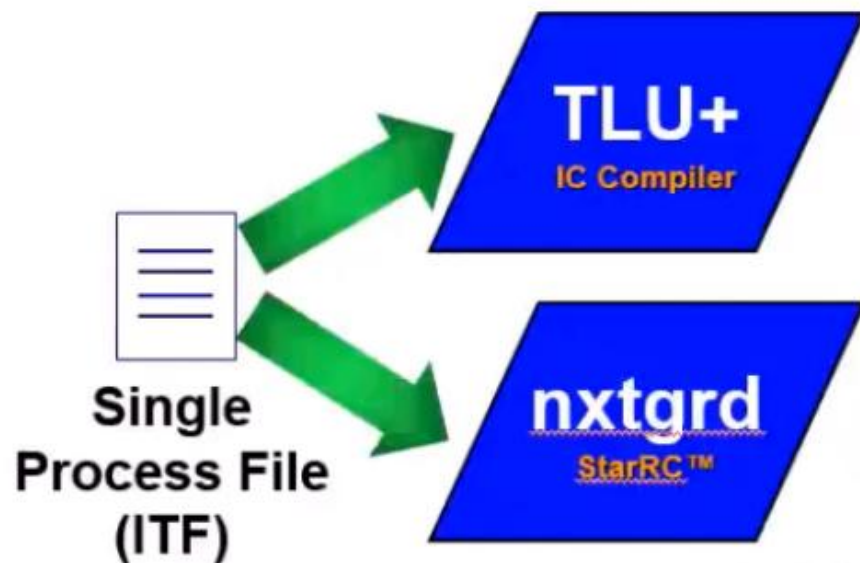
TLU+ Models

- IC Compiler calculates interconnect C and R values using net geometry and the TLU+ look-up tables
- Models UDSM process effects

UDSM Process Effects

- Conformal Dielectric
- Metal Fill
- Shallow Trench Isolation
- Copper Dishing:
 - Density Analysis
 - Width/Spacing
- Trapezoid Conductor

- Some vendors provide only an *ITF* process file
- User must then generate *TLU+* from *ITF* (see below)



5a. Check the Libraries

- The check_library command reports library inconsistencies, for example:
 - Between logic (link_library) and physical libraries:
 - ◆ Missing cells
 - ◆ Missing or mismatched pins
 - Within physical libraries:
 - ◆ Missing CEL (layout) or FRAM (abstract) view cells
 - ◆ Duplicate cell name in multiple reference libraries
- The check_tlu_plus_files command performs a sanity check on the TLU+ files and settings

```
set_check_library_options -all  
check_library  
check_tlu_plus_files
```

5b. Verify Logical Libraries Are Loaded

Ensure that all the required logical libraries (specified by `set_app_var link_library`) have been loaded

`list_libs`

```
icc_shell> list_libs
```

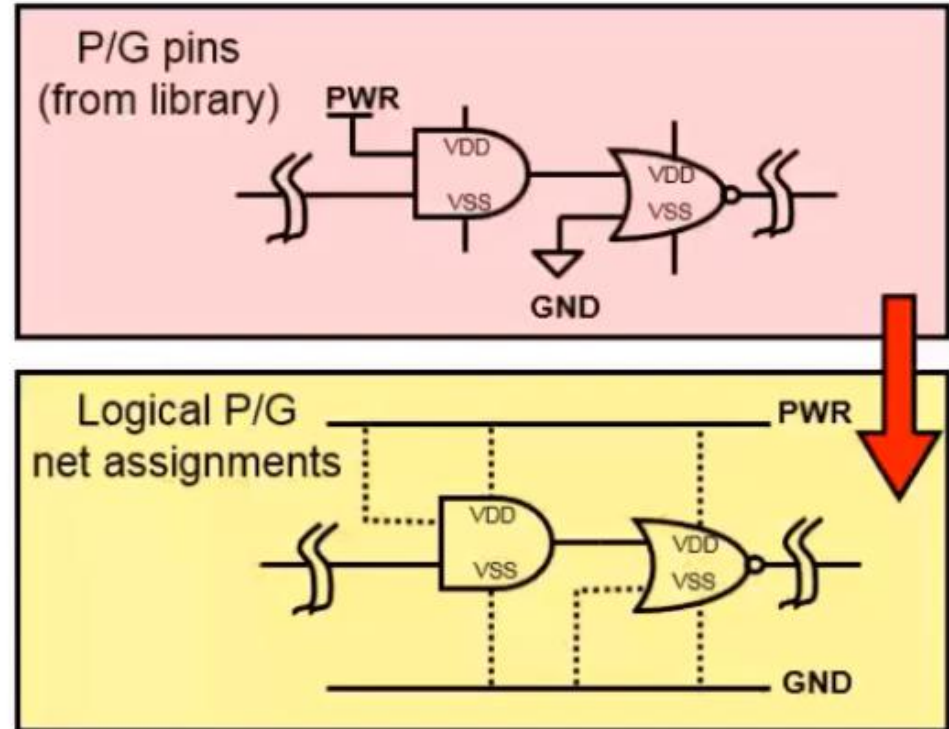
Logical Libraries:

Library	File	Path
M <code>std_cells_max</code>	<code>sc_max.db</code>	<code>/projects/XYZ_design/libs/sc/LM</code>
m <code>std_cells_min</code>	<code>sc_min.db</code>	<code>/projects/XYZ_design/libs/sc/LM</code>
M <code>io_pads_max</code>	<code>io_max.db</code>	<code>/projects/XYZ_design/libs/io/LM</code>
m <code>io_pads_min</code>	<code>io_min.db</code>	<code>/projects/XYZ_design/libs/io/LM</code>
M <code>macros_max</code>	<code>macros_max.db</code>	<code>/projects/XYZ_design/libs/macros/LM</code>
m <code>macros_min</code>	<code>macros_max.db</code>	<code>/projects/XYZ_design/libs/macros/LM</code>
<code>gtech</code>	<code>gtech.db</code>	<code>/global/apps3/icc_2010.03-SP2/libraries/syn</code>
<code>standard.sldb</code>	<code>standard.sldb</code>	<code>/global/apps3/icc_2010.03-SP2/libraries/syn</code>

The `gtech` and `standard` libraries are generic libraries that are loaded by default – used during synthesis

6. Define Logical Power/Ground Connections

- Define P/G net names and create “logical connections” between P/G pins and P/G nets
- Create connections between tie-high/low inputs and P/G nets



```
derive_pg_connection -power_net PWR -power_pin VDD \  
-ground_net GND -ground_pin VSS  
derive_pg_connection -power_net PWR -ground_net GND \  
-tie  
check_mv_design -power_nets
```

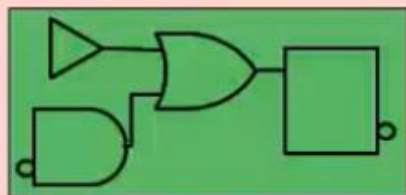
7. Apply and Check Timing Constraints

```
read_sdc ./cons/orca.sdc  
check_timing  
report_timing_requirements  
report_disable_timing  
report_case_analysis
```

Only needed if reading in
an ASCII netlist

Constraints File *orca.sdc*

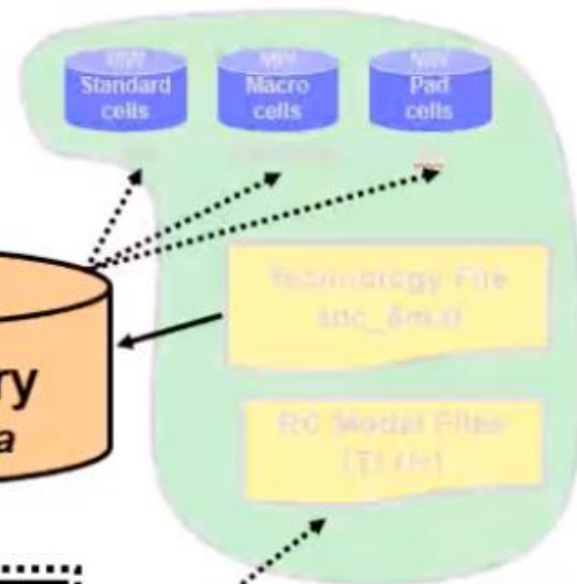
```
create_clock -period 10 .....  
set_input_delay -max 1.2 ...  
set_output_delay -max 2.5 ...
```



Gate-Level Netlist *orca.v*



Design 'CEL'



8. Ensure Proper Modeling of Clock Tree

- Ensure your SDC constraints to model estimates of clock skew, latency and transition times for all clocks

report_clock -skew

Object	Rise Delay	Fall Delay	Min Delay	Max Delay	Min Delay	Max Delay	Uncertainty Plus	Uncertainty Minus
SYS_2x_CLK	0.80	0.80	0.40	0.80	0.40	0.80	0.10	0.20
SDRAM_CLK	-	-	-	-	-	-	0.10	0.15

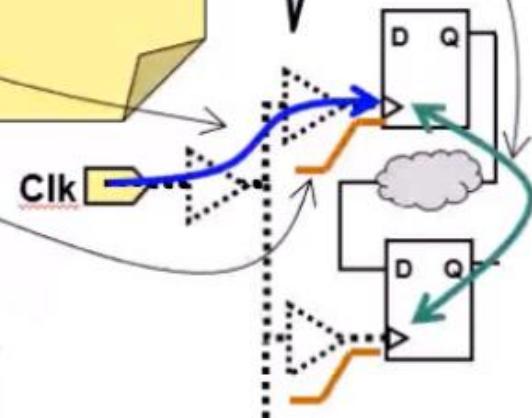
Object	Max Transition Rise	Max Transition Fall	Min Transition Rise	Min Transition Fall
SYS_2x_CLK	0.07	0.07	-	-
SDRAM_CLK	0.07	0.07	-	-

Pre-CTS
clock
modeling

- Ensure no clocks are defined as “propagated” clocks

report_clock

Clock	Period	Waveform	Attrs	Sources
SDRAM_CLK	7.50	{0 3.75}	p	{sdrclk}
SYS_2x_CLK	4.00	{0 2}		{sys_2x_clk}



9. Apply Timing and Optimization Controls

- Timing and optimization in IC Compiler is controlled by many variables and commands, for example:

```
set_app_var timing_enable_multiple_clocks_per_reg true  
set_fix_multiple_port_nets -all -buffer_constants  
group_path -name INPUTS -from [all_inputs]
```

- These variables and commands can affect design planning, placement, CTS and routing
- Therefore, they should be applied prior to design planning (*and re-applied after re-starting IC Compiler*)¹

```
source tim_opt_ctrl.tcl
```

- Learning all the available variables and commands can be a challenge – the GUI provides help!

10. Perform a 'Timing Sanity Check'

- Before starting placement it is important to ensure that the design is not over-constrained
 - Constraints should match the design's specification
- Report 'ZIC' timing before placement
 - Check for unrealistic or incorrect constraints
 - Investigate large zero-interconnect timing violations



```
set_zero_interconnect_delay_mode true
```

```
Warning: Timer is in zero interconnect delay mode. (TIM-177)
```

```
report_constraint -all
```

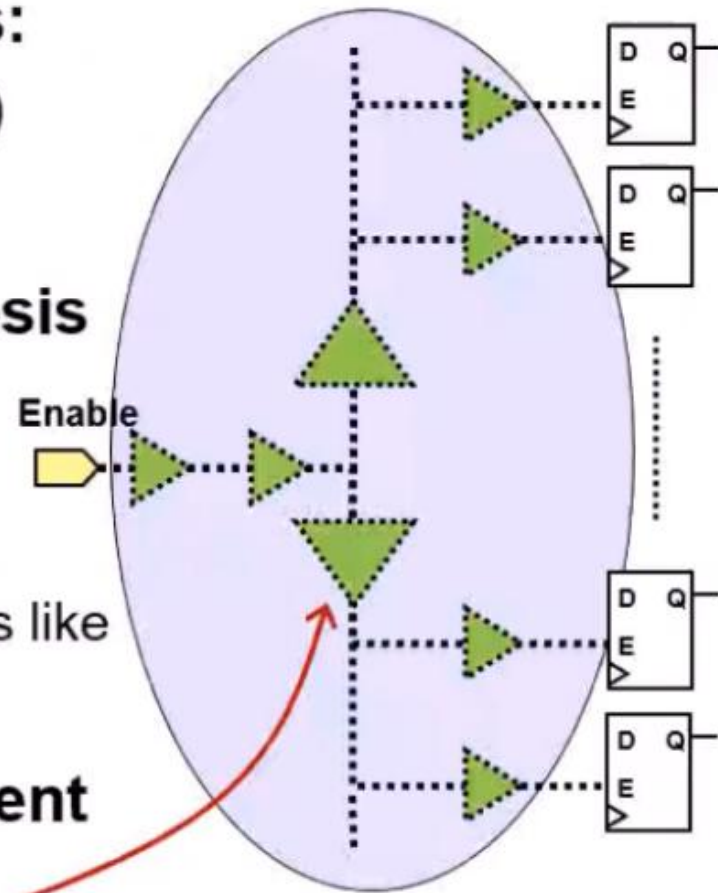
```
report_timing
```

```
set_zero_interconnect_delay_mode false
```

```
Information: Timer is not in zero interconnect delay mode. (TIM-176)
```


11. Remove Unwanted “Ideal Net/Networks”

- Your SDC constraints may contain either of the following commands:
 - `set_ideal_network` (preferred)
 - `set_ideal_net` (obsolete)
- These commands prevent synthesis (Design Compiler) from building buffer trees on specified signals, which is deferred to the physical design phase (typically high fanout nets like *set/reset*, *enable*, *select*, etc.)
- To allow buffering during placement remove the constraints:



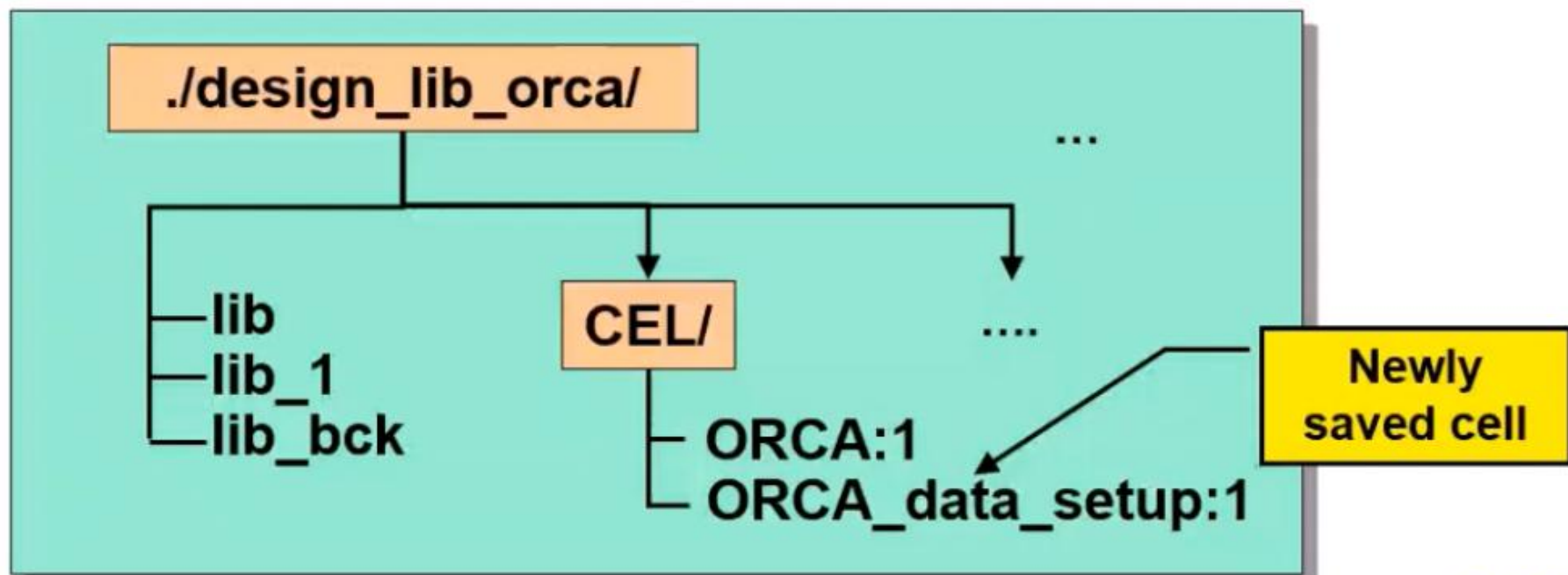
```
remove_ideal_network [get_ports "Enable Select Reset"]
```

12. Save the Design

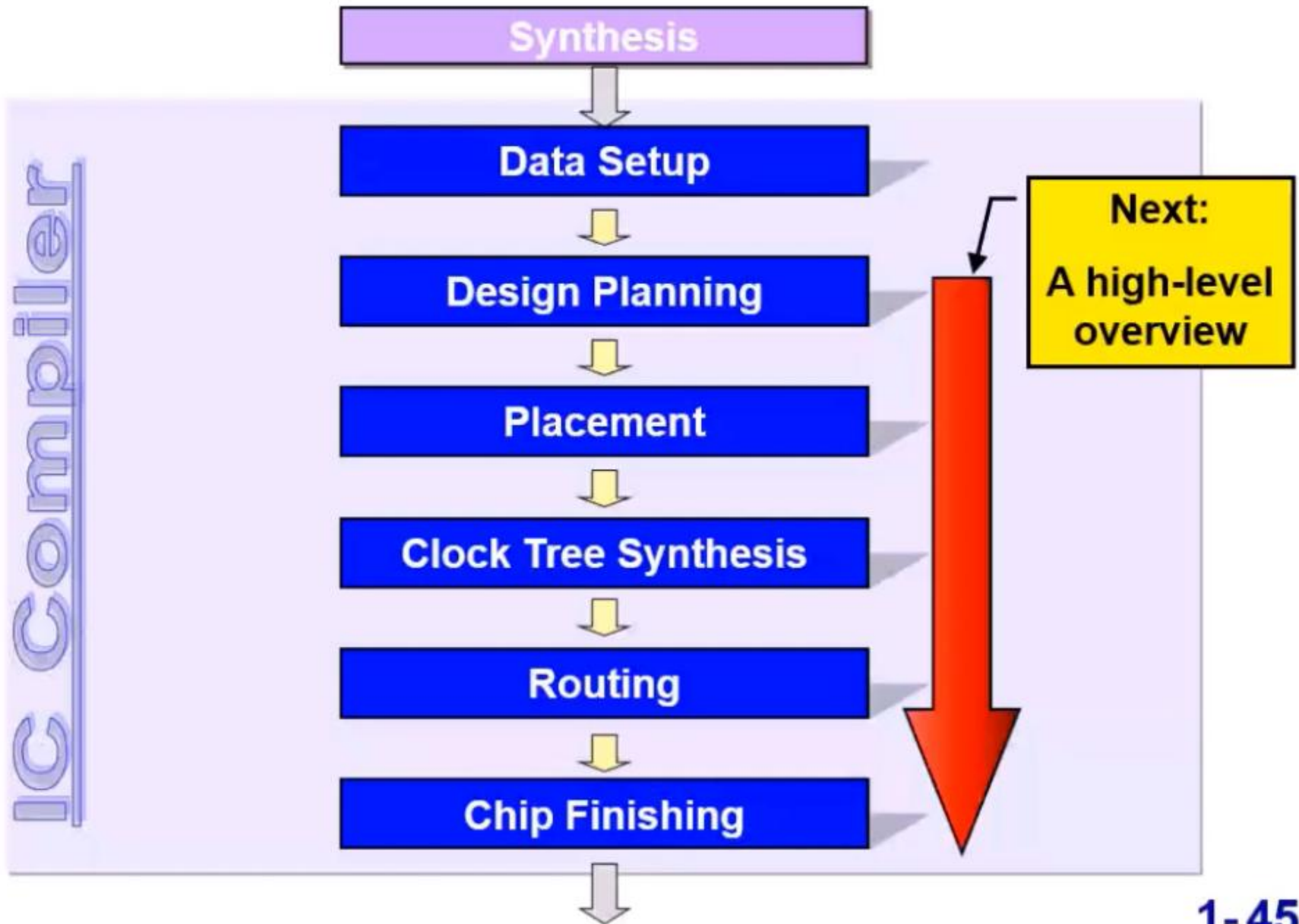
It's good practice to save the design after each key design phase, for example: data setup, design planning, placement, CTS and routing:

```
save_mw_cel -as ORCA_data_setup
```

- Note: The open cell is still the original *ORCA* cell !!



General IC Compiler Flow



Design Planning

Data Setup

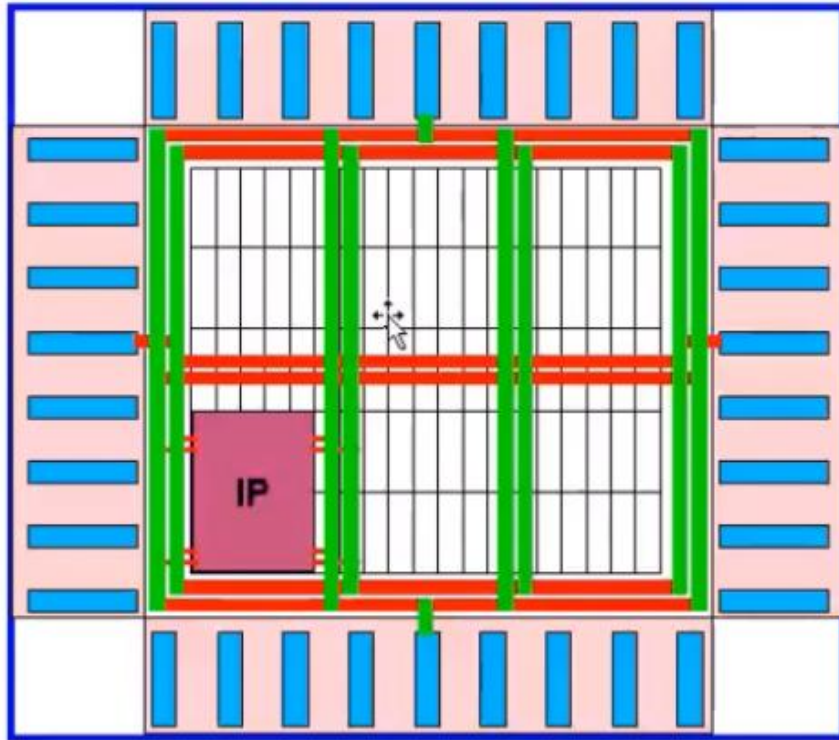
Design Planning

Placement

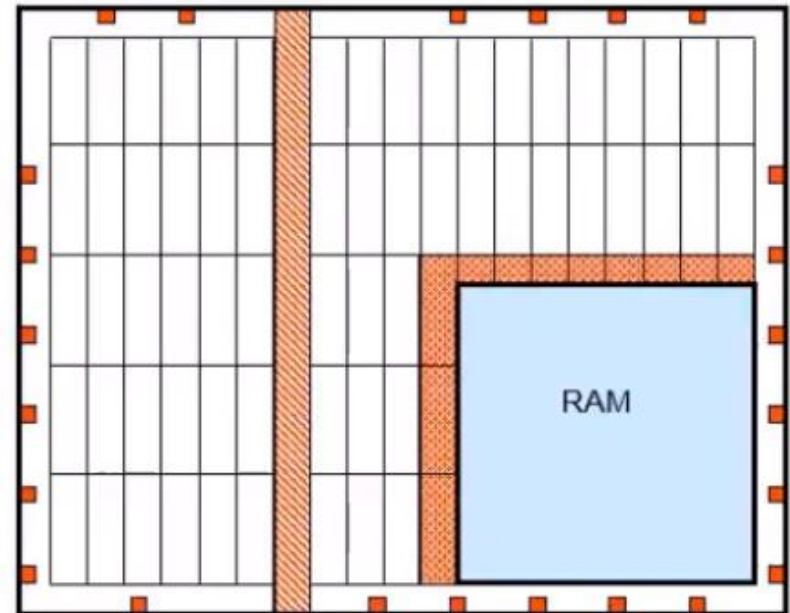
Clock Tree Synthesis

Routing

Chip Finishing



Chip-level Floorplan



Block-level Floorplan

“Design planning” is the iterative design phase of defining a “floorplan”

Placement and Related Optimizations

Data Setup

Design Planning

Placement

Clock Tree Synthesis

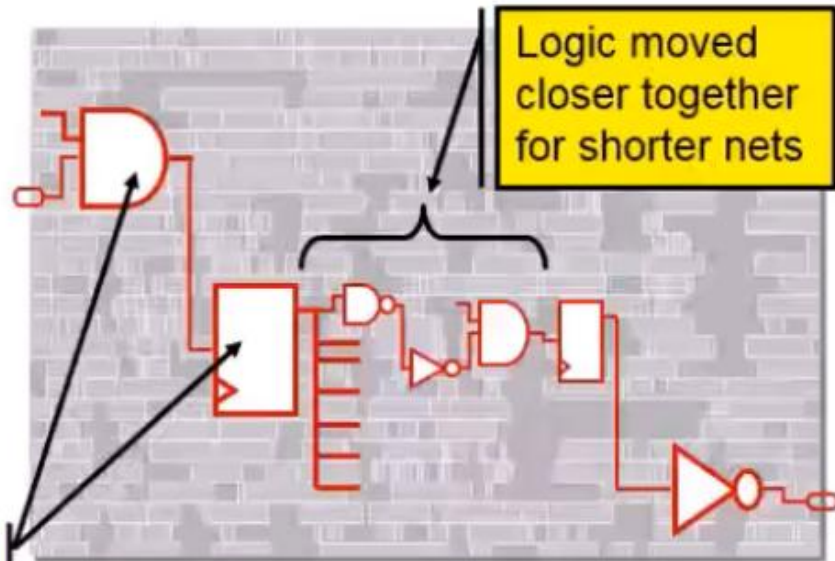
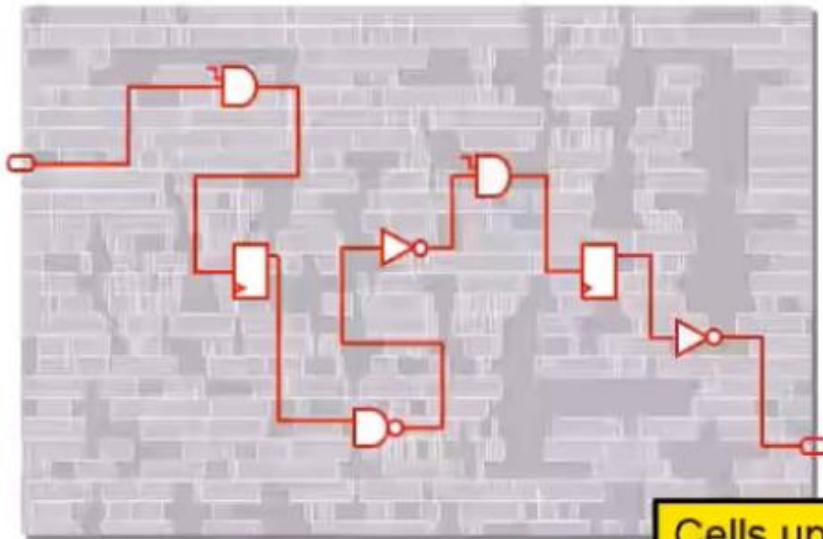
Routing



1. Set placement options

2. Run `place_opt`

- Performs iterative placement and logic optimization
- Objective: Fix timing violations and congestion



Cells upsized
for optimal
drive/speed

Clock Tree Synthesis

Data Setup

Design Planning

Placement

Clock Tree Synthesis

Routing

Chip Finishing

1. Set clock tree options/exceptions

2. Run `clock_opt`

- Builds the clock trees
- Performs incremental logic and placement optimizations
- Runs clock tree optimizations
- Routes the clock nets
- Can fix hold time violations
- Can perform inter-clock balancing



Routing

Data Setup

Design Planning

Placement

Clock Tree Synthesis

Routing

Chip Finishing

1. Set routing options/exceptions
2. Run `route_opt`

Performs

- Global Route
- Track Assignment
- Detailed Route

Concurrently performs

- Logic, placement and routing optimizations
- Objective: Complete routing and meet timing



Chip Finishing

Data Setup Design Planning Placement Clock Tree Synthesis Routing **Chip Finishing**

- Also known as 'Design for Manufacturing'
- Entails:
 - Antenna Fixing
 - Wire Spreading
 - Double Via Insertion
 - Filler Cell Insertion
 - Metal Fill Insertion

**Discussed in the Routing
and Chip Finishing units**

Analyzing the Results (1/2)

After each placement, CTS and routing step you should:

- **Examine the log output for design summaries:**

- Utilization
- Worst Negative Slack (WNS)
- Total Negative Slack (TNS)
- Legality of cell placement
- Cell count and area
- Design rule violations

- **Use `report_qor` to see:**

- WNS/TNS per path group (clock group)
- Other statistics

Analyzing the Results (2/2)

■ Generate more detailed reports

- Show all violating path end points

```
report_constraint -all_violators
```

- Show details of the worst violating setup path

```
report_timing
```

- Report physical design statistics (e.g. utilization)

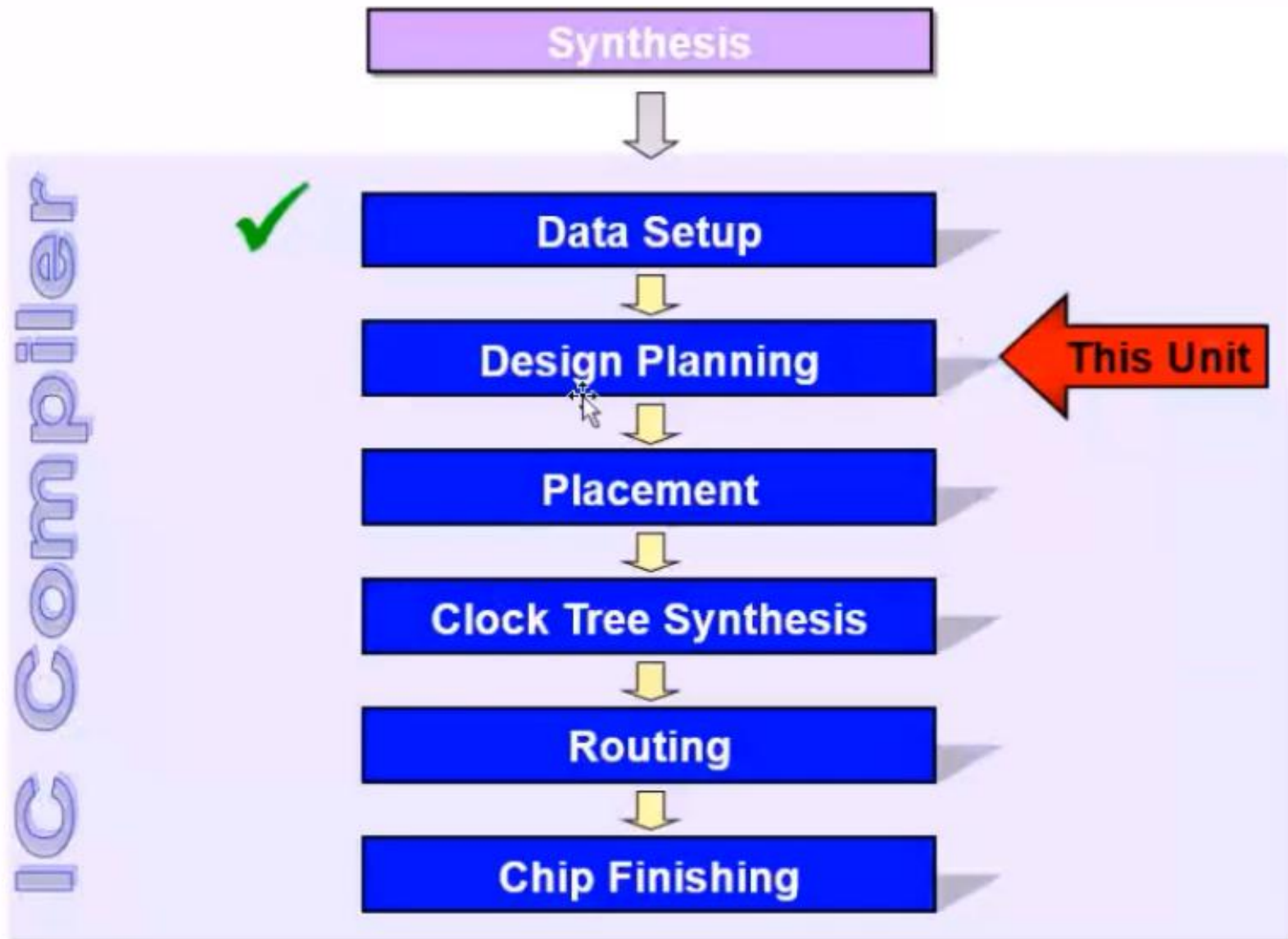
```
report_design -physical
```

- Analyze the congestion

- ◆ Congestion map (GUI)

```
report_congestion
```

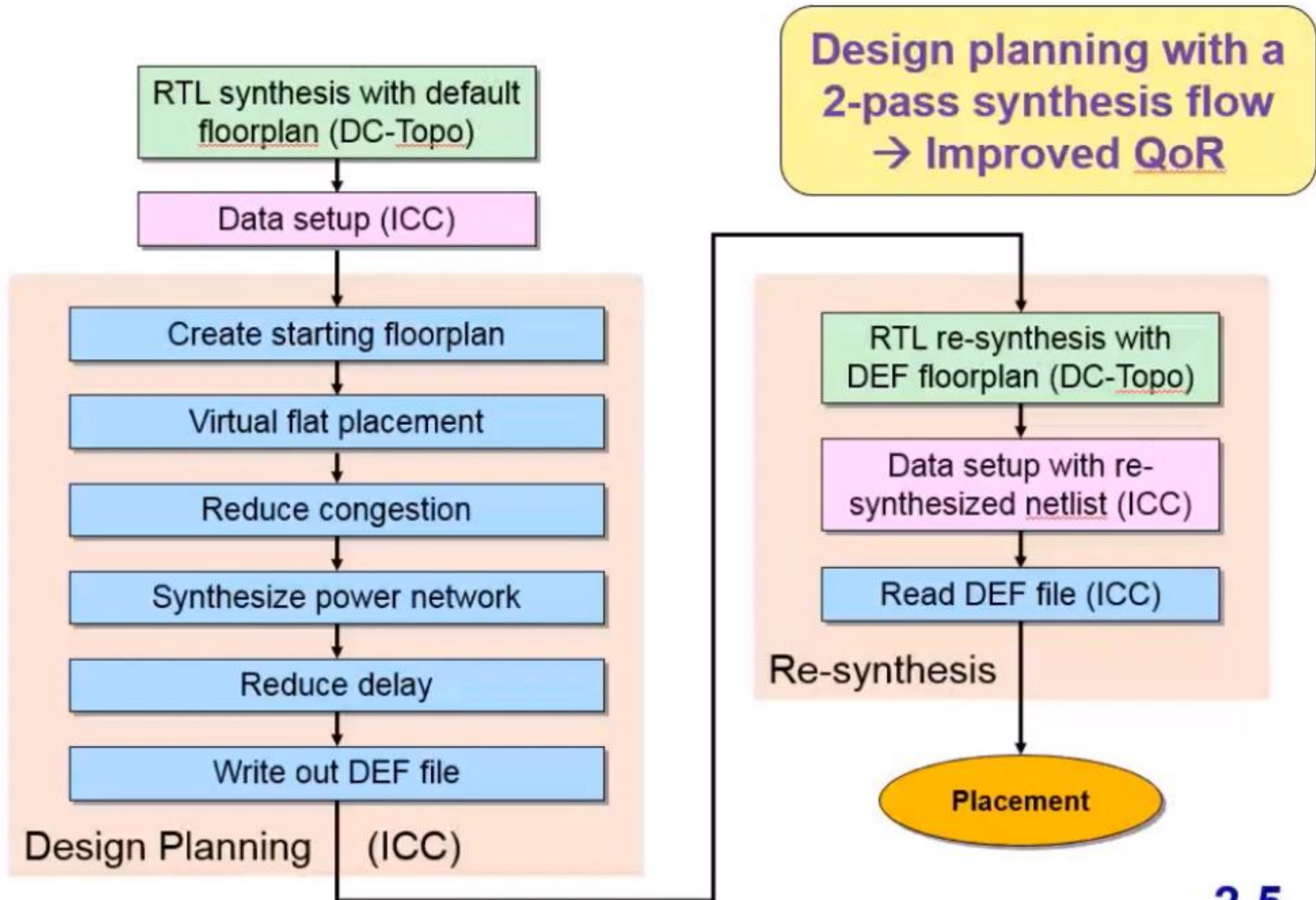
General IC Compiler Flow



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

Starting Floorplan

Placement

Reduce Congestion

PNS

Reduce Delay

Write DEF

DC-T synthesis with
default floorplan

Data setup

Create starting floorplan

Virtual flat placement

Reduce congestion

Synthesize power network

Reduce delay

Write out DEF file

Design Planning

Create physical-only pad cells

Specify pad cell locations

Initialize the floorplan

Insert pad filler cells

Create P/G pad rings

Specify ignored routing layers

Define known macro/std cell placement

Define known power structure

Define known placement blockages

Create Starting Floorplan

- 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

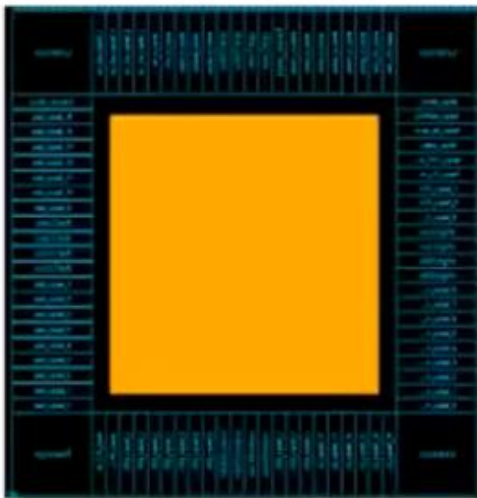
```
create_cell {vss_l vss_r vss_t vss_b} pv0i  
create_cell {vdd_l vdd_r vdd_t vdd_b} pvdi  
create_cell {CornLL CornLR CornTR CornTL} pfrelr
```


Initialize the Floorplan

Starting 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

Control type
☒ Aspect ratio ☐ Width and height ☐ Row number ☐ Boundary

Core utilization: Row/core ratio: Num rows:

Aspect ratio (H/W): Core width: Core height:

☒ Horizontal row ☒ Double back ☐ Start first row ☐ Flip first row

Core to left: Core to right:

Core to bottom: Core to top:

☐ Keep macro place ☐ Keep std cell place ☐ Min pad height ☐ Pad limit

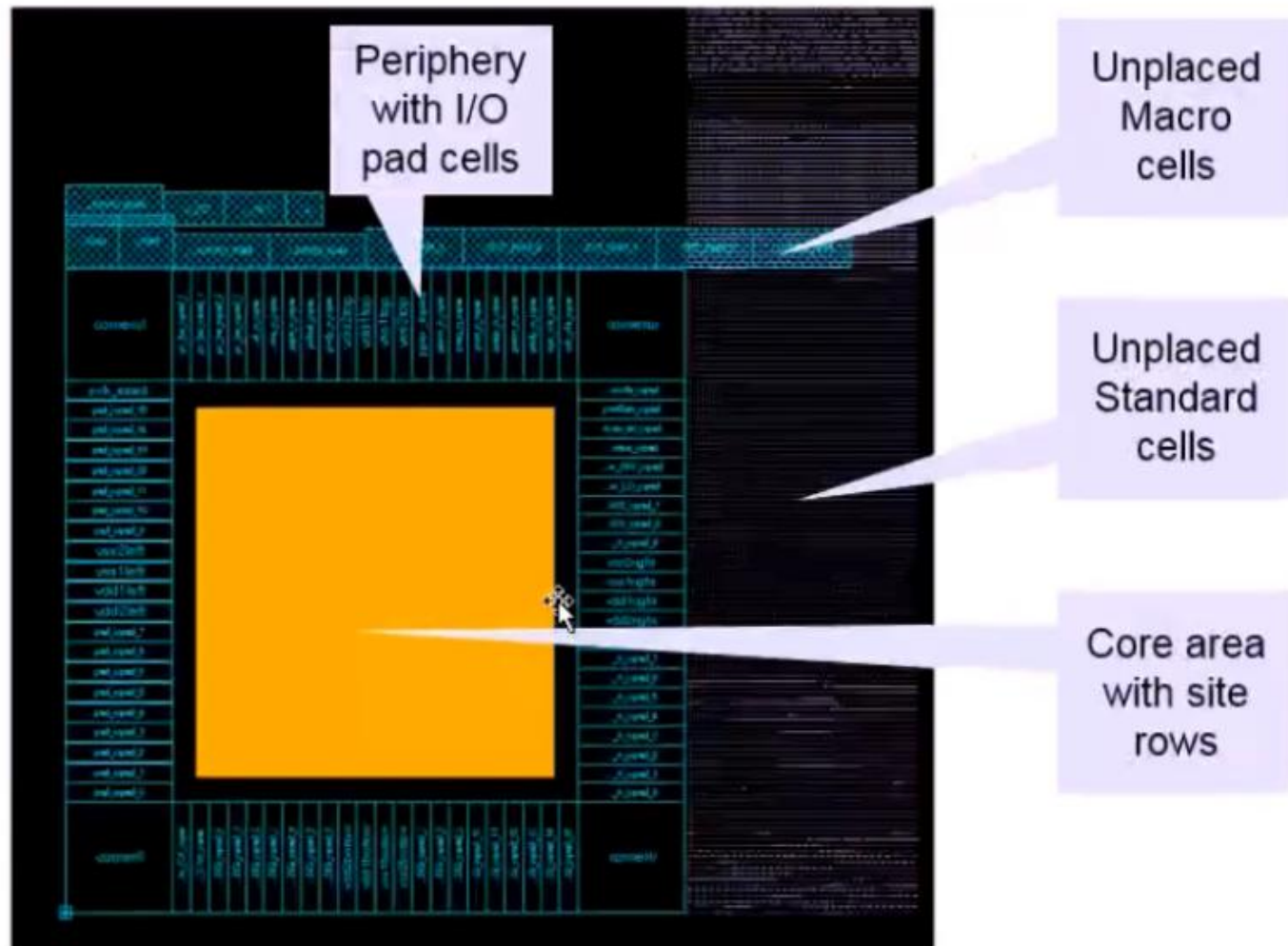
☐ Pin snap ☐ Keep I/O placement

OK Cancel Apply Default ▾

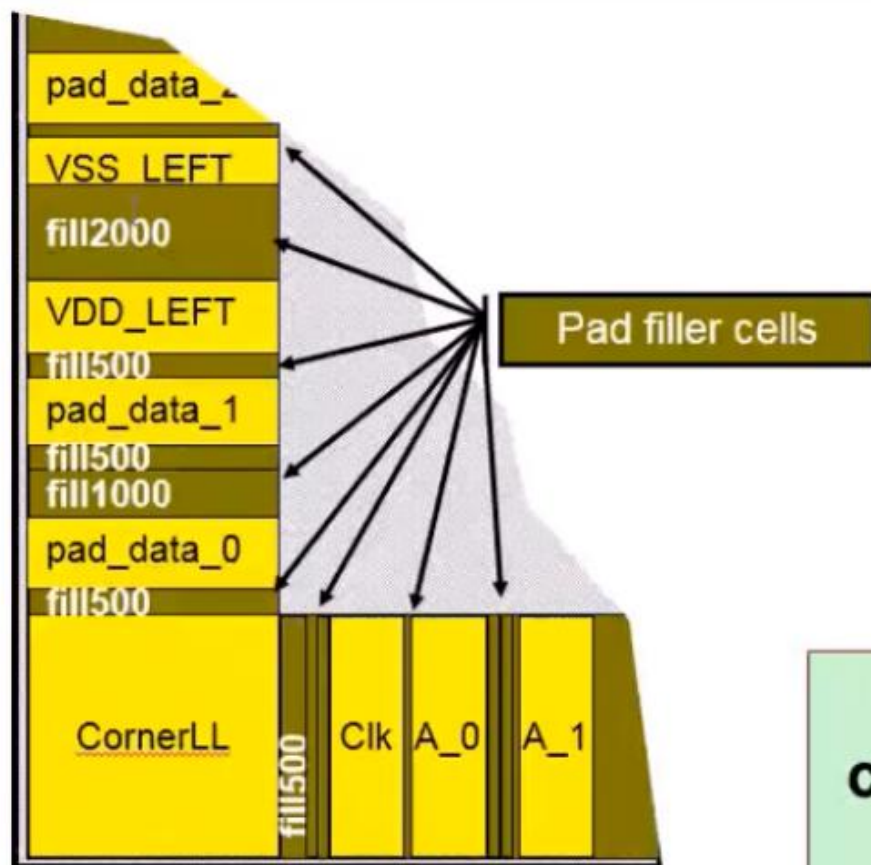
`initialize_floorplan ...`

Floorplan After Initialization

Starting Floorplan



Insert Pad Filler Cells



Starting Floorplan

Insert Pad Filler

Pin/Blockage cells: fill5000 fill2000 fill1000 fill500 fill200 fill100 fill50 fill20

Pin/Blockage overlap cells:

Working area

☒ Whole chip ☐ Use specified bounding box:

Coordinates:

Instance prefix:

Boundary placement

☒ Top ☒ Bottom ☒ Left ☒ Right

Voltage areas:

OK Cancel Apply Default

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

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

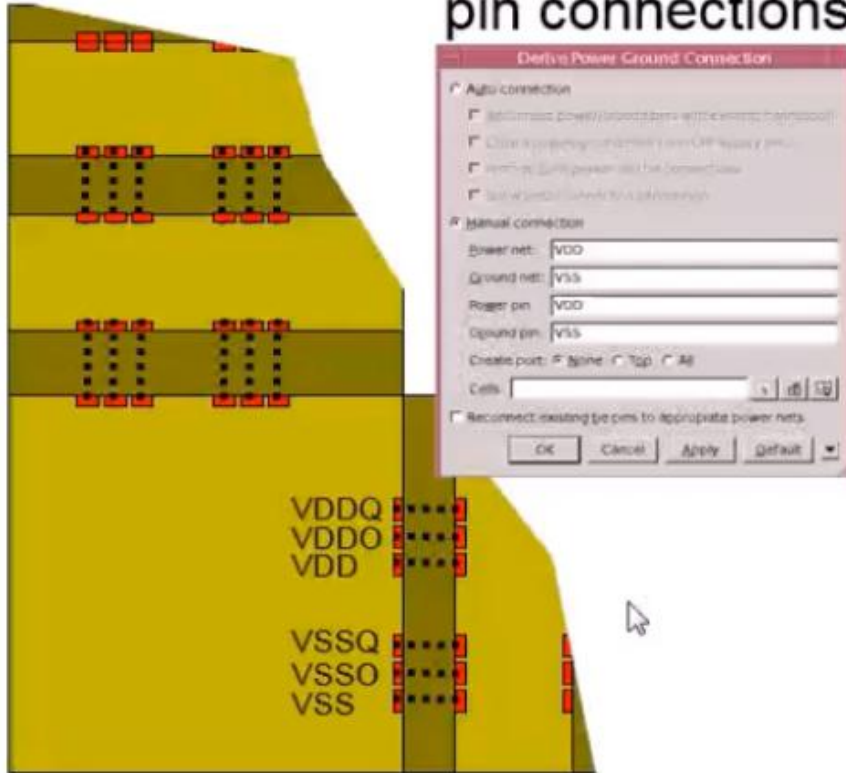


Order of cells is important – list largest to smallest from left to right to use minimum number of cells!

