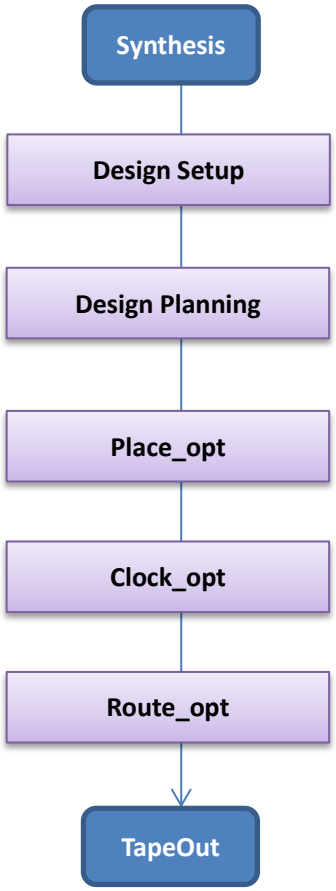
IC Compiler is for place and route and it is used after synthesis which can be done with Synopsys DC compiler or Power compiler. IC Compiler goes through the following steps and its outputs go to tapeout.



No multiple instantiated designs

IC compiler doesn't support non-uniquified design, ie. designs with multiple instantiations of the same module. There ICC script needs to uniquify design as below:

```
current_design My_Top_Design
uniquify
```

Reason for non-uniquified design:

1. These instantiations need to have their own physical locations
2. Need to connect clock to these instantiations and each instantiation needs a different clock signal.

How do you find out what libraries are loaded?

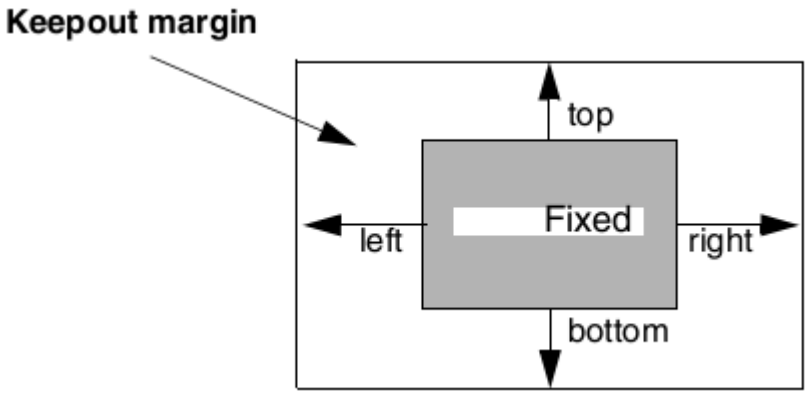
list_libs

What if the same cell is defined in more than one library, and all libraries are in link_library?

IC Compiler will pick the named cell from the first library it finds it in.

Keepout margin

A keepout margin is a region around the boundary of fixed macros in your design in which no other cells are placed.



Keeping the placement of cells out of such regions avoids congestion and net detouring and produces better QoR(quality of results).

IC Compiler supports NLDM and CCS (Composite Current Source) libraries
NLDMs are not accurate enough for 90 nm and below → Use CCS

Generating TLU+ Models

Generating TLU+ Models

- ITF (process file) provided by the vendor
- TLU+ model is generally not provided
- Generate TLU+ from ITF

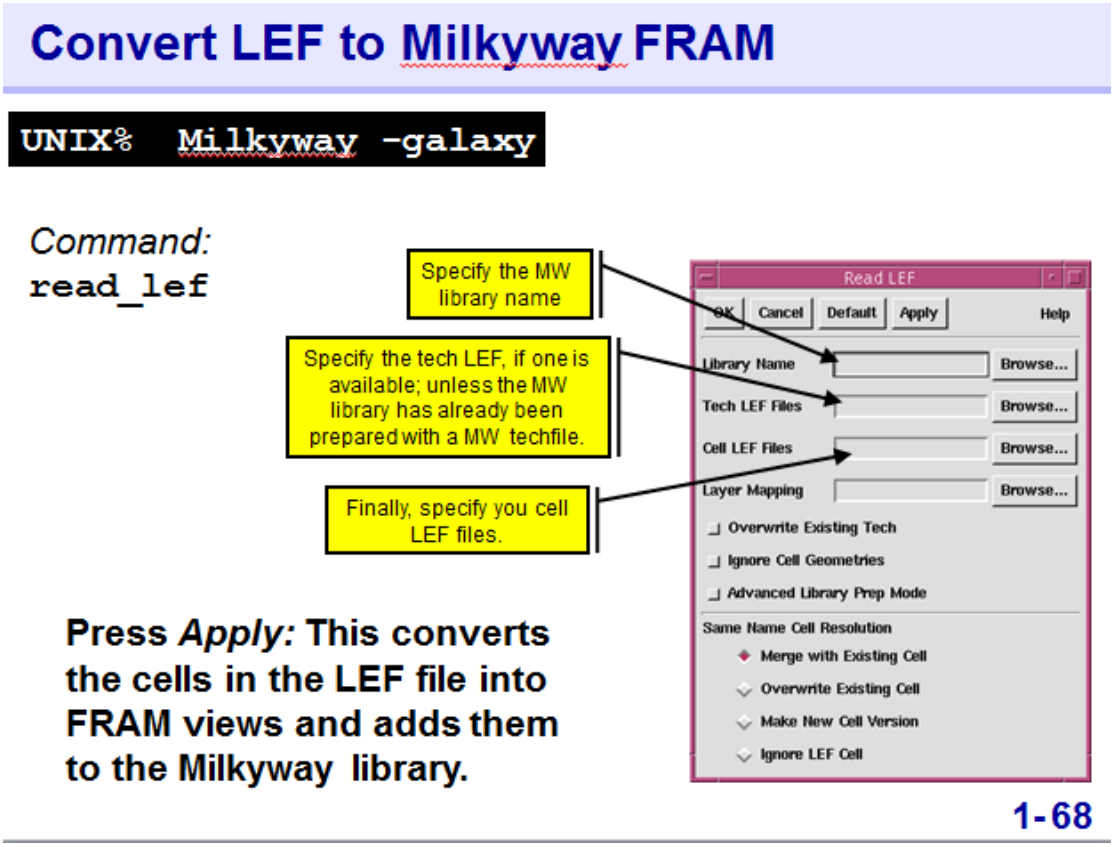
```
unix% grdgenxo -itf2TLUPlus -i <ITF file>
                            -o <TLU+ file>
```

Where:		
	`-itf2TLUPlus`	generates TLU+ instead of nxtgrd file
	`-i`	is the ITF file
	`-o`	is the output, binary TLU+ model file

Always use the latest Star-RCXT release to generate the models.

1-56

Convert LEF to Milkyway FRAM

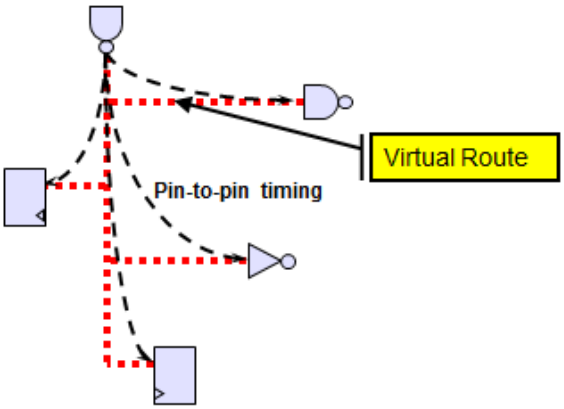


The above applies to reference libraries only.

Delay Calculation Algorithm

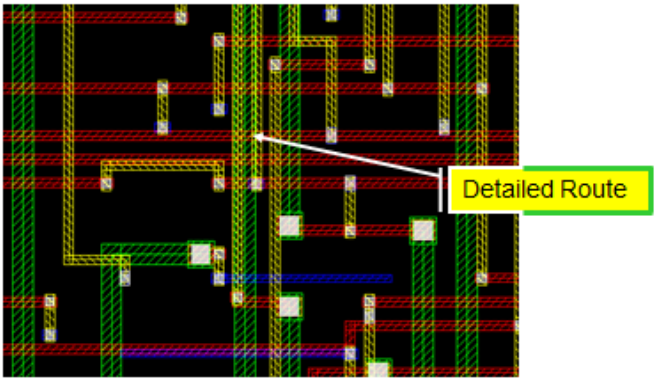
- **Calculating Net Delay is done using Delay Calculation algorithms: Elmore, Arnoldi**

PreRoute Delay Calculation Algorithm

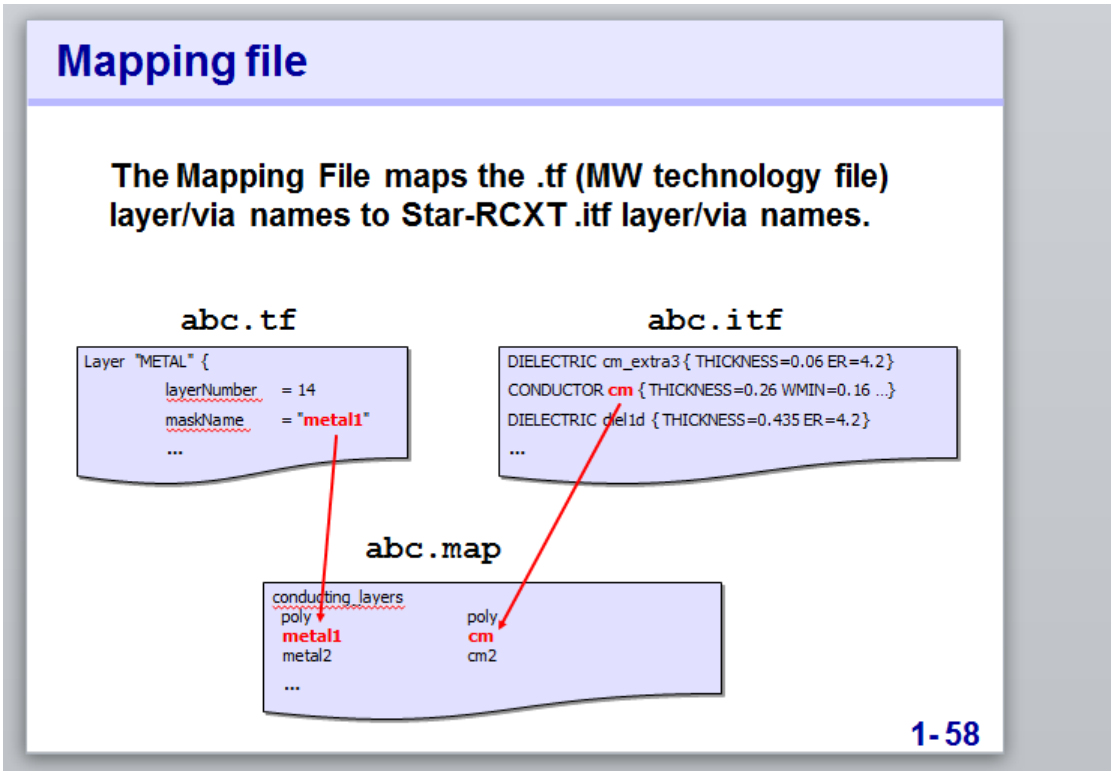


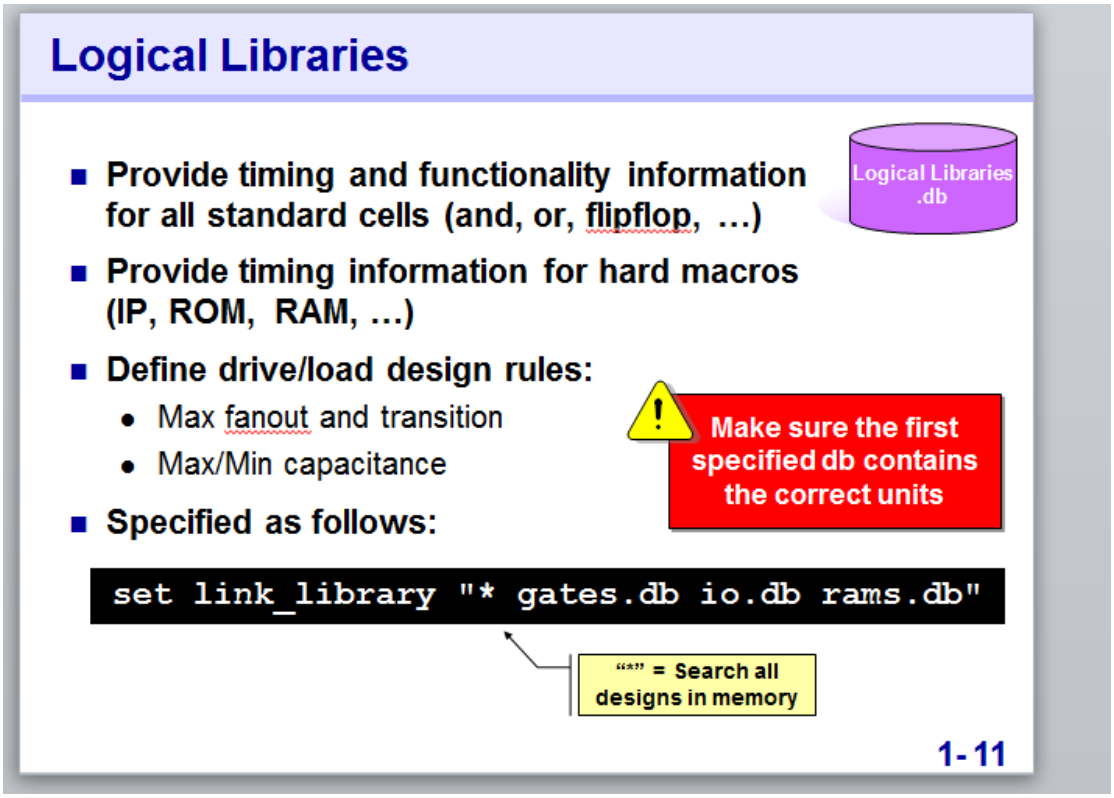
- Prior to routing, net geometry is estimated based on a Virtual Route
- Since Virtual Routing is only an estimate, an Elmore model is used for delay calculation

PostRoute Delay Calculation Algorithms



- After routing, detailed nets are available and extraction can be more accurate
- By default, Elmore is still used
- Arnoldi can be turned on for postroute calculations





The Tcl command “set” assigns the string “* gates.db io.db rams.db” to the *variable* link_library.
IC Compiler searches for library file names as indicated by this variable, in the order they’re listed.
The “*” stands for all designs that have already been loaded into IC Compiler. When the link is performed, it

uses design information already loaded into the memory first. If the design data cannot be found in memory, it loads the missing libraries as necessary.
These .db files can be located in the LM views of a reference library, but this is not required.
How do you find out what libraries are loaded?
list_libs
What if the same cell is defined in more than one library, and all libraries are in link_library?
IC Compiler will pick the named cell from the first library it finds it in.

1. Milkyway Reference Libraries

Information is stored in so-called "views", for example:

- **CEL: The full layout view**
- **FRAM: The abstract view used for P&R**
- **LM: Logic Model with Timing and Power info *(optional*)*** 。(***Optional) here means that the logical libraries do not have to be stored within the Milkyway library structure, but can be located anywhere else. IC Compiler only reads logical libraries (.db) specified through the link_library variable.***

2. Technology File (.tf file)

- **Tech File is unique to each technology**
- **Contains metal layer technology parameters:**
  - Number and name designations for each layer/via
  - Dielectric constant for technology
  - 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

Example of a Technology File:

```
Technology    {
        dielectric          = 3.7
        unitTimeName        = "ns"
        timePrecision       = 1000
        unitLengthName      = "micron"
        lengthPrecision     = 1000
        gridResolution      = 5
        unitVoltageName         = "v"
        }
...
Layer   "m1" {
        layerNumber         = 16
        maskName            = "metal1"
        pitch               = 0.56
        defaultWidth        = 0.23
        minWidth            = 0.23
        minSpacing          = 0.23
```

3 How does IC Compiler Find Files?

You may specify where to look for files:
lappend search_path ./design_data ../scripts
lappend search_path [glob $MW_libs/*/LM]
In the example above, the command lappend is used to "append" two paths to the current setting of search_path. Never use the command "set" to set this variable, as by default it is already set to directories in the installation tree of IC Compiler.

You can also use set instead of lappend to modify the search_path:
set search_path "$search_path ./design_data ../scripts"
set search_path "$search_path [glob $MW_libs/*/LM]"
Typically, the target_library points to your standard cells only，other cell librays such as used for SRMA or ROM are set in the link_path,for example:

Set target_library "slow.db"
Set link_library "* fast.db slow.db io_max.db ram32x32.db"

Gate-level netlists contain references to standard cells and macros, which are stored in the logical libraries, as well as other hierarchical logic blocks. The link command will ensure that all references can be resolved.

4 create library

You may also create the library first, then add the reference libraries afterwards:
create_mw_lib design_lib_orca –technology techfile.tf
set_mw_lib_reference design_lib_orca –mw_reference_library "sc io ram32"
If the "-open" switch is not used, the library needs to be opened in order to import the design:
open_mw_lib design_lib_orca
You can also add reference libraries using a reference control file. Create a file, e.g. libs.ref:

```
Library design_lib_orca
    Reference ../ref/mw_lib/sc
    Reference ../ref/mw_lib/io
    Reference ../ref/mw_lib/ram32
```

Then, in IC Compiler, create the library:

```
create_mw_lib design_lib_orca \
    -technology techfile.tf
    -reference_control_file libs.ref
    -open
```

TIMING CHECK

5. saving and loading the design

- **Once setup is complete, save the Milkyway design:**
- **By default, *link_library, search_path, target_library and TLU+ settings* are stored with the CEL**
  - If library files change or move to somewhere else, the settings have to be re-applied. *See notes section below!*
  - **When you re-open the CEL, by default the stored settings are not re-applied, unless you set:**

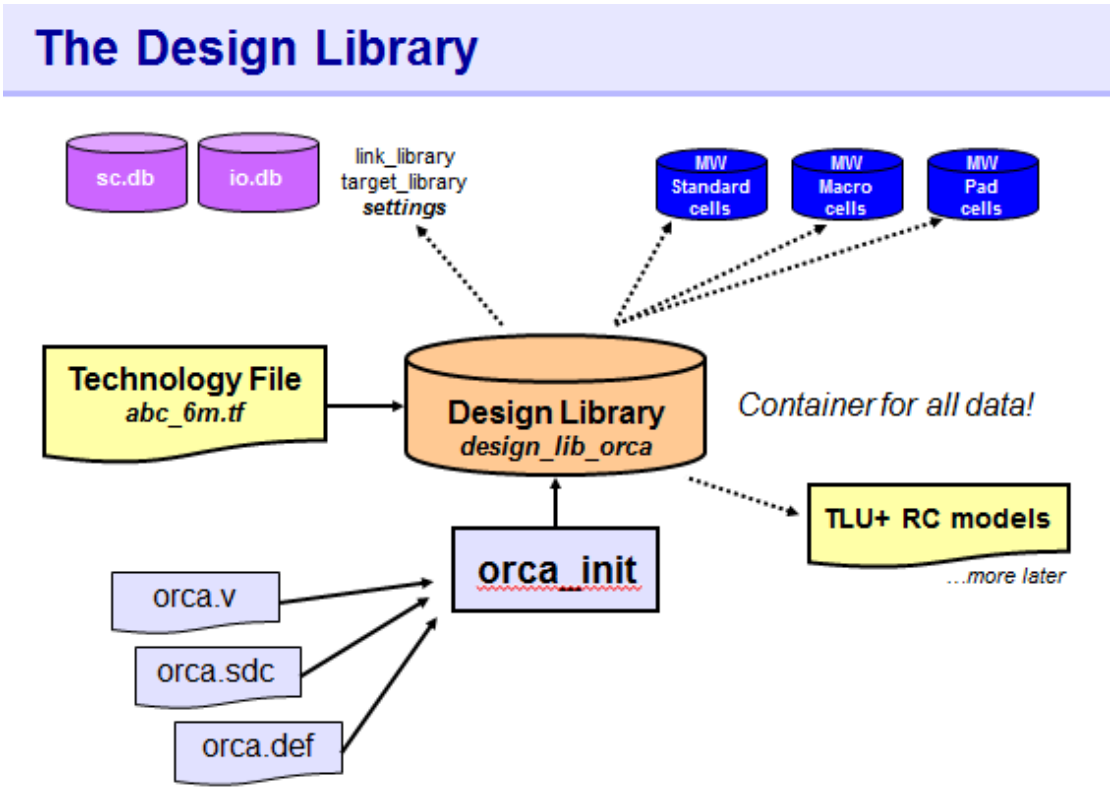
set auto_restore_mw_cel_lib_setup true
open_mw_cel orca_init

Whether settings are stored with the MW library or not is controlled through the variable save_mw_cel_lib_setup. This variable is set to true by default.

When opening a CEL with the variable auto_restore_mw_cel_lib_setup set to true, IC Compiler will display the following:
Information: AUTO-RESTORE: Setting variable search_path to . /path/to/icc/dw/sim_ver ../ref/db. (UIG-10)
Information: AUTO-RESTORE: Setting variable link_library to * sc_max.db … ram16x128_max.db. (UIG-10)
Information: AUTO-RESTORE: Setting variable target_library to sc_max.db. (UIG-10)
Information: AUTO-RESTORE: Setting variable itf_tlu_plus_library to ../ref/tlup/abc_6m_max.tluplus…. (UIG-10)

When the logical libraries change, please follow this procedure:
set auto_restore_mw_cel_lib_setup false
set search_path "new search paths"
set link_library "* new link libraries"
set target_library "new target libraries"
open_mw_cel <cell name> -lib <lib name>
link
set_tlu_plus_files –max_tlu <> –min_tlu <> –tech2itf <>

6 the design library



7 UNIX Structure of a Design Milkyway Database

Creating a design library, orca_design_lib, results in the creation of a UNIX directory with the same name, inside your present working directory.
This library data structure or design database is called *Milkyway.*
Milkyway is also supported by Astro, Star-RCXT, Hercules, JupiterXT, and Physical Compiler.
The Milkyway database is being expanded to work for all Synopsys' tools, including PrimeTime.

8 UNIX Manipulation of a Milkyway Database

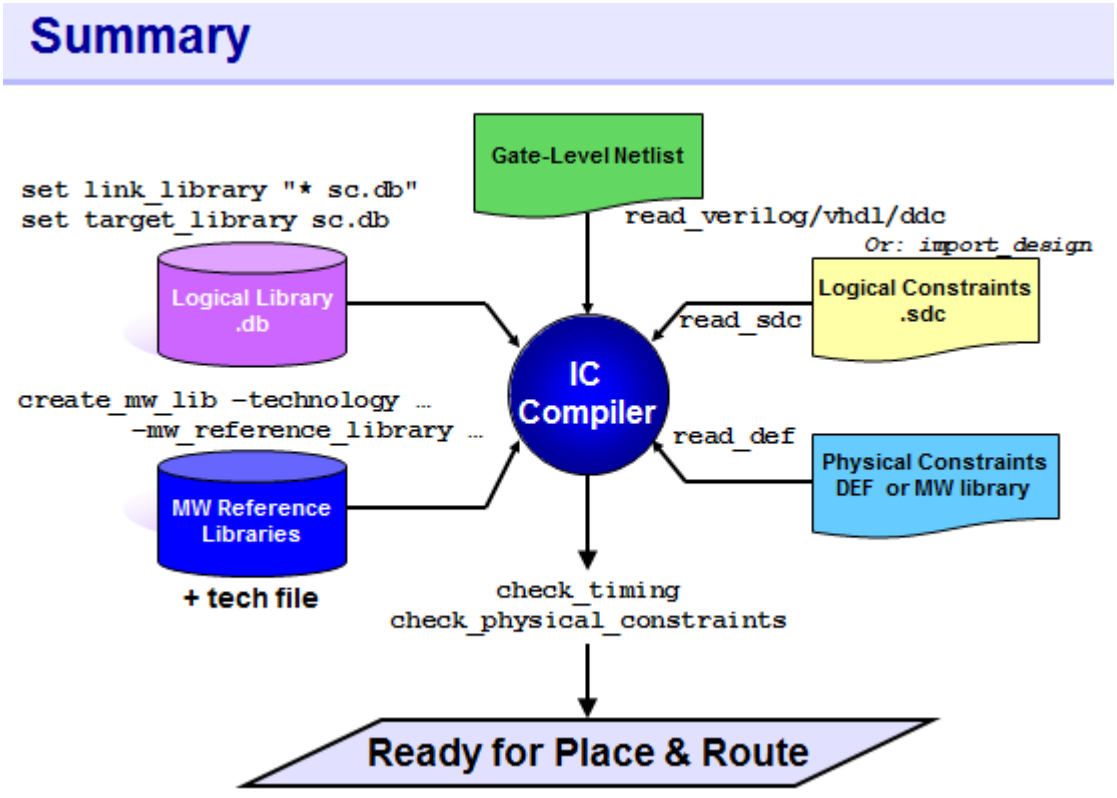
In the example above, the UNIX *rm* and *cp* commands were used to replace our CEL view ORCA_placed with Joe's version. Doing so will corrupt the database, more specifically the lib files, which contain the binary Table of Content (T.O.C.) of the library database.
Likewise the *mv* and *mk_dir* commands should not be used to manipulate the directory and file structure under the library directory (example: design_lib_orca).
Instead, use the commands rename_mw_cel, copy_mw_cel, remove_mw_cel. They are Milkyway-aware, UNIX is not.
Note: It is acceptable to copy (cp –r) or delete (rm –r) the entire library directory and its contents.
If files are deleted, it is possible to rebuild the table of contents using rebuild_mw_lib.

9. test for understanding

1. **List the 2 variables that need to be set up to successfully read all design files!**
   link_library, target_library
2. **What is the difference between the link_library and the target_library?**
   The target_library is the library that IC Compiler uses to pick cells for optimization and re-mapping. It is typically set to only the standard cells library. The link_library contains every library that contains cells that are referenced by the netlist.
3. **IC Compiler requires a chip-level floorplan including IO PADs.** ***True / False***
   False. IC Compiler can use block or chip-level floorplans. PADs are not required, but at least port locations are!
4. **A floorplan must always be input to IC Compiler by reading a DEF file.** ***True / False***
   False. IC Compiler can copy a floorplan from an existing MW cell as well.
5. **Which of the following is <u>not</u> recommended?**
   a. **lappend search_path my_path**
   b. **set search_path my_path**
   c. **set search_path "$search_path my_path"**

   Choice b is not recommended since it will overwrite the existing search_path value.

10 Summary

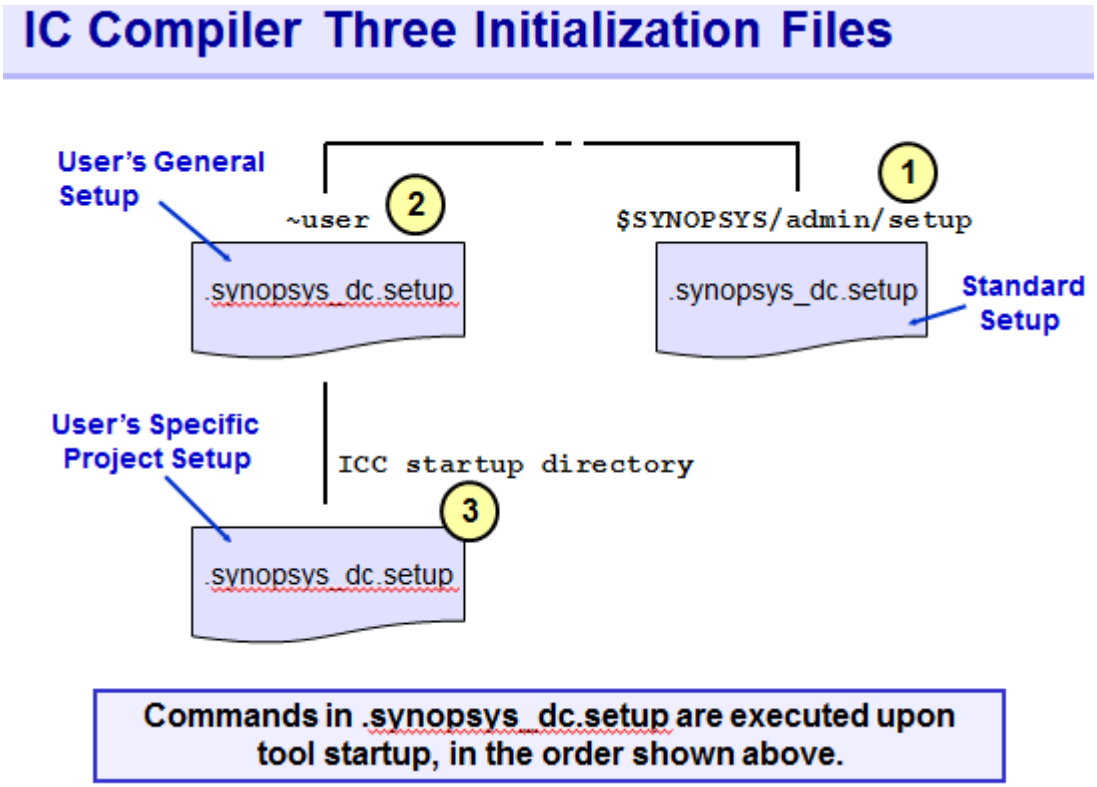


11. ICC recommended setup

```
# load common settings & useful procedures
source ../ref/icc_settings.tcl
lappend search_path ./scripts ../ref/sdb ../ref/db
set symbol_library "sc_icon.sdb io_icon.sdb"
set link_library "* sc_max.db io.db ram16x128_max.db"
set target_library "sc_max.db"
```

The example above reads a DDC file created by Design Compiler. The DDC file contains timing constraints already, therefore read_sdc was not used.
The following two variables are set by default by the tool. If logic 0/1 are connected to different supply names, settings must be changed.
set mw_logic0_net “VSS”
set mw_logic1_net “VDD”
If the settings are changed, you need to make sure that the variables are changed before you read the design into IC Compiler.
Note that import_designs has to be run after the design library has been created using create_mw_lib.

12 IC Compiler Three Initialization Files



3. import design

```
import_designs orca.v \
      -format verilog    \
      -top ORCA_TOP
```

Can be replaced by:

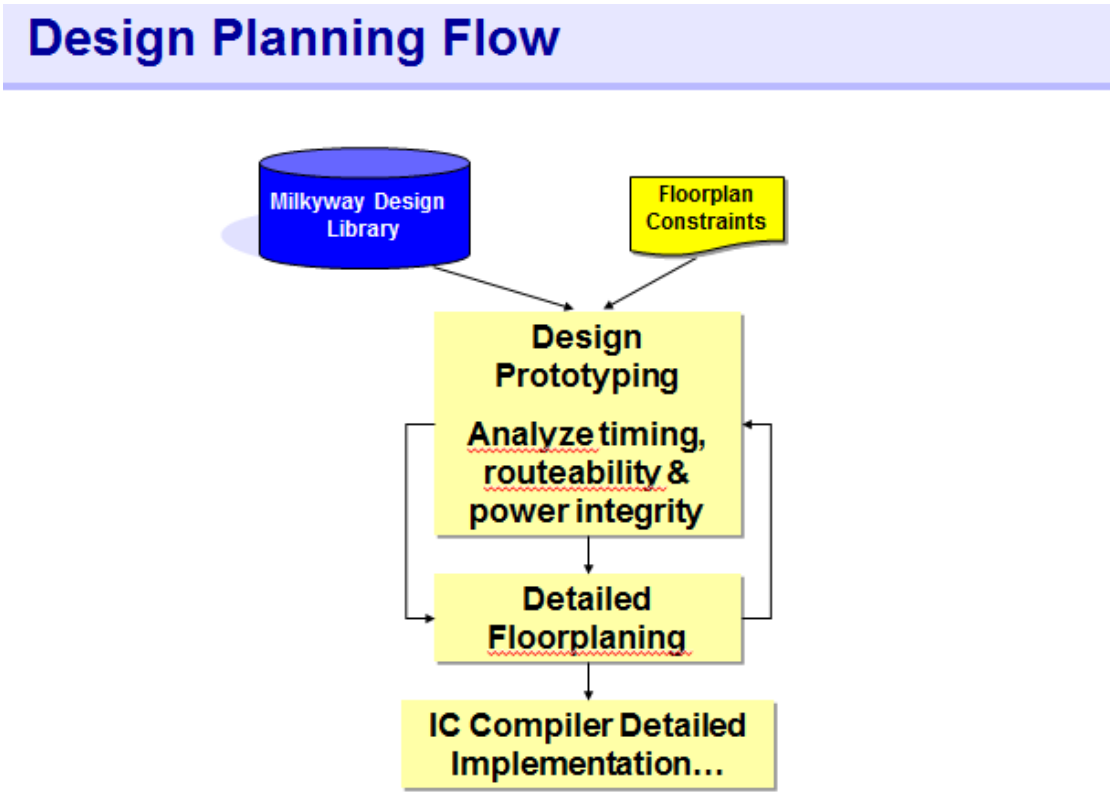
```
read_verilog –netlist orca.v
current_design ORCA_TOP
uniquify
link
save_mw_cel –as ORCA_TOP
```

4 design planning

IC Compiler Flow

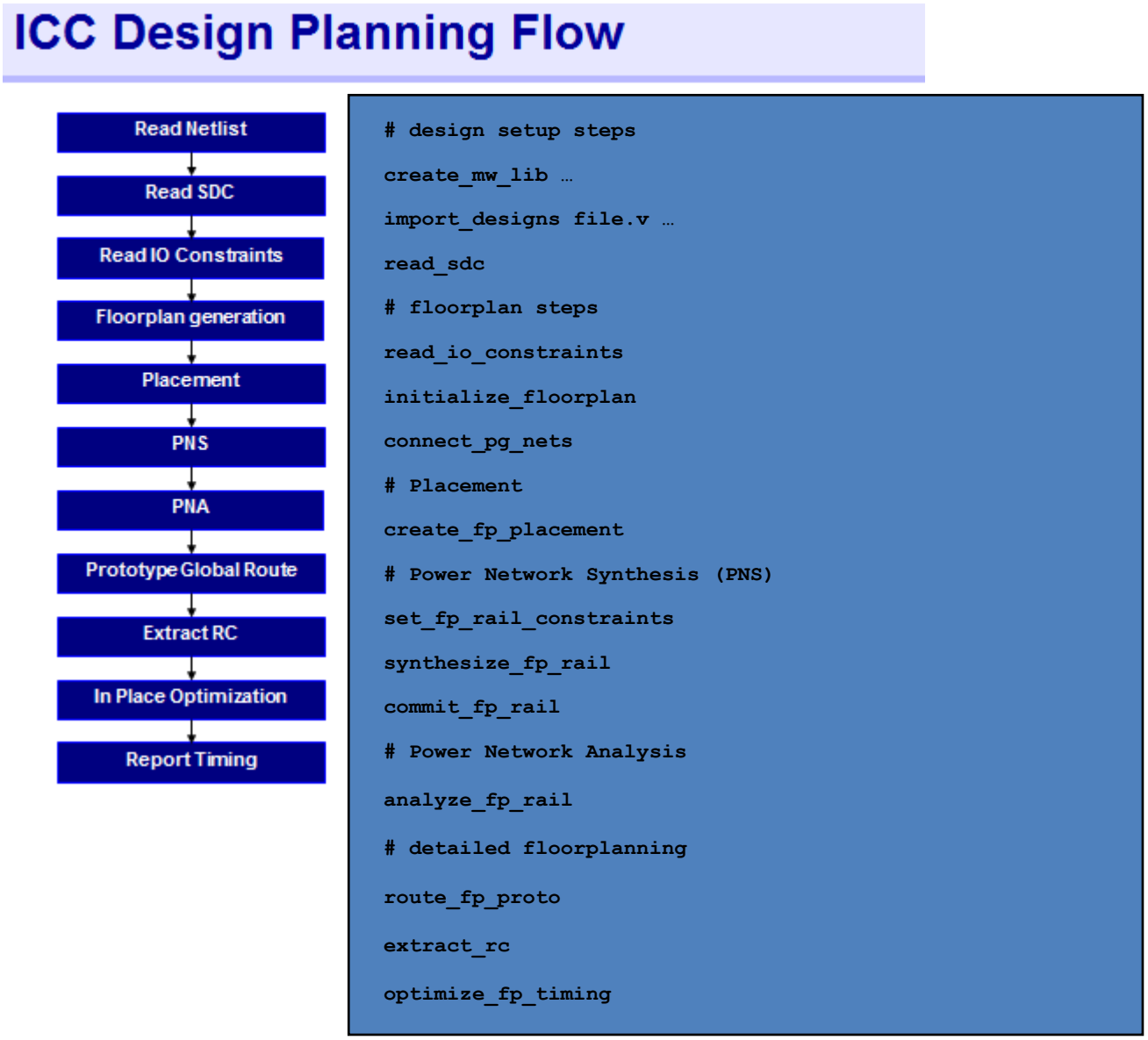
DP Features Flat vs. Hierarchical

Design Planning Flow



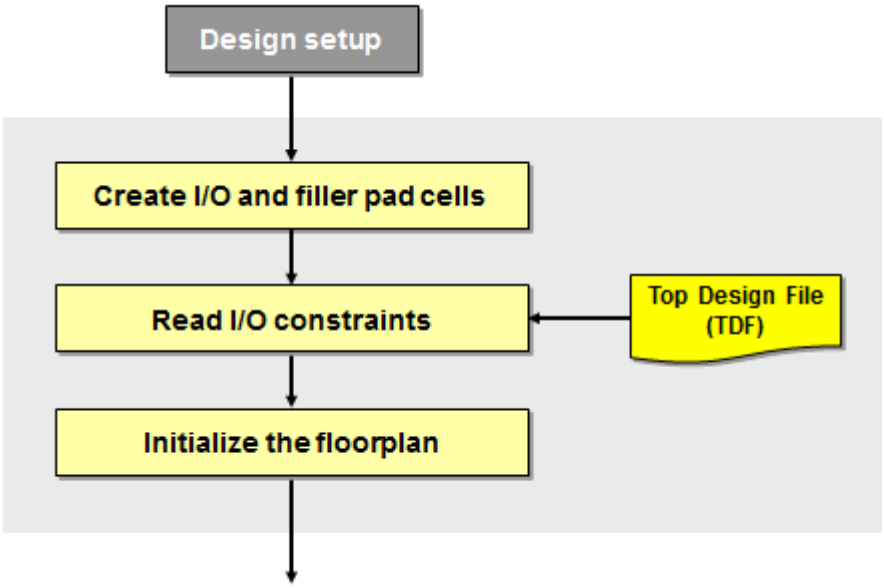
The Milkyway library was created during the design setup stage.
It contains the design connectivity (netlist), timing/area/power constraints, pointers to required reference libraries, TLU+ models for your process and the UNIX directory structures to hold it all.

ICC Design Planning Flow



If the design contains black boxes or the netlist is dirty, use the read_mw_verilog command in place of import_designs.
Also include adding of power pads (VSS,VDD) and insertion of pad fillers;

Create the Starting Floorplan



Connecting Power and Ground Ports

The macro cells and modules in your design contain power and ground pins that must be connected before initializing the floorplan. The `derive_pg_connection` command (or Preroute > Derive PG Connection in the GUI) connects power, ground, and tie-off pins to power and ground nets.

You can perform automatic power and ground connections for the power and ground pins of cells in the design, as well as direct rail-tie connections to power and ground nets by using the `derive_pg_connection` command. This is the only recommended method for creating a power and ground (PG) network for nonmultivoltage designs. Use this command before using any of the optimization commands.

`icc_shell>` **derive_pg_connection -power_net VDD -power_pin VDD **
-ground_net VSS -ground_pin VSS

In the case of a single voltage design, the `-power_net` *name* and `-ground_net` *name* options are required.

Using the GUI Menu Command to Perform Power and Ground Connections

To perform automatic power and ground and tie connections,
1. Choose Preroute > Derive PG Connection.
The Derive Power Ground Connection dialog box appears.
2. Set the options, depending on your requirements.
• Auto connection – Performs automatic power and ground connections for the power and ground pins of cells in the design as well as direct rail-tie connections to the power and ground nets. This is the default.
• Reconnect power/ground pins with existing connections – Uses existing connections to reconnect all power and ground pins in the design.

• Create power/ground nets from UPF supply nets – Creates the power and ground nets based on power domains defined in the multivoltage design and from UPF supply nets.
• Perform both power and tie connections – Performs direct rail tie-off connections in the design. For multivoltage designs with power domains, the tie pins are connected to the proper power nets consistent with their cell instances' power connections. For a single voltage design, the power and ground nets specified in the "Power net" and "Ground net" text boxes are used as tie-off nets.
• Show detailed connection information – Displays detailed information of the changes to the power and ground connections.
• Manual connections – Performs manual power and ground connections for designs with no power domain information.
Enter the names of the power and ground nets to use for the power and tie-high connections and the ground and tie-low connections.
Enter the names of the power and ground pins on cells to connect to the specified power and ground nets.
• Create port – Creates I/O ports for the specified power and ground nets. Select "Top" to create top-level power and ground ports only. Select "All" to create ports on the child cells (soft macros and black boxes). The default is "None" (no ports are created).
• Cells – Enter a list of cells to connect to the specified power and ground nets.

Note:
Make sure the PG network already exists in the design before you run the `derive_pg_connection -tie` command. If you run the `derive_pg_connection -tie` command before the PG network is created, it can cause unexpected results for the PG tie-off connections in the design flow.

Creating Power and Ground Ports(optional)

Depending on the floorplan creation flow you use, you can add top-level power and ground ports at the same time that you connect the power and ground pins in your design. Use the `derive_pg_connection` command with the `-create_ports top` option to automatically create ports for the nets specified by the `-power_net` and `-ground_net` options. In the GUI,choose Preroute > Derive PG Connection. In the Derive Power Ground Connection dialog box that opens, select "Manual connection" and then select Top as the "Create port" option.

Adding Power, Ground, and Corner Cells(optional)

Physical-only cells for power, ground, and corner placement might not be part of the synthesized netlist and must be added to design. Use the `create_cell` command to add a leaf or hierarchical cell to the current design.

Setting the I/O Pad Constraints(optional)

Before initializing the floorplan, you can create placement and spacing settings for I/O pads by using the `set_pad_physical_constraints` command. This command specifies the pad cell

ordering, orientation, placement side, offset from die edge, and pad-to-pad spacing for each I/O pad. After setting the constraints with the `set_pad_physical_constraints`command, the `initialize_floorplan` command places the I/O pad cells accordingly. The constraints are stored in the Milkyway database when you save the design.

The `initialize_floorplan` command places constrained pads first. Any unconstrained pads are placed next, using any available pad location. The tool does not place unconstrained pads between consecutively ordered constrained pads.

The following example script portion describes two corner pad locations and several I/O pad locations for the floorplan.

```
create_cell {cornerll cornerlr cornerul cornerur} cornercell
create_cell {vss1left vss1right vss1top vss1bottom} vsscell
# additional create_cell commands not shown
set_pad_physical_constraints -pad_name "cornerul" -side 1
set_pad_physical_constraints -pad_name "cornerur" -side 2
# additional corner pad constraints not shown
set_pad_physical_constraints -pad_name "pad_iopad_0" -side 1 -order 1
set_pad_physical_constraints -pad_name "pad_iopad_1" -side 1 -order 2
# additional left side pad constraints not shown
set_pad_physical_constraints -pad_name "pc_be_iopad_0" -side 2 -order 1
set_pad_physical_constraints -pad_name "pc_be_iopad_1" -side 2 -order 2
# additional top side pad constraints not shown
set_pad_physical_constraints -pad_name "sdram_A_iopad_0" -side 3 -order 1
set_pad_physical_constraints -pad_name "sdram_A_iopad_1" -side 3 -order 2
# additional right side pad constraints not shown
set_pad_physical_constraints -pad_name "CK_iopad" -side 4 -order 1
set_pad_physical_constraints -pad_name "CKn_iopad" -side 4 -order 2
# additional bottom side pad constraints not shown
```

Setting the Pin Constraints

You can use the `set_pin_physical_constraints` command to set constraints on individual pins or nets. Use the `set_fp_pin_constraints` command to set global constraints for a block. If a conflict arises between the individual pin constraints and the global pin constraints, the individual pin constraints have higher priority. The constraints are stored in the Milkyway database.

Saving the Pin and Pad Constraints

You can save the current pin and pad constraints for your design with the `write_pin_pad_physical_constraints` command. This command creates a constraints file that contains `set_pin_physical_constraints` and `set_pad_physical_constraints` commands that you can use to reapply pin and pad constraints. The positional information for the pins and pads is based on their current location in the design.

To create the pin and pad constraints file, use the `write_pin_pad_physical_constraints` command (or by choosing Floorplan > Write Pin/Pad Physical Constraints in the GUI).

Reading an Existing Pad and Pin Constraints File

Use the `read_pin_pad_physical_constraints` command (or by choosing Floorplan >Read Pin/Pad Physical Constraints in the GUI) to read a file that contains `set_pin_physical_constraints` and `set_pad_physical_constraints` commands. The `read_pin_pad_physical_constraints` command applies the constraints to the current design or to another design that you specify.

The constraints defined in the file can either remove the current constraints or append to them, based on the behavior you choose. For example, if physical constraints for pin A are specified and you do not define any constraints for pin A in the constraints file, the `read_pin_pad_physical_constraints` command removes the existing physical constraints on pin A by default. To maintain the existing constraints and append new constraints contained in the constraints file, specify the `-append` option (or click the"Append" check box in the GUI).

Reporting the Pad and Pin Constraints

Use the `report_pin_pad_physical_constraints` command to display a list of `set_pin_physical_constraints` and `set_pad_physical_constraints` commands that define the pin and pad constraints for the current design. You can report only pin constraints, only pad constraints, only chip-level pad constraints, or all constraints depending on the command options you specify.

Removing the Pad and Pin Constraints

Use the `remove_pin_pad_physical_constraints` command to remove all constraints previously set by using the `set_pin_physical_constraints` and `set_pad_physical_constraints` commands. You can remove only pin constraints, only pad constraints, only chip-level pad constraints, or all constraints based on the command options you specify. You can also remove constraints from a design other than the currently open design.

Core Area

Control Parameters

*** Aspect Ratio**

→Utilization

→Aspect ratio (H/W)

→Row/core ratio

*** Width & Height**

→Width

→Height

→Row/core ratio

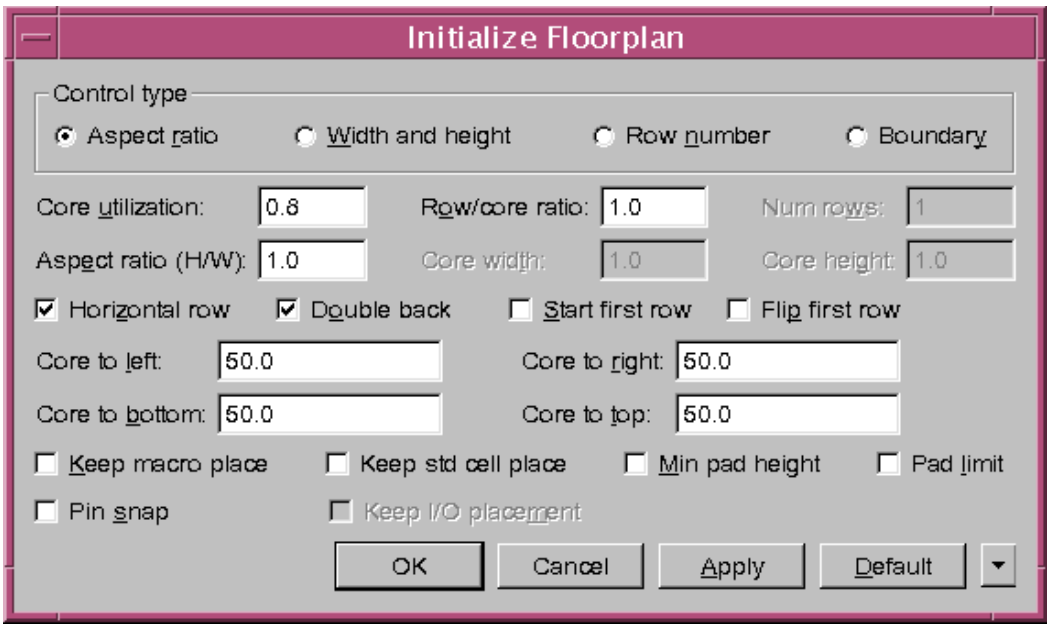
Initialize the Floorplan

Initializing a Rectangular Floorplan

Using the `initialize_floorplan` command, you can define a rectangular chip boundary and periphery based on aspect ratio, specific width and height, or number of cell rows. The command also places I/O pad and corner cells based on the constraints you defined by using `set_pin_physical_constraints` commands.

To create the initial floorplan, use the `initialize_floorplan` command (or by choosing Floorplan > Initialize Floorplan in the GUI).

