IC Compiler® II
RedHawk Analysis Fusion
Application Note

Version N-2017.09-SP3, February 2018

SYNOPSYS®

RedHawk Analysis Fusion

The RedHawk™ power integrity solution is integrated with the IC Compiler II implementation flow through the RedHawk Analysis Fusion interface. You can use the RedHawk Analysis Fusion feature for rail analysis at different points in the physical implementation flow after power planning and initial placement are completed. This enables you to detect potential power network issues before you perform detail routing and thus significantly reduce the turnaround time of the design cycle.

When global routing is complete, use RedHawk Analysis Fusion to verify the robustness of the power and ground routing network. When placement is complete, use RedHawk Analysis Fusion to perform voltage drop analysis to calculate power consumption and to check for voltage drop violations. You can perform voltage drop analysis at other stages in the design flow, such as after detail routing. When chip finishing is complete, use RedHawk Analysis Fusion to perform electromigration analysis to check for current density violations.

This document describes how to use the RedHawk Analysis Fusion feature to perform rail analysis in the IC Compiler II environment. For a detailed description about the RedHawk analysis flows and commands, see the *RedHawk User Manual*.

This chapter includes the following topics:

- Overview
- Prerequisites for RedHawk Analysis Fusion
- Preparing RedHawk Design and Input Data
- Specifying Ideal Voltage Sources as Taps
- Missing Vias and Floating Pins Checking
- Performing RedHawk Voltage Drop Analysis
- Performing RedHawk Electromigration Analysis
- Performing Minimum Path Resistance Analysis
- Appendix: Frequently Asked Questions

Overview

RedHawk Analysis Fusion rail analysis supports gate-level rail analysis capabilities, including both static and dynamic rail analysis. You can use RedHawk Analysis Fusion analysis to perform the following types of rail analysis in the IC Compiler II environment:

- Missing vias and floating pin shapes checking

  To check for missing vias or unconnected pin shapes in the design, you must first configure the checking-related settings by using the `set_missing_via_check_options` command, and then perform the check by using the `analyze_rail -check_missing_via` command.

  For more information about checking for missing vias and floating pins in the design, see Missing Vias and Floating Pins Checking.

- Voltage drop analysis

  To perform voltage drop analysis, use the `analyze_rail -voltage_drop` command or choose Rail > Analyze Rail and select "Voltage drop analysis" in the GUI.

  Set the `-voltage_drop` option to `static`, `dynamic`, `dynamic_vcd`, or `dynamic_vectorless` to choose the type of analysis you want to perform.

  For more information about performing voltage drop analysis, see Performing RedHawk Voltage Drop Analysis.

- Electromigration analysis

  To perform electromigration analysis, use the `analyze_rail -electromigration` command.

  For more information about performing electromigration analysis, see Performing RedHawk Electromigration Analysis.

- Minimum path resistance analysis

  To perform minimum path resistance analysis, use the `analyze_rail -min_path_resistance` command.

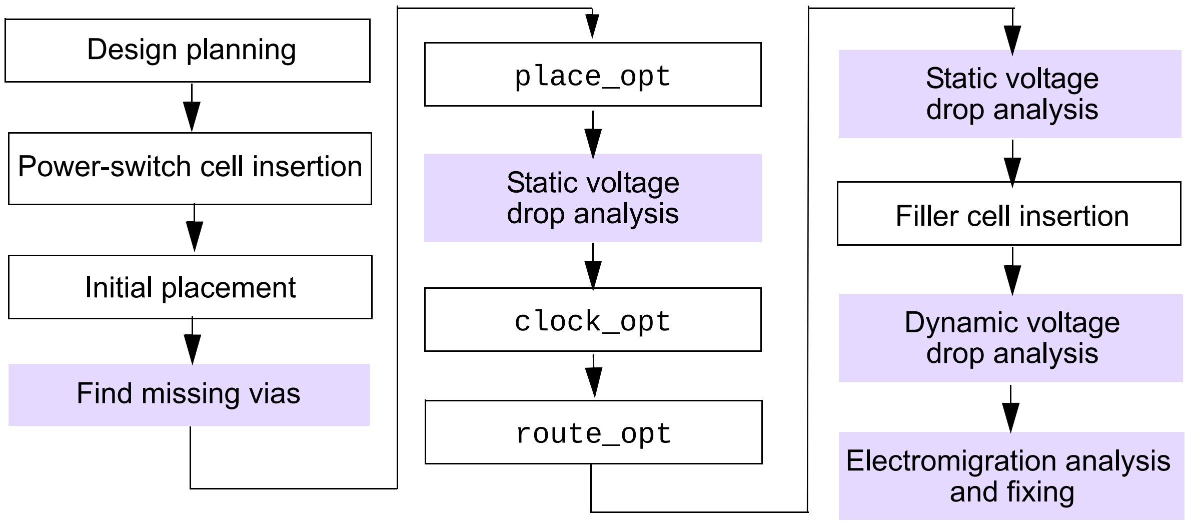
  For more information about performing minimum path resistance analysis, see Performing Minimum Path Resistance Analysis.

You can perform rail analysis on both multicorner-multimode and non-multicorner-multimode designs; however, when analyzing a multicorner-multimode design, RedHawk Analysis Fusion analyzes only the current scenario.

Depending on the type of analysis you run, the tool generates visual displays (maps) of the results that you can view in the IC Compiler II GUI, as well as error data that you can display in the IC Compiler II error browser. These maps and error data help you discover problem areas and determine corrective action without leaving the IC Compiler II environment.

Figure 1 shows where you would use these analysis capabilities in the typical design flow.

Figure 1 Using RedHawk Analysis Fusion in the IC Compiler II Design Flow


Design planning
Power-switch cell insertion
Initial placement
Find missing vias
place_opt
Static voltage drop analysis
clock_opt
route_opt
Static voltage drop analysis
Filler cell insertion
Dynamic voltage drop analysis
Electromigration analysis and fixing

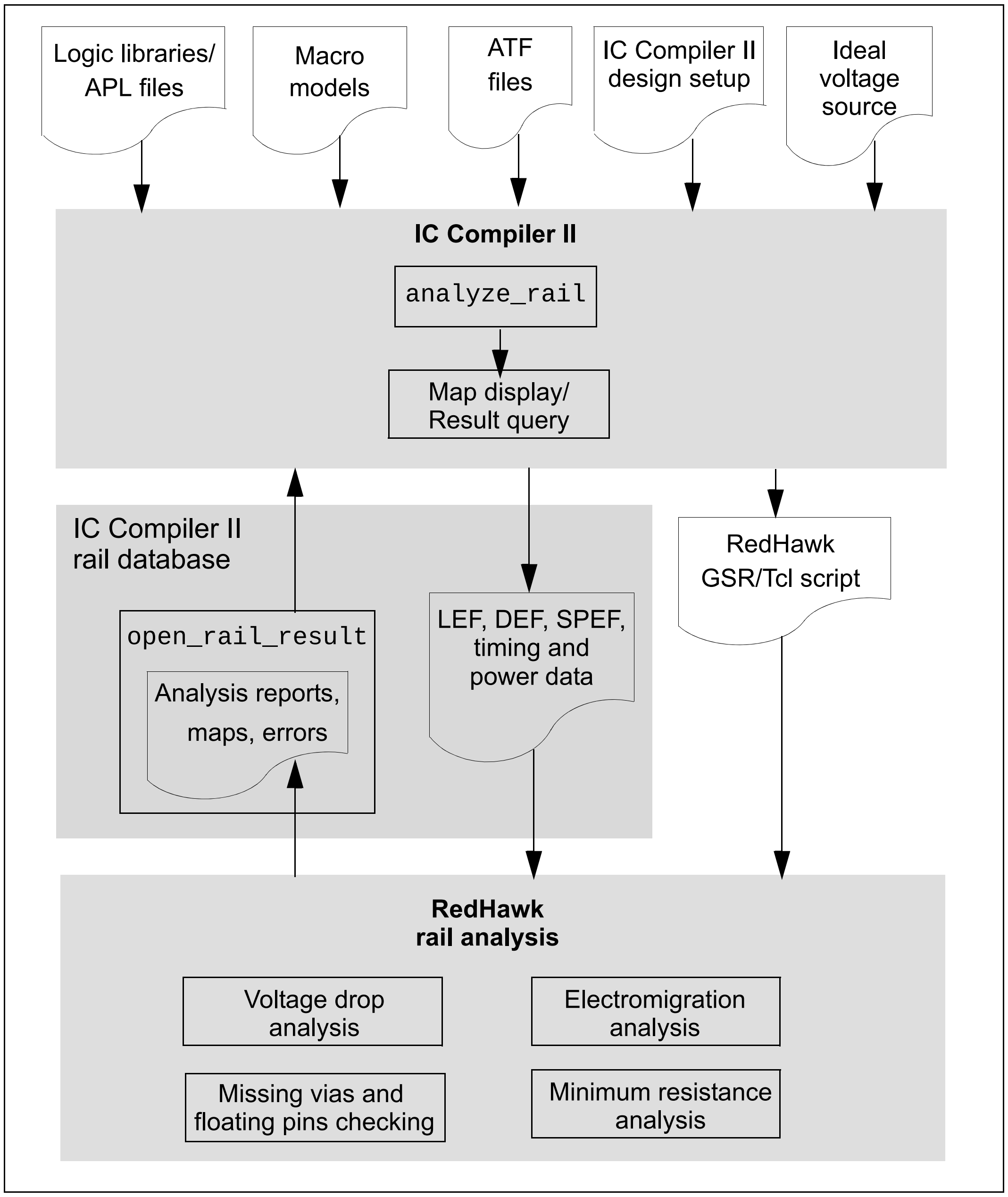
RedHawk Analysis Fusion Data Flow

The RedHawk Analysis Fusion feature allows you to perform rail analysis during the implementation stage. With the required input files, the IC Compiler II tool creates the RedHawk run script and the Global System Requirements (GSR) configuration file for invoking the RedHawk tool to run power and ground net extraction, power analysis, voltage drop analysis, and electromigration analysis.