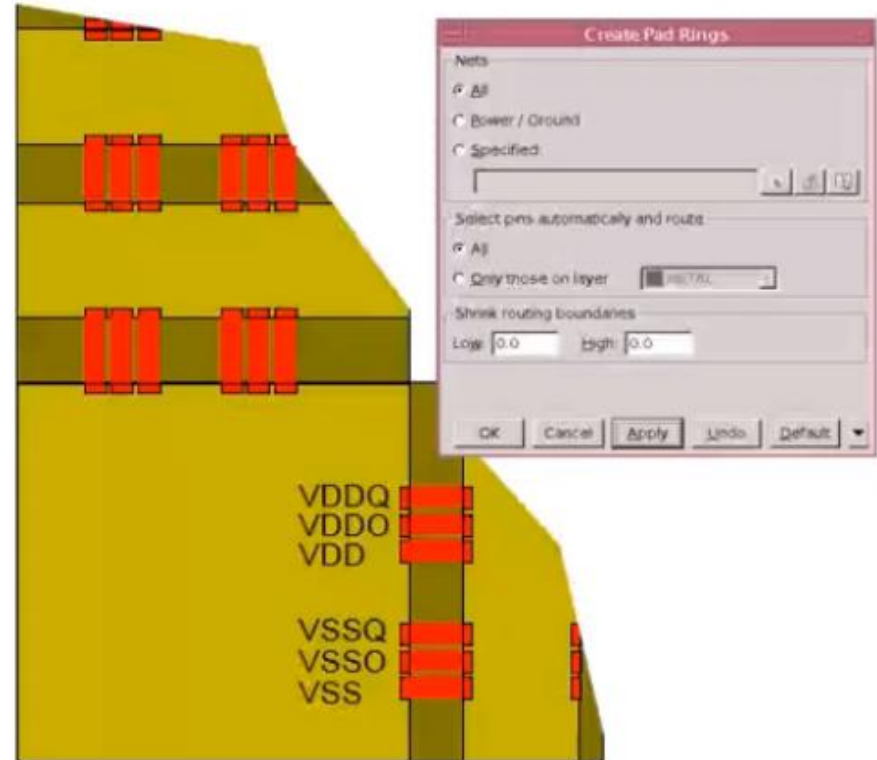
Create P/G Pad Rings

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
```

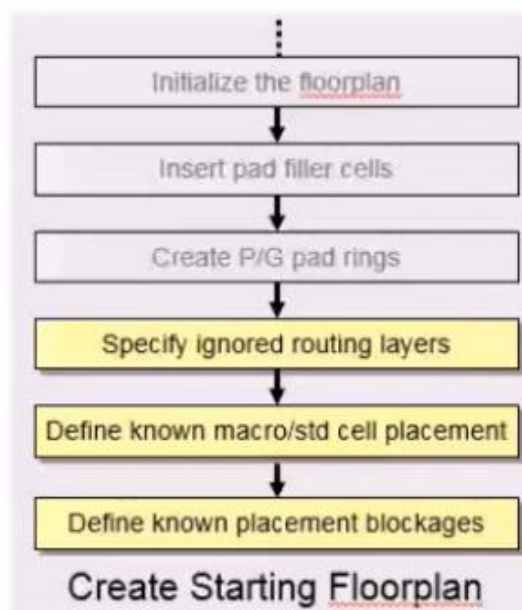
May be needed for P/G
continuity

■ 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



- By default IC Compiler will use all 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
```

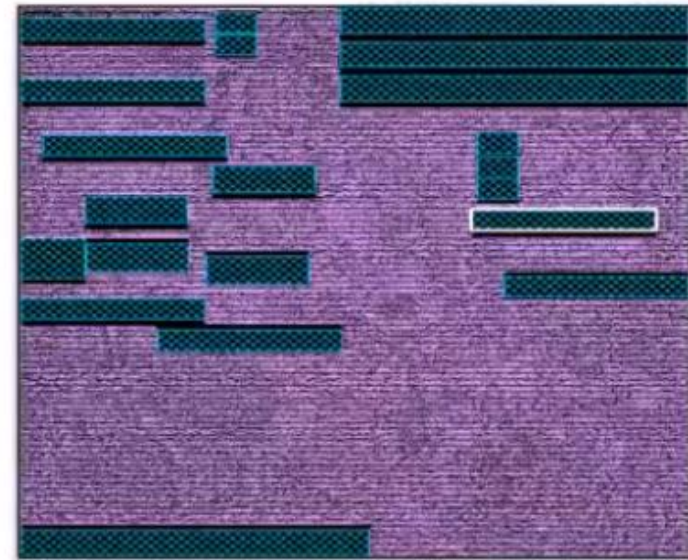
```
report_ignored_layers  
remove_ignored_layers
```


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

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 ever 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

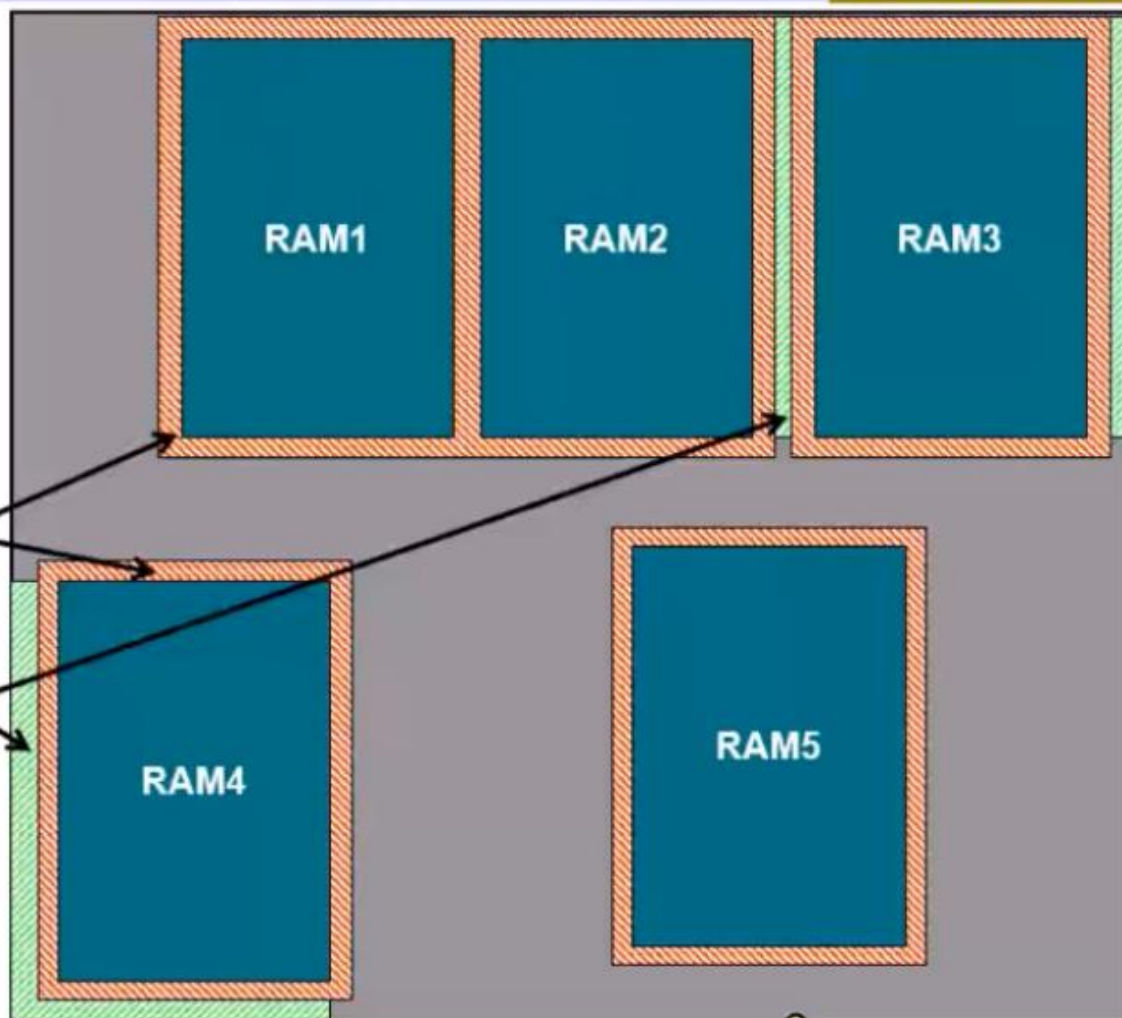
Starting Floorplan

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

Summary: Create a Starting Floorplan

Starting Floorplan

Placement

Reduce Congestion

PNS

Reduce Delay

Write DEF

Data setup

Create physical-only pad cells

Specify pad cell locations

Initialize the floorplan

Insert pad filler cells

Create P/G pad rings

Specify ignored routing layers

Define known macro/std cell placement

Define known placement blockages

```
open_mw_cel DESIGN_data_setup
create_cell ...
set_pad_physical_constraints ...
initialize_floorplan ...
insert_pad_filler ...
derive_pg_connection ...
create_pad_rings ...
set_ignored_layers -max M7
# Manually place macro cells, as
# applicable, and "fix" their placement:
set_dont_touch_placement [all_macro_cells]
set_fp_macro_options ...
set_fp_macro_array ...
set_fp_relative_location ...
set_app_var physopt_hard_keepout_distance <#>
set_app_var placer_soft_keepout_channel_width <#>
set_keepout_margin ...
```

Create Starting Floorplan

Perform Virtual Flat Placement

Starting Floorplan **Placement** Reduce Congestion PNS Reduce Delay Write DEF

DC-T synthesis with
default floorplan

Data setup

✓
Create starting floorplan

➔ Virtual flat placement

Reduce congestion

Synthesize power network

Reduce delay

Write out DEF file

Set placement strategy parameters

Perform virtual flat placement

Virtual Flat Placement

Design Planning

Set Placement Strategy Parameters

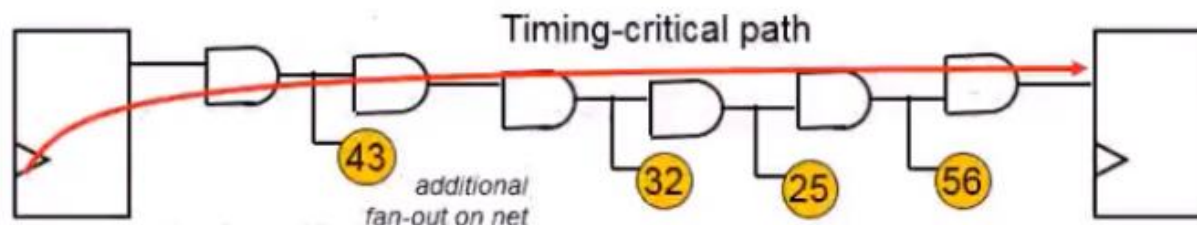
- Placement strategy parameters are used to control the following during `create_fp_placement`:
 - How macros are handled
 - Optimization algorithms and effort
- Consider setting a “sliver size” and turning on “virtual IPO”:

```
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

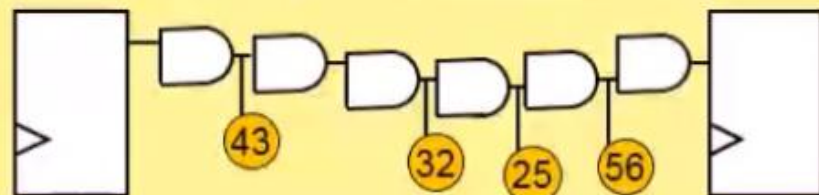
VF Placement with *Virtual IPO* (VIPO)

Placement



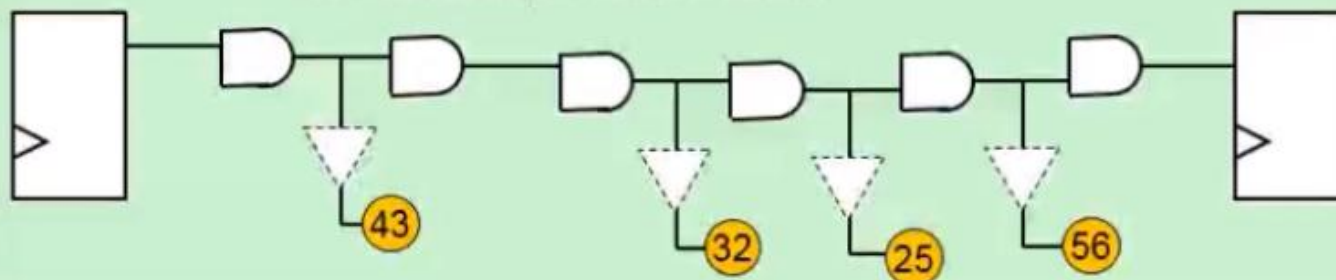
Original netlist contains large fan-outs along the critical path which can contribute significantly to delay

VF Placement without VIPO



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

VF Placement with VIPO



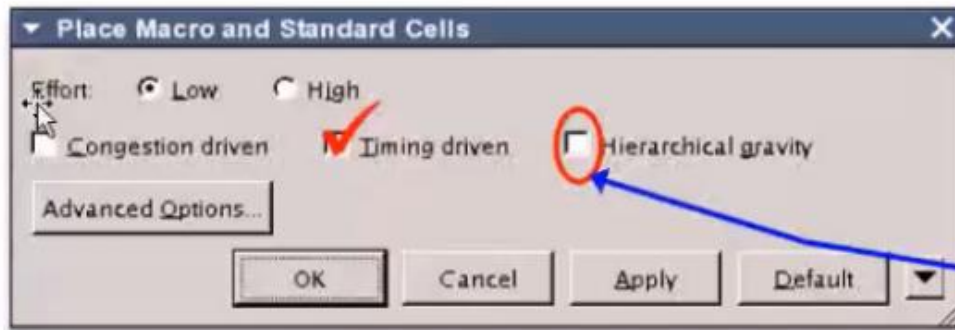
With VIPO turned ON, sizing and buffer insertion (which mimics more realistic placement optimization) is performed "in memory", thus relaxing the need to unnecessarily shorten the distance between the critical cells

Perform Virtual Flat Placement

Placement

Perform quick placement, which will be used to:

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



TURN OFF this
option for flat
design planning



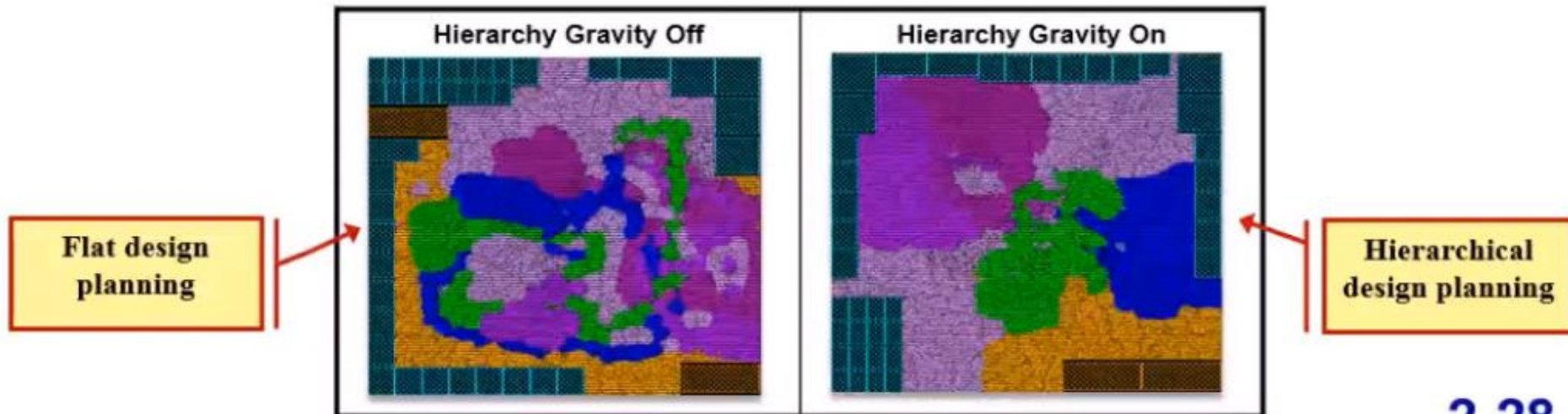
```
create_fp_placement -timing_driven -no_hierarchy_gravity
```

- ❖ 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

Placement

- 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

Starting Floorplan

Placement

Reduce Congestion

PNS

Reduce Delay

Write DEF

Create starting floorplan

Define placement parameters

Perform virtual flat placement

Virtual Flat Placement

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

Reduce Congestion

Starting Floorplan Placement **Reduce Congestion** PNS Reduce Delay Write DEF

DC-T synthesis with
default floorplan

Data setup

✓ Create starting floorplan

✓ Virtual flat placement

➔ Reduce congestion

Synthesize power network

Reduce delay

Write out DEF file

Design Planning

Analyze congestion

Modify placement constraints

Perform congestion-driven
virtual flat placement

Analyze congestion

Perform high-effort congestion-driven
virtual flat placement

Analyze congestion

Modify the floorplan

"Fix" all macro cell placements

Reduce Congestion

Synthesize the Power Network (PNS)

Starting Floorplan Placement Reduce Congestion **PNS** Reduce Delay Write DEF

DC-T synthesis with
default floorplan

Data setup

✓ Create starting floorplan

✓ Virtual flat placement

✓ Reduce congestion

➔ Synthesize power network

Reduce delay

Write out DEF file

Design Planning

Save the cell

Define logical P/G connections

Create P/G rings around macro groups

Apply power network constraints

Synthesize the power network

Analyze IR drop

Modify constraints and re-synthesize

Add P/G pad cells, if needed

Commit the power network

Connect P/G pins to power network

Create power rails

Analyze IR drop

Apply *pnet* options

Legalize placement

Synthesize Power Network

Define Logical Power/Ground Connections

PNS

Derive Power Ground Connection

☐ Auto connection

- ☐ Reconnect power/ground pins with existing connection
- ☐ Create power/ground nets from UPF supply nets
- ☐ Perform both power and tie connections
- ☐ Show detail connection information

☒ Manual connection

Power net:

Ground net:

Power pin:

Ground pin:

Create port: ☒ None ☐ Top ☐ All

Cells:

☐ Reconnect existing tie pins to appropriate power nets

- **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**

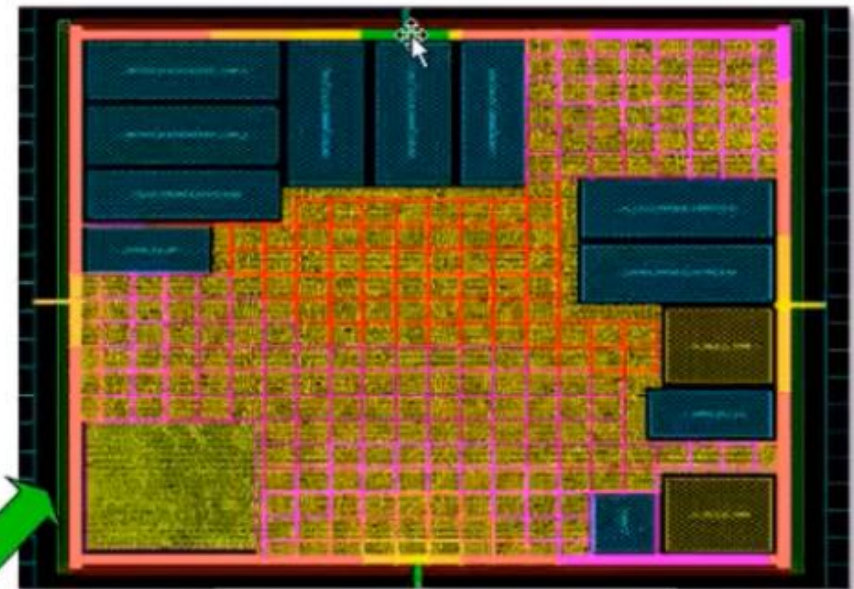
```
derive_pg_connection -power_net VDD -power_pin VDD -ground_net VSS -ground_pin VSS
derive_pg_connection -power_net VDD0 -power_pin VDD0 -ground_net VSS0 -ground_pin VSS0
derive_pg_connection -power_net VDDQ -power_pin VDDQ -ground_net VSSQ -ground_pin VSSQ
derive_pg_connection -power_net PWR -ground_net GND -tie
```

Synthesize and Analyze the Power Network

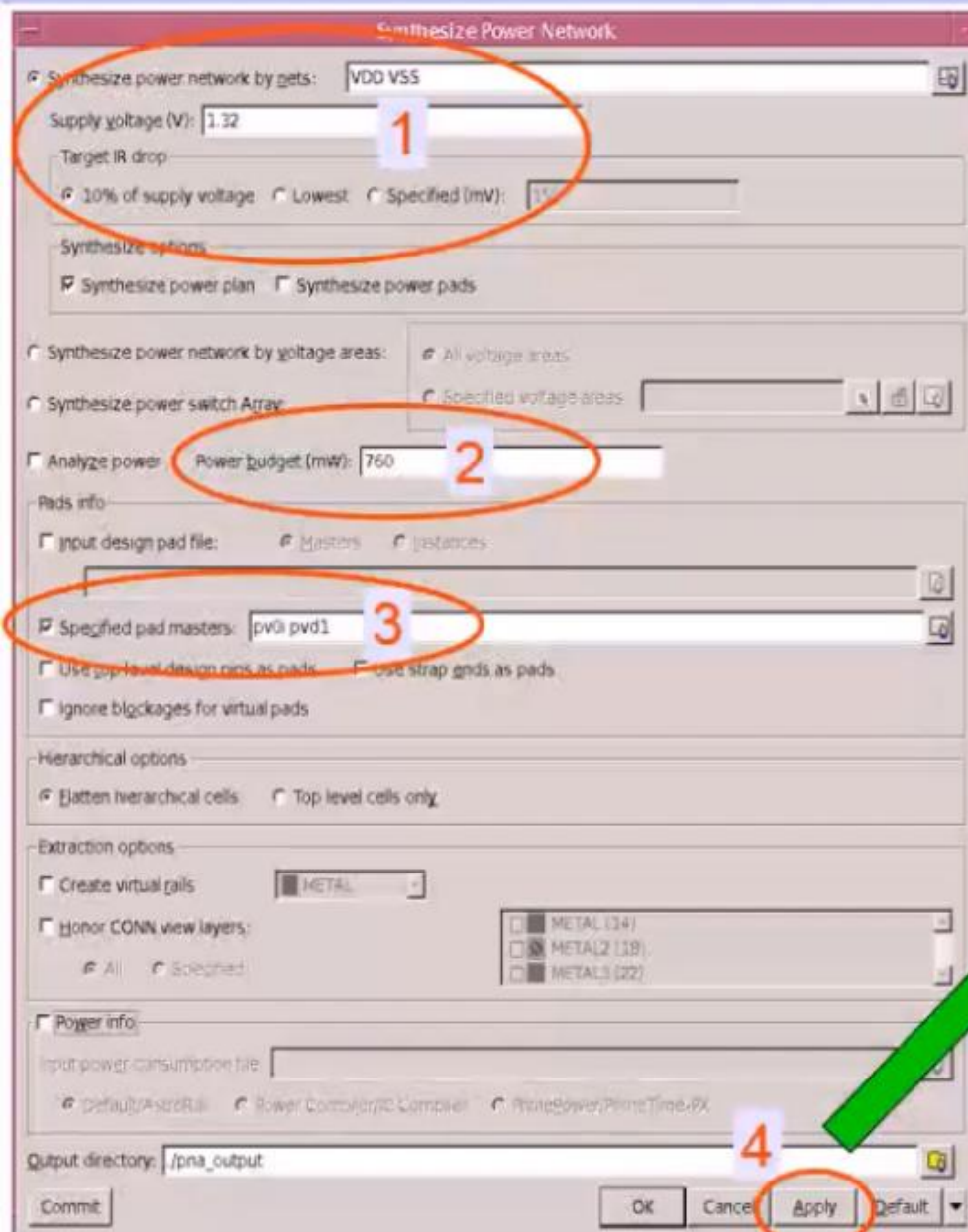
PNS

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

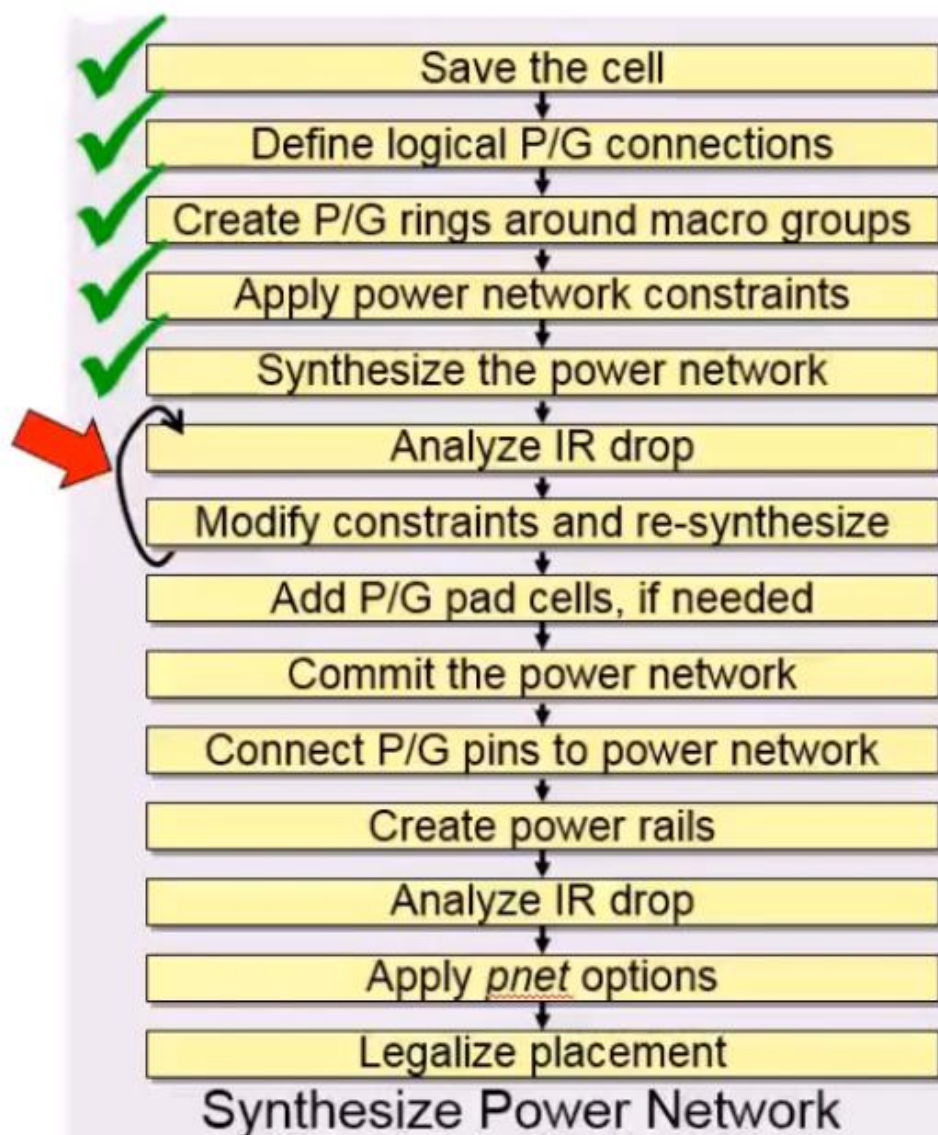


`synthesize_fp_rail ...`

A screenshot of the 'Synthesize Power Network' dialog box. The dialog is divided into several sections. The 'Synthesize power network by gets:' section is circled in red and contains a dropdown menu set to 'VDD VSS', a 'Supply voltage (V):' field set to '1.32' (labeled with a red '1'), and a 'Target IR drop:' section with radio buttons for '10% of supply voltage', 'Lowest', and 'Specified (mV):' (set to '1.1'). The 'Synthesize options' section has radio buttons for 'Synthesize power plan' (selected) and 'Synthesize power pads'. The 'Synthesize power network by voltage areas:' section has radio buttons for 'All voltage areas' and 'Specified voltage areas'. The 'Synthesize power switch Array:' section has a 'Specified voltage areas' field. The 'Analyze power' section has a 'Power budget (mW):' field set to '760' (labeled with a red '2'). The 'Pads info' section has radio buttons for 'Input design pad file:', 'Masters', and 'Instances'. The 'Specified pad masters:' field is set to 'pvd0 pvd1' (labeled with a red '3'). The 'Use top-level design pins as pads' and 'Use strap gnds as pads' options are also present. The 'Ignore blockages for virtual pads' option is checked. The 'Hierarchical options' section has radio buttons for 'Flatten hierarchical cells' and 'Top level cells only'. The 'Extraction options' section has a 'Create virtual rails' checkbox and a 'Metal' dropdown menu. The 'Honor CONN view layers:' section has radio buttons for 'All' and 'Specified', and a list of metal layers: 'METAL (14)', 'METAL (2 (118))', and 'METAL (3 (22))'. The 'Power info' section has a 'Input power consumption file' field and radio buttons for 'default/AsizeRail', 'Power Compiler/PC Compiler', and 'PrimePower/PrimeTime-RX'. The 'Output directory:' field is set to '/pna_output'. The 'Commit' button is at the bottom left, and the 'Ok', 'Cancel', 'Apply' (labeled with a red '4'), and 'Default' buttons are at the bottom right.

Modify Constraints and Re-synthesize

PNS

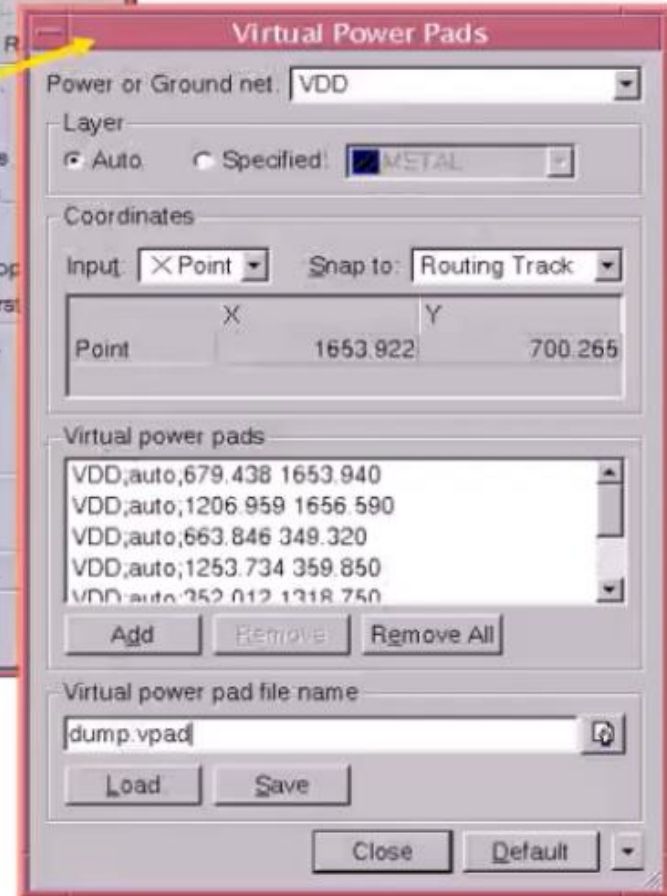
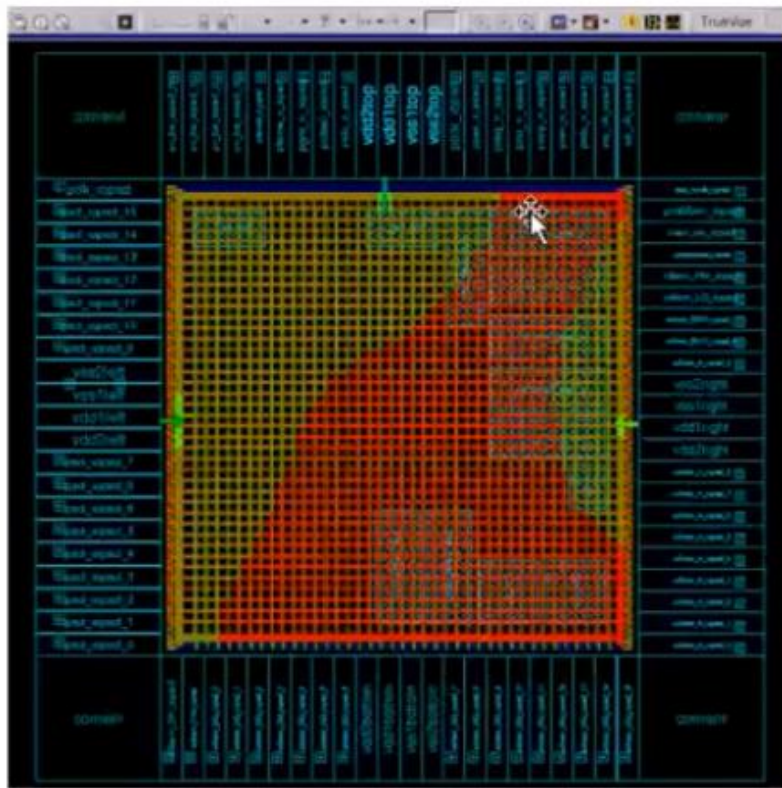


If the maximum IR drop is not acceptable

- 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

Create starting floorplan

:

Save the cell

Define logical P/G connections

Create P/G rings around macro groups

Apply power network constraints

Synthesize the power network

Analyze IR drop

Modify constraints and re-synthesize

Add P/G pad cells, if needed

Commit the power network

Connect P/G pins to power network

Create power rails

Analyze IR drop

Apply *pnet* options

Legalize placement

Synthesize Power Network

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

Synthesize Power Network

☒ Synthesize power network by gpts: VDD VSS

Supply voltage (V): 1.32

Target IR drop:

☒ 10% of supply voltage ☐ Lowest ☐ Specified (mV): 150

Synthesize options:

☒ Synthesize power plan ☐ Synthesize power pads

☐ Synthesize power network by voltage areas: ☐ All voltage areas

☐ Synthesize power switch Array: ☐ Specified voltage areas

☐ Analyze power: Power budget (mW): 760

Pads info:

☐ Input design pad file: ☐ Hierarchy ☐ Instances

☒ Specified pad masters: pvd0 pvd1

☐ Use top level design pins as pads ☐ Use strap gnds as pads

☐ Ignore blockages for virtual pads

Hierarchical options:

☒ Flatten hierarchical cells ☐ Top level cells only

Extraction options:

☐ Create virtual pads: METAL

☐ Ignore CONN view layers: ☐ METAL (1) ☐ METAL2 (18) ☐ METAL3 (22)

☐ All ☐ Specified

☐ Power info:

Input power to mainboard file:

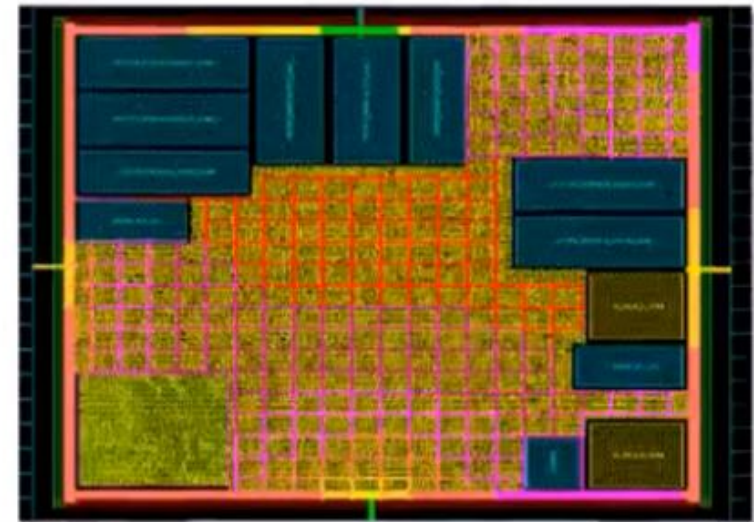
☐ Use pvd0/pvd1 ☐ Power Comp (pvd0/pvd1) ☐ Power (pvd0/pvd1)

Output directory: /pna_output

Commit OK Cancel Apply Default

commit_fp_rail

Preview (Apply)



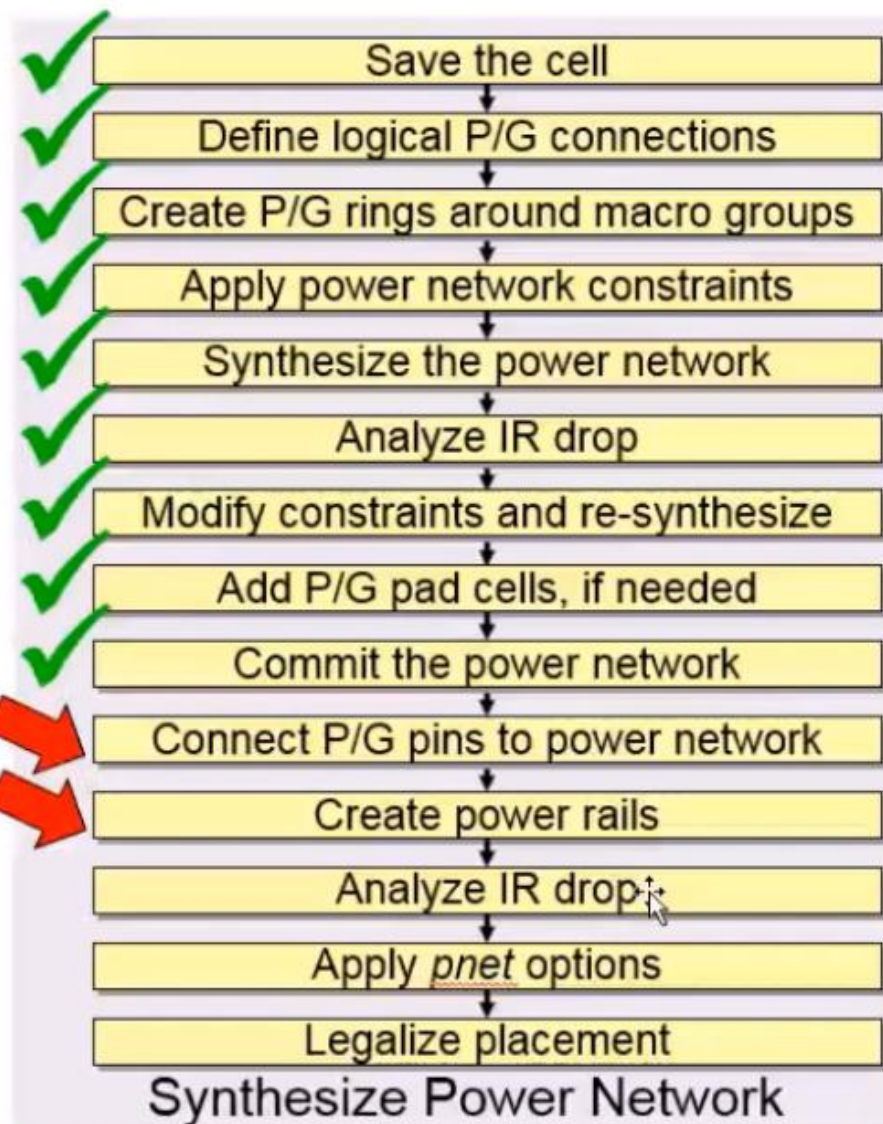
Build (Commit) the power network



Straps and rings
are routed

Connect P/G Pins and Create Power Rails

PNS

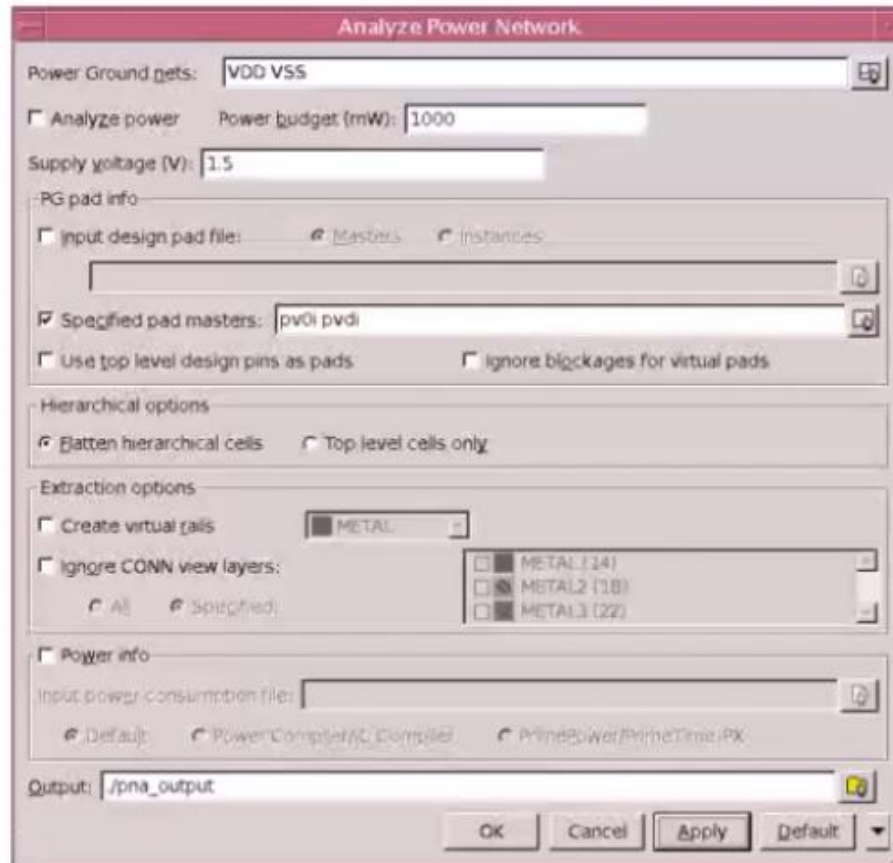


- Connect P/G macro pins and pad pins to core rings and straps
- Create power rails along the standard cell placement rows

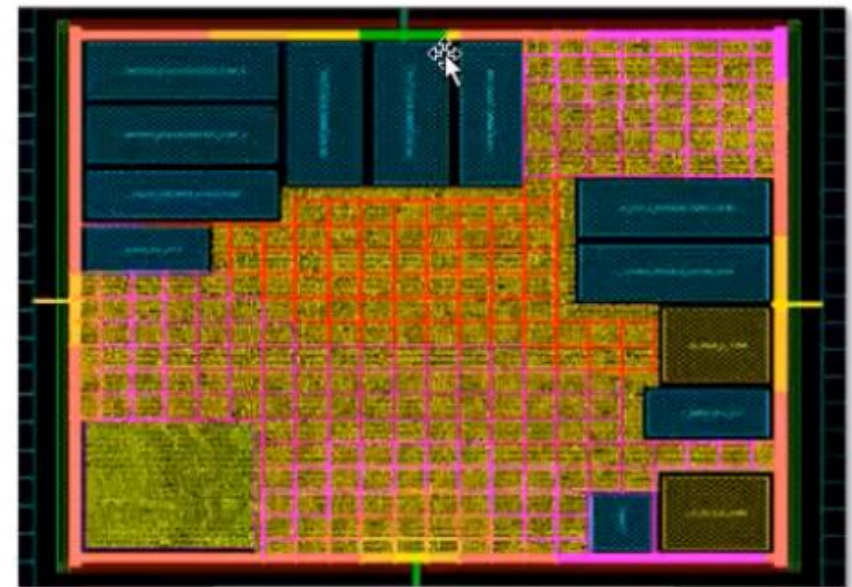
```
preroute_instances  
preroute_standard_cells \  
  -fill_empty_rows \  
  -remove_floating_pieces
```

Analyze the Power Network

PNS



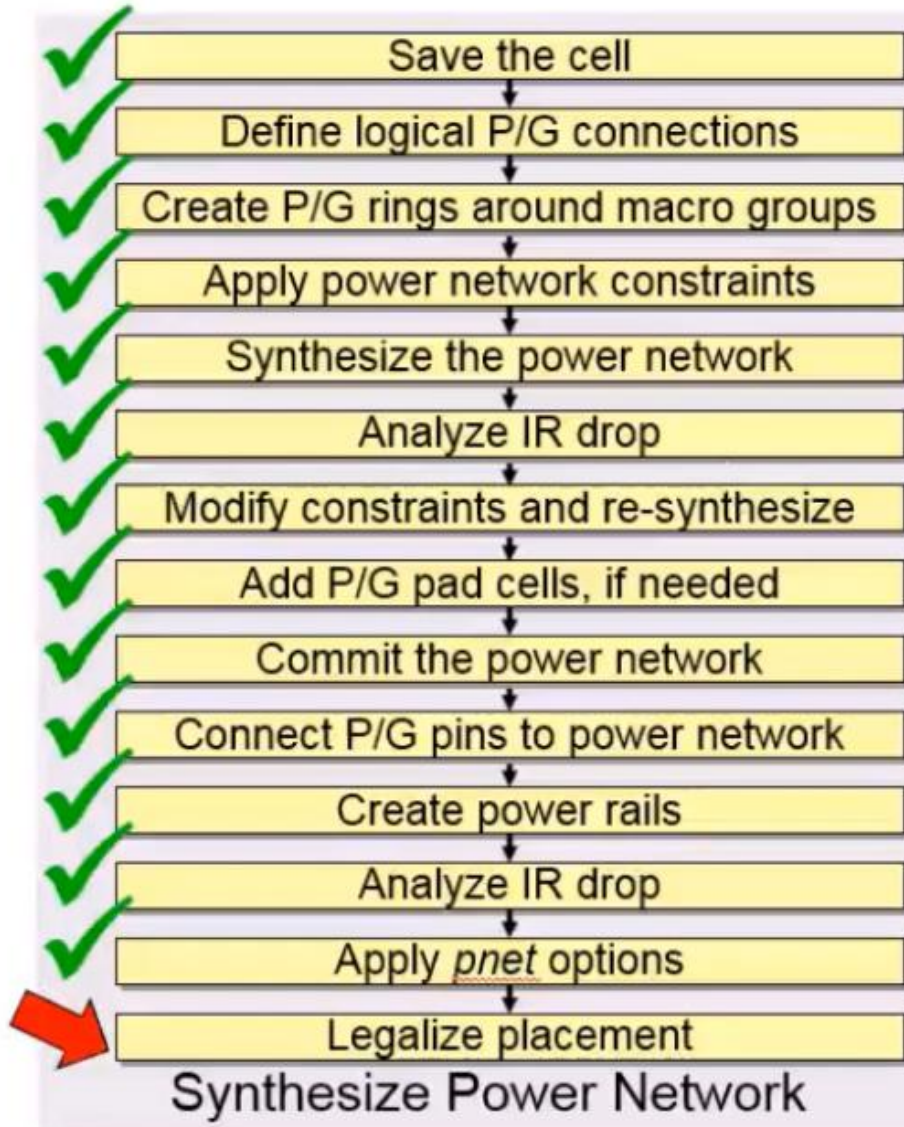
After "commit" and after P/G pins and rails have been pre-routed → Analyze the power network for a more accurate IR-drop map



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


Perform Placement Legalization

PNS

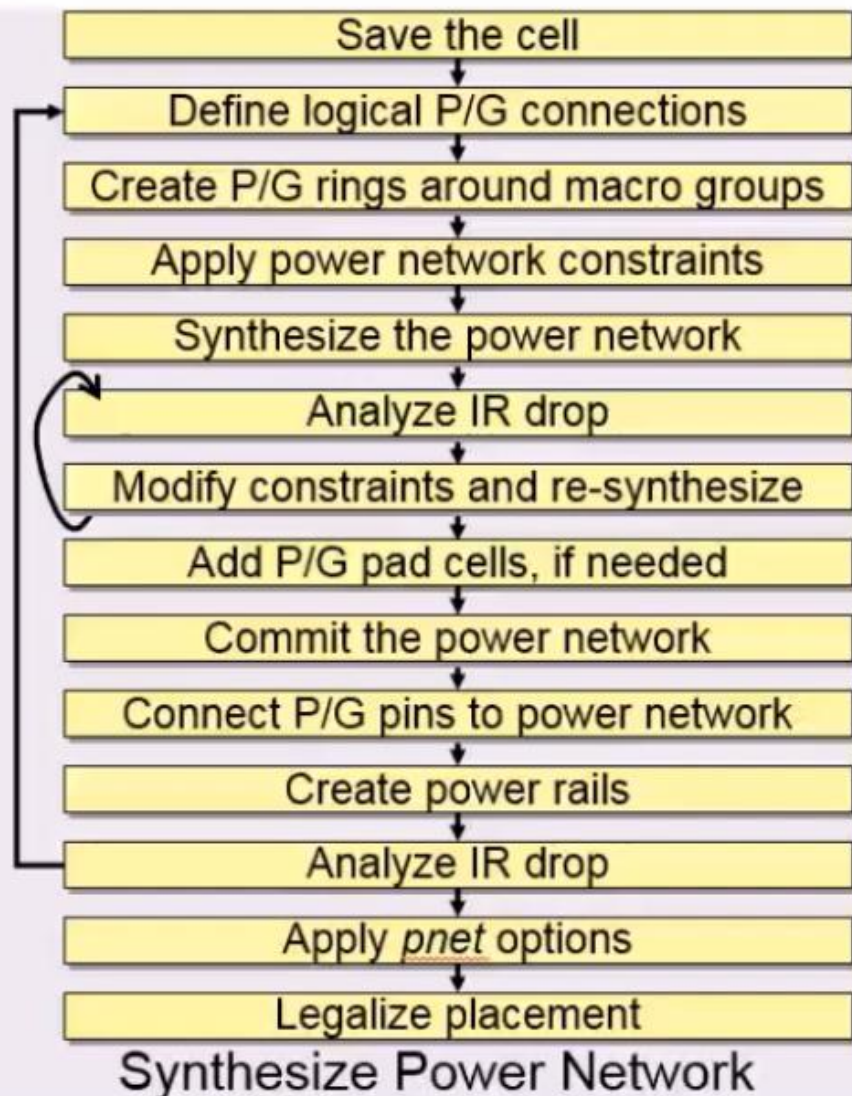


If you have applied *partial* or *complete* power net placement blockages you will need to legalize placement to move violating standard cells away from the power straps

legalize_fp_placement

Summary: Synthesize the Power Network

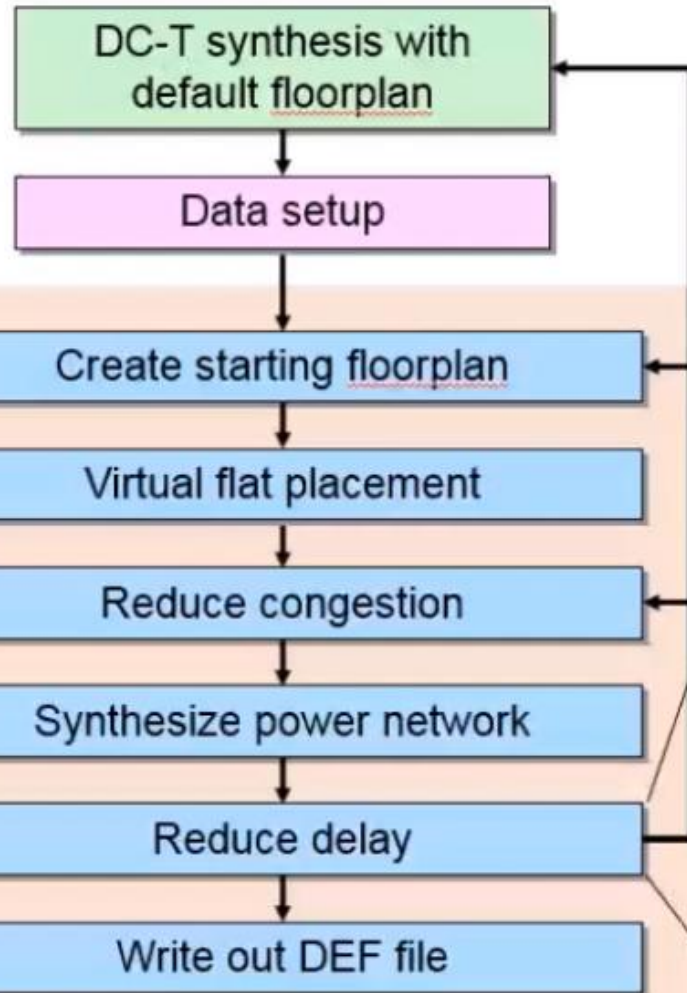
Starting Floorplan Placement Reduce Congestion **PNS** Reduce Delay Write DEF



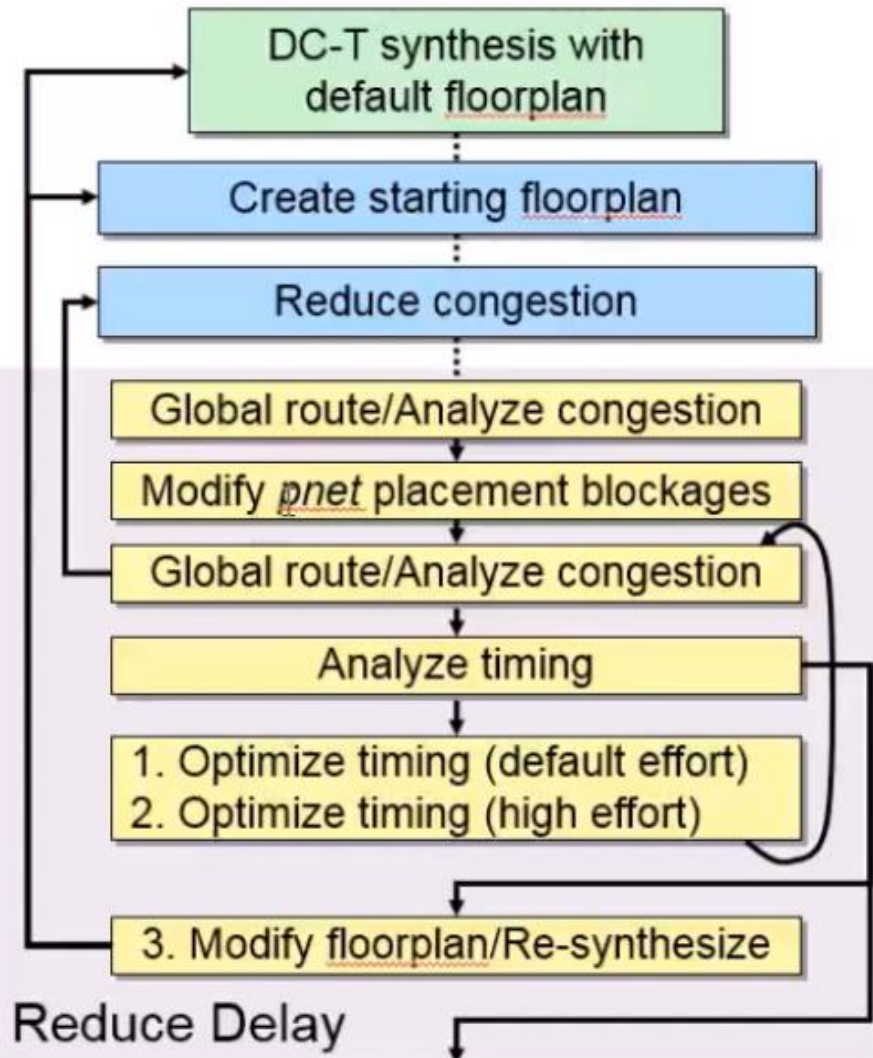
```
save_mw_cel -as DESIGN_pre_pns
derive_pg_connection ... | I
set_fp_rail_region_constraints ...
create_fp_group_block_ring ...
commit_fp_group_block_ring
set_fp_rail_region_constraints -remove
set_fp_rail_constraints ...
synthesize_fp_rail ...
# Modify power network constraints
# and re-synthesize, 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