- **Generates the basic elements of the FP**
  - Place IO pads/pins, as defined in the TDF file.
  - Create chip/core boundary
  - Create rows and tracks honoring user defined values
    - aspect ratio/width & height/row number boundary
    - core utilization
    - …

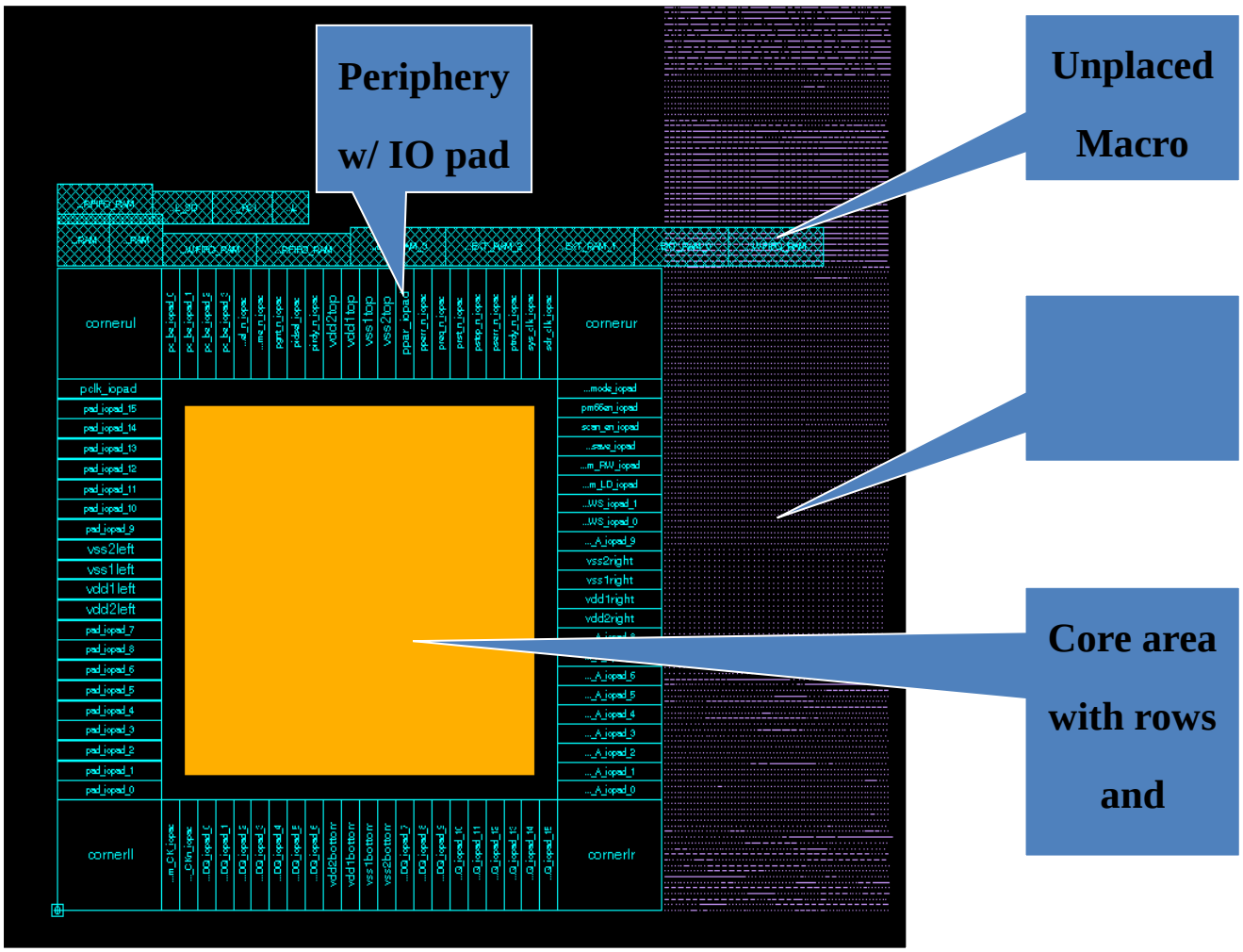


GUI: Floorplan → Floor Planner

When you select one of the four Control Param switches at the top of the dialog (e.g. aspect ratio or width & height), a corresponding subset of the six available parameter fields is highlighted to allow for user input

(example: Core Utilization, Core Aspect Ration (H/W), Row/Core Ratio, etc.)
For rectilinear, use initialize_rectilinear_block command.

Floorplan After Initialization



Refining the Floorplan(optional)

Adjusting the Floorplan

After you have created the floorplan by using the `initialize_floorplan` command or thebinitialize_rectilinear_block command, you can make adjustments by using the `adjust_fp_floorplan` command. The `adjust_fp_floorplan` command supports many of the same arguments as the `initialize_floorplan` command.

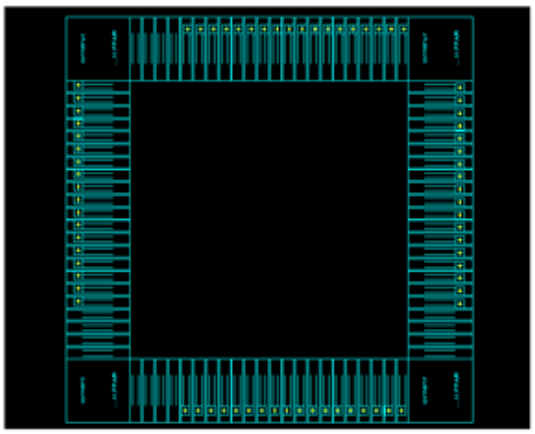
There is no GUI for the `adjust_fp_floorplan` command.

Figure 2-18 shows a floorplan that was initialized by using the `initialize_floorplan` command, and modified by using the `adjust_fp_floorplan` command. The `adjust_fp_floorplan` command makes the following changes:
• Changes the row orientation to vertical

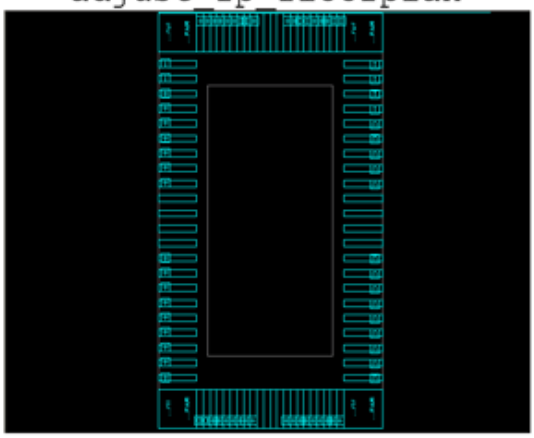
• Expands the spacing between the core and I/O pads from 30 microns to 100 microns on the left and right
• Expands the spacing between the core and I/O pads from 30 microns to 300 microns on the top and bottom
• Changes the core aspect ratio to 3.0

```
icc_shell> initialize_floorplan -core_utilization 0.8 -left_io2core 30 \
-bottom_io2core 30 -right_io2core 30 -top_io2core 30
icc_shell> adjust_fp_floorplan -use_vertical_row true -left_io2core 100 \
-bottom_io2core 300 -right_io2core 100 -top_io2core 300 \
-core_aspect_ratio 3.0
```

Original Floorplan



Floorplan after `adjust_fp_floorplan`



Manually Modifying the Floorplan

You can perform detailed corrections to the floorplan, such as changing the die area, modifying standard cell rows, and changing wire tracks.

Adjusting the I/O Placement

You can refine the spacing and relative locations of I/O pads by using the `adjust_fp_io_placement` command (or by choosing Floorplan > Adjust I/O Placement inthe GUI). The `adjust_fp_io_placement` command operates on one side of the chip at a time and supports an undo function that reverts the design to its previous state.

Saving the Floorplan Information

You can write the current floorplan to a Tcl file by using the `write_floorplan` command (or by choosing Floorplan > Write Floorplan in the GUI). The `write_floorplan` command creates a Tcl file that contains objects, placement, and attribute information for I/Os, terminals, standard cells, hard macros, and routing tracks in the floorplan. The Tcl file can be used to rebuild the floorplan by using the `read_floorplan` command. You can control the information written to the Tcl file by using command options. The command does not write relative placement group and relative placement keepout information.

You can save the current floorplan to a DEF file by using the `write_def` command (or by choosing File > Export > Write DEF in the GUI). The `write_def` command writes the floorplan, including design netlist, layout, and constraint information.

Reading In an Existing Floorplan

To read a DEF file, use the `read_def` command (or by choosing File > Import > Read DEF in the GUI).

You can use the `read_floorplan` command to import a previously saved floorplan file. Afloorplan file is Tcl script file created by using the `write_floorplan` command (or by choosing Floorplan > Write Floorplan in the GUI).

Perform initial virtual flat placement

Virtual flat placement is the simultaneous placement of standard cells and macros for the whole chip. The initial virtual flat placement is very fast and is optimized for wire length, congestion, and timing. For designs with macros, plan groups, or voltage areas that have not already been placed, virtual flat placement can help you decide on the locations, sizes, and shapes of the top-level physical blocks.

Evaluating Initial Hard Macro Placement

No straightforward criteria exist for evaluating the initial hard macro placement. Measuring the quality of results (QoR) of the hard macro placement can be very subjective and often depends on practical design experience. You should observe some basic rules when measuring QoR, such as pushing hard macros to the core boundary and aligning similar hard macros. However, the critical measurements that are used to evaluate the placement results are timing and routability.

QoR can be measured by the following criteria:
• Routability
You can measure the routability of your design by analyzing the routing congestion produced by the global router. The data used to analyze congestion consists of a textual report and visual heat maps that show where the routing congestion hot spots exist in the design. Highly congested designs are generally not routable. Hard macros tend to create congestion around their edges and corners. You should analyze hard macro placements for routability early in the design cycle.
• Timing
Timing calculations can use the placement information to better estimate interconnect loading. The interconnect timing calculations should take into account detours in routes caused by the presence of hard macros. Good placement will minimize timing violations.
• Wire length
Smaller values of total wire length are a good indicator of placement quality. The total wire length is output in the placement log file. You should record and monitor this number when assessing multiple placement solutions.

Another aspect of wire length is localized to hard macros. By looking at the signal (flyline)connections of a hard macro, you can quickly determine whether the hard macro is placed in an optimal location. A hard macro placed in the lower-left corner of the core area that is connected to logic in the upper-right corner of the die indicates a poor location for the hard macro.

• Data flow

If you know the logic and intended operation of the design, you can assess the data flow by using the hierarchical browser at any stage in the design flow to observe where logical modules and hard macros are physically placed. Using this technique can help you make sensible placement decisions.

• Standard cell placement areas

You can visually assess the placement of macros and standard cells. Small areas surrounded by hard macros usually cause congestion hot spots. Unless the connections to and from standard cells in these areas are completely localized, it is difficult to complete the connections from within these areas to the objects outside these areas.Generally, a contiguous standard cell placement area without bottlenecks is desirable.

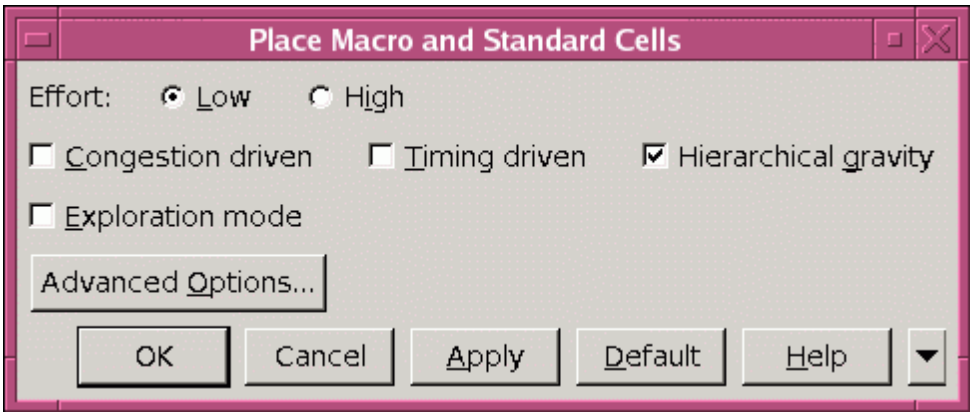
This placement is “virtual” because it temporarily considers the design to be entirely flat, without hierarchy. After you decide on the shapes and locations of the physical blocks, you restore the design hierarchy and proceed with the block-by-block physical design flow.

Virtual flat placement is typically done after the floorplan has been initialized with the `initialize_floorplan` command or by using Floorplan > Initialize Floorplan in the GUI, but before any macros or standard cells have been placed. The chip boundary should be defined, the I/O pad cells placed, and the site rows defined. The timing constraints should be set and meeting these constraints should be reasonably achievable. The power structure is typically not yet defined, except possibly for the I/O pads.

Using virtual flat placement typically consists of the following steps:

1. Set the virtual flat placement strategy options by using the set_fp_placement_strategy command.
2. Set the macro placement options by using the set_fp_macro_options command.
3. Define any necessary relative placement constraints.
4. Set the blockage, margin, and shielding options.
5. Perform virtual flat placement by using the create_fp_placement command.
6. Analyze the results for timing and congestion. Perform manual editing, if applicable.
7. If the results are not satisfactory, repeat steps 1 through 6.

To perform virtual flat placement using the GUI layout window, first set the window task mode to Design Planning if it is not already in that mode (File > Task > Design Planning). Then choose Placement > Place Macro and Standard Cells.

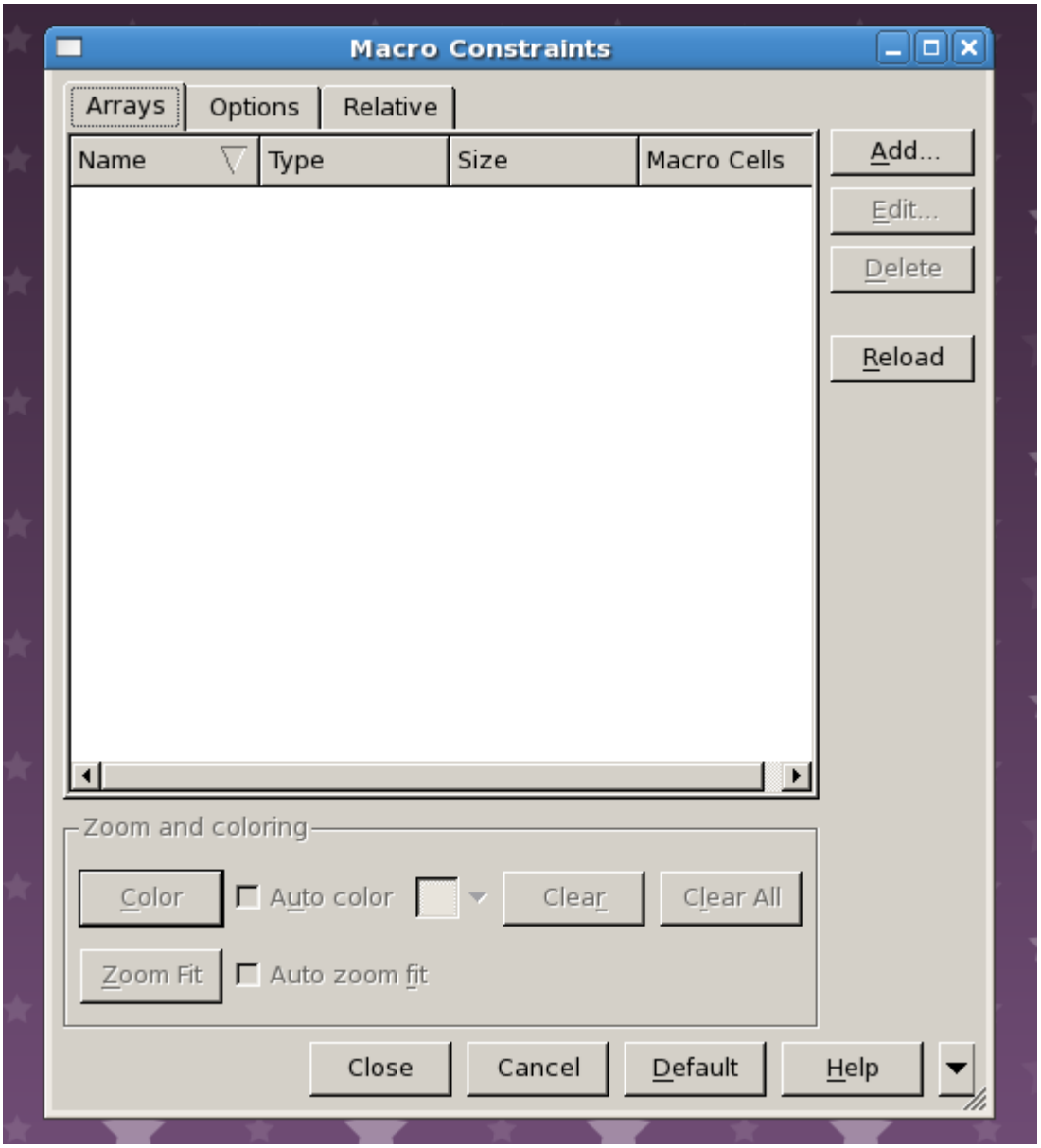


You can set the effort level to Low (the default) or High. The Low effort mode produces good placement results for hard macros and plan groups. Select the High effort mode for even better quality of results at the cost of more runtime.

Specifying Hard Macro Placement Constraints

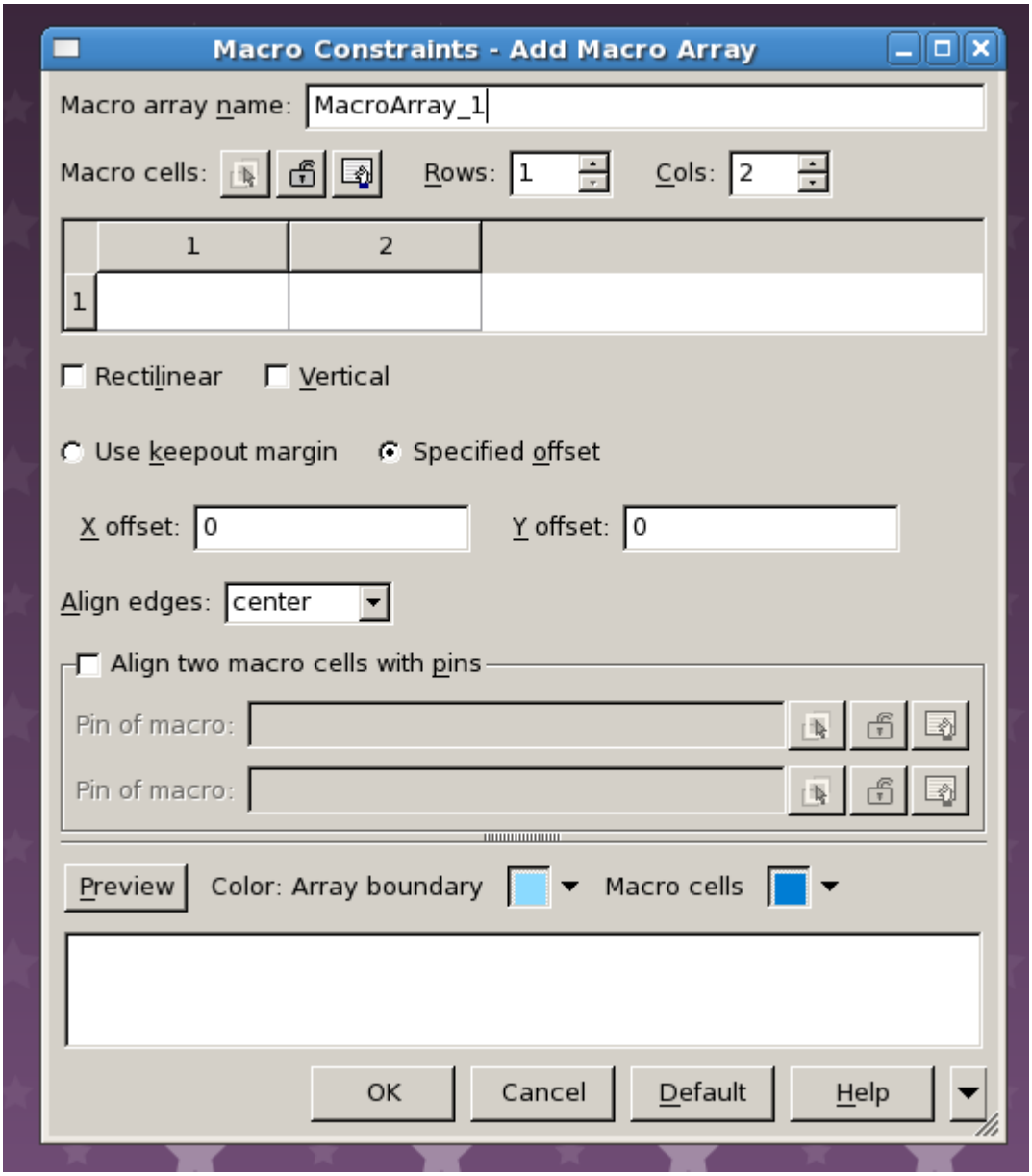
Creating a User-Defined Array of Hard Macros

To create user-defined array of hard macro cells,
1. Choose Placement > Macro Constraints.
The Macro Constraints dialog box appears.



Alternatively, you can use the `set_fp_macro_array` command.
Each constraint is displayed with a unique name, type, and list of macros or macro arrays.
You can sort the list by name or type by clicking on the column header. Selected macro constraints are annotated graphically in the layout window. The original macro cells are also highlighted in the layout window.

2. To create a new macro array constraint, select the Add button on the Macro Constraints dialog box.From the menu choose Macro Array.The Add Macro Array dialog box appears.



3. Set the options, depending on your requirements.
• Macro array name – Specify the macro array name. This name is required.
• Macro cells – You can set the order for the macro cells that will compose the array by using lists and sublists. The macro array template shows how selected macro cells are arranged into an array.

Specify each row of the array as a single list. Each sublist represents a row in the array. An array starts from the topmost row, and a row starts from the left. Based on the list, IC Compiler automatically determines whether to create a one-dimensional or two-dimensional array.
• Rectilinear and Vertical – Select the Rectilinear option to create a two-dimensional array with a rectilinear outline. You can also use the Vertical option with the Rectilinear option to create a vertical row structure for the two-dimensional array. By default, a rectilinear array has a horizontal row structure.
• Use Keepout Margin – When arraying two adjacent macros, you can apply the Use Keepout Margin option to leave the smallest possible spacing between the rows or columns of the two macro arrays. The spacing is greater than or equal to the keepout margins on the two macros.
This option is mutually exclusive with the X offset and Y offset options.

• Specified offset – You can specify the distance between two adjacent cells in a macro array in the x-direction, y-direction, or both directions by using the X offset and Y offset options.
• Align edges – You can align one row of the macro cells of a heterogeneous one-dimensional macro array along an edge that you specify by selecting this option. The edge can be top, bottom, left, right, or center. The default is bottom.

• Align two macro cells with pins – You can select this option when you want to create two-dimensional macro arrays and align numerous small macro cells that have uneven shapes.

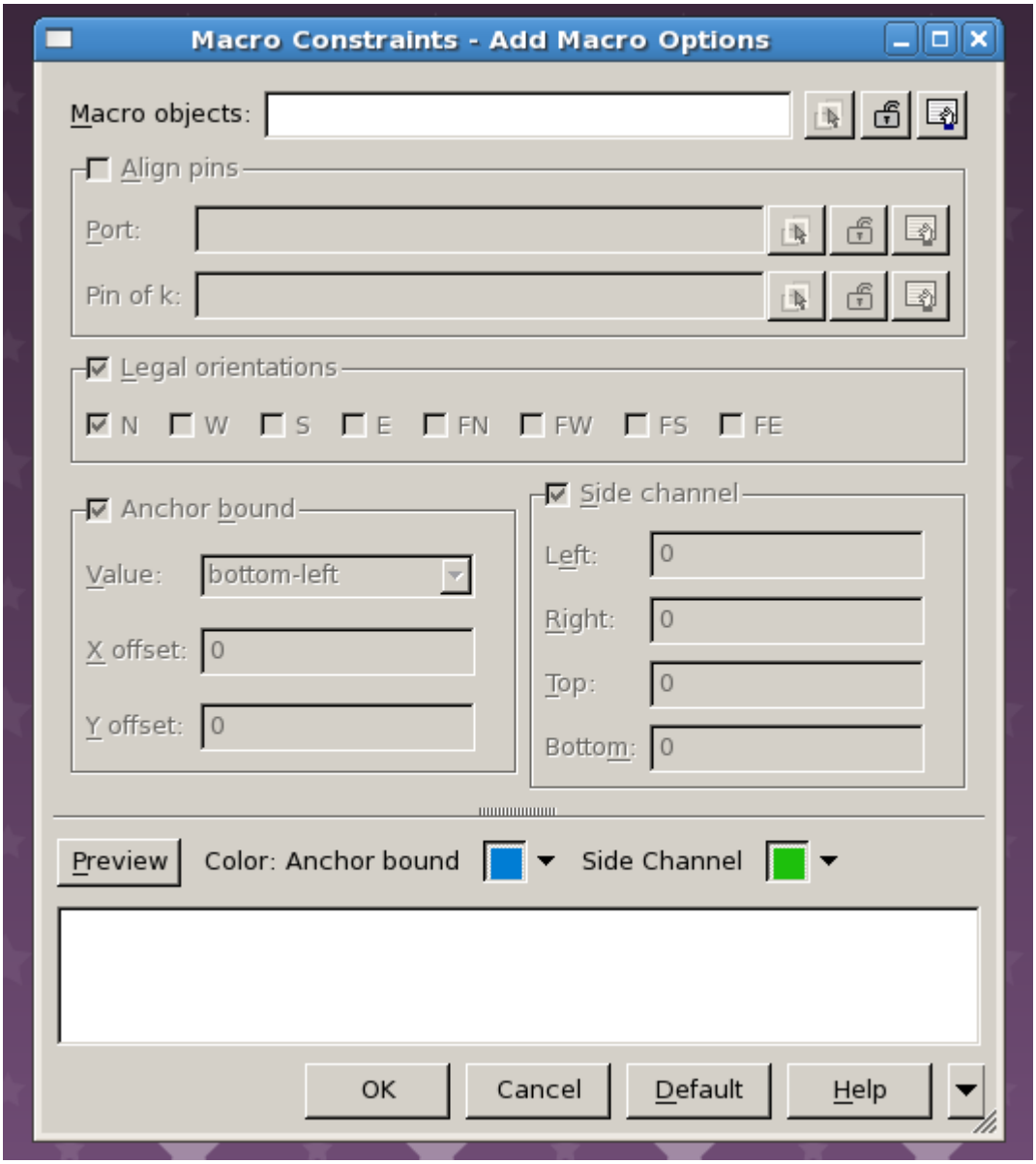
4. Click OK.

Setting Floorplan Placement Constraints On Macro Cells

During floorplanning, you can define placement constraints that restrict specified macro cells and macro arrays to particular regions, orientations, alignments, and so forth. This constrains the `create_fp_placement` command in placing the macro cells in the floorplan.

When you specify the constraints on macro cells and macro arrays, keep the following points in mind:
• The physical library and design must be loaded.
• Set these floorplan placement constraints before you run the `create_fp_placement` command.
• During placement, macro cells can be flipped or rotated according to legal orientation constraints from the library or by constraints you set.
• You must provide a list of macro cells to which the constraints apply. You can provide this list by using the `get_cell` command (for example, [`get_cell RAM`]). If you list a macroncell that is not found, an error occurs.
• To report the options that are set, use the `report_fp_macro_options` command. The report provides details about a particular macro cell or array or all the macro cells or arrays. Information includes the name of the macro cell or macro array, anchor bounds,offsets, and legal orientation for macro cells. (Legal orientation does not apply to macro arrays.)

To set floorplan placement constraints that restrict specified macro cells and macro arrays during virtual flat placement,
1. Choose Placement > Macro Constraints.
The Macro Constraints dialog box appears.
Alternatively, you can use the `set_fp_macro_options` command.
Each constraint is displayed with a unique name, type, and list of macros or macro arrays.
2. To create a new macro options constraint, select the Add button on the Macro Constraints dialog box.
From the menu choose Macro Options.
The Add Macro Options dialog box appears.



3. Set the options, depending on your requirements.

• Macro objects – Enter a list of macro cells to be constrained.

• Legal orientations – Select this option to restrict the orientation to a valid orientation that you specify. If you do not use this option, orientation is not restricted during placement. The default is enabled.

You can provide a single orientation to force the placement engine to place the macro cell in a fixed orientation. After the cell orientation is fixed, the placement tool cannot change the orientation.

• Align Pins – You can align a specified port or pin to a pin of a constrained macro cell by using the this option. The argument is a list of two objects: reference port or pin followed by the constrained pin.

Placing a Macro Cell Relative to an Anchor Object

To place a macro cell relative to an anchor object,

1. Choose Placement > Macro Constraints.

The Macro Constraints dialog box appears.

Alternatively, you can use the `set_fp_relative_location` command.
Each constraint is displayed with a unique name, type, and list of macros or macro arrays.
2. To create a new relative location macro constraint, select the Add button on the Macro Constraints dialog box.
From the menu choose Relative Location.

3. Set the options, depending on your requirements.
• Constraint name – Enter a name of the relative location constraint.
• Anchor object name – Enter the name of the anchor object. The object can be a plan group, a fixed macro cell, a macro cell that has its own relative location, or a core area. Specify the corner of the anchor object's bounding box that the constraint uses. You can choose from the following designations: bl (bottom-left corner), br (bottom-right corner), tl (top-left corner), and tr (top-right corner).
• Target macro – Specify the name of the macro cell to which the constraint is applied.
• Orientation – Specify the orientation in which to place the target cell. You can choose from the following values: N, S, E, W, FN, FS, FE, and FW. The default is N.
• Corner – Specify the corner of the target cell on which to apply the constraint. You can choose from the following designations: bl (bottom-left corner), br (bottom-right corner), tl (top-left corner), and tr (top-right corner). The default is bl.
• X offset – Specify the distance in microns from the target cell to the anchor corner of the anchor cell in the X-dimension. You can enter a positive, negative, or zero value. The default is 0.
• Y offset – Specify the distance in microns from the target cell to the anchor corner of the anchor cell in the Y-dimension. You can enter a positive, negative, or zero value. The default is 0.

Reporting Relative Location Constraints

You can use the `report_fp_relative_locations` command to report macro relative location constraints set by the `set_fp_relative_location` command.

Removing Relative Location Constraints

You can remove relative location constraints set on the specified target cells and any other cells that are anchored to these cells by using the `remove_fp_relative_location` command. By default, all relative location constraints are removed.

Using a Virtual Flat Placement Strategy

Use the `report_fp_placement_strategy` command to display floorplan-related option values set using the following commands.
They are grouped into four categories:
• Hard Macro
• Net Weighting
• Congestion
• Miscellaneous
Note:

You can set all the default values by using the `set_fp_placement_strategy -default` option.

Creating Macro Blockages for Hard Macros

If you do not want hard macros to be placed in certain areas of your design, you can create a macro blockage to restrict the placement of hard macros.

Padding the Hard Macros

To avoid placing standard cells too close to macros, which can cause congestion or DRC violations, you can set a user-defined padding distance or keepout margin around the macros. You can set this padding distance on a selected macro's master cell. During virtual flat placement, no other cells will be placed within the specified distance from the macro's edges.
To set a padding distance (keepout margin) on a selected macro's master cell,
1. Choose Placement > Set Keepout Margin.
The Set Keepout Margin dialog box appears.
Alternatively, you can use the `set_keepout_margin` command.
2. Specify the type of keepout margin to be created, either hard or soft. The default is hard
3. Choose whether to set a user-defined padding distance on all macros (the default), on specified macros, or on specified instances. Enter the names of the specified masters or instances, if applicable.
4. If you selected "All macros," you can choose to either derive the amount of padding based on the number of pins and track width, or you can specify explicit keepout distances on all four sides. If you select "Specified masters" or "Specified instances", you must specify explicit keepout distances on all four sides.
If you choose "Derive outer keepout coordinates", specify the following:
• The number of tracks per macro pin.
If you have macros with lots of pins, you should set the padding to 0.5 to 1.0 track per pin (0.5 reserves one track for every two pins) to reserve some routing area around the macros and give the router enough space to route to the macro's pin.
• The minimum padding per macro size.
This is useful when there are a few pins on some sides of some macros. The default is 0.
• The maximum padding per macro size.
The default is `-1`, which means that the placer would apply a 40x metal1 pitch limit on all macros.
If you choose "Use specified outer keepout coordinates", enter the explicit padding in microns in the Top, Bottom, Right, and Left fields.

5. Click OK or Apply.

Placing Hard Macros and Standard Cells

You can perform a virtual flat placement and obtain a flat placement of the hard macros and standard cells. Based on the placement results, you can then decide the relative locations,shapes, and sizes of the top-level logic blocks. If you have a design with plan groups or voltage areas, the

refined placement honors the exclusiveness of the plan groups and voltage areas.
Note:
The main consideration when placing a large hard macro is congestion; wire length becomes secondary. A hard macro is considered large if it takes up 5 percent or more of the chip area or 40 percent or more of the chip width. To improve routability, place large hard macros at the edges of the chip boundary.
To place the hard macros and standard cells simultaneously,
1. Choose Placement > Place Macros and Standard Cells.
The Place Macros and Standard Cells dialog box appears.
Alternatively, you can use the `create_fp_placement` command.
2. Click Default, and then click OK.

Controlling the Placement

This section describes the placement constraints that you can use to place hard macros and standard cells and control the placement results.

Choose Placement > Place Macro and Standard Cells. The Place Macro and Standard Cells dialog box appears.

Alternatively, you can use the `create_fp_placement` command.

• Selecting an effort level
The effort level controls the trade-off between the QoR and the amount of CPU runtime spent performing an initial flat placement. Select the low-effort option (the default) for very fast results that provide good plan group locations and hard macro locations.
Raising the effort level to high improves the quality of the placement, but incurs an increase in CPU runtime.

• Using the “Congestion driven” option
You can perform congestion-driven placement by selecting the “Congestion driven” option. The default is off. Standard cells and hard macros are placed together to minimize wire length and congestion in the design and to avoid cell overlaps.
Congestion-driven placement:

- Reduces congestion by adding padding to cells in congested areas.
  The amount of padding needed for each cell depends on the congestion around each cell.
- Moves hard macros and standard cells in congested areas.
- Pads macros in automatically generated arrays, thereby changing the size of the array.

If there is congestion between hard macros, congestion-driven placement increases the channel size and keepout region around the macros to reduce congestion. Additional cells are not placed in the channels. After placement, the command saves these keepouts to the file design_planning_blockages.tcl in the current working directory, and removes the added keepouts from the design. To re-insert the keepouts, source the file design_planning_blockages.tcl prior to the in-place optimization step.

• Using the “Timing driven” option
You can perform incremental timing-driven placement to improve the timing result for your design by selecting the “Timing driven” option. The default is off.
Timing-driven placement:

    - Improves the timing by moving cells on critical paths closer together.
    - Moves hard macros on critical paths.

• Using the “Hierarchical gravity” option

The “Hierarchical gravity” option tends to keep cells of a hierarchical design logic block physically together in the chip layout. This type of grouping usually results in better placement because designs tend to partition well along hierarchical boundaries.

However, you should deselect this option if you know that the logic hierarchy does not represent a good physical partitioning of the design. This allows the placement engine the flexibility to place cells of a hierarchical block far from other cells from the same block.The default is on.

• Choosing a maximum fanout

Enter a maximum fanout as a positive integer. The wire lengths of any nets with a fanout higher than the number you specify are ignored by the placement. The default is 512.

• Performing an incremental placement

If you want the placement to start with existing cell locations and incrementally improve on them during the placement, select the “Incremental” option. Use this option if you want to start with a good relative placement but do not want the cells to move too far from their current locations. (If cells need to be moved far from their current locations and you use this option, the result will be a poor placement.) This option is off by default, and the placer ignores existing cell locations and creates a placement from scratch.

Another reason to use this option would be if you have already run a placement on the design with placed plan groups inside the core area and there is a congestion or timing problem.

When plan groups are placed inside the core area, enable the Incremental option and then select from the following three suboptions to further refine the placement on selected parts of your design.

All – Runs incremental placement on all the standard cells and hard macros in the design. This is the default.

Top level cells only – Runs incremental placement only on the top-level cells that do not belong to a plan group.

Specified plan groups – Performs a refined placement on only the selected exclusive plan group cells. You can select one or more plan groups.

Specified voltage areas – Performs a refined placement on standard cells and hard macros within the specified voltage areas. The virtual flat placement engine will recognize voltage areas as physical constraints in the floorplan.

To create a floorplan with multiple voltage areas, you need to specify each logic module that belongs to the separate top-level power domains. Placement voltage areas are used to constrain the placement of cells to certain locations, based on the cell’s power domain. A power domain is a set of cells with the same voltage requirements. This means that each delineated power domain should include only cells driven from the same powersupply.

In a single-voltage design, the entire design operates under the same voltage, the default voltage. In a multivoltage design, you need to define areas where all cell instances use a higher or lower voltage than the default voltage. The rest of the design area (outside of any defined voltage areas) uses the default voltage. For multivoltage designs, the default voltage area equals the design boundary minus all the voltage areas.

• Legalizing the resulting placement

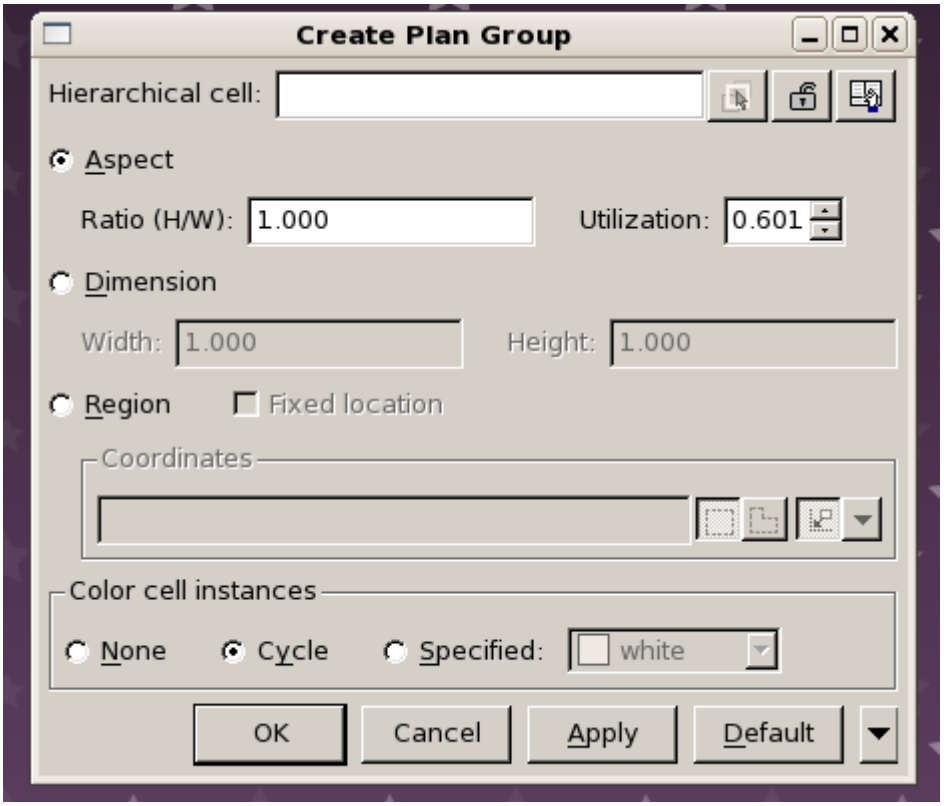
Select the “Legalize resulting placement” option to resolve cell placement conflicts after doing initial

placement. Overlaps are removed, and standard cells are placed at legal sites. Legalization is appropriate for the final placement. The default is on.
Alternatively, you can use the `legalize_fp_placement` command.

• Optimize pins
Select the “Optimize pins” option to perform simultaneous placement and pin assignment for block-level designs. The placement engine places the cells near the port locations according to the TDF pin constraints (side, side and order, and side and location) that you have set. The result is a simultaneous virtual flat placement and pin assignment for blocklevel designs with a smaller wire length of nets connected to the constrained pins
• Ignore scan chain connectivity
Select this option if you want the placement engine to ignore scan chain connections during virtual flat placement.
• Specifying the number of CPUs
Enter the number of CPUs to be used in parallel during initial virtual flat placement. The number you enter should be an integer value less than or equal to the number of free CPUs on your machine. The default is one CPU.

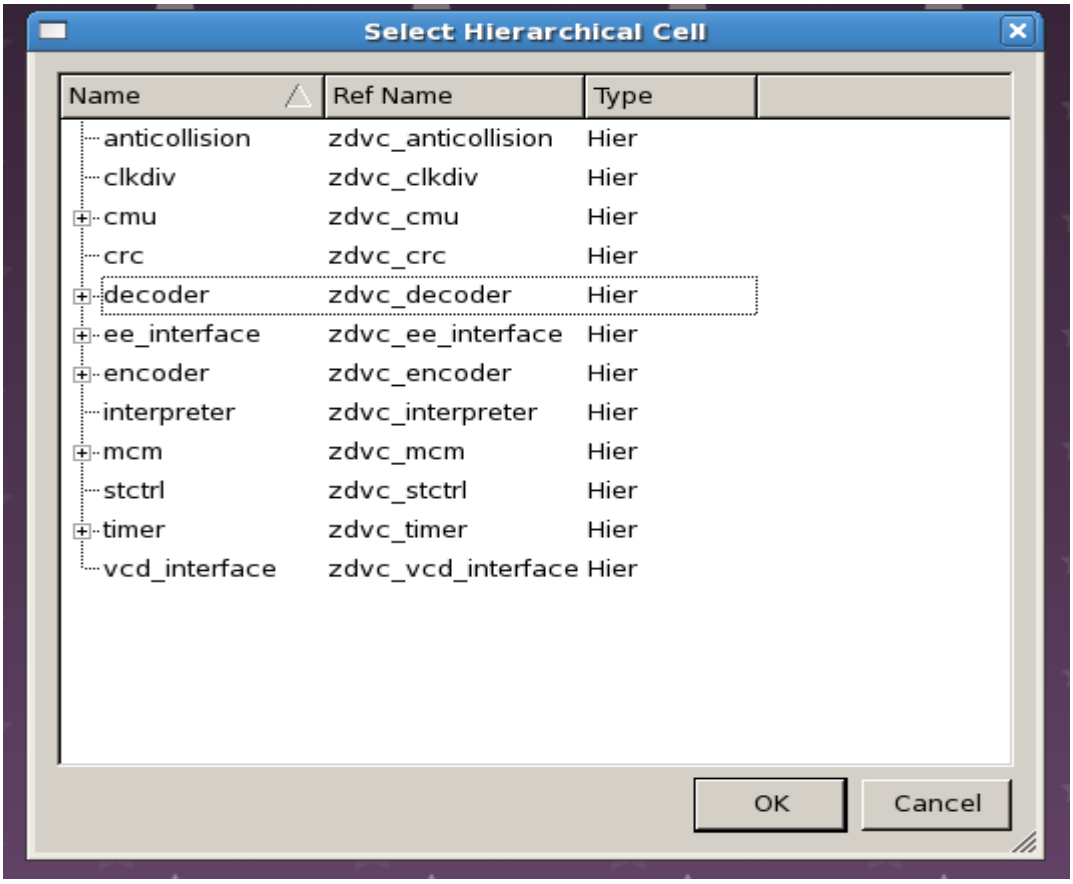
Create and shape plan groups

A plan group is a physical partition with a physical boundary.
To create a plan group,
1. Choose Floorplan > Create Plan Group.
The Create Plan Group dialog box appears.
Alternatively, you can use the `create_plan_groups` command.



2. Click in the “Hierarchical cell” text box and enter the hierarchical cell names. A plan group is

created for each valid hierarchical cell name in the list.



3. Define the plan group shape by doing one of the following:
• Select the Aspect option to define a plan group shape by its aspect ratio and utilization. This is the default.
Enter an aspect ratio (height divided by width) for the plan group area. If you specify a ratio of 1.00 (the default), the height and width are the same, and therefore the core is a square. If you specify a ratio of 3.00, for example, the height is three times the width.
Enter a utilization value. This number is calculated as the total plan group area divided by the area occupied by the cells of the plan group.
• Select the Dimension option to define a plan group shape by its dimensions and enter the width and height in microns. The plan groups are placed outside the core area.
• Select the Region option to define a plan group by using coordinates.
To define a rectangular plan group shape, select the Rectangle button and draw the rectangle in the layout view, or enter the lower-left and upper-right coordinates in the Coordinates field.
To define a rectilinear plan group shape, select the Rectilinear button and draw the rectilinear shape in the layout view, or enter the lower-left and upper-right coordinates in the Coordinates field.

4. (Optional) You can apply a specific color to a plan group. To do this, deselect the “Cycles colors” option to assign a new color to each new plan group automatically (the default).
Select the “Specified” option to specify a particular color for the plan group color from the list. Select “None” for no color.
By default, IC Compiler automatically assigns a color to the plan group.

Removing the Plan Groups

To remove (delete) plan groups from the current design, choose Floorplan > Delete Floorplan Objects > Delete All Plan Groups. This command lets you selectively delete one or more plan groups.
Alternatively, you can use the `remove_plan_groups` command.
Note:
Deleting a plan group means cancelling the grouping for future placement. The existing placed cells are not affected.

Adding Padding to Plan Groups

To prevent congestion or DRC violations, you can add padding around plan group boundaries. Plan group padding sets placement blockages on the internal and external edges of the plan group boundary to prevent cells from being placed in the space around the plan group boundaries. Internal padding is equivalent to boundary spacing in the core area. External padding is equivalent to macro padding.
You can pad the plan groups to ensure that the placed standard cells do not extend over the plan group boundary and that cells belonging to the plan group remain inside the plan group. The cells that do not belong to the plan group remain outside the plan group when the physical hierarchy is committed. The boundary space is also needed for pins and the routing to those pins.
You should pad plan groups before refining the placement to ensure that the placement honors the plan group boundaries.
Padding the plan groups also reserves space for the pins on soft macros.
When the physical hierarchy is committed, the padding is transferred to a soft macro cell as core-boundary spacing and not as placement blockages in the child soft macro.
The plan group padding is visible in the layout window. The plan group padding is dynamic, which means that it follows the changes of the plan group boundaries.
To add padding to plan groups,
1. Choose Floorplan > Create Plan Group Padding.
The Create Plan Group Padding dialog box appears.
Alternatively, you can use the `create_fp_plan_group_padding` command.
2. Specify the plan groups for which you want to add padding.
• All – Select this option to add padding for all plan groups. This is the default.
• Specified – Select this option to add padding for selected plan groups.
Enter the names of the plan groups.
3. Specify whether the padding is internal, external, or both.
• Internal – Select this option to add padding inside the plan group boundaries. Enter a width value. Internal padding should be at least 1 micron (the default) wide to allow space for pins and pin routing.
• External – Select this option to add padding outside the plan group boundaries. Enter a width value. The default is 0.
If you want to add both internal and external padding, select the “Internal” and “External” options and enter the width values.
4. Click OK or Apply.

Removing the Plan Group Padding

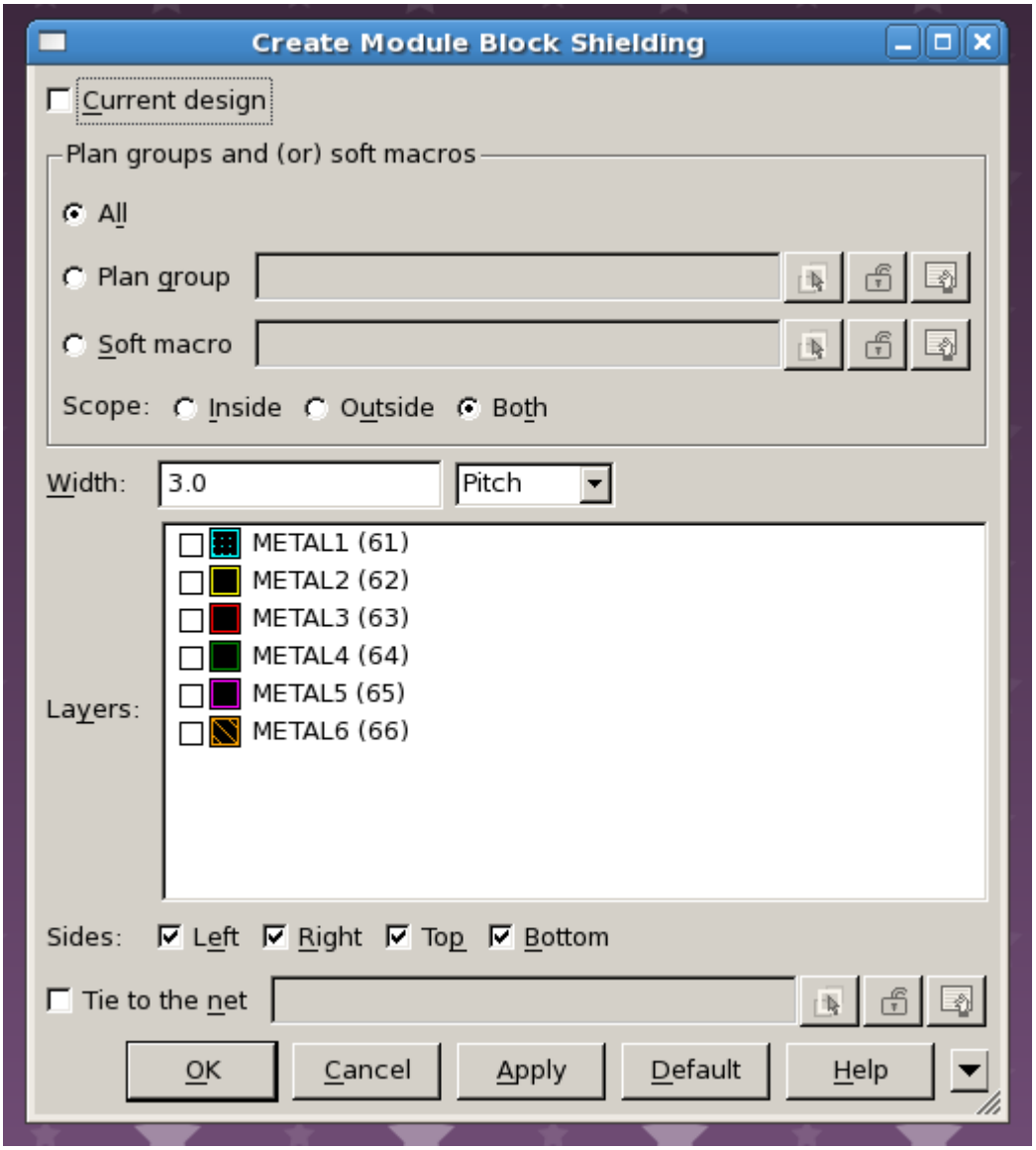
To remove (delete) both external and internal padding for the plan groups, choose Floorplan > Delete Floorplan Objects > Delete Plan Group Padding. The Delete Plan Group Padding dialog box appears.
Alternatively, you can use the `remove_fp_plan_group_padding` command.
You can delete padding for selected plan groups or for all plan groups. The default is all plan groups.