When analysis is complete, the IC Compiler II tool generates analysis reports and maps using the results calculated by the RedHawk tool. You can then check for hot spots graphically in the IC Compiler II GUI. Error data and ASCII reports are also available to check for locations where limits are violated.

Figure 2 illustrates the data flow when using RedHawk Analysis Fusion to perform rail analysis in the IC Compiler II environment.

Figure 2   RedHawk Analysis Fusion Data Flow

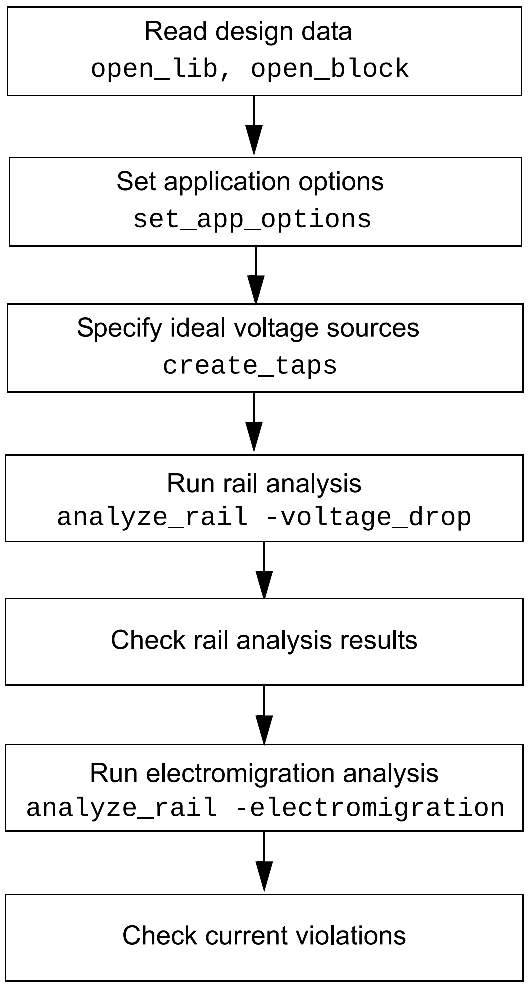

Logic libraries/ APL files
Macro models
ATF files
IC Compiler II design setup
Ideal voltage source
IC Compiler II
analyze_rail
Map display/ Result query
IC Compiler II rail database
open_rail_result
Analysis reports, maps, errors
LEF, DEF, SPEF, timing and power data
RedHawk GSR/Tcl script
RedHawk rail analysis
Voltage drop analysis
Electromigration analysis
Missing vias and floating pins checking
Minimum resistance analysis

RedHawk Analysis Fusion Analysis Flow

When you have specified the necessary input and design data, you can perform voltage drop and electromigration analyses on the design.

Figure 3 illustrates the steps in the basic static rail analysis flow.

Figure 3 Required Steps for a Basic Rail Analysis Using RedHawk Analysis Fusion


Read design data
open_lib, open_block
Set application options
set_app_options
Specify ideal voltage sources
create_taps
Run rail analysis
analyze_rail -voltage_drop
Check rail analysis results
Run electromigration analysis
analyze_rail -electromigration
Check current violations

Prerequisites for RedHawk Analysis Fusion

Before you can perform RedHawk Analysis Fusion for rail analysis, ensure that you have the correct licenses and input files, as described in the following topics:

- License Requirements
- Required Input Files
- Setting Up the Executables

License Requirements

You must have the following license to invoke the RedHawk tool within the IC Compiler II tool:

ICCompilerII-AF

ICCompilerII-AFN

SNPS_INDESIGN_RH_RAIL

The following RedHawk features are not supported with the SNPS_INDESIGN_RH_RAIL license key:

- Hierarchical designs with more than three levels of hierarchy
- Dynamic rail analysis with lumped or SPICE packages
- Signal electromigration
- In-rush current analysis

If you have the RedHawk licenses for other signoff analysis capabilities, such as analyzing hierarchical designs or performing dynamic analysis with lumped or SPICE packages,

1. Modify your GSR file or RedHawk run script to include the related configuration settings.
2. Set the following application option to enable the RedHawk signoff features:

   ```
   icc2_shell> set_app_options \
    -name rail.allow_redhawk_license_checkout -value true
   ```

   Setting the `set_app_options -name rail.allow_redhawk_license_checkout` application option to `true` allows RedHawk to retrieve the related RedHawk licenses for performing signoff features inside the IC Compiler II tool. The default is `false`.
3. Run the `analyze_rail` command to perform the analyses defined in the GSR or RedHawk script file.

Note:

RedHawk Analysis Fusion does not have interface to support signal electromigration and in-rush current analysis features. You cannot display the RedHawk signoff analysis results in the IC Compiler II GUI.

Required Input Files

Before invoking the RedHawk tool to perform power analysis, power and ground net extraction, or voltage drop analysis in the IC Compiler II environment, you must provide the required input files, as listed in Table 1:

Table 1 Required Input Data for Static and Dynamic Rail Analysis

Required Input Data	Note
Liberty logic library	
Apache technology file (*.tech)	
Apache macro models	
Apache power library (APL) files	
DEF/LEF files	When not specified, the IC Compiler II tool generates these files by using the data in the design library.
Static timing window (STA) file	
Signal SPEF files	
Instance power files that are generated by the IC Compiler II `report_power` command. When available, signal SPEF files are not required. The instance power files are recommended only for static analysis.	

Setting Up the Executables

To run RedHawk Analysis Fusion features, you must enable the RedHawk Analysis Fusion feature and specify the location of the RedHawk executable by using the `set_app_options -name rail.enable_redhawk` and `set_app_options -name rail.redhawk_path` commands, respectively.

You must ensure that the version of the RedHawk executable that you specify is compatible with the IC Compiler II version that you are using. To report the version compatibility, use the `report_versions` command.

Example:

```
icc2_shell> set_app_options -name rail.enable_redhawk -value true
icc2_shell> set_app_options –name rail.redhawk_path \
 –value /tools/RedHawk_Linux64e5_V19.0.2p2/bin
```

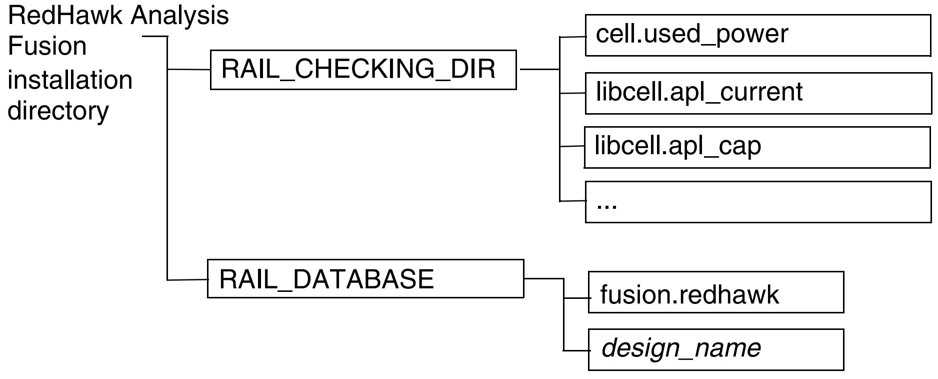
Specifying and Understanding Working Directories

During rail analysis, RedHawk Analysis Fusion creates a working directory to store the generated files, including analysis logs, scripts, and various output data. By default, the working directory is named RAIL_DATABASE. To use a different name for the working directory, use the `rail.database` application option.

Example:

```
icc2_shell> set_app_options –name rail.database \
–value RAIL_DATABASE_STATIC_RUN
```

The following figure shows a graphical representation of the data structures.



- RAIL_CHECKING_DIR: The directory where RedHawk Analysis Fusion saves report files on the missing information during rail analysis.

For example, the tool generates the following report files in the RAIL_CHECKING_DIR directory when static rail analysis is run:

- libcell.apl_current
- libcell.apl_cap
- libcell.missing_liberty

- RAIL_DATABASE: The directory where RedHawk Analysis Fusion saves and retrieves results and log files that are generated during supply net extraction, power calculation, and rail analysis.

  This directory contains the following subdirectories:

  - in-design.redhawk: The directory where the tool saves the input files.
  - *design_name*: The directory where the tool writes and retrieves analysis results. For example, running the `open_rail_result` command loads the data in this *design_name* directory for displaying maps in the GUI.

Preparing RedHawk Design and Input Data

To analyze voltage drop and current density violations in a design using the RedHawk Analysis Fusion capability, use the `set_app_options` command to specify the location of the libraries and the required input files.

Table 2 lists the application options for specifying required and optional design and input data for running RedHawk rail analysis within the IC Compiler II environment.