Starting Floorplan Placement Reduce Congestion PNS **Reduce Delay** Write DEF



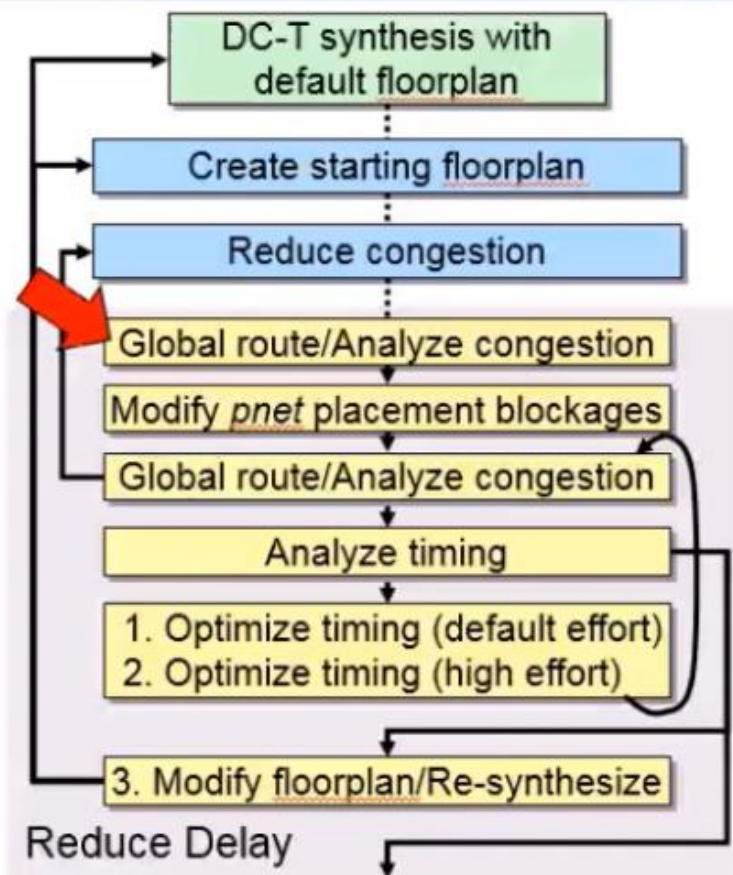
Design Planning



Reduce Delay

Global Route and Analyze Congestion

Reduce Delay



- Before performing timing analysis perform an actual global route

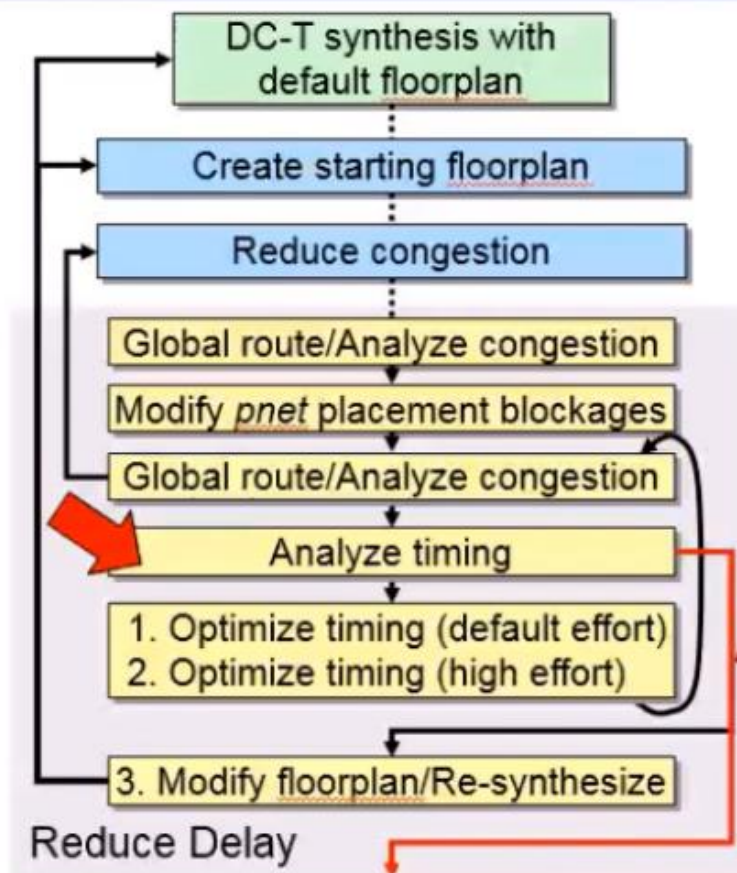
route_zrt_global

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

- Analyze the congestion map

Analyze Timing

Reduce Delay



■ Generate a timing report

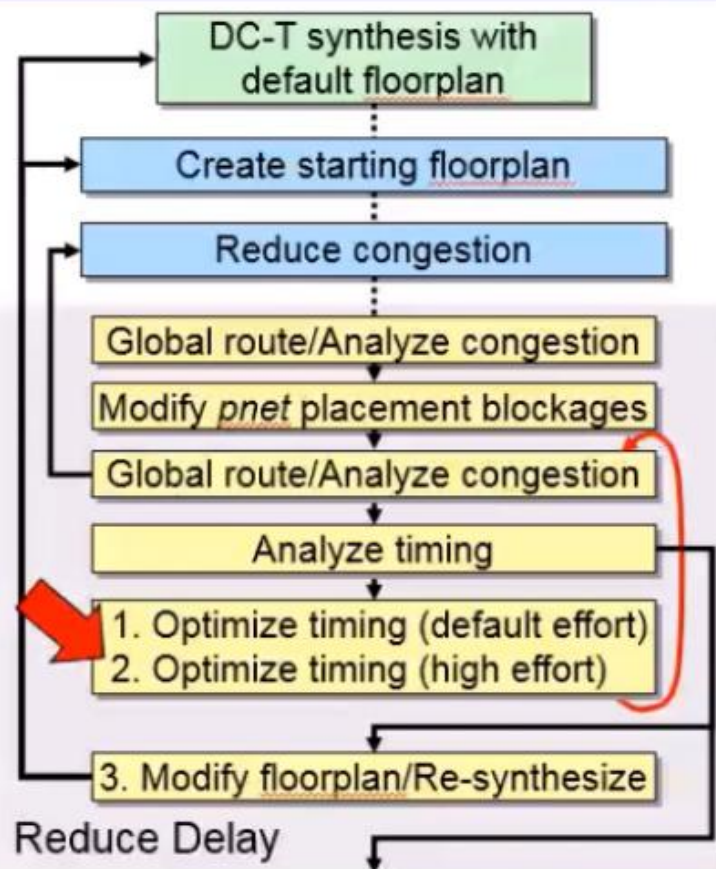
- Parasitic RC extraction is automatically invoked (extract_rc)
- Acceptable timing:
 $WNS < 15\text{-}20\%$ of required delay

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*

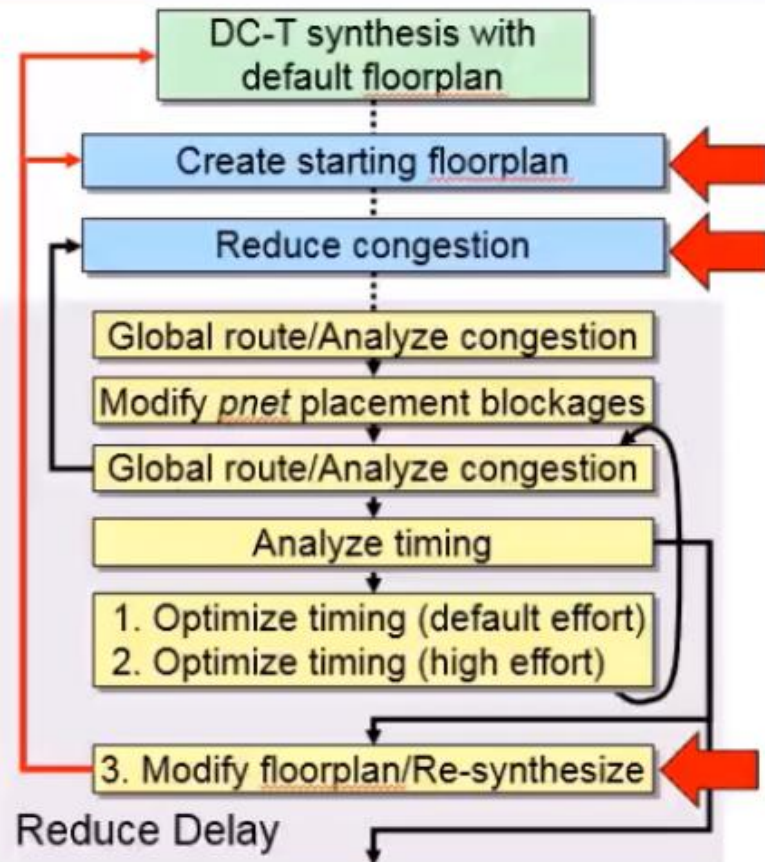
```
optimize_fp_timing -fix_design_rule  
<-effort high>
```

- 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

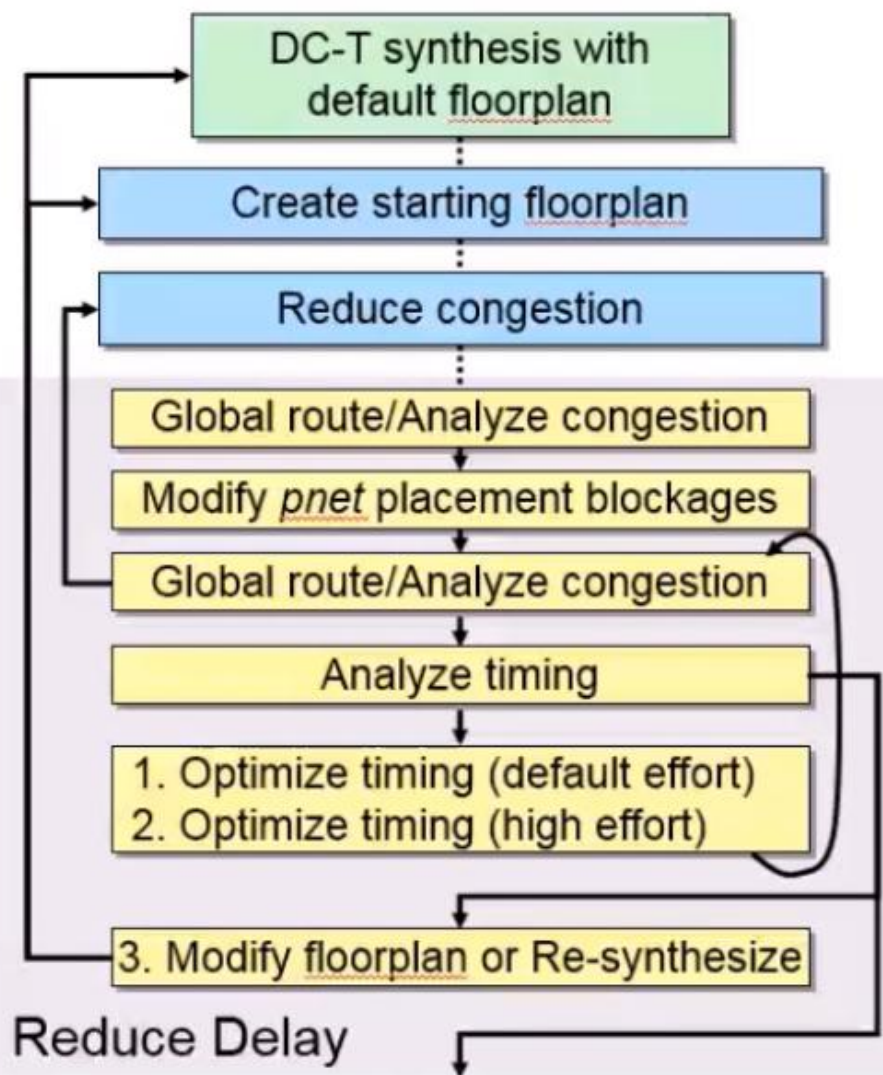


If the timing violations are still unacceptably large after *high-effort optimization*:

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

Summary: Reduce Delay

Starting Floorplan Placement Reduce Congestion PNS Reduce Delay Write DEF



```
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

Starting Floorplan Placement Reduce Congestion PNS Reduce Delay **Write DEF**

DC-T synthesis with
default floorplan

Data setup

✓ Create starting floorplan

✓ Virtual flat placement

✓ Reduce congestion

✓ Synthesize power network

✓ Reduce delay

➔ Write out DEF file

Design Planning

Capture placement and layer constraints

Remove placed standard cells

Write out a DEF file

Close the design

Write DEF File

Capture placement and layer constraints



Remove placed standard cells



Write out a DEF file



Close the design

Write DEF File

■ Write out a *DEF* file:

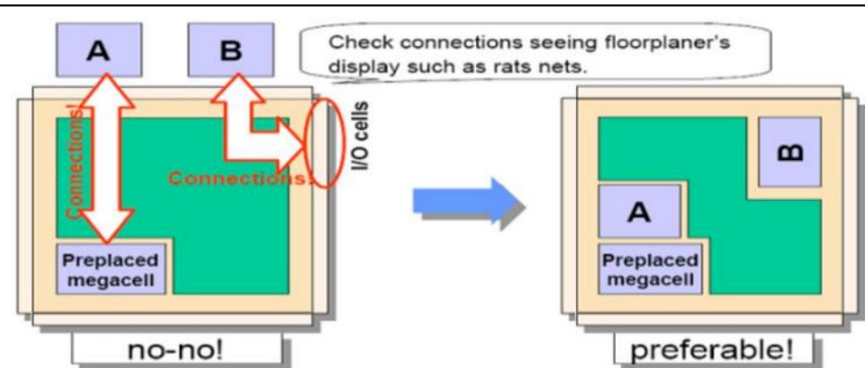
- Will be used by *Design Compiler Topographical* for re-synthesis
- Will be loaded into IC Compiler after re-synthesis and data setup

■ Close the design cell without saving it

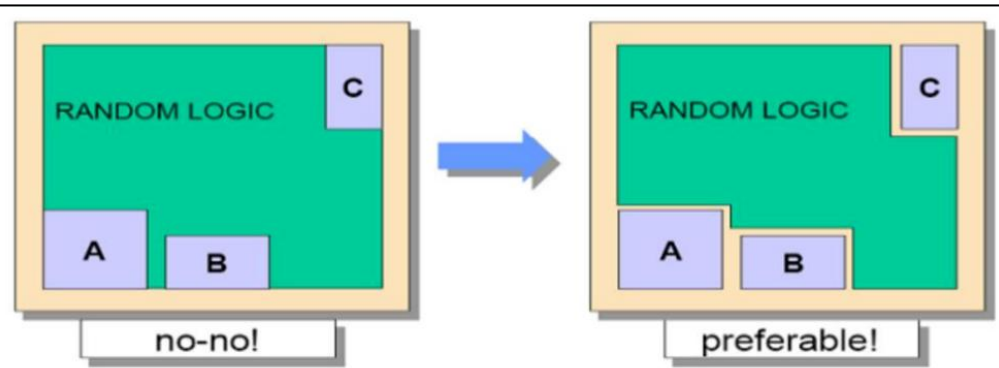
- Only the above files are needed

```
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
```

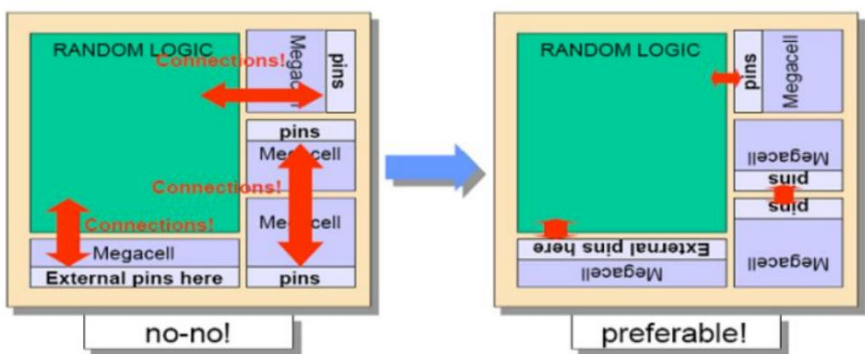

Place Macro



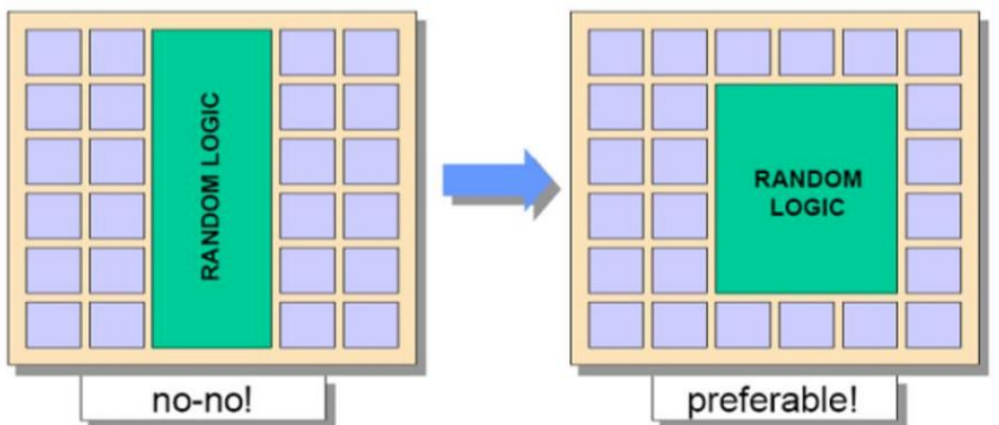
放置新单元时候考虑已经固定的单元之间的连接



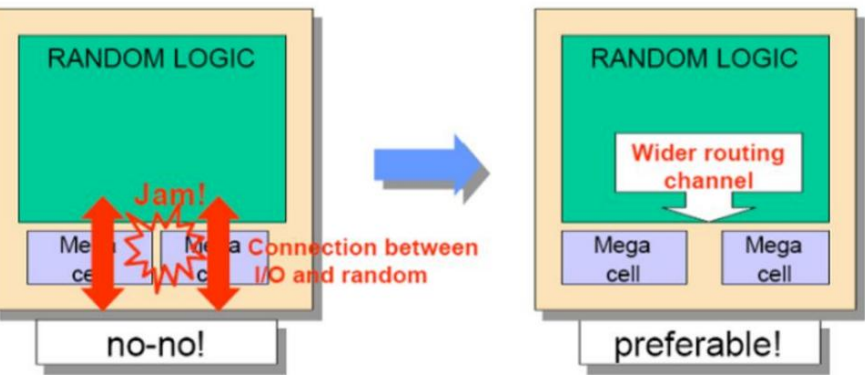
删除宏周围的 rows



确定单元出 pin 方向与有联系的单元靠近



保持摆放标准单元区域的宽高比接近 1.0

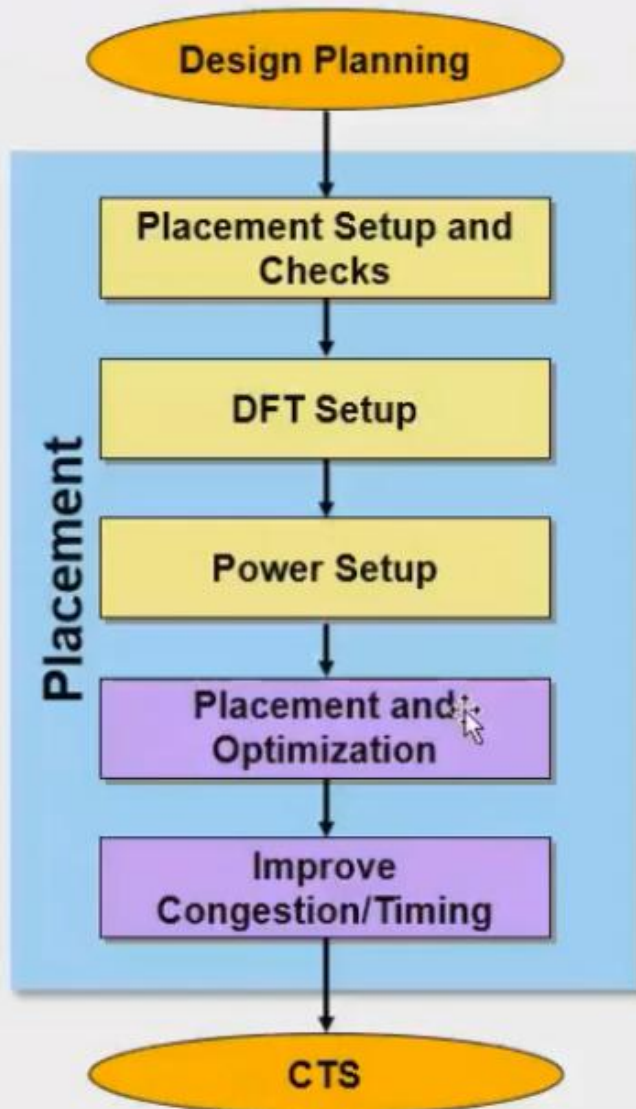


保证单元之间的间距足够出 pin 布线

Design Status Prior to Placement

- ***Design Planning* is completed**
 - If using 3rd party floorplanning tool “Floorplan Exploration” is done (Refer to “Design Planning” Unit 2 Appendix)
- ***Second-Pass Synthesis* is completed**
 - Based on actual floorplan
- ***Second-Pass Data Setup* is completed**
 - New design cell is generated based on 2nd pass netlist
- **“Floorplanned cell” is generated – ready for placement**
 - Core and periphery areas defined
 - Macros are placed and “fixed”
 - Placement blockages defined
 - Power grid pre-routed
 - Standard cell placement is discarded

IC Compiler Placement Flow



The “placement phase” involves several key steps:

- Setup steps to control placement, DFT and power
- Placement and optimization
- Incremental refinement to improve congestion and/or setup timing

Note: The flow diagrams included in this unit represent an **example** flow, not the recommended flow

“Fix” all Macro Cell Placements

- In most situations macro cell placement is determined during design planning and their placement is “fixed”
- It is a good practice to fix all macro placements again, just in case....
 - You may have manually moved a macro after design planning and have forgotten to fix its placement