Adding Block Shielding to Plan Groups or Soft Macros

When two signals are routed parallel to each other, crosstalk can occur between the signals, leading to an unreliable design. You can protect signal integrity by adding modular block shielding to plan groups and soft macros.
The shielding creates rectangular metal layers around the outside of the soft macro boundary or plan group in the top level of the design, and around the inside boundary of the soft macro or plan group, or both.
To handle signal integrity when creating hierarchical designs, the router must be prevented from laying down routes that run collinear to the soft macro or plan group boundary. To do this, rectangles that the router creates are placed along and inside the soft macro's or plan group's edges on metal layers whose preferred direction crosses the soft macro or plan group boundary. If the plan groups are committed, the inner-boundary shielding is then transferred to the soft macro.

The rectangles act as hierarchical signal shielding by allowing only routes that are perpendicular to the soft macro or plan group boundary to pass through the metal layers. By preventing collinear routing along the soft macro and plan group boundaries, signal crosstalk among routing in the soft macro and routing in the top-level channels is minimized.

To add block shielding for plan groups or soft macros,
1. Choose Floorplan > Create Module Block Shielding.
The Create Module Block Shielding dialog box appears.
Alternatively, you can use the `create_fp_block_shielding` command.
2. Select the type of objects to which you want to add signal shielding.
You can add shielding for all plan groups and soft macros, or you can specify selected plan groups or soft macros. The default is "All".
3. Select a scope option.
• Inside – Select this option if you want to create metal route blockages (shielding) around the boundary inside the plan groups or soft macros.
• Outside – Select this option if you want to create metal route blockages (shielding) around the boundary outside the plan groups or soft macros.
• Both – Select this option if you want to create metal route blockages (shielding) around the boundary both inside and outside the plan groups or soft macros. This is the default.
4. Specify a shielding width.
Click in the Width box and a enter a value. The shielding width is determined by multiplying the layer pitch with the value you specify. The default multiplier is 3.0. You can also specify a value in microns.



5. Select the metal layers.

Specify the metal layer or layers on which the rectangles (shielding) will be created. If no layers are specified, the command creates shielding on each metal layer and uses the layer's preferred direction to determine on which sides shielding is created.

For example, if metal 2's preferred direction is horizontal, then metal 2's rectangle blockages are created on the top and bottom sides of the plan group and/or soft macro.

6. Specify the sides on which to create the shielding.

Select the sides (Left, Right, Top, or Bottom) on which to create the shielding rectangles. If you do not specify a side, shielding is created on all sides of the plan groups or soft macros.

7. Click OK or Apply.

Removing Module Block Shielding

You can remove the signal shielding (route blockages) created by modular block shielding. Choose Floorplan > Delete Floorplan Objects > Delete Module Block Shielding. The Delete Module Block Shielding dialog box appears.

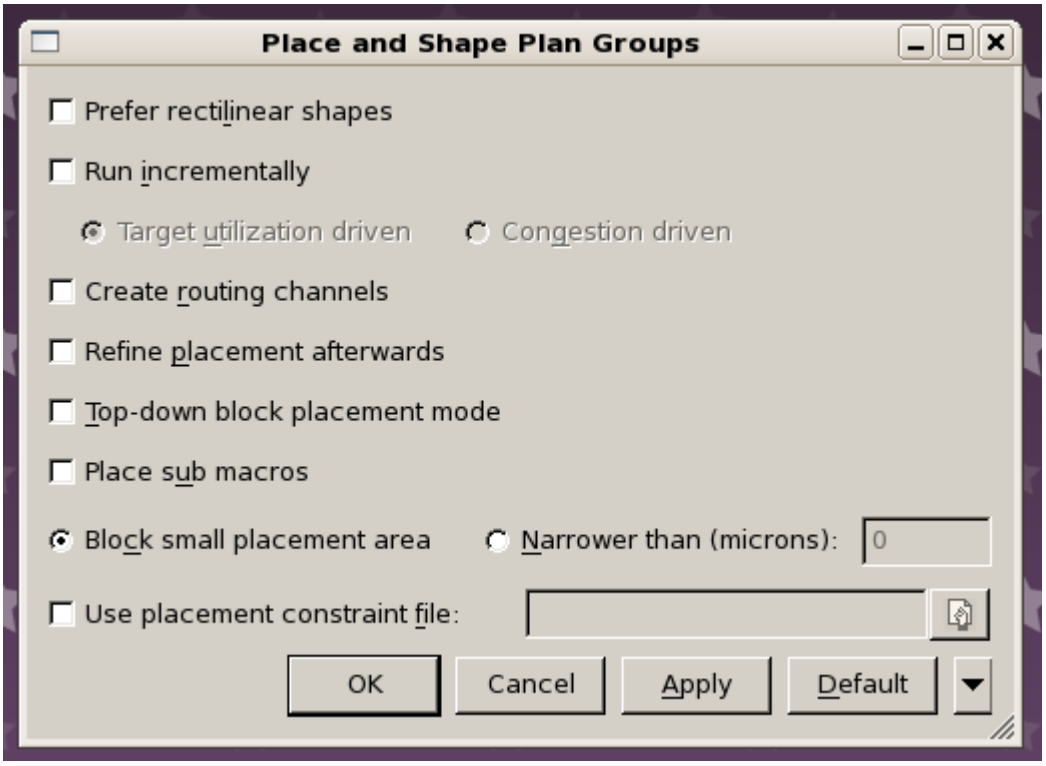
Alternatively, you can use the `remove_fp_block_shielding` command.

You can remove shielding rectangles from all plan groups and soft macros or from specified plan groups or soft macros. You can also select the sides as well as the metal layers on which shielding should be removed.

Automatically Placing and Shaping Objects in a Design Core

Plan groups are automatically shaped, sized, and placed inside the core area based on the distribution of cells resulting from the initial virtual flat placement. Blocks (plan groups,voltage areas, and soft macros) marked as fixed (they have the `is_fixed` property set to `true`) remain fixed; the other blocks, whether or not they are inside the core, are subject to being moved or reshaped.
Note:
It is assumed that an initial virtual flat placement has been run on the design.

To automatically place and shape objects in the design core,
1. Choose Placement > Place and Shape Plan Groups.
The Place and Shape Plan Groups dialog box appears.
Alternatively, you can use the `shape_fp_blocks` command.



2. Select the “Place and Shape plan groups” option. The default is on.

3. Select the remaining options as needed.
• Prefer rectilinear shapes – If you do not select this option, the `shape_fp_blocks` command creates only rectangular plan groups or soft macros boundaries unless they would overlap fixed objects, in which case the shape might need to be rectilinear if it needs to be cut out around fixed objects.
If you select this option, the virtual flat placer tries to put the plan group boundary around the set of preplaced cells associated with that plan group. However, in cases where you have one small block and one large block in your design, the small block is placed by one of the corners of the large block, creating a rectilinear L-shaped boundary. L-shaped plan groups can be created even if no fixed objects are encountered on the floorplan.The default is off.
• Run Incrementally – When you select this option, the floorplan is adjusted to meet the new plan group or voltage area target utilizations or to reduce the placement

congestion in channels between blocks or inside the blocks themselves. The input is a legal, nonoverlapping floorplan and a congestion map if you select the congestion driven mode. The `shape_fp_blocks` command incrementally resizes and reshapes the blocks (plan groups, voltage areas, and soft macros) in the design, changing the floorplan as little as possible, so that the estimates based on the congestion map are satisfied.

The run incrementally option runs in two modes: target utilization driven and congestion driven.

• Create routing channels – If you select this option, routing channels are created between plan groups, black boxes, and soft macros. The channel widths are based on pin counts with nonperpendicular connections to objects on the associated object edge. The default is off.
• Refine placement afterwards – If you select this option, the placement is refined after the placement and shaping of all unfixed plan groups, soft macros, and black boxes. (This is the equivalent of running the `create_fp_placement` command without any arguments.) The default is off.
4. Click OK or Apply.

Controlling the Placement and Shaping of Objects

You can use the `set_fp_shaping_strategy` command, to control how the `shape_fp_blocks` command places and shapes objects in the design core.

Block Level Pin Location Assignment

- **place_fp_pins –block_level**
  - Assigns pin locations based on current cell placement
  - Pins are placed on wire track

- Uses pin constraints, pin blockages and pin guides

Pin Placement Control

Use pin constraints to set:

- TDF constraints on/off
- Allowed layers, pin spacing, corner spacing keepout and keep buses together with ordering

Define pin blockages to keep pins from placing into specified area:

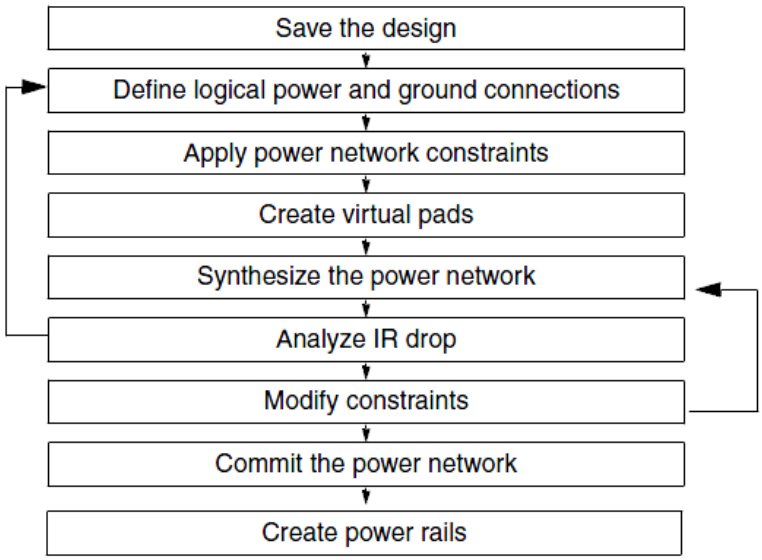
```
create_placement_blockage –bbox {140 20 160 50} \
    –type pin
```

Define pin guides to force some pins to a specified area:

```
create_pin_guide –bbox {170 20 180 50} \
    [get_ports Addr*]
```

Power Network Synthesis (PNS)

Figure 8-1 Power Network Synthesis for Single-Voltage Designs

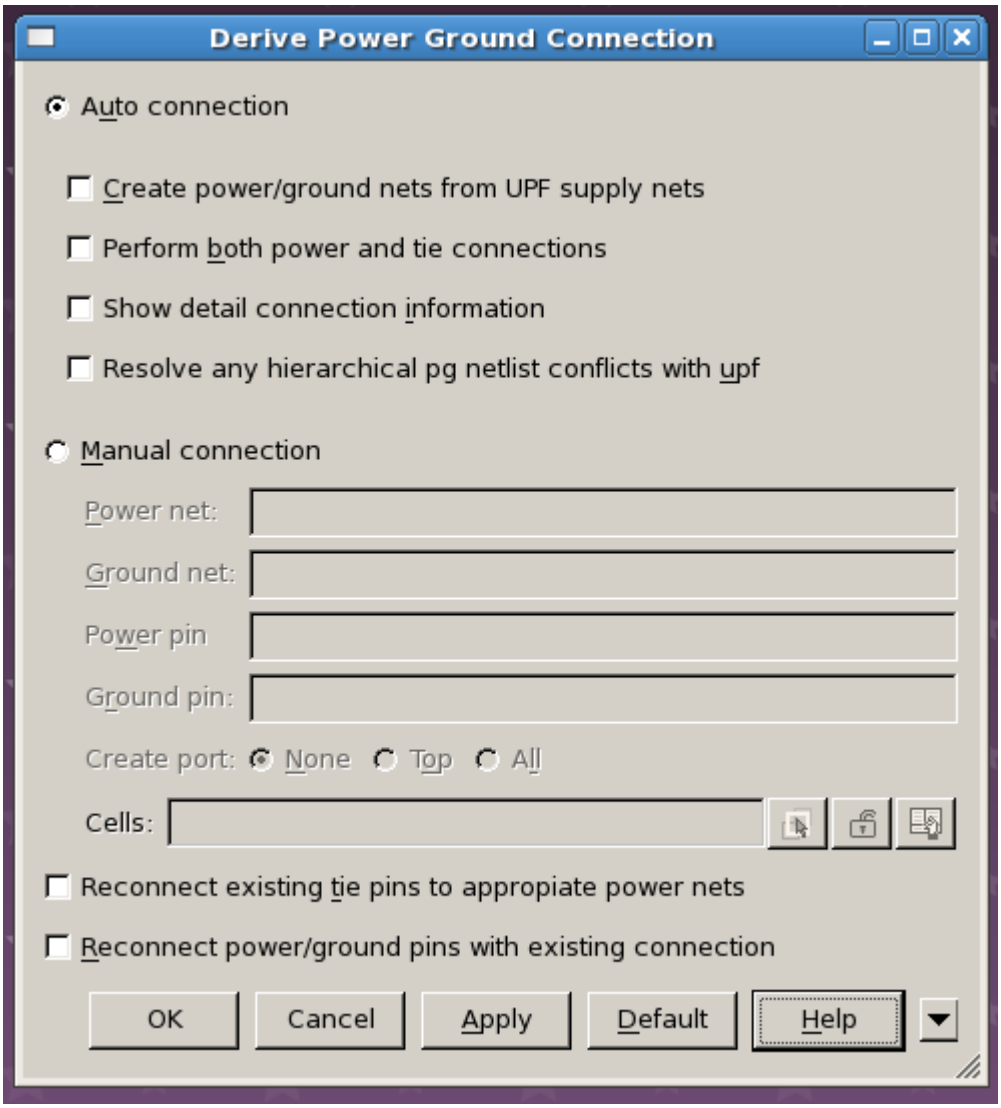


- **Constraint based PNS speeds up a normally long, tedious and iterative task**
  - Creates macro power rings
  - Creates the power grid
- **PNS automates:**
  - Power topology definition
  - Calculation of the width and the number of power straps to meet IR constraints
  - Detailed P/G connections and via placement

Connect P/G Nets before Running PNS/PNA

To perform automatic power and ground and tie connections,
1. Choose Preroute > Derive PG Connection.

The Derive Power Ground Connection dialog box appears.



2. Set the options, depending on your requirements.

• Auto connection – Performs automatic power and ground connections for the power and ground pins of cells in the design as well as direct rail-tie connections to the power and ground nets. This is the default.

• Reconnect power/ground pins with existing connections – Uses existing connections to reconnect all power and ground pins in the design.

• Create power/ground nets from UPF supply nets – Creates the power and ground nets based on power domains defined in the multivoltage design and from UPF supply nets.

• Perform both power and tie connections – Performs direct rail tie-off connections in the design. For multivoltage designs with power domains, the tie pins are connected to the proper power nets consistent with their cell instances' power connections. For a single voltage design, the power and ground nets specified in the "Power net" and "Ground net" text boxes are used as tie-off nets.

• Manual connections – Performs manual power and ground connections for designs with no power domain information.

Enter the names of the power and ground nets to use for the power and tie-high connections and the ground and tie-low connections.

Enter the names of the power and ground pins on cells to connect to the specified power and ground nets.

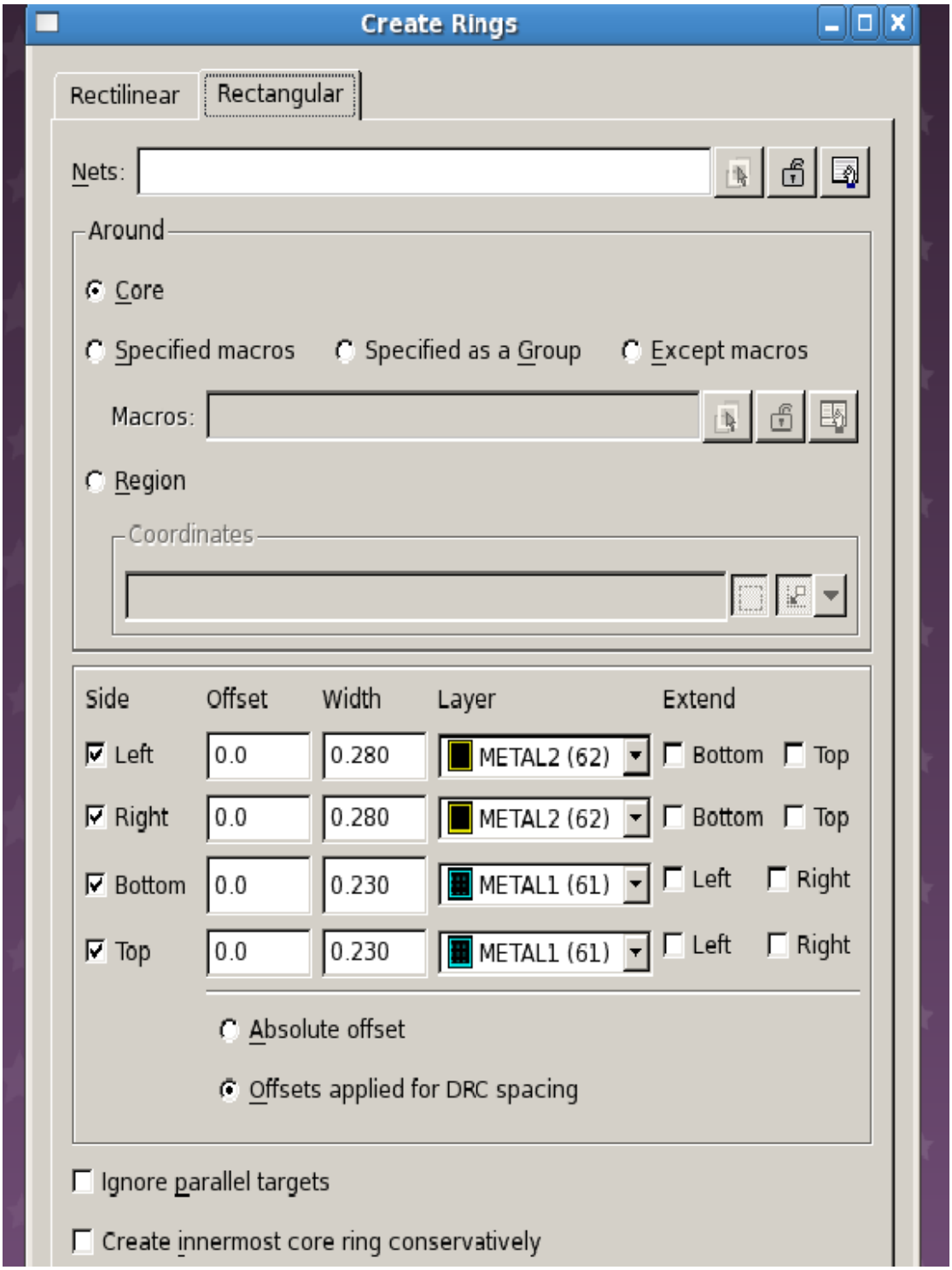
• Create port – Creates I/O ports for the specified power and ground nets. Select "Top"

to create top-level power and ground ports only. Select “All” to create ports on the child cells (soft macros and black boxes). The default is “None” (no ports are created).
• Cells – Enter a list of cells to connect to the specified power and ground nets.
The `derive_pg_connection` command will not touch any power and ground tie-off pins that are already connected to a PG network in the design; it only connects unconnected PG pins and maps tie-off nets to the real PG network, if you specify the `-tie` option.
Note:
Make sure the PG network already exists in the design before you run the `derive_pg_connection -tie` command. If you run the `derive_pg_connection -tie` command before the PG network is created, it can cause unexpected results for the PG tie-off connections in the design flow.

pins and maps tie-off nets to the real PG network, if you specify the `-tie` option.

Adding Power and Ground Rings

After you do floorplanning, you need to add power and ground rings.
To add power and ground rings,
1. Choose Preroute > Create Rectangular Rings.
The Create Rectangular Rings dialog box appears.
Alternatively, you can use the `create_rectangular_rings` command.



2. Specify the power ring characteristics for both power and ground.

• Nets – Enter the names of the nets for which you want to specify rings. To specify two nets, separate the names with commas.

• Side – Select the sides of the ring that you want to create. Select all sides of the ring to create a rectangular ring.

• Width – Enter the widths (left (L), right (R), bottom (B), and top (T)) of the respective sections of the ring.

• Layer – Enter the metal layer (left (L), right (R), bottom (B), and top (T)) to be used for the respective sections of the ring.

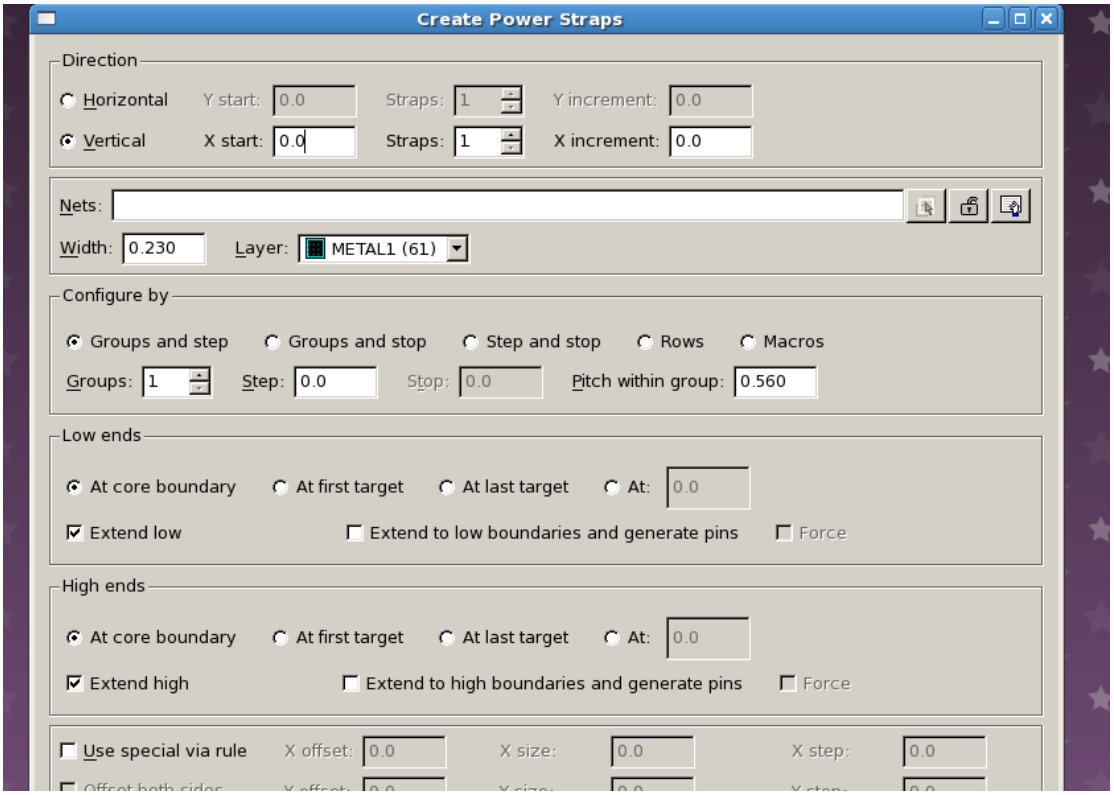
• Extend – You can extend any ring segment to the cell boundary or to the first target on the same net. To do so, enable the desired extension options, as defined in the following table.

• Create innermost core ring conservatively
Select this option if you want the innermost core ring placed far enough away from the core boundary to allow cell placement without design rule checking violations. You might want to use this option when you are creating core rings before cell placement. The tool scans all the standard cells to determine the maximum distance any cell can extend beyond the core boundary after placement. That maximum distance is placed between the innermost core ring and the core boundary.

• Ignore parallel targets
By default, each ring segment connects all passing targets. Select this option to skip those targets that are parallel to the segment.
• Extend for multiple connections
Select this option to extend ring segments to reach more targets on the same net. The extension continues until either there are no more targets in the respective directions, or the next target exceeds the spacing threshold set in the “for gap less than” box.
In the “for gap less than” text box, enter the space threshold for multiple connections. The default is 0.0.
• Use advanced via rules
Select this option to use the advanced via rules that have been previously set by the `set_preroute_advanced_via_rule` command. You can also Choose Preroute > Set Preroute Advanced Via Rule. For more information, see “Setting Preroute Advanced Via Rules” on page 8-33.
• Around
Select the area for which you want to specify rings.
3. Click OK.

Adding Power and Ground Straps

After you add the power and ground rings, you need to add the power and ground straps.The straps are automatically connected to the closest power and ground ring at or beyond, both ends of the straps.
To add power and ground straps,
1. Choose Preroute > Create Power Straps.
The Create Power Straps dialog box appears.
Alternatively, you can use the `create_power_straps` command.



2. Select the direction to specify that the strap starting position be horizontal or vertical. The default is horizontal.
3. Enter the names of the nets for which you want to specify straps. To specify two or more nets, separate the names with commas. The number of net names specified determines the number of straps in a group.
Next, enter the width you want for the straps, followed by the metal layer on which to create the straps.

Prerouting Standard Cells

You can connect power and ground pins in standard cells to the straps and rings of the power mesh and connect power and ground rails in the standard cells. To make sure the global router can recognize the routing obstruction, preroute the standard cells before performing global routing.

If the top cell does not have pads, you can use the “Extend to boundaries and generate pins” option in the Preroute Standard Cells dialog box. This option creates a power and ground extension wire to the cell boundary and generates a power and ground pin if the following conditions are met:
• The power and ground pin is not already connected to power and ground in the direction in which IC Compiler would create the extension wire.
• Creating the power and ground extension wire does not cause a design rule violation. Pins are marked as fixed to prevent them from being moved by place and route operations.

To preroute standard cells,
1. Choose Preroute > Preroute Standard Cells.
The Preroute Standard Cells dialog box appears.
Alternatively, you can use the `preroute_standard_cells` command.
2. Click the Connection Mode tab.
• Select one of the following options:

Rail – Connects standard cells by rails. This is the default.
Tie – Connects standard cells by tying all the power and ground pins to nearby targets.
If you select this option, enter the maximum routing width. The default is 0.
Net – Connects pins of standard cells that are included with the specified net to nearby rings or straps.
If you choose the Net option, you can also specify a maximum fanout. This is the maximum number of pins that are connected to the ground rings or straps. The default is 10.
You can also set the routing layer and width for both vertical and horizontal segments.
3. Click the Routing Options tab.
• Connect pins only on layer number – Select this option to connect pins on a specified metal layer. By default, any layer could be used.
• Connect – Select one or both of the following:
Vertically – Connects power and ground in the vertical direction.
Horizontally – Connects power and ground in the horizontal direction. This option is selected by default.
• Determine the pin width – You can determine the pin width by the width of the pin at its extreme edges or by the width of the most extended pin.
4. Click the Scope tab.
• Use top cell's bounding box – This option sets the working area to the top cell's bounding box.
Net names – Enter a list of net names in the text box. The net names in the list must be separated by commas.
• Use specified bounding box – You can select this option to specify a user-defined bounding box.
Coordinates – Enter the x- and y-coordinates in the Coordinates field, or select the Rectangle button and draw the rectangle in the layout view; or select the Rectilinear button and draw the rectilinear shape in the layout view.
• Filter Ports by name – Select from the following:
Filter off – Use this option when the selection of pins for routing is not dependent on their names.
Connect matched – Routes only those pins whose names are matched to at least one specified pattern.
Skip matched – Routes all pins except those pins whose names are matched to at least one specified pattern.
• Filter Ports of cells by master name – Select from the following:
Filter off – Use this option when the selection of pins for routing is not dependent on the names of their cell masters.
Connect matched – Routes only those pins whose cell master names are matched to at least one specified pattern.
Skip matched – Routes all pins except those pins whose cell master names are matched to at least one specified pattern.
• Filter Ports of cells by instance name – Select from the following:
Filter off – Use this option when the selection of pins for routing is not dependent on the names of their cell instances.
Connect matched – Routes only those pins whose cell instance names are matched to at least one specified pattern.
Skip matched – Routes all pins except those pins whose cell instance names are matched to at least one specified pattern.

• Filter Ports of cells by voltage area located in– Select from the following:
Filter off – Use this option when the selection of pins for routing is not dependent on the name of the voltage area to which there cell instance is placed.
Connect matched – Routes only those pins whose cell instances are placed in a voltage area whose name is matched to at least one specified pattern.
Skip matched – Routes all pins except those pins whose cell instances are placed in a voltage area whose name is matched to at least one specified pattern.

6 Click the DRC button to open the Set Preroute DRC Options dialog box.

7 Click OK or Apply.

Setting Preroute Design Rule Checking Options

You can set preroute design rule checking (DRC) options to change the internal rules for design rule checking for the power and ground commands.
To change the internal rules for design rule checking,
1. Choose Preroute > Set Preroute DRC Options.
The Set Preroute DRC Options dialog box appears.
Alternatively, you can use the `set_preroute_drc_strategy` command.
2. Set the options, depending on your requirements.
• Spacing – Specify the mode for design rule checking.
Select “Radial” if your spacing rules are defined radially. A radial rule checks for spacing violations within a circular area. This is the default.
3. Click OK or Default.

Performing Power Network Synthesis

As the design process moves toward smaller geometries, issues related to power and signal integrity, such as power grid generation, voltage (IR) drop, and electromigration, have become more significant and complex. In addition, this complex technology lengthens the turnaround time needed to identify and fix power and signal integrity problems.
Before committing to a detailed power plan structure, you can perform power network synthesis to get a rough power plan estimation by experimenting with different user-defined power topologies to create power and ground meshes with rectangular or rectilinear power rings.
By performing power network synthesis, you can preview an early power plan that reduces the chances of encountering electromigration and voltage drop problems later in the detailed power routing.

Specifying Global Constraints for Power Planning

To specify global constraints,
1. Choose Preroute > Power Network Constraints > Global Constraints.
The Global Power Network Constraints dialog box appears.
Alternatively, you can use the `set_fp_rail_voltage_area_constraints` command.

Set the options, depending on your requirements.

- Remove floating segments – If you enable this option, power network synthesis removes any dangling wire segments cut by hard macros or blockages. This reduces the need for power plan editing. Disable this option if you do not want floating segments to be removed. The default is enabled.
- Force same width sizing – If you enable this option, power network synthesis creates wires that are the same width for both power and ground nets. If you deselect this option, power network synthesis creates wires with different widths for the power and ground nets. The default is enabled.
- Keep rings outside of core area – If you enable this option, power network synthesis creates the core ring outside of the core area. The default is disabled.
- No routing over plan groups – If you enable this option, power network synthesis will not route over plan groups, despite the absence of blockages. The default is disabled.
- Generate stacked vias – If you enable this option, power network synthesis uses stacked vias. If this option is disabled, power network synthesis drops vias only on adjacent layers. The default is enabled.
- Optimize tracks usage – If you enable this option, power network synthesis sizes wires so that it leaves enough space for the power and ground wire's adjacent tracks to be used for signal routes. The default is disabled.
- No routing over hard macros – If you enable this option, power network synthesis does not route over hard macros, despite the absence of blockages. The default is disabled.
- No routing over soft macros – If you enable this option, power network synthesis does not route over soft macros, despite the absence of blockages. The default is disabled.
- Ignore all the blockages – If you enable this option, power network synthesis creates power wires despite hard macro blockages. By default, power network synthesis honors hard macro blockages in order to create a DRC-clean power plan.

2. Click OK or Apply.

Specifying Core Ring and Strap Constraints for Power Planning

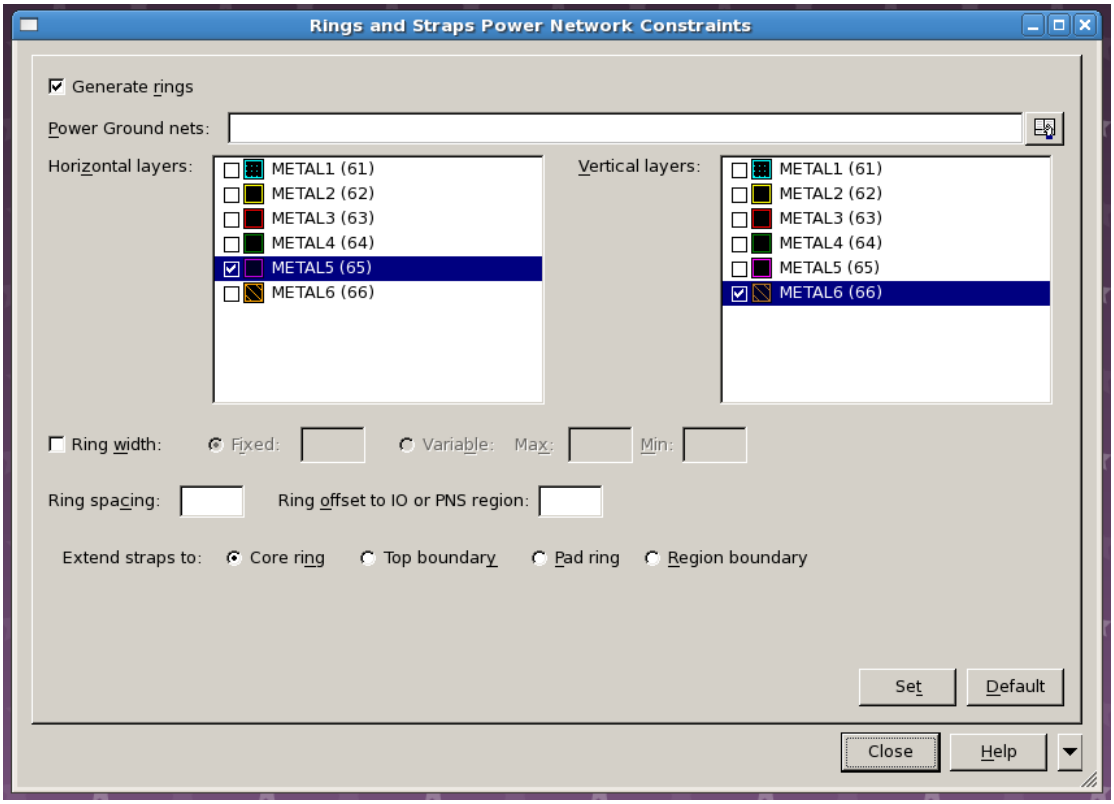
To specify ring and strap constraints,

1. Choose Preroute > Power Network Constraints > Ring and Strap Constraints.

The Rings and Straps Power Network Constraints dialog box appears.

Alternatively, you can use the `set_fp_rail_voltage_area_constraints` command.

Set the options, depending on your requirements.



• Horizontal layer and Vertical layer

You can specify the horizontal and vertical layers for the power ring. The default horizontal layer is the highest horizontal layer in the database; the default vertical layer is the highest vertical layer in the database.

• Ring width and Ring offset

To meet the IR drop requirements, the ring width is automatically determined.

Ring width – Enter a fixed width or a range of widths for the power ring.

Ring spacing – Specify the distance between the power and ground rings in the voltage area. (Optional) The units are measured in microns.

Ring offset to IO or PNS region – Specify the offset from the I/O pads to the power rings. The units are measured in microns.

• Extend strap

If you choose to generate a power plan without a core ring, you need to instruct power network synthesis on how to connect (extend) the generated horizontal and vertical power straps. There are three ways to do this:

Core ring – When this option is enabled (the default), power network synthesis extends the power straps to the existing core ring.

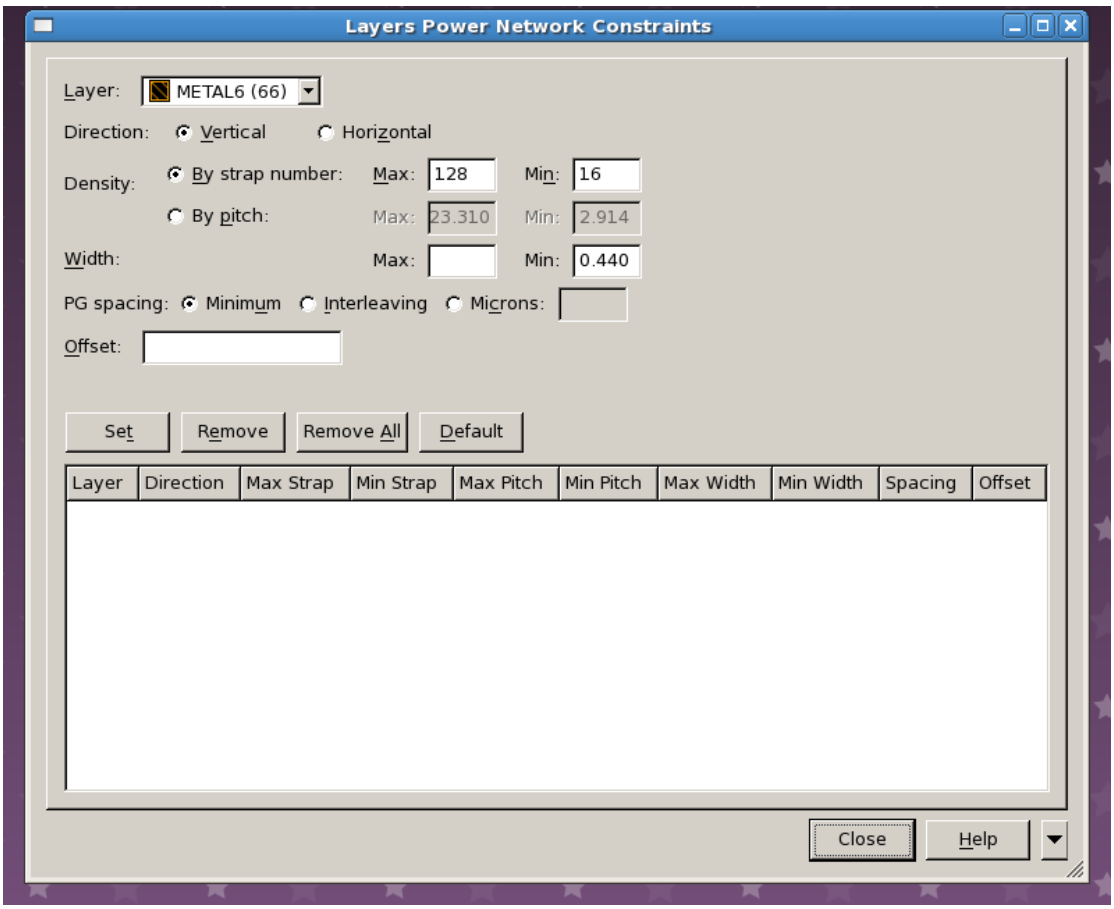
Top boundary – When this option is enabled, power network synthesis extends the power straps to the top-level cell boundary and creates power pins.

Pad ring – When this option is enabled, power network synthesis extends power straps to an existing pad ring. When this option is disabled, the straps stop at the pad ring boundary.

2. Click OK or Apply.

Specifying Layer Constraints for Power Planning

To specify layer constraints,
1. Choose Preroute > Power Network Constraints > Layer Constraints.
The Layer Power Network Constraints dialog box appears.

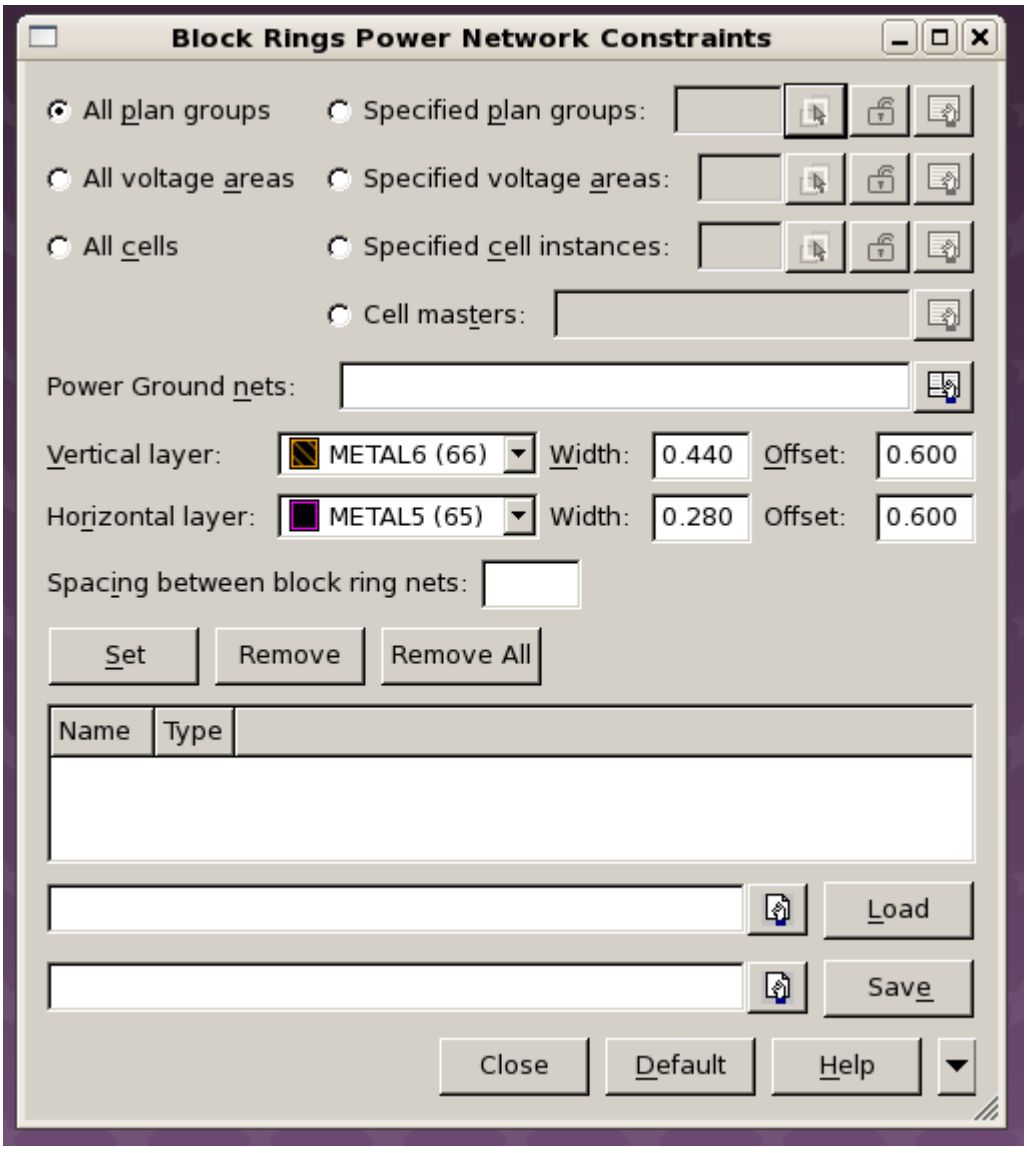


Alternatively, you can use the `set_fp_rail_voltage_area_constraints` command.
Set the options, depending on your requirements.
• Layer
Specify the metal layer on which to create the power grid. The default is the highest metal layer.
• Direction
Specify the direction layer (vertical or horizontal) for the power and ground wires. The default is vertical.
• Density and Width
You can control the density by specifying the number of straps or the pitch. Enter the maximum and minimum number of straps or the pitch in the text boxes. The default is 128 and 16, respectively, for the straps.
You can control the width of the power wires by entering the maximum and minimum width in the appropriate text boxes. The default is the maximum and minimum width defined in the technology file.
Note:
If you specify the maximum and minimum number of straps or the pitch and the same maximum and minimum width, power network synthesis bypasses the optimization engine and generates a custom power mesh as specified.
• Offset – Enter the distance from the voltage boundary to the left-most and bottom-most power straps. The units are measured in microns. By default, the offset

distance is calculated based on the number, width, and pitch of the straps.
2. Click the Set button to set the constraints for the currently selected layer. Click Remove to remove the settings for the currently selected layer or Remove All to remove all of them. Click Default to return the dialog box to its default settings. When you are done setting the layer constraints, click Close.

Specifying Constraints for Block Ring Synthesis

To create power and ground rings,
1. Choose Preroute > Power Network Constraints > Block Ring Constraints.
The Block Rings Power Network Constraints dialog box appears.
Alternatively, you can use the `set_fp_block_ring_constraints` command.



2. Set the options, depending on your requirements.
• You can select blocks by plan group cell instances, voltage areas, or cell masters. The power and ground rings are created around the specified cell instances, plan groups, or voltage areas.
• Power Ground nets
Enter the net names for the block rings. You can enter multiple net names to create multiple block rings. The first net will be the innermost block ring. For example, to create four rings, enter
`"VDD VSS VDD VSS"`
• Vertical layer

Select the vertical layers in the block ring. The default is the highest layer with the preferred vertical direction. Only one layer is allowed for each block ring constraint.
Specify the width of the vertical wires. The default value is the minimum width in the technology file.
Specify the vertical offset from the block boundary to the block ring. The default value is the minimum spacing in the technology file.
• Horizontal layer
Select the horizontal layers in the block ring. The default is the highest layer with the preferred horizontal direction. Only one layer is allowed for each block ring constraint.
Specify the width of the horizontal wires. The default value is the minimum width in the technology file.
Specify the horizontal offset from the block boundary to the block ring. The default value is the minimum spacing in the technology file.
• Spacing between block ring nets
Enter a value to specify the spacing between the block rings. The default is the minimum spacing defined in the technology file.
3. Click the Set button to set the constraints for the currently selected layer. Click Remove to remove the settings for the currently selected layer or Remove All to remove all of them. Click Default to return the dialog box to its default settings. When you are done setting the block ring constraints, click Close.

Why Use Power Network Synthesis?

Power network synthesis offers advanced power planning technology and helps solve signal integrity problems without lengthy and tedious iterations. By performing power network synthesis, you can preview an early power plan and thereby reduce the chance of encountering electromigration and voltage drop problems later in the detailed power routing. Power network analysis also helps you avoid overdesigning the power plan early in the design cycle because you know the electromigration and voltage drop effects in the power planning stage.
Power network synthesis provides these benefits:
• Produces automated IR-aware power planning
Power network synthesis allows you to enter the IR drop as a constraint, along with trunk width and mesh density constraints, so that you can quickly create a power mesh in multimillion-instance designs.
• Manages an IR drop strategy before power plan creation
The preview feature in power network synthesis gives you early insight into IR drop's impact on the power structure before a single power wire is created. If your virtual power mesh, generated by power network synthesis, does not meet with your satisfaction, you can revise the constraints in power network synthesis and then regenerate the power plan. Because power planning creation in power network synthesis takes only few minutes (even on a multimillion-instance design), you can easily experiment with different power mesh topologies.
• Develops a production-quality power plan from preroute technology
Power network synthesis is built on top of preroute technology. It creates preroute commands and internally executes them when you use the `commit_fp_rail` command to commit the power plan. The power plan meets your specified via generation rules and fat wire rules in addition to being DRC-clean.
• Saves routing resources
The power network synthesis optimization engine helps you find the optimal power routing structure to meet IR drop specifications. Power network synthesis generates a

power mesh that meets IR drop, and at the same time, it minimizes the power routing resources. When power network synthesis finishes power mesh generation, it reports the routing resources used. This information can be valuable for congestion management later in the design process.

• Drives other power planning tools

Power planning is a very customized process because your design style and power mesh topology vary from one process to another. Power network synthesis helps you generate a *starting point* power plan for reference. You can use the power plan generated by the power network synthesis as the final power plan and continue the implementation, or you can use the information provided by power network synthesis to drive customized or third-party power planning tools. An example of this might be a 25 x 25 power mesh with a 8.4-micron trunk width and a 10 percent metal5 and 12.5 percent metal6 routing resource to meet a 100 millivolt IR drop.

• Manages congestion and utilization prior to floorplanning

You can use power network synthesis to generate a power plan before instances are placed but after the die is initialized. Enter the power budget for the entire design, and use power network synthesis to estimate the power structure needed to meet the IR drop requirements. Then use the routing resources information to budget routing for signal nets. If the power structure uses too much routing resources, you can reinitialize the design with a smaller utilization ratio to meet signal routing resource needs.

• Generates a custom power mesh

You can use power network synthesis to create a power mesh (assuming that you do not want to specify IR drop constraints). To do this, set the maximum and minimum number of constraints and the width to the same value. When this runs, power network synthesis bypasses the optimization engine and generates a power mesh you have specified.

• Produces a rectilinear power mesh

One of the biggest challenges in power planning is rectilinear mesh construction. Power network synthesis provides you with one-step rectilinear power mesh generation that meets IR drop requirements, is DRC-clean, and is perfectly snapped to the power plan grid.

• Reduces the need for complex power plan editing

Power planning can sometimes be labor-intensive, because it involves more than routing a wire from one point to the other. If your power planning tools do not generate output with the concept of a power structure, you might be required to spend more time editing. Power network synthesis creates output with the concept of a power structure and avoids physical restrictions when creating the power structure. For example, rather than stopping a power trunk at a blockage, power network synthesis cuts the trunk to accommodate the blockage and snaps the line (optional) to the closest intersection. Also, you can specify that dangling wire segments be removed, which reduces the need for power plan editing.

• Prepares a power grid prototype

Because power network synthesis makes power grid creation less complex, you can use it as a power grid prototyping tool. Used in this way, you can experiment with different power mesh designs and optimize the power mesh to meet your specific design needs.