Table 2 Application Options for Specifying Design and Input Data

Application option	Description

`rail.lib_files`	Specifies the RedHawk library files.
	Example:
	`icc2_shell>` **`set_app_options -name rail.lib_files \`**
	`-value { test1.lib test2.lib }`

`rail.tech_file`	Specifies the Apache technology file for performing power and ground extraction and electromigration analysis.
	Example:
	`icc2_shell>` **`set_app_options -name rail.tech_file \`**
	`-value ChipTop.tech`

Table 2 Application Options for Specifying Design and Input Data (Continued)

Application option	Description

`rail.apl_files`	Specifies the Apache APL files, which contain current waveforms and intrinsic parasitics for performing dynamic rail analysis.
	Example:
	`icc2_shell> set_app_options -name rail.apl_files \`
	`-value {cell.current current cell.cdev cap}`

`rail.switch_model_files`	Specifies the RedHawk switch-cell model files for performing dynamic rail analysis.
	Example:
	`icc2_shell> set_app_options \`
	`-name rail.switch_model_files \`
	`-value {switch.model}`

`rail.macro_models`	Specifies RedHawk macro model files.
	Example:
	`icc2_shell> set_app_options -name rail.macro_models \`
	`-value {gds_cell_name1 mm_dir1 gds_cell_name2 mm_dir2}`

`rail.lef_files`	(Optional) Specifies a list of LEF files if available. When not specified, the IC Compiler II tool generates one using the data in the design library.
`rail.def_files`	(Optional) Specifies a list of DEF files. When not specified, the IC Compiler II tool generates one using the available data in the design database.
`rail.sta_file`	(Optional) Specifies the RedHawk static timing file (STA), which contains the final slew and delay information. When not specified, the IC Compiler II tool generates the static timing window information using the data in the design library.
`rail.spef_files`	(Optional) Specifies a list of SPEF files, which contain the detailed resistive and capacitive loading data of the signal nets. When not specified, the IC Compiler II tool generates the SPEF file using the data in the design library.

Generating Rail Analysis Script Files

Alternatively, you can run the `analyze_rail -script_only` command to generate the settings required for running rail analysis based on the information in the design library without actually executing the analysis run.

By default, the `analyze_rail -script_only` command writes data to the RAIL_DATABASE directory under the current working directory. The output files that are saved to the directory are

- The RedHawk Global System Requirements (GSR) file, which contains configuration settings for running RedHawk rail analysis
- The SDC file
- The signal parasitic file in SPEF format
- The analyze_rail.tcl file, the RedHawk run script file which includes the commands that are required for running rail analysis

Specifying Ideal Voltage Sources as Taps

Taps are used to model the external voltage source environment in which the device under analysis operates; they are not part of the design itself, but can be thought of as virtual models of voltage sources. The location of the ideal voltage sources and the ideal power supplies in the design are required to achieve accurate rail analysis results.

To create a tap point, use the `create_taps` command. You can create two or more tap points close to one another by running multiple `create_taps` commands to specify taps in either of the following ways.

- Treating top-level PG pins as taps

  Use the `-top_pg` option when analyzing a block-level design or when the designs do not have physical pad or bump information available.

  Use the `-supply_net` option to explicitly associate a tap to the supply (either power or ground) net connected to the object from which the tap was defined. To implicitly associate a tap with the supply net, run the `create_taps` command without the option.

  The following example creates taps for all the top-level supply pins.

  ```
  icc2_shell> create_taps -supply_net VDD -top_pg
  ```

- Creating taps on absolute coordinates

  Use the `-layer` and `-point` options to create taps at the specified coordinates. If there is no supply net, supply pin, or via shape at the specified location, which means there is no conductive path to the supply network, the tap has no effect on rail analysis despite being present.

  If you define a tap point in a location where a tap already exists, the tool issues a warning message and replaces the original tap with the new one.

  You must specify the supply net with which this tap point is associated by using the `-supply_net` option.

  Note:

  When you use the `-point` option in conjunction with the `-of_objects` option, the location is relative to the origin of the specified object. Otherwise, it is relative to the origin of the current top-level block.

  You cannot specify the location with the `-point` option when using the `-top_pg` option.

  The following example creates a tap using absolute coordinates. The command checks whether it touches any layout shape.

```
icc2_shell> create_taps -supply_net VDD -layer 3 -point {100.0 200.0}
Warning: Tap point (100.0, 200.0) on layer 3 for net VDD does not
touch any polygon (RAIL-305)
```

- Using existing instances as taps

  To create tap points on the specified objects, use the `create_taps -of_objects` command. Each item in the object list can be a cell or a library cell, or a collection of cells and library cells. By default, the tool creates the taps on the power and ground pins of the objects. If you specify the `-point` option, the tool creates the taps at the specified physical location relative to the origin of each object.

  Use existing instances for creating taps when analyzing a top-level design where pad cells are instantiated physically.

  Use the `-supply_net` option to explicitly associate a tap to the supply (either power or ground) net connected to the object from which the tap was defined. To implicitly associate a tap with the supply net, run the `create_taps` command without the option.

  The following example creates taps for each instance of the some_pad_cell library cell. The tool creates one tap per instance on the metal1 layer at a location of {30.4 16.3} relative to the origin of the cell instance.

```
icc2_shell> create_taps -of_objects [get_lib_cell */some_pad_cell] \
            -layer metal1 -point {30.4 16.3}
```

The following example creates taps on all power and ground pins of all cell instances that match the *power_pad_cell* pattern.

```
icc2_shell> create_taps -of_objects [get_cells -hier *power_pad_cell*]
```

- Importing a tap file

  If you are working with a block that does not yet have pins defined, or if the location or design of the pad cells is not finalized, you can define tap locations, layer numbers, and supply nets in an ASCII file and then import it into the tool by using the `create_taps -import` command.

  The supported tap file format is as follows:

  ```
  net_name layer_number X-coord Y-coord R(R_value] L(L_value) C(C_value)
  ```

  The tap file supports lumped resistance-inductance-capacitance (RLC) networks. The unit is ohm for resistance, henry for inductance, and farad for capacitance.

  Lines starting with semicolon (;) or pound (#) symbols are treated as comments. The x-coordinate and y-coordinate values are specified in microns.

  The following example shows how to import taps from a tap file without specifying lumped RLC:

  ```
  icc2_shell> exec cat tap_file
     VDD 3 500.000 500.000
     VSS 5 500.000 500.000
  icc2_shell> create_taps -import tap_file
  ```

  The following example shows how to import taps from a tap file in which lumped RLC is defined:

  ```
  icc2_shell> exec cat tap_file_rlc
     VDD 3 500.000 500.000 R(1.0) L(1.0e-9) C(2.0e-12)
     VSS 5 500.000 500.000
  icc2_shell> create_taps -import tap_file_rlc
  ```

- Creating taps with packages

  You can use a tap as a simple package RLC (resistance-inductance-capacitance) model with a fixed set of electrical characteristics assigned to all power and ground pads.

  More complicated package modeling, in which different values can be assigned to each pad, are provided by a Spice subcircuit model or an S-parameter model. These three types of models are described in the following sections.

  For a detailed description about simple package RLC models, see the related section in *RedHawk User Manual*.

You can specify a name for the created tap point with `-name tap_name` option. If the specified name is already present in the design, the tool issues a warning message, and replaces the original one with the new one. When you create multiple tap points by a single invocation of the `create_taps` command, all of the created taps are named by using the *tap_name*_NNN naming convention, where NNN is a unique integer identifier and *tap_name* is the string argument to the `-name` option.

When creating taps, by default the `create_taps` command checks whether the physical location touches a layout shape. To place taps outside a shape without issuing a warning message, disable the checking by using the `-nocheck` option.

To find a substitute location for the invalid taps, rerun the `create_taps` command with the `-snap_distance` option.

Example:

```
icc2_shell> create_taps -point { 2056.818 1515.832 } \
 -layer 38 -supply_net DVSS  -snap_distance 100
```

For more information, see Validating Taps and Finding Invalid Taps.

Validating Taps

When creating taps, by default the `create_taps` command checks whether the physical location, specified by the `-point` and `-layer` options, touches a layout geometry. If the defined tap point does not touch any layout, the tool issues a warning message. To place taps outside a shape without issuing a warning message, disable the checking by using the `-nocheck` option.

Tap Attributes

During validation checking, each tap object inserted in the current session is marked with the `is_valid` attribute to indicate its validity status. When validation checking is enabled, the attribute value is `YES` for valid taps and `NO` for invalid taps. When validation checking is disabled, the attribute value is `UNKNOWN` for all taps.

To check the attribute value for a tap, use the `get_attribute [get_taps] is_valid` command.

The tool updates the tap attributes in the subsequent voltage drop and minimum path resistance analyses. Each tap belonging to the power and ground nets is checked against the extracted shape. When the analysis is complete, the tool updates the tap validity status based on the rail analysis results.

Example

The following command creates a tap with the `-nocheck` option enabled. No warning message is issued.

```
icc2_shell> create_taps -supply_net VDD -layer 3 \
          -point {100.0 200.0} -nocheck
```

The following command creates the 101st tap point. No warning message is issued since only the first 100 taps are checked:

```
icc2_shell> create_taps -supply_net VDD -layer 3 -point {100.0 200.0}
```

The following example creates taps using the `create_taps` command with the `-nocheck` option.

```
icc2_shell> create_taps -nocheck -point {100 200} -layer M1 \
             -supply_net VDD
icc2_shell> create_taps -nocheck -point (150 250} -layer M2 \
             -supply_net VSS
```

Missing Vias and Floating Pins Checking

Before you run missing vias and floating pins checking, make sure you have the required input files as described in Preparing RedHawk Design and Input Data.