```
open_mw_cel DESIGN_floorplanned  
source tim_opt_ctrl.tcl  
set_dont_touch_placement [all_macro_cells]
```

Re-apply timing and
optimization controls

Verify *Layer* and *Placement* Constraints

The following checks are recommended because their settings are not saved in the *DEF floorplan* file – they must be re-applied after creation of the 2nd pass design library and loading of the *floorplan*

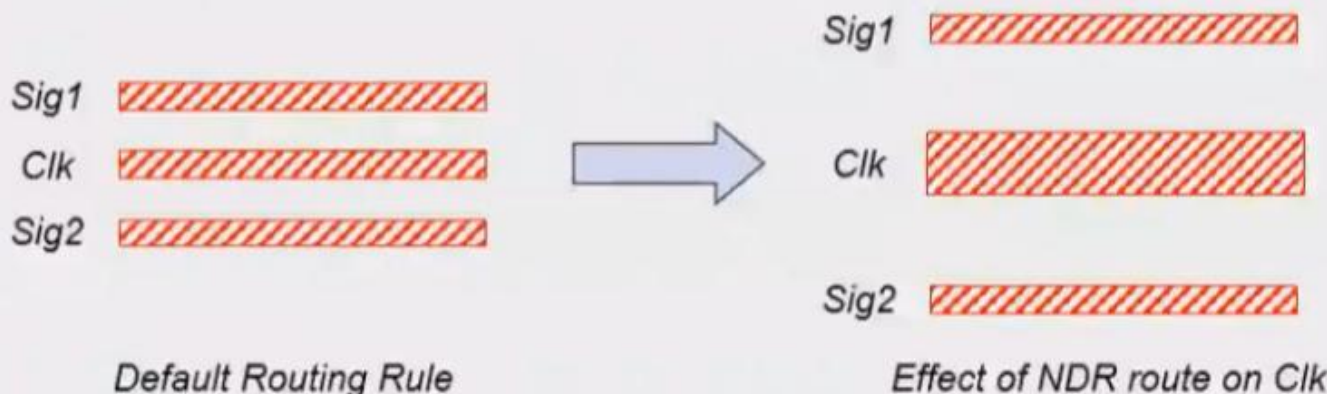
- Ensure that the following settings have been re-applied, as appropriate:
 - Ignored routing layers
 - Placement blockages under P/G straps - *pnet* options
 - Global placement *keepout* variable settings

```
report_ignored_layers  
report_pnet_options  
printvar physopt_hard_keepout_distance  
printvar placer_soft_keepout_channel_width
```

- Apply or correct as needed – see notes below

Non-Default Clock Routing

- IC Compiler can route clock nets using non-default routing (NDR) rules, e.g. double-spacing, double-width, shielding
- Non-default rules are often used to “harden” the clock, e.g. to make the clock routes less sensitive to *cross-talk* or *electro-migration (EM)* effects
- NDR nets use more routing resources (tracks) which can increase congestion
- This congestion impact can be taken into account during global-route driven placement



Specify Non-Default Routing Rules

■ Define the NDR rules:

Can also be used to define shielding and tapering rules – see notes below

```
define_routing_rule MY_ROUTE_RULES \  
  -widths {METAL3 0.4 METAL4 0.4 METAL5 0.8} \  
  -spacings {METAL3 0.42 METAL4 0.63 METAL5 0.82}
```

■ Configure the clock tree routing:

```
set_clock_tree_options -clock_trees [all_clocks] \  
  -routing_rule MY_ROUTE_RULES \  
  -layer_list "METAL3 METAL5"
```

You may also specify the min/max layers to be used for clock tree routing

Check Placement Readiness

`check_physical_design -stage pre_place_opt` checks

the readiness for placement of:

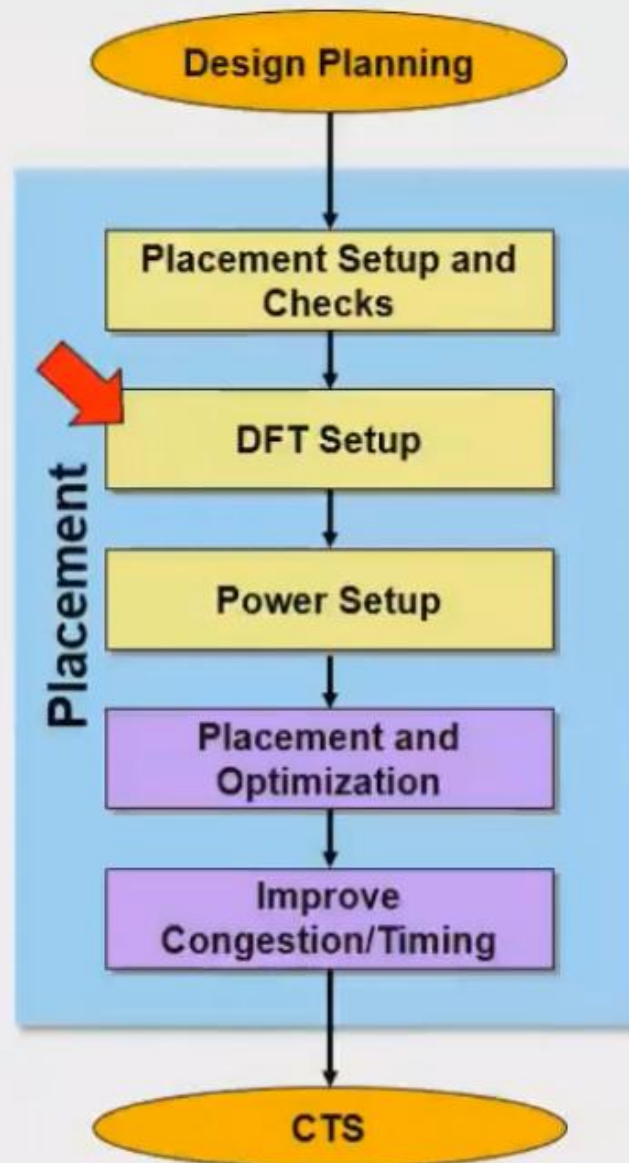
- Floorplan
- Netlist
- Design constraints

`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

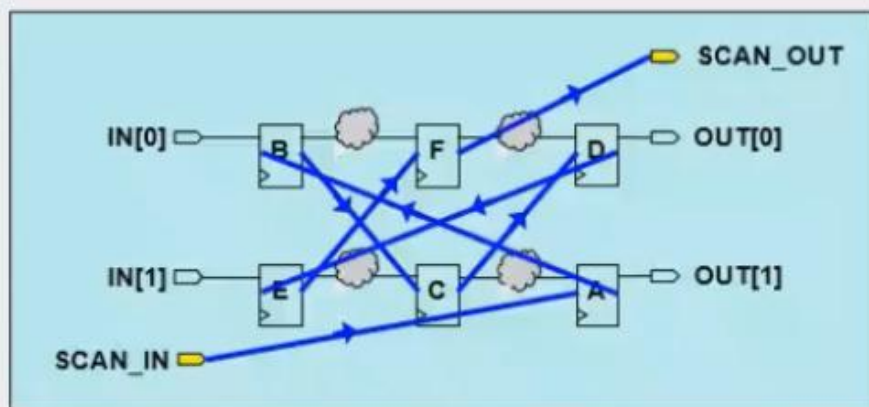
Modify the floorplan, constraints or libraries as needed

Design-for-Test (DFT) Setup



Skip this if your design does not include scan chains.

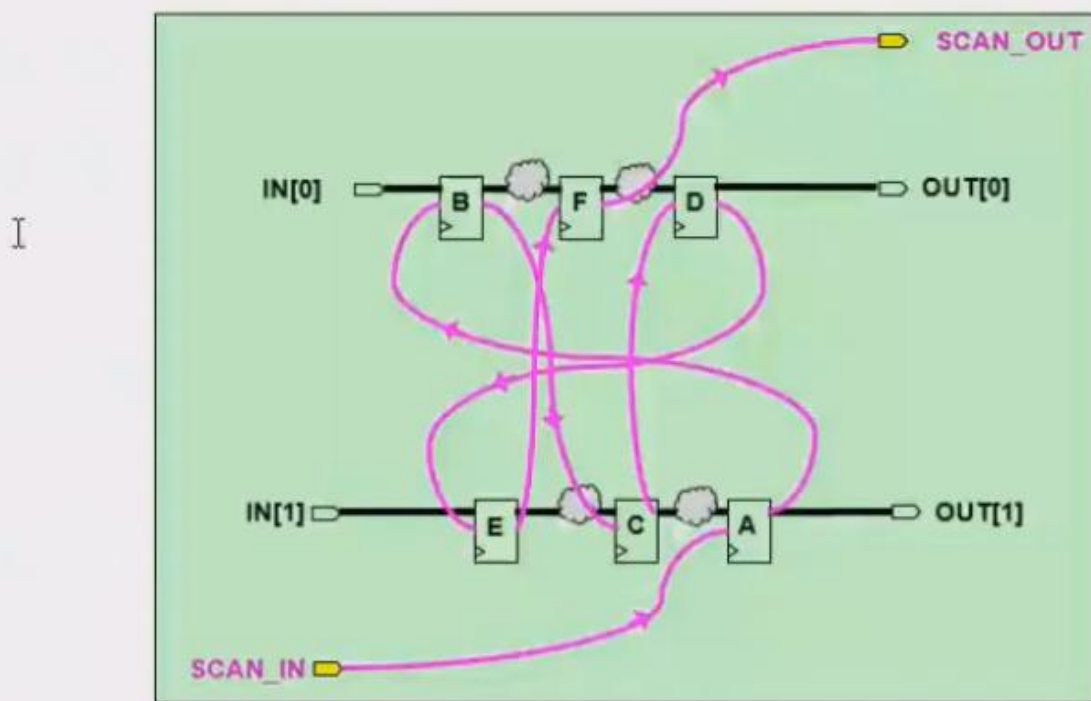
Pre-Existing Scan Chains



- If your design flow includes “design-for-test”, the netlist will contain “scan chains”:
 - Groups of “scan registers” that are serially connected through SI/SO pins, and inserted during synthesis
- Scan chain paths are active only during “test mode”, not during “functional mode”
- Registers are typically connected in alphanumeric order during synthesis – irrelevant for DFT, but not optimal for routing

The Issue with Existing Scan Chains

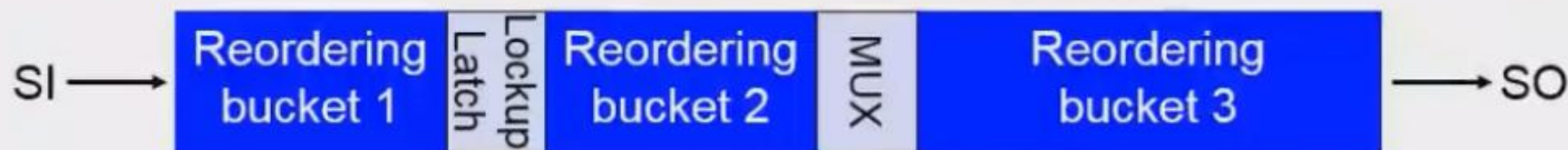
What happens if placement is done with the pre-existing order of the scan chains?



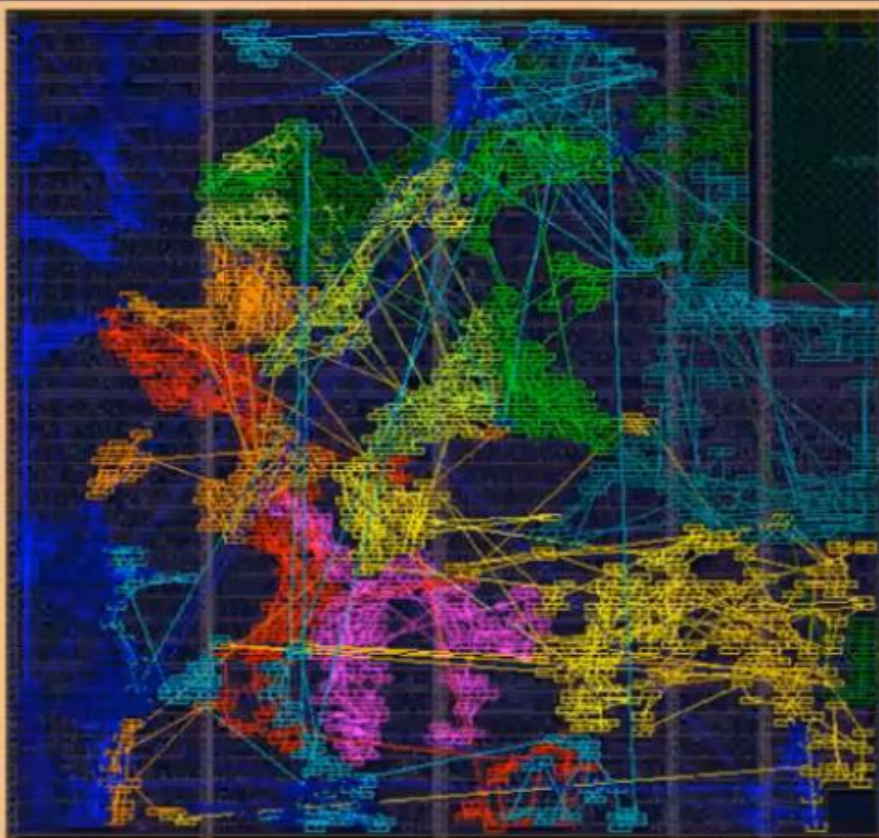
- Serially connected scan registers may be placed far apart which may require a lot more routing resources

SCANDEF-Based Chain Reordering

- IC Compiler can perform placement-aware reordering of scan cells
- The scan chain information (*SCANDEF*) from synthesis can be transferred to IC Compiler in two ways:
 - By loading the netlist in ddc format (imbedded *SCANDEF*)
 - By loading a *SCANDEF* file (generated after scan insertion)
- Reordering occurs within each scan chain
- Lockup latches or multiplexers break up scan chains further into “reordering buckets”
- If present, reordering happens within the buckets



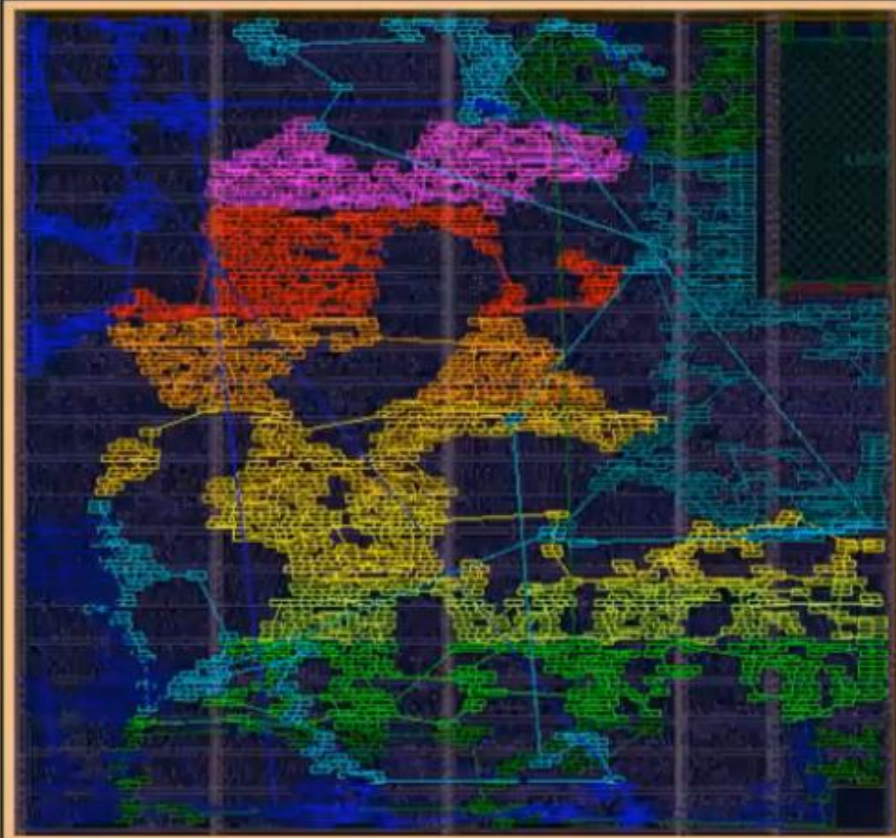
Placement-Based Scan Chain Re-Ordering



Pre-existing Scan Ordering

```
place_opt
```

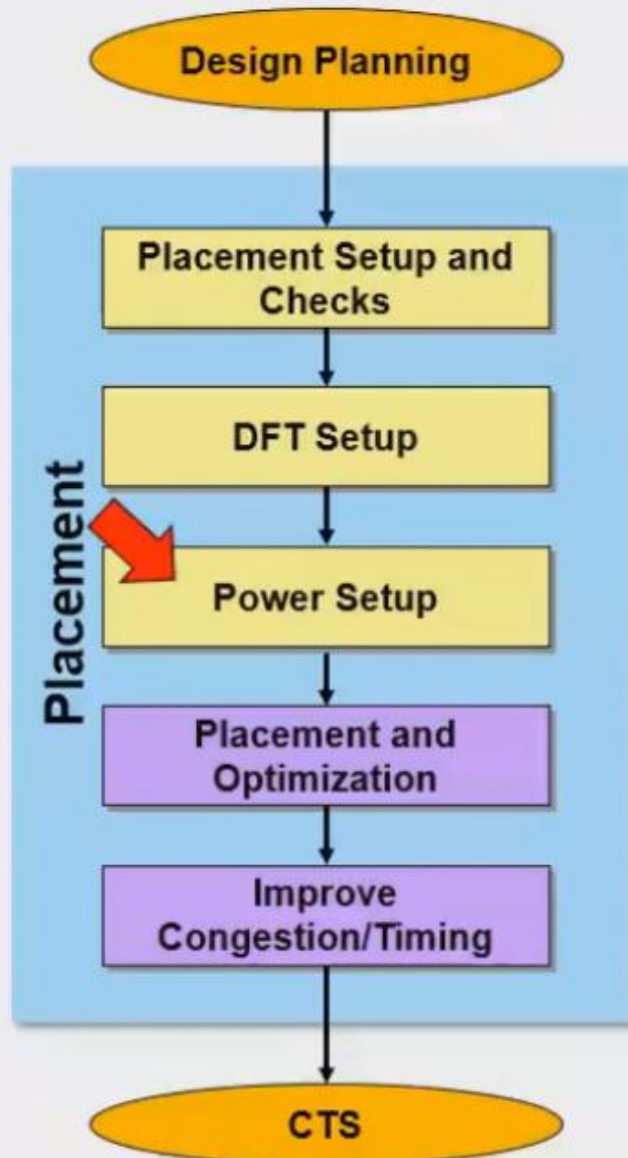
Not needed if imported design was
in *ddc* format – already included!



Placement-Based Re-Ordering

```
read_def DESIGN.scandef  
report_scan_chain  
place_opt -optimize_dft
```

Power Setup



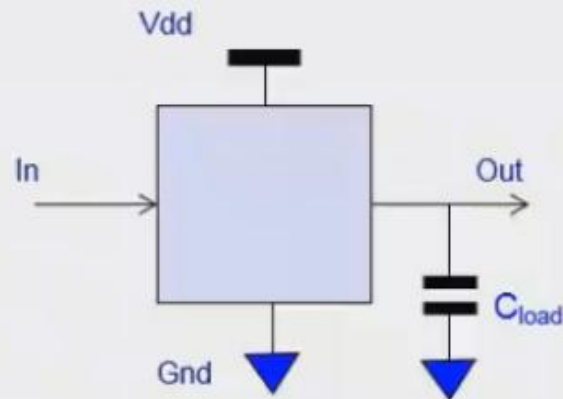
Skip this if power optimization is not a priority

Power optimization will generally:

- Not impact critical path delays or congestion
- Increase run time

Where Does Power Dissipation Occur?

$$\text{Total Power} = \text{Static Power} + \text{Dynamic Power}$$

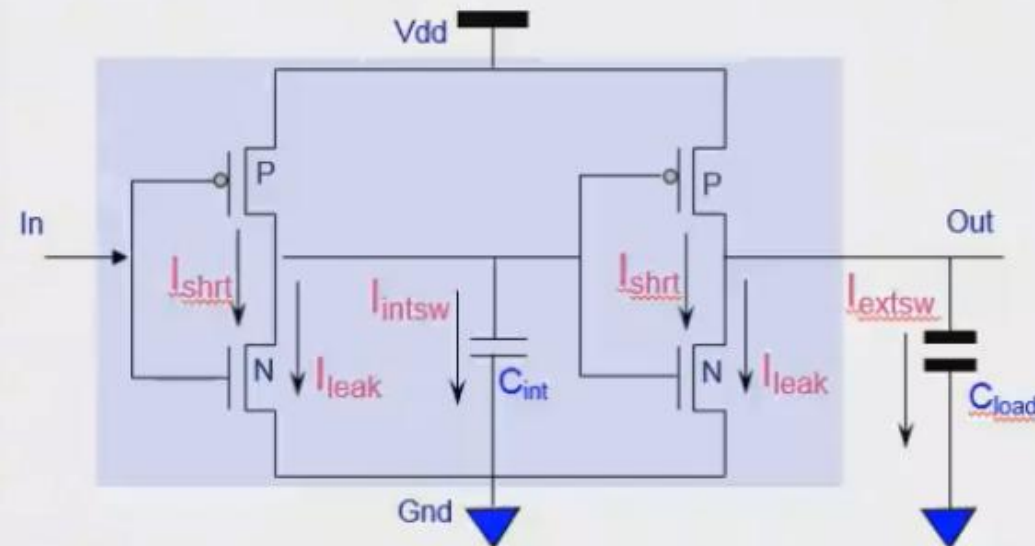


■ Static or leakage power dissipation comes from:

- Current leakage through devices that are "OFF" during static states

■ Dynamic or switching power dissipation comes from:

- Charging external loads
- Charging internal loads
- Short circuit or transient current while both devices are ON during a logic swing



Leakage Power Optimization

Low- V_{th} Cells
Fast, High Leakage

Trade-Off

High- V_{th} Cells
Slow, Low Leakage



- A multi- V_{th} library is the key to minimize leakage power

- Low V_{th} cells used on critical max-delay paths to help timing



- High V_{th} cells used on non-critical max paths to save leakage power

```
set_app_var target_library "sc_max_hvt.db sc_max_lvt.db"
set_app_var link_library "* $target_library io_max.db \
                           macros_max.db"

set_min_library sc_max_hvt.db -min_version sc_min_hvt.db
set_min_library sc_max_lvt.db -min_version sc_min_lvt.db
set_min_library io_max.db -min_version io_min.db
set_min_library macros_max.db -min_version macros_min.db
create_mw_lib ... -mw_reference_library \
    "mw/sc_hvt mw/sc_lvt mw/io mw/ram32"
...
set_power_options -leakage true|false
```

Report Multi- V_{th} Cells

After placement you can generate a report summarizing how many of each type of cells were used

Report

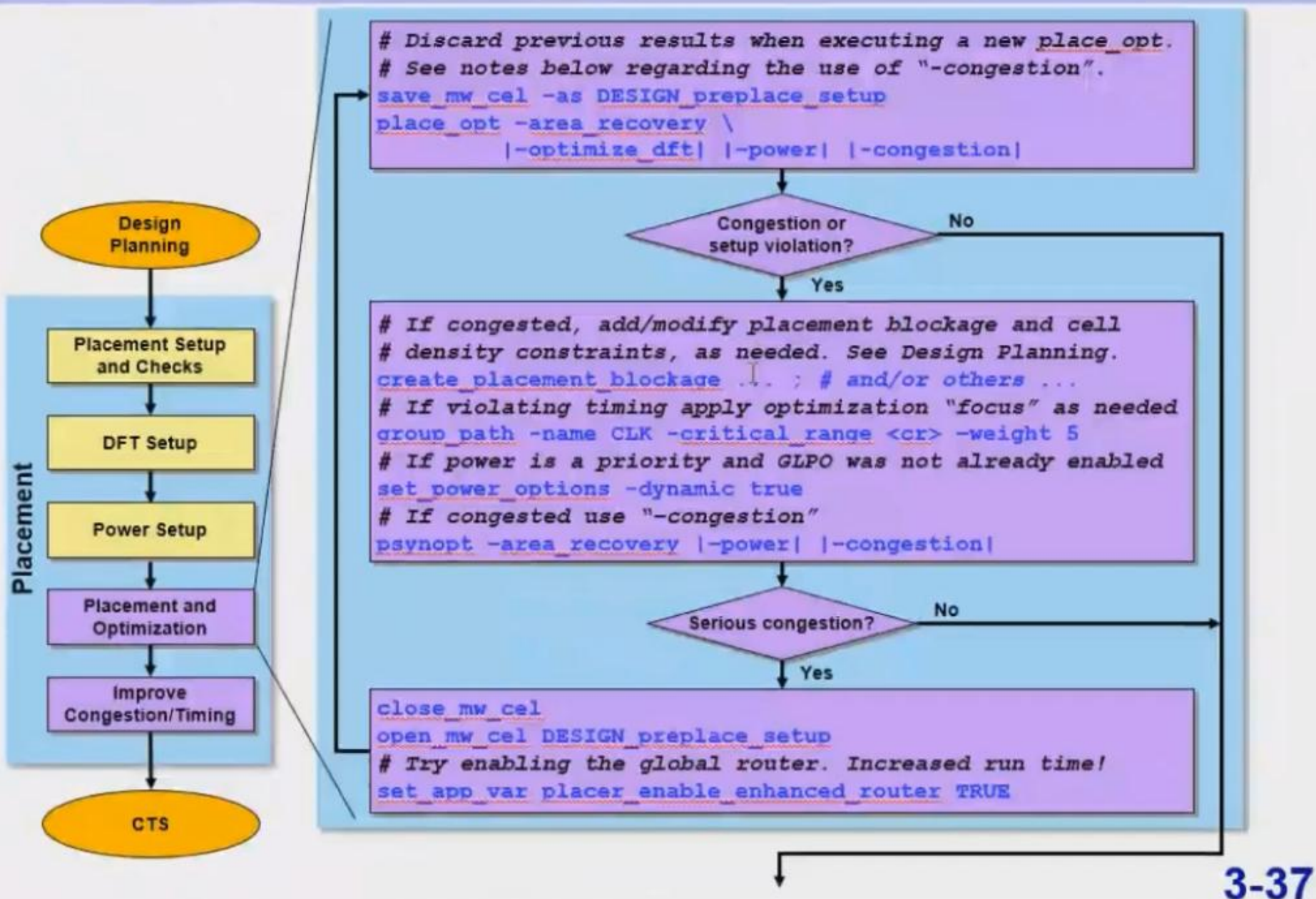
```
icc_shell> report_threshold_voltage_group
```

```
*****
                          Threshold Voltage Group Report
Threshold Voltage Group      Number of Cells      Percentage
*****
LVt                           90                   8.33%
HVt                          931                   86.12%
SVt                           59                   5.46%
undefined                     1                   0.09%
*****
```

These are typically macro or IP cells, which are not defined as low or high V_{th}

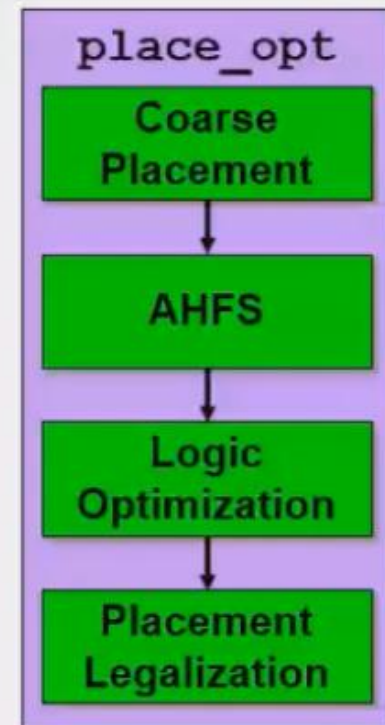
Note: This reporting requires special library attributes. See notes below.

Overview: Placement and Optimization



Placement and Logic Optimization

<u>place_opt</u>	Performs timing- and congestion-driven placement and logic optimization
- <u>area_recovery</u>	Enables buffer removal and cell down-sizing of non-critical paths
- <u>optimize_dft</u>	Performs scan chain re-ordering
-power	Invokes leakage and/or dynamic power optimization
-congestion	Invokes additional congestion-driven algorithms

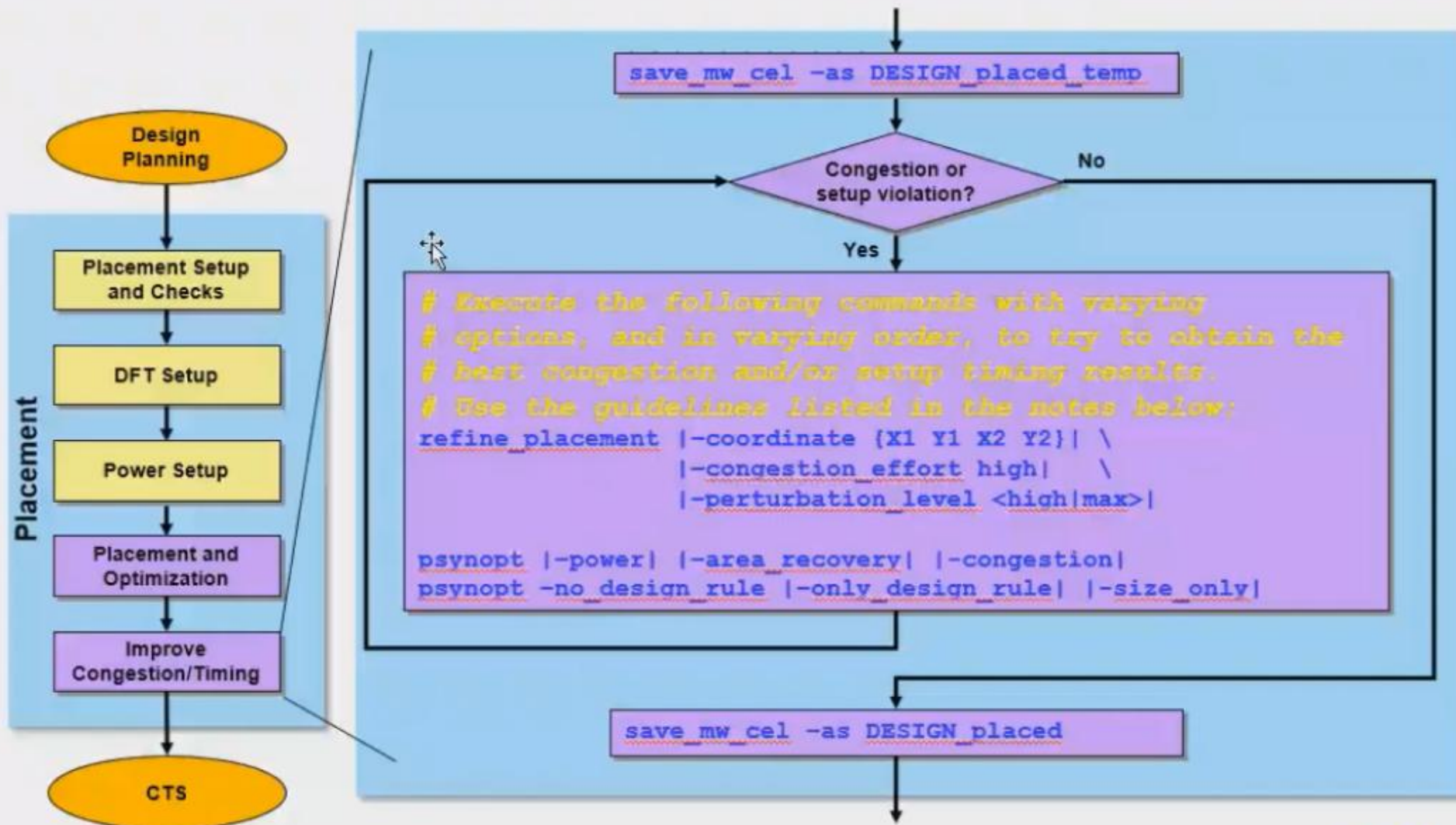


```
place_opt -area_recovery \  
| -optimize_dft | | -power | | -congestion |
```

Consider:

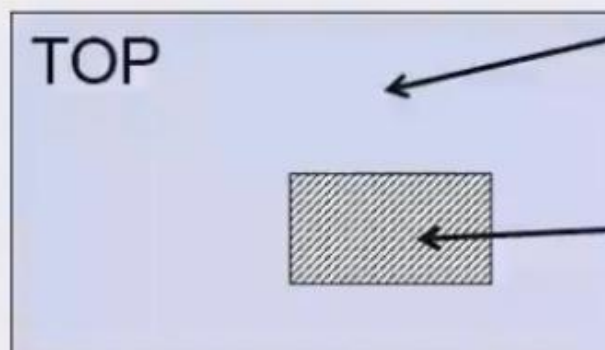
- Using `-area_recovery`: Can help reduce congestion and power; May impact run time
- Using `-optimize_dft` and `-power` only if DFT and power optimization are required, resp.
- Using `-congestion` only if congestion is known to be a serious issue (see next slide)

Overview: Improve Congestion/Timing



refine_placement

- Optimizes placement incrementally to improve congestion
 - Does not modify the netlist → Timing may degrade
- Can be executed back-to-back with different options
 - Discarding previous placement may improve results
- Can be followed with psynopt to improve timing



Incremental run can
be applied to the
entire design...

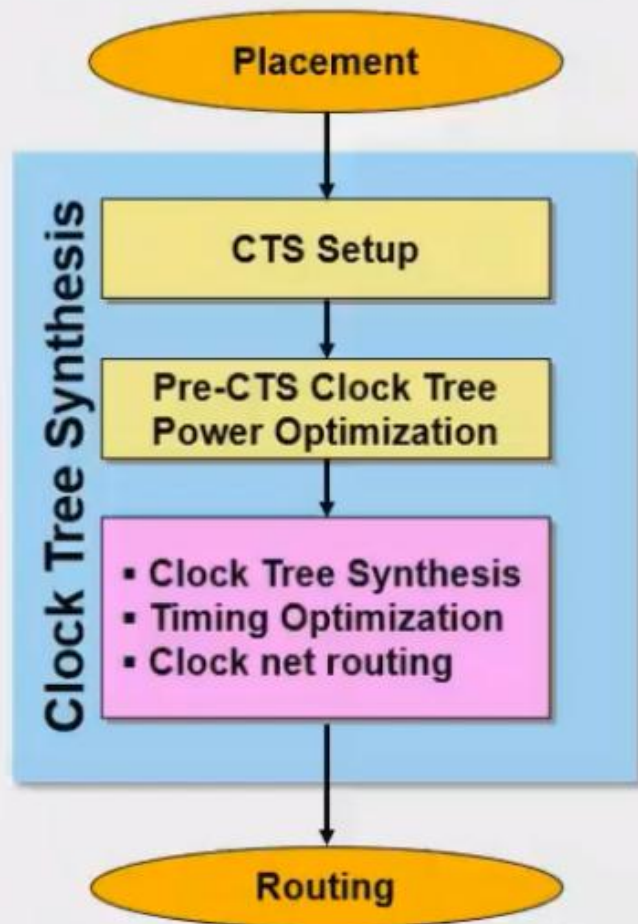
...or to a region using
XY coordinates

```
refine_placement  |-coordinate {X1 Y1 X2 Y2}| \
                   |-congestion_effort high|   \
                   |-perturbation_level <high|max>|
```


- Performs incremental timing-driven logic optimization with placement legalization (setup timing, by default)
 - With `-congestion` will try to maintain or improve congestion
- Use `-power`, `-congestion` or `-area` as necessary
 - If timing is higher priority try omitting one or more of these options
- Sometimes better timing is obtained by focusing or limiting `psynopt` with `-no_design_rule`, `-only_design_rule` or `-size_only`
- It is not necessary to discard previous results before executing a subsequent `psynopt`