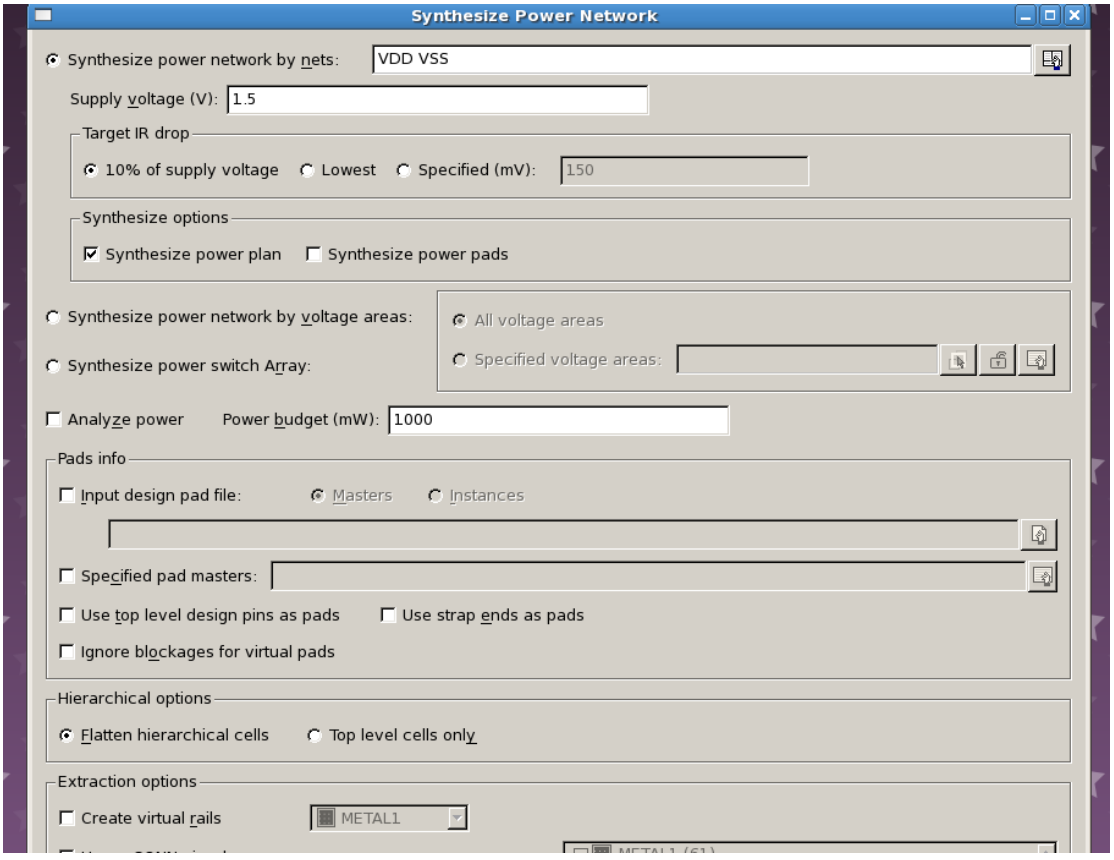
• Replays power plan creation

While power network synthesis creates the power plan, it simultaneously generates a command script containing the `create_power_straps` command that you can view and edit. If there are minor incremental floorplan changes, you can also use this script to revise the power plan. Customizing the script for your particular power planning requirements is also possible with this process.

• Detects placement problems
Power network synthesis lets you quickly create a power plan and simultaneously lets you see any placement problems that would block power trunk traffic. (For example, a hard macro with blockage on metal5 is placed in the extending power routing area.) By using power network synthesis power planning, you can budget and make tradeoff decisions based on power routing resources and optimal placement locations. After you make the placement adjustment, you can replay the power plan creation script generated by power network synthesis or simply rerun power network synthesis.
• Creates region-based power mesh generation for multivoltage support
You can use power network synthesis to easily create incremental regional power mesh structures by specifying the mesh regions one at a time.

Opening the Power Network Synthesis User Interface

The preferred time to run power network synthesis is right after virtual flat placement. At that point, the cells are closer to the final placement, so power network synthesis can do a better job of estimating power distribution. You can also run power network synthesis before the placement is finalized and after the core is initialized.
To open power network synthesis,
1. Choose Preroute > Synthesize Power Network.
The Synthesize Power Network dialog box appears.
Alternatively, you can use the `synthesize_fp_rail` command.



This is where you enter all the information for your power plan. You can get a rough power plan estimation by experimenting with different user-defined power topologies to create rectangular or rectilinear power rings before committing to a detailed power plan structure.

You can synthesize a power network for a single voltage design, for a multivoltage design(see

“Performing Power Network Synthesis On Multivoltage Designs” on page 8-51), or for a power switch array based on user-specified constraints.
2. Set the options, depending on your requirements.
• In the “Synthesis power network by net” field, specify the names of the power and ground nets on which to perform the power network analysis. Both nets are processed simultaneously.
• Perform constraint-driven power network synthesis by specifying a total power budget for the analyzed design.
The power budget is divided among the instances according to their size. The power unit is milliwatts. The default is 1,000.
For hard macros or standard cells, the power budget is computed based on the percentage of the total area. For a hierarchical block, the power budget is computed based on the sum of all the cells and blocks inside it. If there are no logical connections to the analyzed net, the allocation is zero.
For example, if the power budget for the entire design is 500 milliwatts, the total area of the top-level blocks is 500,000 square microns, and one of the blocks has an area of 25,000 square microns, power network analysis allocates the power for that 25,000-square-micron block as follows:
500mW * (25,000/50,000) = 25mW
Note:
You can also obtain the power calculation source from a PWR view (for hard macros) or from Power Compiler and PrimePower report files.
• Specify a voltage supply by entering the voltage of the net you want synthesized. The unit is volts. The default is 1.5.
• Specify the target IR drop value. The default is 10 percent of the power pad voltage supply. If the voltage supply is 1.5 volts, the default drop would be 150 millivolts.
Select the Lowest option if you want to direct the synthesize engine to synthesize a power plan or a pad that would give the lowest-possible IR drop value based on user-defined constraints.
• Specify the “synthesize power plan” option to perform normal voltage power network synthesis. This is the default behavior. You can specify this option together with the “synthesize power pads” option to perform simultaneous power network and power pad synthesis.
• Specify the “synthesize power network by voltage area” option to perform multivoltage power network synthesis on voltage areas that have been specified by using the `set_fp_rail -voltage_area constraints` command. For more information, see “Performing Power Network Synthesis On Multivoltage Designs” on page 8-51.

Select the “All voltage areas” option to synthesize power networks in all voltage areas that have been specified by using the `set_fp_rail -voltage_area constraints` command. If you want to specify a voltage areas in which to perform multivoltage power network analysis or power switch synthesis, select the “Specified voltage areas” option, and enter the name of the voltage area.

• Specify the “synthesize power switch array” option to perform power switch synthesis on the specified powered-down voltage area
• Specify the power and ground pad information. Pad information is critical for power network synthesis. By default, power network synthesis assumes that any power pad with a logical connection to the power net is the power source for the power net.

However, this assumption might not hold where pads have a logical power net connection that is used to power the pad signal driver, such as a power pad ring. In such a case you might want to specify an exact power pad instance or master.
The following four power and ground pad definition types are supported:
Type 1: Specifying the input design pad file name – Enter the name of the pad file.
If the Instances option is active, you can enter a power and ground pad definition file in instance-based format.
The format is
instance_name net_name
(The *net_name* is optional.) For example,
`VDD1 VDD`

If the Masters option is active, you can enter a power and ground pad definition file in master-based format. This is the default.
The format is
master_name net_name
(The *net_name* is optional.) For example,
`VDD.FRAM VDD`
Type 2: Specifying pad masters – Enter the name of the pad master in the text box.
Type 3: Use top-level pins as pads – Select this option if you want pins to be regarded as pads during block-level simulation.
Type 4: Use strap ends as pads – Use this option during block-level power planning when top-level power and ground pins are missing. Power network synthesis will use the ends of straps at the design boundary or core ring as power and ground pads. Because the pads are defined, you do not need to define virtual pads and additional pad information.
By default, power network synthesis places the power and ground pads at four sides of the strap end: north, south, east, and west. If you want to control which direction the sides of the strap ends are used as power and ground pads, set the following environment variable:
`PNS_BOUNDARY_PAD_SIDE`
Power network synthesis creates a "`straps_end.` *netname*`.vpad`" virtual pad file to store the power pad information defined by this option.
Specify a hierarchical option.
Flatten Hierarchical Cells (the default) – The power network synthesis engine analyzes the design as if it were flat.
Top Level Cells Only – The power network synthesis engine ignores cells inside soft macros but considers all the power and ground net wires (and vias) in the design.
• Specify the extraction options.
• Create virtual rails – During the floorplanning stages, metal1 straps for the standard cell pin connections are usually not available. Power network synthesis uses this option to create virtual pseudo straps. By default, metal1 is used to generate the pseudo straps for the standard cell pin connections. Power network synthesis uses the virtual straps for analyze the IR drop and electromigration to predict what the effects might be when the metal1 straps are available.
Horizontal pseudostraps are created with specified layers for standard cell pin connections, based on the row information in the database.
Ignore CONN view layers – Reduces memory usage in power network analysis and power network synthesis and speeds up the IR simulation.

• Specify power information – There are four ways to obtain the power calculation source required to analyze the power:
Power consumption file – To manually set power on a cell instance or cell master through a file, enter the name of the power consumption file. This file specifies the amount of power consumed by each instance. Select the file format: Default/ Power Compiler, or Prime Power/PrimeTime PX.
• Specify an output directory.
The analysis results are output to the default directory, pna_output. The specified directory stores the IR drop and electromigration results with the file name *design_name*. *net_name*.pw_hl.pna and the power allocation results with the file name *design_name*. *net_name*.power.
The IR drop and electromigration results are stored in the analysis file:
./pna_output/ *designname*. *netname*.pw_hl.pna
The power allocation results of the computed cells and blocks are stored in the analysis file:
./pna_output/ *designname*. *netname*.power
Note:
Because these files can be quite large, make sure you have enough disk space in your ./pna_output directory
3. Click OK or Apply.

Performing Power Network Stacked Via Removal to Reduce Congestion

As the voltage-level in designs continues to scale down, the noise introduced by voltage IR drop in the power or ground network becomes more critical. These designs might require a denser power network to minimize the voltage drop. However, this can cause the creation of a large number of stacked vias from the top mesh down to the standard cell rails. These stacked vias might result in congestion issues. You can use the `reduce_fp_rail_stacked_via` and `remove_fp_rail_stacked_via` commands to identify and select a set of stacked vias in a complete or partially-built power network for removal to reduce congestion and then to remove these via objects physically from the design.

Selecting Stacked Vias for Removal

Given a complete or partially-built power network, you can use the `reduce_fp_rail_stacked_via` command to identify all removable stacked vias in the power network. A set of critical wires in the least congested area of the power network is identified to ensure that each originally connected standard cell or macro block can receive power from power pads through this set of wires. All stacked vias that are in a congested area and do not contain any critical vias are identified as “feasible” stacked vias. You can select these vias for removal such that the voltage drop constraint is honored and the connectivity of the existing power network is maintained.

Removing the Selected Stacked Vias

Once you have selected a set of power network stacked vias for removal by using the

`reduce_fp_rail_stacked_via` command, you can physically remove the power network stacked vias by using the `remove_fp_rail_stacked_via` command. The command purges the selected via objects from the design based on the stacked via reduction results.

Specifying Global Constraints for Power Planning

To specify global constraints,
1. Choose Preroute > Power Network Constraints > Global Constraints.
The Global Power Network Constraints dialog box appears.
Alternatively, you can use the `set_fp_rail_voltage_area_constraints` command.
Set the options, depending on your requirements.
2. Click OK or Apply.

Specifying Core Ring and Strap Constraints for Power Planning

To specify ring and strap constraints,
1. Choose Preroute > Power Network Constraints > Ring and Strap Constraints.
The Rings and Straps Power Network Constraints dialog box appears.
Alternatively, you can use the `set_fp_rail_voltage_area_constraints` command.
Set the options, depending on your requirements.
2. Click OK or Apply.

Specifying Constraints for Generating Multiple Core Rings or Sandwich Core Rings

If a synthesized core ring is too wide, metal slotting is required. To avoid metal slotting, you can use the options in the Rings and Straps Power Network Constraints dialog box to generate multiple core rings or sandwich core rings. (The term sandwich rings refers to rings on different layers that are superimposed on each other.)
Alternatively, you can use the `set_fp_rail_voltage_area_constraints` command.

Specifying Layer Constraints for Power Planning

You can specify the current layer on which to create the power grid and the direction (vertical or horizontal) for the power and ground wires.
To specify layer constraints,
1. Choose Preroute > Power Network Constraints > Layer Constraints.
The Layer Power Network Constraints dialog box appears.
Alternatively, you can use the `set_fp_rail_voltage_area_constraints` command.
Set the options, depending on your requirements.

2. Click the Set button to set the constraints for the currently selected layer. Click Remove to remove the settings for the currently selected layer or Remove All to remove all of them. Click Default to return the dialog box to its default settings. When you are done setting the layer constraints, click Close.

Specifying Constraints for Block Ring Synthesis

Committing the Power Plan

Once the IR drop map meets the IR drop constraints, you can run the `commit_fp_rail` command to transform the IR drop map into a power plan. The `commit_fp_rail` command calls the `create_power_straps` command to generate the power net wires and vias. Instead of running the `commit_fp_rail` command, you can click the Commit button in the Synthesize Power Network dialog box to actually generate a real power plan with net wires and vias.

Checking the Integrity of the Power Network

After you run power network synthesis you can check the integrity of your power and ground network early in the design planning stage by using the `check_fp_rail` command. Running an integrity check after you commit the power plan is important because it can reveal cases where the power network might not be exactly the same as the synthesized result.The integrity of your power network can be compromised by broken core rings or by floating segments that were left behind. If you have a multivoltage design with power-down voltage areas, the connection of the input pin of the power switch cell to the permanent power and ground straps might fail during the power network synthesis commit process.
The `check_fp_rail` command uses the following syntax:
`check_fp_rail-nets` *nets*`-ring-floating_segment-power_switch_connection`

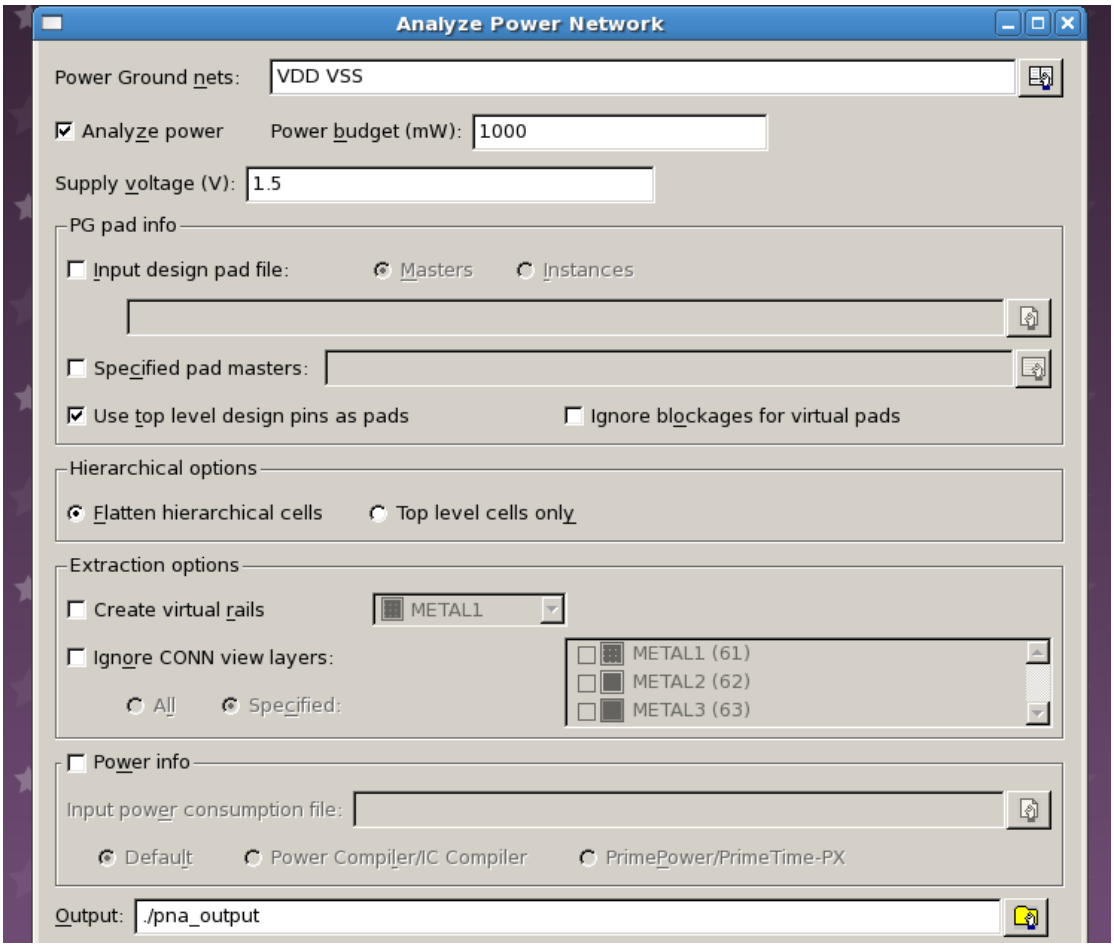
Analyzing the Power Network

You can perform power network analysis to predict IR drop at different floorplan stages on both complete and incomplete power nets in your design.
Power network analysis consists of extraction and analysis of the power or ground rail specified by the given net names. You can analyze IR drop and electromigration effects during initial power grid planning while the floorplan with top-level blocks is still being implemented in the design and the power ports of the standard cells are not yet connected. Then later in the design cycle, when the details of top-level blocks have gone through detailed cell placement and power rail routing and connections have been established, you can verify whether power nets are of sufficient size and quantity.
For a selected hierarchy, power network analysis provides static rail analysis on power planned and on routed or unrouted designs as well as what-if analysis on virtual topological and voltage source changes.

Performing Power Network Analysis

To perform power network analysis,
1. Choose Preroute > Analyze Power Network.
The Analyze Power Network dialog box appears.
Alternatively, you can use the `analyze_fp_rail` command.

Set the options, depending on your requirements.
Power network analysis requires you to enter a power budget for the whole design.



• Specify the names of the power and ground nets on which to perform the power network analysis. All nets are processed simultaneously.
• Specify a total power budget for the analyzed design.
The power budget is divided among the instances, according to their size. The power unit is milliwatts. The default is 1,000.
For hard macros or standard cells, the power budget is computed based on the percentage of the total area.
For a hierarchical block, the power budget is computed based on the sum of all the cells and blocks inside it.
Note:
If there are no logical connections to the analyzed net, the allocation is zero.

Example:
If the power budget for the entire design is 500 milliwatts, the total area of the top-level blocks is 500,000 square microns, and one of the blocks has an area of 25,000 square microns, power network analysis allocates the power for that 25,000-square-micron block as follows:
500mW * (25,000/500,000) = 25mW
Note:
You can also obtain the power calculation source from a PWR view (for hard macros) or from Power Compiler and PrimePower report files.
• Specify a voltage supply by entering the voltage of the net you want analyzed. The unit is volts. The default is 1.5.

• Specify the power and ground pad information. The following power and ground pad definition types are supported:
Type 1: Specifying the input design pad file name – Enter the name of the pad file.
If the Instances option is active, you can enter a power and ground pad definition file in instance-based format.
The format is
instance_name net_name
(The *net_name* is optional.) For example,
```
VDD1 VDD
```
If the Masters option is active, you can enter a power and ground pad definition file in master-based format. This is the default.
The format is
master_name net_name
(The *net_name* is optional.) For example,
```
VDD.FRAM VDD
```
Type 2: Specifying pad masters – Enter the name of the pad master in the text box.
Type 3: Using top-level pins as pads – Select this option if you want pins to be regarded as pads during block-level power network synthesis.
• Specify a hierarchical option.
If your design contains macro cells that are placed and routed, you can sequentially search downward into these macro cells and bring all of the power and ground segments up to the top-level resistance network.
Flatten Hierarchical Cells (the default) – The power network analysis engine analyzes the design as if it were flat.
Top Level Cells Only – The power network analysis engine ignores cells inside soft macros but considers all the power and ground net wires (and vias) in the design.
• Specify an extraction option.
Create virtual rails – During the floorplanning stages, metal1 straps for the standard cell pin connections are usually not available. Power network analysis uses this option to create virtual pseudostraps that represent the metal1 straps for the standard cell pin connections. During power planning, it analyzes the IR drop and electromigration to predict what the effects might be when the metal1 straps are available.
Horizontal pseudostraps are created with specified layers for standard cell pin connections, based on the row information in the database.
Ignore CONN view layers – Reduces memory usage in power network analysis and power network synthesis and speeds up the IR simulation.
If you select the All option, all the metal layers in the CONN (connectivity) views are ignored.
If you select the Specified option, click the mask names of the layers to be ignored in the CONN views.
• Specify power information – There are four ways to obtain the power calculation source required for power network analysis:

Viewing the Analysis Results

When power and rail analysis are complete, you can check for the voltage drop and electromigration violations in your design by using the voltage drop map and the electromigration map. You can also detect the power and ground weaknesses that a voltage drop

map cannot by using a resistance map.

You can save the results of voltage drop and electromigration current density values to the database by saving the CEL view that has just been analyzed. Doing so allows you to close the database and retrieve the results later.

Displaying the Voltage Drop Map

Voltage drop violations are displayed in a highlight map, which provides a graphical view of the results.
To display the voltage drop violations in a highlight map,
1. Choose Preroute > Power Network Voltage Drop Map.
The PNA Voltage Drop Map Mode panel appears.
Alternatively, you can use the `load_fp_rail_map` command.
2. Select metal layers for analysis, and enter the maximum and minimum threshold values.
By default, power network analysis automatically sets the upper and lower bounds of the net you specify to the calculated minimum and maximum threshold values.
3. Click Reload.
The Load Voltage Drop Map Data dialog box appears.
4. Select the net for which you want to load the rail analysis results for displaying the voltage drop map.
5. Click OK to load and display the IR drop map in the current layout view.
A voltage drop map divides the core area into a grid of colored boxes. Each box represents the voltage drop levels in that area of the design.The box edges are colored and labeled to show the voltage drop values. Each map color represents a range of voltage drop values called a bin.
The voltage drop map legend also displays the color and the data count. The colored histogram bars on the right side of the legend represent the relative distribution of voltage drop values in the bins.
To view voltage drop data in the map by using the Query tool,
1. Click the Query button on the Map Mode panel to enable the Query tool.
The Query Object panel appears.
2. Click a location in the map for which you want to view the voltage drop data.
The voltage drop data is displayed on the Query Object panel. The object is highlighted in the query color (white) and the information for the object appears on the Query Object panel.

Displaying the Electromigration Map

To display the electromigration violations in a highlight map,
1. Choose Preroute > Power Network Electromigration Map.
The PNA Electromigration Map Mode panel appears.
Alternatively, you can use the `load_fp_rail_map` command.
2. Select metal layers for analysis, and enter the maximum and minimum threshold values.
By default, power network analysis automatically sets the upper and lower bounds of the net you specify to the calculated minimum and maximum threshold values.
3. Click Reload.
The Load Electromigration Map Data dialog box appears.

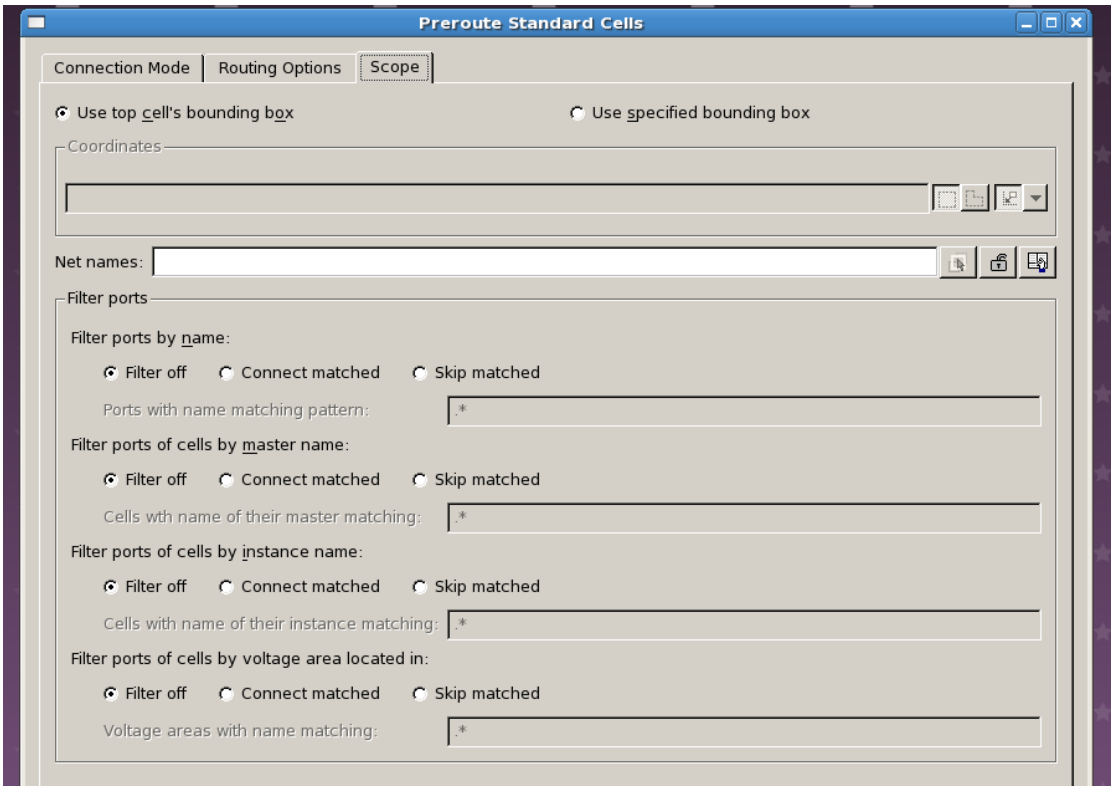
4. Select the net for which you want to load the rail analysis results for displaying the electromigration map.
5. Click OK to load and display the electromigration map in the current layout view.
An electromigration map divides the core area into a grid of colored boxes. Each box represents the electromigration levels in that area of the design.The box edges are colored and labeled to show the electromigration values. Each map color represents a range of electromigration values called a bin.
The electromigration legend also displays the color and the data count. The colored histogram bars on the right side of the legend represent the relative distribution of electromigration values in the bins.
To view electromigration data in the map by using the Query tool,
1. Click the Query button on the Map Mode panel to enable the Query tool.
The Query Object panel appears.
2. Click a location in the map for which you want to view the electromigration data.
The electromigration data is displayed on the Query Object panel. The object is highlighted in the query color (white) and the information for the object appears on the Query Object panel.

Displaying the Resistance Power Map

Prerouting Standard Cells

You can connect power and ground pins in standard cells to the straps and rings of the power mesh and connect power and ground rails in the standard cells. To make sure the global router can recognize the routing obstruction, preroute the standard cells before performing global routing.
If the top cell does not have pads, you can use the “Extend to boundaries and generate pins” option in the Preroute Standard Cells dialog box. This option creates a power and ground extension wire to the cell boundary and generates a power and ground pin if the following conditions are met:
• The power and ground pin is not already connected to power and ground in the direction in which IC Compiler would create the extension wire.
• Creating the power and ground extension wire does not cause a design rule violation.
Pins are marked as fixed to prevent them from being moved by place and route operations.

To preroute standard cells,
1. Choose Preroute > Preroute Standard Cells.
The Preroute Standard Cells dialog box appears.
Alternatively, you can use the `preroute_standard_cells` command.



2. Click the Connection Mode tab.

• Select one of the following options:

Rail – Connects standard cells by rails. This is the default.

Tie – Connects standard cells by tying all the power and ground pins to nearby targets.

If you select this option, enter the maximum routing width. The default is 0.

Net – Connects pins of standard cells that are included with the specified net to nearby rings or straps.

If you choose the Net option, you can also specify a maximum fanout. This is the maximum number of pins that are connected to the ground rings or straps. The default is 10.

You can also set the routing layer and width for both vertical and horizontal segments

3. Click the Routing Options tab.

• Connect pins only on layer number – Select this option to connect pins on a specified metal layer. By default, any layer could be used.

• Connect – Select one or both of the following:

Vertically – Connects power and ground in the vertical direction.

Horizontally – Connects power and ground in the horizontal direction. This option is selected by default.

• Determine the pin width – You can determine the pin width by the width of the pin at its extreme edges or by the width of the most extended pin.

4. Click the Scope tab.

• Use top cell's bounding box – This option sets the working area to the top cell's bounding box.

Net names – Enter a list of net names in the text box. The net names in the list must be separated by commas.

• Use specified bounding box – You can select this option to specify a user-defined bounding box.

Coordinates – Enter the x- and y-coordinates in the Coordinates field, or select the Rectangle button and draw the rectangle in the layout view; or select the Rectilinear

button and draw the rectilinear shape in the layout view.
5. Click the DRC button to open the Set Preroute DRC Options dialog box.
6. Click OK or Apply.

Performing Prototype Global Routing

You can perform prototype global routing to get an estimate of the routability and congestion of your design. Global routing is done to detect possible congestion hot spots that might exist in your floorplan due to the placement of the hard macros or inadequate channel spacing.

During global routing, IC Compiler assigns nets to the global routing cells through which they pass. For each global routing cell, the routing capacity is calculated according to the blockages, pins, and routing tracks inside the cell. Although the nets are not assigned to the actual wire tracks during global routing, the number of nets assigned to each global routing cell is noted. IC Compiler calculates the demand for wire tracks in each global routing cell and reports the overflows, which are the number of wire tracks still needed after IC Compiler assigns nets to the available wire tracks in a global routing cell.

IC Compiler might reduce overflows by detouring nets around congested areas and increasing the wire length. You can examine the global routing report that appears in the command window and display congestion maps to help you decide whether your design can be routed.

The global router considers spacing and wide-wire variable routing rules as well as shielding variable routing rules, when calculating congestion.

Global routing is divided into the following phases:
• One initial routing phase, where all the unconnected nets are routed
• One or more rip-up and reroute phases, where for a selected set of nets, the routing results from the previous phase are deleted and nets are rerouted to reduce the congestion

Note:
If, after the second routing phase, the maximum overflow in any direction is greater than 50, both global routing and prototype routing stop because the design is too congested and it is now

unroutable.

To perform global prototype routing do the following:
1. Choose Route > Global Route.
The Zroute dialog box appears.
Alternatively, you can use the `route_zrt_global -effort minimum` command.
2. Select the “Minimum” radio button in the Effort section of the GUI. This option enables prototype global routing.
3. Click OK or Apply.

Performing In-Place Timing Optimization

In-place timing optimization is used to improve the timing of a given design, and in particular, to meet the timing constraints on the design. It is an iterative process based on virtual routing. If you run in-place optimization after global routing, the tool deletes the global routes and optimizes the design based on the virtual route timing.

Running In-Place Timing Optimization

1. Choose Timing > In Place Optimization.
The In Place Optimization dialog box appears.
Alternatively, you can use the `optimize_fp_timing` command.
2. Set the options in the GUI depending on your requirements.
• Effort
You can specify how much effort is used to minimize the worst negative slack in the design. If you select an effort level of high, more effort is expended to improve the timing of the design, resulting in more CPU time. In-place optimization stops when it finds the worst negative slack in the design cannot be further improved. The output is a legally placed netlist.
The default effort is medium.
• Fix design rule
Enable this option to fix design rules. The default is disabled. Design rule violations, such as maximum transition, maximum capacitance, and maximum fanout, are fixed by use of buffer insertion, gate sizing, and automatic high-fanout net (HFN) synthesis for handling medium- and high-fanout nets.
When you run in-place optimization at the virtual route stage, only obvious design rule violations are fixed because wire locations and interconnect are estimated at this stage since timing analysis cannot be completely accurate and runtime is shorter. After you finish global routing, optimization takes longer to run, but the results are based on more accurate timing information.
• Area recovery
Enable this option to direct the in-place optimization engine to invoke additional

operations to try to reduce the cell area without causing timing violations. The area is optimized by removing cells and decreasing the size of the cells on noncritical timing paths. This allows more space for optimization on critical paths.
The default is disabled.
Note:
Area recovery is an operation that tries to reduce the total area of cells used in the design. Area recovery will not cause timing to become worse on the critical paths, but some paths might see an increase in timing as long as it is nonviolating. If the path has positive slack, area recovery might reduce this slack to zero. Area recovery does not cause nonviolating paths (paths with nonnegative slack) to become violating paths.
Area recovery, however, is a difficult and expensive operation, resulting in longer runtime but with a smaller area being used. Designs with high utilization will benefit from area recovery but with a cost that results in higher runtimes.

- Only add buffers for feedthrough nets

Enable this option to only add buffers for feedthrough nets. Optimization is not performed and timing is not updated. Using this option minimizes design changes in the late stages of design planning. The default is disabled.

- High fanout synthesis optimization only

Enable this option to perform only high-fanout synthesis (HFS) on the design. You can combine this option with the “Don’t add any new cells at top level” option when implementing fully abutted or narrow channel designs.
The default is disabled.

3. Click OK or Apply.

Performing Clock Planning

This chapter describes how to reduce timing closure iterations by performing clock planning on a top-level design during the early stages of the virtual flat flow, after plan groups are created and before the hierarchy is committed. You can perform clock planning on a specified clock net or on all clock nets in your design.
Clock planning tries to minimize clock skew by running block-level and top-level clock tree synthesis during the early stages of the design flow to determine the clock budgets, allocate resources for clock buffers and clock routes, determine optimal clock pin locations for soft macros, and provide an estimate of the block-level insertion delays and skew for each plan group prior to finalizing the floorplan. Having optimal clock pin locations is a key factor in meeting the final clock skew and insertion delay numbers with the optimal number of buffers.

Setting Clock Planning Options

Before you can compile clock trees inside the plan groups and build clock trees at the top level, you must first set different clock planning options such as anchor cell insertion, specifying nets for clock planning, and whether or not to route the clock nets after clock

planning. You can also modify the clock tree constraint settings.

To set clock planning options,

1. Choose Clock > Set Clock Plan Options.

The Set Clock Plan Options dialog box opens.

Alternatively, you can use the `set_fp_clock_plan_options` command.

2. Set the options, depending on your requirements.

- Clock Nets – Enter the name of the clock nets on which to do clock planning.
- No Feedthroughs in Plan Groups – Enter a list of plan groups on which you do not want buffers placed. During the top-level clock tree synthesis phase of clock tree planning, no buffers are placed on the plan groups, unless they drive sinks inside those plan groups. This minimizes the creation of feedthroughs.
- Anchor Cell – Enter the name of the cell that is inserted as an anchor cell for all the plan groups. The anchor cell is a clock buffer cell. All plan groups should use the same buffer type. Inverters are not supported. This option is required. If you do not specify the name of the anchor cell, the IC Compiler tool issues an error message.

For each clock interface net (nets that cross plan group boundaries), an anchor cell (driver cell) is inserted for each plan group on every input clock net that crosses a hierarchical block. This partitions the plan group level clock subtree from the top-level clock tree.

An isolation cell is also inserted at the top level (for each output clock port on the plan group) to isolate the plan group level clock subtree from the top-level clock tree.

The anchor cell is inserted inside the plan group at the center of the mass of flip-flops that are connected to each clock net to isolate the top-level clock net from the clock net that is inside the plan group. The input pin of the anchor cell is driven by the clock net, and the output pin of the anchor cell drives the root clock pin of the block.

- Route Mode – Choose whether or not to route the top-level clock nets after clock planning. Based on the routing information, clock pins are created and assigned a location where the route crosses the plan group boundary.

Global route – Select this option to perform global routing on the clock nets.

Detailed route – Select this option to perform detail routing on the clock nets.

None – Select this option if you do not want to route the clock nets. This is the default.

3. Click OK or Apply to set the clock planning options.

You can get a report of the clock plan options by using the report_fp_clock_plan_options command. If no clock plan options have been set, this command reports the default values.

You can remove (reset) the database entries for the clock planning options you set by using the reset_fp_clock_plan_options command.

Performing Clock Planning Operations

Clock planning is done during the early stages of the virtual flat flow (after plan groups have been created and before the hierarchy is committed) to allocate clock resources and provide an estimate of the block-level insertion delay and skew for each plan group, prior to finalizing the floorplan.

You can perform clock planning operations to compile the clock trees inside plan groups and build clock trees at the top level based on the options you have selected in the Set Clock Plan Options dialog box (set_fp_clock_plan_options command).

To perform clock planning operations,

1. Choose Clock > Compile Clock Plan.
The Compile Clock Plan dialog box appears.
Alternatively, you can use the `compile_fp_clock_plan` command.
2. Set the options, depending on your requirements.
- Operation Condition – Select the operating conditions (Max, Min, or Min/Max) for top-level clock tree synthesis and optimization. The default is Max.
- Insert Anchor only – If you select this option, the clock planning tool only inserts anchor cells on the input ports of the plan groups, and does not synthesize the clock plan.

By default, the clock planning tool inserts anchor cells on the input ports of the plan groups plan groups, and then synthesizes the clock plan.
3. Select OK or Apply.
During clock planning, the following operations are performed:
- Anchor cells are inserted on the input ports of the plan groups to isolate the clock trees inside the plan groups from the top-level clock tree.
- Fast clock tree synthesis is run inside each plan group to estimate the insertion delay and skew values at the input of the anchor cells.
- The clock tree synthesis results are annotated on floating pins, which are defined on the anchor cells.
- The top-level clock tree is synthesized to the anchor cell floating pins.
- Detail routing is run on the clock interface nets.

To generate clock tree reports, choose Clock > Report Clock Tree or use the `report_clock_tree` command. By default, the global skew is reported for all the generated clock trees.

Performing Routing-Based Pin Assignment

You can create top-level soft macro pins in hierarchical designs, constrain the pins during pin creation, edit and modify the soft macro pins, and analyze the quality of the pin assignment results. Both pad and pin assignment are supported concurrently.
IC Compiler provides two ways to perform pin assignment: on soft macros using the traditional pin assignment or on plan groups, using the pin-cutting flow.

Setting Pin Assignment Constraints

You can set constraints prior to performing pin assignment. These pin constraints are honored during pin assignment on soft macros during traditional pin assignment and on plan groups in the pin-cutting flow. The pin assignment constraints are saved in the Milkyway database.
Various pin constraint settings are available and can be applied to all soft macros or to a selected set of soft macros. Pin creation layers can be limited to a set of selected layers. Pins can be constrained to avoid corner placement by various factors. You can also set the spacing between pins and preroutes.

To assign pin constraints,
1. Choose Pin Assignment > Pin Constraints.
The Pin Assignment Constraints dialog box appears.
Alternatively, you can use the `set_fp_pin_constraints` command.

You can apply constraints to all soft macros and plan groups or to specified soft macros or plan groups. By default, the specified constraints apply to all soft macros and plan groups.
You can apply constraints to block-level pins instead of soft macro pins. To do this, enable the “Block level pins” option. The default is disabled.
• Specified Ports – Enter a list of specified ports to which the specified block-level non-edge pin constraints apply.
2. Click the Add button to open the Add Pin Assignment Constraints dialog box.
3. Under “Pin Constraints”, click the Settings tab and complete the necessary constraint settings.
Allowed layers from – Select this option to specify the range of consecutive metal layers on which to place soft macro pins.
• Specify the range of allowed layers, from the lowest to highest allowed layers.
4. Under “Pin Constraints,” click the Feedthrough Settings tab and complete the necessary feedthrough settings.
5. Click Apply to write the pin constraint settings to the CEL views of the selected set of soft macros.

Performing Traditional Pin Assignment

To assign soft macros pins,
1. Choose Pin Assignment > Assign Pins.
The Assign Pins dialog box appears.
Alternatively, you can use the `place_fp_pins` command.
2. Choose whether to assign pins in the current design to all soft macros or to selected soft macros. The default assigns pins to all soft macros.
3. Select the effort used to determine how the pins are placed.
4. (Optional) Select the Verbose option to control the amount of output messages that are written to the log file during pin assignment.
5. Click OK or Apply to assign pins to the soft macros.

Finalize and Save the Floorplan

- Lock down the macros

```
set_dont_touch_placement [all_macro_cells]
```

- Save the design in the Milkyway database

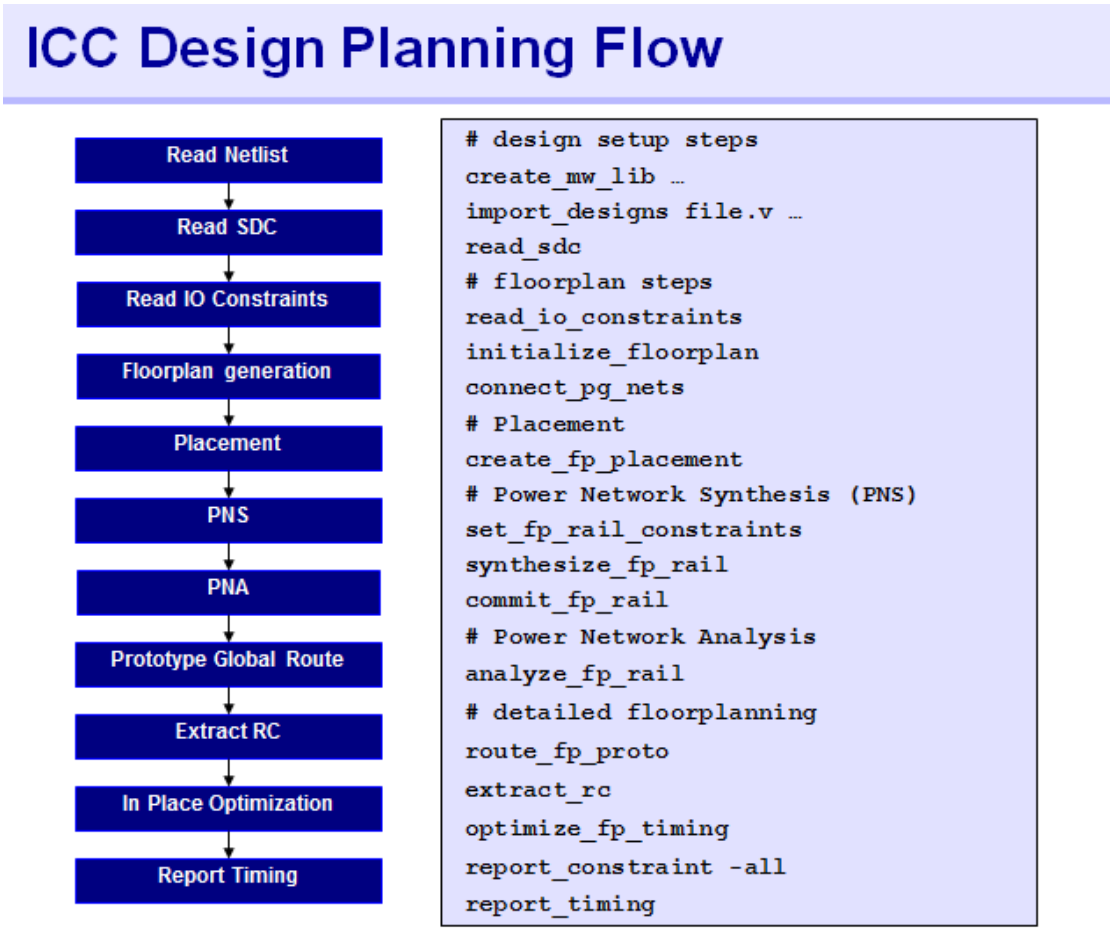
```
save_mw_cel -as floorplanned
```

- Save the floorplan only

```
write_def -output floorplan.def
```

```
write_def -output floorplan.def
write_floorplan floorplan.tcl
```

ICC Design Planning Flow



1. False. Flylines can be drawn to all (including standard cells), macros or IO
2. False. Although it is likely that the core has the worst IR drop because the core is furthest away from

the IO ring, this is not always the case.

3. False. Virtual pads are saved with the CEL. The .vpad file serves as documentation, or to apply the vpads manually.
4. False. Protoroutes are even less accurate than global routes, and cannot be used other than for congestion analysis.

5. import timing constraint

"Timing Constraints" are required to communicate the design's timing intentions to IC Compiler. They should be the same ones used for synthesis with Design Compiler (preferably SDC)。

read_sdc timing_constraints.sdc

If you want to remove the current timing constraints before applying new constraints:
remove_sdc # Removes all existing SDC constraints

remove_sdc

- **Removes all SDC constraints**

remove_ideal_network -all

- **Removes ideal_network attributes, latencies and transitions**

remove_annotations

- **Removes all annotated delays, transition, resistance, capacitance, checks**

To remove all settings:
reset_design

- **Removes all optimization attributes (dont_touch, size_only…) and all constraints.**

6. check timing

- **Before proceeding, you should ensure that the design is completely constrained**
- **IC Compiler will not optimize paths that are not constrained for timing**
- **No checking for missing external loads or drive characteristics will be performed!**

- **check_timing reports all unconstrained paths**
- **False paths are also considered unconstrained!**

- **To verify that unconstrained paths are OK:**
  report_timing_requirements

- Reports false paths set on design
- Compare these paths to the ones reported by **check_timing**

8 palcement

Placement stage

Logical Data | Physical Data | place_opt | clock_opt | route_opt | Analysis | Output

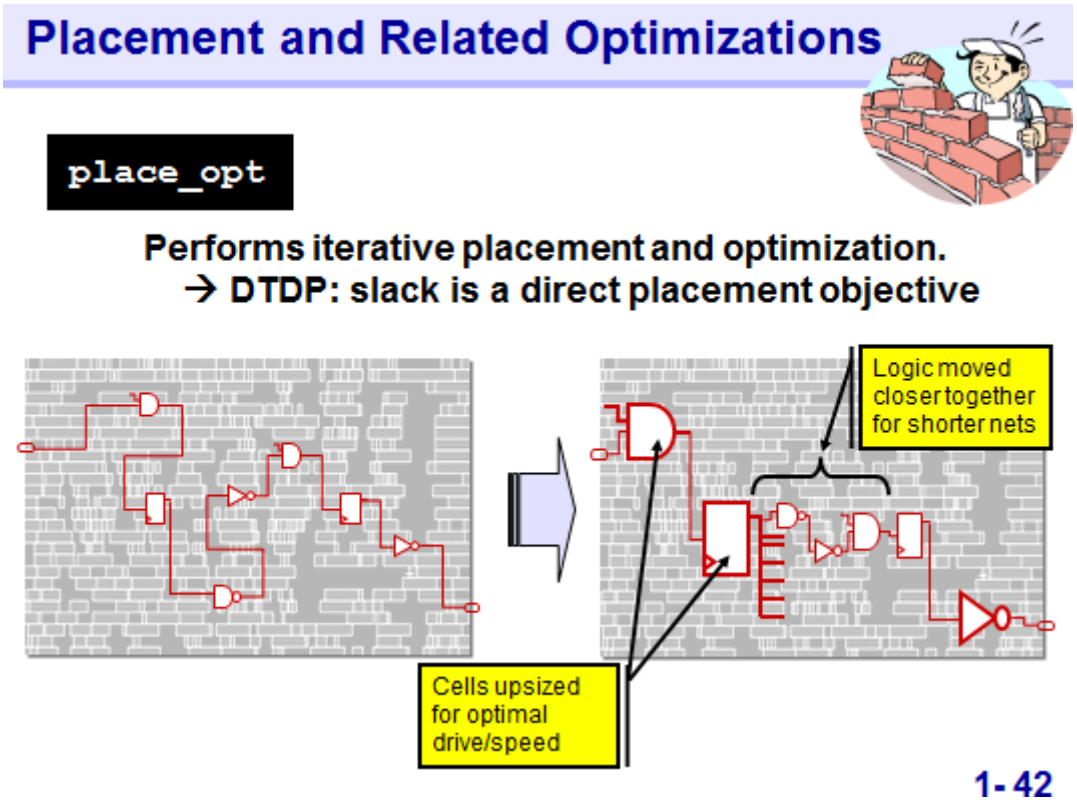
Before starting placement & optimization:

- **Do not over-constrain the design**
  - Constraints should match design specification

- **Report timing before placement**
  - Check for unrealistic or incorrect constraints

```
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)
```

1-41



The place_opt command has switches for congestion, area recovery and power dissipation control.
In the basic flow, the place_opt command is used without switches.
SYNTAX

place_opt
[-area_recovery]
[-congestion]
[-effort]
[-power_mode none | all | leakage]

The Direct Timing-Driven Placement engine (DTDP) is used in the mainstream place_opt flows.
DTDP is the first in the industry to incorporate slack directly into the placement objective. Until now, placers were using indirect slack models like increasing the weight/importance of timing-critical nets.

Defining Placement Blockages

Placement blockages are areas that leaf cells must avoid during placement and legalization, including overlapping any part of the placement blockage. Placement blockages can be hard or soft.

- A hard blockage prevents cells from being put in the blockage area.
- A soft blockage restricts the coarse placer from putting cells in the blockage area, but optimization and legalization can place cells in a soft blockage area.

If you define both hard and soft placement blockages in your design, the hard placement blockages take priority over the soft placement blockages in places where they overlap.

You can define a placement blockage by either specifying a region around fixed macros (keepout margins) or by specifying a rectangular blockage area (area-based placement blockages).

Defining Keepout Margins

A keepout margin is a region (the unshaded portion in Figure 6-1) around the boundary of fixed macros in your design in which no other cells are placed.

Keeping the placement of cells out of such regions avoids congestion and net detouring and produces better QoR.

Keepout margins can be defined as hard or soft. In addition, you can define global keepout margins, which apply to all fixed macros in the design, or cell-specific keepout margins.

Defining Global Keepout Margins

To define a hard global keepout margin, specify the width, in microns, of the keepout margin by setting the physopt_hard_keepout_distance variable. The specified width is used for all sides of each fixed macro in your design.

For example, to specify a hard keepout margin of 10 microns around each fixed macro in your design, enter
icc_shell> set_app_var physopt_hard_keepout_distance 10

To define a soft global keepout margin, specify the width, in microns, of the keepout margin by setting the placer_soft_keepout_channel_width variable.
A soft global keepout margin does not apply to every fixed macro in your design or to every side of the fixed macro; it applies only in the following cases:
- The distance between the fixed macro boundary and the core area boundary is less than the soft keepout margin.
- The distance between two fixed macros is less than the soft keepout margin, forming a thin channel between the objects.

For example, to specify a soft keepout margin of 25 microns around each fixed macro in your design, enter
icc_shell> set_app_var placer_soft_keepout_channel_width 25

Defining Cell-Specific Keepout Margins

To define a keepout margin with different widths on each side or to define a keepout margin for specific cells, use the set_keepout_margin command (or choose Placement > Set Keepout Margin in the GUI). This is the full command syntax:
set_keepout_margin
[-type hard | soft]
[-outer {lx by rx ty}]
[-tracks_per_macro_pin value]
[-min_padding_per_macro value]
[-max_padding_per_macro value]

[-all_macros] [-macro_masters][-macro_instances]
[object_list]
[-north]
For example,
icc_shell> set_keepout_margin -type hard -all_macros \
-outer {10 10 10 10}
Use -type hard or -type soft to specify the type of keepout, either hard or soft. The default is hard. Use -all_macros to apply the keepout margins to all macros, -macro_masters object_list for all instances of specified macro masters, or -macro_instances object_list for specified instances of macros.
You can specify explicit keepout margins by using -outer {lx by rx ty}, where the four numbers are the left, bottom, right, and top margins. A value of 0 results in no keepout margin for that side. The -north option sets the margins with respect to the north orientation of the cell.
To report the cell-specific keepout margins in your design, use the report_keepout_margin command. It reports the keepout margin values set by the set_keepout_margin command, the values of any pin-count-based parameters defined by the set_keepout_margin command, and all derived keepout margins for the specified macro cells and standard cells.
To remove the cell-specific keepout margins from your design, use the remove_keepout_margin command.