Check for missing vias or floating pins in the design during voltage drop analysis or in a separate run. For example, you can check for missing vias or floating pins after completing the power structure and before the `place_opt` command. This allows you to find power network errors in the design before rail analysis.

By default, RedHawk Analysis Fusion detects the following types of power network errors:

- Floating pin shapes

  Floating pin shapes are pin shapes that are not contiguously and physically connected to an ideal voltage source.

- Missing vias

  If two overlapping metal shapes do not contain a via within their overlapping area, it is considered a potential missing via error.

To check for missing vias or unconnected pins,

1. Define one or more options for finding missing vias by using the `set_missing_via_check_options` command, as described in Setting Checking Options.
2. Save the block by using the `save_block` command.
3. Perform the checking by using the `analyze_rail -check_missing_via` command, as described in Checking Missing Vias and Floating Pins.
4. Examine the checking results by opening the generated error files in the error browser or writing the checking results to an output file, as described in Viewing Error Data and Writing Checking Reports.

5. Insert PG vias to fix the reported missing vias, as described in Fixing Missing Vias.

Setting Checking Options

Before checking for missing vias in the design, use the `set_missing_via_check_options` command to specify the necessary options for missing via and floating pin checks.

To check the filter options set by the `set_missing_via_check_options` command, use the `report_missing_via_check_options` command.

To remove the filter options set the `set_missing_via_check_options` command, use the `remove_missing_via_check_options` command.

Table 3 lists the commonly used options for the `set_missing_via_check_options` command. For a detailed description about each option, see the command man page.

Table 3 Commonly Used Options for the set_missing_via_check_options Command

Option	Description
`-redhawk_script_file`	Uses an existing RedHawk script file for the missing via check.
`-exclude_stack_via`	Excludes stack vias from the missing via check. When not specified, by default the tool includes stacked vias during analysis.
`-check_valid_vias`	Reports if the metal overlap can accommodate a valid via model.
`-exclude_cell`/ `-exclude_instance`	Excludes the area covered by all instances of the specified cell, or by a list of cell instances from the missing via check. When not specified, by default the tool checks missing vias at the full-chip level.
`-exclude_regions`	Specifies multiple rectangular regions to exclude from the missing via check.
`-sort_by_hot_inst`	Sorts missing vias by voltage drop values. By default, the tool sorts by average delta voltage between defined layers.
`-threshold -1 | voltage_V`	Sets a voltage threshold that triggers a missing via check. Set the option to -1 to disable the voltage check filtering. The default is 1 mV.

Table 3 Commonly Used Options for the set_missing_via_check_options Command

Option	Description
`-min_width`	Ignores segments smaller than the specified width.
`-pitch`	Specifies the pitch for missing vias on parallel wires crossing or touching GDS blocks.
`-gds`	Flags missing vias that are inside the GDS block region and are in routes across the GDS blocks. By default, the missing via check ignores items that are inside or overlapped a GDS2DEF or GDSMMX block.
`-ignore_inter_met`	Ignores wires on all layers between the specified top layer and the bottom layer.

Checking Missing Vias and Floating Pins

Use the `analyze_rail -check_missing_via` command to check for missing vias and floating pins in the block with or without the `-voltage_drop` option.

When running missing via check during voltage drop analysis, the tool reports vias that are found to be unconnected or floating, and those whose voltage differences at two ends are larger than the threshold specified with the `set_missing_via_check_options -threshold` command. When checking for missing vias without the `-voltage_drop` option, the tool lists only the missing vias in the report file.

When the missing via check is complete, open the generated error data to examine the missing via errors in the error browser. For details, see Viewing Error Data.

Output Report Data

When rail integrity checking is complete, the tool writes the following data to the RAIL_DATABASE working directory.

- apache.missingVias

  Lists the missing vias found in the design. RedHawk Analysis Fusion uses this file and populates it as an IC Compiler II missing via error view. The voltage difference cross a via's two ends is included in the error cell description, and is used to sort the reported missing vias by voltage differences from high to low.

- design_name.PG.unconnect

  Lists the instances that are physically unconnected to the power and ground nets

- design_name.VDD.unconnect

Lists the instances that are physically unconnected to the power nets

- design_name.GND.unconnect

  Lists the instances that are physically unconnected to the ground nets

- design_name.PinInst.unconnect

  Lists the pin instances that are physically unconnected to the power and ground nets

- design_name.PinInst.unconnect.logic

  Lists the pin instances that are logically unconnected to the power and ground nets

Example

The following example sets a threshold value to compare voltages cross two ends of a via, and runs the missing via check during static voltage drop analysis.

```
icc2_shell> set_missing_via_check_options -threshold voltage_value
icc2_shell> analyze_rail -check_missing_via -voltage_drop static
```

By default, *voltage_value* is set to -1 to disable the voltage check.

For a detailed description about how to run the missing via check with the `analyze_rail` command, see the man page.

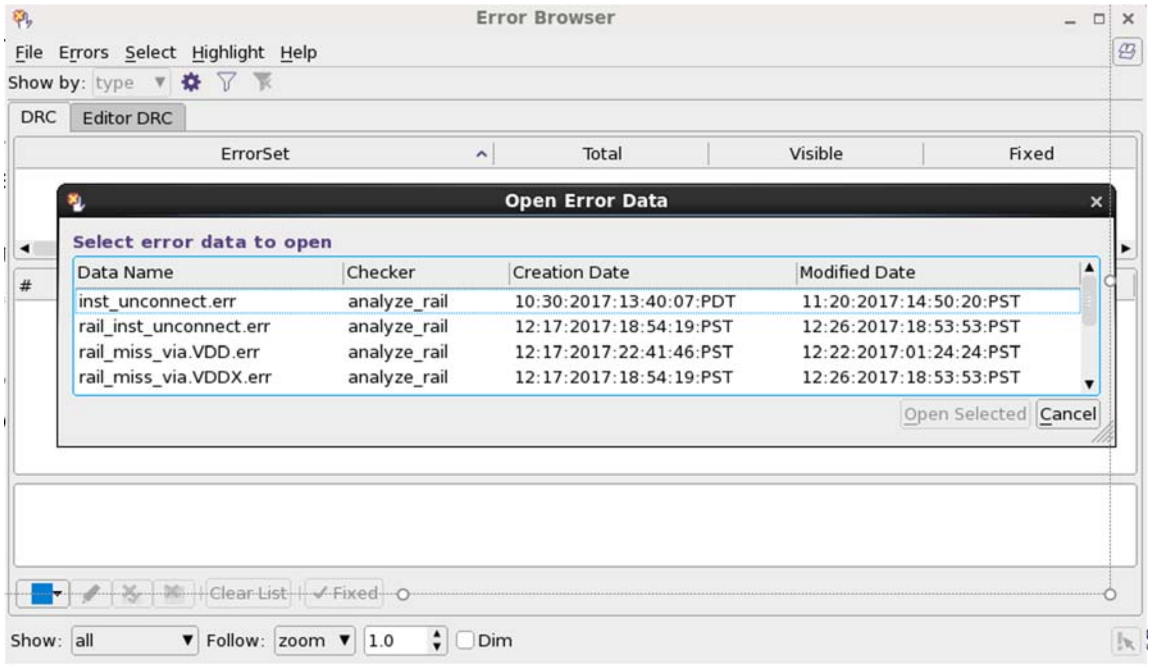
Viewing Error Data

RedHawk Analysis Fusion reads and populates the RedHawk checking reports to error data and saves the error views, named missing_via.*supplyNetName*.err, in the working directory. The tool generates one error data file for each supply net. Before you quit the current session, be sure to save the block to keep the generated error views in the design.

To open the error views in the error browser, choose File > Read Error File from IC Compiler II GUI, and then select the error file to open in the Open Error Data dialog box (see Figure 4).

For more information about the error browser, see “Using the Error Browser” in the *IC Compiler II Graphical User Interface User Guide*.

Figure 4 Opening Missing Via Errors in Error Browser


Error Browser
File Errors Select Highlight Help
Show by: type
DRC Editor DRC
ErrorSet Total Visible Fixed
#
Open Error Data
Select error data to open
Open Selected Cancel
Clear List ✓ Fixed
Show: all Follow: zoom 1.0 Dim

Data Name	Checker	Creation Date	Modified Date
inst_unconnect.err	analyze_rail	10:30:2017:13:40:07:PDT	11:20:2017:14:50:20:PST
rail_inst_unconnect.err	analyze_rail	12:17:2017:18:54:19:PST	12:26:2017:18:53:53:PST
rail_miss_via.VDD.err	analyze_rail	12:17:2017:22:41:46:PST	12:22:2017:01:24:24:PST
rail_miss_via.VDDX.err	analyze_rail	12:17:2017:18:54:19:PST	12:26:2017:18:53:53:PST

Writing Checking Reports

To write the missing via checking results to an output file, use the `report_rail_result` command. Specify the error type to report by using the `-type` option.