```
psynopt | -power | | -area_recovery | | -congestion |  
psynopt -no_design_rule | -only_design_rule | | -size_only |
```

IC Compiler Clock Tree Synthesis Flow



The “CTS phase” involves several key steps:

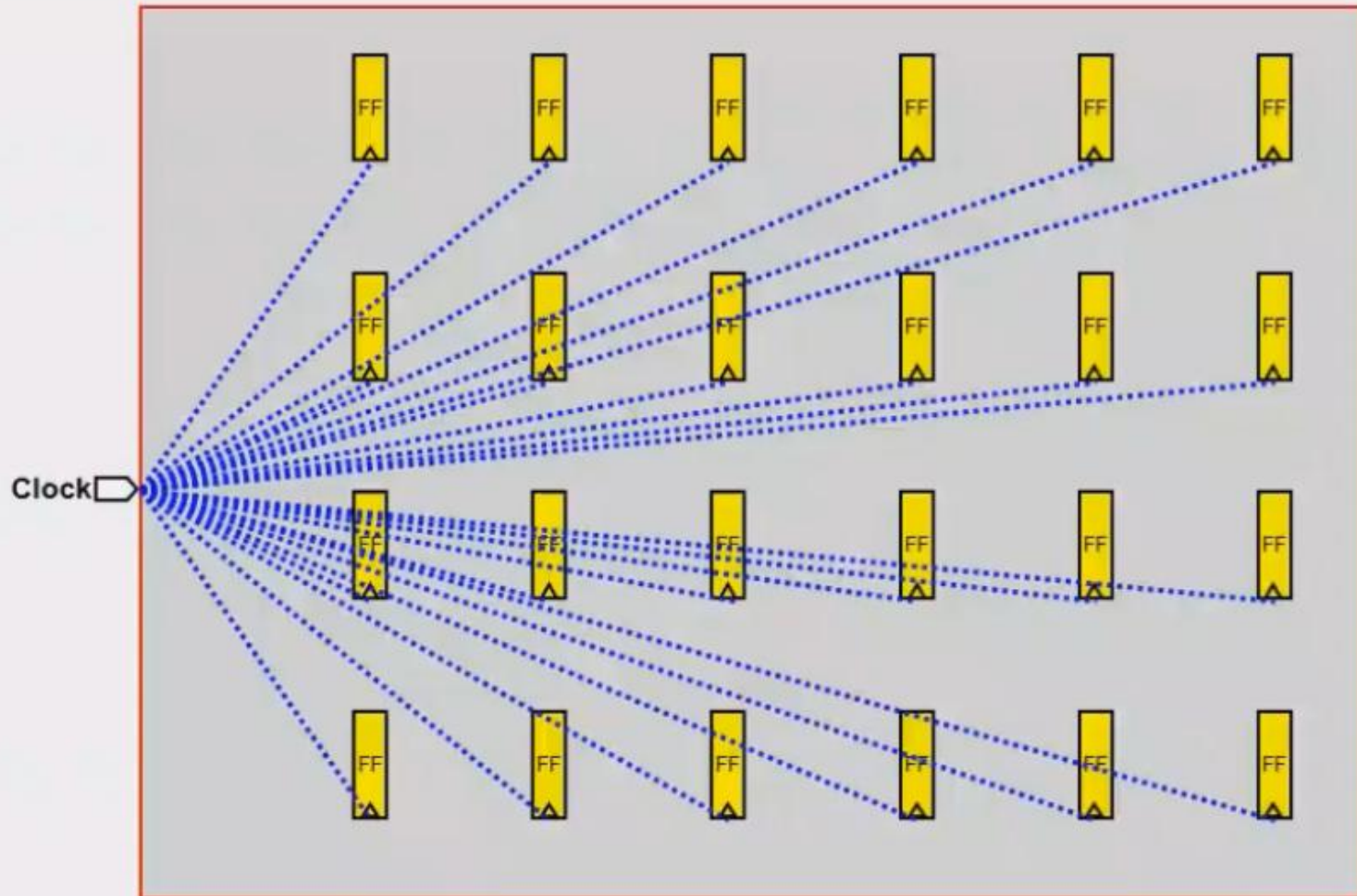
- Setup steps to control CTS
- Optional Pre-CTS power optimization
- Clock Tree Synthesis
- Timing Optimization
- Routing of clock nets

Note: The flow diagrams included in this unit represent an **example** flow, not the recommended flow

Design Status Prior to Clock Tree Synthesis

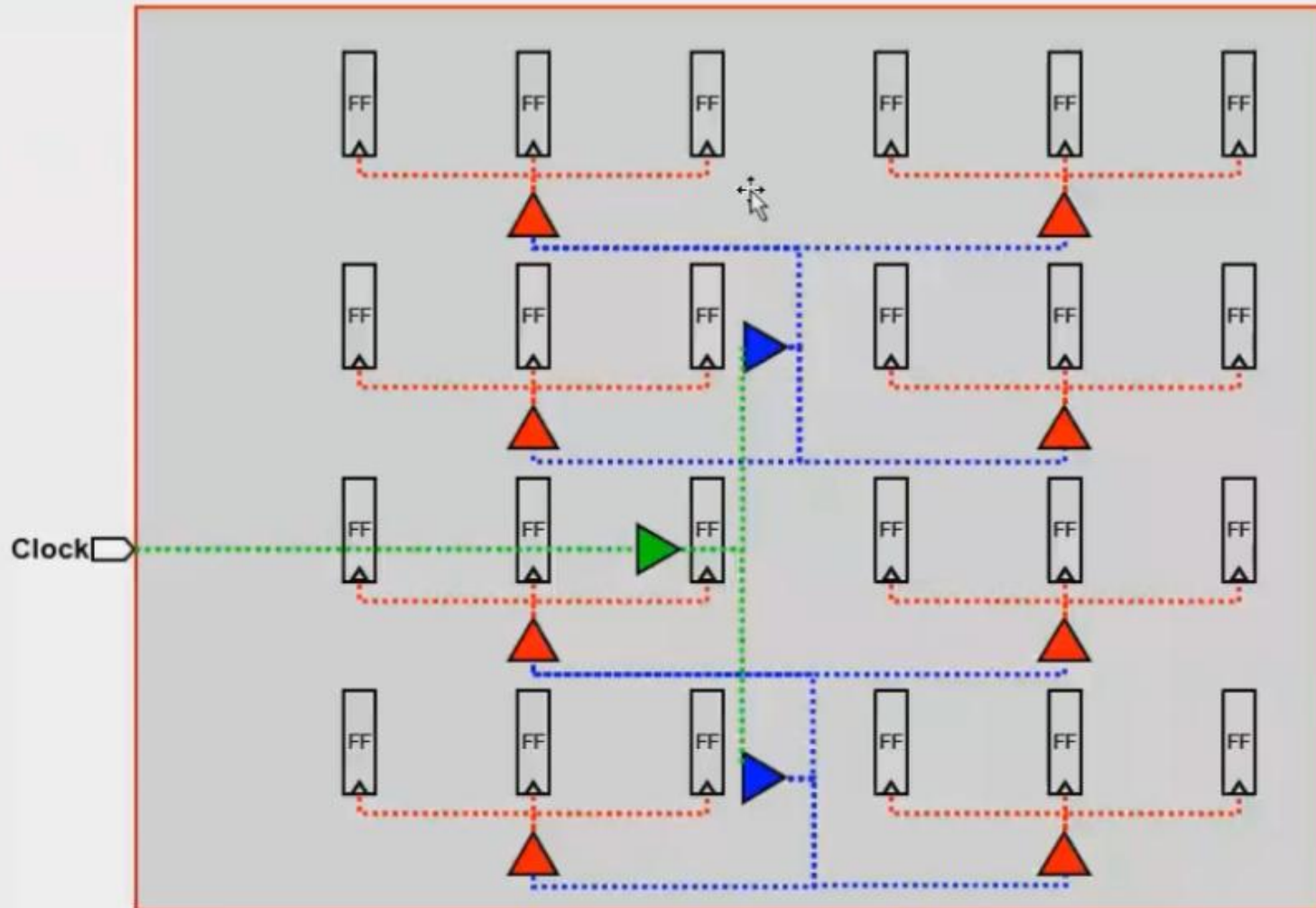
- Placement - completed
- Power and ground nets – prerouted
- Estimated congestion – acceptable
- Estimated setup timing – acceptable (~0ns slack)
- Estimated max cap/transition – no violations
- High fanout nets:
 - Reset, Scan Enable synthesized with buffers
 - Clocks are still not buffered

Starting Point before CTS



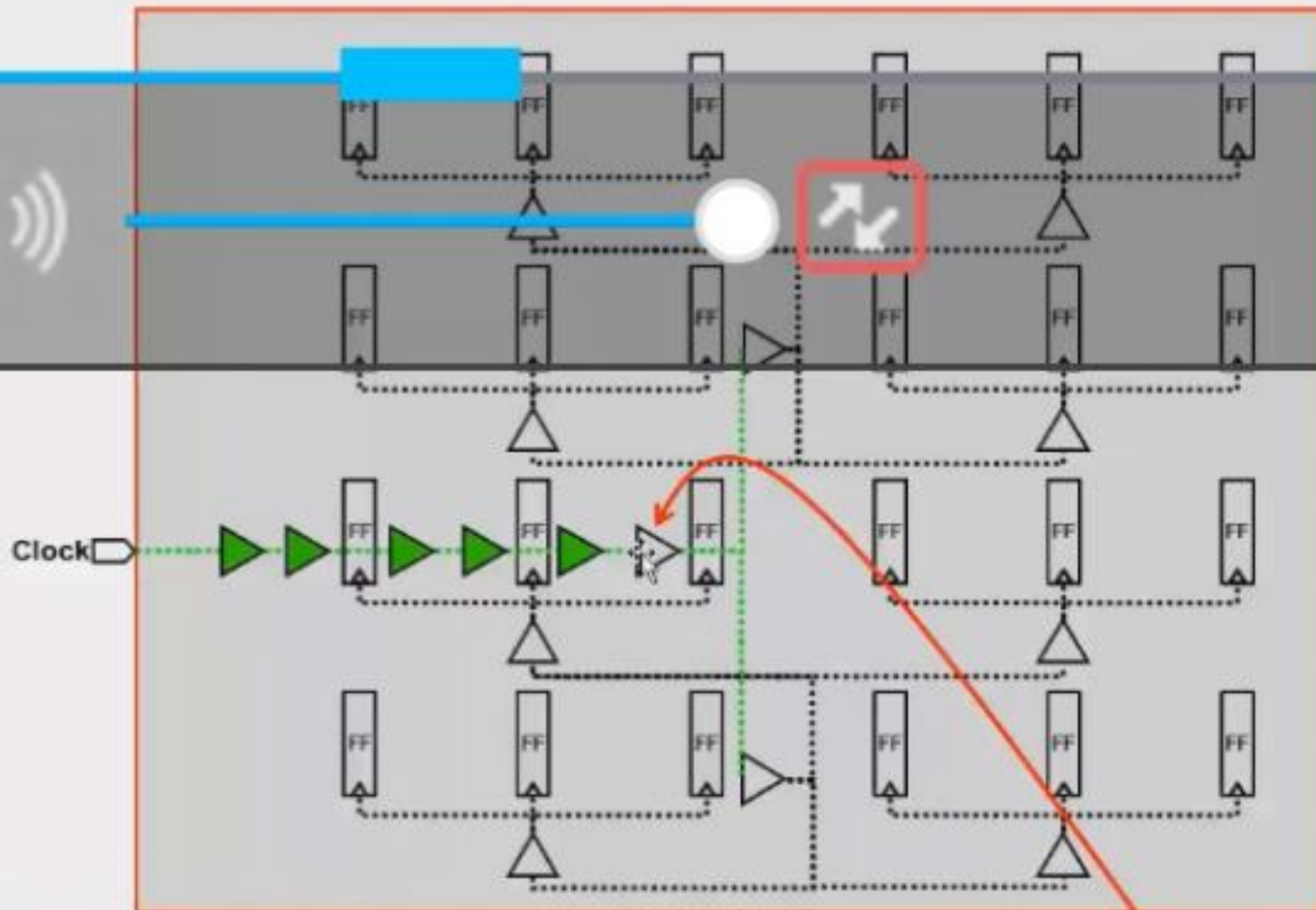
All clock pins are driven by a single clock source.

Clock Cells Are Inserted and then Resized



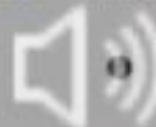
**Buffers are inserted to balance the loads,
meet DRCs and minimize the skew**

Delay Cells Are Added to Meet Min. Insertion



Delay cells are placed behind the common single buffer to minimize clock skew impact

■ Meet the clock tree Design Rule Constraints (DRC):

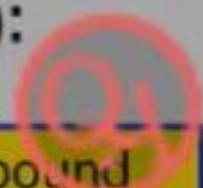


maximum transition delay

- Maximum load capacitance
- Maximum fanout
- Maximum buffer levels



Constraints are upper bound goals. If constraints are not met, violations will be reported.



■ Meet the clock tree targets:

- Maximum skew
- Minimum insertion delay



Targets are "nice to have" goals. If targets are not met, no violations will be reported.

Default Clock Tree Targets

Targets



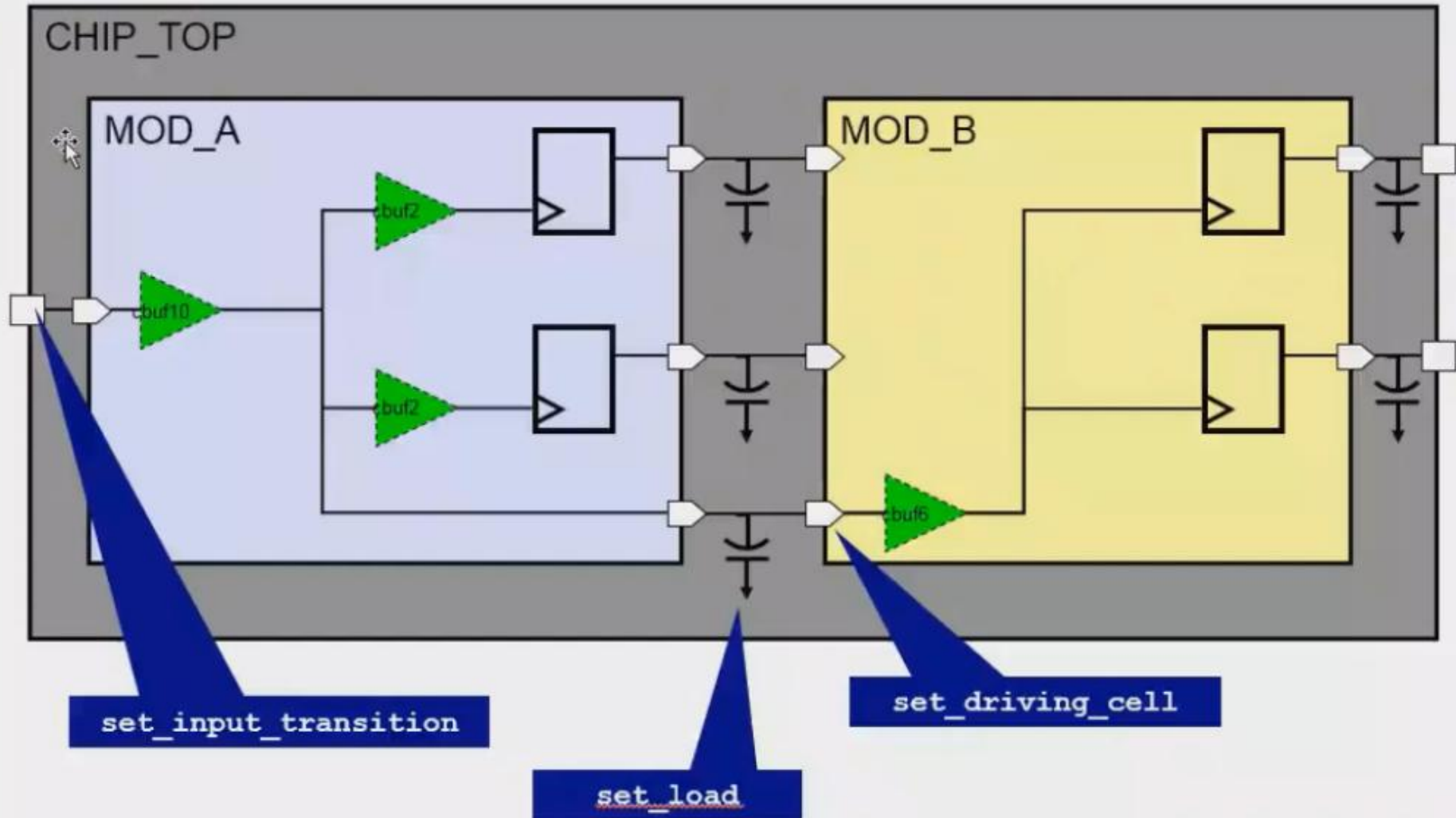
The default CTS target for skew and insertion delay is 0ns



- Uncertainty and insertion delay SDC constraints are ignored
- It is recommended to relax the clock skew target as much as possible
 - Reduces overall buffer count and run time
- Specify minimum clock latencies as needed

Are all Clock Drivers and Loads Specified?

Constraints

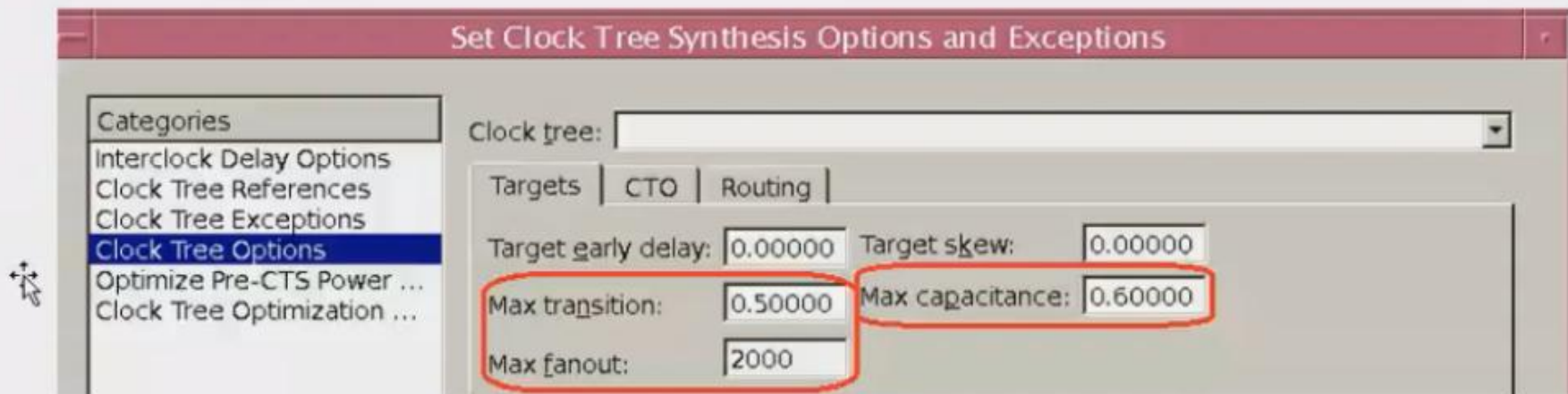


Remove “Skew” from *Uncertainty*

- Your SDC constraints will most likely include a `set_clock_uncertainty -setup <number>` applied to each clock
 - This command is used to model estimated *clock skew*, but can also be used to model the effects of *clock jitter* and to include some additional *timing margin*
 - The specified setup number reduces the effective clock period of all paths captured by the specified clock and is used during synthesis to estimate clock behavior
- Timing analysis post-CTS will also include the effects of this command, therefore:
 - Remove clock uncertainty if only skew is included
`remove_clock_uncertainty [all_clocks]` OR
 - Reduce the uncertainty number by the estimated skew

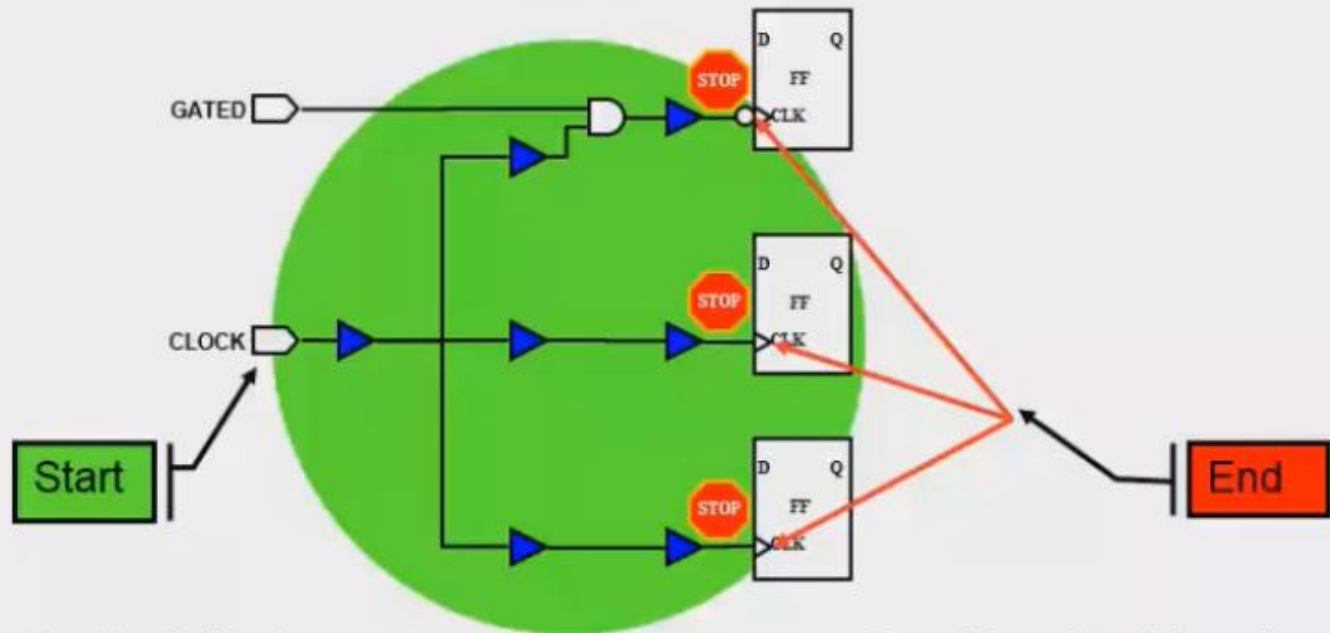
Defining CTS-Specific DRC Values

- **Max transition and max capacitance design rules can be specified in three ways: Library, SDC and CTS setting**
 - By default IC Compiler will use the smallest of the three
- **The default CTS settings are set for today's technologies**
 - If using >> 90nm you may need to relax (increase) the numbers



Where Does the Clock Tree Begin and End?

Control



Clock trees start at their
“source” defined by
create_clock ...

Synthesizable clock
trees end (by default)
on clock pins of
registers or macros
(*stop* or *float* pins)

Stop, Float and Exclude Pins

■ Stop Pins:

- CTS optimizes for DRC and clock tree targets (skew, insertion delay) to the *external* clock pin

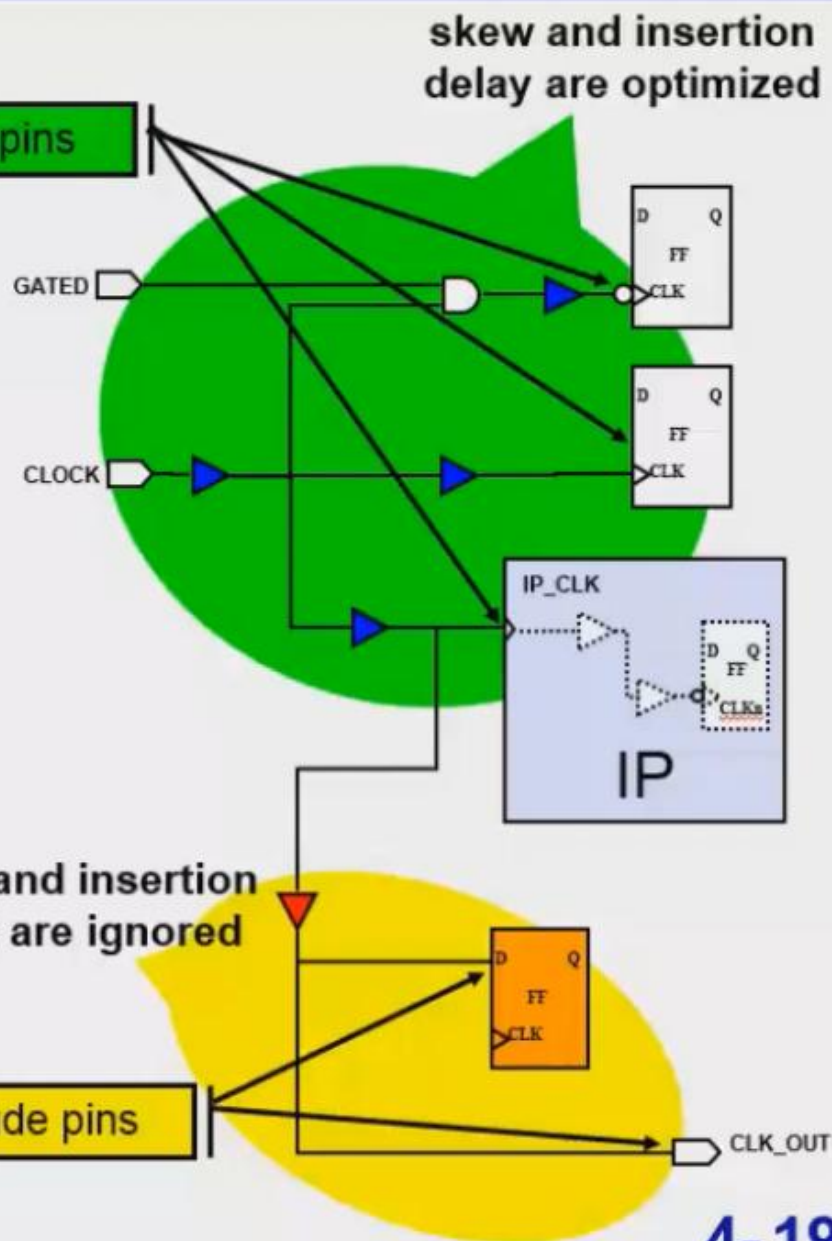
■ Float Pins:

- Like Stop pins, but considers *internal* clock latency

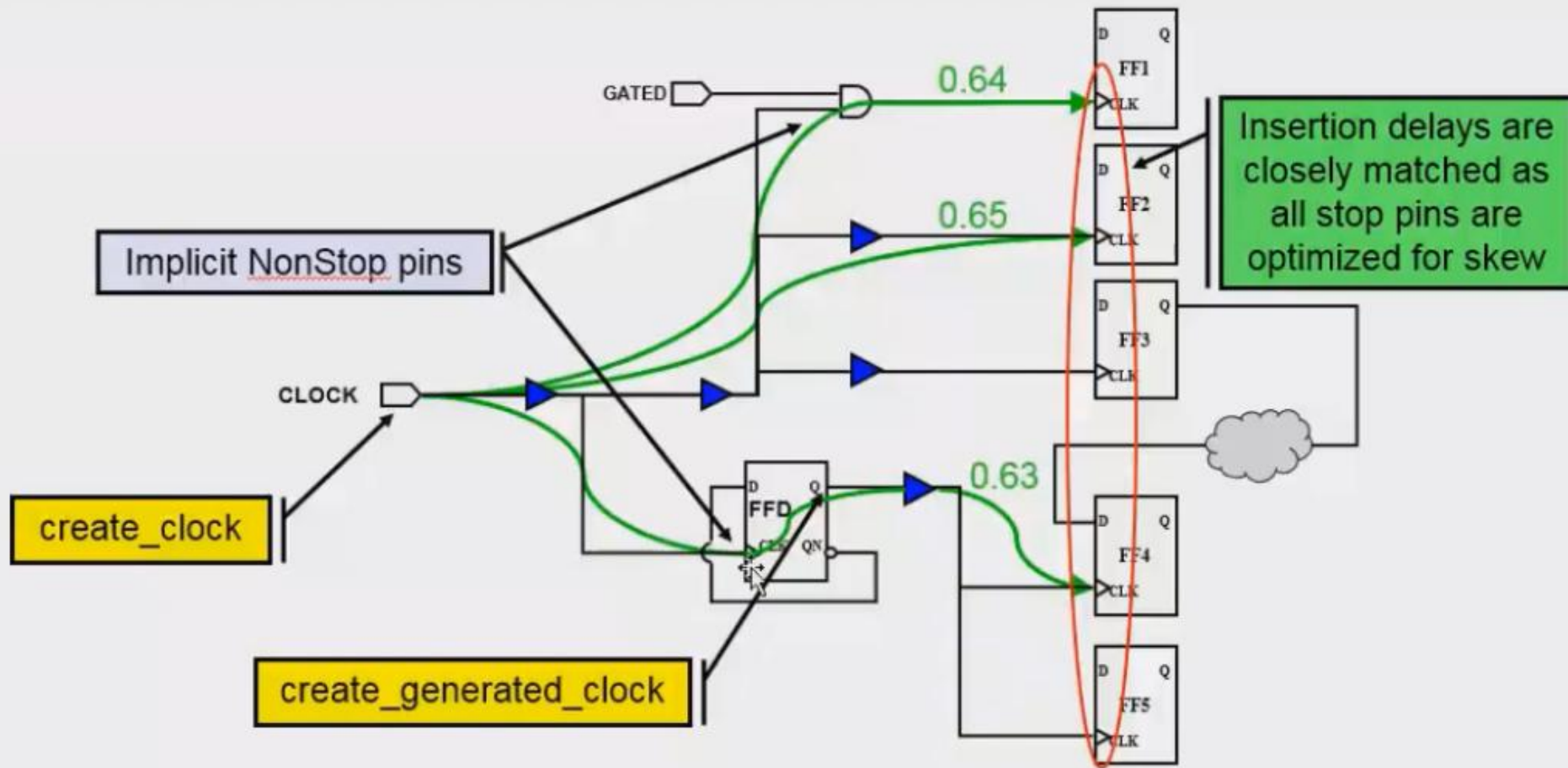
■ Exclude (Ignore) Pins:

- CTS ignores skew and insertion delay targets
- CTS fixes clock tree DRCs

Implicit Stop or Float pins



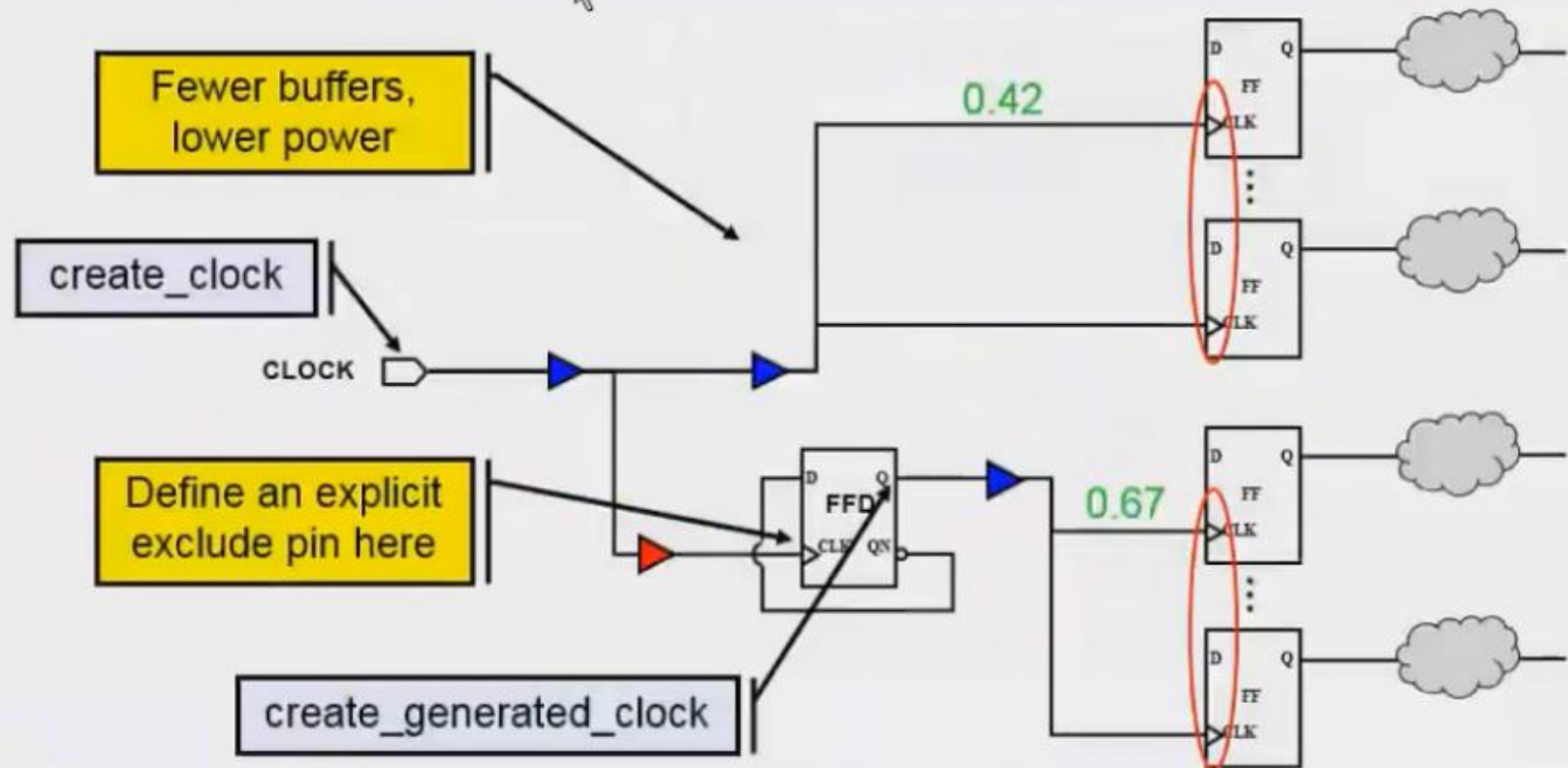
Generated and Gated Clocks



Skew will be balanced 'globally', within each clock domain, across all clock-pins of both master and generated clock.

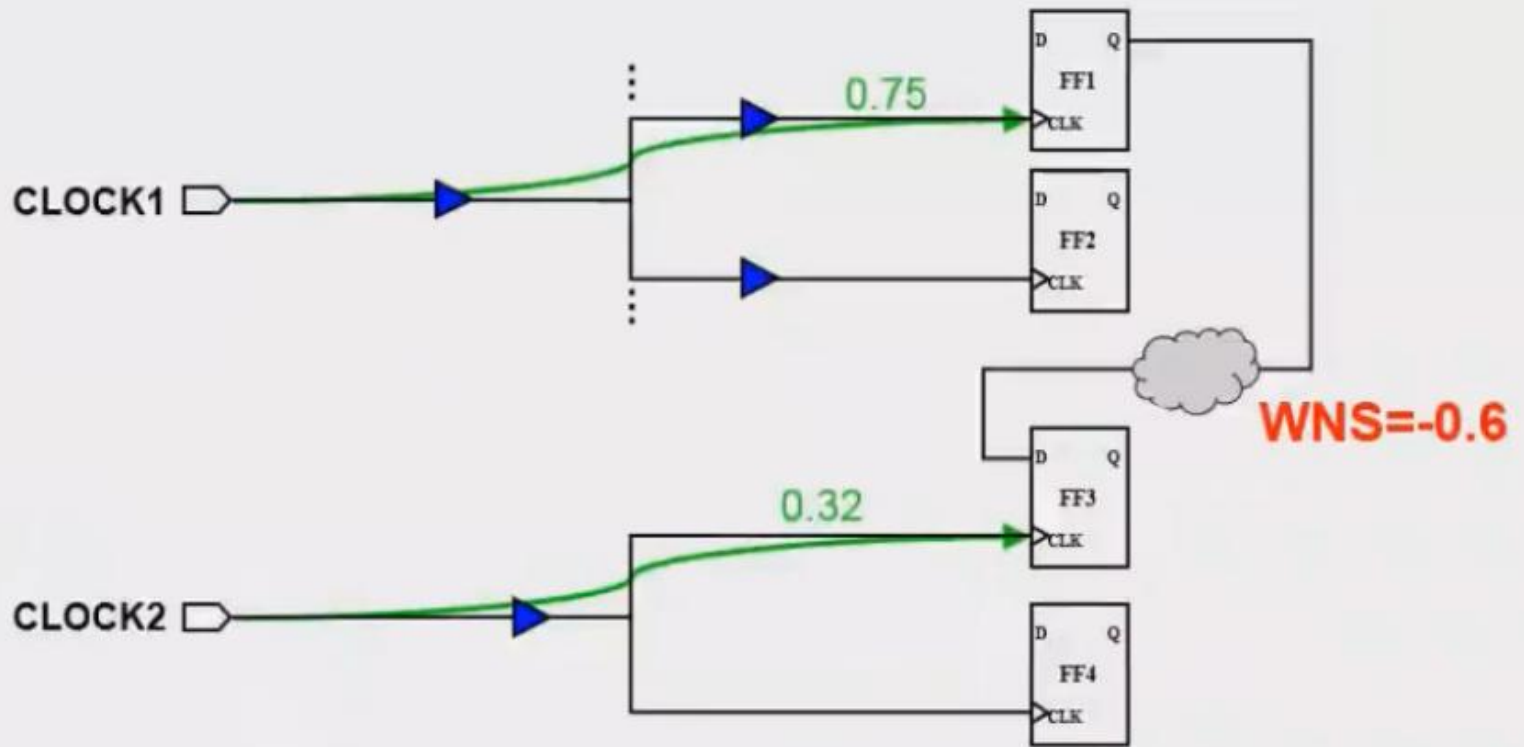
Skew Balancing not Required?

If a generated clock domain is independent of the master domain (no crossing paths) skew balancing may not be important



```
set_clock_tree_exceptions -exclude_pins [get_pins FFD/CLK]
```

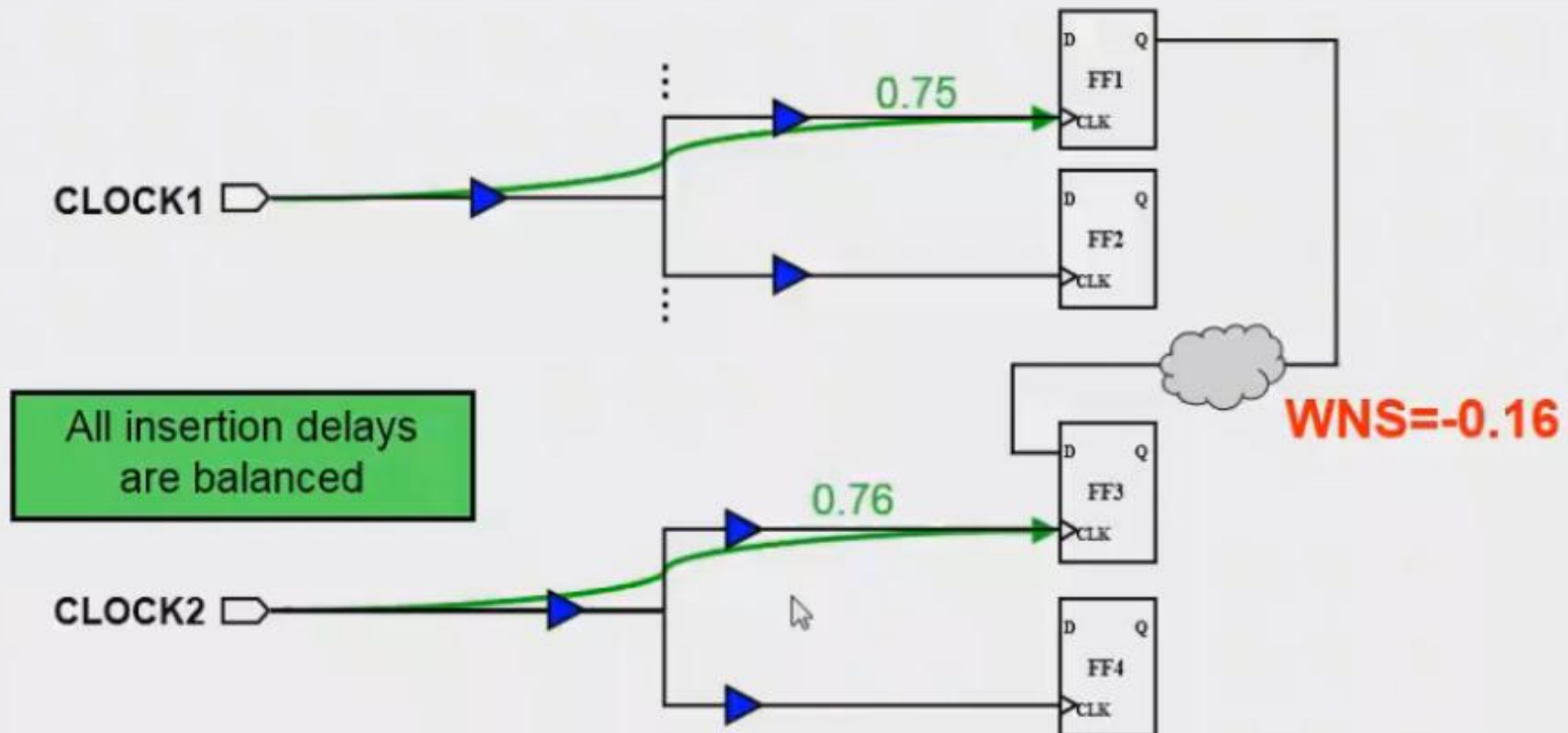
No Inter-Clock Skew Balancing by Default



By default CTS does not perform inter-clock skew balancing → May result in worse setup timing violations

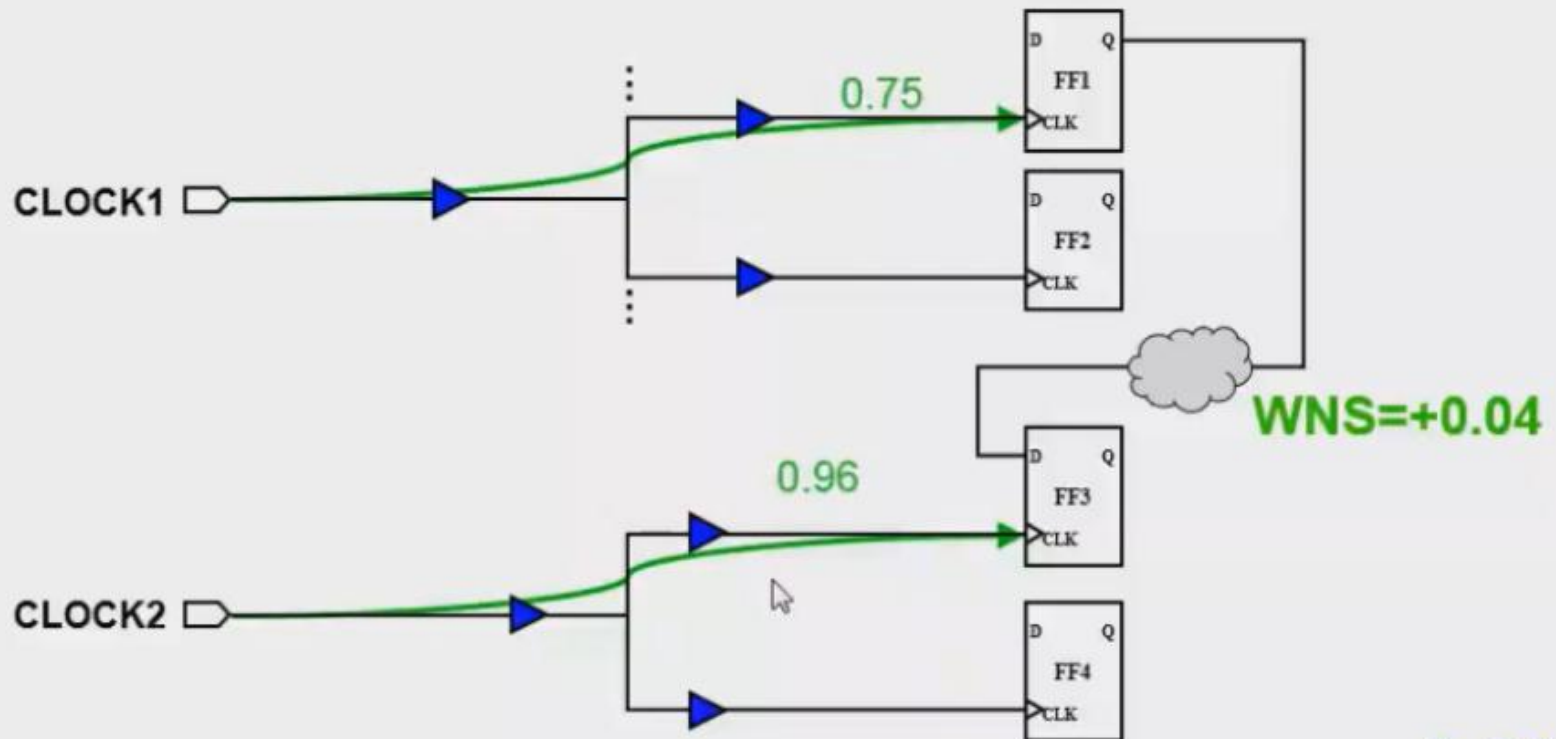
Setup for Inter-Clock Delay Balancing

```
# Balances skew between specified clocks  
set_inter_clock_delay_options \  
    -balance_group "CLOCK1 CLOCK2"  
clock_opt -inter_clock_balance
```



Setup for Inter-Clock Delay Balancing: Offset

```
set_inter_clock_delay_options \  
  -offset_from CLOCK1 -offset_to CLOCK2 \  
  -delay_offset 0.2  
clock_opt -inter_clock_balance
```



SDC Latencies



Does CTS respect SDC `set_clock_latency`?

- CTS does not respect SDC latencies by default!
- If you need your minimum insertion delays to match the SDC specified latencies:

```
set_inter_clock_delay_options -honor_sdc true  
clock_opt -inter_clock_balance
```

I

- Note: Insertion delay will not be minimized if given SDC latency is less than initial CTS insertion delay

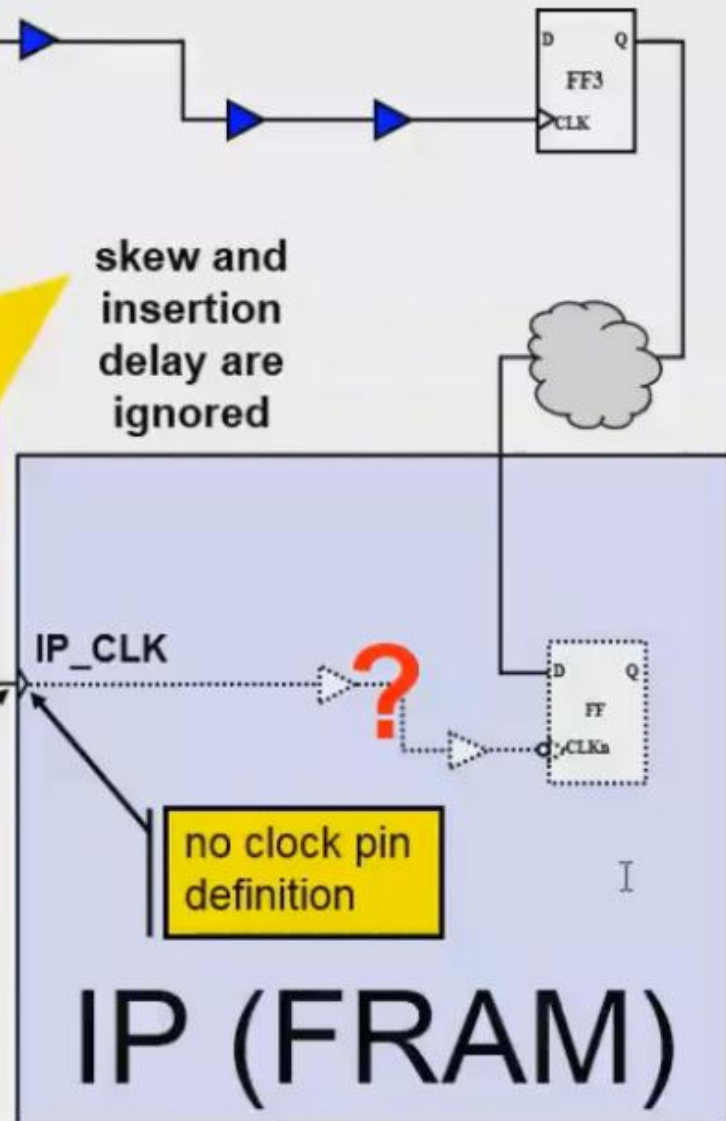
User-defined or Explicit Stop Pins

Scenario: If a macro cell clock pin is defined, CTS will treat that pin as an implicit stop pin. In this example the clock pin is not defined. What is the problem here?

Implicit exclude pin

skew and insertion delay are ignored

The macro's clock pin is marked as an implicit exclude pin – no skew optimization!

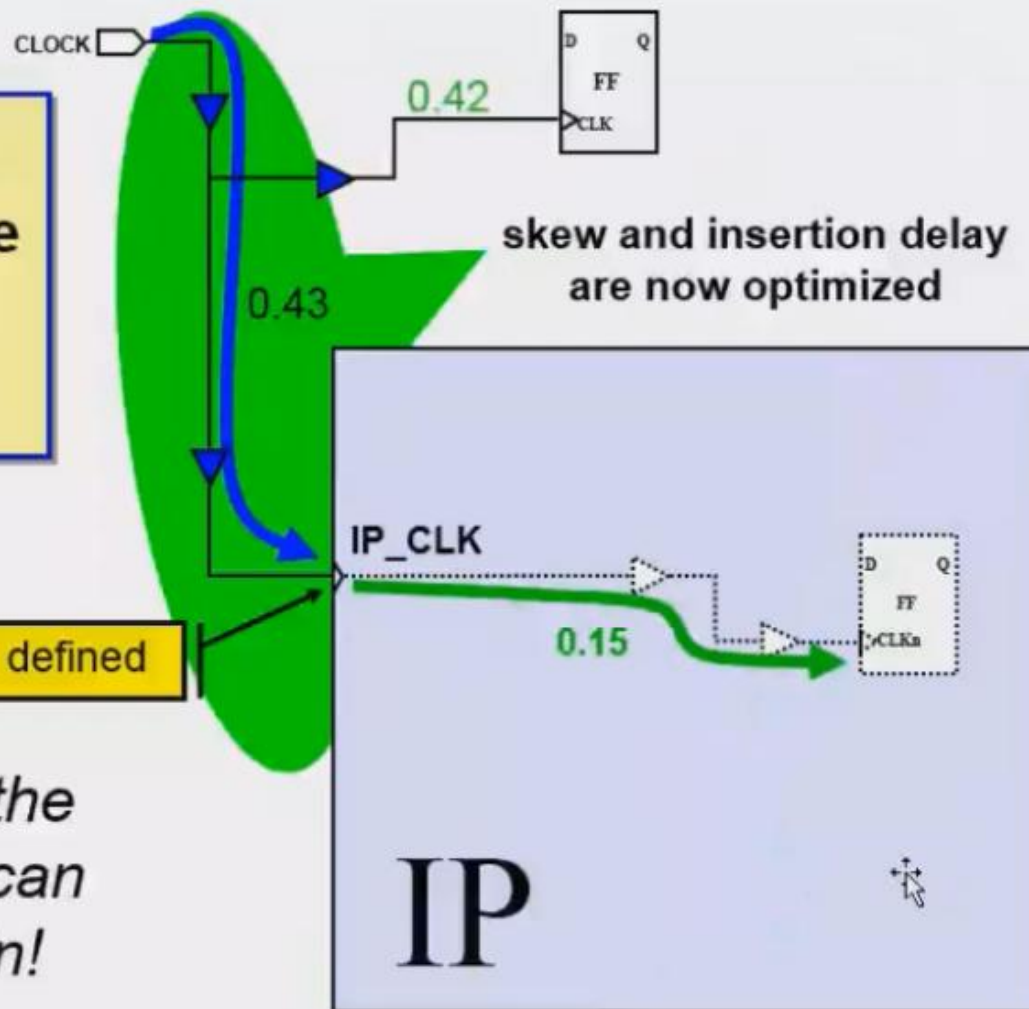


Defining an Explicit Stop Pin

Defining an explicit stop pin allows CTS to optimize for skew and insertion delay targets.

Explicit stop pin defined

CTS has no knowledge of the IP-internal clock delay – it can only “see” up to the stop pin!



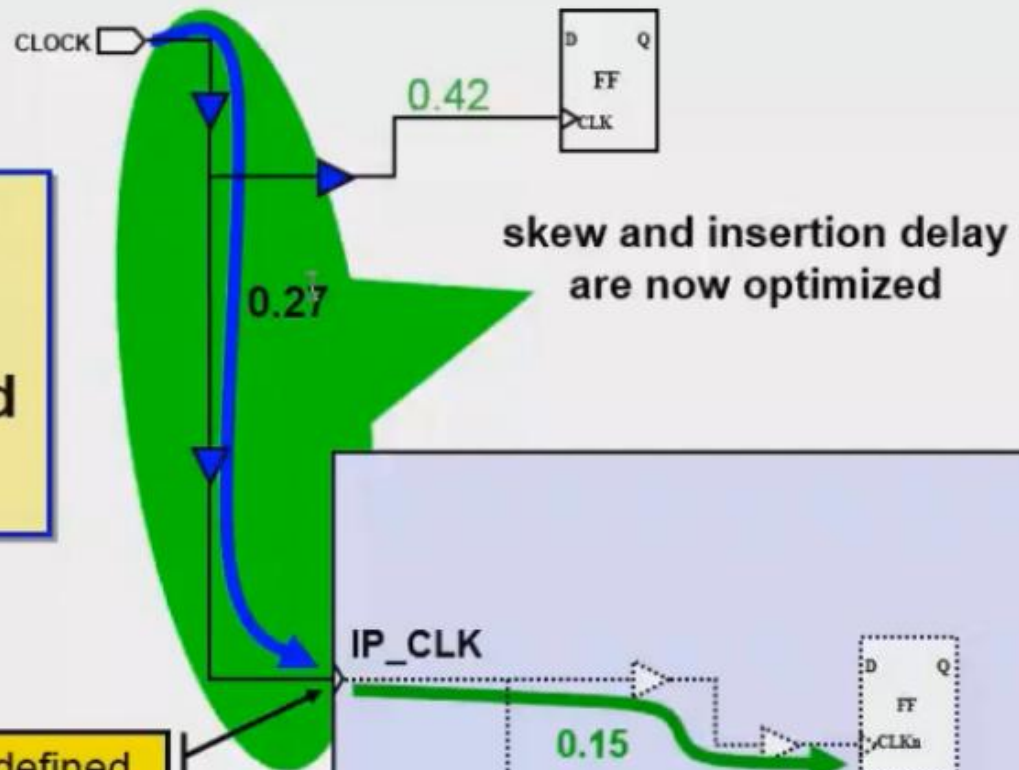
```
set_clock_tree_exceptions -stop_pins [get_pins IP/IP_CLK]
```

Defining an Explicit Float Pin

Defining an explicit float pin allows CTS to adjust the insertion delays based on specification.

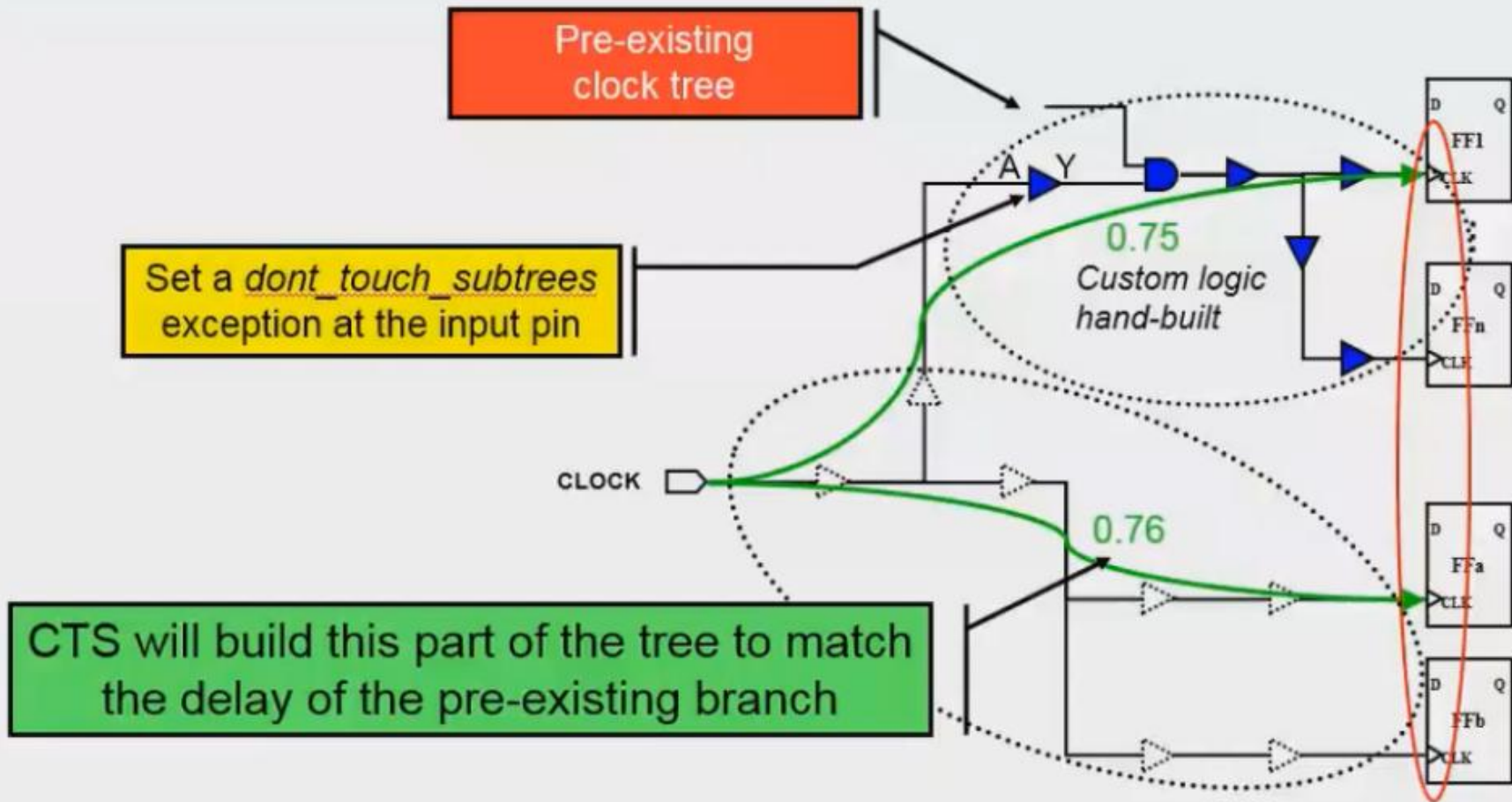
Explicit float pin defined

```
set_clock_tree_exceptions \  
-float_pins IP/IP_CLK \  
-float_pin_max_delay_rise 0.15
```



IP

Preserving Pre-Existing Clock Trees

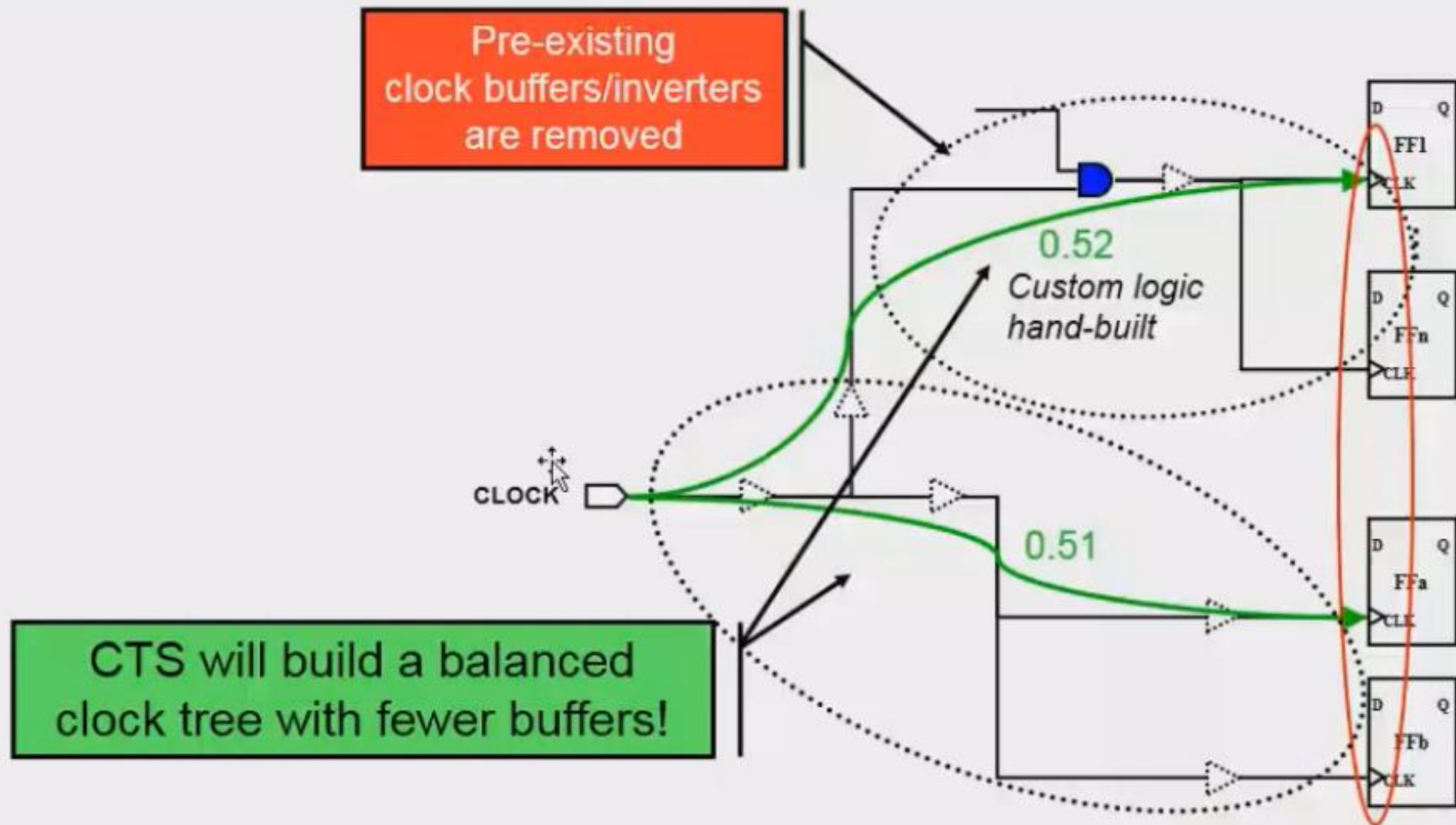


CTS will build this part of the tree to match the delay of the pre-existing branch

```
set_clock_tree_exceptions -dont_touch_subtrees buf/A
```

May create unnecessary additional buffers!

Removing Pre-Existing Clock Buffers

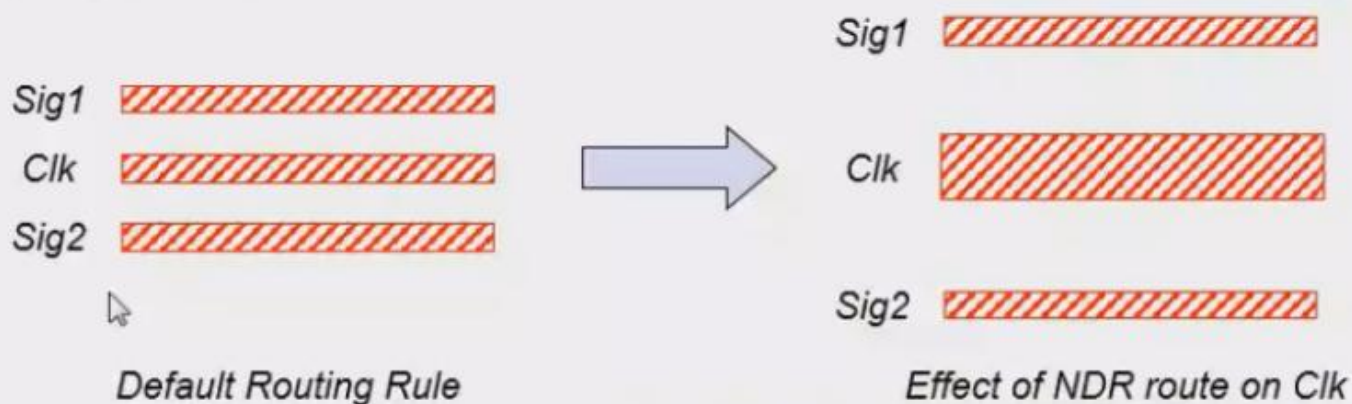


```
remove_clock_tree -clock_tree CLOCK
```

Non-Default Routing Rules

NDRs

- IC Compiler can route the clock nets using *non-default routing* (NDR) rules, e.g. double-spacing, double-width, shielding
- NDR rules are often used to “harden” the clock, e.g. to make the clock routes less sensitive to cross-talk or EM effects

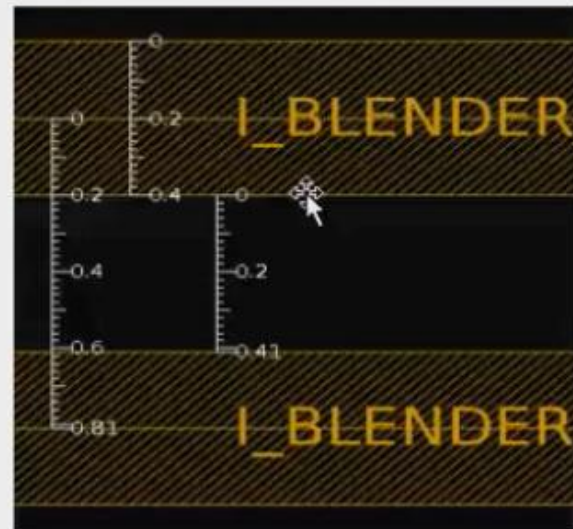


Put NDR on Pitch for Accurate RC Estimation

- Metal traces are always routed “on pitch”
- With clock NDR rules, pre-routing RC estimates of clock nets use NDR width and spacing numbers
- If NDR [spacing + width] numbers are not integer multiples of pitch (i.e. off-pitch), timing estimates pre-route may not correlate well with post-route timing
- Make sure your NDR numbers are on pitch!

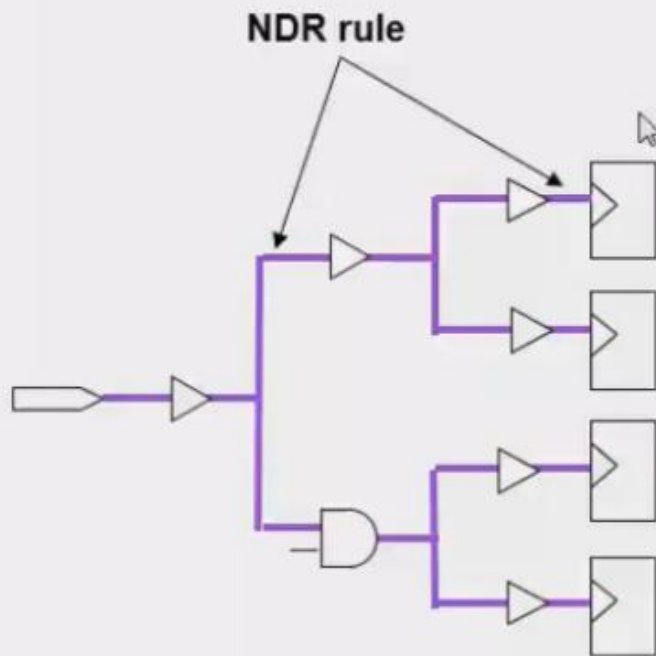
```
Layer      "METAL5" {  
  layerNumber      = 30  
  maskName         = "metal5"  
  pitch            = 0.81  
  defaultWidth     = 0.4  
  minWidth         = 0.38  
  minSpacing       = 0.34
```

For example: For double spacing and double width, do not simply double the minimum DRC numbers and use an NDR width of 0.76 and spacing of 0.68!!



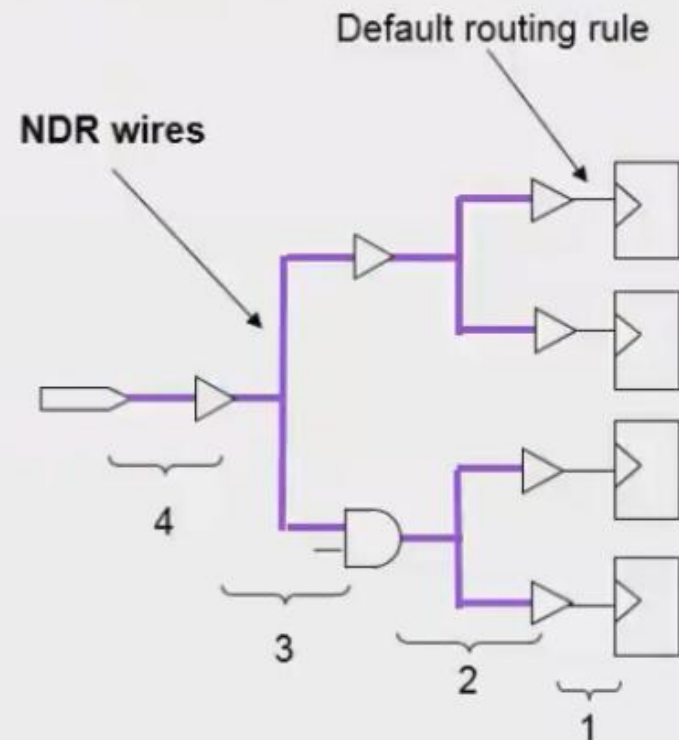
Default Routing Rule for Sink Pins Option

```
set_clock_tree_options \
  -routing_rule my_route_rule
```



Default Behavior

```
set_clock_tree_options \
  -routing_rule my_route_rule \
  -use_default_routing_for_sinks 1
```



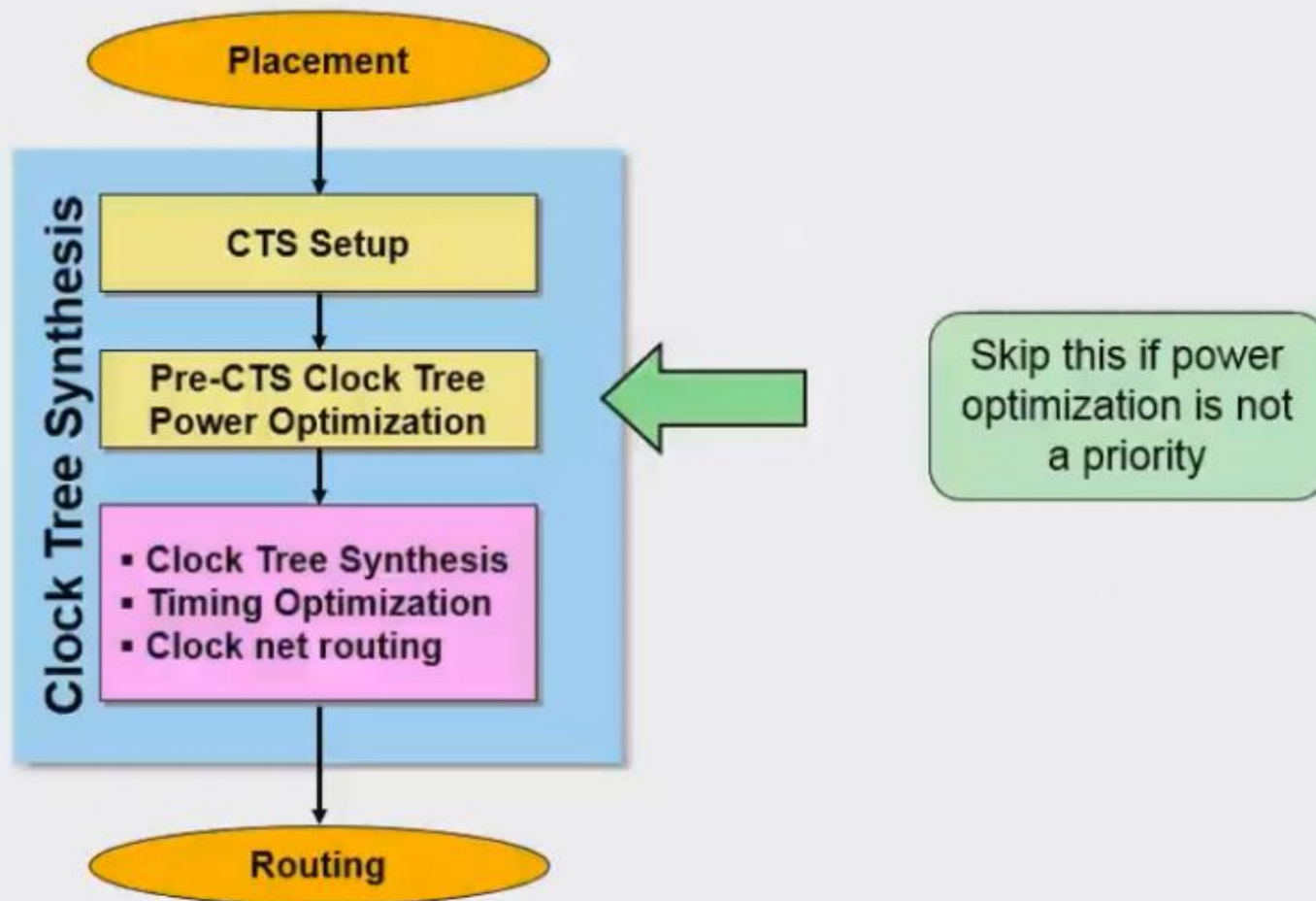
-use_default_routing_for_sinks
can only be used globally!

- Always route clock on metal 3 and above
- Consider using double width to reduce resistance
- Consider using double spacing to reduce crosstalk
- Consider double via insertion to reduce resistance and improve yield
- Avoid NDRs on clock sinks:

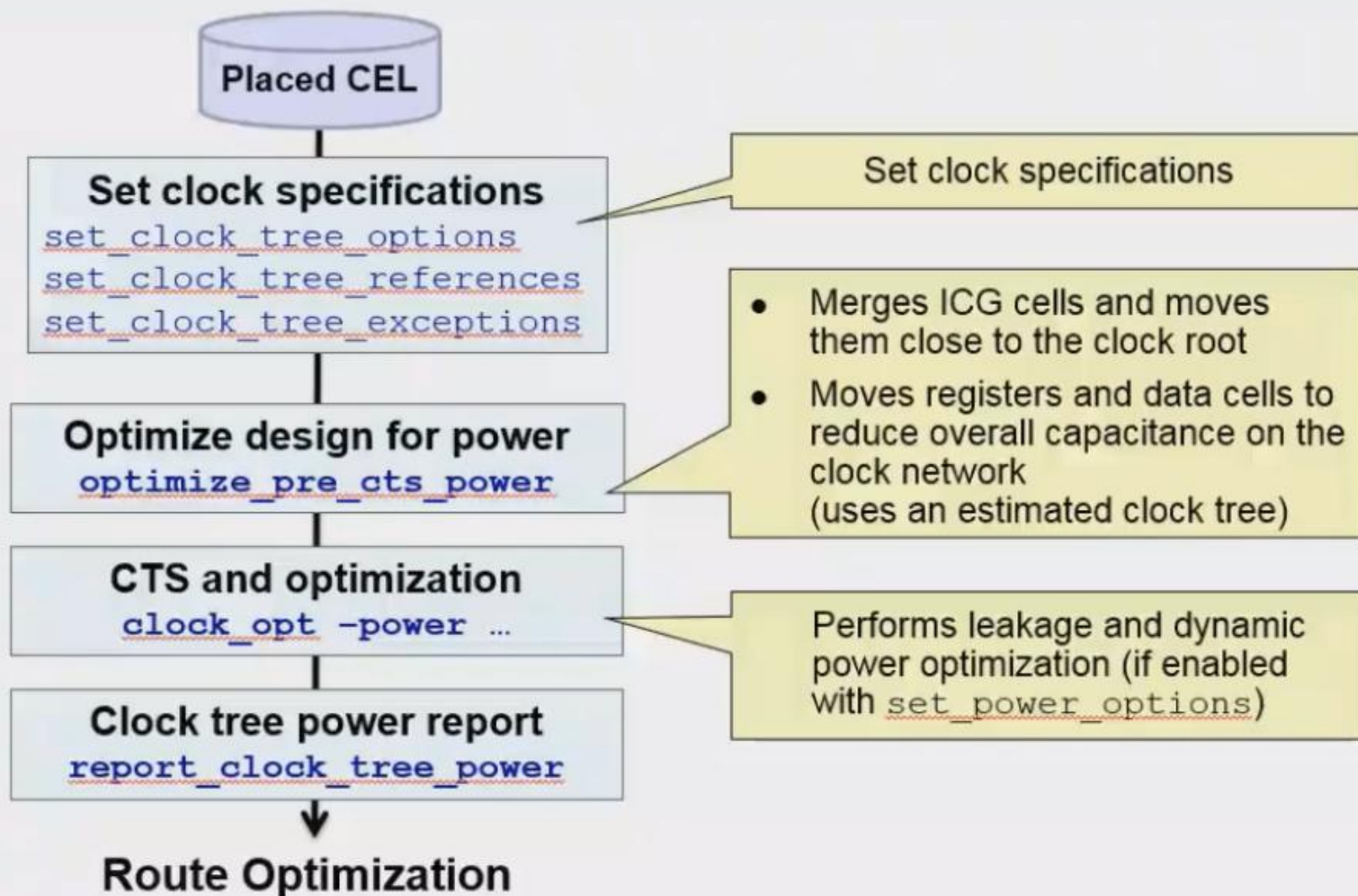
```
set_clock_tree_options -use_default_routing_for_sinks 1
```

- Avoid NDRs on Metal 1
 - May have trouble accessing metal 1 pins on clock buffers and gates
- Put NDR spacing on pitch

Optional Pre-CTS Power Optimization



Clock Tree Power Optimization Pre-CTS



Is the Design Ready for CTS?

- check_physical_design -stage pre_clock_opt checks for:
 - All previous placement requirements
 - Design is placed
 - Clocks have been defined
- check_clock_tree checks and warns if:
 - A generated-clock with improperly specified master-clock
 - A clock tree has no synchronous pins
 - There are multiple clocks per register
 - There is a master-clock that terminates at a pin without a corresponding generated-clock
 - ...more

How Should clock_opt be Executed?

- The mega command clock_opt performs, by default:
 - Synthesis and balancing of individual clock tree networks
 - Timing and DRC optimization of non-clock logic
 - Routing of clock tree network
- Additional options exist to perform:
 - Inter-clock delay balancing
 - Scan-chain re-ordering
 - Power optimization
 - And more ...
- It is recommended to perform clock_opt in three steps

```
clock_opt  
  -only_cts  
  -only_psyn  
  -no_clock_route  
  -inter_clock_balance  
  -optimize_dft  
  -power  
  ...
```


Recommended Three-Step clock_opt Flow

Using clock_opt in the following manner allows early analysis and intervention, which can lead to increased quality of clock tree synthesis results:

```
clock_opt -no_clock_route -only_cts ...  
analyze...
```

```
...  
clock_opt -no_clock_route -only_psyn ...  
analyze...
```

```
route_zrt_group -all_clock_nets ...
```

clock_opt Functionality

	<u>clock_opt</u>	<u>clock_opt</u> \ -only_cts \ -no_clock_route	<u>clock_opt</u> \ -only_psyn \ -no_clock_route	<u>route_zrt_group</u> \ -all_clock_nets
<u>Clock Tree Synthesis</u> Builds initial clock tree by: - Load balancing - Logic-level balancing	✓	✓		
<u>Clock Tree Optimization*</u> Meets clock tree targets by: - Cell sizing - Delay insertion	✓	✓		
<u>Timing/DRC optimization</u> - Optimizes logic and placement of data cells - Clock cells are fixed	✓		✓	
<u>Routing of Clock Nets</u>	✓			✓

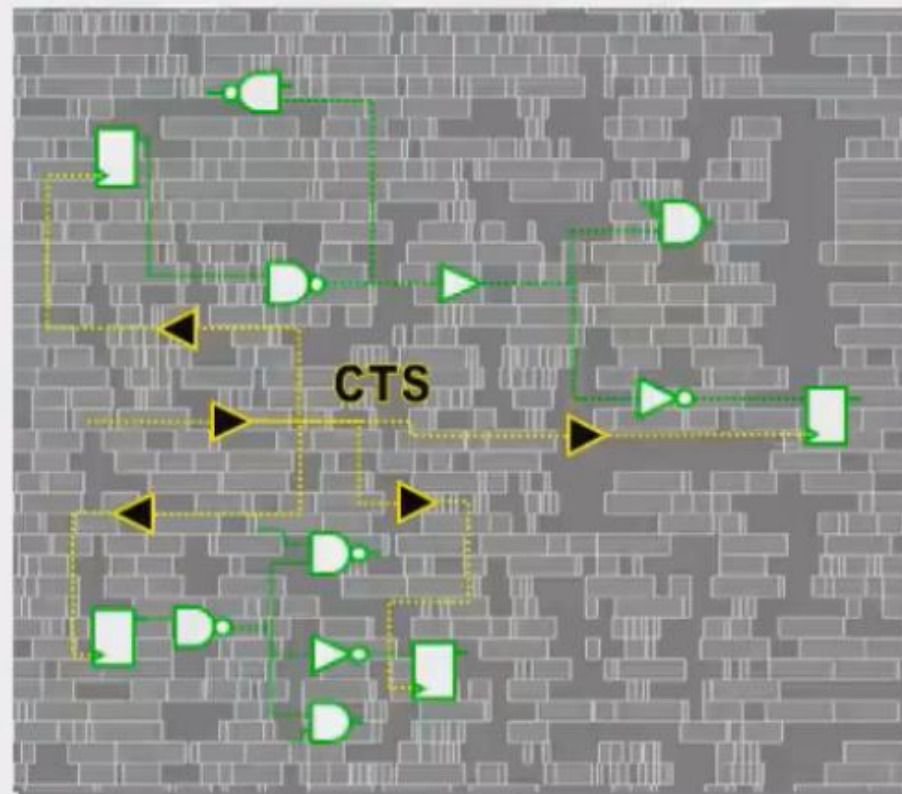


Clock buffers and inverters are fixed (locked down) after clock_opt -only_cts

Effects of Clock Tree Synthesis

```
clock_opt -no_clock_route -only_cts -inter_clock_balance
```

- Builds the clock tree – lots of buffers are added!
- Congestion may increase
- Non clock cells may have been moved to less ideal locations
- Can introduce new timing and max tran/cap violations to non clock paths



How do you analyze clock trees?

Analyzing CTS Results

Details covered in the lab!

■ report_clock_tree

- summary
- settings
- ...

- Reports Max global skew, Late/Early insertion delay, Number of levels in clock tree, Number of clock tree references (Buffers), Clock DRC violations

■ report_clock_timing

- Reports actual, relevant skew, latency, inter-clock latency, etc. for paths that are related.
- Example: report_clock_timing -type skew



How do you handle timing violations?

Enable Hold Time Fixing

- No hold time fixing has been performed on the design up to this point
- Now that the clock tree has been synthesized it is generally considered good practice to enable hold time fixing:



```
set_fix_hold [all_clocks]
```

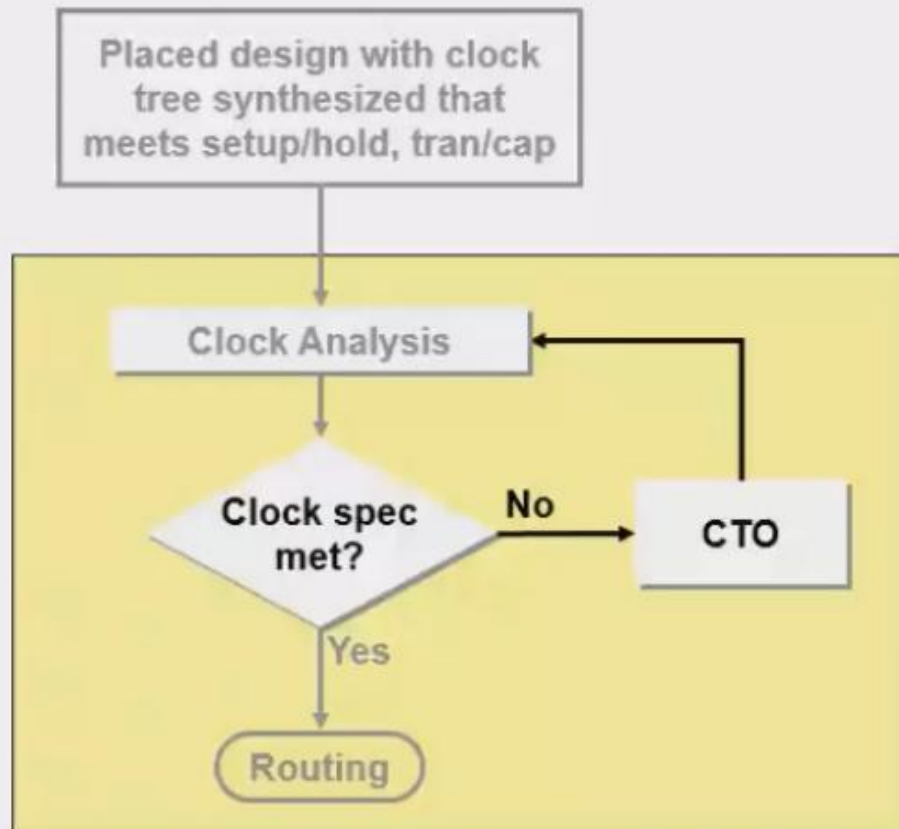
```
extract_rc
```