Defining Area-Based Placement Blockages

Hard placement blockages can be defined in the DEF as shown in Example 6-1.
Example 6-1 Placement Blockages in DEF
BLOCKAGES 2 ;
- PLACEMENT
RECT (0 327600) (652740 327660) ;
- PLACEMENT
RECT (0 327600) (652740 327660) ;
END BLOCKAGES
1
You can also create placement blockages in IC Compiler by using the create_placement_blockage command or by choosing Floorplan > Create Placement Blockage in the GUI.

For example, to create a soft placement blockage in the area enclosed by a rectangle with corners at (10, 20) and (100, 200), you would use the following command:
icc_shell> create_placement_blockage -bbox {10 20 100 200} -type soft

The -type option specifies the type of blockage: hard, soft, pin, hard_macro, or partial.
Each type of blockage restricts usage of the specified area in the following ways:
- A hard blockage prevents the placement of standard cells and hard macros.
- A soft blockage prevents the placement of standard cells and hard macros unless the

congestion is too high to allow placement elsewhere.

- A pin blockage prevents the global router from routing in the area and the pin placer from assigning pins to the area. You can optionally specify which layers are blocked with the -blocked_layers option. If you do not specify the layers, all layers are blocked.
- A hard_macro blockage prevents the placement of hard macros, but not standard cells.
- A partial blockage partially prevents placement of cells in an area. The -blocked_percentage integer option must be used to specify the percentage of the area that is blocked.

For example, to set a partial blockage of 40 percent of the area enclosed by the rectangle with corners at (10,20) and (100,200), you would use the following command:

```
icc_shell> create_placement_blockage -bbox {10 20 100 200} \
-type partial -blocked_percentage 40
```

In this example, the specified area has a blockage of 40 percent, so the maximum allowed cell density is 60 percent. Partial blockage applies only to coarse placement. It has no effect on legalization or optimization.

You can optionally assign a name to a blockage by using the -name option. You can then reference that blockage by name when you use the get_placement_blockage or remove_placement_blockage command.

You can display and select placement blockages in the layout view of the GUI.

If you specify the -no_register option, the coarse placer does not place any register cells in the specified hard, soft, and partial blockage areas.

To report placement blockages in the design, use the report_placement_blockages command.

To return a collection of placement blockages in the current design that match certain criteria, use the get_placement_blockages command.

To remove placement blockages from the design, use the remove_placement_blockage command.

For more information about the create_placement_blockage command, see the man page.

Preventing Cell Placement in the Default Voltage Area

In a UPF flow, a certain multivoltage design style implements abutted voltage areas at the top level, so no space is allowed for cell placement in the default voltage area. You can achieve this design style by setting the mv_no_cells_at_default_va variable to true, changing it from its default of false. Setting this variable prevents the tool from placing cells in the default voltage area, whether at the top level or not, to connect submodules. Instead, the tool inserts cells in the lower-level voltage areas.

For more information about the default voltage area, see “Default Voltage Area” on page 14-31.

Setting Placement Options

There are many configuration settings that affect the behavior of placement in IC Compiler, including the following controls:

- Setting the Congestion Options
- Setting the Move Bounds
- Defining Intercell and Boundary Spacing Rules
- Defining the Buffer Strategy for Optimization
- Setting the Preferred Buffers for Hold Fixing
- Setting Up Multithreading
- Enabling Tie Cell Insertion
- Setting Placement and Optimization Attributes

Setting the Congestion Options

IC Compiler attempts to minimize congestion during placement and optimization. Congestion occurs when the number of wires going through a region exceeds the capacity of that region. This condition is detected by global routing. IC Compiler places and moves cells in such a manner to avoid congestion and to fix congestion problems when they occur.

You can set certain options related to congestion avoidance with the set_congestion_options command:

```
set_congestion_options
[-max_util value]
[-layer name]
[-availability value]
[-coordinate {X1 Y1 X2 Y2}]
```

The -max_util option specifies how densely cells can be packed in uncongested regions to relieve 解除 congestion in other regions. You should set this variable based on how much area is needed for placement. The default maximum utilization is 0.95.
The -layer and -availability options specify how much of the routing resource for the given layer is available to be used. For example, in a design whose M5 and M6 layers will be 70 percent occupied by future power and ground routing, you can set the availability of M5 and M6 to 0.30.

By default, the placer uses the placement virtual route congestion map for congestion removal during placement. If you set the placer_enable_enhanced_router variable to true, changing it from its default of false, the placer uses the global route congestion map during the following congestion-driven placement stages:

- create_placement -congestion
- place_opt -congestion
- place_opt -effort high
- place_opt_feasibility -congestion
- psynopt -congestion
- refine_placement

Setting the Move Bounds

Defining the Buffer Strategy for Optimization

During the optimization step, the place_opt command introduces buffers and inverters to fix timing and DRC violations. However, this buffering strategy is local to some critical paths. The buffers and inverters that are inserted become excess later because critical paths change during the course of optimization. You can reduce the excess buffer and inverter counts after place_opt by using the set_buffer_opt_strategy command, as shown in the following example:
icc_shell> set_buffer_opt_strategy -effort low
This buffering strategy will not degrade the quality of results (QoR).

1 Before starting placement & optimization:

Check for unrealistic or incorrect constraints:

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)

What should you see in a zero interconnect timing report?
It is not necessary that the timing report show positive slack. Small negative slacks can be fixed by IC Compiler's advanced optimization engines. The problems are with large violations. If for example a path shows a timing violation as large as the clock period, then you should investigate whether the constraints were applied correctly, or whether the design was not synthesized with the correct constraints.

2 Placement and Related Optimizations

The place_opt command has switches for congestion, area recovery and power dissipation control.
In the basic flow, the place_opt command is used without switches.
SYNTAX

```
place_opt
                [-area_recovery]
                [-congestion]
                [-effort]
                [-power_mode none | all | leakage]
```

The Direct Timing-Driven Placement engine (DTDP) is used in the mainstream place_opt flows.
DTDP is the first in the industry to incorporate slack directly into the placement objective. Until now, placers were using indirect slack models like increasing the weight/importance of timing-critical nets.

Performs iterative placement and optimization.

9 clock tree synthesis

Clock Tree Synthesis

Logical Data | Physical Data | place_opt | clock_opt | route_opt | Analysis | Output

1. **Set the clock tree options/exceptions**
2. **Run the `clock_opt` command**

`clock_opt`

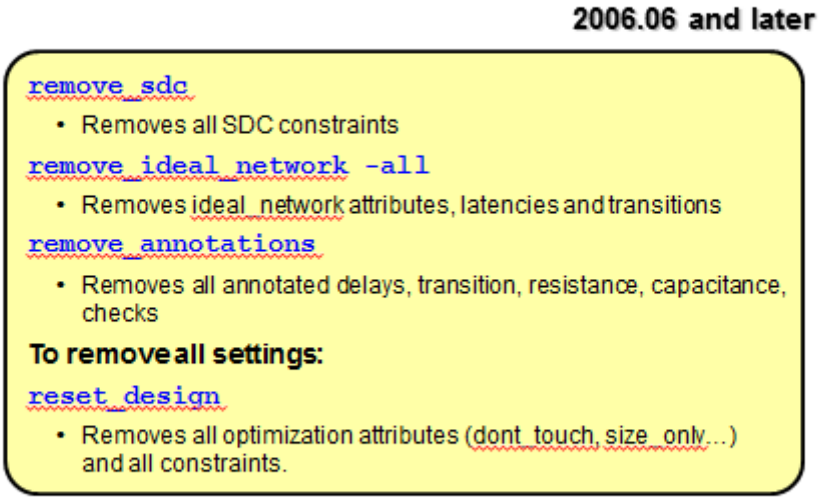
- Builds the clock trees
- Performs incremental logic and placement optimizations
- Runs clock tree optimizations
- Routes the clock nets

Optionally, `clock_opt` can

- Fix hold time violations
- Perform inter-clock balancing

1-43

Constraint Management



SDC 1- 19

Prerequisites for Clock Tree Synthesis

Design Prerequisites

Before running clock tree synthesis, your design should meet the following requirements:

- The design is placed and optimized.

Use the check_legality -verbose command to verify that the placement is legal.
Running clock tree synthesis on a design that does not have a legal placement might result in long runtimes and reduced QoR.
The estimated QoR for the design should meet your requirements before you start clock tree synthesis. This includes acceptable results for

- Congestion

If congestion issues are not resolved before clock tree synthesis, the addition of clock trees can increase congestion. If the design is congested, you can rerun place_opt with the -congestion and -effort high options, but the runtime can be long.

- Timing
- Maximum capacitance
- Maximum transition time

To ensure that the clock tree can be routed, verify that the placement is such that the clock sinks are not in narrow channels and that there are no large blockages between the clock root and its sinks. If these conditions occur, fix the placement before running clock tree synthesis.

- The power and ground nets are prerouted.
- High-fanout nets, such as scan enables, are synthesized with buffers.

Library Prerequisites

Library Prerequisites
Before you run clock tree synthesis, your libraries must meet the following requirements:
- Any cell in the logic library that you want to use as a clock tree reference (a buffer or inverter cell that can be used to build a clock tree) or for sizing of gates on the clock network must be usable by clock tree synthesis and optimization.

By default, clock tree synthesis and optimization cannot use buffers and inverters that have the dont_use attribute to build the clock tree. To use these cells during clock tree synthesis and optimization, you can either remove the dont_use attribute by using the remove_attribute command or you can override the dont_use attribute by specifying the cell as a clock tree reference by using the set_clock_tree_references command, as described in “Specifying the Clock Tree References” on page 7-22.
- The physical library should include
- All clock tree references (the buffer and inverter cells that can be used to build the clock trees)
- Routing information, which includes layer information and nondefault routing rules
- TLUPlus models must exist.

Extraction requires these models to estimate the net resistance and capacitance.

Analyzing the Clock Trees

Before running clock tree synthesis, analyze each clock tree in your design to determine its characteristics and its relationship to other clock trees in the design.
For each clock tree, determine
- What the clock root is
- What the desired clock sinks and clock tree exceptions are

IC Compiler supports the following types of clock tree exceptions: exclude pins, stop pins, float pins, don’t touch subtrees, don’t buffer nets, and don’t size cells.
- Whether the clock tree contains preexisting cells, such as clock-gating cells

If your design contains existing clock trees, you might want to either identify them or remove them before running clock tree synthesis. For information about handling existing clock trees, see “Handling Existing Clock Trees” on page 7-53.
- Whether the clock tree converges, either with itself (a convergent clock path) or with another clock tree (an overlapping clock path)
- Whether the clock tree has timing relationships with other clock trees in the design, such as interclock skew requirements
- What the logical design rule constraints (maximum fanout, maximum transition time, and maximum capacitance) are
- What the routing constraints (routing rules and metal layers) are

Use this information when you define the clock trees and to validate that IC Compiler has the correct clock tree definitions.
Note:
You can generate clock tree reports to analyze the clock tree structure, even before you perform clock tree synthesis and optimization. For information about generating clock tree reports, see “Generating Clock Tree Reports” on page 7-107.

Defining the Clock Trees

IC Compiler uses the clock sources defined by the create_clock command as the clock roots and derives the default set of clock sinks by tracing through all cells in the transitive fanout of the clock roots. You can designate either an input port or internal hierarchical pin as a clock source. To disallow clock sources defined on a hierarchical pin, set the cts_enable_clock_at_hierarchical_pin variable to false before using the create_clock command. This variable is true by default.

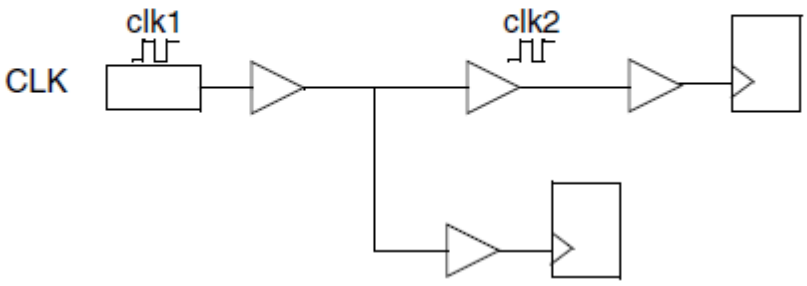
Note:
IC Compiler will not synthesize a clock tree if its source is set to a constant value by using set_case_analysis.

In addition to simple clock trees, IC Compiler also supports cascaded clock trees (a clock tree that contains another clock tree in its fanout). The nested clock tree can either have its own source (identified by the create_clock command) or be a generated clock (identified by the create_generated_clock command).

Cascaded Clocks

If a nested clock tree has its own source, as shown in the example that is given in Figure 7-1, IC Compiler considers the source pin of the driven clock (clk2 in this example) to be an implicit exclude pin of the driving clock (clk1 in this example). Sinks of the driven clock are not considered sinks of the driving clock.

Figure 7-1 Cascaded Clock With Two Source Clocks

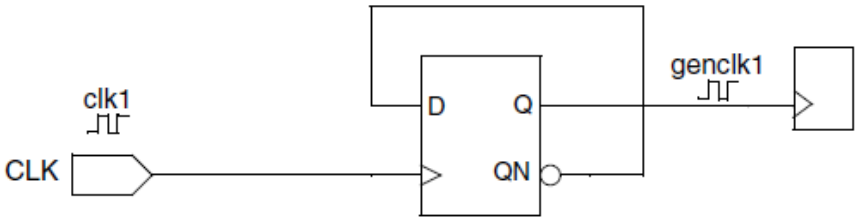


To verify that the clock sources are correctly defined, use the `check_clock_tree` command.

Cascaded Generated Clocks

If a nested clock tree has a generated source, as shown in Figure 7-2, IC Compiler traces back to the master clock source from which the generated clock is derived and considers the sinks of the generated clock to be the sinks of the driving clock tree.

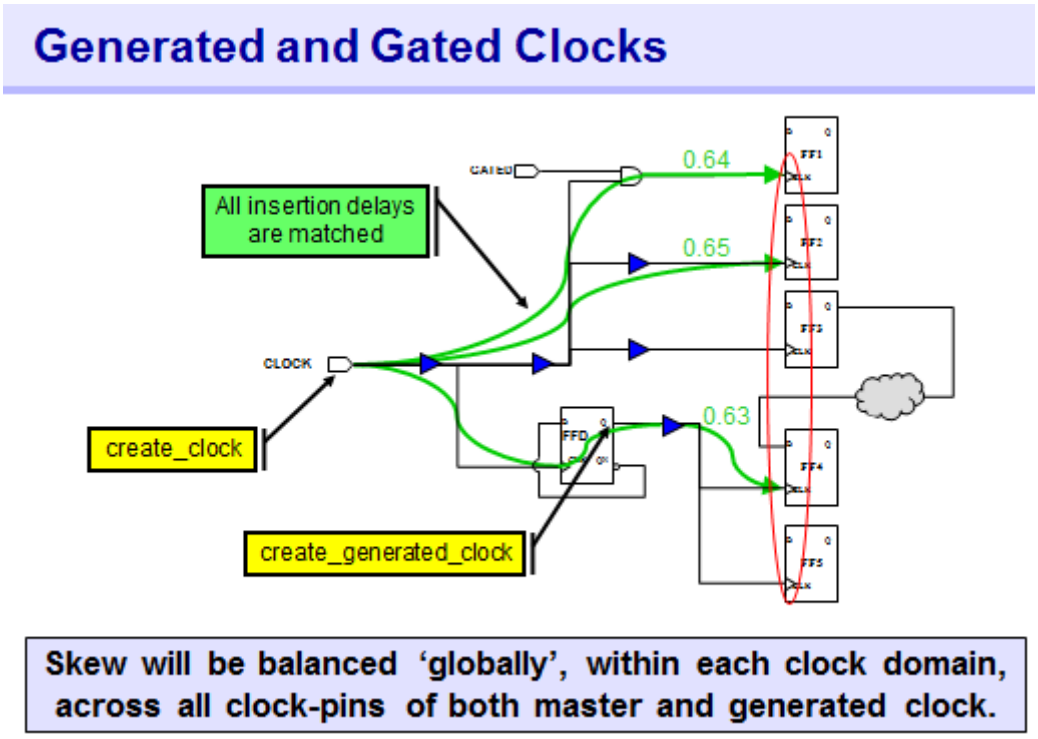
Figure 7-2 Cascaded Clock With Generated Clock



Incorrectly defining the master clock source, as in the following cases, results in poor skew and timing QoR.

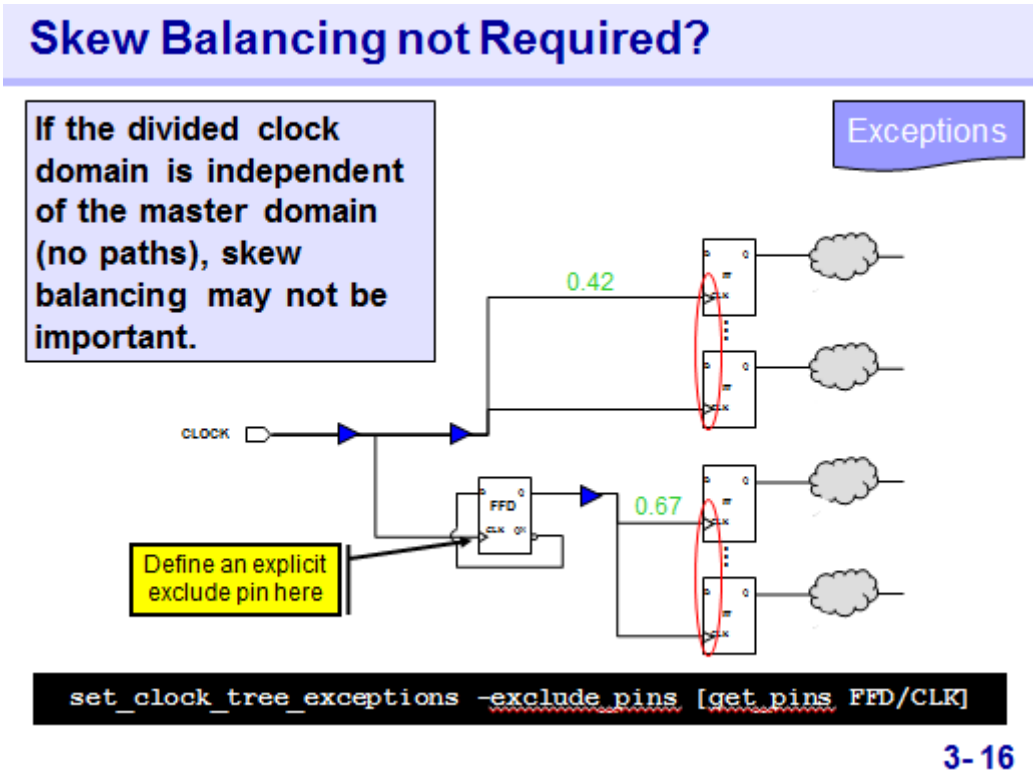
- If IC Compiler cannot trace back to the master clock source, the tool cannot balance the sinks of the generated clock with the sinks of its source.
- If the master clock source is not a clock source defined by create_clock or create_generated_clock, IC Compiler cannot synthesize a clock tree for the generated clock or its source.

Generated and Gated Clocks



The start point of a generated clock is traced back to the source of the create_clock command that the

generated clock is derived from.
By default, CTS will not balance the skew of FFs that cross clock domains. Generated clocks are considered to be part of the same clock domain as their source clock, and are therefore synthesized together with the source clock. In case of generated clocks, a skew report of the master clock also considers and contains the sinks of the generated clock. Also, ICC generates a separate clock tree report for the generated clock also.
By default, CTS will balance the skew 'globally'. This means that the skew of all sequential devices in the same clock domain, related or not, will be balanced. Two sequential devices are related if there is a data path connection between them.



By defining an explicit exclude pin, the two "domains" – CLOCK and divided CLOCK – will no longer be optimized together for skew.

- Exclude pins

Exclude pins are clock tree endpoints that are excluded from clock tree timing calculations and optimizations. IC Compiler uses exclude pins only in calculations and optimizations for design rule constraints.

Identifying the Clock Tree Endpoints

Implicit STOP (sync) pins are automatically defined by CTS on:
- Clock pins of sequential cells (FFs and latches)
- Clock pins of macros

Implicit EXCLUDE (ignore) pins are automatically defined by CTS on:
- Non-clock input pins of sequential cells (D, Set, Reset, etc)
- output ports
-Pins of combinational cells without any fanout or with disabled timing arcs
-Pins with 3-state enable arc
-Select/Control pin of mux used in the data path
-Input pin of pre-existing gate, if all pins in its fanout are *exclude pins*
- Incorrectly defined clock pins (missing or incorrect pin definition in the standard cell or macro cell FRAM view)
-Clock pins without trigger edge info
-Clock pins without a timing arc to the corresponding output pin

When CTS defines implicit EXCLUDE pins these are reported as warnings in the log file. Since an implicit ignore pin may imply a problem with either a reference library cell or design connectivity, it is important to verify that all implicit ignore pins are acceptable and intended.

Clock paths have two types of endpoints:

- Stop pins

Stop pins are the endpoints of the clock tree that are used for delay balancing. During clock tree synthesis, IC Compiler uses stop pins in calculations and optimizations for both design rule constraints and clock tree timing (skew and insertion delay).
Stop pins are also referred to as sink pins.

- Exclude pins

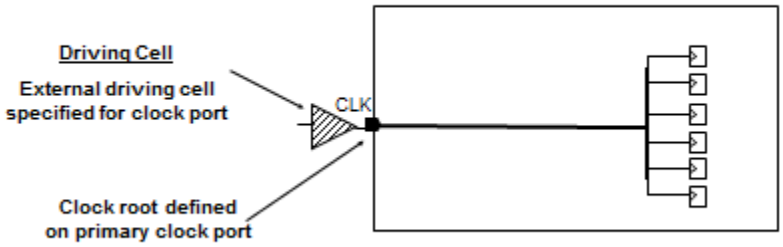
Exclude pins are clock tree endpoints that are excluded from clock tree timing calculations and optimizations. IC Compiler uses exclude pins only in calculations and optimizations for design rule constraints.

Defining the Clock Root Attributes

Define Clock Root Attributes (1/2)

- **When the clock root is a primary port of a block**
  - Ensure that an appropriate driving cell is defined

    `set_driving_cell`
  - The synthesis constraints may include a weak driving cell for all inputs, including the clock port
  - Because the clock is ideal during synthesis it has no effect on design QoR
  - But **a weak driver on the clock port affects clock tree QoR during CTS**

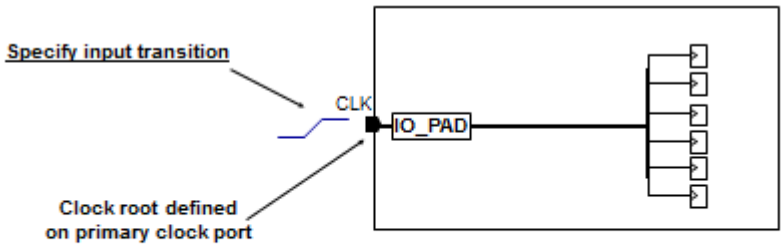


3-12

Define Clock Root Attributes (2/2)

- **When the clock root is a primary port, but at the CHIP-level through an IO-PAD**
  - Ensure that an appropriate input transition is defined

    `set_input_transition`



3-13

set_input_transition –rise 0.3 [get_ports CLK]
set_input_transition –fall 0.2 [get_ports CLK]

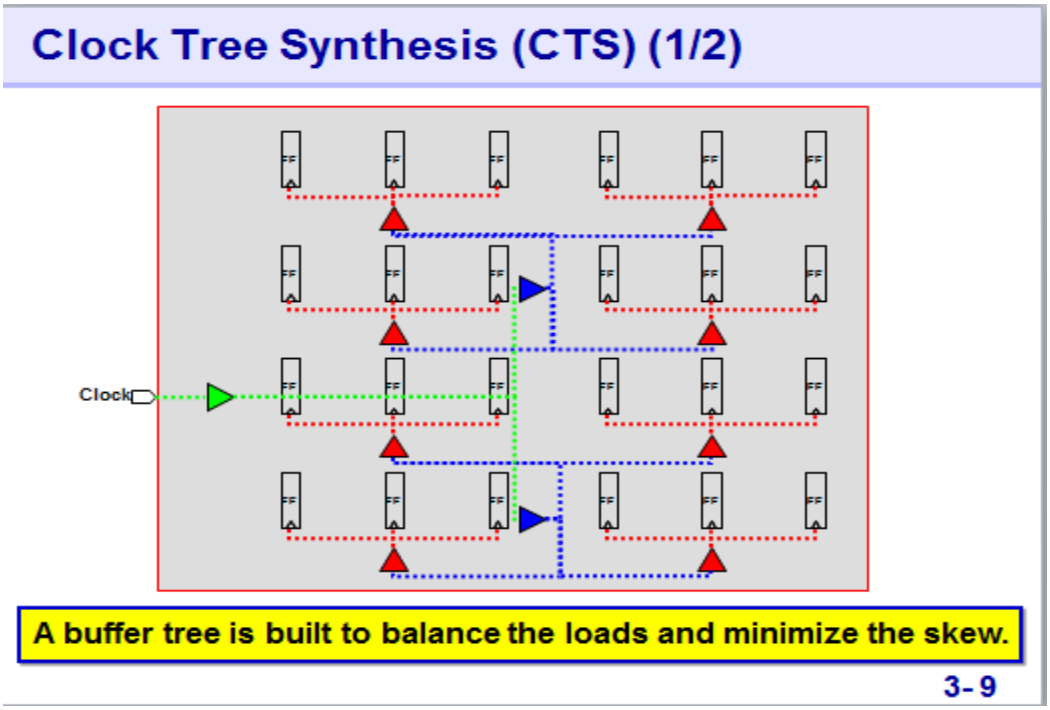
If the clock root is an input port (without an I/O pad cell), you must accurately specify the driving cell of the input port. A weak driving cell does not affect logic synthesis, because

logic synthesis uses ideal clocks. However, during clock tree synthesis, a weak driving cell can cause IC Compiler to insert extra buffers as the tool tries to meet the clock tree design rule constraints, such as maximum transition time and maximum capacitance.

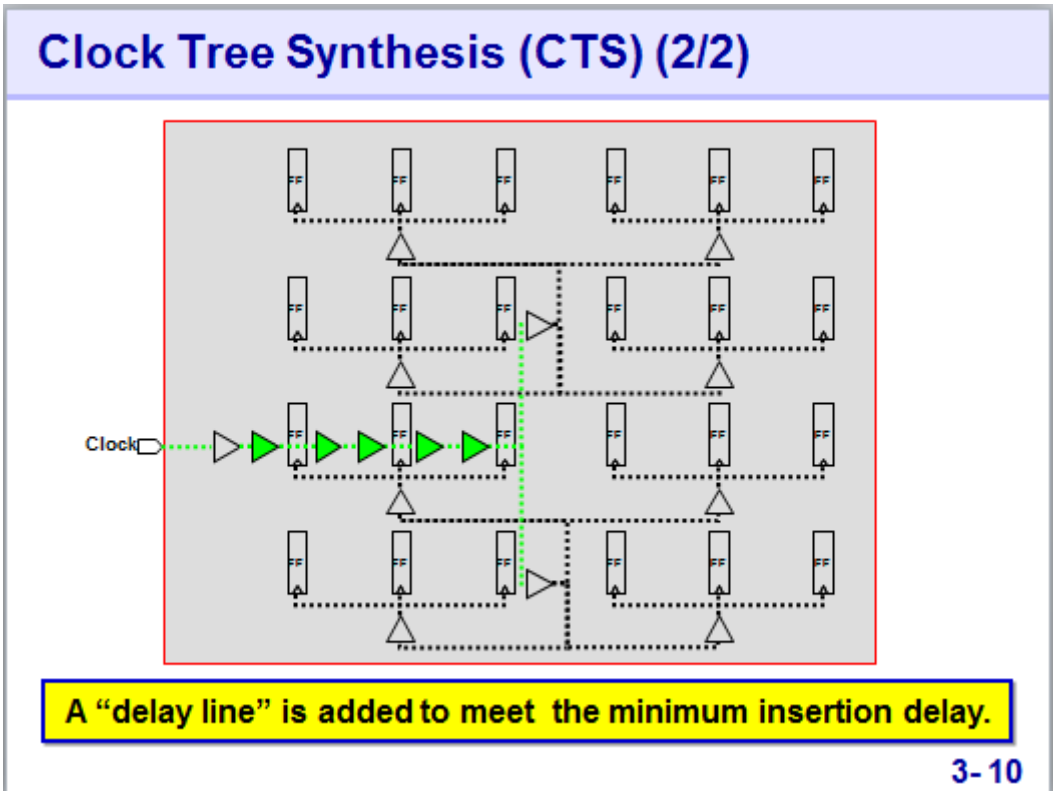
For example, if clock tree CLK1 has input port CLK1 as its root and CLK1 is driven by cell CLKBUF, enter
icc_shell> set_driving_cell -lib_cell mylib/CLKBUF [get_ports CLK1]
If you do not specify a driving cell (or drive strength), IC Compiler assumes that the port has infinite drive strength.

If the clock root is an input port with an I/O pad cell, you must accurately specify the input transition time of the input port.
For example, if clock tree CLK1 has input port CLK1 as its root and the I/O pad cell has already been inserted, enter
icc_shell> set_input_transition -rise 0.3 [get_ports CLK1]
icc_shell> set_input_transition -fall 0.2 [get_ports CLK1]

理解 skew 和 clock insertion delay



This example shows a clock tree with three buffer levels between the clock source and the clock pins of FFs. Buffers are added to balance the loads (max tran/cap/fanout rules). The insertion delays from the clock source to the clock pin of all FFs are optimized to be as equal as possible in order to minimize the clock skew.
The clock tree is built from the leafs (registers) up.



A “delay line” is typically made up of back to back buffers.
通常 min **insertion delay** 都设成 0.。。。在 hier 的设计中才会有不为 0 的需要吧
对于设定 cts 中的 Min**delay**，建议设置为 0。如果不设置为零，cts 时工具会在 root 与第一级 buffer 间加 buffer，以满足你设定的值，这样不推荐

Where does the Clock Tree Begin and End?

Clock tree begins at SDC-defined clock source:
create_clock
Clock tree ends at “sinks”. These are:
Stop pins
Float pins
Exclude pins (aka ignore pins)
The command report_clock_tree is a good tool to analyze your clock tree structure *before* actually synthesizing it. The command can output:

- Overview for each clock
- A clock tree summary
- A list of Ignore Pins
- The overlap of clock domains
- And more…

Specifying Clock Tree Exceptions

exception can only be set for master clock
To define clock tree exceptions, use the set_clock_tree_exceptions command or choose Clock > Set Clock Tree Exceptions in the GUI. You can set clock tree exceptions on pins or hierarchical pins. If a pin is on paths in multiple clock domains, you can define different clock

tree exceptions for each clock domain.

By default, when you use the set_clock_tree_exceptions command, the clock tree exceptions apply to all clocks for all multicorner-multimode scenarios. You can specify clock tree exceptions on a per-clock basis for the current scenario by using the -clocks option. For example, to specify a stop pin exception for pins on the path in the CLK1 clock domain, enter

```
icc_shell> set_clock_tree_exceptions -stop_pins \
-clocks CLK1 [get_pins pins_on_the_clock_path]
```

When you use the -clocks option, the following restrictions apply:

- The specified clock tree exceptions are for the current scenario.
- The allowed clock tree exceptions are nonstop pins, exclude pins, float pins, and stop pins.
- The specified clocks must come from the current scenario.
- The specified clocks must be master clocks, which are defined by the create_clock command. If you specify generated clocks, the tool issues an error message.
- The specified pins should be on the paths of either the specified master clocks or the generated clocks based on the specified master clocks.

When you use the set_clock_tree_exceptions command without the -clocks option, the tool

- Applies the clock tree exceptions to all clocks for all multicorner-multimode scenarios. To specify clock tree exceptions for the current scenario only, use the -current_scenario option. Do not use the -current_scenario option with the -clocks option.
- Defines the clock tree exceptions for all master clocks that contain the paths of the pin, including the paths of the generated clocks.

For example, a master clock CLK1 has a generated clock GCLK1, which has a generated clock GGCLK1, and another master clock CLK2 has a generated clock GCLK2, which has a generated clock GGCLK2. The PIN_A pin is on the paths of generated clocks GGCLK1 and GGCLK2, but not on the paths of the other clocks. When you use the following command, IC Compiler sets PIN_A as a float pin for the CLK1 and CLK2 clock domains.

```
icc_shell> set_clock_tree_exception -float_pins [get_pins PIN_A]
```

To see the clock tree exceptions defined in your design, generate a clock tree exceptions report by running the report_clock_tree -exceptions command or by choosing Clock > Report Clock Tree in the GUI and selecting Exceptions.

You remove clock tree exceptions based on how they are defined:

- If a clock tree exception is defined without the -clocks option, use the remove_clock_tree_exceptions command without the -clocks option.
- If a clock tree exception is defined with the -clocks option, use the remove_clock_tree_exceptions -clocks command.

You can also remove clock tree exceptions by choosing Clock > Remove Clock Tree Exceptions in the GUI.

Note:

If your design contains sequential cells with unconnected outputs, the clock pins of these

cells are marked as implicit ignore pins. When you run clock tree synthesis, these unloaded sequential cells are deleted from the design. As a result, at the end of clock tree synthesis, you no longer see these implicit ignore pins. To prevent the removal of these sequential cells, set the physopt_delete_unloaded_sequential_cells variable to false before running clock tree synthesis.

Precedence of Clock Tree Exceptions

If you issue the set_clock_tree_exceptions command multiple times for the same pin, the pin keeps the highest-priority exception. IC Compiler prioritizes the clock tree pin exceptions in the following order:

1. Nonstop pins
2. Exclude pins
3. Float pins
4. Stop pins

Note:

The don't touch subtree exception is compatible with the nonstop, exclude, float, or stop pin exception. You can set both exceptions on a pin, and clock tree synthesis honors both exceptions.

In the following example, PIN_A is on the path in clock domain CLK1. The first command specifies PIN_A as a float pin for CLK1. The second command specifies PIN_A as an exclude pin, which takes higher precedence over a float pin exception. The third command attempts to specify PIN_A as a stop pin. Because higher precedence is given to an exclude pin exception, PIN_A remains an exclude pin for CLK1.

```
icc_shell> set_clock_tree_exceptions -clocks CLK1 \
-float_pins [get_pins PIN_A]
icc_shell> set_clock_tree_exceptions -clocks CLK1 \
-exclude_pins [get_pins PIN_A]
icc_shell> set_clock_tree_exceptions -clocks CLK1 \
-stop_pins [get_pins PIN_A]
```

When you define multiple exceptions on the same pin, the set_clock_tree_exceptions command with the -clocks option always overrides the command without the -clocks option. The same rule applies even when the command without the -clocks option sets a higher-priority exception than the command with the -clocks option.

For example, the following commands set PIN_A as a stop pin for CLK1 but a nonstop pin for other clocks.

```
icc_shell> set_clock_tree_exceptions -non_stop_pins [get_pins PIN_A]
icc_shell> set_clock_tree_exceptions -clocks CLK1 \
-stop_pins [get_pins PIN_A]
```

After you set a clock tree exception on a pin globally, if you create a new clock definition that contains a path with that pin, the new clock definition inherits the same clock tree exception.

Specifying Nonstop Pins

Nonstop pins are pins that would normally be considered endpoints of the clock tree, but instead IC Compiler traces through them to find the clock tree endpoints. The clock pins of sequential cells driving generated clocks are implicit nonstop pins. In addition, IC Compiler

supports user-defined (or explicit) nonstop pins.
To specify a nonstop pin, use the set_clock_tree_exceptions -non_stop_pins command.

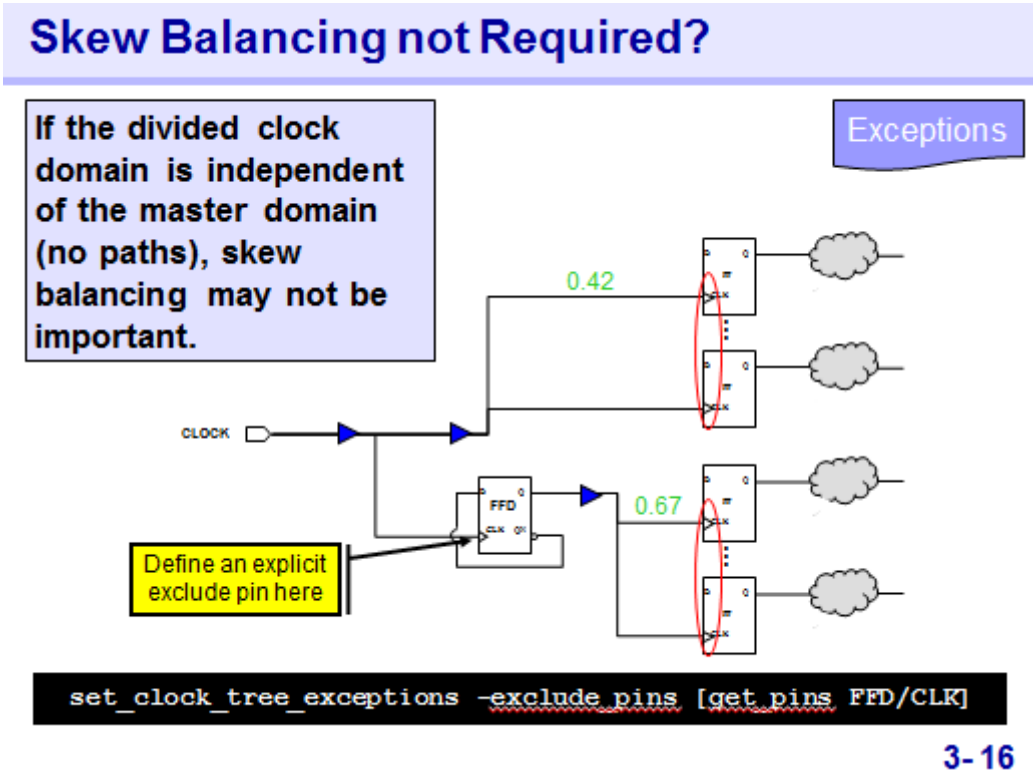
For example, to specify pin U2/CLK as a nonstop pin, enter
icc_shell> set_clock_tree_exceptions -non_stop_pins [get_pins U2/CLK]
To remove the nonstop pin definition from a pin, use the remove_clock_tree_exceptions -non_stop_pins command.

For example, to remove the nonstop pin definition from pin U2/CLK, enter
icc_shell> remove_clock_tree_exceptions -non_stop_pins [get_pins U2/CLK]

Specifying Exclude Pins

Exclude pins are clock tree endpoints that are excluded from clock tree timing calculations and optimizations. IC Compiler uses exclude pins only in calculations and optimizations for design rule constraints. In addition to the exclude pins inferred by IC Compiler (the implicit exclude pins), IC Compiler supports user-defined (or explicit) exclude pins. For example, you might define an exclude pin to exclude all branches of the clock tree that fan out from some combinational logic or to exclude an implicit stop pin.

- **EXCLUDE Pins:**
  - CTS ignores both clock tree DRCs and targets
  - clock_opt's psynopt will fix DRCs to meet library or SDC constraints

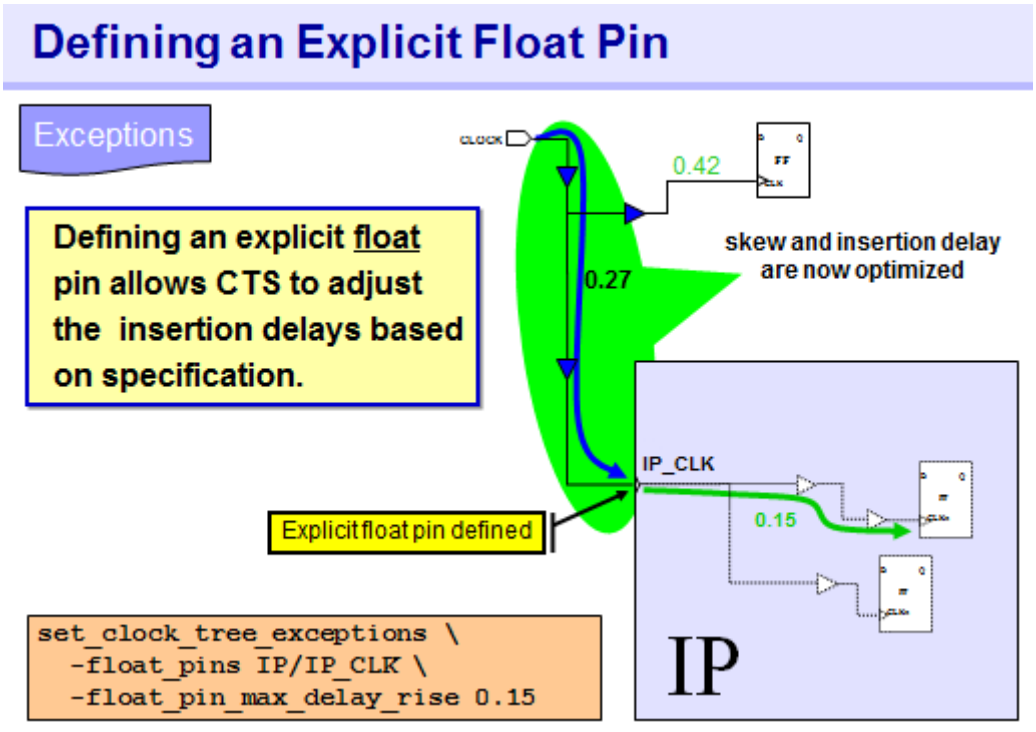


By defining an explicit exclude pin, the two "domains" – CLOCK and divided CLOCK – will no longer be optimized together for skew.

Implicit EXCLUDE (ignore) pins are automatically defined by CTS on:
- Non-clock input pins of sequential cells (D, Set, Reset, etc)
- output ports
Pins of combinational cells without any fanout or with disabled timing arcs
Pins with 3-state enable arc
Select/Control pin of mux used in the data path
Input pin of pre-existing gate, if all pins in its fanout are exclude pins
- Incorrectly defined clock pins (missing or incorrect pin definition in the standard cell or macro cell FRAM view)
Clock pins without trigger edge info
Clock pins without a timing arc to the corresponding output pin

During clock tree synthesis, IC Compiler isolates exclude pins (both implicit and explicit) from the clock tree by inserting a guide buffer before the pin. Beyond the exclude pin, IC Compiler never performs skew or insertion delay optimization, but does perform design rule fixing.
To specify an exclude pin, use the set_clock_tree_exceptions -exclude_pins command.
For example, to exclude clock sink U2/CLK, enter
icc_shell> set_clock_tree_exceptions -exclude_pins [get_pins U2/CLK]
To remove the exclude pin definition from a pin, use the remove_clock_tree_exceptions -exclude_pins command.
For example, to remove the exclude pin definition from pin U2/CLK, enter
icc_shell> remove_clock_tree_exceptions -exclude_pins [get_pins U2/CLK]

Specifying Float Pins



The max and min options are used to specify the delays under max (worst) and min (best) operating conditions, if min_max CTS is performed.

Float pins are clock pins that have special insertion delay requirements. IC Compiler adds the float pin delay (positive or negative) to the calculated insertion delay up to this pin.
To specify a float pin and its timing characteristics, use the following set_clock_tree_exceptions options:

- -float_pins [get_pins pin_list]
- -float_pin_max_delay_fall max_delay_fall_value
- -float_pin_max_delay_rise max_delay_rise_value
- -float_pin_min_delay_fall min_delay_fall_value
- -float_pin_min_delay_rise min_delay_rise_value
- -float_pin_logic_level logic_level_value

The -float_pin_logic_level option specifies the number of logic levels beyond the float pin (this information is used for logic-level balancing). For more information about logic-level balancing, see “Enabling Logic-Level Balancing” on page 7-40.

- **FLOAT Pins:**
  - Like Stop pins, but with delays on clock pin

Note:
If you use the -float_pins option, you must specify at least one of the float pin delay options or an error occurs.

IC Compiler assumes the active edge of the float pin based on the specified float delay options. Table 7-2 describes the edge-aware float pin behavior.

For example, to define an active-low float pin (so that clock tree synthesis considers only the falling edge), enter

```
icc_shell> set_clock_tree_exceptions -float_pins [get_pins U1/CLK] \
-float_pin_max_delay_fall 0.10 -float_pin_min_delay_fall 0.08
```

The clock tree exceptions report (report_clock_tree -exceptions) shows the float pin values only for the active edge. For example, the report for the float pin defined above shows

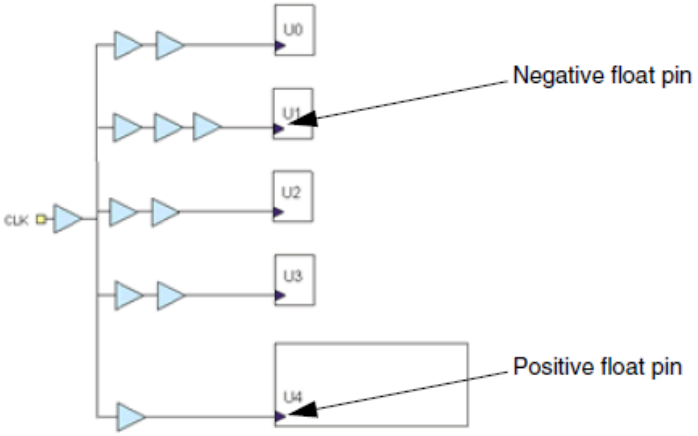
```
Explicit sync pins: 1
(F) U1/CLK ( - 0.100 - 0.080 )
```

The float pin delay values can be either positive or negative, depending on your timing requirements. To increase the path delay to a pin, specify a negative float pin delay. To decrease the path delay to a pin, specify a positive float pin delay.

For example, the following float pin exceptions result in a clock tree similar to that shown in

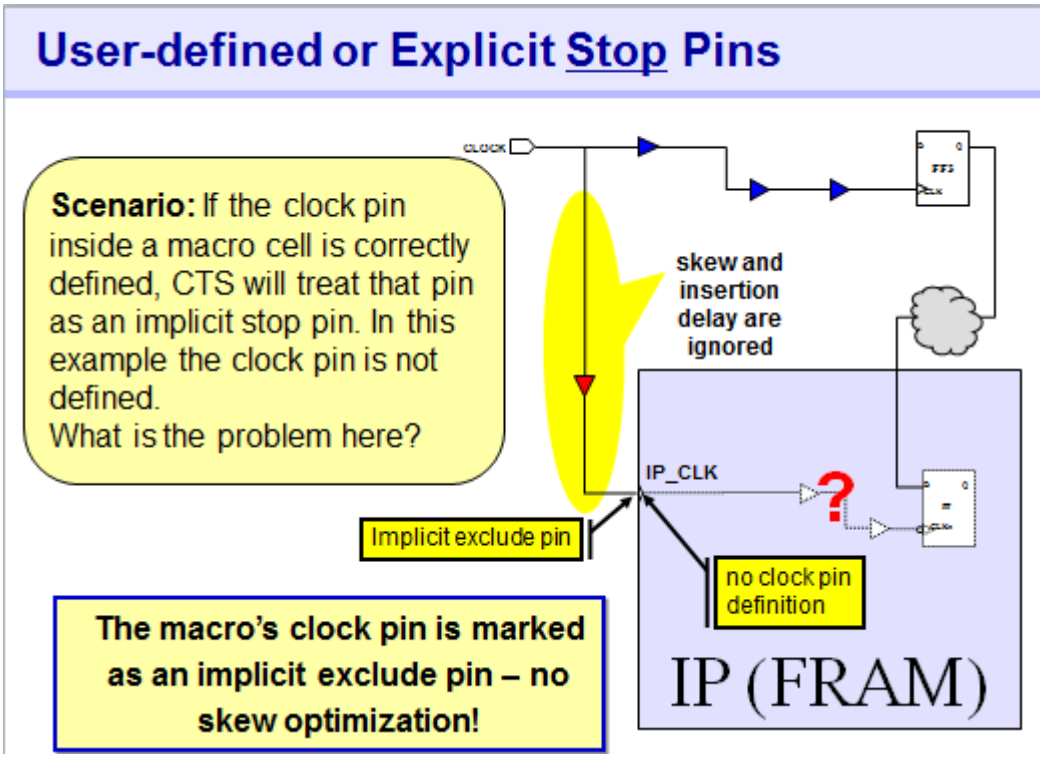
Figure 7-5:
Specifying a negative float pin
icc_shell> set_clock_tree_exceptions -float_pins U1/CLK \
-float_pin_max_delay_rise -0.5 -float_pin_max_delay_fall -0.5
Specifying a positive float pin
icc_shell> set_clock_tree_exceptions -float_pins U4/CLK \
-float_pin_max_delay_rise 0.5 -float_pin_max_delay_fall 0.5

Figure 7-5 Float Pin Timing



For an example of defining float pins, see “Handling Hard Macro Cells” on page 7-52.
To remove the float pin definition from a pin, use the remove_clock_tree_exceptions -float_pins command.