The supported error types are:

- `pg_pin_power`
- `voltage_drop_or_rise`
- `effective_voltage_drop`
- `minimum_path_resistance`
- `effective_resistance`
- `instance_minimum_path_resistance`
- `instance_power`
- `missing_vias`
- `unconnected_instances`

In the output report file, the tool lists the error in different formats, based on the error types. For example,

- For the `missing_vias` type, the format is

  `net_name via_location_x_y top_metal_layer bottom_via_layer delta_voltage`

- For the `unconnected_instances` error type where either or both of power and ground nets are physically disconnected to ideal voltage sources, the format is

  `unconnection_type {net_names} instance_name instance bbox`

- For the `unconnected_instances` type, the format depends on the error condition. If the power or ground nets are either floating or are logically floating but physically connected to ideal voltage sources, the format is

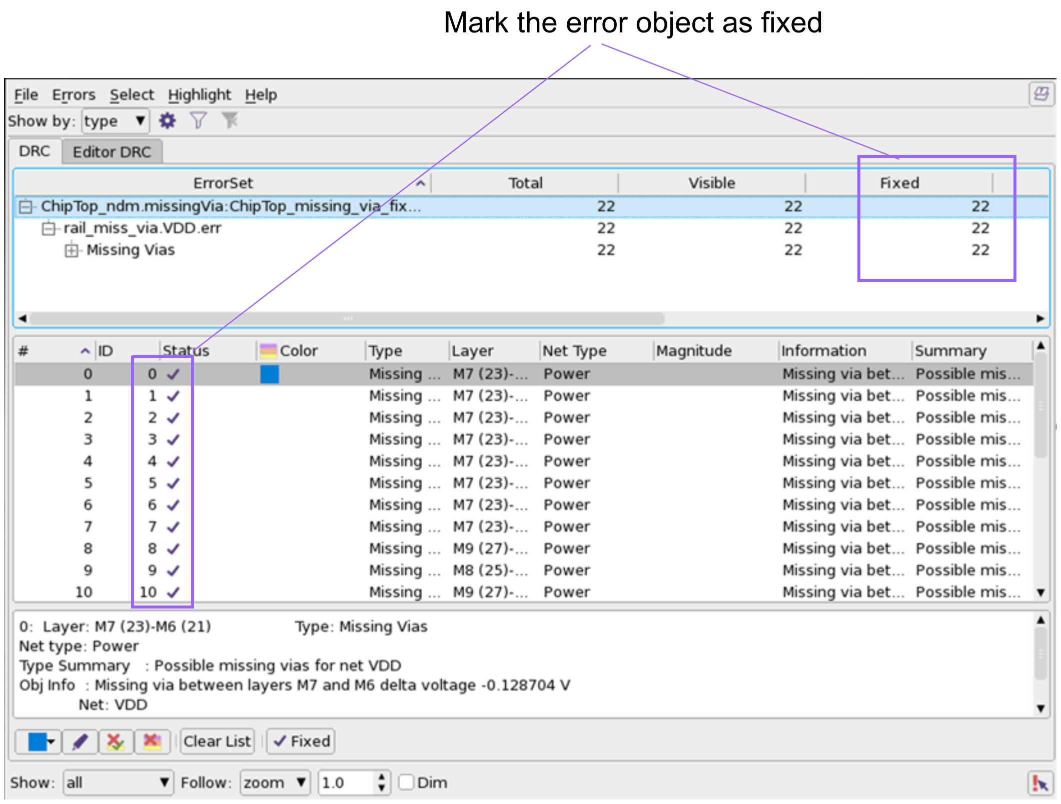
  `unconnection_type {net_names} instance_name:pin_name instance bbox`

Fixing Missing Vias

To fix the missing vias by inserting PG vias based on the specified DRC error objects written by the `analyze_rail` command, use the `fix_pg_missing_vias` command.

When the tool inserts a PG via for a missing via error object, it sets the status of the error object to `fixed` in the error browser (see Figure 5). This allows you quickly check if all the missing vias are fixed.

Figure 5 Error Status After Inserting PG Vias for Missing Vias


Mark the error object as fixed
File Errors Select Highlight Help
Show by: type
DRC Editor DRC
Clear List
Fixed
Show: all Follow: zoom 1.0 Dim

ErrorSet	Total	Visible	Fixed
ChipTop_ndm.missingVia:ChipTop_missing_via_fix...	22	22	22
rail_miss_via.VDD.err	22	22	22
Missing Vias	22	22	22

#	ID	Status	Color	Type	Layer	Net Type	Magnitude	Information	Summary
0	0	✓	🟦	Missing ...	M7 (23)-...	Power		Missing via bet...	Possible mis...
1	1	✓		Missing ...	M7 (23)-...	Power		Missing via bet...	Possible mis...
2	2	✓		Missing ...	M7 (23)-...	Power		Missing via bet...	Possible mis...
3	3	✓		Missing ...	M7 (23)-...	Power		Missing via bet...	Possible mis...
4	4	✓		Missing ...	M7 (23)-...	Power		Missing via bet...	Possible mis...
5	5	✓		Missing ...	M7 (23)-...	Power		Missing via bet...	Possible mis...
6	6	✓		Missing ...	M7 (23)-...	Power		Missing via bet...	Possible mis...
7	7	✓		Missing ...	M7 (23)-...	Power		Missing via bet...	Possible mis...
8	8	✓		Missing ...	M9 (27)-...	Power		Missing via bet...	Possible mis...
9	9	✓		Missing ...	M8 (25)-...	Power		Missing via bet...	Possible mis...
10	10	✓		Missing ...	M9 (27)-...	Power		Missing via bet...	Possible mis...

```
0:  Layer: M7 (23)-M6 (21)              Type: Missing Vias
Net type: Power
Type Summary   : Possible missing vias for net VDD
Obj Info  : Missing via between layers M7 and M6 delta voltage -0.128704 V
         Net: VDD
```

Example

The following example inserts PG vias for all error objects that are reported by the `analyze_rail -check_missing_via` command:

```
icc2_shell> set errdm [open_drc_error_data rail_miss_via.VDD.err]
icc2_shell> set errs [get_drc_errors -error_data $errdm]
icc2_shell> fix_pg_missing_vias -error_data $errdm $errs
```

Performing RedHawk Voltage Drop Analysis

Note:

Before running the `analyze_rail` command to calculate voltage drops and current violations, ensure you have provided the input files required by each analysis mode.

To invoke the RedHawk tool for calculating voltage drops of the specified power and ground network, use the `analyze_rail` command with the following options.

- Use the `-voltage_drop` option to specify the analysis mode.
  - `static`: Performs static voltage drop analysis.
  - `dynamic`: Performs dynamic voltage drop analysis.
  - `dynamic_vectorless`: Performs dynamic voltage drop analysis without Verilog Change Dump (VCD) inputs.
  - `dynamic_vcd`: Performs dynamic voltage drop analysis with VCD inputs.
- Use the `-nets` option to specify the power and ground supply nets to analyze. The tool considers all the switched or internal power nets of the specified power nets in the analysis. You do not need to explicitly specify the internal power nets.

  When the `-nets` option is specified, you need to specify at least one of the following options: `-min_path_resistance` or `-voltage_drop`.
- (Optional) Use the `-switching_activity` option to specify an optional switching activity file. The `file_format` argument can be `VCD` (a VCD file generated from a gate-level simulation), `SAIF`, or `ICC2_ACTIVITY`.

  A VCD or SAIF file can reference a top-level design in a different hierarchy than the one used in the IC Compiler II session. The `strip_path` argument removes the specified string from the beginning of the object name. Using this option modifies the object names in the VCD or SAIF file to make them compatible with the object names in the design.

  For more information about how to specify the switching activity file, see the related sections in *RedHawk User Guide*.
- (Optional) By default, the tool uses the RedHawk power analysis feature to calculate the power consumption of the specified power and ground network. If you prefer having the IC Compiler II tool generate the necessary power data for rail analysis, set the `-power_analysis` option to `icc2`.

  For a detailed description about how to run power analysis using the IC Compiler II `report_power` command, see the “Analyzing the Power” topic in *IC Compiler II Implementation User Guide*.

- (Optional) By default, the `analyze_rail` command automatically generates a GSR file for running RedHawk rail analysis. To specify an external GSR file to append to the GSR file that is generated in the current run, use the `-extra_gsr_option_file` option.
- (Optional) Use the `-redhawk_script_file` option to specify an existing RedHawk script file that was generated in an earlier `analyze_rail` run.

  When using an existing script file for rail analysis, the tool honors the settings in this specified script file and ignores all other currently enabled options. For example, if you run the `analyze_rail -voltage_drop static -redhawk_script_file myscript.tcl` command, the tool ignores the `-voltage_drop` option setting.

  When the `-redhawk_script_file` option is not specified, you must specify the nets to analyze by using the `-nets` option.

For a detailed description about each option of the `analyze_rail` command, see the man page.

Example

The following example performs static voltage drop analysis on the VDD and VSS nets with the optional settings defined in a GSR configuration file.

```
icc2_shell> analyze_rail -voltage_drop static -nets {VDD VSS} \
 -extra_gsr_option_file add_opt.gsr
```

RedHawk Voltage Drop Analysis Results

When analysis is complete, by default the tool saves analysis results and logs in the RAIL_DATABASE directory. To save the results to a different directory, set the `rail.database` application option as follows:

```
icc2_shell> set set_app_options –name rail.database -value dir_name
```

The `analyze_rail` command generates the following files to invoke the RedHawk tool for running rail analysis:

- The RedHawk run script (analyze_rail.tcl*)
- The RedHawk GSR configuration file (*.gsr)

RedHawk generates the following files when rail analysis is complete:

- The RedHawk analysis log (analyze_rail.log.*)
- Timing window file (*.sta)
- Signal SPEF files (*.spef)
- DEF/LEF files (*.def, *.lef) for the full-chip analysis

- Instance power files (*.ipf) if power analysis is done by the IC Compiler II tool. When available, signal SPEF files are not required.