```
clock_opt -no_clock_route -only_psyn \  
-optimize_dft -area_recovery -power
```

Force global route instead of virtual route extraction for timing optimization during the next step

Stand-Alone Clock Tree Optimization (CTO)

Perform additional Clock Tree Optimization as necessary to further improve clock skew after clock tree synthesis and timing optimization



CTO is run inside `clock_opt`, and can be run independently as well:

`optimize_clock_tree`

Routing Operations of route_opt

■ route_opt performs:

- Global Routing
- Track Assignment
- Detail Routing



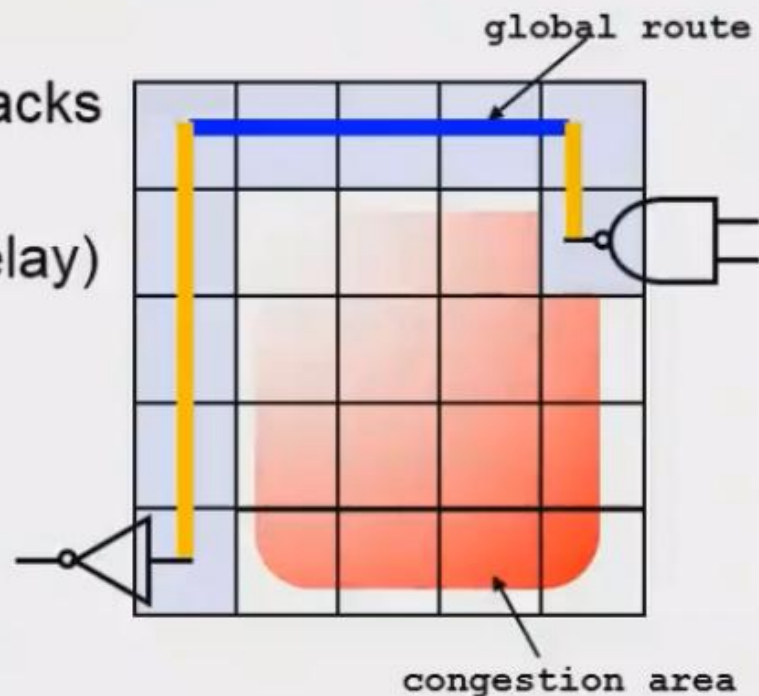
■ After route_opt all nets will be completely connected

■ Also performs concurrent optimization of:

- Timing, area and power
- Buffering DRCs (max_cap/max_transition, etc)
- Physical DRCs

Route Operations: Global Route (GR)

- GR assigns nets to specific metal layers and global routing cells (Gcells)
- GR tries to avoid congested Gcells while minimizing detours:
 - Congestion exists when more tracks are needed than available
 - Detours increase wire length (delay)
- GR also avoids:
 - P/G (rings/straps/rails)
 - Routing blockages

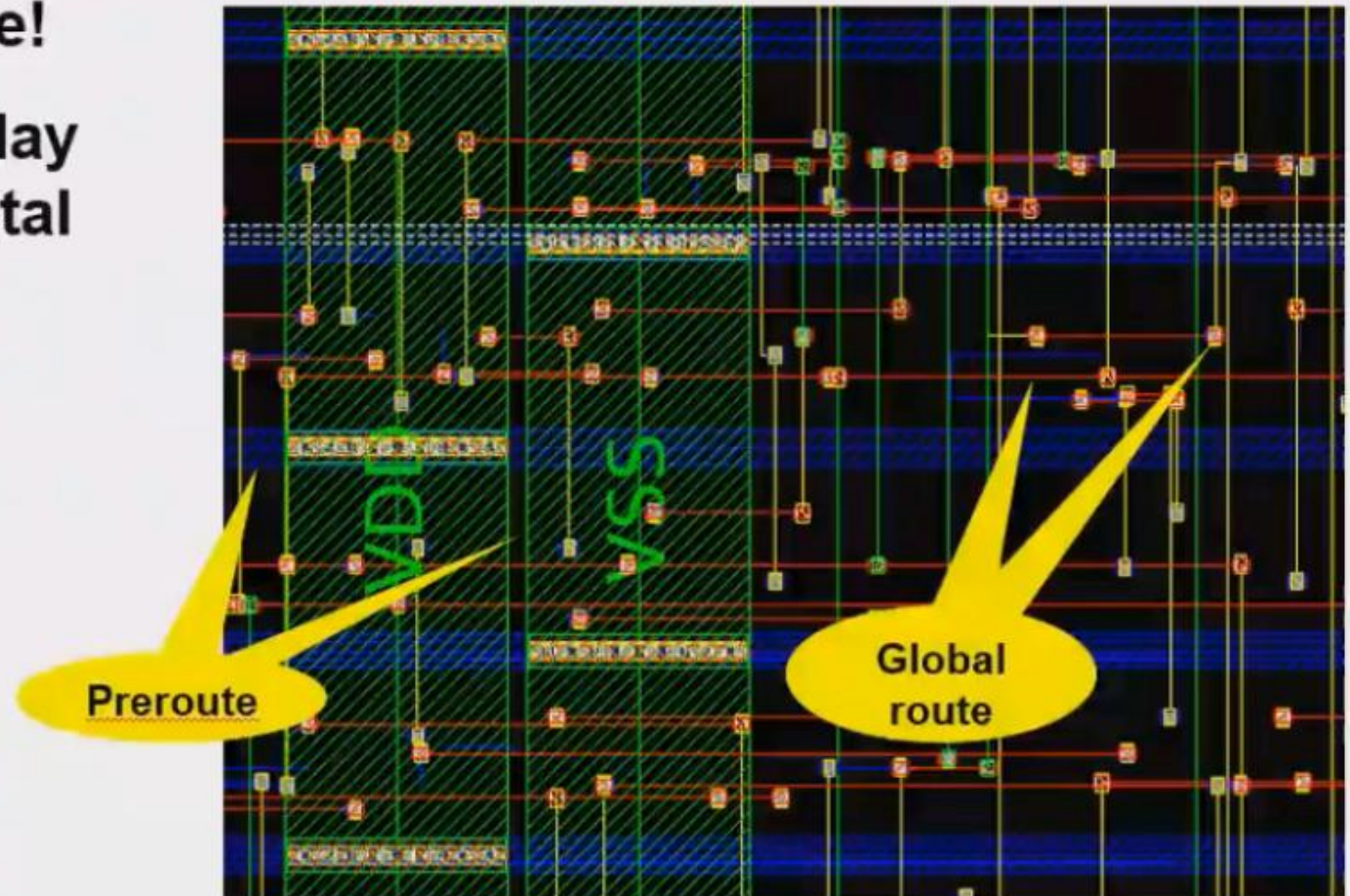


Metal traces exist after Global Route. **True or False?**

Route Operations: Global Route Summary

Answer: False!

GR does not lay down any metal traces.



Route Operations: Track Assignment

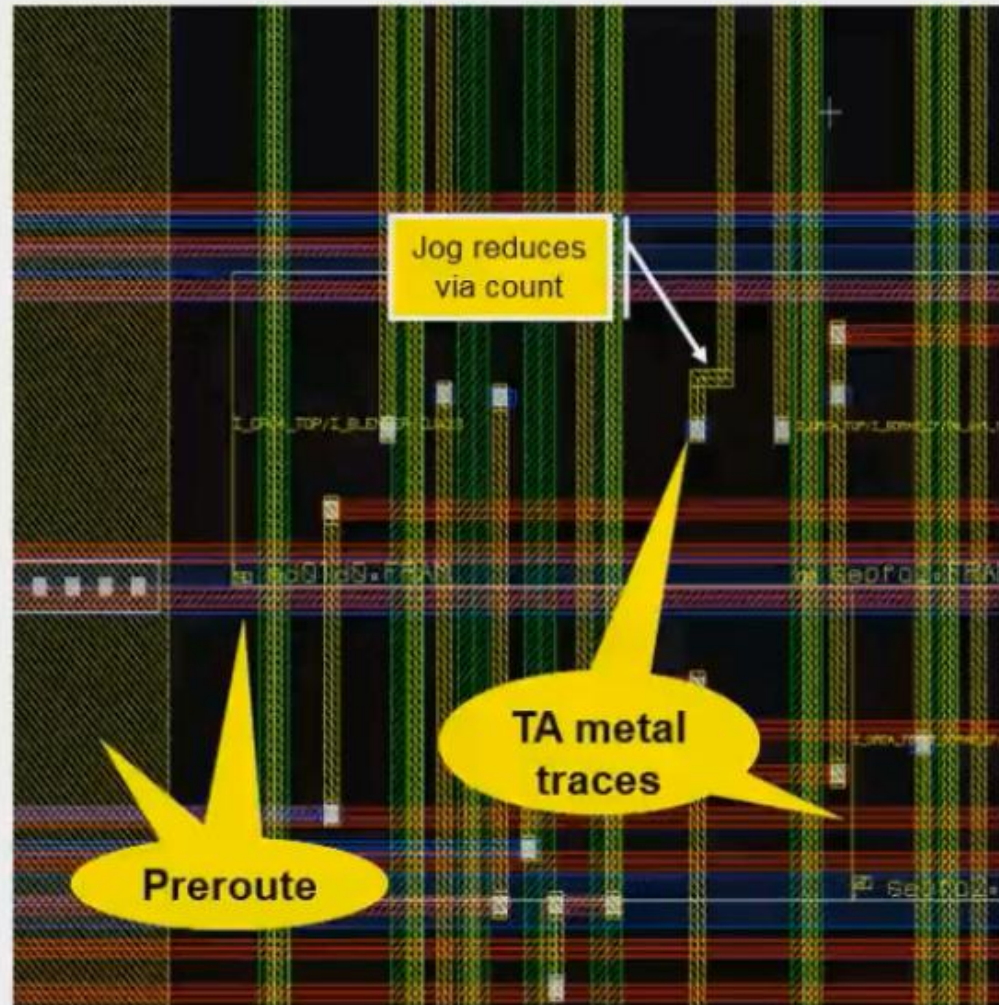
■ Track Assignment (TA):

- Assigns each net to a specific track and lays down the actual metal traces

■ It attempts to:

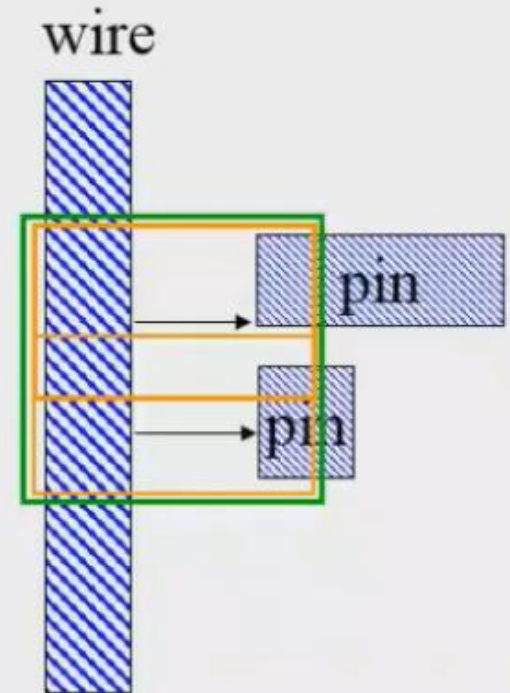
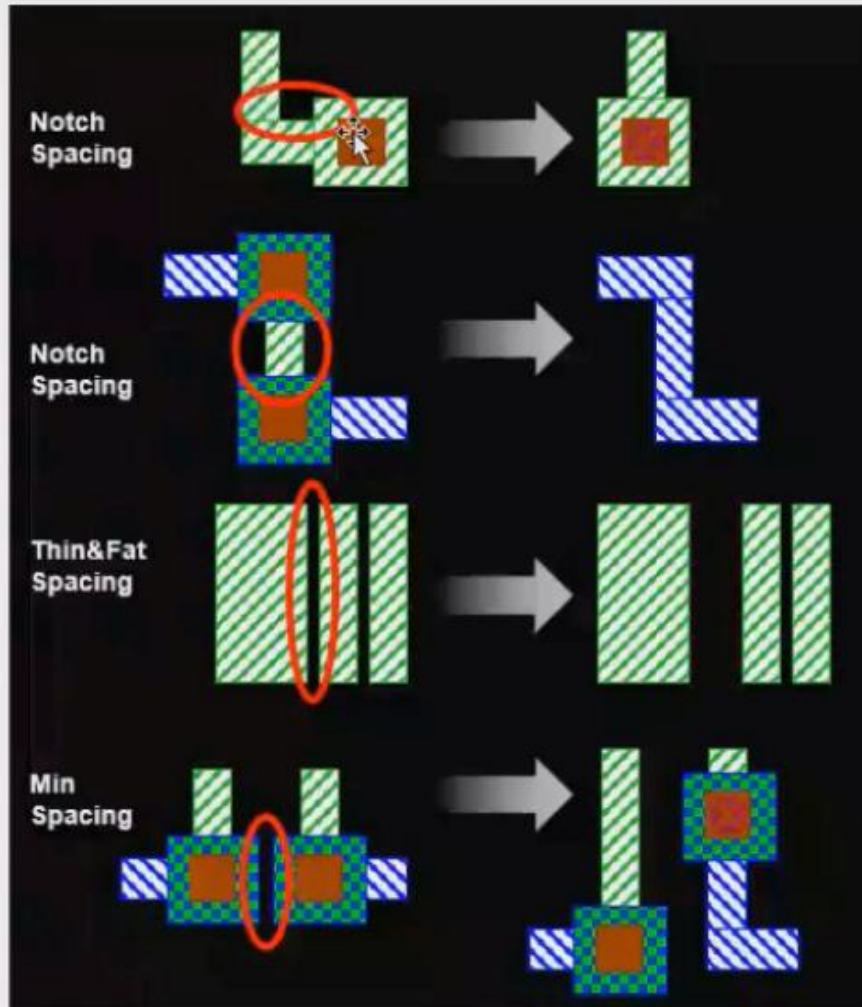
- Route each layer in its preferred direction
- Make long, straight traces
- Reduce the number of vias

■ TA does not check or follow physical design rules



Route Operations: Detail Routing

Detail route fixes physical design rule violations



Violations of identical rules that are in close proximity to each other will be merged and reported as one violation

Design Status, Start of Routing Phase

- Placement - completed
- CTS – completed
- Power and ground nets – pre-routed
- Estimated congestion - acceptable
- Estimated timing - acceptable (~0ns slack)
- Estimated max cap/transition – no violations

```
report_constraints -all
```


Pre-Route Checks

- Check design for routing stage readiness
- There should be no unexpected:
 - Ideal nets
 - High fanout nets
- Use check_physical_design to check a design's prerequisites for routing and report a list of violations

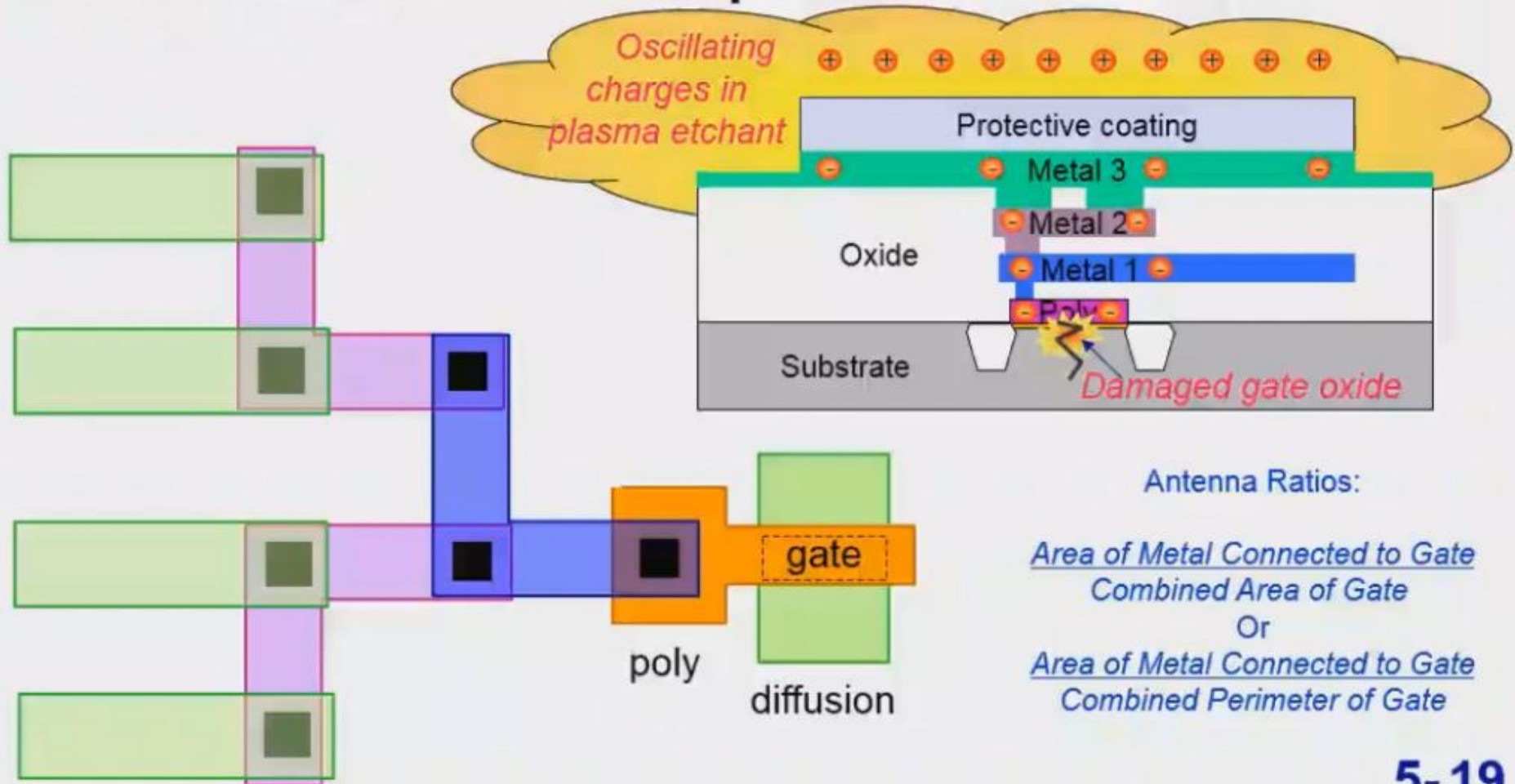
*Fix before
running
Zroute*



```
check_physical_design -stage pre_route_opt  
all_ideal_nets  
# Default high fanout threshold is 1000  
#(default for variable high_fanout_net_threshold)  
all_high_fanout -nets <-threshold #>  
report_preferred_routing_direction
```

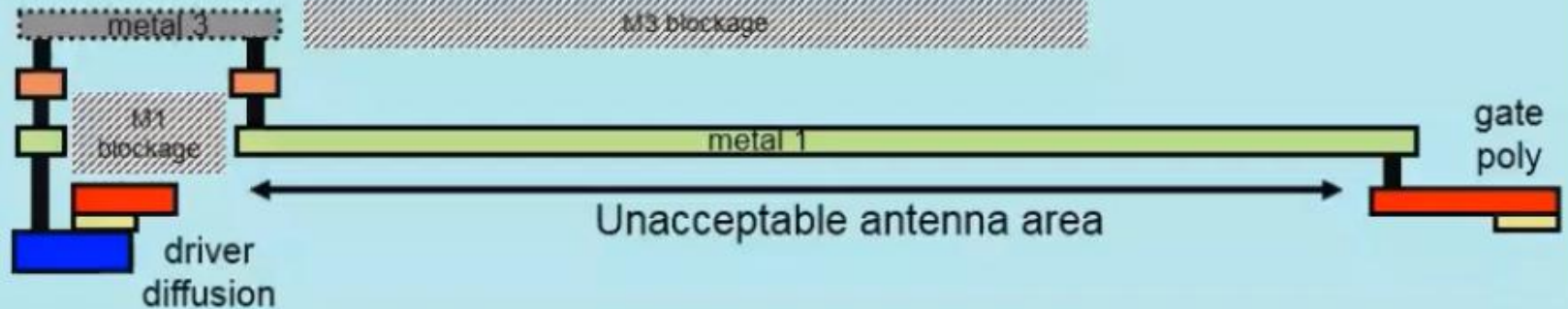
Antenna Violations

- As the total area of a wire increases during processing, the voltage stressing the gate oxide increases
- Antenna rules define acceptable total areas of wires

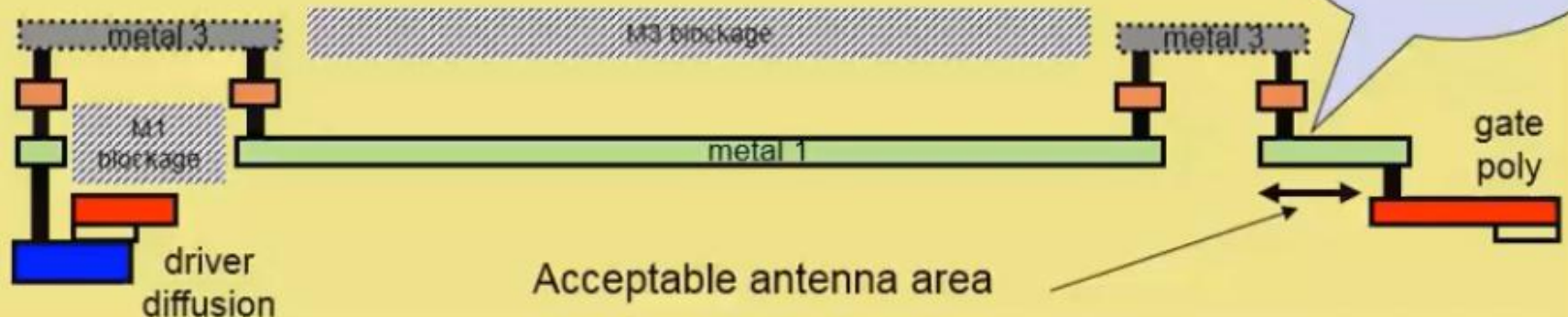


Fixing Antenna Violation by Layer Jumping

Before layer jumping

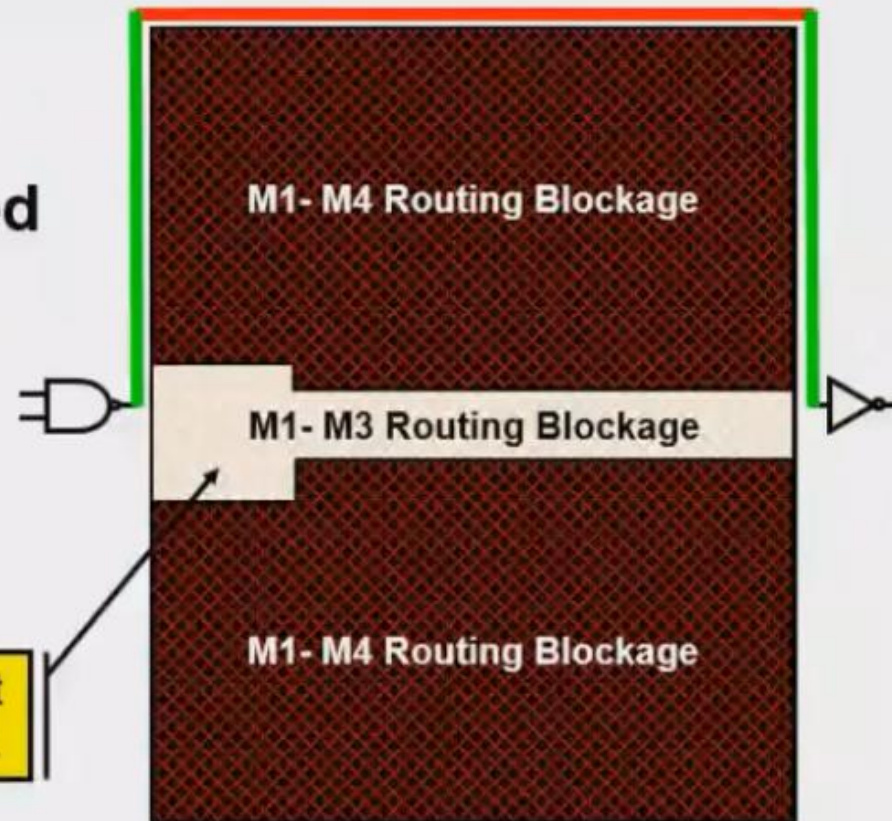


After layer jumping, to meet Antenna rules



Router Follows Preferred Routing Direction

By default IC Compiler will not route nets for a long distance in the non-preferred routing direction



M4 has a horizontal routing channel but its preferred routing direction is vertical.

Macro with routing blockages

You need to change the preferred routing direction!

Define Routing Blockages

A routing blockage defines a region where no routing is allowed on a specific layer

- Affected layer must be specified using *blockage layers* instead of the techfile layer names (see notes)
- The region of the blockage is specified in one of two ways:
 - ◆ Use the -bbox option to specify the region for a rectangular routing blockage.
 - ◆ Use the -boundary option to specify the region for a rectilinear routing blockage.

```
# Create rectangular routing blockages on  
# metall and via1 blockage layers  
create_routing_blockage -bbox {30 100 120 340} \  
-layers {metallBlockage via1Blockage}
```

Redundant Via Insertion

- Replaces single-cut vias with multiple-cut via arrays or another single-cut via with a different contact code
- Redundant via insertion algorithm options:
 - Concurrent soft-rule-based redundant via insertion
 - Postroute redundant via insertion (discussed in next Unit)



Redundant Via Insertion: Setup

- Zroute reads default via definitions from the technology file and generates an optimized via mapping table
- Check the default via mapping table:
 - If OK, proceed with redundant via insertion
 - If changes are required, define new redundant via rules

```
# Check default via mapping table:
insert_zrt_redundant_vias -list_only

# Optional definition of new redundant via rules
define_zrt_redundant_vias -from_via {VIA23 VIA34} \
                           -to_via {VIA23 VIA34} \
                           -to_via_x_size {1 2} \
                           -to_via_y_size {2 1}
```

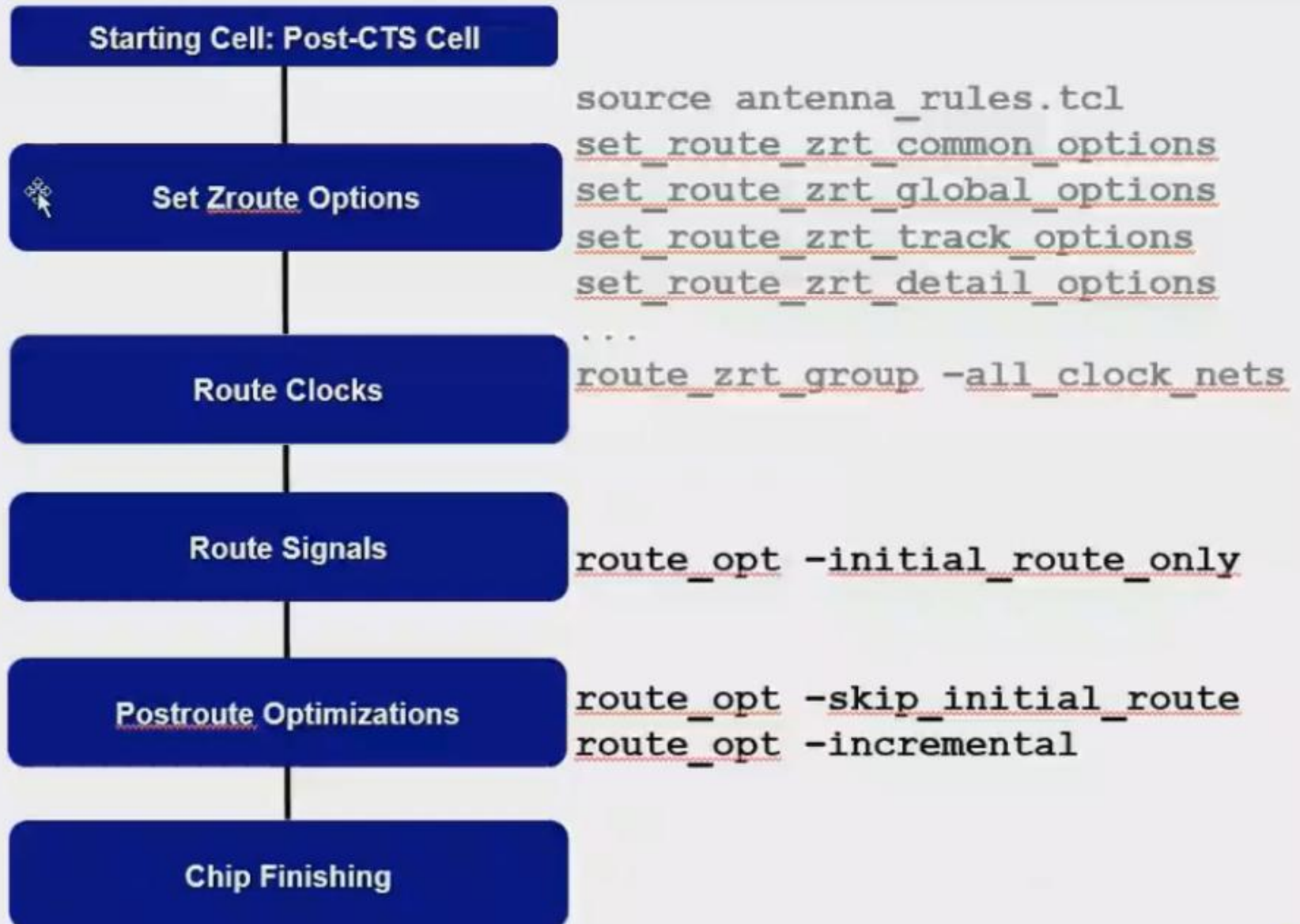
Redundant Via Insertion: Execution

Redundant via insertion (RVI) I

- Checks physical design rules to minimize DRC violations
- Happens during any detail routing operation

```
# Enable automatic insertion of redundant vias
# in subsequent routing steps
set_route_zrt_common_options \
  -post_detail_route_redundant_via_insertion medium
set_zrt_detail_route_options \
  -optimize_wire_via_effort_level medium
define_zrt_redundant_vias ...
route_opt -initial_route_only; # RVI happens here
...
route_opt -skip_initial_route; # RVI happens here
```

Basic Zroute Flow



Core Routing: route_opt

route_opt

-effort low | medium | high

-stage global | track | detail

-power

-xtalk_reduction

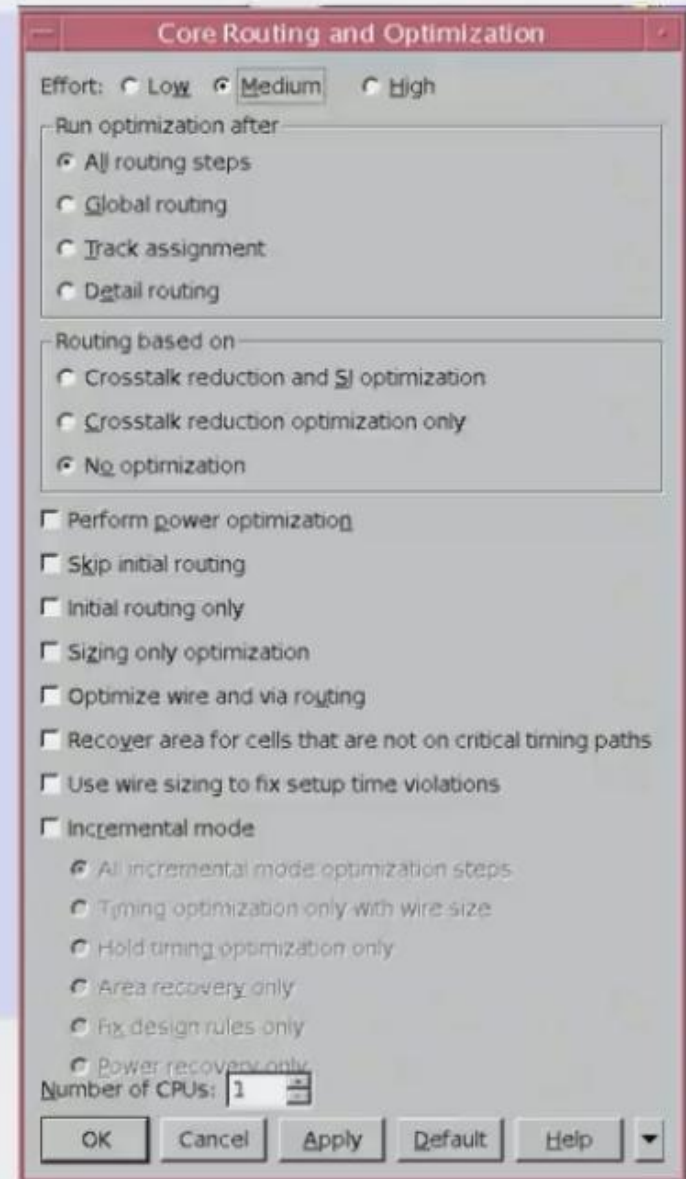
-initial_route_only

-skip_initial_route

-incremental

-area_recovery

...



First *route_opt*

■ Perform initial route only

- Allows analysis (setup/hold timing, DRC, clock skew)
- Helps to determine post initial-route options

```
route_opt -initial_route_only
```

■ Initial routing entails *global routing, track assignment and detail routing*

■ All unrouted (signal) nets are now fully connected

■ May have timing, max_tran/cap, and physical DRC violations

- Follow up with post initial-route route_opt optimizations

Post Route Optimization Examples

■ Post initial-route full optimization

```
route_opt -skip_initial_route -effort medium -power
```

■ If you have logical DRC violations

- Switch from default priority of timing over DRC

```
set_app_var routeopt_drc_over_timing true  
route_opt -effort high -incremental -only_design_rule
```

■ If additional specific optimization is needed:

```
route_opt -size_only | -only_hold_time | \  
           -only_wire_size | -wire_size
```


Getting Design Statistics

- **After redundant via insertion, Zroute generates a conversion report:**
 - Reports the optimized via conversion rate for each layer
 - ◆ The optimized via conversion rate includes both double vias and DFM-friendly bar vias, which have a single cut but a larger metal enclosure.
 - ◆ The distribution of optimized vias by weight for each layer
 - ◆ The overall double via conversion rate for the design
- **The report_design -physical command gives a routing summary including double via conversion rates, global routing, track assignment, and detail routing**

Check for Physical Design Rule Violations

```
verify_zrt_route; # Uses Zroute DRC engine  
route_zrt_detail -incremental true; # Fix DRCs
```

The screenshot displays a PCB design tool interface. On the left, the 'Error Browser' window is open, showing a list of errors. The 'Detail Route' error is selected, showing 7 instances. The error details for ID 1053184 are expanded, showing the error type 'Same-net notch/gap', layer 'METAL-METAL', and a summary of DRC violations. The main workspace shows a zoomed-in view of a PCB layout with a yellow box highlighting a specific area. The text '[BLENDER_1/mult_171/n51]' is visible in the top right corner of the workspace.

Error Browser

Reload Help Options: ▾

Error Type	Count
route_opt	7
Detail Route	7
Diff-net spacing	1
Same-net notch/gap	6

<< Prev 1 of 1 Next >> Per page: 1000 ▾

Id	Type	Layer	Summary
1053184	Same-net notch/gap	METAL-METAL DRC	
1053185	Same-net notch/gap	METAL-METAL DRC	
1053186	Same-net notch/gap	METAL-METAL DRC	

Error ID: 1053184
Error Type: Same-net notch/gap
Error Layer: METAL-METAL
Error Obj Info : DRC Violations (Layer 14 14)
Error Summary : DRC Violations (Layer 14 14)
Error bbox : (373.3200 328.1450) (373.4600 328.3500)

Clear Filter Filter... Fixed

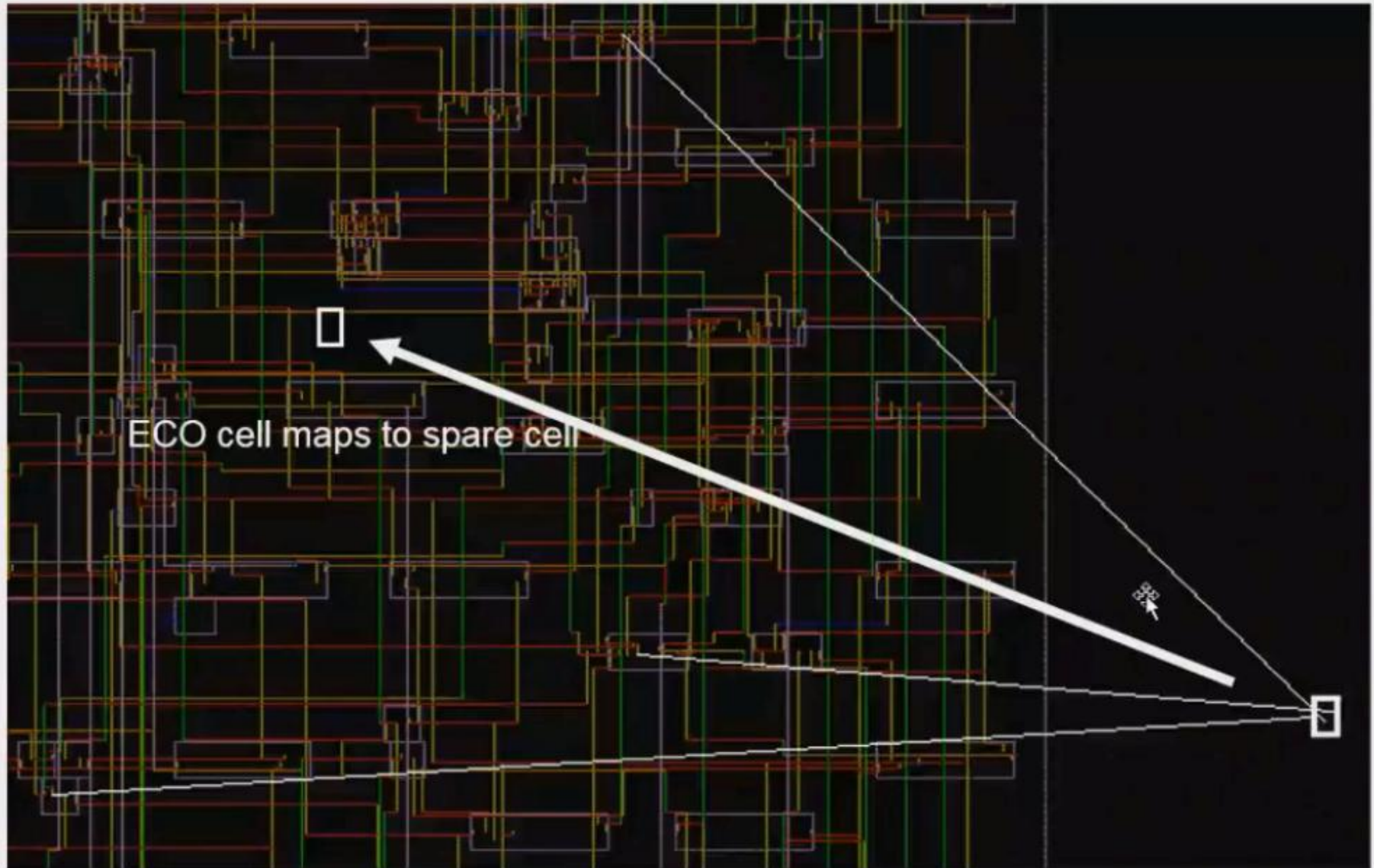
Layout View Error Display Options
Show: All ▾ Follow: Zoom ▾ 2 ▾ Dim

Verify Route: verify_zrt_route

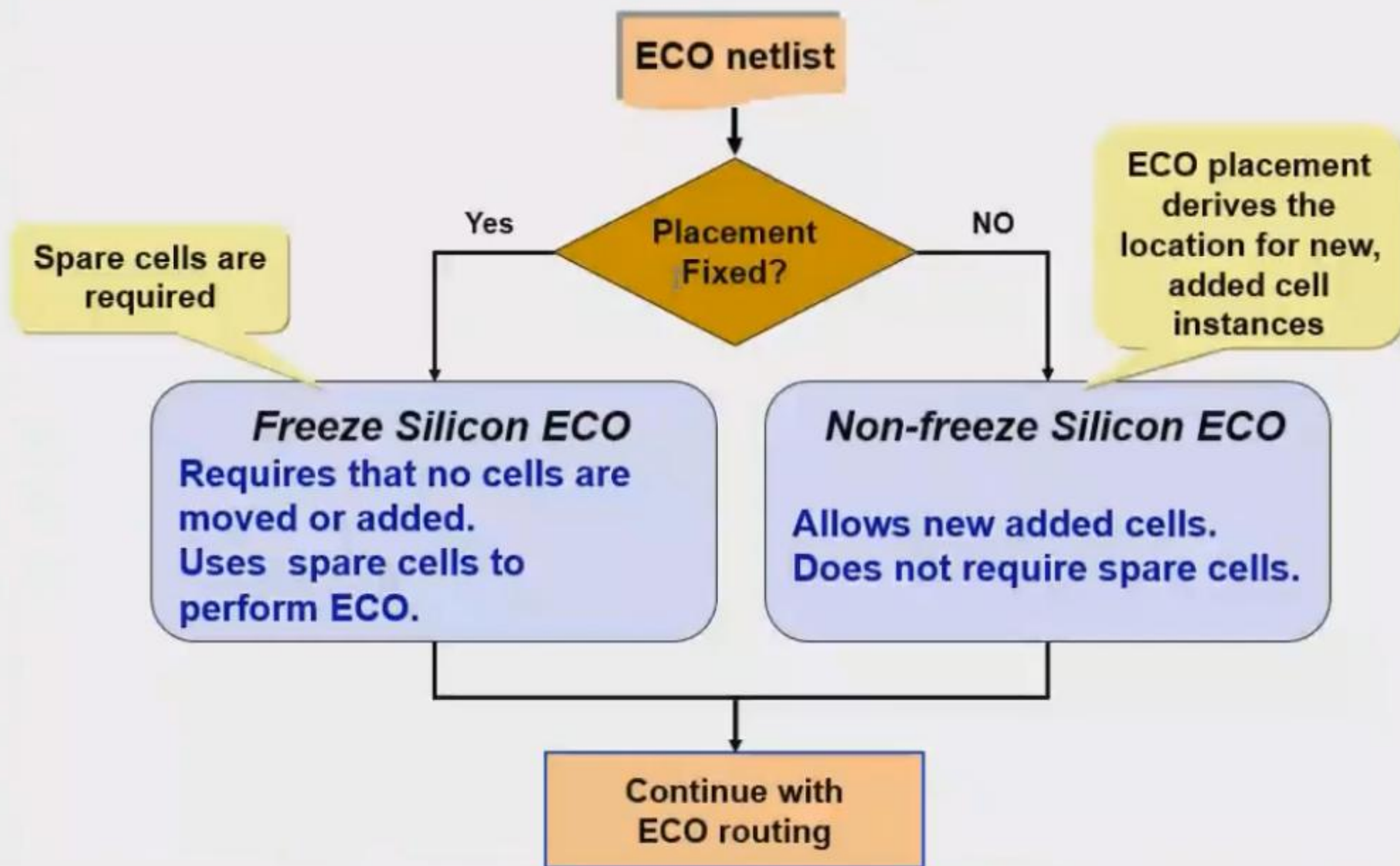
- **Checks signal and clock routing for**
 - Physical DRCs, opens, shorts and antenna violations
- **Does not check DRCs:**
 - Amongst pre-routed nets only (e.g. P/G grid structure)
 - On nets marked as type “user”
 - ◆ Wire shapes hand-created by the designer , not associated with any net (e.g. Logo, alignment marker, etc.)
- **Use verify_lvs to help debug opens**