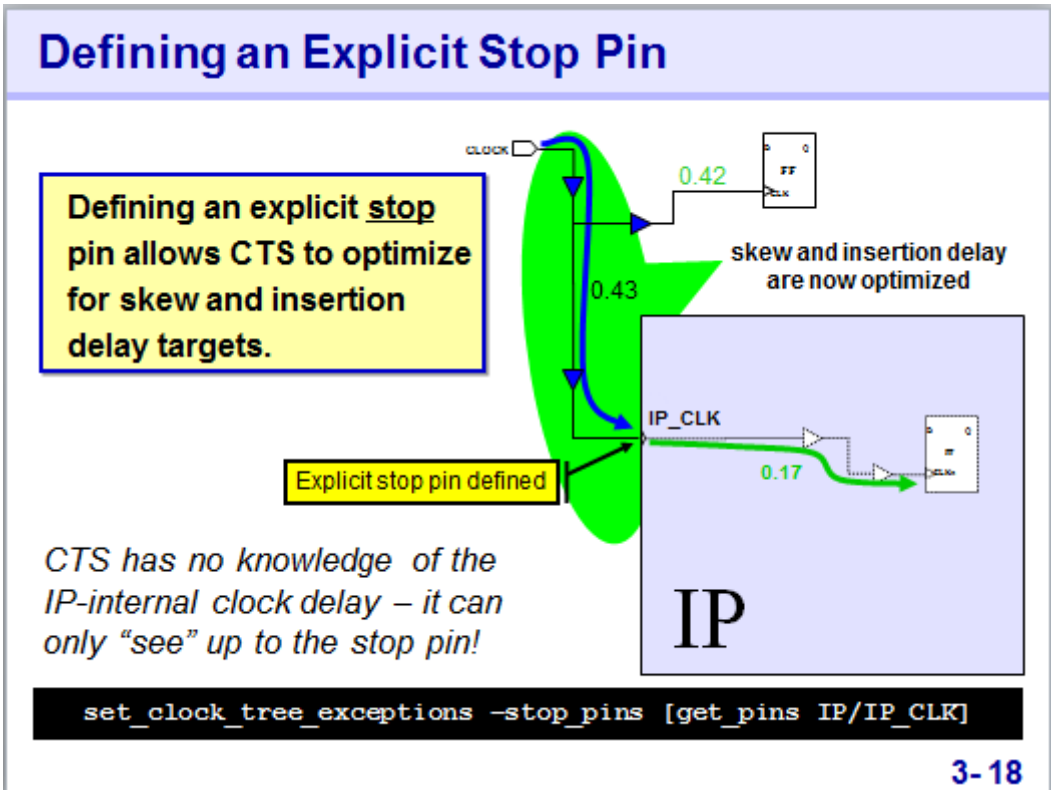
Specifying Stop Pins



Typically the clock port of a macro cell is defined as: a stop pin with correct clock tree delay information.

CTS would use the internal clock tree delay to decide how much more it needs to add outside the macro cell during CTS to optimize for skew and insertion delay with the rest of the stop pins outside the macro cell.
When a clock pin is not defined or defined incorrectly, CTS does not have enough information to optimize it for skew and insertion delay. This forces CTS to assign it an implicit exclude pin where it gets buffered by a single buffer.
Incorrectly defined clock pins (missing or incorrect pin definition in the standard cell or macro cell FRAM view):

- Clock pins without trigger edge info
- Clock pins without a timing arc to the corresponding output pin



In the above example, CTS is doing the right thing – balancing the insertion delays up to all stop pins. CTS doesn't know that internal to the IP, there is a 0.17 delay to the actual clock pins of the flip flops.

Stop pins are the endpoints of the clock tree that are used for delay balancing. During clock tree synthesis, IC Compiler uses stop pins in calculations and optimizations for both design rule constraints and clock tree timing (skew and insertion delay).

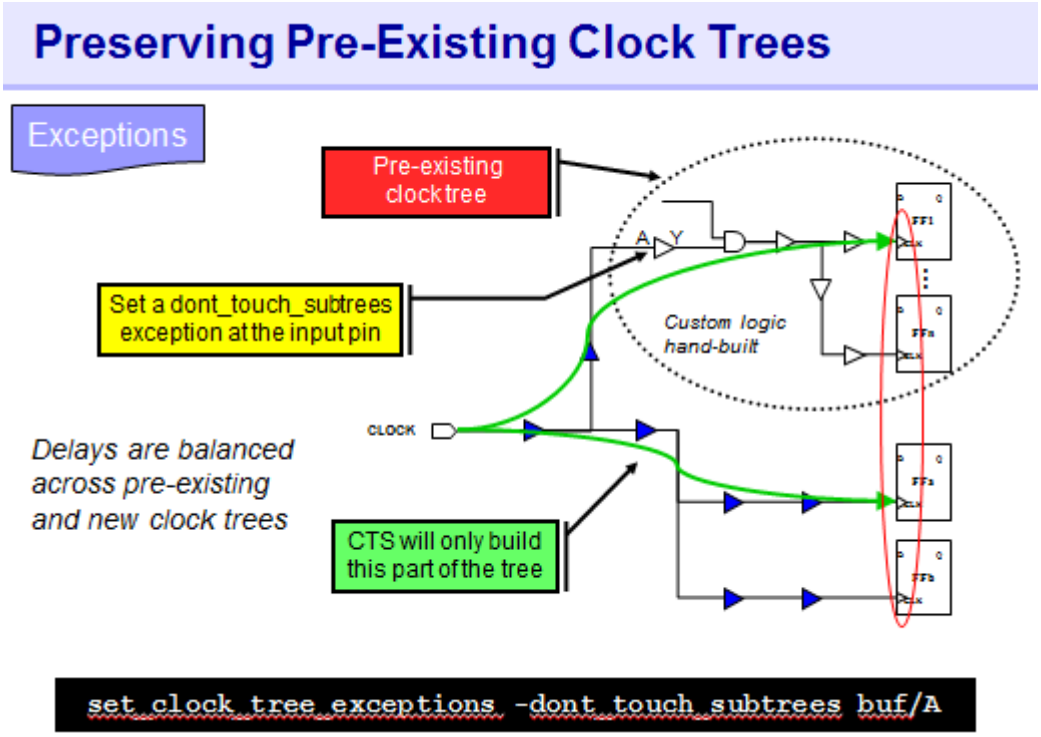
The default clock sinks are implicit stop pins. Implicit STOP (sync) pins are automatically defined by CTS on:
- Clock pins of sequential cells (FFs and latches)
- Clock pins of macros

- **STOP Pins:**
  CTS optimizes for DRC and clock tree targets (skew, insertion delay

In addition, IC Compiler supports user-defined
(or explicit) stop pins. For example, you might define a stop pin to end a branch at an input to a combinational cell or to use an implicit exclude pin as a clock sink.
IC Compiler assigns a phase delay of zero to all stop pins (implicit and explicit) and uses this delay during delay balancing.
To specify a stop pin, use the set_clock_tree_exceptions -stop_pins command.

For example, to specify pin U2/A as a stop pin, enter
icc_shell> set_clock_tree_exceptions -stop_pins [get_pins U2/A]
To remove the stop pin definition from a pin, use the remove_clock_tree_exceptions -stop_pins command.
For example, to remove the stop pin definition from pin U2/A, enter
icc_shell> remove_clock_tree_exceptions -stop_pins [get_pins U2/A]

Specifying Don't Touch Subtrees



Total insertion delay includes the delay of the pre-existing clock tree.
If you don't preserve an existing clock tree, CTS will not rip-out the clock tree and rebuild a new one for you. Instead, it will use whatever is there to meet its goals and may add more logic. This may not always be desirable. Instead, you may want to delete the pre-existing clock tree entirely. To remove the clock tree instead of preserving it, use the command remove_clock_tree .

Impact of Preexisting Clock Cells

- **Any preexisting clock buffers and cells are counted as clock gate levels**
  - Any clock gate level is considered as a balancing point, therefore…
  - Preexisting clock buffers/inverters might create unnecessary clock levels for CTS
- **Use `remove_clock_tree` to remove existing clock buffers**
  - Will generally lead to higher quality clock trees

In some cases you will want to preserve a portion of an existing clock tree. You need to do this, for example, when two clock networks share part of some clock logic behind a multiplexer. The portion of the clock tree that is preserved is called a don't touch subtree.

To specify a don't touch subtree, specify the root pin of the don't touch subtree by using the set_clock_tree_exceptions -dont_touch_subtrees command.
Although IC Compiler does not make any modifications to the don't touch subtree during clock tree synthesis, it does propagate the clock tree attributes and the nondefault routing rules beyond the don't touch subtree. To prevent propagation of the clock attributes and the nondefault routing rules, set the cts_traverse_dont_touch_subtrees variable to false.
IC Compiler considers the sinks in the don't touch subtree when balancing clock delays and computing the clock skew.
To remove the don't touch subtree exception from a pin, use the remove_clock_tree_exceptions -dont_touch_subtrees command.

Test for understanding

What are the two main goals of CTS?

→ Meet DRC (max tran/cap/fanout)
→ Meet the clock tree targets (max skew and min insertion delay)

What is the difference between *stop* and *exclude pins*?

List some examples of implicit stop/exclude pins.

CTS stops at both. CTS optimizes for DRC for both, but only optimizes for clock tree targets to the stop pins.
Implicit stop pins: clock pins of FFs, latches and macro cells.
Implicit exclude pins: non-clock input pins of sequential cells (data, set, reset, etc); clock output ports; clock pins of sequential cells where the output is floating; incorrectly defined clock pins.

How is a float pin different from a stop pin?

A float pin allows the user to define when the clock edge is supposed to arrive at the pin. This can be used to give IP blocks or RAMs extra margin/setup time if needed.

Specifying Don't Buffer Nets

In some cases you might be able to improve the results by preventing IC Compiler from buffering certain nets (IC Compiler still performs global prerouting on these nets).
Note:
During clock tree synthesis, the don't buffer nets exception has priority over the clock tree design rule constraints. However, the clock tree specification in the clock tree configuration file has priority over the don't buffer nets exception.
To specify nets that should not be buffered, use the set_clock_tree_exceptions -dont_buffer_nets command.
For example, to specify net n1 as a don't buffer net, enter
icc_shell> set_clock_tree_exceptions -dont_buffer_nets [get_nets n1]
To remove the don't buffer net exception, use the remove_clock_tree_exceptions -dont_buffer_nets command.
For example, to remove the don't buffer net exception from net n1, enter
icc_shell> remove_clock_tree_exceptions -dont_buffer_nets [get_nets n1]

Specifying Don't Size Cells

During clock tree synthesis and optimization, IC Compiler ignores the dont_touch attribute on cells and nets. To prevent sizing of cells on the clock path during clock tree synthesis and optimization, you must identify the cells as don't size cells.
To specify cells that should not be sized, use the set_clock_tree_exceptions -dont_size_cells command.
For example, to specify cell U1/U3 as a don't size cell, enter
icc_shell> set_clock_tree_exceptions -dont_size_cells [get_cells U1/U3]
To remove the don't size cell exception, use the remove_clock_tree_exceptions -dont_size_cells command.
For example, to remove the don't size cell exception from cell U1/U3, enter
icc_shell> remove_clock_tree_exceptions \
-dont_size_cells [get_cells U1/U3]

Specifying Size-Only Cells

During clock tree synthesis and optimization, size-only cells can only be sized, not moved or split. If a size-only cell overlaps with an adjacent cell after sizing, the size-only cell might be

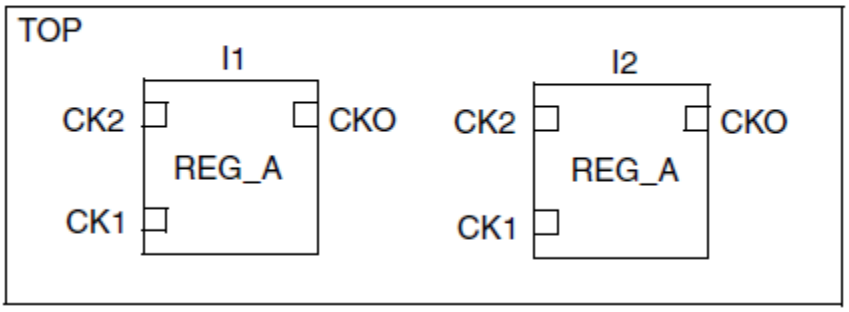
moved during the legalization step. To specify size-only cells, use the set_clock_tree_exceptions -size_only_cells command.
For example, to specify cell U1/U3 as a size-only cell, enter
icc_shell> set_clock_tree_exceptions -size_only_cells [get_cells U1/U3]
To remove the size-only cell exception, use the remove_clock_tree_exceptions -size_only_cells command.

For example, to remove the size-only cell exception from cell U1/U3, enter
icc_shell> remove_clock_tree_exceptions \
-size_only_cells [get_cells U1/U3]

Preserving the Clock Pins of Existing Hierarchies

In some cases, clock tree synthesis and optimization might cluster clock sinks from hierarchies to create new clock pins. You can prevent the clustering of clock sinks in the desired hierarchies that have clock sinks in other logical hierarchies by using the set_clock_tree_exceptions -preserve_hierarchy command, so the clock pins of the desired logical hierarchies are preserved.
For example, suppose your design has two hierarchical block instances I1 and I2 in the top level, and each block has clock pins CK1, CK2, and CKO, as shown in Figure 7-6.

Figure 7-6 Hierarchical Clock Pins



To preserve pin CK1 in cell instance I1, enter
icc_shell> set_clock_tree_exceptions -preserve_hierarchy \
[get_pins I1/CK1]
To preserve pins CK1, CK2, and CKO in cell instance I2, enter
icc_shell> set_clock_tree_exceptions -preserve_hierarchy \
[get_cells I2]
To preserve pins CK1, CK2, and CKO in cell instances I1 and I2, enter
icc_shell> set_clock_tree_exceptions -preserve_hierarchy \
[get_references -hierarchical REG_A]
To remove the preserve-hierarchy exception, use the remove_clock_tree_exceptions -preserve_hierarchy command. If the specified hierarchical pins or cells contain any leaf pins or cells that are not in the clock network, the tool issues a warning message.

If you don't preserve an existing clock tree, CTS will not rip-out the clock tree and rebuild a new one for you. Instead, it will use whatever is there to meet its goals and may add more logic. This may not always be desirable. Instead, you may want to delete the pre-existing clock tree entirely. To remove the clock tree instead of preserving it, use the command remove_clock_tree .

Test for understanding

What are the two main goals of CTS?
→ Meet DRC (max tran/cap/fanout)
→ Meet the clock tree targets (max skew and min insertion delay)

What is the difference between *stop* and *exclude pins*?
List some examples of implicit stop/exclude pins.
→CTS stops at both. CTS optimizes for DRC for both, but only optimizes for clock tree targets to the stop pins.
Implicit stop pins: clock pins of FFs, latches and macro cells.
Implicit exclude pins: non-clock input pins of sequential cells (data, set, reset, etc); clock output ports; clock pins of sequential cells where the output is floating; incorrectly defined clock pins.

How is a float pin different from a stop pin?
→A float pin allows the user to define when the clock edge is supposed to arrive at the pin. This can be used to give IP blocks or RAMs extra margin/setup time if needed.

Specifying the Clock Tree References

Clock specific references are not supported for generated clock.

IC Compiler uses four clock tree reference lists:
- One for clock tree synthesis
- One for boundary cell insertion
- One for sizing
- One for delay insertion

By default, each clock tree reference list contains all the buffers and inverters in your technology library.
To fine-tune the results, you can restrict the set of buffers and inverters used for one or more of these operations. For example, if your clock tree has too many levels, it could be that the clock tree synthesis references have a low drive strength.

To define a clock tree reference list, use the set_clock_tree_references command (or choose Clock > Set Clock Tree References in the GUI). When you define a clock tree reference list, ensure that the buffers and inverters that you specify have a wide range of drive strengths, so that clock tree synthesis can select the appropriate buffer or inverter for each cluster.

Note:
The clock tree synthesis reference list must include at least one inverter, or clock tree synthesis fails. If you are using the default clock tree reference list, you must ensure that your target library contains at least one inverter that does not have a dont_use attribute. If you define a clock tree synthesis reference list, you must ensure that it contains at least one inverter.

Table 7-3 shows the options used for generating the reference lists. If you specify a reference list, but do not select any of these options, the specified reference list is used for clock tree synthesis. If you specify a reference list with one or more of these options, the specified clock tree reference lists are created (the clock tree synthesis reference list is not changed).

Table 7-3 Clock Tree Reference List Options

Reference list type	Option
Clock tree synthesis	N/A
Boundary cell insertion	`-boundary_cell_only` ("Boundary cell only" check box in the GUI)
Sizing	`-sizing_only` ("Sizing only" check box in the GUI)
Delay insertion	`-delay_insertion_only` ("Delay insertion only" check box in the GUI)

When you run the set_clock_tree_references command, IC Compiler verifies that the cells you specify exist in the target libraries, and it generates a warning message if it cannot find a cell.

Note:

For multicorner-multimode designs, IC Compiler checks only the libraries associated with the clock tree synthesis scenario. You need to ensure that the specified clock references exist in the target library specified for the clock tree synthesis scenario.

When you explicitly include a cell in a clock tree reference list, IC Compiler can use the cell for the task associated with the reference list, even if the cell has a dont_use attribute. However, if you set the dont_use attribute on a cell after it is included in a clock tree reference list, IC Compiler honors the dont_use attribute.

IC Compiler uses this clock tree reference list for all clock trees.

If you issue the set_clock_tree_references command multiple times, the new references you specify are added to existing references. References you previously listed but omitted from a later list are not deleted. To delete references, use the reset_clock_tree_references command or choose Clock > Set Clock Tree References in the GUI and click Default.

For example, to create a clock tree synthesis reference list, enter

```
icc_shell> set_clock_tree_references \
-references {clk1a6 clk1a9 clk1a15 clk1a27}
```

IC Compiler uses this clock tree reference list for all clock trees.

Specifying Clock Tree Synthesis Options

NDR Recommendations

- **Always route clock on metal 3 and above**
- **Avoid NDR on clock sinks:**

  `set_clock_tree_options -use_default_routing_for_sinks 1`
- **Avoid NDR on Metal 1**
  - may have trouble accessing metal 1 pins on buffers and gates
- **Put NDR on pitch – try to avoid blind double spacing**
  - Preserve routing resources/keep preroute RC estimation accurate
- **Consider double width to reduce resistance**
- **Consider double via to reduce resistance and improve yield**

To define clock tree synthesis options, use the `set_clock_tree_options` command (or choose Clock > Set Clock Tree Options in the GUI).

Table 7-4 Clock Tree Synthesis Options

Option	Scope	Default	Description
Clock Tree Selection Options			
`-clock_trees` ("Clock tree" field)	N/A	All clock trees	Specifies the clock trees that the options apply to.
Clock Tree Design Rule Constraints (see "Setting Clock Tree Design Rule Constraints" on page 7-29)			
`-max_capacitance` ("Max capacitance" field in the "Target and CTO" tab)	global or per-clock	0.6 pF	Specifies the maximum capacitance constraint.
`-max_fanout` ("Max fanout" field in the "Target and CTO" tab)	global or per-clock	2000	Specifies the maximum fanout constraint.
`-max_transition` ("Max transition" field in the "Target and CTO" tab)	global or per-clock	0.5 ns	Specifies the maximum transition time constraint.

Clock Tree Timing Goals (see "Setting Clock Tree Timing Goals" on page 7-31)

`-target_skew` ("Target skew" field in the "Target and CTO" tab)	global or per-clock	0	Specifies the maximum skew goal.
`-target_early_delay` ("Target early delay" field in the "Target and CTO" tab)	global or per-clock	0	Specifies the minimum insertion delay goal.

Clock Tree Level Restrictions (see "Setting Level Restrictions" on page 7-32)

`-max_buffer_levels` ("Max buffer levels" field in the "Target and CTO" tab)	global or per-clock	20	Specifies the maximum number of clock tree levels.

Clock Tree Routing Options (see "Setting Clock Tree Routing Options" on page 7-32)

`-layer_list` (Layers check boxes in the "Routing" tab)	global or per-clock	All routing layers	Specifies the preferred layers for clock tree routing.
`-routing_rule` ("Routing rule" field in the "Routing" tab)	global or per-clock	Default routing rule	Specifies the nondefault routing rule used for clock tree routing.
`-use_default_routing_for_sinks` ("Use default routing for sinks at level" check box in the "Routing" tab)	global	Specified routing rule for all nets	Specifies that default routing rules are used for the clock tree levels closest to the clock sink.

Boundary Cell Insertion (see "Inserting Boundary Cells" on page 7-38)

`-insert_boundary_cell` ("Insert boundary cell" check box)	global	Disabled	Enables boundary cell insertion.

Clock Tree Clustering (see "Selecting the Clock Tree Clustering" on page 7-39)

`-ocv_clustering` ("OCV clustering" check box)	global	Clustering based on minimum wire length	Selects clustering based on on-chip variation (OCV).

Logic Level Balancing (see "Enabling Logic-Level Balancing" on page 7-40)

`-logic_level_balance` ("Logic level balance" check box)	global	Disabled	Enables logic-level balancing.

Embedded Clock Tree Optimizations
(see "Specifying Clock Tree Optimization Options" on page 7-44)

`-buffer_relocation` ("Buffer relocation" check box)	global or per-clock	Enabled	Enables buffer relocation during optimization.
`-buffer_sizing` ("Buffer sizing" check box)	global or per-clock	Enabled	Enables buffer sizing during optimization.
`-gate_relocation` ("Gate relocation" check box)	global or per-clock	Enabled	Enables gate relocation during optimization.
`-gate_sizing` ("Gate sizing" check box)	global or per-clock	Disabled	Enables gate sizing during optimization.

Clock Tree Configuration File Options (see "Inserting User-Specified Clock Trees" on page 7-47)

`-config_file_read` ("Read file" field in "Configuration files" box)	global	None	Specifies the clock configuration file to read.
`-config_file_write` ("Write file" field in "Configuration files" box)	global	None	Specifies the name of the clock configuration file that is written after clock tree synthesis.

The set_clock_tree_options command is additive; you can run the command multiple times to set different options. If you specify the same option multiple times, the new specification overrides the old specification.

To reset clock tree synthesis options to their default values, use the reset_clock_tree_options command (or choose Clock > Set Clock Tree Options in the GUI and click Default).

To see the current settings for the clock tree synthesis options, use the report_clock_tree -settings command or choose Clock > List Clock Tree Options in the GUI.

Specifying the Clock Tree Synthesis Goals

The optimization goals used for synthesizing the design and the optimization goals used for synthesizing the clock trees might differ. Perform the following steps to ensure that you are using the proper constraints:

• Set the clock tree design rule constraints
• Set the clock tree timing goals

IC Compiler prioritizes the clock tree synthesis optimization goals as follows:

1. Design rule constraints
a. Meet maximum capacitance constraint
b. Meet maximum transition time constraint
c. Meet maximum fanout constraint
2. Clock tree timing goals
a. Meet maximum skew target
b. Meet minimum insertion delay target

Setting Clock Tree Design Rule Constraints

IC Compiler supports the following design rule constraints for clock tree synthesis:

- Maximum capacitance (set_clock_tree_options -max_capacitance)

If you do not specify this constraint, the clock tree synthesis default is 0.6 pF.

- Maximum transition time (set_clock_tree_options -max_transition)

If you do not specify this constraint, the clock tree synthesis default is 0.5 ns.

- Maximum fanout (set_clock_tree_options -max_fanout)

If you do not specify this constraint, the clock tree synthesis default is 2000.

You can specify the clock tree design rule constraints for a specific clock (by using the `-clock_trees` option to specify the clock) or for all clocks (by omitting the `-clock_trees`

option).

Table 7-5 Clock Tree Synthesis Design Rule Constraints

	Variable settings		

Design rule constraint	Default behavior:
	`cts_use_lib_max_fanout =false`
	`cts_use_sdc_max_fanout =false`
	`cts_force_user_ constraints=false`
		Use library and SDC settings for maximum fanout:
		`cts_use_lib_max_fanout =true`
		`cts_use_sdc_max_fanout =true`
		`cts_force_user_ constraints=false`
			Use only clock tree synthesis settings:
			`cts_force_user_ constraints=true`

Maximum capacitance	The minimum value from • The clock tree synthesis value • The logic library • The SDC constraints	The minimum value from • The clock tree synthesis value • The logic library • The SDC constraints	The clock tree synthesis value
Maximum transition time	The minimum value from • The clock tree synthesis value • The logic library • The SDC constraints	The minimum value from • The clock tree synthesis value • The logic library • The SDC constraints	The clock tree synthesis value
Maximum fanout	The clock tree synthesis value	The minimum value from • The logic library • The SDC constraints	The clock tree synthesis value

Note:
IC Compiler does not support the specification of per-clock design rule constraints for overlapping clock domains.

In addition to the clock tree design rule constraint values that you specify, IC Compiler also considers the design rule constraint values from the logic library and the design. Table 7-5 summarizes how IC Compiler determines the design rule constraint values used during the design rule fixing stage of clock tree synthesis and optimization. The clock tree synthesis value refers to the value you specified by using the set_clock_tree_options command (or the clock tree synthesis default if you did not specify a value).

Setting Clock Tree Timing Goals

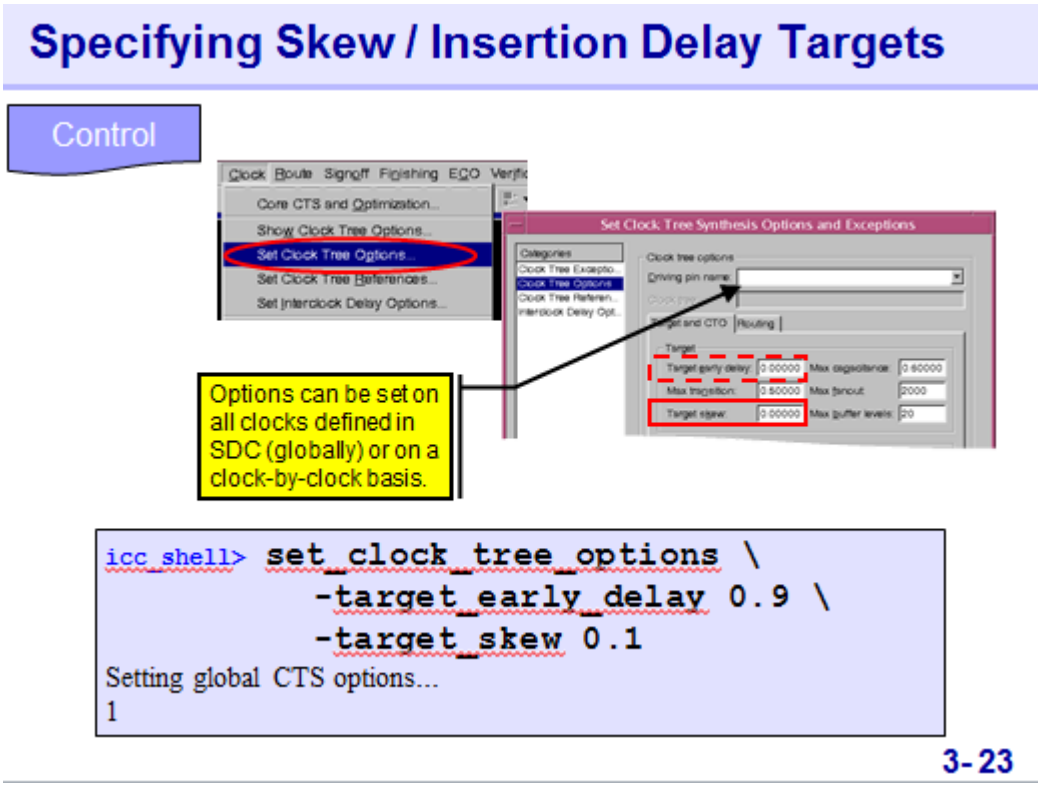
During clock tree synthesis, IC Compiler considers only the clock tree timing goals. It does not consider the latency (as specified by the `set_clock_latency` command) or uncertainty (as specified by the `set_clock_uncertainty` command).

You can specify the following clock tree timing goals for a clock tree:
- Maximum skew (set_clock_tree_options -target_skew)

During optimization, IC Compiler computes the skew value by comparing the arrival times of all clock signals in a clock domain, including those that do not communicate through data paths (global skew).
If you do not specify a maximum skew value, IC Compiler uses 0 as the maximum skew.

- Minimum insertion delay (set_clock_tree_options -target_early_delay)

IC Compiler checks the minimum insertion delay after synthesizing the initial clock tree.
If the synthesized clock tree does not meet the specified minimum insertion delay,
IC Compiler inserts buffers at the clock root to match the requirement.
If you do not specify a minimum insertion delay value, IC Compiler uses 0 as the minimum insertion delay.
You can specify the clock tree timing goals for a specific clock by using the -clock_trees option to specify the clock or for all clocks by omitting the -clock_trees option.



Minimum insertion delay (target early delay) is done by adding a chain of buffer as needed if the insertion of the clock tree is less than the target specified.
The Max buffer levels constraint is not supported!

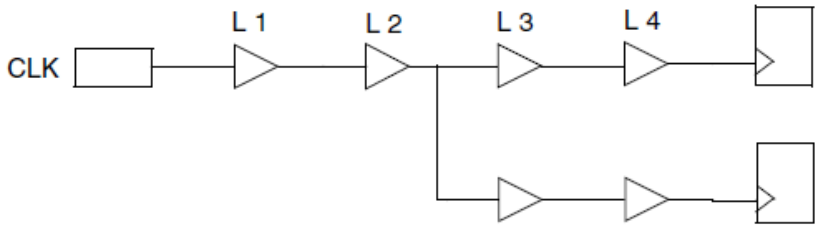
Setting Level Restrictions

By default, IC Compiler allows a maximum of 20 levels in each subtree of a clock tree. If you require a different value, use the `set_clock_tree_options -max_buffer_levels` command to specify the maximum number of levels per subtree.
You can specify the maximum level count for a specific clock (by using the `-clock_trees` option to specify the clock) or for all clocks (by omitting the `-clock_trees` option).

Note:
During clock tree synthesis, the maximum number of levels has priority over the clock tree design rule constraints.

For example, the following command specifies that all subtrees of the clock tree CLK are to have a maximum of four levels as shown in Figure 7-7:
`icc_shell>` **set_clock_tree_options -clock_trees CLK -max_buffer_levels 4**

Figure 7-7 Maximum Clock Buffer Levels



Setting Clock Tree Routing Options

IC Compiler allows you to specify the following options to guide the clock tree routing:

- Which routing rule (type of wire) to use
- Which clock shielding methodology to use
- Which routing layers to use
- Which nondefault routing rules to use with which cell types

Specifying Routing Rules

If you do not specify which routing rule to use for clock tree synthesis, IC Compiler uses the default routing rule (default wires) to route the clock trees. To reduce the wire delays in the clock trees, you can use wide wires instead. Wide wires are represented by nondefault routing rules.

Before you can use a nondefault routing rule, the rule must either exist in the Milkyway design library or have been previously defined by using the define_routing_rule command. For example, to define the clk_rule nondefault routing rule, enter the following command:

```
icc_shell> define_routing_rule clk_rule \
-widths {M1 0.28 M2 0.28 M3 0.28 M4 0.28 M5 0.28 M6 0.28 M7 0.28} \
-spacings {M1 0.28 M2 0.28 M3 0.28 M4 0.28 M5 0.28 M6 0.28 M7 0.28}
```

To see the current routing rule definitions, run the report_routing_rules command.

To set the clock tree routing rule, use the set_clock_tree_options -routing_rule command.

You can specify the clock tree routing rule for a specific clock tree by using the -clock_trees option to specify the clock or for all clocks by omitting the -clock_trees option.

For example, to use the previously defined clk_rule nondefault routing rule for routing all clock trees, enter the following command:

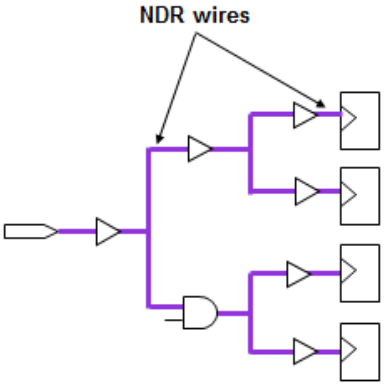
```
icc_shell> set_clock_tree_options -routing_rule clk_rule
```

By default, the specified routing rule is used for all nets in the clock tree. However, wide wires

are often not required on the nets closest to the clock sinks. To use default wires on the nets connected to the clock sinks and the bottom n-1 levels of the clock tree, use the -use_default_routing_for_sinks option on the command line.

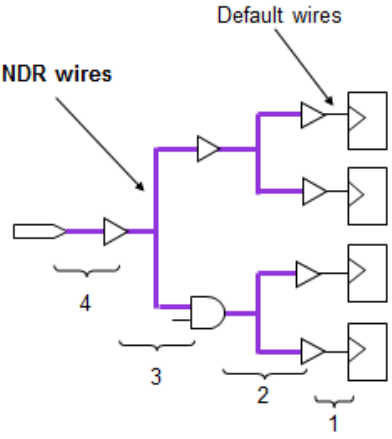
Nondefault Rule Options

```
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!

3-29

NDR Recommendations

- **Always route clock on metal 3 and above**
- **Avoid NDR on clock sinks:**

  `set_clock_tree_options -use_default_routing_for_sinks 1`
- **Avoid NDR on Metal 1**
  - may have trouble accessing metal 1 pins on buffers and gates
- **Put NDR on pitch – try to avoid blind double spacing**
  - Preserve routing resources/keep preroute RC estimation accurate
- **Consider double width to reduce resistance**
- **Consider double via to reduce resistance and improve yield**

3-30

A word on double-spacing: The minimum spacing indicated in the tech file is generally not pitch - width. So if you want double-spaced wires for clocks, you need to use the double pitch (along with width) as your indicator for finding a good spacing value. Wires will end up on pitch – so this improves correlation with

what you have before detail routing.

Note:
If you enable default routing for sinks, it applies to all clock trees. You cannot enable this capability on a per-clock basis. In addition, the default routing applies only to clock sinks connected to flip-flops. Clock sinks connected to macro cells are not affected by this option.

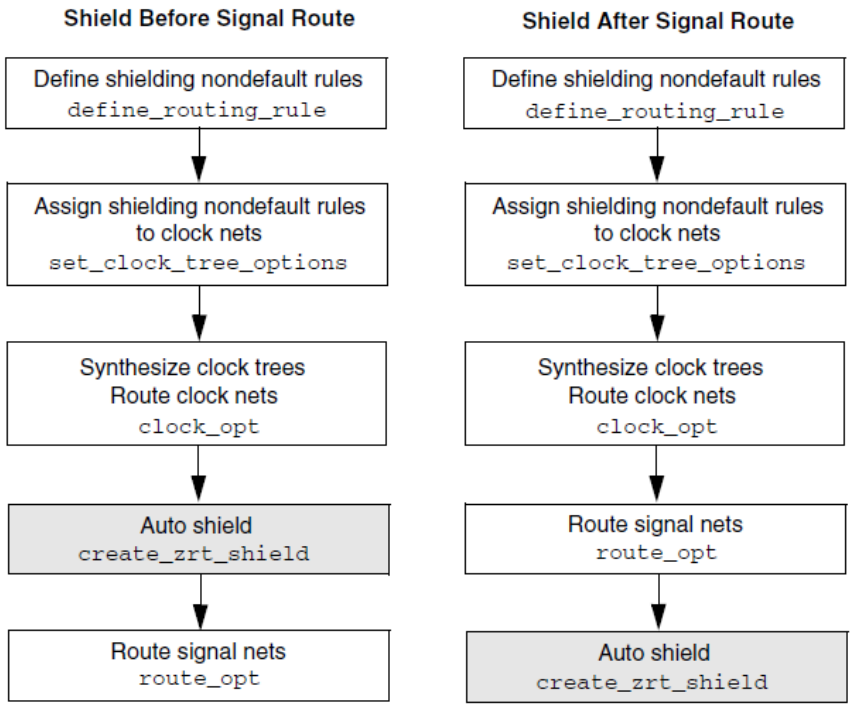
To see the nondefault routing rules defined for the clock trees in your design, run the report_clock_tree -settings command.

Shielding Clock Nets

IC Compiler implements clock shielding using nondefault routing rules. You can choose either to shield clock nets before routing signal nets or vice versa. The methodology of shielding clock nets before routing signal nets yields better shielding coverage but can cause more DRC violations during signal net routing compared to the methodology of routing signal nets before shielding clock nets.

Figure 7-8 shows the flow of both clock shielding methodologies.

Figure 7-8 Clock Shielding Methodologies



To define nondefault routing rules for clock shielding, use the define_routing_rule command.
The syntax is

define_routing_rule rule_name
[-snap_to_track]
[-shield_spacings shield_spacings]
[-shield_widths shield_widths]

To assign nondefault routing rules to clock nets before clock tree synthesis, use the set_clock_tree_options command. For more information about nondefault routing rules, see “Using Nondefault Routing Rules” on page 8-13.
The syntax is
set_clock_tree_options [-clock_tree_name clock_tree_name]
[-root pin_name]
[-routing_rule rule_name]
[-use_default_routing_for_sinks]

The nondefault routing rules of clock shielding apply only to nets that are assigned with nondefault routing rules. You can indicate whether to add shielding to leaf pins by using the -use_default_routing_for_sinks option.
After assigning nondefault routing rules to clock nets, you can synthesize clock trees and route clock nets using the clock_opt command.

IC Compiler router and extractor honor virtual shielding rules. Virtual shielding rules require that the router leaves enough routing resources for shielding to be inserted later and that the extractor considers the shielding effect before shielding metal is inserted. Virtual shielding is supported by virtual routing, global routing, track assignment, and detail routing stages.

To route signal nets, you can use standalone commands such as route_zrt_global, route_zrt_detail, and route_zrt_auto, or the route_opt command. After routing signal nets, you can add shielding to the routed clock nets using the create_zrt_shield command. Alternatively, you can choose to add shielding to the routed clock nets before routing signal nets.

The following sample script shows the methodology for routing signal nets before clock shielding.

```
#Create new nondefault routing rule named SP
define_routing_rule SP \
-widths {M1 0.14 M2 0.14 M3 0.14 M4 0.14 M5 0.14 M6 0.42} \
-spacings {M1 0.14 M2 0.42 M3 0.42 M4 0.42 M5 0.42 M6 1.60} \
-via_cuts {V12 “1x1” V23 “1x1” V34 “1x1’ V45 “1x1” V56 “1x1”}
-shield_widths {M1 0.14 M2 0.14 M3 0.14 M4 0.14 M5 0.14 M6 0.42} \
-shield_spacings {M1 0.14 M2 0.42 M3 0.42 M4 0.42 M5 0.42 M6 1.60}

#Assign nondefault routing rule to clock nets
set_clock_tree_options -clock_tree_name CLK \
-root [get_ports CLK] -routing_rule SP

#Synthesize and route clock trees
clock_opt
set clock_nets [get_nets -of [all_fanout -clock_tree]]
```

```
#Route signal nets
route_opt

#Add shielding to clock nets
create_zrt_shield -nets $clock_nets
```

Specifying Routing Layers

If you do not specify which routing layers to use for clock tree synthesis, IC Compiler can use any routing layers. For more control of the clock tree routing, you can specify preferred routing layers by using the set_clock_tree_options -layer_list command.

You can specify the preferred clock tree routing layers for a specific clock tree by using the -clock_trees option to specify the clock or for all clocks by omitting the -clock_trees option. For example,

```
icc_shell> set_clock_tree_options -clock_trees CLK1 \
-layer_list {metal4 metal5}
```

When you specify the clock tree routing layers by using this command, the specified layers apply to all levels of the clock tree. For finer control of the clock tree routing layers, you can specify the layer constraints in a clock configuration file. For information about the clock configuration file, see “Inserting User-Specified Clock Trees” on page 7-47. Clock tree layer constraints defined in a clock configuration file override those set by using set_clock_tree_options.

By default, IC Compiler treats the minimum layer specification as a soft constraint and can use lower layers for clock tree routing, if necessary. To require clock tree routing on the specified layers, set the following option before running clock tree synthesis:

```
icc_shell> set_route_zrt_common_options -min_layer_mode hard
```

Note:
If you have defined layer constraints on signal nets, you must reset this option to soft before performing detail routing on the design.
To remove the restrictions on the clock tree routing layers, use the reset_clock_tree_options -layer_list command.

Association of Nondefault Routing Rules With Reference Cells

Electromigration problems result from an increase in current densities, which often occurs when strong cells drive thin nets. Electromigration can lead to opens and shorts due to metal ion displacement caused by the flow of electrons and can lead to the functional failure of the IC device. To prevent these problems in clock networks, you can associate reference cells with compatible nondefault routing rules by using the set_reference_cell_routing_rule command.

You use the following syntax to associate reference cells with nondefault routing rules:
set_reference_cell_routing_rule
-routing_rule NDR_name
-references list_of_reference_cells

You must specify both the -routing_rule and -references options. If you use the set_reference_cell_routing_rule command, specifying only the -routing_rule option but not the -references option, the nondefault routing rule does not apply during clock tree synthesis and optimization.

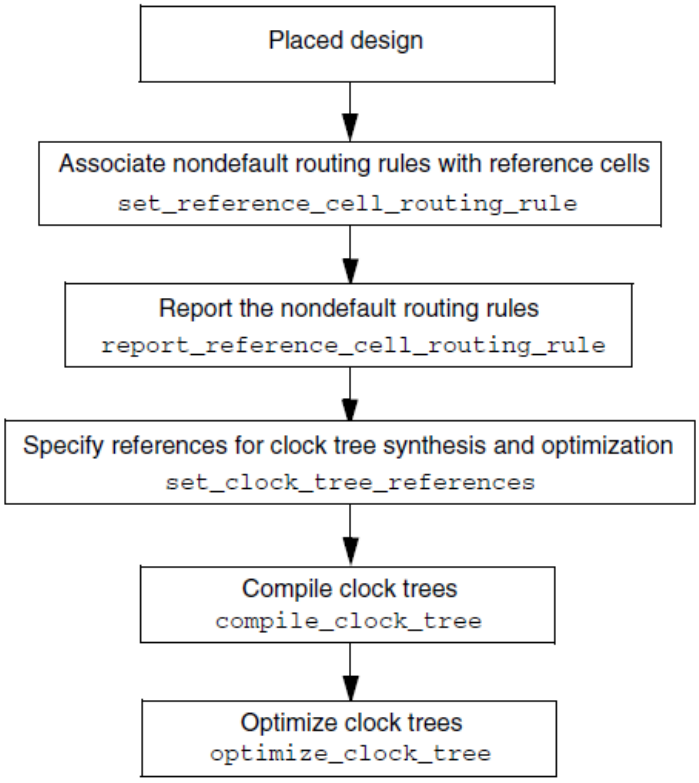
For example, to use the CLOCK_RULE nondefault routing rule for nets that are driven by instances of the BUF, BUF2, and INV1 reference clock cells, enter
icc_shell> set_reference_cell_routing_rule \
-routing_rule CLOCK_RULE -references {BUF1 BUF2 INV1}

When you associate reference cells with nondefault routing rules,
• If multiple nondefault routing rules are associated with a reference cell, clock tree synthesis and optimization uses the nondefault routing rule that provides the best result for each instance of the reference cell.
• The compile_clock_tree command, the optimize_clock_tree command, and interclock delay balancing consider these nondefault routing rules.
• These nondefault routing rules do not apply to nets beyond exception pins.
Note:
The -use_default_routing_for_sinks option of the set_clock_tree_options command is not supported with these nondefault routing rules. If you use both the -use_default_routing_for_sinks option and nondefault routing rules associated with reference cells, the -use_default_routing_for_sinks option is ignored.

To report the nondefault routing rules, use the report_reference_cell_routing_rule command. If you specify the -routing_rule option, the command lists the corresponding reference cells for each specified nondefault routing rule. If you specify the -references option, the command lists the corresponding nondefault routing rules for each specified reference cell.
To reset all nondefault routing rules, use the reset_reference_cell_routing_rule command.

Figure 7-9 shows the flow of associating nondefault routing rules with reference cells.

Figure 7-9 Flow of Associating Nondefault Routing Rules With Reference Cells



Inserting Boundary Cells

When you are working on a block-level design, you might want to preserve the boundary conditions of the block's clock ports (the boundary clock pins). A boundary cell is a fixed buffer that is inserted immediately after the boundary clock pins to preserve the boundary conditions of the clock pin.

To enable boundary cell insertion during clock tree synthesis,
1. Specify the buffers (or inverters) used for boundary cell insertion. (See "Specifying the Clock Tree References" on page 7-22.)
2. Enable boundary cell insertion by using the set_clock_tree_options -insert_boundary_cell true command.

If you enable boundary cell insertion, it applies to all clock trees. You cannot enable boundary cell insertion on a per-clock basis.
Note:
You cannot use boundary cell insertion together with a clock tree configuration file. If you specify both options, IC Compiler disables the boundary cell insertion and generates a warning message.

When boundary cell insertion is enabled, IC Compiler inserts a cell from the buffer insertion clock tree reference list immediately after the boundary clock pins. For multivoltage designs, IC Compiler inserts the boundary cells in the default voltage area.

IC Compiler does not insert a boundary cell when the net is either a don't buffer net or a bidirectional net or when there is a large blockage at the boundary clock pin, which would cause a large distance between the boundary cell and the clock pin.

The boundary cells are fixed for clock tree synthesis; after insertion, IC Compiler does not move or size the boundary cells. In addition, no cells are inserted between a clock pin and its boundary cell.

Selecting the Clock Tree Clustering

IC Compiler performs clustering of the clock sinks to minimize wire length. If your design is sensitive to on-chip variation (OCV), IC Compiler can also consider on-chip variation effects during clustering.

If you are using a multicorner design flow, you can reduce skew variation by using RC constraint-based clustering. To use RC constraint-based clustering, you must use a clock configuration file to specify the clock tree structure. For more information about clock configuration files and RC constraint-based clustering, see "Inserting User-Specified Clock Trees" on page 7-47.

Note:
As of version Z-2007.03-SP3, IC Compiler no longer considers the cts_target_transition and cts_target_cap values during clustering. To control the transition time and maximum capacitance constraints, use set_clock_tree_options to define the clock tree synthesis values for these constraints and set the cts_force_user_constraints variable to true.

Enabling On-Chip-Variation-Aware Clustering

The optional OCV-aware clustering considers the timing constraints between clock sinks to influence clustering. Sinks with timing-critical paths driven by the same gates will be clustered together. To enable the OCV-aware clustering, use the set_clock_tree_options -ocv_clustering true command. When you set this option, it applies to all clock trees in your design.

When you use timing derating, using the set_timing_derate command, OCV-aware clustering can result in better timing (worst negative slack and total negative slack) with minimal impact on the clock tree skew and insertion. However, using OCV-aware clustering can increase the runtime and power.
Note:
You cannot use OCV-aware clustering with clock tree configuration files. If you specify the -ocv_clustering option when these options are set, IC Compiler ignores the -ocv_clustering option. If you specify a clock tree configuration file after setting the -ocv_clustering command, IC Compiler generates an error message and ignores both

settings.

Enabling Logic-Level Balancing

If on-chip variation is an issue for your design, use the logic-level balancing mode.
Note:
If the level count or fanout varies greatly between the branches of the initial clock tree, logic-level balancing might not be able to achieve good clock tree QoR.
By default, IC Compiler balances the delay in each branch of the clock tree, but it does not consider the number of logic levels in each branch. IC Compiler can take into account both delay and the number of logic levels when balancing the clock tree. This feature is called logic-level balancing. Figure 7-10 shows a clock tree that was synthesized using logic-level balancing.

Enabling Region-Aware Clock Tree Synthesis

Region-aware clock tree synthesis considers region constraints to create more balanced clock trees and to avoid DRC violations in designs with complex floorplans. For designs with region constraints, using region-aware clock tree synthesis can produce better QoR. This capability is enabled by default. If you want to disable region-aware clock tree synthesis, set the cts_region_aware variable to false, changing it from its default of true.

Region-aware clock tree synthesis can identify the following region constraints:
- Logic modules with move bounds
- Logic modules with target library subset constraints
- Disjoint voltage areas
- Power guides in power-down regions

During region-aware clock tree synthesis, the tool performs the following steps:
1. Enables the clock_opt or compile_clock_tree command to group buffers in the target library by the same operating condition, power state, and target library subset associated with move bounds.
2. Partitions a design into regions according to constraints such as voltage areas, plan groups, and move bounds: hard and exclusive, as shown in Figure 7-13.
3. Associates each buffer group with a region and vice versa.
4. Associates each power guide with the region that contains it.
5. Performs region-aware clustering.

Specifying Clock Tree Optimization Options

IC Compiler optimizes the clock trees during the design stages that are shown in Table 7-6. During the optimization phases, IC Compiler can perform several optimization tasks, which you can enable or disable by setting the appropriate options.

Table 7-6 Design Stages Using Clock Tree Optimization

Design Stage	Command
Clock tree synthesis	`compile_clock_tree`
Preroute clock tree optimization	`optimize_clock_tree`
Postroute clock tree optimization	`optimize_clock_tree`

Note:
IC Compiler uses the clock tree synthesis design rule constraints for all optimization phases, as well as for clock tree synthesis. For information about setting the clock tree synthesis design rule constraints, see “Setting Clock Tree Design Rule Constraints” on page 7-29.

Controlling Embedded Clock Tree Optimization

To set the optimization options for embedded clock tree optimization, use the set_clock_tree_options command or choose Clock > Set Clock Tree Options in the GUI. You can set these options either for a specific clock by using the -clock_trees option or for all clock trees by omitting the -clock_trees option. Table 7-7 shows the available options and their default values.

Controlling Preroute Clock Tree Optimization

During clock_opt, the clock tree optimization phase performs all the optimization tasks listed in Table 7-8. You cannot independently control these tasks for clock_opt.

To set the optimization options for standalone preroute clock tree optimization, specify the options when you run the optimize_clock_tree command (or choose Clock > Optimize Clock Tree in the GUI). Table 7-8 shows the available options and their default values.

Option	Default	Description
`-buffer_relocation`	`true`	Optimizes the placement of the buffers and inverters in the synthesized clock trees.
`-buffer_sizing`	`true`	Optimizes the sizing of the buffers and inverters in the synthesized clock trees.
`-delay_insertion`	`true`	Inserts delays on clock paths to reduce the clock skew, while at the same time ensuring that the longest clock path does not change.
`-gate_relocation`	`true`	Optimizes the placement of the preexisting gates in the clock trees by moving them closer to the clock sinks. Gates marked as fixed are not moved.
`-gate_sizing`	`true`	Optimizes the sizing of the preexisting gates in the clock trees.

Controlling Postroute Clock Tree Optimization

To set the optimization options for postroute clock tree optimization, specify the options

when you run the optimize_clock_tree command or choose Clock > Optimize Clock Tree in the GUI. Table 7-9 shows the available options and their default values.

Table 7-9 Postroute Clock Tree Optimization Options

Option	Default	Description
-buffer_sizing	true	Optimizes the sizing of the buffers and inverters in the synthesized clock trees.
-gate_sizing	true	Optimizes the sizing of the preexisting gates in the clock trees.

Saving Intermediate Results During Preroute Optimization

You can enable clock_opt checkpointing to analyze your designs during preroute optimization by using the set_checkpoint_strategy -enable command. Checkpointing is disabled by default.

When checkpointing is enabled, the clock_opt command

- Saves design snapshots in the Milkyway database at a periodic interval. You can analyze the intermediate results while the optimization is still proceeding.
- Updates the log file with checkpoint design names.

To analyze your checkpoint designs, you can use timing analysis, extraction, legalization, routing congestion analysis, and multicorner-multimode scenario commands. For information about what commands are allowed, see “Saving Intermediate Results During Preroute Optimization” on page 6-22.

The following example shows how to use the set_checkpoint_strategy command in an optimization flow:

```
icc_shell> set_checkpoint_strategy -enable
icc_shell> clock_opt
```

The checkpoint designs created are

```
clock_opt_checkpoint_MYDESIGN_080908_120402_1
clock_opt_checkpoint_MYDESIGN_080908_120402_2
clock_opt_checkpoint_MYDESIGN_080908_120402_3
...
```

To remove checkpoint designs, use the remove_checkpoint_designs command. For example, to remove all checkpoint designs, enter

```
icc_shell> remove_checkpoint_designs
```

To remove all checkpoint designs created by clock_opt only, enter

```
icc_shell> remove_checkpoint_designs -command clock_opt
```

To disable checkpointing, enter

```
icc_shell> set_checkpoint_strategy -disable
```

Inserting User-Specified Clock Trees

IC Compiler supports the use of a clock configuration file to enable user specification of the clock tree structure. The following sections describe how to

- Read a clock configuration file before clock tree synthesis
- Reduce skew variation by using RC constraint-based clustering
- Save a clock configuration file after clock tree synthesis
- Describe your clock tree structure in a clock configuration file

Reading a Clock Configuration File

If you have a configuration file that describes the desired clock tree structure, run the set_clock_tree_options -config_file_read config_file command before running clock tree synthesis. For details about the syntax of the configuration file, see “Defining the Clock Tree Structure” on page 7-48.

Reducing Skew Variation by Using RC Constraint-Based Clustering

If you use a configuration file to specify the clock tree structure for a multicorner design, you can use RC constraint-based clustering to reduce the skew variation across corners.To enable this capability, set the cts_enable_rc_constraints variable to true before running clock tree synthesis

When you enable this capability, IC Compiler uses the RC values to derive maximum delay and maximum skew constraints, which are then used during clustering to reduce the skew variation. Although RC constraint-based clustering reduces skew variation, you should be aware that it can increase the buffer count and runtime. You can reduce the buffer count and runtime impact by relaxing the derived constraints. To relax the constraints, set the cts_rc_relax_factor variable to a number greater than 1.0. You can also tighten the constraints by setting this variable to a number between 0 and 1.0.

Alternatively, you can use either the -max_rc_delay_constraint option or the -max_rc_scale_factor option of the set_clock_tree_options command to enable RC constraint-based clustering. Note that these two options are mutually exclusive.

To use the -max_rc_delay_constraint option, you specify a maximum RC constraint value in the design time unit. For example, to set the RC constraint value to 0.05 ns if the time unit is ns, enter

```
icc_shell> set_clock_tree_options -max_rc_delay_constraint 0.05
```

To use the -max_rc_scale_factor option, specify a scale factor with which IC Compiler multiplies the derived RC value during clock tree synthesis. For example, to relax the derived RC value by a factor of 1.5, enter,

```
icc_shell> set_clock_tree_options -max_rc_scale_factor 1.5
```

Saving a Clock Configuration File

To save the generated clock tree structure in a clock configuration file after clock tree synthesis, run the set_clock_tree_options -config_file_write config_file command before running clock tree synthesis.

Defining the Clock Tree Structure

Use the following syntax to define the structure for each user-specified clock tree in your design:

```
begin_clock_tree number_of_levels
clock_net net_name [routing_rule rule_name]
[routing_layer_constraints min_layer max_layer]
{buffer_level reference_cell number_of_buffers
[buffer_level_pin instance/pin]
[routing_rule rule_name]
[routing_layer_constraints min_layer max_layer]
...}
...
end_clock_tree
```

begin_clock_tree number_of_levels

This statement starts the clock tree structure definition and specifies the number of levels for the clock tree.
You must specify an integer value for this argument. If you specify 0, IC Compiler determines the number of levels (and the number of buffers in each level) for the clock tree.

clock_net net_name
The net_name argument is a string that specifies the name of the clock net. You can specify the clock net name as a regular expression, for example, clk.* or clk[10].*

routing_rule rule_name
The rule_name argument is a string that specifies the name of the routing rule.
To define a nondefault routing rule that applies to the entire clock tree, insert a routing_rule statement after the clock_net statement that defines the clock root. If you do not define a nondefault routing rule for the root net, IC Compiler uses the default routing rule.
To define a net-specific nondefault routing rule for a net other than the root net, insert a routing_rule statement after the associated clock_net statement.
To define a level-specific nondefault routing rule, insert a routing_rule statement after the associated buffer_level statement.

If you specify a net- or level-specific nondefault routing rule, it overrides the clock tree routing rule (whether it is default or nondefault). If you specify a nondefault clock tree routing rule, you can use the default routing rule on specific nets or levels by specifying routing_rule default for that net or level.

If a routing rule has not been explicitly defined for a net or level, IC Compiler determines the routing rule as follows:

- If the clock tree structure is completely specified, IC Compiler uses the clock tree routing rule (whether it is default or nondefault).
- If the clock tree structure is not completely specified because the number of levels is not specified, the routing rule for the previous level is used.
- If the clock tree structure is not completely specified because the number of buffers for a level is not specified, IC Compiler uses the clock tree routing rule (whether it is default or nondefault).

The clock tree routing rules specified in the configuration file override the routing rules specified by using the set_clock_tree_options command.
For more information about nondefault routing rules, see “Specifying Routing Rules” on page 7-32.

routing_layer_constraints min_layer max_layer
Specifies the minimum and maximum routing layer to use for clock tree synthesis. You can specify the layers using either the layer names (for example, M2 and M4) or the metal layer numbers (for example, 2 and 4).
To define global layer constraints that apply to the entire clock tree, insert a routing_layer_constraints statement after the clock_net statement that defines the clock root. If you do not define layer constraints for the root net, IC Compiler can use any routing layer for clock tree routing.
To define net-specific layer constraints for a net other than the root net, insert a routing_layer_constraints statement after the associated clock_net statement.
To define level-specific layer constraints, insert a routing_layer_constraints statement after the associated buffer_level statement.
If you specify net- or level-specific layer constraints, they override the clock tree layer constraints. If you define clock tree layer constraints, you can remove these constraints for specific nets or levels by specifying routing_layer_constraints 0 0 for that net or level. The clock tree layer constraints specified in the configuration file override the routing rules specified by using the set_clock_tree_options command.

buffer_level reference_cell number_of_buffers
Specifies the reference cell and buffer count for each level in the clock tree (one statement per level). The first buffer_level statement defines the level closest to the clock source; the last buffer_level statement defines the level closest to the clock sink. If you do not define all levels, the specified levels are closest to the clock sink.
The reference_cell argument is a string that specifies the buffer or inverter that is used for all clock tree instances at this level.
The number_of_buffers argument is an integer value that specifies the total number of buffers or inverters at this level. If you specify 0, IC Compiler determines the number of buffers or inverters at this level.
Note:
If the specification in the clock tree configuration file conflicts with a don’t buffer nets exception, the clock tree configuration file has priority.
By default, IC Compiler determines the sink pins connected at each level of the clock tree. For more control, such as when you want a RAM to be placed near the clock root, you

can explicitly specify the sink pins by using the buffer_level_pin statement.

buffer_level_pin instance/pin
Specifies the sink pin connections for the associated buffer level.
You define all user-specified clock trees in a single clock configuration file. For any clock tree that is not specified in the clock configuration file, IC Compiler determines the structure for that clock tree.

Complete Specification Without Sink Pins
The following example completely specifies a clock tree that has six levels. Clock tree synthesis does not change the number of levels or the number of buffers in each level.

```
begin_clock_tree 6
clock_net core/clk
buffer_level bufx6 2
buffer_level invx12 4
buffer_level bufx12 7
buffer_level invx12 11
buffer_level bufx12 48
buffer_level bufx12 268
end_clock_tree
```

Complete Specification With Sink Pins
The following example completely specifies a clock tree that has six levels. The sink pins are specified for the first two levels (closest to the clock root) and last level (closest to the clock sinks) of the clock tree. The first level connects to two multiplexers, the second level connects to the RAM, and all other flip-flops are connected at the last level.

```
begin_clock_tree 6
clock_net core/clk
buffer_level bufx6 2
buffer_level_pin top/inst1/inst2mux/u1/A
buffer_level_pin top/inst1/inst3mux/u2/A
buffer_level invx12 4
buffer_level_pin top/inst1/inst_ram/CKB
buffer_level bufx12 7
buffer_level invx12 11
buffer_level bufx12 48
buffer_level bufx12 268
buffer_level_pin top/inst1/my_reg[5]/CKB
buffer_level_pin top/inst1/my_reg[4]/CKB
...
end_clock_tree
```

Incomplete Specification—Unspecified Levels
The following example defines the bottom four levels of a clock tree but lets IC Compiler determine the structure of the top levels (those closest to the clock root) of the clock tree.

```
begin_clock_tree 0
```

```
clock_net core/clk
buffer_level bufx12 7
buffer_level bufx12 11
buffer_level bufx12 48
buffer_level bufx12 268
end_clock_tree
```

Incomplete Specification—Unspecified Buffer Count

The following example defines all six levels of a clock tree but lets IC Compiler determine the number of buffers in the third level of the clock tree.

```
begin_clock_tree 6
clock_net core/clk
buffer_level bufx6 2
buffer_level invx12 4
buffer_level bufx12 7
buffer_level invx12 0
buffer_level bufx12 48
buffer_level bufx12 268
end_clock_tree
```

Verifying the Clock Trees

Before you synthesize the clock trees, use the check_clock_tree command to verify that the clock trees are properly defined.

```
icc_shell> check_clock_tree -clocks my_clk
```

If you do not specify the -clocks option, IC Compiler checks all clocks in the current design.

The check_clock_tree command checks for the following issues:

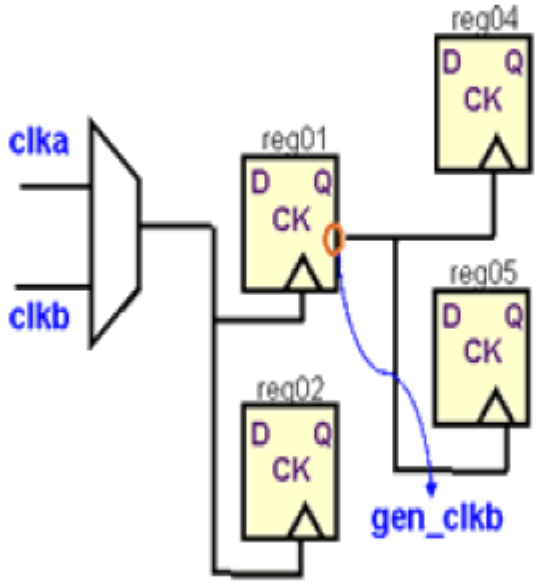
- Hierarchical pin defined as a clock source
- Generated clock without a valid master clock source

A generated clock does not have a valid master clock source in the following situations:

- The master clock specified in create_generated_clock does not exist
- The master clock specified in create_generated_clock does not drive the source pin of the generated clock
- The source pin of the generated clock is driven by multiple clocks, and some of the master clocks are not specified with create_generated_clock.

For example, in Figure 7-15, the reg01/Q pin is driven by both clka and clkb. If only clkb is specified as a master clock in a create_generated_clock command, gen_clkb does not have a valid master clock source.

Figure 7-15 Generated Clock With Invalid Master Clock Source



- Clock (master or generated) with no sinks
- Looping clock
- Cascaded clock with an unsynthesized clock tree in its fanout
- Multiple-clocks-per-register propagation not enabled, but the design contains overlapping clocks
- Ignored clock tree exceptions
- Stop pin or float pin defined on an output pin
- Buffers with multiple timing arcs used in clock tree references
- Situations causing empty buffer list

Each message generated by the check_clock_tree command has a detailed man page that describes how to fix the identified issue. Fixing these issues can improve the clock tree and timing QoR.

To see the current clock tree definition, generate the clock tree settings and exceptions reports, as described in “Generating Clock Tree Reports” on page 7-107.

Handling Specific Design Characteristics

Several design styles might require special considerations during clock tree synthesis. These design styles include

• Hard macro cells
• Preexisting clock trees
• Non-unate gated clocks
• Integrated clock-gating (ICG) cells
• Multiple clocks (with balanced skew)
• Hierarchical designs
• Multivoltage designs
• Multimode designs

The following sections describe how to use IC Compiler clock tree synthesis with these design styles.

Handling Hard Macro Cells

The internal delays for a hard macro cell are represented in the cell's timing model. IC Compiler uses the timing model to determine the external clock pins of the hard macro cell and uses these pins as clock sinks. During clock tree synthesis, IC Compiler performs skew balancing and insertion delay minimization up to these external clock pins. No additional specification is required for hard macro cells

If you do not have a timing model for the hard macro cell or you want to modify the timing characteristics of the hard macro cell, use float pins to specify the timing characteristics of the clock trees internal to the hard macro. You define the timing characteristics by specifying the minimum and maximum insertion delay seen from the float pin to the clock sinks that are internal to the macro.

Handling Existing Clock Trees

If your design contains existing clock trees, you must decide how to handle them before you perform clock tree synthesis. By default, IC Compiler keeps an existing clock tree and might modify it. Instead, you can either prevent IC Compiler from modifying the existing clock tree or you can remove the clock tree.

The following sections describe how to

• Identify clock trees synthesized in Astro or a third-party tool
• Preserve all or part of an existing clock tree
• Remove a clock tree

Identifying Existing Clock Trees

You can identify clock trees in your netlist that were created in Astro or a third-party tool and optimize them in IC Compiler. To identify a clock tree, use the mark_clock_tree -clock_synthesized command (or choose Clock > Mark Clock Tree in the GUI and select "Clock synthesized"). Specify the startpoint for the clock tree by using the -clock_trees option (or the "Clock Tree Pin" field in the GUI). The startpoint can be any port or pin on the clock tree; it does not have to be the clock root. If you do not specify a startpoint, IC Compiler uses the clock roots defined by the create_clock and create_generated_clock

commands.

Preserving Portions of an Existing Clock Tree

In some cases you will want to preserve a portion of an existing clock tree. You need to do this, for example, when two clock networks share part of some clock logic behind a multiplexer. The portion of the clock tree that is preserved is called a don't touch subtree.

Removing Clock Trees

To remove a clock tree, use the remove_clock_tree command (or choose Clock > Remove Clock Tree in the GUI).
For example,
icc_shell> **remove_clock_tree -clock_trees *my_clock***

When IC Compiler removes a clock tree, it traverses from the clock root to each endpoint (stop pin, exclude pin, float pin, or don't touch subtree) and by default, removes all buffers and inverters along those paths, including those with dont_touch attributes. If you want to keep buffers and inverters that have a dont_touch attribute, specify the -honor_dont_touch option when you run remove_clock_tree (or select "Honor don't touch" in the GUI). If you want to remove only those buffers and inverters inserted by IC Compiler, specify the -synopsys_only option when you run remove_clock_tree (or select "Remove only Synopsys CTS added buffers/inverters" in the GUI). In addition, IC Compiler traverses beyond exclude pins, stop pins, and float pins and removes only buffers and inverters inserted by IC Compiler (whether or not you specify -synopsys_only). The dont_touch attributes on buffers and inverters beyond the exception pins are honored only if you specify -honor_dont_touch.

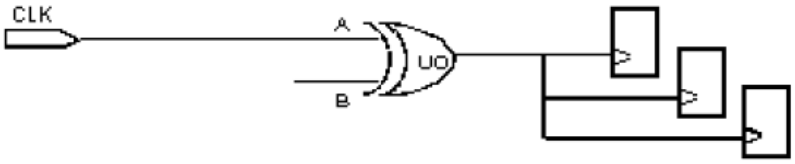
Note:
This command removes only unrouted clock trees. To remove a routed clock tree, you must first remove the clock tree routing.

Handling Non-Unate Gated Clocks

If the clock-gating logic uses a non-unate cell, such as an XOR or XNOR gate, IC Compiler uses both the positive-unate timing arc and the negative-unate timing arc when tracing the clock path. If this does not correspond to the functional mode of your design, use the set_case_analysis command to hold all nonclock inputs of the cell at a constant value. In this way you force the cell into the desired functional mode for timing analysis during clock tree synthesis.

For example, suppose your design has the gating logic shown in Figure 7-14.

Figure 7-14 Non-Unate Gated Clock



To force the XOR gate (U0) into functional mode for timing analysis, enter the following command:
icc_shell> set_case_analysis 0 U0/B

Handling Integrated Clock-Gating Cells

Optimizing for Clock Tree QoR

A balanced clock tree provides the best clock tree timing results. Following these setup recommendations before inserting the integrated clock-gating cells results in a more balanced clock tree:

Invoke CTS-- Implementing the Clock Trees

The recommended process for implementing the clock trees in your design is to use the clock_opt command, which performs clock tree synthesis and incremental physical optimization. This process results in a timing optimized design with fully implemented clock trees.

Note:
Before implementing the clock trees, save your design. This allows you to refine the clock tree synthesis goals and rerun clock tree synthesis with the same starting point, if necessary.

By default, IC Compiler uses the following naming convention for buffers and inverters inserted during clock tree synthesis
reference_GxByIz
where reference is the library reference cell of the buffer or inverter, x is the gate level, y is

the buffer level, and z is the instance count. To more easily locate the inserted buffers and inverters in your netlist, you can add a prefix to the instance names by setting the cts_instance_name_prefix variable. Similarly, you can add a prefix to any nets inserted during clock tree synthesis by setting the cts_net_name_prefix variable.

To perform clock tree synthesis, clock tree optimization, and incremental physical optimization, use the clock_opt command or choose Clock > Core CTS and Optimization in the GUI.

By default, IC Compiler ignores the dont_touch attribute on cells and nets during clock tree synthesis and clock tree optimization. To prevent sizing of cells during clock tree synthesis and clock tree optimization, use the set_clock_tree_exceptions -dont_size_cells command.

By default, the clock_opt command uses virtual routing during clock tree synthesis, but the optimization process uses the integrated clock global router to estimate the wire delay and capacitance. To ensure better postroute correlation, the integrated clock global router saves clock global routing information in the Milkyway database to be used by clock routing.

The `clock_opt` command does the following:

1. (Optional) Performs clock tree power optimization

To perform clock tree power optimization during the clock_opt process, enable physical optimization of the integrated clock-gating cells and power-aware placement, as described in “Optimizing for Power” on page 7-58, and use the -power option of the clock_opt command.

2. Synthesizes the clock trees

Before implementing the clock trees, IC Compiler upsizes, and possible moves, the existing clock gates, which can improve the quality of results (QoR) and reduce the number of clock tree levels.

Note:

To prevent the upsizing of existing clock gates before clustering, set the cts_prects_upsize_gates variable to false. To prevent the moving of existing clock gates before clustering, set the cts_move_clock_gate variable to false.

In addition, IC Compiler might move the existing gates, including integrated clock-gating (ICG) cells, when this could improve QoR. To prevent IC Compiler from moving existing gates, including integrated clock-gating cells, before clustering, set the cts_move_clock_gate variable to false.

IC Compiler builds clock trees that meet the clock tree design rule constraints, while balancing the loads and minimizing the clock skew. In addition, IC Compiler optimizes the clock paths beyond exclude pins, stop pins, and float pins to fix any design rule constraint violations.

Note:

Optimization is not performed on don’t buffer nets or inside interface logic models (ILMs).

By default, the clock sink cells might be moved or sized during the legalization and optimization steps that occur after clock tree synthesis. To prevent any modification to the clock sink cells after clock tree synthesis, set the cts_fix_clock_tree_sinks variable

to true. Note that fixing the clock sinks can impact the timing QoR.
You can also run clock tree synthesis as a standalone process, using the compile_clock_tree command, as described in “Performing Clock Tree Synthesis” on page 7-76.

3. Optimizes the clock trees
During clock tree optimization, IC Compiler uses the optimization techniques, such as buffer relocation, buffer sizing, delay insertion, gate sizing, and gate relocation, to further improve the skew.
Note:
During clock tree optimization, IC Compiler ignores the dont_touch attribute on cells and nets. To prevent sizing of cells during clock tree optimization, use the set_clock_tree_exceptions -dont_size_cells command.
You can also run clock tree optimization as a standalone process, using the optimize_clock_tree command, as described in “Performing Clock Tree Optimization” on page 7-78.

4. (Optional) Performs interclock delay balancing
To perform interclock delay balancing during the clock_opt process, define the interclock delay balancing requirements, as described in “Balancing Multiple Clocks” on page 7-62, and use the -inter_clock_balance option of the clock_opt command.
Note:
IC Compiler performs interclock delay balancing by performing delay insertion at the clock root. If the clock root net has a don’t buffer net exception, IC Compiler cannot perform interclock delay balancing.
If the clock root is defined as a port of a pad cell, the delay insertion is performed on the net driven by the pad cell.
You can also run interclock delay balancing as a standalone process, using the balance_inter_clock_delay command, as described in “Running Interclock Delay Balancing” on page 7-81.
5. Performs detail routing of the clock nets
You can also perform detail routing of the clock nets as a standalone process, using the route_zrt_group -all_clock_nets -reuse_existing_global_route true command.
To prevent routing of the clock nets, use the -no_clock_route option of the clock_opt command.

6. Performs RC extraction of the clock nets and computes accurate clock arrival times

7. (Optional) Adjusts the I/O timing
To adjust the input and output delay based on the actual clock arrival times, use the -update_clock_latency option of the clock_opt command. IC Compiler uses the adjusted input and output delays during placement and timing optimization.
You can also update the I/O timing as a standalone process, using the update_clock_latency command, as described in “Adjusting the I/O Timing” on page 7-81.

8. (Optional) Optimizes the scan chains

To optimize the scan chains by reordering the chains to minimize the number of buffer crossings in the scan chain, use the -optimize_dft option of the clock_opt command.
For more information about scan designs in IC Compiler, see Chapter 4, “Design for Test.”

9. Fixes the placement of the clock tree buffers and inverters

10.Performs placement and timing optimization
If you specify -update_clock_latency, IC Compiler uses the adjusted input and output delays during placement and timing optimization. IC Compiler uses propagated arrival times for all clock sinks.

You can customize the placement and timing optimization process by specifying the following options: -area_recovery, -in_place_size_only, and -size_only. You can perform leakage optimization and -power. For more information about these options, see Table 7-11 on page 7-70.
You can also run only placement and timing optimization as a standalone process, using the psynopt command. For more information about the psynopt command, see “Using Physical Optimization” on page 6-25.
To prevent placement and timing optimization, use the -only_cts option of the clock_opt command.
To run only placement and timing optimization (and not clock tree synthesis, clock tree optimization, or clock tree routing), use the -only_psyn option of the clock_opt command.

11.(Optional) Performs power optimization
You can perform leakage power optimization and dynamic power optimization during the clock_opt process by enabling the selected optimizations with the set_power_options command and using the -power option of the clock_opt command. To enable leakage power optimization, use the set_power_options -leakage command. To enable dynamic power optimization, use the set_power_options -dynamic command.

12.(Optional) Fixes hold time violations
To fix hold time violations during the clock_opt process, use the -fix_hold_all_clocks option of the clock_opt command.

Analyzing Optimization Feasibility After Clock Tree Synthesis

After you finish implementing the clock trees, you can evaluate the design constraints for setup and hold time by analyzing optimization feasibility. You perform the feasibility analysis at an early design stage to fine-tune the design constraints for optimization. To perform the feasibility analysis, you use the clock_opt_feasibility command with the -only_psyn option.

Running optimization feasibility provides the following benefits:

- Improves timing QoR by resolving setup time violations and performing hold time fixing analysis.
- Reduces the runtime relative to the comparable clock_opt flow for optimization.

The following recommended script shows how to analyze optimization feasibility:

```
icc_shell> place_opt -congestion -power -area_recovery
icc_shell> clock_opt -only_cts -no_clock_route \
-inter_clock_balance -update_clock_latency
icc_shell> clock_opt_feasibility -only_psyn -congestion \
-power -area_recovery
```

You can specify any of the options of the clock_opt command when using the clock_opt_feasibility command. To reduce runtime, the clock_opt_feasibility command does not perform congestion removal or clock routing by default. You can optionally run congestion removal and clock routing by specifying the -congestion and -clock_route options respectively. To enable area recovery, specify the -area_recovery option.

Standalone Clock Tree Synthesis Capabilities

Using the clock_opt command is the recommended method for performing clock tree synthesis and optimization with IC Compiler. However, in cases where finer control is required, IC Compiler also provides the following standalone clock tree synthesis capabilities:

- Clock tree power optimization
- Clock tree synthesis
- High-fanout net synthesis
- Clock tree optimization
- Interclock delay balancing
- I/O timing adjustment

The script in Example 7-1 provides an example of performing clock tree synthesis and optimization by using the standalone capabilities. The following sections provide details about these capabilities.

Example 7-1 Clock Tree Synthesis and Optimization Using Standalone Capabilities

```
optimize_pre_cts_power
compile_clock_tree
optimize_clock_tree
balance_inter_clock_delay
route_zrt_group -all_clock_nets -reuse_existing_global_route true
update_clock_latency
set_fix_hold [all_clocks]
psynopt -area_recovery -power
```

Performing Clock Tree Power Optimization

For information about performing clock tree power optimization, see “Optimizing for Power” on page 7-58.

Performing Clock Tree Synthesis

Clock tree synthesis is the process of implementing the clock trees based on your requirements. Clock tree synthesis is performed during the clock_opt process (see “Implementing the Clock Trees” on page 7-69) and can also be run as a standalone process.

IC Compiler clock tree synthesis is blockage-aware by default. The blockage-aware capability avoids routing and placement blockages to reduce DRC violations in designs with complex floorplans. Furthermore, it implements clock trees with minimum clock insertion delay, small clock skew, low buffer count, and small clock cell area to produce the best quality of results (QoR).

During clock tree synthesis, IC Compiler

- Upsizes and moves the existing clock gates, which can improve the QoR and reduce the number of clock tree levels.

Note:
To prevent upsizing of specific cells during this process, use the set_clock_tree_exceptions -dont_size_cells command.

- Inserts buffers and inverters to build clock trees that meet the clock tree design rule constraints, while balancing the loads and minimizing the clock skew.
- Fixes DRC violations beyond clock exceptions if the cts_fix_beyond_clock_exceptions variable is set to true (the default).
- Builds a blockage map infrastructure per voltage area to identify whether a location is blocked for routing or placement, so the legalizer can move buffers to the nearest unblocked locations toward clock sources
- Locates the shortest blockage-avoiding route path from a start point to an end point with minimum delay to prevent DRC violations.

If your design has logical hierarchy, IC Compiler uses the lowest common parent of a buffer’s fanout pins to determine where to insert the buffers.

- If the lowest common parent is not the top level of the design, the buffer is inserted in the lowest common parent.
- If the lowest common parent is the top level of the design, the buffer is inserted in the block that contains the driving pin of the buffer.

IC Compiler adds new ports to the subdesigns where needed. The ports are added such that a minimal number of new ports are added.

To perform standalone clock tree synthesis, use the compile_clock_tree command (or

choose Clock > Compile Clock Tree in the GUI and specify the clock trees in the “Clock tree names” field).
Note:
If you compile one clock at a time, be aware that the order in which you compile the clocks can affect the clock tree QoR. For best results, compile the most critical clock first.

High-Fanout Net Synthesis

You can use the compile_clock_tree command to perform high-fanout net synthesis by using the -high_fanout_net nets_or_driving_pins option (or by choosing Clock > Compile Clock Tree in the GUI and specifying the high-fanout nets in the “High fanout nets” field).
Note:
In a single compile_clock_tree run, you can perform either high-fanout net synthesis (by specifying the -high_fanout_net option) or clock tree synthesis (by specifying no clock names or by specifying the clock trees with the -clock_trees option); you cannot perform both tasks in a single run.

When you use compile_clock_tree -high_fanout_net to perform high-fanout net synthesis, the result is a balanced buffer tree (called a high-fanout tree), which is similar to a clock tree. When you use the create_buffer_tree command to perform high-fanout net synthesis, the resulting buffer tree might not be balanced.
The compile_clock_tree command performs high-fanout net synthesis by balancing the arrival times from the drivers of the nets specified in -high_fanout_net to the fanouts of those nets. High-fanout net synthesis does not traverse through preexisting gates on the high fanout net, nor does it support the use of clock tree exceptions.

Important:
If a clock tree exception exists on any fanout pin of a high-fanout net, high-fanout net synthesis generates an error message and fails. You must remove the clock tree exceptions and rerun high-fanout net synthesis.

By default, high-fanout clock tree synthesis can use any of the buffers or inverters in the library. To restrict the set of buffers or inverters used by high-fanout clock tree synthesis, use the set_clock_tree_references command, as described in “Specifying the Clock Tree References” on page 7-22.
High-fanout clock tree synthesis determines the clock tree design rules in the same way as standard clock tree synthesis. To define the clock tree design rules, use the set_clock_tree_options command, as described in “Setting Clock Tree Design Rule Constraints” on page 7-29.

By default, high-fanout clock tree synthesis uses the rising edge to determine the skew and arrival times. To use the falling edge instead, use the -sync_phase fall option when you

run compile_clock_tree. To use both edges, use the -sync_phase both option when you run compile_clock_tree.
When you perform high-fanout clock tree synthesis, neither the endpoints nor the inserted buffers are fixed after high-fanout net synthesis. This is to allow psynopt to optimize the timing of the high-fanout trees.
To report the skew and path delay of the synthesized high-fanout net, use the report_clock_tree -high_fanout_net pins_or_nets command. You can use the following report_clock_tree options together with the -high_fanout_net option: -summary, -structure, -level_info, -drc_violators, -operating_condition, and -nosplit. All other report_clock_tree options are not supported with -high_fanout_net.
High-fanout clock tree synthesis has the following limitations:

- High-fanout clock tree synthesis does not support multicorner or multimode designs.
- High-fanout clock tree synthesis does not insert level shifters or isolation buffers in multivoltage designs. You must insert the level shifters and isolation buffers before running high-fanout clock tree synthesis.
- You cannot use optimize_clock_tree to optimize the high-fanout trees. Use the psynopt command instead.

Menu: CTS → Core CTS and Optimization

clock_opt use recommendation:
Using clock_opt in the following manner has been found to be more flexible across designs and flows:

clock_opt -only_cts -no_clock_route
analyze…
clock_opt -only_psyn -no_clock_route
analyze…
route_group -all_clock_nets

Effects of Clock Tree Synthesis:

- **Clock buffers added**
- **Congestion may increase**
- **Non clock cells may have been moved to less ideal locations**
- **Can introduce new timing and max tran/cap violations**

Congestion pattern should be similar to the one before CTS. It is, however, generally higher due to the extra clock buffers added during CTS.

If the optimal location for the CTS buffer overlaps with cells placed previously then the previously placed cells need to be moved to open up space for the clock tree buffers.

Incremental Placement / Optimization

- **clock_opt –only_psyn**
  - Reduces disturbances to other cells as much as possible
  - Performs logical and placement optimizations to fix possible timing and max tran/cap violations, based on propagated clock arrivals
- **To enable hold time fixing**

```
set_fix_hold [all_clocks]
clock_opt -only_psyn
```

  - To prioritize TNS over WNS, set:

```
set_fix_hold_options -prioritize_tns
```

  - To prioritize min over max, set:

```
set_fix_hold_options -prioritize_min
```

The alternative to using fix hold attribute is to use –fix_hold_all_clocks option. However, it's recommended to use set_fix_hold so the fix hold attribute is saved to the Milkyway database.

Minimize Hold Time Violations in Scan Paths

clock_opt –only_psyn –optimize_dft

- **Reorders to minimize crossings between clock buffers**
- **Can reduce unnecessary hold time violations in the scan chain**

Recommended Flow:

```
place_opt
# timing ok, congestion ok!
set_clock_tree_options ...
set_clock_tree_exceptions ...
set_clock_tree_references ...
remove_clock_uncertainty [all_clocks]
# OR: adjust uncertainty to contain only jitter, not skew component
clock_opt -only_cts -no_clock_route
#   Analyze
set_fix_hold [all_clocks]
clock_opt -only_psyn -optimize_dft -no_clock_route
```

clock_opt:

- Builds the clock trees
- Performs incremental logic and placement optimizations
- Runs clock tree optimizations
- Routes the clock nets

Optionally, clock_opt can

- Fix hold time violations
- Perform inter-clock balancing

Analysis using the CTS GUI

- **CTS browser**
  - Properties and attributes on clock tree objects
  - Traversing clock tree levels
  - Symbols for CTS objects like buffers, gates and sinks
- **CTS schematic**
  - Trace forward/backward in schematic view
  - Collapses all sinks in the fanout of a CTS buffer for clearer CTS schematic
  - Highlight CTS objects in the layout view
- **Clock arrival histogram**

- **CTS → New Clock Tree Synthesis Window**

Analyzing CTS Results

- **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, interclock latency etc. for paths that are related.
  - Example: **report_clock_timing –type skew**

Clock Tree Optimization

Gui: clock > optimize clock tree

CT optimization is run inside clock_opt, and can be run independently as well:
optimize_clock_tree

optimize_clock_tree -clock_trees {CLK} -buffer_sizing -buffer_relocation -delay_insertion -search_repair_loop 2 -operating_condition max -routed_clock_stage None

Gate/Buffer Sizing
Sizes up or down buffers/gates to improve both skew and insertion delay.

Will not modify the clock tree structure during this optimization process.

Buffer Relocation
Physical location of the buffer is moved to reduce skew and insertion delay.

Will not add or remove hierarchy during optimization process.

Delay Insertion
Delay is inserted for shortest paths.

Inter-Clock Delay Balancing

1. **set_inter_clock_delay_options **
   -balance_group "CLOCK1 CLOCK2"

clock_opt -inter_clock_balance
Performs CTS and balances the clocks.

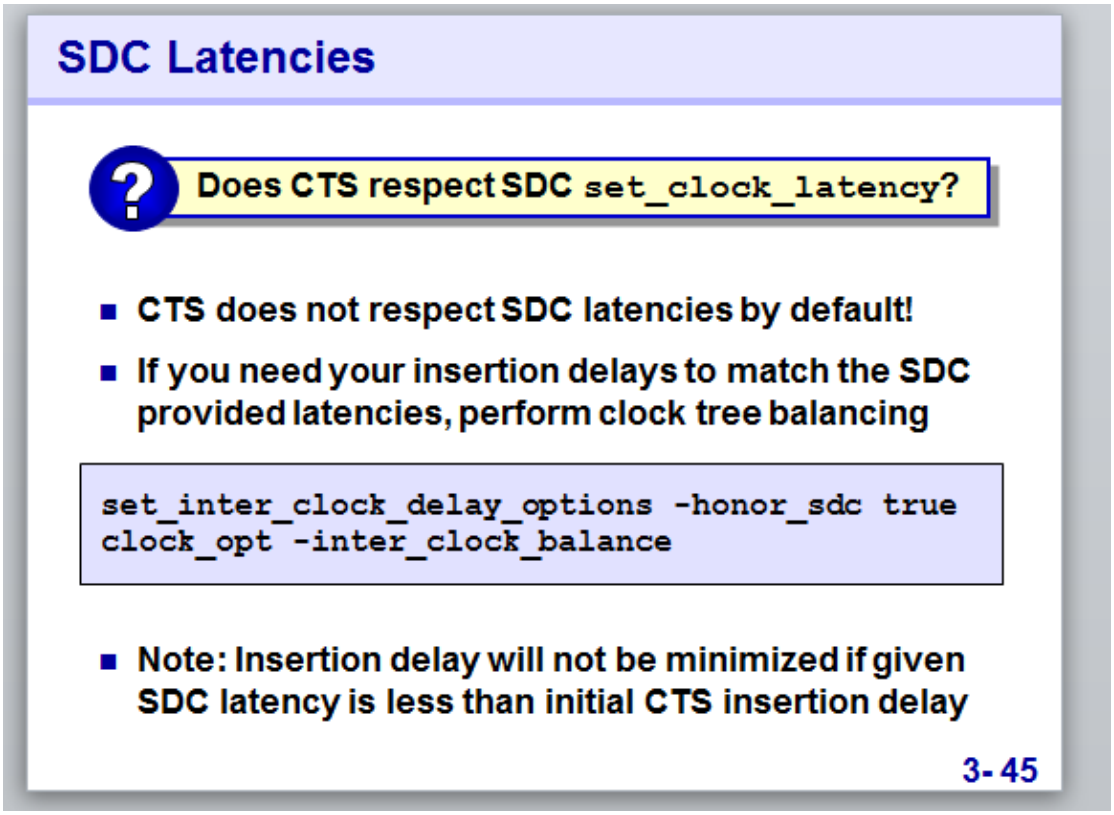
2. Clock tree balancing can also be performed stand-alone after CTS using the command:
balance_inter_clock_delay -clock_trees "CLOCK1 CLOCK2"

Inter-Clock Delay Balancing with Offset

```
set_inter_clock_delay_options \
   -offset_from CLOCK1 -offset_to CLOCK2 \
   -delay_offset 0.2
clock_opt -inter_clock_balance
# or stand-alone after CTS:
# balance_inter_clock_delay -clock_trees "CLOCK1 CLOCK2"
```

Clock trees can be balanced using an offset requirement, as shown above, or using targets, using the syntax below or through SDC set_clock_latency commands.
The following command performs inter clock delay balancing using the maximum target delay options:
balance_inter_clock_delay -max_target_delay 0.5
The following command performs inter clock delay balance for the CLK1 clock tree to 0.5ns and CLK2 to 0.8ns:
set_inter_clock_delay_options -target_delay_clock CLK1 -target_delay_value 0.5
set_inter_clock_delay_options -target_delay_clock CLK2 -target_delay_value 0.8
balance_inter_clock_delay

To honor the latencies defined in the SDC file, use
set_inter_clock_delay_options -honor_sdc true



Fixing DRC Beyond Exception Pins

- **To fix DRC violations beyond exception pins to meet clock tree DRCs,**
  compile_clock_tree -fix_drc_beyond_exceptions

- **To report DRC violations beyond exceptions,**
  report_clock_tree –all_drc_violators

```
clock_opt -only_cts -no_clock_route
report_clock_tree -all_drc_violators
compile_clock_tree -fix_drc_beyond_exceptions
clock_opt -only_psyn -no_clock_route
```

When running the full clock_opt, drc beyond exceptions is fixed by the embedded timing optimization (psynopt) to meet the logical/library DRCs and not clock tree DRCs.
If you want clock tree DRC fixing on your excluded pins, AND you want to use clock_opt, then you need to execute the following commands:
clock_opt -only_cts -no_clock_route
compile_clock_tree -fix_drc_beyond_exceptions
route_group -all_clock_nets

clock_opt –only_cts: leaves DRCs beyond exclude pins untouched.
compile_clock_tree -fix_drc_beyond_exceptions: fixes all nets beyond exclude pins to meet clock tree DRCs. It also maintains the ct-net attribute on the nets involved so route_net –all_clock_nets will route those nets with the specially defined non-default routing rule.

Flow Using Atomic Commands

```
place_opt
# timing ok, congestion ok!
set_clock_tree_options ...
set_clock_tree_exceptions ...
set_clock_tree_references ...
define_routing_rule my_route_rule …
set_clock_tree_options \
  -routing_rule my_route_rule \
  -use_default_routing_for_sinks 1
compile_clock_tree –clock_trees "CLOCK1 CLOCK2"
compile_clock_tree –fix_drc_beyond_exceptions
optimize_clock_tree
remove_clock_uncertainty [all_clocks]
# OR: adjust uncertainty to contain only jitter, not skew component
set_fix_hold [all_clocks]
psynopt –area_recovery
optimize_clock_tree
route_group -all_clock_nets
```

Test for Understanding

1. **CTS tries to:**
   a) Minimize skew only
   b) Minimize skew and insertion delay
   c) Minimize skew and maximize insertion delay
   d) Minimize skew and meet minimum insertion delay target
2. **How do you set a clock skew target of 0.1 for clk1, and a minimum insertion delay of 0.7 for clk2? What is the skew target for clk2?**
   set_clock_tree_options -root clk1 -target_skew 0.1
   set_clock_tree_options -root clk2 \

-target_early_delay 0.7

If no skew target was defined for clk2, the default is 0.

3 Write the command(s) to balance the two clock trees clk1 and clk2, so clk2 arrives 0.3 earlier!

set_inter_clock_delay_options -offset_from clk2 \
-offset_to clk1 -delay_offset 0.3
clock_opt –inter_clock_balance

OR after CTS: balance_inter_clock_delay -clock_trees "clk1 clk2"

4 Why is it important to remove or adjust the clock uncertainty *before* executing clock_opt?

Uncertainty should be removed before clock_opt in case post-cts optimizations are done as well. Otherwise, the optimizations will take a very pessimistic skew into account (actual clock skew + uncertainty)!

place_opt with CTS Technology

- **Enables CTS and CTO inside of place_opt**

"place_opt –cts" Flow:

- Coarse placement
- Optimization
- CTS
- CTO
- Optimization
- Clock routing

CTS in place_opt Options

set_place_opt_cts_strategy
-operating_conditions {min|max|min_max}
-no_clock_route
-inter_clock_balance
-fix_hold
-update_clock_latency
place_opt -cts
Most options are same as clock_opt

-operating_condition {min|max|min_max}
Specifies the operating condition. Default is max.
-no_clock_route
Do not perform routing of clock nets. By default, the clock nets are routed and extracted.
-inter_clock_balance
Perform inter clock delay balancing after CTC and CTO.
-optimize_clock_gr
Perform global routing of clock nets after CTC and CTO.

-fix_hold
Perform hold time violation fixing
-update_clock_latency
Update real and virtual clock object latencies.
-remove_uncertainty
Remove clock uncertainty before post CTS optimizations.

Recommendations for place_opt –cts

- **Recommended designs:**
  - Simple clock tree structure
  - Legacy design methodology
    - Example: no MCMM
  - Designs with same SDC for placement and CTS stages
- **Not recommended designs**
  - Complex clock tree structure
  - Designs using detail scripts for CTS

Unit Objectives Summary

You should now be able to:

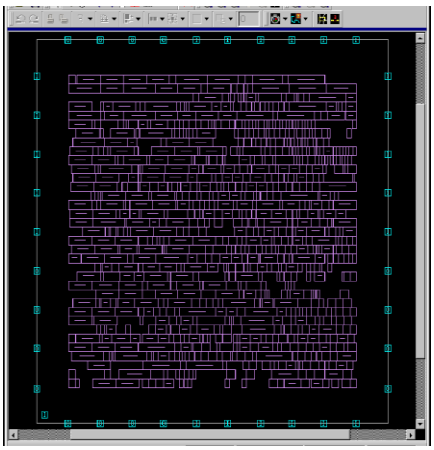
- **List the status of the design prior to CTS**
- **Set up the design for clock tree synthesis**
- **Identify implicit clock tree start/end points and when explicit modifications are needed**
- **Control the constraints and targets used by CTS**
- **Execute the recommended clock tree synthesis and optimization flow**
- **Analyze timing and clock specifications post CTS**

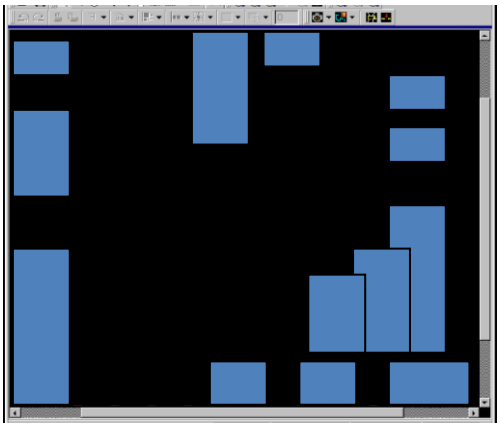
CTS Modes

- **Block (default)**
  - Proven good results across many designs
- **Top**
  - Uses same clustering function as block mode on lower level clock tree (bottom-up)
  - Switches to top-mode specific algorithm at upper level of clock tree (top-down)
  - Creates global grid cells to determine buffer/inverter pair locations. CTS-specific global grid cells are macro and blockage aware
- **Logic Level Balance**

The top and logic level balance modes are selected using the set_clock_tree_options command.

Which design is better suited for Top mode?

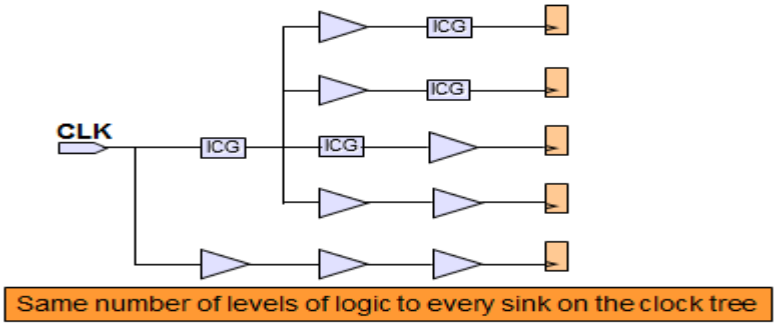




Top mode will yield better results, especially for insertion delay, on designs with many macros, which require unusual clock routes.

Logic-Level Balanced CTS

- **Creates a logic level balanced clock structure**
  - Same number of levels to each sink pin
  - A level can be a gate, or a clock cell (buffers or inverters)



ICG = Integrated Clock Gating cell

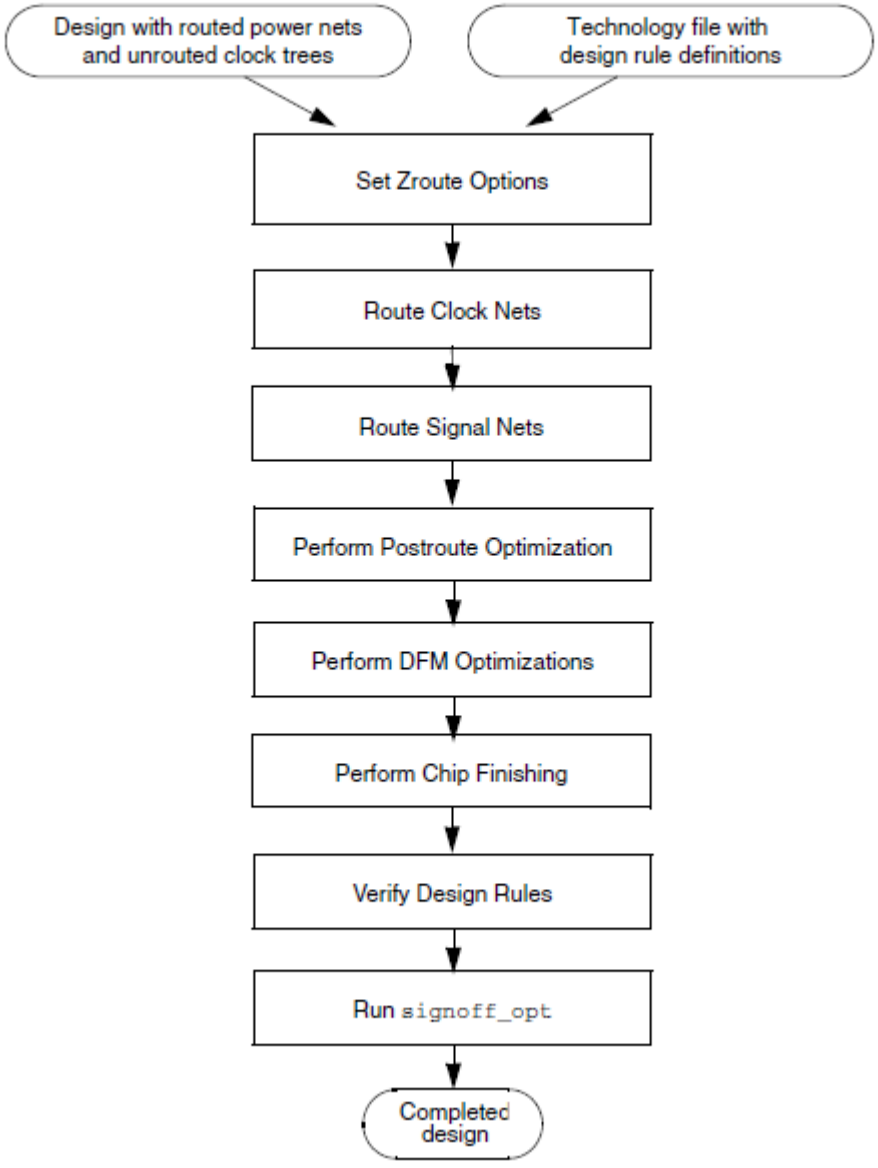
Comparison of CTS Modes

- **Each mode satisfies different QoR requirements**
  - Removing redundant preexisting clock buffers produced better QoR, and on top of this…

	Top mode	Block mode	Logic-level balancing
Skew	Close to logic level balancing (within 0.6%)	Comparable to top mode (slightly worse)	Best
Insertion delay	Best	Comparable to top mode (slightly worse)	High
Clock power/ Buffer area	Comparable to block mode	Best	High

Zroute

Figure 8-1 Basic Zroute Flow



Prerequisites for Routing

Checking Routability

After placement is completed, you can have IC Compiler check whether your design is ready for detail routing. The tool checks pin access points, cell instance wire tracks, pins out of boundaries, minimum grid and pin design rules, and blockages to make sure they meet design requirements. It creates an error file named after the top-level design (*top_design_name*.err), with a list of violations that you should correct before performing detail routing.

To verify that your design is ready for detail routing, use the check_routeability command (or choose Route > Check Routability in the GUI).

After you run the check_routeability command, you can use the report_error_coordinates command to report the location of each error. You can also use the error browser to examine the errors in the GUI. For more information about the error browser, see “Examining Routing and Verification Errors” on page A-53 and the “Examining Routing and Verification Errors” topic in IC Compiler Help.

Setting Up for Routing

You can specify general routing setups for Zroute to use whenever you perform routing. The following sections describe how to specify these setups:

- Enabling Multicore Processing
- Creating Route Guides
- Setting the Preferred Routing Direction
- Controlling Pin Connections
- Using Nondefault Routing Rules
- Specifying the Routing Layers
- Setting Zroute Options
- Setting Signal Integrity Options
- Setting Multivoltage Options

Enabling Multicore Processing

You can use multicore processing to reduce the elapsed time for routing, crosstalk reduction during the route_opt command, and postroute optimization.

The following routing-related tasks support multithreading:

- Routing

Zroute is designed as a multithreaded router. During multithreaded routing, Zroute performs routing tasks in parallel. Each of these tasks is performed by a separate thread. When you run routing with a single thread, the result is deterministic; if you start with the same design, you will always get the same result. However, if you use multiple threads, the routing result are not deterministic; the final routing will be slightly different between runs due to the varying division of tasks between threads.

- Signal integrity analysis during the route_opt -only_xtalk_reduction command
- Route verification using IC Validator with the signoff_drc command

The following routing-related tasks support distributed processing:

- Postroute optimization using the route_opt, psynopt, or focal_opt command
- Route verification using IC Validator with the signoff_drc command

You use the set_host_options command to configure multicore processing for both multithreading and distributed processing. For more information about using the set_host_options command to configure multicore processing, see “Enabling Multicore Processing” on page 2-54.

Creating Route Guides

Setting the Preferred Routing Direction

Use the set_preferred_routing_direction command to reset and override the default preferred routing direction specified in the library or the design for a specific layer. The layer direction set with this command applies to the current design only.

Controlling Pin Connections

Setting Zroute Options

You can specify global routing options, track assignment options, detail routing options, and common routing options (options that affect all three routing engines). Zroute uses these settings whenever you perform routing functions. When you run a routing command, Zroute writes the settings for any routing options that you have set (or that the tool has set for you) in the routing log. To display the settings for all routing options, not only those that have been set, set the verbose_level common Zroute option to 1.

icc_shell> **set_route_zrt_common_options -verbose_level 1**

Setting Common Route Options

Common route options are used to define routing options that affect global routing, track assignment, and detail routing. When you save the design in Milkyway format, the common route option settings are saved with the design.

- To set the common route options, use the set_route_zrt_common_options command

(or choose Route > Routing Setup > Set Common Route Options in the GUI). To reset the options to their default values, use set_route_zrt_common_options -default true (or click Default in the GUI).
• To report the settings of all common route options, use the report_route_zrt_common_options command.
• To return the value of a specific common route option, use the get_route_zrt_common_options command.

Setting Global Route Options

Global route options are used to define routing options that affect global routing. When you save the design in Milkyway format, the global route option settings are saved with the design.
• To set the global route options, use the set_route_zrt_global_options command (or choose Route > Routing Setup > Set Global Route Options in the GUI). To reset the options to their default values, use set_route_zrt_global_options -default true (or click Default in the GUI).
• To report the settings of all global route options, use the report_route_zrt_global_options command.
• To return the value of a specific global route option, use the get_route_zrt_global_options command.

Setting Track Assignment Options

Track assignment options are used to define routing options that affect track assignment. When you save the design in Milkyway format, the track assignment option settings are saved with the design.
• To set the track assignment options, use the set_route_zrt_track_options command (or choose Route > Routing Setup > Set Track Assign Options in the GUI). To reset the options to their default values, use set_route_zrt_track_options -default true (or click Default in the GUI).
• To report the settings of all track assignment options, use the report_route_zrt_track_options command.
• To return the value of a specific track assignment option, use the get_route_zrt_track_options command.

Setting Detail Route Options

Detail route options are used to define routing options that affect detail routing. When you save the design in Milkyway format, the detail route option settings are saved with the design.
• To set the detail route options, use the set_route_zrt_detail_options command (or

choose Route > Routing Setup > Set Detail Route Options in the GUI). To reset the options to their default values, use set_route_zrt_detail_options -default true (or click Default in the GUI).

- To report the settings of all detail route options, use the report_route_zrt_detail_options command.
- To return the value of a specific detail route option, use the get_route_zrt_detail_option command.

Routing Clock Nets

You can use Zroute for initial routing of clock nets, redundant via insertion on clock nets, shielding of clock nets, and ECO routing of clock nets. The following sections describe how to perform these tasks.

Note:

When you use Zroute for clock tree routing, you must use Arnoldi delay calculation for the clock nets. To enable Arnoldi delay calculation for the clock nets, use the following command:

icc_shell> **set_delay_calculation -clock_arnoldi**

Clock Net Routing

You can route clock nets before routing the rest of the nets in the design by using the route_zrt_group -all_clock_nets command.

icc_shell> **route_zrt_group -all_clock_nets**

Note:

If you have an IC Compiler-PC package, use the route_zrt_clock_tree command to route the clock nets; the route_zrt_group command is not included in the IC Compiler-PC package.

The route_zrt_group -all_clock_nets command runs global routing, track assignment, and detail routing on the specified nets. By default, existing global routes are ignored. To perform incremental global routing by reusing existing global routes, use the -reuse_existing_global_route true option. During detail routing, Zroute also performs search and repair. By default, the detail router performs a maximum of 40 iterations. Zroute stops before completing the maximum number of iterations if it determines that all violations have been fixed or that it cannot fix the remaining violations. You can control the maximum number of detail routing iterations by using the -max_detail_route_iterations option.

Zroute supports three modes for global routing of clock nets:

- Balanced

To use balanced mode global routing, you must perform incremental global routing, which reuses the global route information left by the integrated clock global router during clock tree optimization.

icc_shell> **set_delay_calculation -clock_arnoldi**

icc_shell> **clock_opt -only_cts -no_clock_route**

icc_shell> **route_zrt_group -all_clock_nets **
-reuse_existing_global_route true

• Comb
To use comb mode global routing, set the clock_topology global route option to comb before routing the clock nets.
icc_shell> **set_delay_calculation -clock_arnoldi**
icc_shell> **clock_opt -only_cts -no_clock_route**
icc_shell> **set_route_zrt_global_options -clock_topology comb**
icc_shell> **route_zrt_group -all_clock_nets**

• Normal
To use normal mode global routing, set the clock_topology global route option to normal before routing the clock nets. Note that this is the default setting for this option.
icc_shell> **set_delay_calculation -clock_arnoldi**
icc_shell> **clock_opt -only_cts -no_clock_route**
icc_shell> **set_route_zrt_global_options -clock_topology normal**
icc_shell> **route_zrt_group -all_clock_nets**

Clock Net Redundant Via Insertion

Zroute can insert redundant vias on clock nets either during or after routing. This section describes how to insert redundant vias during clock routing. For information about postroute redundant via insertion, see “Inserting Redundant Vias” on page 9-23.
To insert redundant vias on clock nets during clock routing,
1. Define the redundant vias in a nondefault routing rule by using the -via_cuts option of the define_routing_rule command.
icc_shell> **define_routing_rule clock_via_rule **
-via_cuts {V12 "1x2" V23 "1x2" V34 "1x2" V45 "1x2" V56 "1x2"}
For information about nondefault routing rules, see “Using Nondefault Routing Rules” on page 8-13.
2. Assign the nondefault routing rule to the clock nets by using the -routing_rule option of the set_clock_tree_options command.
icc_shell> **set_clock_tree_options -routing_rule clock_via_rule**
3. Run clock tree synthesis.
icc_shell> **set_delay_calculation -clock_arnoldi**
icc_shell> **clock_opt -only_cts -no_clock_route**
4. Route the clock nets.
icc_shell> **route_zrt_group -all_clock_nets **
-reuse_existing_global_route true

Clock Net Shielding

Zroute uses nondefault routing rules to define the shielding width and spacing. For information about nondefault routing rules, see “Using Nondefault Routing Rules” on page 8-13.

Note:
If you do not define the shielding spacing for clock nets, Zroute adds default spacing as shield spacing on clock nets.
After defining the nondefault routing rules and routing the clock nets, use the create_zrt_shield command to shield the clock nets. For information about shielding nets, see “Shielding Nets” on page 9-35.

Postroute Clock Tree Optimization

When Zroute is enabled and you run the optimize_clock_tree command on a routed design, the optimize_clock_tree command uses Zroute to perform ECO routing.
If only the clock nets are routed, run the following command to perform postroute clock tree optimization and ECO routing of the clock nets:

```
icc_shell> optimize_clock_tree -routed_clock_stage detail \
-buffer_sizng -gate_sizing
```

If both the clock and signal nets are routed, run the following commands to perform postroute clock tree optimization and ECO routing of the clock and signal nets:

```
icc_shell> optimize_clock_tree \
-routed_clock_stage detail_with_signal_routes \
-buffer_sizing -gate_sizing
```

Routing Signal Nets

Before you route the signal nets, all clock nets must be routed without violations.
You can route the signal nets by using one of the following methods:
• Use the basic commands to perform the standalone routing tasks.
To perform global routing, use the route_zrt_global command. To perform track assignment, use the route_zrt_track command. To perform detail routing, use the route_zrt_detail command.

When you run a standalone routing command, such as route_zrt_global or route_zrt_detail, Zroute reads in the design database at the beginning of each routing command and updates the database at the end of each command. The router does not check the input data. For example, if the track assignment step is skipped and you run detail routing directly, Zroute might generate bad routing results.

• Use automatic routing (the route_zrt_auto basic command).
The route_zrt_auto basic command performs global routing, track assignment, and detail routing.
When you run route_zrt_auto, Zroute reads the design database before starting routing and updates the database when all routing steps are done. If you stop automatic routing before it performs detail routing, Zroute checks the input data when you restart

routing with this command.
Use the route_zrt_auto command when you run routing to verify convergence, congestion, and design rule quality-of-results (QoR). You also might want to use route_zrt_auto if congestion QoR is your main goal, rather than timing QoR.
• Use the route_opt core command.
The route_opt command performs global routing, track assignment, detail routing, and postroute optimization.
Use the route_opt command when you are finished with the feasibility runs and want to finalize the routing and perform postroute optimization.
Zroute can insert redundant vias during signal routing. For information about this capability, see “Inserting Redundant Vias” on page 9-23.

Routing Signal Nets by Using the route_opt Command

Before you use the route_opt command to route the signal nets, you must define the Zroute options (common, global route, track assignment, and detail route), as described in “Setting Zroute Options” on page 8-23 and set the route_opt routing and optimization strategy, as described in the following section. If logical DRC violations remain after running the route_opt command, you can use focal optimization to address these remaining violations, as described in “Performing Focal Optimization” on page 8-65.

Setting the Routing and Optimization Strategy

IC Compiler provides strategy controls that you set to guide how routing and postroute optimizations are run.
To set the routing and optimization strategy, use the set_route_opt_strategy command (or choose Route > Set Core Routing and Optimization Strategy in the GUI).

Setting the Crosstalk Reduction Options

By default, when you run the route_opt command with the -xtalk_reduction or -only_xtalk_reduction option, Zroute performs crosstalk reduction during detail routing on nets with setup violations and static noise violations.
You can use the set_route_opt_zrt_crosstalk_options command to control net selection for crosstalk reduction based on setup violations, hold violations, transition time violations, and static noise violations. The following sections describe these controls.
To report your settings, use the report_route_opt_zrt_crosstalk_options command.
To get the value of a specific setting, use the get_route_opt_zrt_crosstalk_options command

Running the route_opt Command

To run routing and postroute optimization, use the route_opt command (or choose Route > Core Routing and Optimization in the GUI).
By default, the route_opt command performs the following tasks:
• Global routing
By default, global routing is timing-driven and does not do crosstalk prevention.
To disable timing-driven global routing, use the set_route_zrt_global_options -timing_driven false command (or choose Route > Routing Setup > Set Global Route Options in the GUI and deselect “Timing driven” in the Run Control tab).
To enable crosstalk-prevention mode, use the set_route_zrt_global_options -crosstalk_driven true command (or choose Route > Routing Setup > Set Global Route Options in the GUI and select “Crosstalk driven”) and use the -xtalk_reduction or -only_xtalk_reduction option when you run the route_opt command.
• Track assignment
By default, track assignment is timing-driven and does not do crosstalk prevention.
To disable timing-driven track assignment, use the set_route_zrt_track_options -timing_driven false command (or choose Route > Routing Setup > Set Track Assign Options in the GUI and deselect “Timing driven”).
To enable crosstalk-prevention mode, use the set_route_zrt_track_options -crosstalk_driven true command (or choose Route > Routing Setup > Set Track Assign Options in the GUI and select “Crosstalk driven”) and use the -xtalk_reduction or -only_xtalk_reduction option when you run the route_opt command.

• Detail routing
By default, detail routing is timing-driven and does not do crosstalk reduction.
To disable timing-driven detail routing, use the set_route_zrt_detail_options -timing_driven false command (or choose Route > Routing Setup > Set Detail Route Options in the GUI and deselect “Timing driven” in the Run Control tab).
To enable crosstalk-reduction mode, use the -xtalk_reduction or -only_xtalk_reduction option when you run the route_opt command.

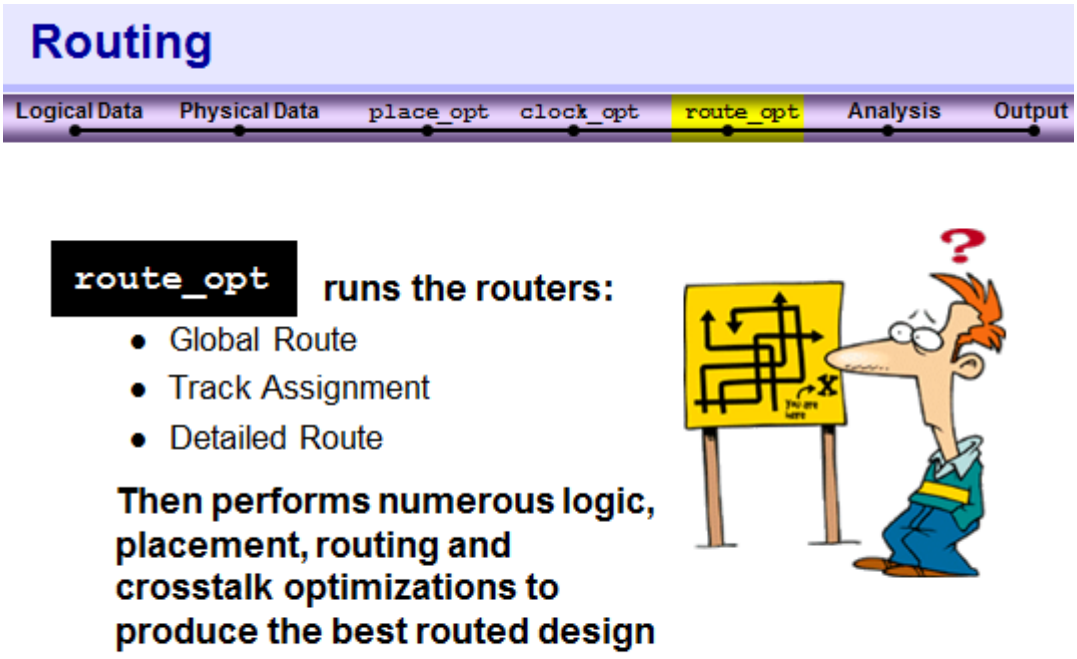
• Postroute optimization
In addition to performing routing optimization, you can perform the following optimizations: signal integrity, leakage power, and area recovery.
By default, the postroute optimization step generates QoR reports before and after the postroute optimization. If you do not need the QoR reports that are generated after postroute optimization, you can reduce runtime by setting the routeopt_skip_report_qor variable to true. Setting this variable to true prevents generation of QoR reports after postroute optimization; this variable does not affect the generation of QoR reports before postroute optimization.
To enable signal integrity optimization, set the signal integrity options as described in “Setting Signal Integrity Options” on page 8-39, and use the -xtalk_reduction option when you run the route_opt command.
To enable power-aware postroute optimization, use the set_route_opt_strategy -power_aware_optimization true command. By default, when power-aware postroute optimization is enabled, leakage power has a lower cost priority than setup, hold, and design rule constraints. You can change these cost priorities by setting the

following variables to true: routeopt_leakage_over_setup, routeopt_leakage_over_hold, and routeopt_leakage_over_drc.

To enable leakage power optimization and recovery, use the -power option when you run the route_opt command. For information about leakage power optimization, see Chapter 5, “Power Optimization.”

To enable area recovery, use the -area_recovery option when you run the route_opt command.

10 route_opt



Prerequisites for Routing

Before you can perform routing, your design must meet the following conditions:

- Power and ground nets have been routed after design planning and before placement.

For more information, see the IC Compiler Design Planning User Guide.

- Clock tree synthesis and optimization have been performed.

For more information, see the clock tree synthesis chapter in the IC Compiler Implementation User Guide.

- Estimated congestion is acceptable.
- Estimated timing is acceptable (about 0 ns of slack).
- Estimated maximum capacitance and transition have no violations.

To verify that your design meets the last three prerequisites, you can check the routability of its placement as explained in the following section, “Checking Routability.”

Checking Routability

After placement is completed, you can have IC Compiler check whether your design is ready for detail routing. The tool checks pin access points, cell instance wire tracks, pins out of boundaries, minimum grid and pin design rules, and blockages to make sure they meet design requirements. It creates an error file named after the top cell in your design (top_cell_name.err), with a list of violations that you should correct before performing detail

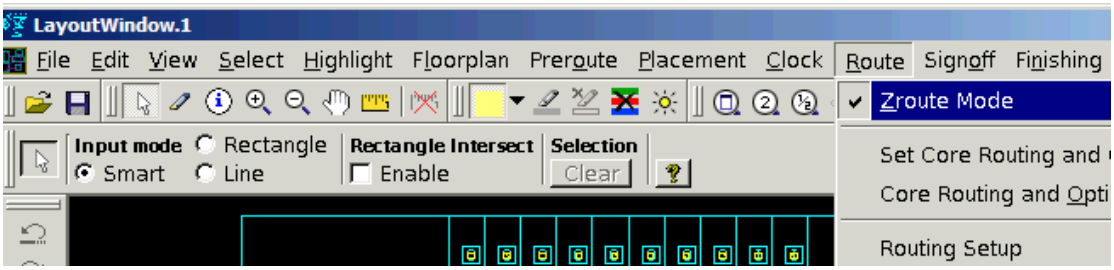
routing.

To verify that your design is ready for detail routing, use the check_routeability command (or choose Route > Check Routability in the GUI).

Setting Up for Routing

Enabling the Classic Router

To enable the classic router, set the Zroute route mode option to false. The setting for the Zroute mode option is saved in the Milkyway design database.
If you are using the command line, enter the following command:
icc_shell> set_route_mode_options -zroute false
If you are using the GUI, deselect Zroute Mode in either the Route or Finishing menu, as shown in Figure 2-1.



Note:
If you are using the IC Compiler-XP package, the classic router is the default router. You do not need to enable it explicitly.
When you enable the classic router, the following commands use the classic router instead of Zroute:

- estimate_fp_area
- report_congestion
- place_opt -congestion (when the placer_enable_enhanced_router variable is true)
- create_placement -congestion (when the placer_enable_enhanced_router variable is true)
- clock_opt
- route_opt
- optimize_clock_tree (when run on a postroute design)
- signoff_opt

Specifying Route Guides

You can create or remove route guides that do the following for a specific area of your design:

- Prevent routing for signal or prerouted nets, either completely or partially
- Change the wiring direction

- Control wiring density
- Fix violations

To create a route guide, use the create_route_guide command (or choose Floorplan > Create Route Guide in the GUI). Table 2-1 defines the create_route_guide options. For more information about these options, see the man page.

Defining Routing Blockages

A routing blockage defines a region where no routing is allowed on a specific layer. You define routing blockages by using the create_routing_blockage command. To define a routing blockage, you must specify

- The blockage layers to which the routing blockage applies

Use the -layers option to specify the blockage layers. Valid values for the blockage layers are metal1Blockage-metal15Blockage, via1Blockage-via14Blockage, polyBlockage, and polyContBlockage. To query the Milkyway design library for the blockage layers, use the get_layers -include_system command.

You can specify one or more blockage layers. If the routing blockage is defined on a metal blockage layer, it is a metal routing blockage. If the routing blockage is defined on a via blockage layer, it is a via routing blockage.

- The region of the blockage

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.

For example, to create rectangular routing blockages on the metal1 and via1 blockage layers, enter the following command:

```
icc_shell> create_routing_blockage \
-layers {metal1Blockage via1Blockage} -bbox {x1 y1 x2 y2}
```

When IC Compiler creates routing blockages, it assigns a name of RB_id to each routing blockage.

To find routing blockages, use the get_routing_blockages command. For example, to get all the routing blockages in your design, enter

```
icc_shell> get_routing_blockages *
```

Setting the Preferred Routing Direction

Use the set_preferred_routing_direction command to reset and override the default preferred routing direction specified in the library or the design for a specific layer. The layer direction set with this command applies to the current design only.

The syntax is

```
set_preferred_routing_direction
-layers list_of_layers
-direction horizontal | vertical
```

For example, to set the preferred routing direction to vertical for layer M5 and M7, enter

```
icc_shell> set_preferred_routing_direction -layers "M5 M7" \
```

-direction vertical
Note:
Settings with the create_route_guide -switch_preferred_direction command, which changes the preferred direction within the area that is covered by the route guide, will override the settings made with the set_preferred_routing_direction command.
Use the report_preferred_direction command to report the preferred routing direction for all the routing layers. The report lists the library and user-defined routing directions, as well as the direction that the tool will use. Example 2-1 shows a sample report.

Using Nondefault Routing Rules

The classic router supports the use of nondefault routing rules, both for routing and for shielding. Before you can use a nondefault routing rule, you must define it. The following sections describe how to define and apply nondefault routing rules.

Defining Nondefault Routing Rules

You define nondefault routing rules for specific nets by using the define_routing_rule command (or by choosing Route > Routing Setup > Define Routing Rule in the GUI). These rules define wire width and spacing rules and via types. Table 2-2 defines the define_routing_rule options. For more information about these options, see the man page.

For example, to define a nondefault routing rule named new_rule that uses the default routing rule as the reference rule and defines nondefault width and spacing rules for metal1 and metal4, enter the following command:

```
icc_shell> define_routing_rule new_rule -default_reference_rule \
-widths {m1 0.8 m4 0.9} -spacings {m1 1.0 m4 1.0}
```

Applying Nondefault Routing Rules

IC Compiler provides two commands for assigning nondefault routing rules to nets:

- set_clock_tree_options

This command assigns nondefault routing rules to clock nets before clock tree synthesis.

- set_net_routing_rule

This command assigns nondefault routing rules to signal nets and to clock nets after clock tree synthesis.

Applying Nondefault Routing Rules to Clock Nets

To assign a nondefault routing rule to clock nets, use the -routing_rule option of the set_clock_tree_options command.
Note:
This command works only before clock tree synthesis. If the clock tree has already been synthesized, use the set_net_routing_rule command to apply the nondefault routing rules, as described in “Applying Nondefault Routing Rules to Signal Nets” on page 2-10.

For example, to assign a shielding rule called shield_rule to the nets in the CLK clock tree before clock tree synthesis, enter the following command:

```
icc_shell> set_clock_tree_options -clock_tree CLK \
-routing_rule shield_rule
```

By default, the shielding rule applies to all nets in the clock tree. To use the default routing rule for the nets closest to the sink pin, use the -use_default_routing_for_sinks option. The default routing rules are used for the specified number of clock tree levels closest to the clock sink. For more information about the set_clock_tree_options command, see Chapter 7, “Clock Tree Synthesis,” in the IC Compiler Implementation User Guide.

Applying Nondefault Routing Rules to Signal Nets

To apply a nondefault routing rule to one or more nets, use the set_net_routing_rule command (or choose Route > Routing Setup > Set Net Routing Rule in the GUI). Table 2-3 defines the set_net_routing_rule options. For more information about these options, see the man page

Reporting Nondefault Routing Rules

To report the nondefault routing rules for specific nets, use the report_net_routing_rules command.

```
icc_shell> report_net_routing_rules [get_nets *]
```

To output a Tcl script that contains the define_routing_rule commands used to define the nondefault routing rules for the specified nets, use the -output option when you run the report_net_routing_rules command.
The name of the nondefault routing rule for a net is stored in the var_route_rule net attribute. You can access the value of this attribute by using the get_attribute command.

To report the design-specific nondefault routing rules defined by the define_routing_rule command, use the report_routing_rules command. By default, this command reports all of the nondefault routing rules for the current design. To limit the report to a specific nondefault routing rule, specify the rule name as an argument to the command.

```
icc_shell> report_routing_rules rule_name
```

To output a Tcl script that contains the define_routing_rule commands used to define

the specified nondefault routing rule or all nondefault routing rules for the design if you do not specify a routing rule, use the -output option when you run the report_routing_rules command.

Specifying the Routing Layers

IC Compiler lets you specify which layers can be ignored for routing and which layers are to be ignored for RC and congestion estimation. You can also specify the routing layers to use for specific nets (net layer constraints).
By default, when you set a maximum routing layer, it is a hard constraint. You can change it to a soft constraint or you can allow the use of higher layers only for pin connections by setting the hardMaxLayerConx detail route option. This option applies to both ignored layers (set_ignored_layers command) and net layer constraints (set_net_routing_layer_constraints command).

By default, when you set a minimum routing layer, it is a soft constraint. You can change it to a hard constraint or you can allow the use of lower layers only for pin connections by setting the hardMinLayerConx detail route option. This option applies to both ignored layers (set_ignored_layers command) and net layer constraints (set_net_routing_layer_constraints command).

Specifying the Ignored Layers

To specify the ignored layers, use the set_ignored_layers command (or choose Route > Routing Setup > Set Ignored Layers in the GUI). By default, RC estimation and congestion analysis use the same layers as routing.

To specify the minimum and maximum routing layers, use the -min_routing_layer and -max_routing layer options, respectively. If you use only these options, the same layers are used for routing, RC estimation, and congestion analysis. For example, to use layers M2 through M7 for routing, RC estimation, and congestion analysis, use the following command:

```
icc_shell> set_ignored_layers \
-min_routing_layer M2 -max_routing_layer M7
```

Setting Signal Integrity Options

To improve signal integrity, you can enable crosstalk prevention during global routing and track assignment and enable crosstalk fixing during postroute optimization. To perform these signal integrity tasks when you run the route_opt command, you must use the -xtalk_reduction option.
For more information about the IC Compiler signal integrity features, see Chapter 10, “Signal Integrity,” in the IC Compiler Implementation User Guide.

Enabling Crosstalk Prevention

By default, the classic router does not perform crosstalk reduction. If you enable signal integrity mode, the classic router performs crosstalk reduction during global routing and track assignment. To enable signal integrity mode, enter the following command:
icc_shell> set_si_options -route_xtalk_preventation true

The default crosstalk prevention threshold is 0.35 volts, which is probably too relaxed. If your design has crosstalk violations, you should use the set_si_options -route_xtalk_prevention_threshold command to lower the crosstalk prevention threshold during track assignment to a value between in the range of 0.25 to 0.35 volts. When you set the crosstalk prevention threshold, IC Compiler automatically sets the threshold_noise_ratio common route option.
When you enable crosstalk prevention, the classic router avoids putting long, parallel wires on adjacent tracks during track assignment. To minimize noise, track assignment estimates the potential noise with a simplified crosstalk checker and reassigns wires to reduce the potential noise using the noise threshold.

Enabling Crosstalk Fixing

To minimize crosstalk-induced noise, you can specify the aggressors for a victim net. Crosstalk prevention during postroute optimization uses that information to minimize the crosstalk-induced noise from the aggressor nets.
To specify aggressors for a victim net, use the set_net_aggressors command (or choose Route > Routing Setup > Set Net Aggressors in the GUI). To specify the victim net, use the -victim_net option (or the “Victim net” box in the GUI). To specify the aggressor nets, use the -aggressor_nets option (or the “Aggressor nets” box in the GUI).

Crosstalk *Prevention* in IC Compiler

- **During placement and optimization :**
  - set_max_transition
  - set_congestion_options …
  - area_recovery_critical_range and power_recovery_critical_range
    - Recommendation: 15% of main clock period
- **During CTS:**
  - Use of Non-Default Routing rules – e.g. double-spacing

```
define_routing_rule my_route_rule -spacings {…}
set_clock_tree_options [-clock CLK]   \
    -routing_rule my_route_rule
```

- **During Global Route and Track Assign :**

```
set_si_options -delta_delay true \
        -route_xtalk_prevention true
```

6-32

Non-default rules can be specified for width, spacing, shielding, min/max layer, and special via cut with the following commands.

- define_routing_rule
- set_net_routing_rule
- set_net_routing_layer_constraints
- set_clock_tree_options –routing_rule
- preroute_standard_cells –use_special_via_rule

e.g. a non-default routing rule can also be applied to critical nets using:
set_net_routing_rule -rule MYRULE [get_nets CRITICAL*]

Non-default rules can be specified before place_opt, clock_opt or route_opt commands because they are honored by each engine. They are stored in the Milkyway database. Users can report NDRs by using:

- report_routing_rules
- report_net_routing_rules

Or remove them by using:

- remove_routing_rules

Example Full Crosstalk Flow

Example Full Crosstalk Flow

```
place_opt
set_clock_tree_options -max_transition 0.2 -max_fanout 32
set_clock_tree_references -reference $CLK_BUFFER_LIST
define_routing_rule MYRULE -spacings {M2 0.28 M3 0.28…}
set_clock_tree_options -routing_rule MYRULE
clock_opt

set_si_options -route_xtalk_prevention true \
       -route_xtalk_prevention_threshold  0.35 \
       -delta_delay true \
       -static_noise true \
       -static_noise_threshold_above_low  0.3 \
       -static_noise_threshold_below_high 0.3

route_opt -xtalk_reduction
        Optional: -optimize_wire_via -wire_size

report_noise
report_timing -crosstalk_delta
```

Prevention target more aggressive
Default: 0.45
Unit is Volt.

Default: 0.35
Unit is %

6-34

place_opt
set_clock_tree_options -max_transition 0.2 -max_fanout 32
set_clock_tree_references -reference $CLK_BUFFER_LIST
define_routing_rule MYRULE -spacings {M2 0.28 M3 0.28…}
set_clock_tree_options -routing_rule MYRULE
clock_opt

set_si_options -route_xtalk_prevention true \
-route_xtalk_prevention_threshold 0.35 \
-delta_delay true \
-static_noise true \
-static_noise_threshold_above_low 0.3 \
-static_noise_threshold_below_high 0.3

route_opt -xtalk_reduction
Optional: -optimize_wire_via –wire_size

report_noise
report_timing -crosstalk_delta

set_si_options
-route_xtalk_prevention TRUE|FALSE
-route_xtalk_prevention_threshold <float V>
Xtalk aware Global Route / Track-Assign
-delta_delay TRUE|FALSE
-static_noise TRUE|FALSE

-static_noise_threshold_above_low <float %>
-static_noise_threshold_below_high <float %>

NOTE:
The unit for the static_noise_threshold_above_* options has changed with version 2007.03-SP1. The values are now a percentage of the voltage, instead of absolute voltages.
report_si_options
Static Noise Thresholds :
0.3 (0.49V) above low
0.3 (0.49V) below high

“ECO” is an old term which stands for “engineering change order”. In the early days of circuit design, if a change needed to be made to a part of the design that was already defined or specified, you would fill out an “Engineering Change Order” form and have it signed off before making the change. Even though most companies no longer use ECO forms, the term is still used, and refers to design changes late, or near the end of the design phase.

Note:
You should use either the classic router or Zroute to route your design, and not mix the use of routers in your design flow. Although running Zroute on your design is possible after running the classic router, you should not run the classic router on your design after using Zroute. If you run the classic router after Zroute, IC Compiler issues the following message:

Error: It is not recommended to use classic router commands after
Zroute commands. (RT-302)

Route_opt: **runs the routers:**

- Global Route

- Track Assignment
- Detailed Route

Then performs numerous logic, placement, routing and crosstalk optimizations to produce the best routed design

10 Analysis

Examine the screen-outputs of place_opt and route_opt for design summaries:

- Utilization
- WNS – Worst Negative Slack
- TNS – Total Negative Slack
- 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

Generate more detailed reports

- Show all violating path end points
  - **report_constraint –all_violators**
- Show details of the worst violating setup paths
  - **report_timing**
- Report physical design statistics (e.g. utilization)
  - **report_design -physical**
- Analyze the congestion
  - Congestion map (GUI)
  - **report_congestion**

11 Output

Apply consistent naming:
change_names –hierarchy –rules Verilog

Save the design in the Milkyway database:
save_mw_cel -as routed

Save as Verilog netlist:

```
write     -format verilog \
     -hierarchy -output routed.v
```

Save the floorplan only:
write_def -output floorplan.def
write_floorplan floorplan.tcl

Saving the design in the Synopsys Milkyway format will save everything: The netlist, constraints, floorplan, any attributes applied (e.g. set_dont_touch, set_ideal_network, etc.)

Timing and Optimization Setup

Timing and Optimization Setup (1/2)

- **Timing and optimization in IC Compiler is controlled by many variables and commands. Example**

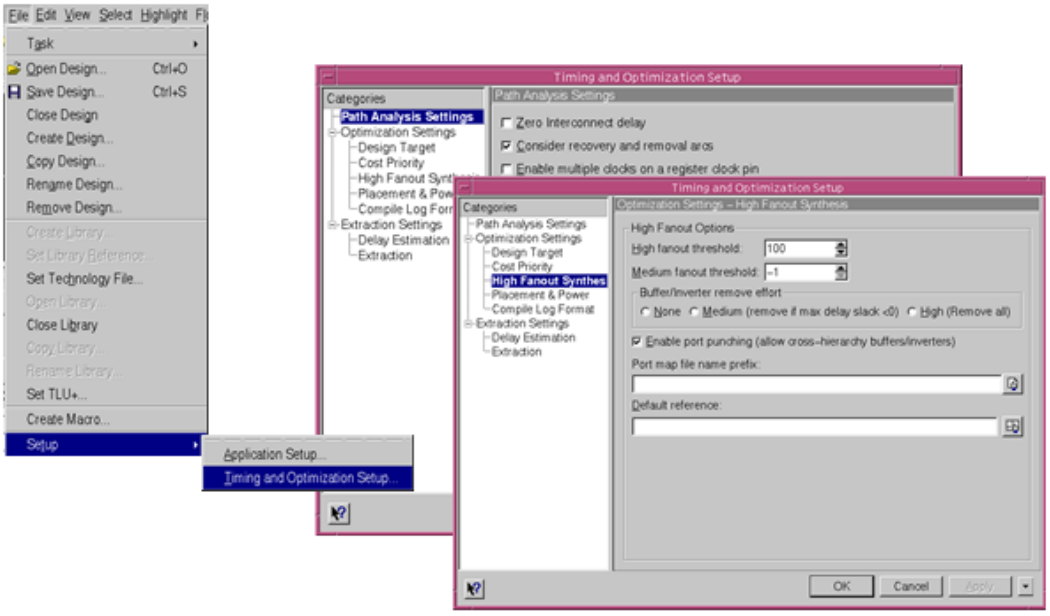
```
set enable_recovery_removal_arcs true
set timing_self_loops_no_skew true
set_cost_priority {max_delay max_capacitance}
set_ahfs_options -enable_port_punching true
```

- **Memorizing all these variables and commands can be a challenge – the GUI provides help!**

Use the GUI to perform your timing and optimization setup, then copy the variables / commands into your setup file

1-52

Timing and Optimization Setup (2/2)



1-53

Example "run" Script:

```
lappend search_path ./design_data ../db ../tlup
set link_library "* gates.db rams.db"
set target_library "gates.db"
create_mw_lib my_lib.mw \
  -technology tech_file.tf \
  -mw_reference_library "mwlib/gates mw_lib/rams" \
  -open
import_designs my_design.v \
    -format verilog  \
    -top MYDESIGN
set_tlu_plus_files \
    -max_tluplus abc_max.tlup \
    -min_tluplus abc_min.tlup \
    -tech2itf_map abc.map    ; explained later…
read_sdc my_design.sdc
check_timing
read_def -allow_physical  my_design.def
check_physical_constraints
place_opt
clock_opt
route_opt
save_mw_cel -as routed
```

The following two variables should also be defined if they are not named VSS/VDD before reading the design:
set mw_logic0_net "VSS"
set mw_logic1_net "VDD"
Notice since we are importing an ASCII verilog file, we must also read in the SDC constraints.

Local Disk Space Usage

When running a design through the IC Compiler flow, you will require local disk space to store the

database. Example:

Design size (instances)	Milkyway database** size place_opt	clock_opt	route_opt
150K	56MB	62MB	120MB
250K	150MB	178MB	303MB
400K	242MB	269MB	460MB
625K	277MB	325MB	569MB
700K	344MB	419MB	741MB
850K	917MB	950MB	1.5GB

*** this includes all CEL/ROUTE/PARA/... views plus any CEL attachments*

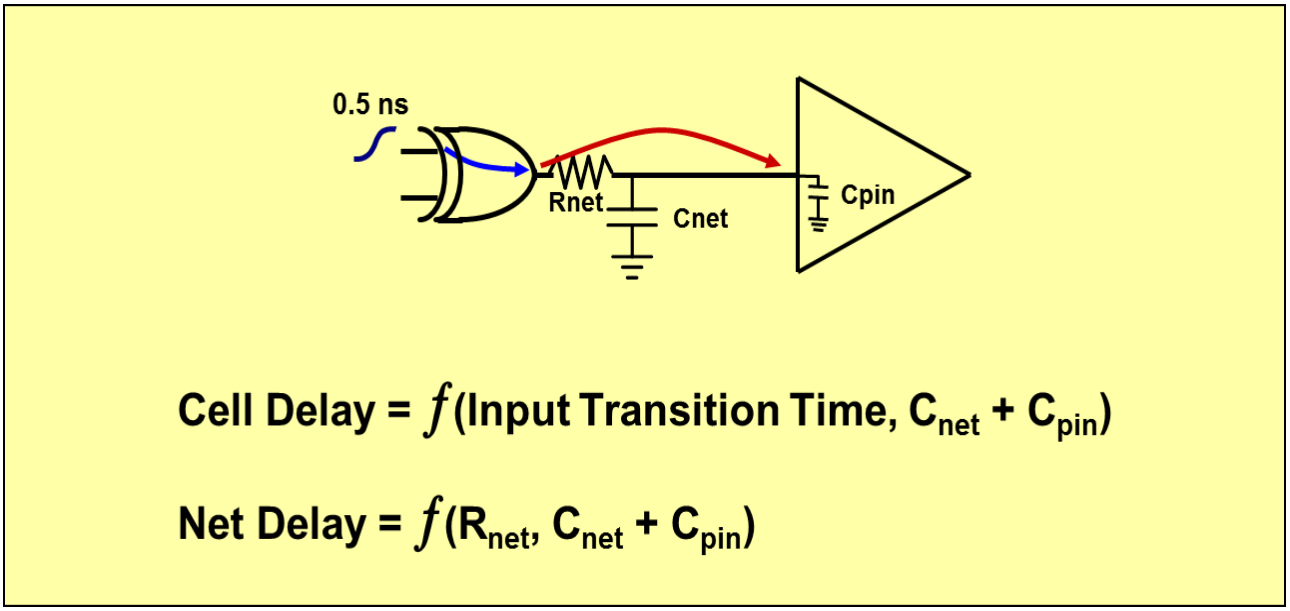
The above table shows the incremental disk space requirements for each step. So, to complete a 150K design, you need 56 + 62 + 120 = 238 MB.

STA

Before starting with IC Compiler, set up the libraries and operating conditions. Example:

```
set link_library "* abcmax.db"
set_operating_conditions \
    -analysis_type on_chip_variation \
    -max abc_wc -max_library abc_max
Using operating conditions 'abc_wc' found in library 'abc_max'.
report_timing -delay max|min
```

Timing and optimization in IC Compiler is controlled by many variables and commands. Example
set enable_recovery_removal_arcs true
set timing_self_loops_no_skew true
set_cost_priority {max_delay max_capacitance}
set_ahfs_options -enable_port_punching true
Timing is Based on Cell and Net Delays:



Cell delay is calculated using non-linear delay models, which are stored in the 'LM' view of each cell. NLDM is highly accurate as it is derived from SPICE characterizations. The delay is a function of the input transition time of the cell (TInput) [also called slew], the driving strength of the cell (RCell), the wire capacitance (CNet) and the pin capacitance of the receivers (CPin). A slow input transition time will slow the rate at which the cell's transistors can change state (from "on" to "off"), as well as a large output load (Cnet + Cpin), thereby increasing the "delay" of the logic gate.
There is another NLDM table in the library to calculate output transition. Output transition of a cell becomes the input transition of the next cell down the chain.

TLU+ Models

IC Compiler calculates C *and* R using the net geometry and the TLU+ look-up tables
UDSM process effects modeled
UDSM = Ultra Deep SubMicron
Generating TLU+ Models:

- **ITF (process file) provided by the vendor**
- **TLU+ model is generally not provided**
- **Generate TLU+ from ITF**

```
unix% grdgenxo -itf2TLUPlus -i <ITF file>   -o <TLU+ file>
```

Where: -itf2TLUPlus **generates TLU+ instead of nxtgrd file**
-i **is the ITF file**
-o **is the output, binary TLU+ model file**

Always use the latest Star-RCXT release to generate the models.

ITF = Interconnect Technology Format
When generating TLUPlus models with grdgenxo, the –itf2TLUPlus option must be the first option specified.
More and more ASIC vendors are supporting TLU+ models and they may provide the binary cap table files

for your use.
If possible, generate TLU+ models for at least two operating corners: min and max.

Loading TLU+ Models

```
set_tlu_plus_files \
    -max_tluplus abc_max.tlup \
    -min_tluplus abc_min.tlup \
    -tech2itf_map  abc.map
```

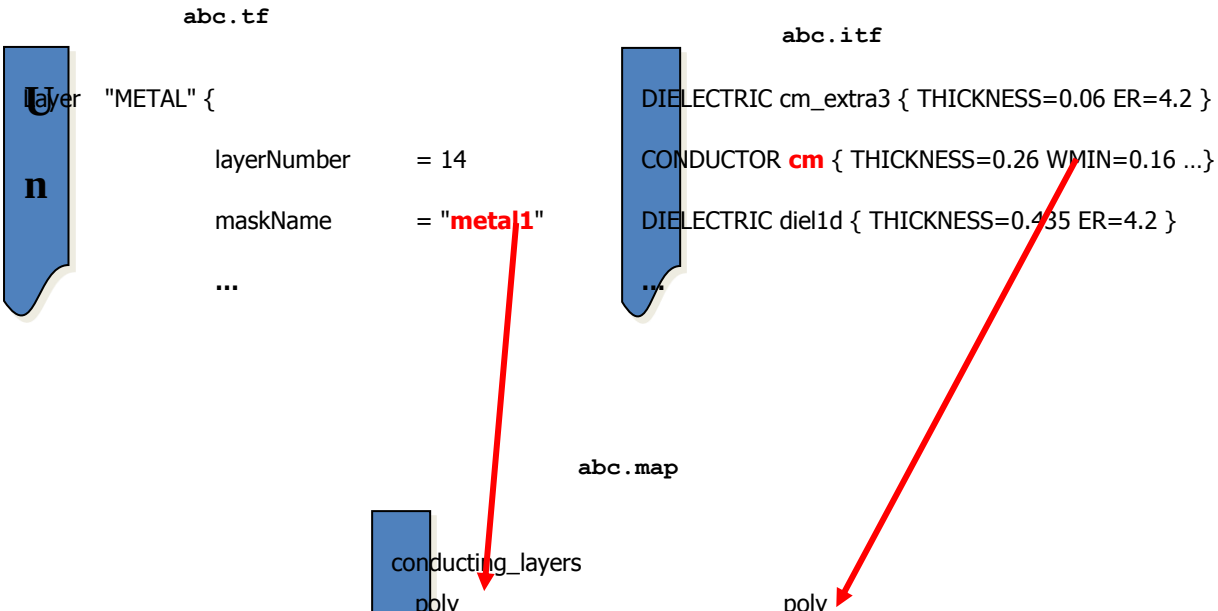
check_tlu_plus_files:

- Performs sanity check on TLU+ settings
- Execute this command after setting TLU+ to ensure correct TLU+ and map file

The TLU+ models are not stored in the Milkyway library, only a pointer to the models is stored with the cell. If you want the settings to be applied automatically when opening the cell, the variable auto_restore_mw_cel_lib_setup needs to be set to true.

Mapping file

The Mapping File maps the .tf (MW technology file) layer/via names to Star-RCXT .itf layer/via names.



Calculating Cell and Net Delay:

For <u>Cell</u> Delays, only C_{total} / C_{eff} is needed
Calculating <u>Net</u> Delay is done using Delay Calculation algorithms: Elmore, Arnoldi.

To calculate cell delays, Elmore uses C_{total}, while Arnoldi uses C_{eff}.
C_{eff} is used to calculate the cell delay. C_{eff} is the effective capacitance that the driver "sees". C_{eff} is calculated using iterative calculations that start with C_{total} which is a function of $\sum(C_{nets} + C_{pins})$
As shown earlier, cell delay = f(input transition, C_{total}).

PreRoute Delay Calculation Algorithm

- **Prior to routing, net geometry is estimated based on a Virtual Route**
- **Since Virtual Routing is only an estimate,**
  an Elmore model is used for delay calculation

PostRoute Delay Calculation Algorithms

- **After routing, detailed nets are available and extraction can be more accurate**
- **By default, Elmore is still used**
- **Arnoldi can be turned on for postroute calculations**

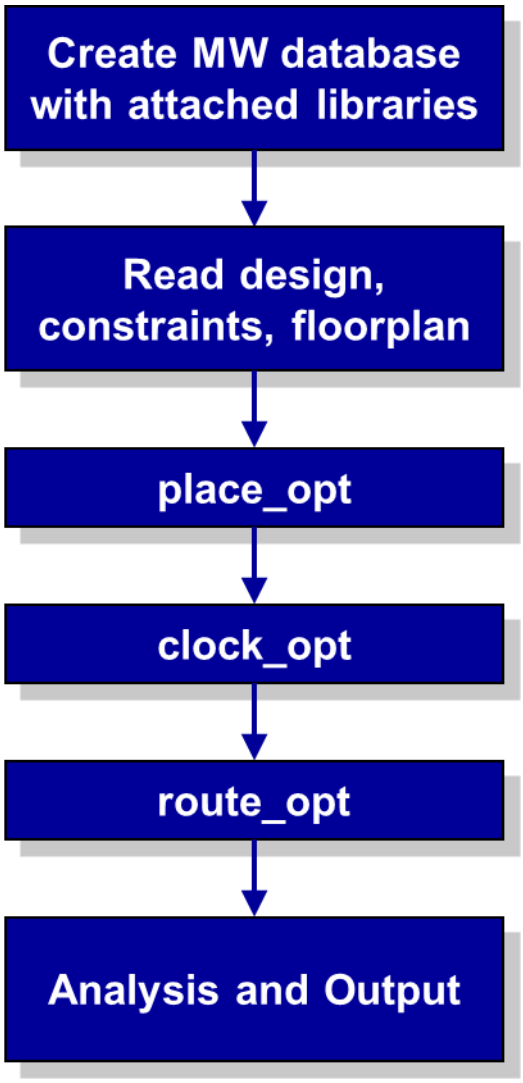
The default delay calculation algorithm in IC Compiler is Elmore.
To turn on the more accurate Arnoldi algorithm, use:
set_delay_calculation -arnoldi

Basic Flow Summary

You should now be able to:

- **Create a Milkyway library to hold your design**
- **Read all necessary files required to run**
  IC Compiler, resolving common errors/warnings
- **Set up timing for analysis and optimizations**
- **Execute the basic flow for placement, CTS and routing in IC Compiler**

Lab 1: Baseline flow for IC Compiler



Goals:

- **Setup design database and timing**
- **Perform baseline place, cts & route operations with associated default optimizations**
-

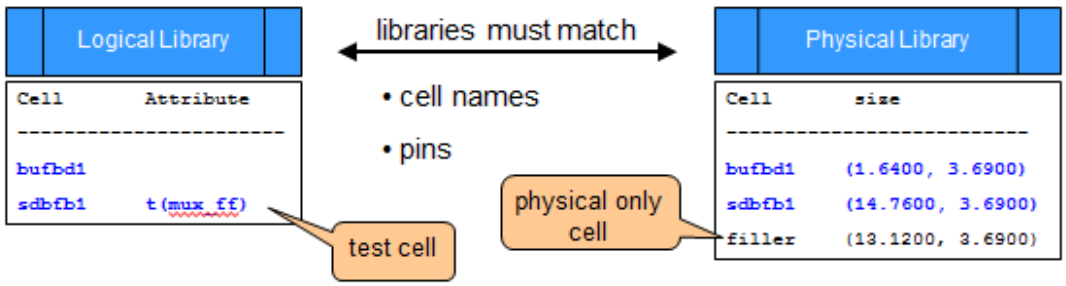
Migration from DB/PDB to MW: Terminology

DC / PC Terminology	Milkyway Terminology
Library Compiler (compile library) `read_lib, write_lib`	***Milkyway tool (create/edit Milkyway Library)*** `read_lef, read_plib, …`
Database (db) Library db *(timing)* PDB *(physical)* Design db/ddc	***Milkyway Library*** Library db (can be part of Ref Lib as LM view) Reference Library Design Library

Setup without using Milkyway:
`search_path`
`link_library`
`target_library`
	Setup using Milkyway library:
	`search_path`
	`link_library`
	`target_library`
	`set_mw_lib_reference`

Reference Library Requirements

- **Logical libraries (.db)**
- **Physical libraries (MW)**
- **Libraries are provided by the ASIC vendor**




```
set target_library {art005.db}
```

```
set_mw_lib_reference {art005}
```