When voltage drop analysis is complete, the tool saves the analysis results (*.result) in the current working directory. The rail analysis results are used to display maps and query attributes to determine the problematic areas with huge voltage drops.

For reuse purposes, you can load the RedHawk Analysis Fusion analysis results back into the IC Compiler II tool without rerunning the analysis.

The following example reopens the rail results that was generated in the previous rail analysis run:

```
icc2_shell> set_app_options -name rail.enable_redhawk -value true
icc2_shell> open_rail_result
```

However, RedHawk Analysis Fusion does not support results that are generated by the RedHawk tool outside of the IC Compiler II environment. The RedHawk `dump icc2_result` command works only within the IC Compiler II session.

For more information about rail analysis results, see Working With the Rail Analysis Results.

Finding Problematic Areas

When analysis is complete, use the following techniques to find the areas with huge voltage drops in the design:

- Displaying Maps in the IC Compiler II GUI
- Opening Error Data Files

Displaying Maps in the IC Compiler II GUI

When rail analysis is complete, the tool generates the following maps in the current working directory:

- A voltage drop map for the specified supply nets

  The voltage drop map is a visual display of the colored range of voltage drop values overlaid on the physical supply nets. For static analysis, the map displays average voltage drop values. For dynamic analysis, the map displays peak voltage drop values, average voltage drop values, or peak voltage rise values.

- A parasitics map for the specified supply nets

  The parasitics map displays the parasitic resistance of a given supply net in the design. It shows the resistance for any given shape of the net.

- An instance power map for the design

  The instance power map displays the power distribution for the design.
- A current map for the design

  The current map displays the current distribution for the design.
- An instance peak current map for the design

  When dynamic rail analysis is complete, you can examine peak current values for different instances or areas in the instance peak current map.

To display maps in the IC Compiler II GUI,

1. Run the `open_rail_result` command to load the design data and analysis results.
2. Run the `gui_start` command to open the IC Compiler II GUI. In the GUI layout window, choose View > Map and select the map to display.

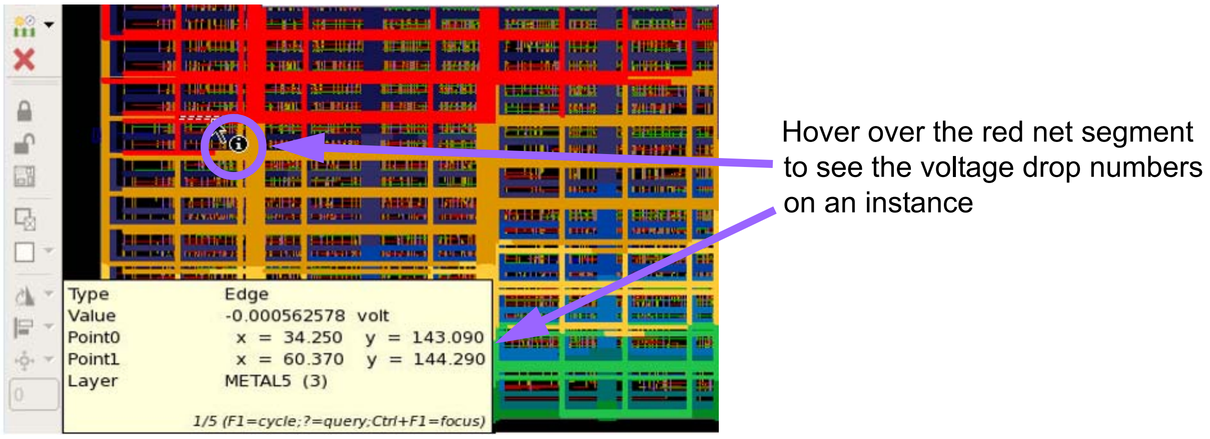
   In the map, problematic areas are highlighted in different colors. Use the options to investigate the problematic areas by selecting layers or nets to display. For example, to examine the voltage of one specific net, deselect all the nets and then select the net to display from the list. To analyze only the shapes on a layer, select the layer from the list.

   Use Histogram to change the range of the violations to display and the display color. This is useful if you want to display only the most critical drop ranges in the map.

   To inspect the voltage drop of the critical area in the map, click Zoom to Critical, and then use the InfoTip tool to examine the value.

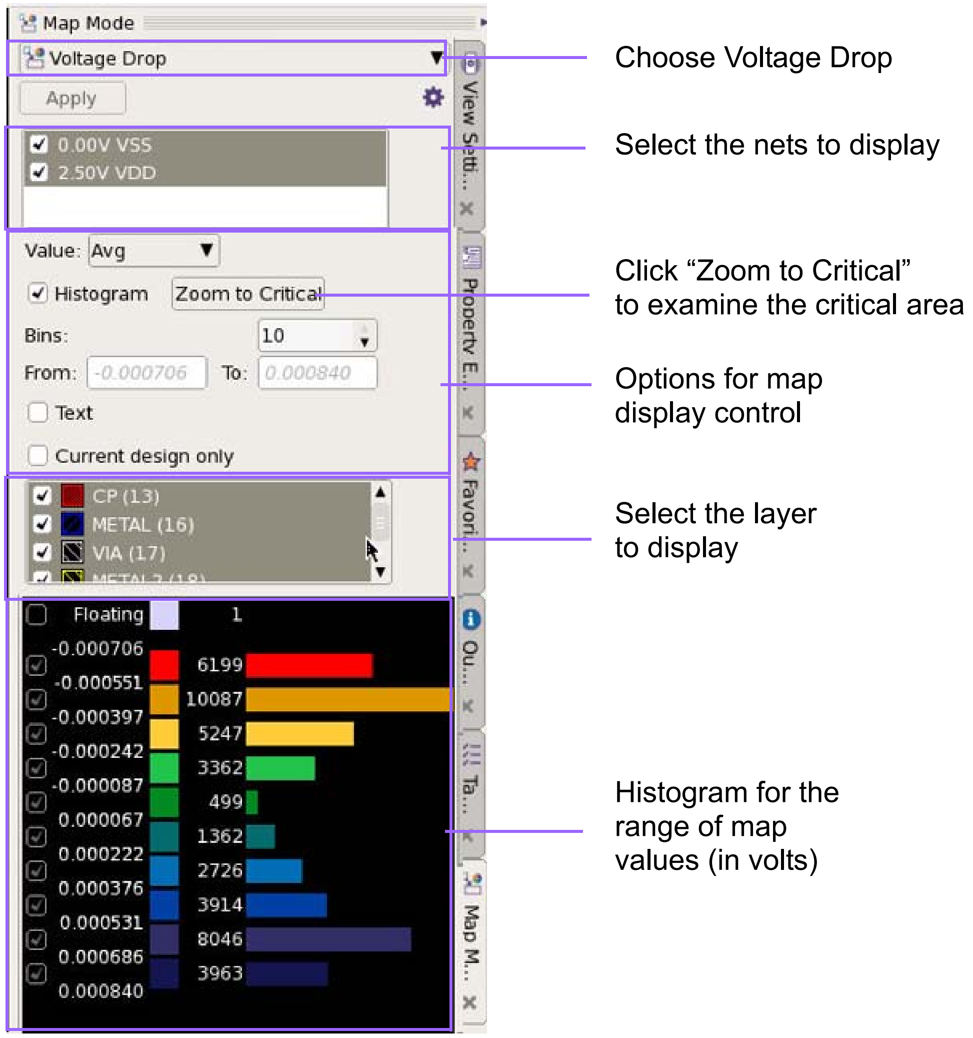
   Figure 6 shows an example of displaying a voltage drop map and checking for hot spots in the design.

Figure 6 Displaying a Voltage Drop Map


Hover over the red net segment to see the voltage drop numbers on an instance

Type	Edge
Value	-0.000562578 volt
Point0	x = 34.250 y = 143.090
Point1	x = 60.370 y = 144.290
Layer	METAL5 (3)

1/5 (F1=cycle;?=query;Ctrl+F1=focus)


Map Mode
Apply
View Setti...
Property E...
Favori...
Qu...
Ta...
Map M...
Choose Voltage Drop
Select the nets to display
Click “Zoom to Critical” to examine the critical area
Options for map display control
Select the layer to display
Histogram for the range of map values (in volts)

Voltage Drop

- ✓ 0.00V VSS
- ✓ 2.50V VDD

Value: Avg

✓ Histogram Zoom to Critical

Bins: 10

From: -0.000706 To: 0.000840

☐ Text

☐ Current design only

- ✓ CP (13)
- ✓ METAL (16)
- ✓ VIA (17)
- ✓ METAL2 (18)

Floating		1
-0.000706	-0.000551	6199
-0.000551	-0.000397	10087
-0.000397	-0.000242	5247
-0.000242	-0.000087	3362
-0.000087	0.000067	499
0.000067	0.000222	1362
0.000222	0.000376	2726
0.000376	0.000531	3914
0.000531	0.000686	8046
0.000686	0.000840	3963

Opening Error Data Files

When rail analysis is complete, the tool creates error files named design_*supplyNetName*_vd in the design library. The error files contain the locations of the voltage drop violations found during voltage drop analysis. You can use the error browser to examine the errors in the GUI.