```
verify_lvs -ignore_short -ignore_min_area
```

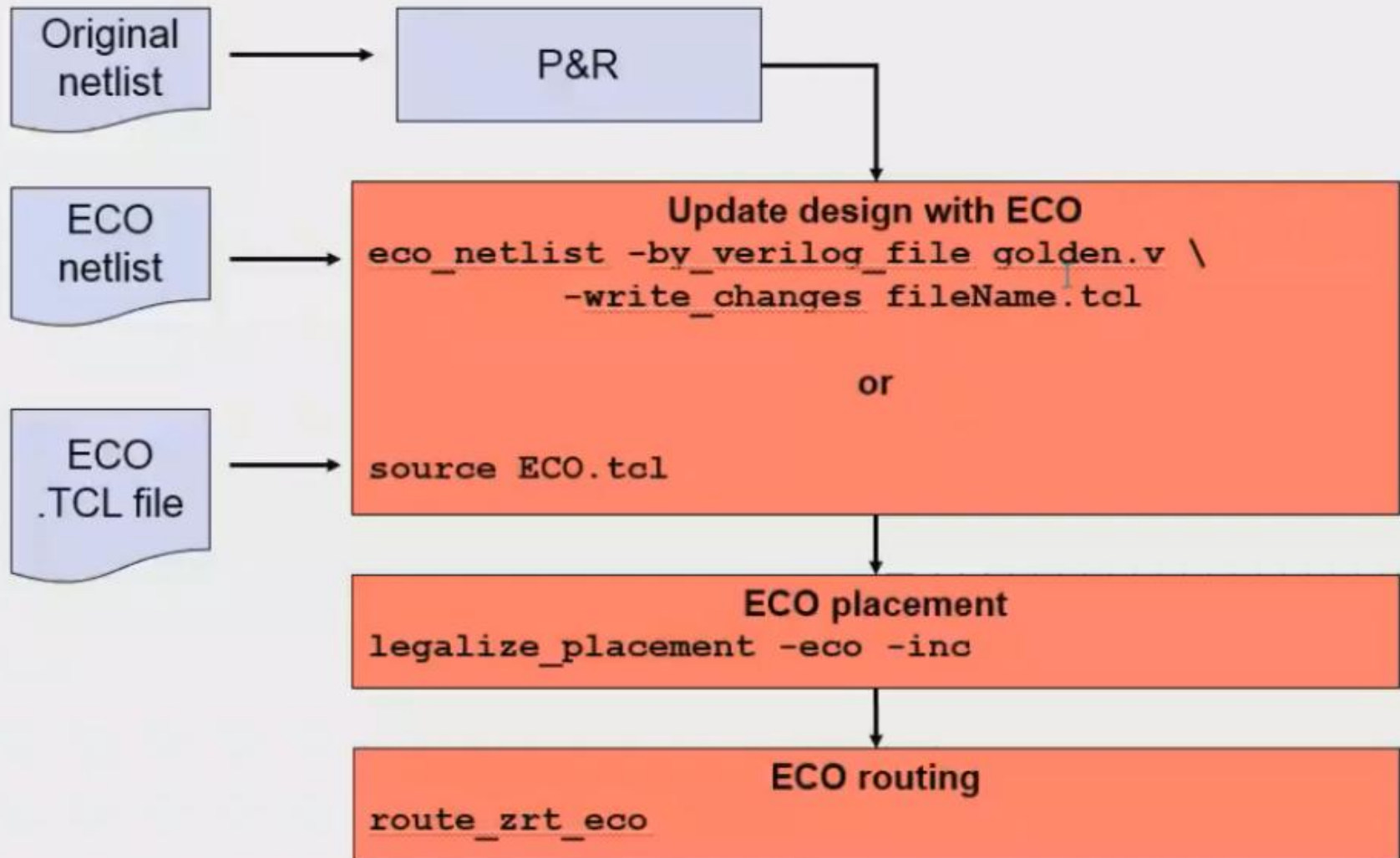

Unit Objectives: Perform functional ECOs



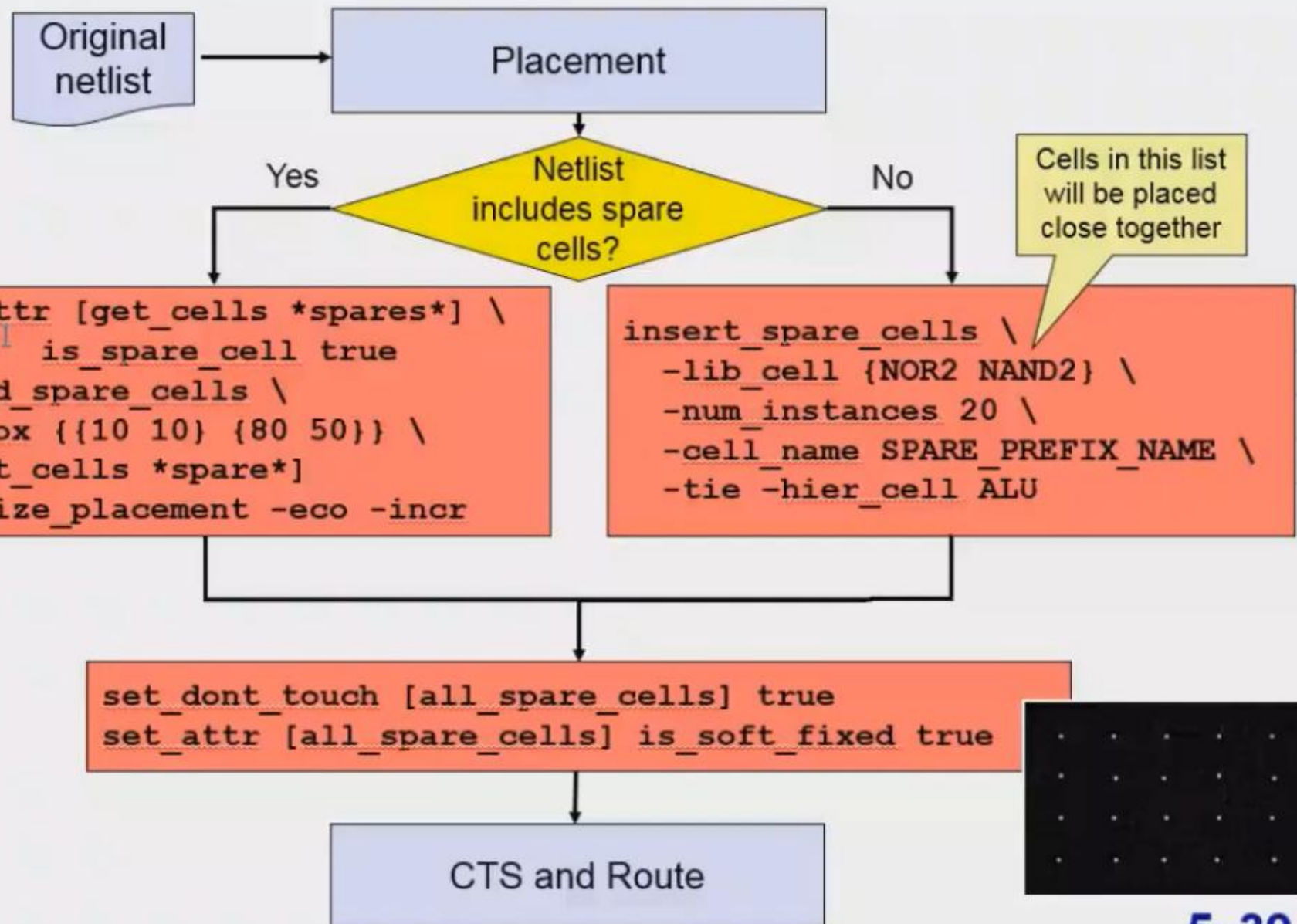
The Two Types of ECO Flows



Non-Freeze Silicon ECO



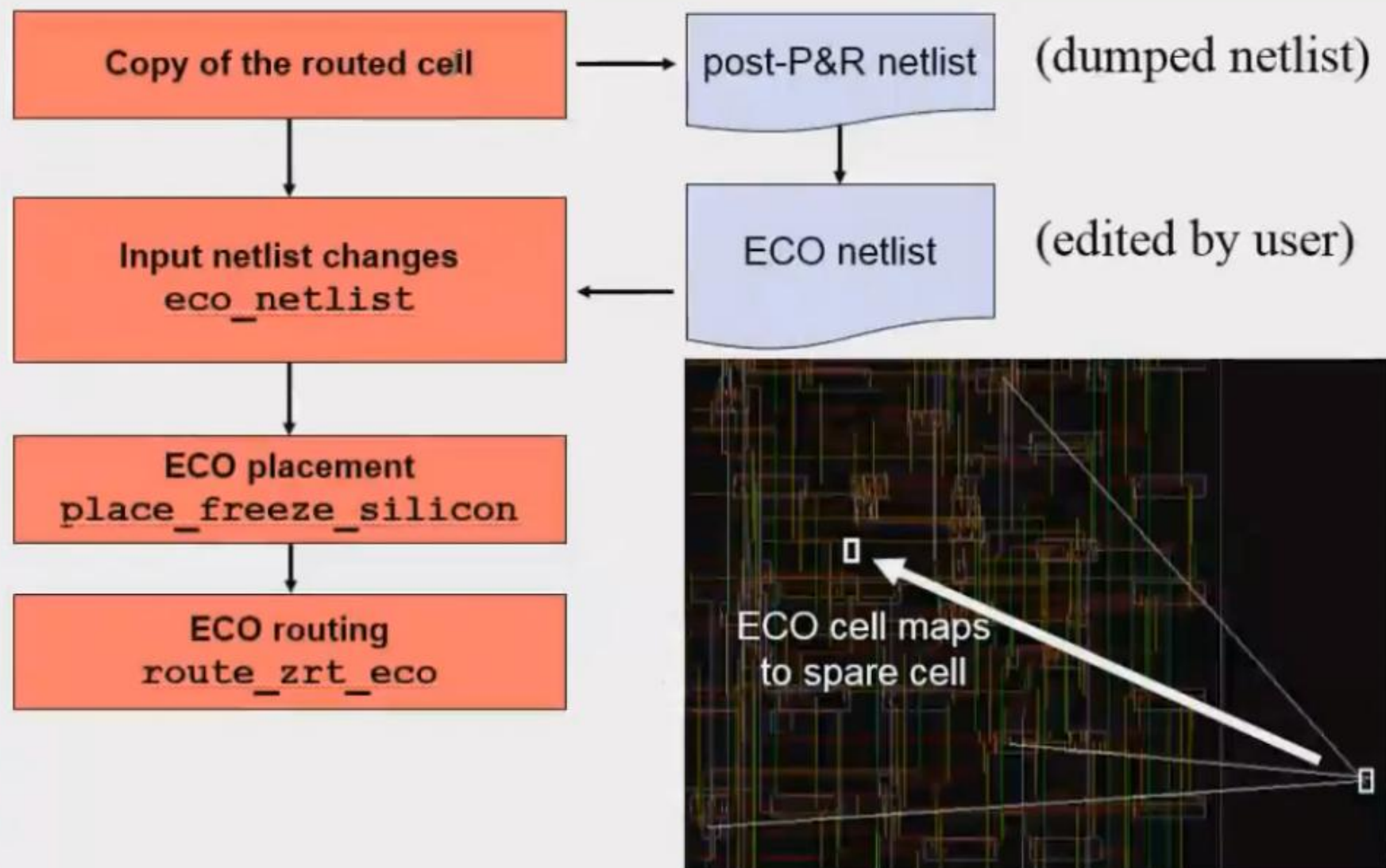
Freeze Silicon ECO Requires Spare Cells



Protecting Spare Cell

- Spare cells are *dont_touch* so ICC optimizations won't remove the unconnected cells
- Set the spare cells to SOFT FIXED once the spare cells are distributed
 - Use `set_attribute` to set the spare cells to SOFT FIXED
- Why set the spare cells to SOFT FIXED?
 - Detailed placer may fail if there are too many fixed cells
 - The soft-fixed attribute prevents incremental coarse placement from moving spare cells
 - The soft fixed cells can still be moved slightly and legalized by CTS and Routing optimizations

Freeze Silicon ECO: Metal Change Only



ECO Route: `route_zrt_eco`



```
route_zrt_eco  
  -nets collection_of_nets  
  -utilize_dangling_wires true | false  
  -open_net_driven  
  -max_detail_route_iterations int
```

- **By default, the basic ECO routing command**
 - Utilizes dangling wires
 - Runs global routing to connect broken nets
 - Runs track assignment to assign global wires
 - Runs detail routing to fix DRC violations
- **Use ECO routing command for netlist changes or after manual changes**

Design Status, Completion of Routing Phase

- Placement – completed
- Clock Tree Synthesis – completed
- Power/Signal/Clock nets – routed
- Setup/Hold timing – Met (positive slack)
- Logical DRC – max cap/transition – no violations
- Physical DRC – no violations

Chip Finishing Flow



Random Particle Defects

Critical Area

Antenna

Filler Cells

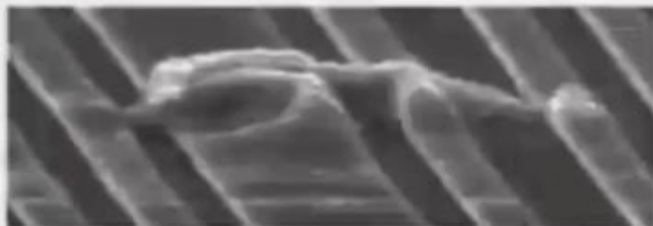
Inc. Timing Opt.

Redundant Vias

Metal Fill

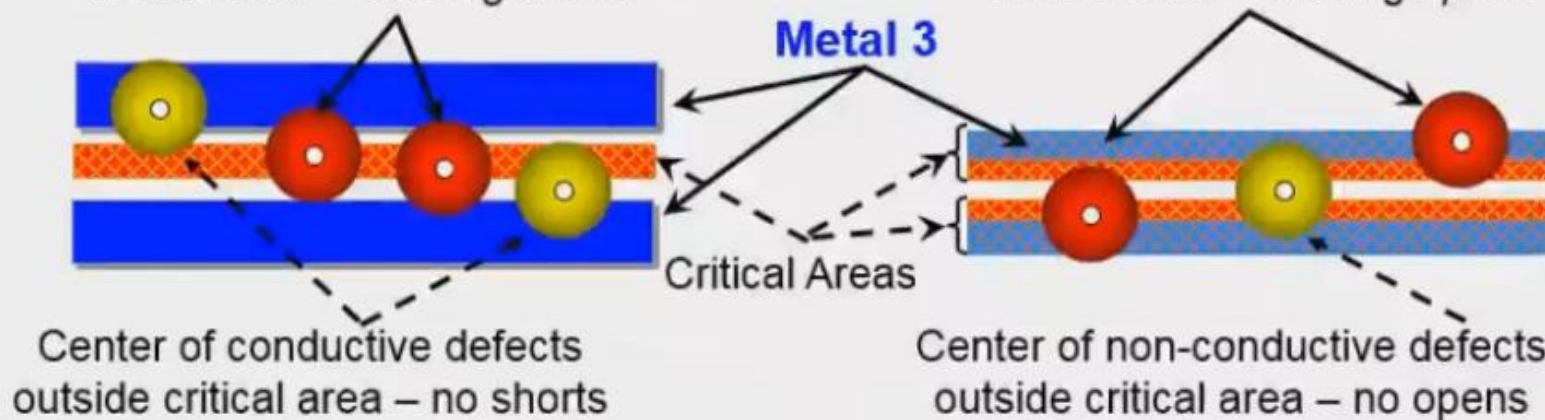
Random particle defects during manufacturing may cause *shorts* or *opens* during the fabrication process

- Wires at minimum spacing are most susceptible to shorts
- Minimum-width wires are most susceptible to opens



Center of conductive defects within critical area – causing *shorts*

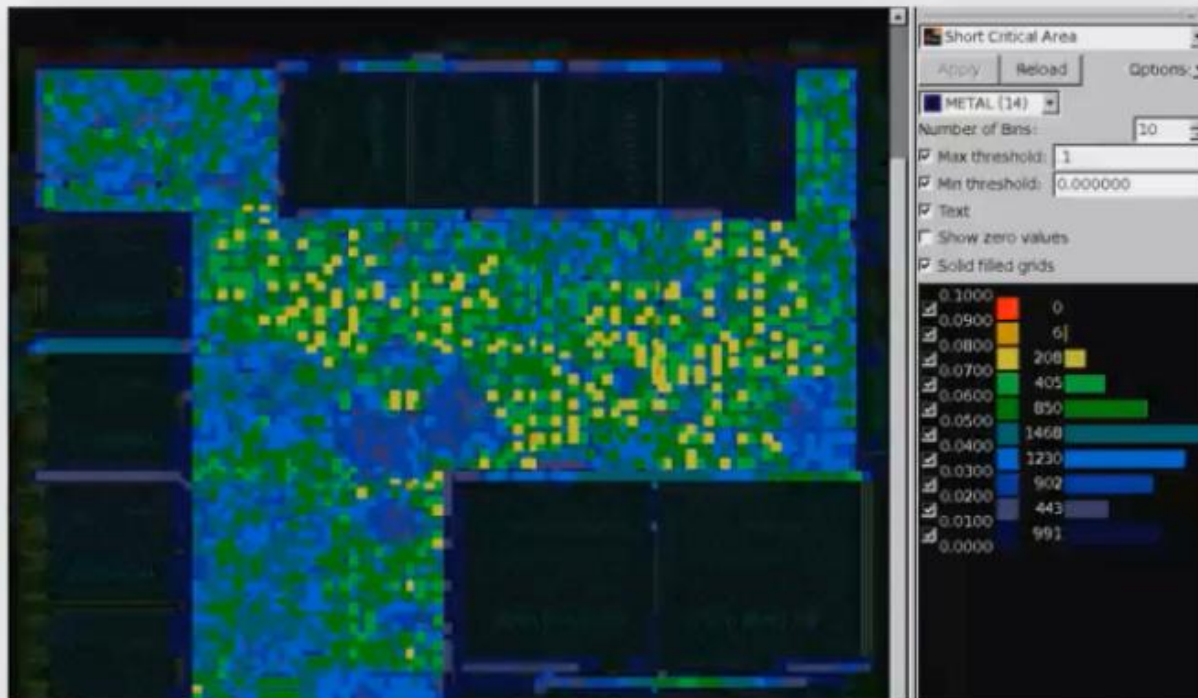
Center of non-conductive defects within critical area – causing *opens*



Reporting the Critical Area

```
report_critical_area  
I -particle_distr_func_file <file>  
  -input_layers {m2 m3 m4}  
  -fault_type {short|open}
```

Generates both textual and graphical output



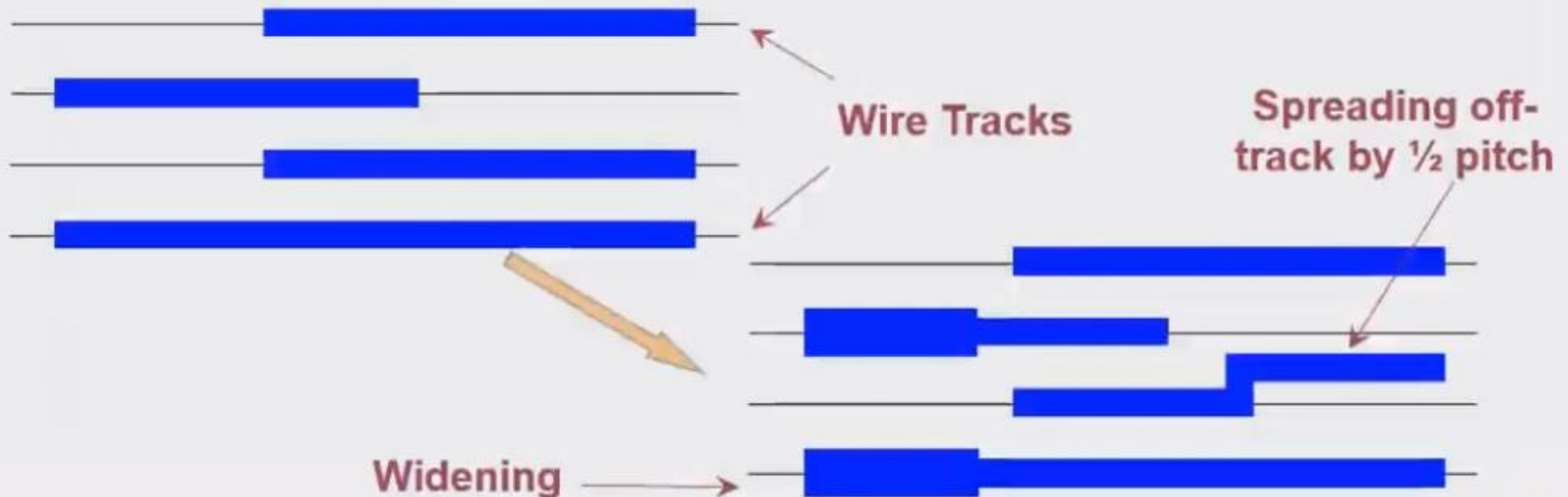
Discrete Defect Size Distribution

- Defect size distribution function depends on the fabrication process
- IC Compiler accepts discrete defect sizes and their probabilities in a table format
- An example

Defect Size	Probability
0.20	0.002778
0.36	0.000922
0.52	0.000412
0.68	0.000219
0.84	0.000130
...	...

Solution: Wire Spreading + Wire Widening

```
spread_zrt_wires \  
  -timing_preserve_setup_slack_threshold .05 \  
  -timing_preserve_hold_slack_threshold .05  
widen_zrt_wires \  
  -timing_preserve_setup_slack_threshold .05 \  
  -timing_preserve_hold_slack_threshold .05
```



Wire Spreading: spread_zrt_wires

spread_zrt_wires

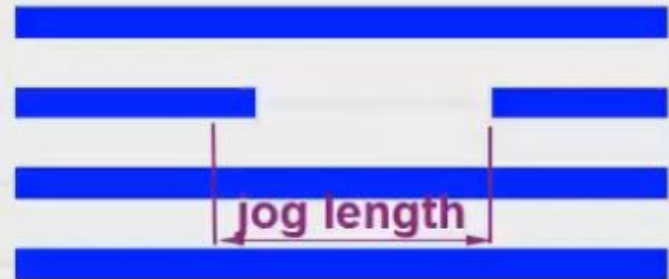
```
-pitch real           ;# of pitches to spread,  
                      default is 0.5  
  
-min_jog_length int  ;# Minimum jog length in layer  
                      unit on preferred direction,  
                      default is 2  
  
-timing_preserve_setup_slack_threshold real  
-timing_preserve_hold_slack_threshold real
```

Wire spreading

- Postroute function for reducing critical area for shorts
- Spread signal wires by $\frac{1}{2}$ pitch or user-specified amount
- Only spread in preferred direction
- Automatic search and repair to fix DRCs
- Timing preservation (optional)

Controlling Minimum Jog Length

- Pushing wires off-track always creates a jog and increases wire length
- Use `-min_jog_length` option to control the minimum jog length (default: 2 pitches)
 - Will not push a wire unless the available space is larger than `'-min_jog_length'`



Wire Widening: widen_zrt_wires

```
widen_zrt_wires  
-timing_preserve_setup_slack_threshold real  
-timing_preserve_hold_slack_threshold real  
-timing_preserve_nets <timing preserve nets>
```

Wire widening

- Postroute function for reducing critical area for opens
- Will not trigger new fat spacing rules when widening
- Automatic search and repair to fix DRCs
- Timing preservation (optional)

Fix Remaining Antenna Violations w/ Diodes

Critical Area

Antenna

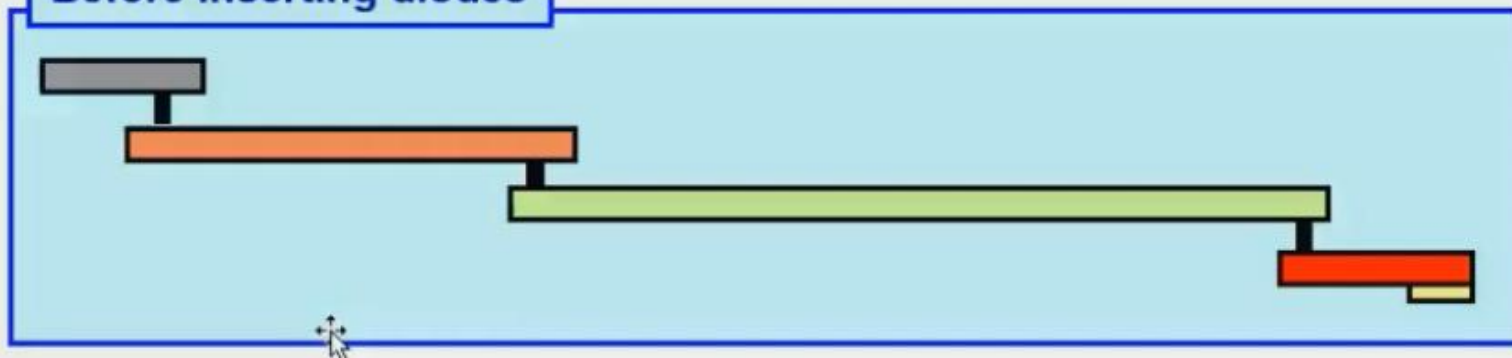
Filler Cells

Inc. Timing Opt.

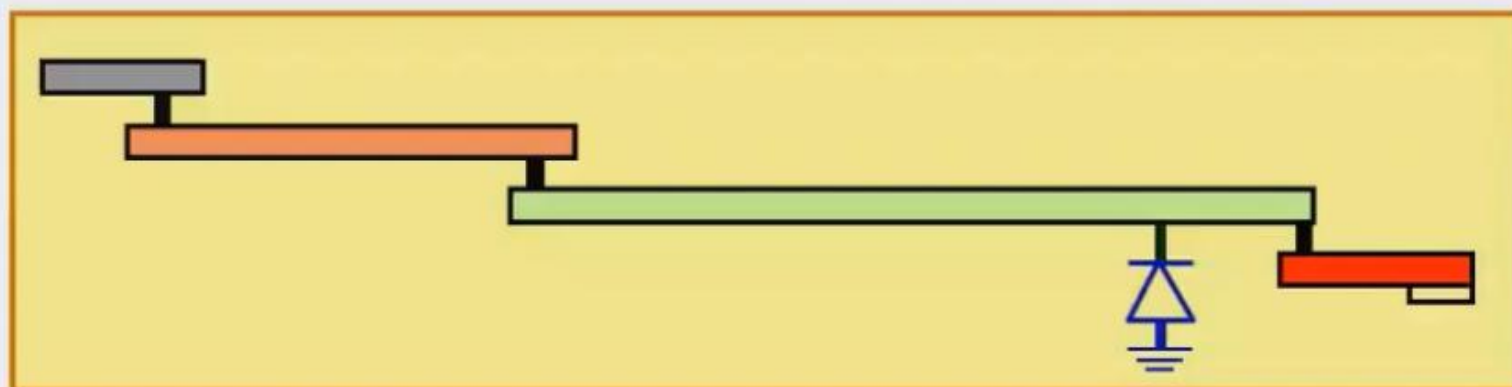
Redundant Vias

Metal Fill

Before inserting diodes



Diode Inhibits large voltage swings on metal tracks



During etch phase, the diode clamps the voltage swings.

Antenna Fixing with Diode Insertion

- Diode insertion is also concurrent when enabled
 - It is NOT recommended during detailed route
- Use diodes to fix antenna violations that are not fixable by layer jumping:
 - Can specify diode names (automatic if none specified)

```
set_route_zrt_detail_options \  
-antenna true \  
-insert_diodes_during_routing true \  
-diode_libcell_names {adiode1 adiode2}  
  
route_zrt_detail -incremental true
```


Why Filler Cell Insertion?

Critical Area

Antenna

Filler Cells

Inc. Timing Opt.

Redundant Vias

Metal Fill

- For better yield, density of the chip needs to be uniform
- Some placement sites remain empty on some rows
 - ICC can fill such empty sites with standard filler cells

Insert Filler Cells in Unused Placement Sites

■ Add filler cells with metal first

- For DRC checking purposes, standard cell PG rails should be complete prior to inserting filler cells with metal

■ Then add filler cells without metal

```
insert_stdcell_filler \  
  -cell_with_metal "fillCap64 fillCap32" \  
  -connect_to_power VDD -connect_to_ground VSS \  
  -between_std_cells_only  
insert_stdcell_filler \  
  -cell_without_metal "fill116 ... Fill11"  
  -connect_to_power VDD -connect_to_ground VSS \  
  -between_std_cells_only
```

Is Incremental Timing Optimization Needed?

Critical Area

Antenna

Filler Cells

Inc. Timing Opt.

Redundant Vias

Metal Fill

- Critical area, antenna fixing or filler cell insertion can create small timing violations
- Can perform incremental timing optimization by re-sizing the existing cells

```
route_opt -incremental -size_only
```

I

Voids in Vias during Manufacturing

Critical Area

Antenna

Filler Cells

Inc. Timing Opt.

Redundant Vias

Metal Fill

■ During routing phase:



```
set_route_zrt_common_options \  
  -post_detail_route_redundant_via_insertion medium  
set_route_zrt_detail_options \  
  -optimize_wire_via_effort_level medium  
route_opt ...
```

■ During chip finishing phase:

```
insert_zrt_redundant_vias ...
```

Redundant Via Insertion: Setup

- Zroute generates via mapping table using all contact codes from the technology file

```
insert_zrt_redundant_vias -list_only
```

- If OK, proceed with redundant via insertion
- If changes are required, define new redundant via rules

```
define_zrt_redundant_vias \  
    -from_via {VIA23 VIA34} \  
    -to_via {VIA23 VIA34} \  
    -to_via_x_size {1 2} \  
    -to_via_y_size {2 1}
```

User defined rules
are saved in the
MW library

Redundant Via Insertion: Setup with Priority

- Some vias may be better for DFM, while others are better for routability
- Use -to_via_weights option to set a priority
 - Weight is 1 to 10
 - Higher weight via will be tried first
 - With equal weights, prioritization is based on routability

See example on the next slide

Example: Prioritizing for DFM

- For DFM, say: $VB > VG > V$
 - Best rate of double-vias in that order
 - If single vias remain, optimize in that order
- Example for Via 1 layer... add others as needed

```
define_zrt_redundant_vias \
  -from_via   {VB1 VG1 V1 VG1 V1 V1 VG1 V1 V1} \
  -to_via     {VB1 VB1 VB1 VG1 VG1 V1 VB1 VB1 VG1} \
  -to_via_x_size {1 1 1 1 1 1 1 1 1} \
  -to_via_y_size {2 2 2 2 2 2 1 1 1} \
  -to_via_weights {5 5 5 4 4 3 2 2 1}
```

- $VB1 \text{ doubled} > VG1 \text{ doubled} > V1 \text{ doubled} > VB1 \text{ not doubled} > VG1 \text{ not doubled}$

Postroute Redundant Via Insertion

- Use insert_zrt_redundant_vias to insert redundant vias

- -effort low | med | high

```
insert_zrt_redundant_vias -effort med
```

- Higher effort level results in better rate (3-5%) but will shift vias more to fit in
 - May be worse for lithography

Timing Preservation Mode

- Inserting double vias changes timing
 - Short nets tend to slow down – increased capacitance
 - Long nets tend to speed up – decreased resistance
- To maximize double-via rate, perform non-timing-driven via insertion first
 - route_opt deletes double vias only on changed net segments and not on the entire net
- During chip finishing, enable timing preservation to prevent insertion of double vias on critical nets

```
insert_zrt_redundant_vias \  
I -timing_preserve_setup_slack_threshold slack \  
-timing_preserve_hold_slack_threshold slack \  
-timing_preserve_nets {collection_of_nets}
```


Problem: Metal Over-Etching

Critical Area

Antenna

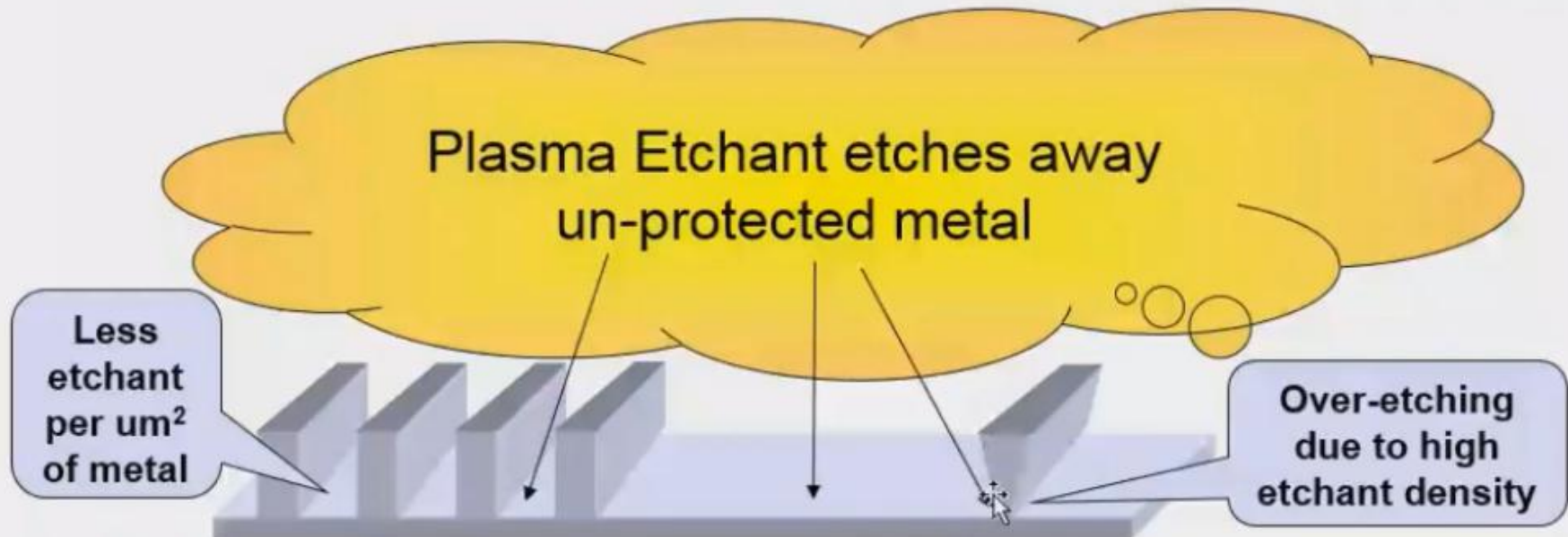
Filler Cells

Inc. Timing Opt.

Redundant Vias

Metal Fill

- A metal wire in a low metal density region receives a higher ratio of etchant and can get over-etched
- Minimum metal density rules are used to control this

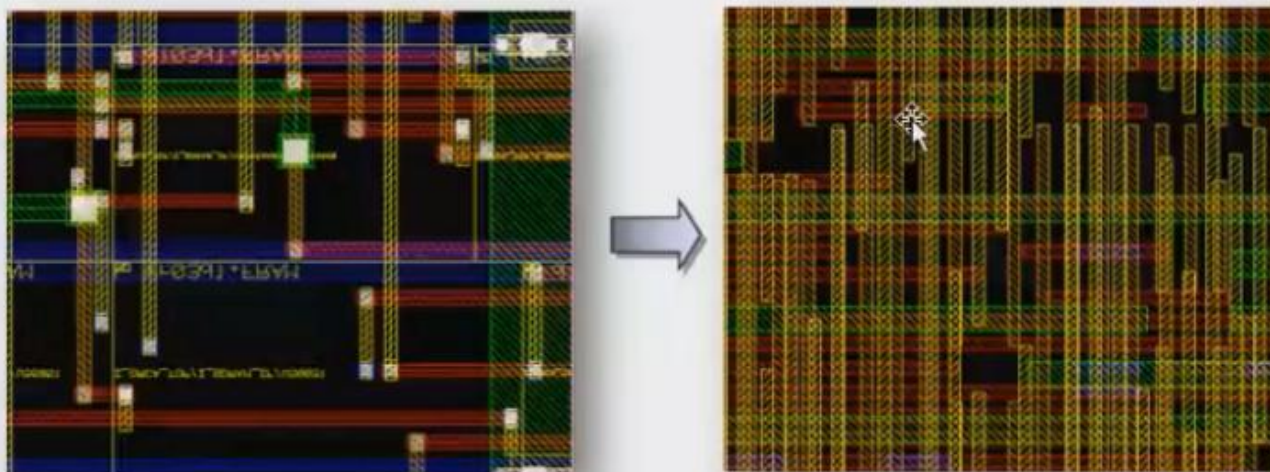


Insert Metal Fill to Prevent Over-Etching

■ Two methods

- IC Compiler: insert_metal_filler
- IC Validator (ICV) based inside ICC: signoff_metal_fill

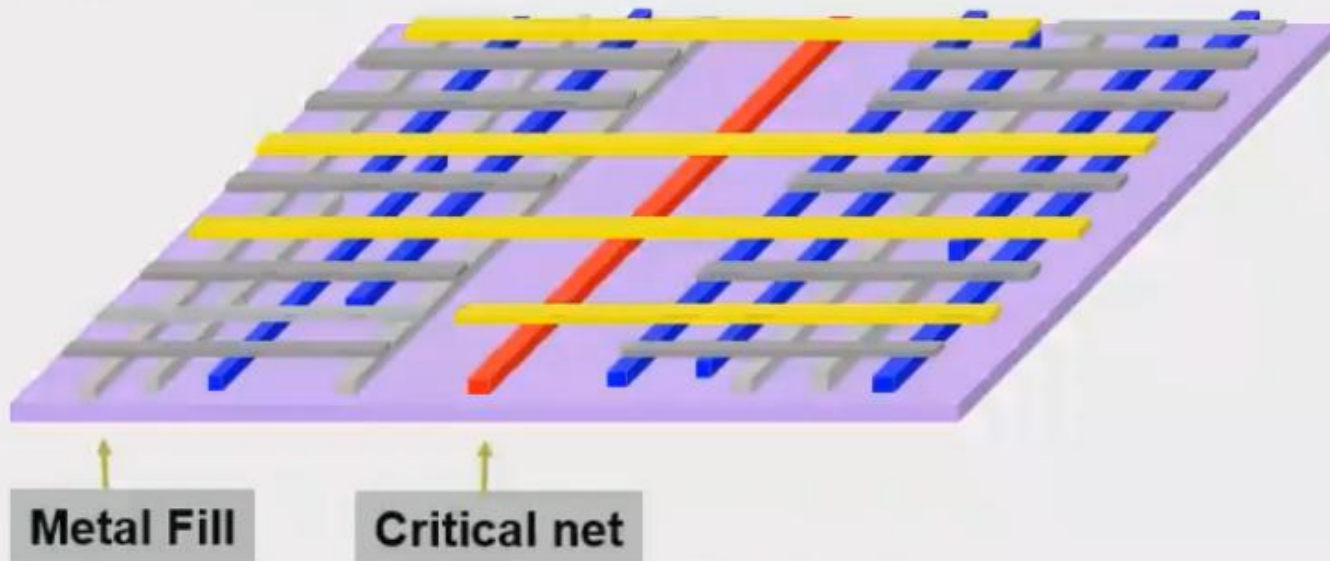
■ ICV is recommended for 45 nm and below



- Fills empty tracks on all layers (default) with metal shapes to meet the minimum metal density rules

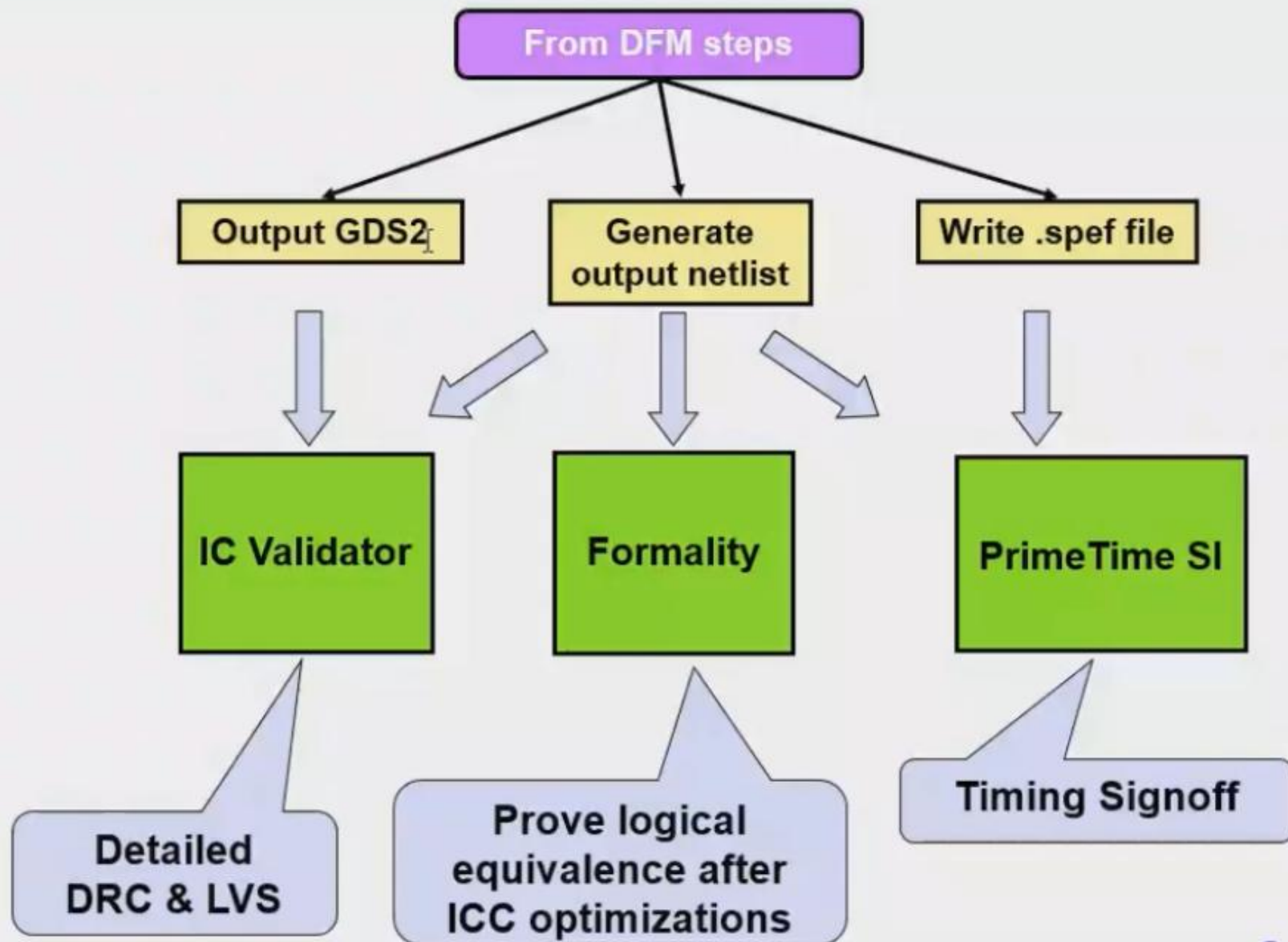
Recommended Metal Fill Options

- Use `-timing_driven` option to preserve timing on critical nets I
 - Metal fill near critical nets on the same layer, upper layer, and lower layer are removed or trimmed
- Use `-routing_space 2` option to specify a 2x `minSpacing` between normal routing wires and the fill metal



Final Validation

停止



Final Validation: Parasitics (SPEF or SBPF)

Wire parasitics for PrimeTime are provided via a .SPEF or .SBPF file

```
write_parasitics
```

```
-output <file_name>
```

```
-format <SPEF|SBPF>
```

```
-compress
```

```
-no_name_mapping
```



Use StarRC extraction for signoff

- Netlists for STA (Static Timing Analysis) do not require output of “Physical only cells” like:
 - Corner pad cells
 - Pad/core filler cells
 - Unconnected cell instances
- Unconnected cell instances (e.g. spare cells) are needed for LVS

```
change_names -hierarchy -rules verilog  
write_verilog -no_corner_pad_cells ... final.v
```


Final Validation: GDSII Output

```
set_write_stream_options ...  
write_stream -cells DFM_clean orca.gdsii
```

- GDSII for external physical verification can be generated from IC Compiler
- Requires output of “physical only cells” like:
 - Corner pad cells
 - Pad/core filler cells
 - Unconnected cell instances