For more information about the error browser, see “Using the Error Browser” in the *IC Compiler II Graphical User Interface User Guid*e.

Working With the Rail Analysis Results

After you have run the `analyze_rail` command, you can access the rail results cache to query the results.

- To load the rail results for displaying maps in the GUI, use the `open_rail_result` command.

  For example,

  ```
  icc2_shell> open_rail_result
  REDHAWK_RESULT
  ```

- To write out the rail analysis cache, use the `report_rail_result -type` *`result_type`* command. The `result_type` argument can be one of the following:

  ❍ `missing_vias`

  ❍ `unconnected_instances`

  ❍ `instance_power`

  ❍ `effective_voltage_drop`

  The following example lists the calculated instance power values after voltage drop analysis is finished.

  ```
  icc2_shell> report_rail_result -type instance_power \
   -supply_nets VDD output_files
  ```

- To query the results stored in the current rail results cache, use the `get_attribute` commands to query the attributes shown in Table 4, which are specific to the rail analysis results.

Table 4 Rail Result Object Attributes

Object	Attributes
Cell	`static_power`

Pin	`static_current`
	`peak_current`
	`static_power`
	`switching_power`
	`leakage_power`
	`internal_power`
	`min_path_resistance`
	`voltage_drop`
	`average_effective_voltage_drop_in_tw`
	`max_effective_voltage_drop_in_tw`
	`min_effective_voltage_drop_in_tw`
	`effective_resistance`

Reporting Effective Voltage Drops

When voltage drop analysis is complete, use the `report_rail_result -type effective_voltage_drop` command to write the calculated effective voltage drop of the specified power and ground nets to an output report file. Alternatively, you can graphically examine the effective resistances in the specified power and ground network by displaying an effective voltage drop map in the IC Compiler II GUI.

The following example writes the calculated effective voltage drop of the VDD and VSS nets to an output report file.

```
icc2_shell> report_rail_result -type effective_voltage_drop \
 -supply_nets VDD output_files
```

The following example shows the content of the effective voltage drop report:

```
#<cell_instance_pg_pin_name> <mapped_pg_pin_name>
 #<supply_net_name> <average_effective_voltage_drop>

oai311d1_G1995/VDD VSS
 VDD -5.096049979e-02
```

```
nr03d1_X1763/VDD VSS
 VDD -4.986800253e-02
nr13d1_Y1763/VDD VSS
 VDD -4.956080019e-02
nd02d0_A2247/VDD VSS
 VDD -4.950900003e-02
or12d1_P1995/VDD VSS
 VDD -4.924959689e-02
or02d0_Q1995/VDD VSS
 VDD -4.924950004e-02
...
```

Performing RedHawk Electromigration Analysis

To analyze the current density on the specified nets and identify the segments with potential electromigration issues, run the `analyze_rail -electromigration` command. Before running this command, ensure the Apache technology file defines the electromigration rules.

By default, the results are saved under the RAIL_DATABASE directory.

When electromigration analysis is complete, you can retrieve and open results for viewing in the IC Compiler II GUI by using the `open_rail_result` command, and then examine the possible current violations by displaying the electromigration map or by opening the error files in the error browser.

The `-electromigration` option must be used with the `-voltage_drop` option.

To perform RedHawk electromigration analysis:

1. Set the `rail.tech_file` application option to specify the Apache technology file (*.tech) that defines the layer-by-layer current density limits.

   Example:

   ```
   icc2_shell> set_app_options -name rail.tech_file \
    -value design_data/tech/test.tech
   ```

2. Run the `analyze_rail` command with the following options:

   - Use the -`voltage_drop` and -`electromigration` options to enable electromigration analysis.
   - Use the `-nets` option to specify the power and ground supply nets to analyze. The tool considers all the switched or internal power nets of the specified power nets in the analysis. You do not need to explicitly specify the internal power nets.

     When the `-nets` option is specified, you need to specify at least one of the following options: `-min_path_resistance` or `-voltage_drop`.

- ❍ (Optional) By default, the `analyze_rail` command automatically generates a GSR file for running RedHawk rail analysis. To specify an external GSR file to append to the GSR file generated in the current run, use the `-extra_gsr_option_file` option.
- ❍ (Optional) To specify an existing RedHawk script file that was generated in an earlier `analyze_rail` run, use the `-redhawk_script_file` option.

  When using an existing script file for rail analysis, the tool honors the settings in this script file and ignores all other currently enabled options. For example, if you run the `analyze_rail -voltage_drop static -redhawk_script_file myscript.tcl` command, the tool ignores the `-voltage_drop` option.

  When the `-redhawk_script_file` option is not specified, you must specify the nets to analyze by using the `-nets` option.

  For a detailed description about how to use the `analyze_rail` command, see the man page.

Viewing Electromigration Analysis Results

When electromigration analysis is complete, you can check for current violations by displaying an electromigration map or checking the generated errors in the error browser.

- Displaying the Electromigration Map
- Checking Electromigration Violations

Displaying the Electromigration Map

An electromigration map is a visual display of the colored range of electromigration values overlaid on the physical supply nets.

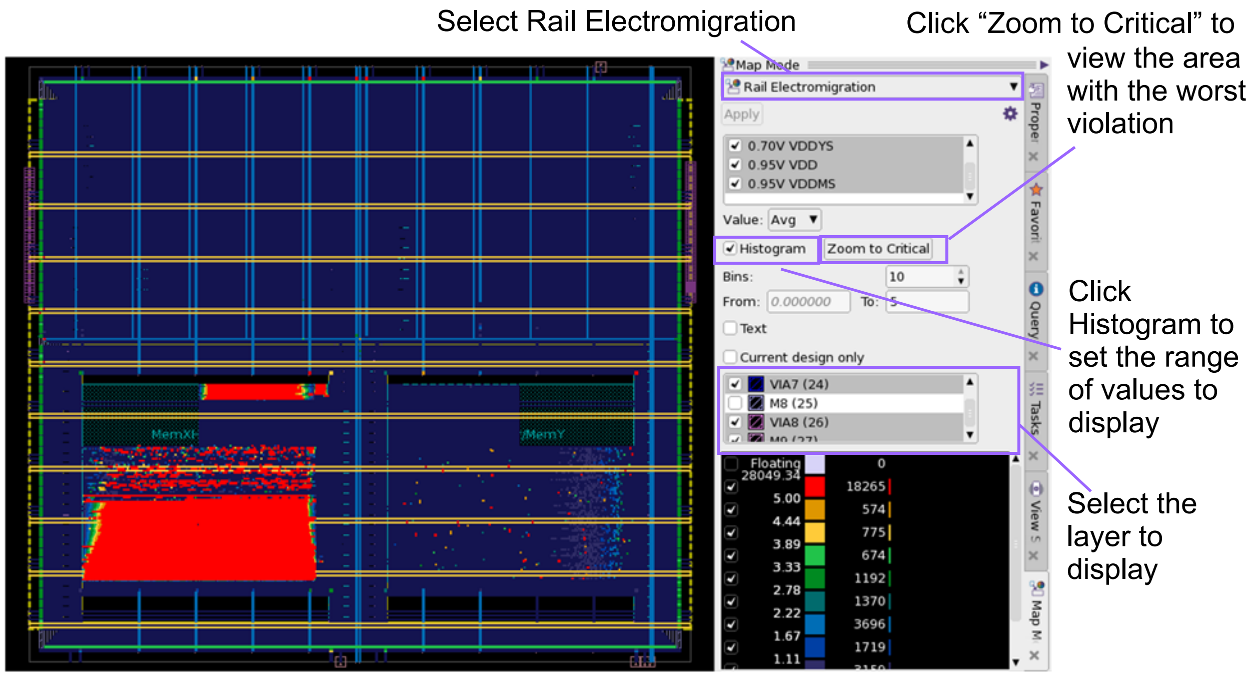
For static analysis, the map displays average electromigration values. For dynamic analysis, the map displays average electromigration values, peak electromigration values, or root mean square electromigration values.

To display electromigration maps in the IC Compiler II GUI,

1. Run the `open_rail_result` command to load the design data and analysis results.
2. Run the `gui_start` command to invoke IC Compiler II GUI. In the GUI layout window, choose View > Map and select Rail Electromigration.

   In the map, problematic areas are highlighted in different colors. Use the options to investigate the problematic areas by selecting layers or nets to display. For example, to examine the current value of one specific net, deselect all the nets and then select the net to display from the list. To analyze only the shapes on a layer, select the layer from the list.

Figure 7 Displaying Electromigration Maps


Select Rail Electromigration
Click “Zoom to Critical” to view the area with the worst violation
Click Histogram to set the range of values to display
Select the layer to display
Map Mode
Apply
Histogram
Zoom to Critical
Text
Current design only
MemXI
/MemY

Rail Electromigration
0.70V VDDYS, 0.95V VDD, 0.95V VDDMS
Value: Avg
Bins: 10
From: 0.000000 To: 5
VIA7 (24), M8 (25), VIA8 (26)

Floating 28049.34	0
5.00	18265
4.44	574
3.89	775
3.33	674
2.78	1192
2.22	1370
1.67	3696
1.11	1719

Checking Electromigration Violations

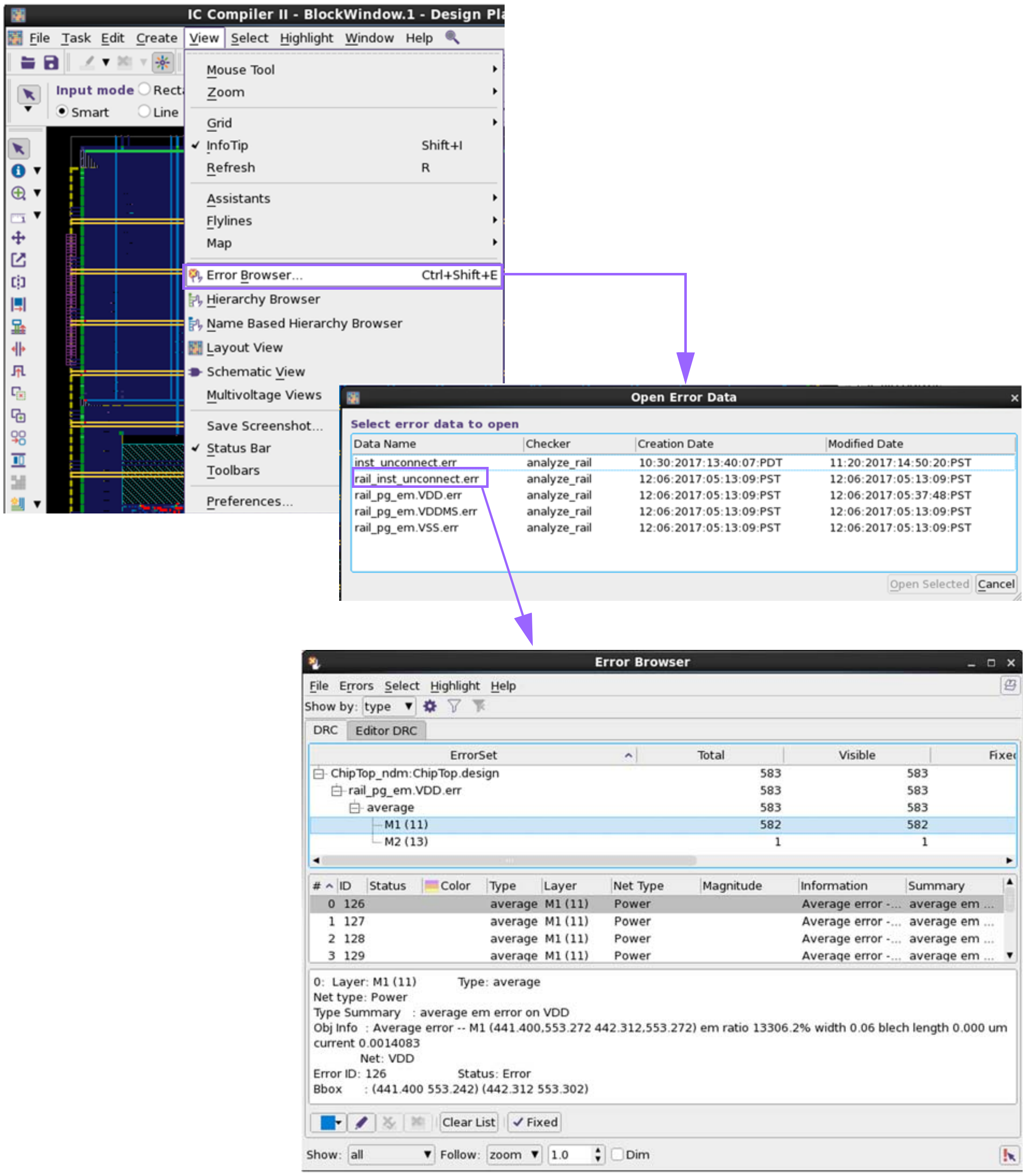
When electromigration analysis is complete, you can check the generated error files in the error browser or write the generated errors to an output file.

The tool does not write the generated errors to the RAIL_DATABASE working directory, meaning the error data is deleted when you exit the tool without saving the block. Ensure you save the block to keep the generated error data if you want to examine the errors in another session.

You can examine the generated error files in the error browser or write the generated errors to an output ASCII file.

- To examine the generated errors in the error browser,
  1. Choose View > Error Browser from the GUI.
  2. In the Open Error Data dialog box that opens, select the errors to examine and click Open Selected.
  3. Check the details of the errors in the Error Browser (see Figure 8).

Figure 8 Opening Generated Errors in Error Browser



Data Name	Checker	Creation Date	Modified Date
inst_unconnect.err	analyze_rail	10:30:2017:13:40:07:PDT	11:20:2017:14:50:20:PST
rail_inst_unconnect.err	analyze_rail	12:06:2017:05:13:09:PST	12:06:2017:05:13:09:PST
rail_pg_em.VDD.err	analyze_rail	12:06:2017:05:13:09:PST	12:06:2017:05:37:48:PST
rail_pg_em.VDDMS.err	analyze_rail	12:06:2017:05:13:09:PST	12:06:2017:05:13:09:PST
rail_pg_em.VSS.err	analyze_rail	12:06:2017:05:13:09:PST	12:06:2017:05:13:09:PST

ErrorSet	Total	Visible	Fixe
ChipTop_ndm:ChipTop.design	583	583	
rail_pg_em.VDD.err	583	583	
average	583	583	
M1 (11)	582	582	
M2 (13)	1	1	

#	ID	Status	Color	Type	Layer	Net Type	Magnitude	Information	Summary
0	126			average	M1 (11)	Power		Average error -...	average em ...
1	127			average	M1 (11)	Power		Average error -...	average em ...
2	128			average	M1 (11)	Power		Average error -...	average em ...
3	129			average	M1 (11)	Power		Average error -...	average em ...

```
0:  Layer: M1 (11)          Type: average
Net type: Power
Type Summary   : average em error on VDD
Obj Info   : Average error -- M1 (441.400,553.272 442.312,553.272) em ratio 13306.2% width 0.06 blech length 0.000 um
current 0.0014083
           Net: VDD
Error ID: 126               Status: Error
Bbox      : (441.400 553.242) (442.312 553.302)
```

- To write errors to an ASCII file, use the following script:

  ```
  set fileName "errors.txt"
  set fd [ open $fileName w]
  set dm [open_drc_error_data rail_pg_em.VDD.err]
  set all_errs [get_drc_errors -error_data $dm *]
  foreach_in_collection err $all_errs {
    set info [ get_attribute $err error_info];
    puts $fd $info
  }
  close $fd
  ```

  Here is an example of the output error file:

  ```
  M1 (414.000,546.584 414.344,546.584) em ratio 23585.9% width 0.06
  blech
   length 0.000 um current 0.0024963
  M1 (414.344,546.584 416.168,546.584) em ratio 23401.3% width 0.06
  blech
   length 0.000 um current 0.0024768
  M1 (416.168,546.584 416.776,546.584) em ratio 23241.7% width 0.06
  blech
   length 0.000 um current 0.0024599
  M1 (416.776,546.584 417.992,546.584) em ratio 23082.2% width 0.06
  blech
   length 0.000 um current 0.0024430
  M1 (417.992,546.584 419.208,546.584) em ratio 22897.7% width 0.06
  blech
   length 0.000 um current 0.0024235
  M1 (419.208,546.584 419.512,546.584) em ratio 22731.1% width 0.06
  blech
   length 0.000 um current 0.0024059
  M1 (419.512,546.584 421.184,546.584) em ratio 22564.4% width 0.06
  blech
   length 0.000 um current 0.0023882
  M1 (421.184,546.584 421.488,546.584) em ratio 22349% width 0.06 blech
   length 0.000 um current 0.0023654
  M1 (421.488,546.584 423.160,546.584) em ratio 22133.5% width 0.06
  blech
   length 0.000 um current 0.0023426
  M1 (423.160,546.584 423.616,546.584) em ratio 21967.2% width 0.06
  blech
   length 0.000 um current 0.0023250
  …
  ```

Performing Minimum Path Resistance Analysis

Perform minimum path resistance analysis to quickly detect power and ground network weaknesses in the design.

The minimum path resistance value of a node is the resistance value on the smallest resistive path from the node to one of the design's boundary nodes, that is, ideal voltage sources (taps). The ideal voltage sources can be power or ground pins, user-defined taps, or packagings. Because this value represents only the resistance on a single path, it does not need to be exactly the same as the effective resistance value from the node to the global ground. It represents an upper bound of the effective resistance of a node to the ground.

You can perform minimum path resistance analysis before or with voltage drop analysis. When the analysis is completed, the tool writes the minimum path resistance values into the same rail result with the voltage drop results.

The following example performs voltage drop analysis and minimum path resistance calculation at the same time.

```
icc2_shell> analyze_rail -nets {VDD VSS} \
 -min_path_resistance -voltage_drop
```

The following example performs only minimum path resistance analysis.

```
icc2_shell> analyze_rail -nets {VDD VSS} -min_path_resistance
```

Tracing Minimum Path Resistance Using the Mouse Tool

Use the minimum path resistance (MPR) mouse tool to interactively trace the path with the least resistance from the specified power and ground net to the boundary nodes in the design.

To trace the calculated minimum path resistance with the MPR interactive tracing tool,

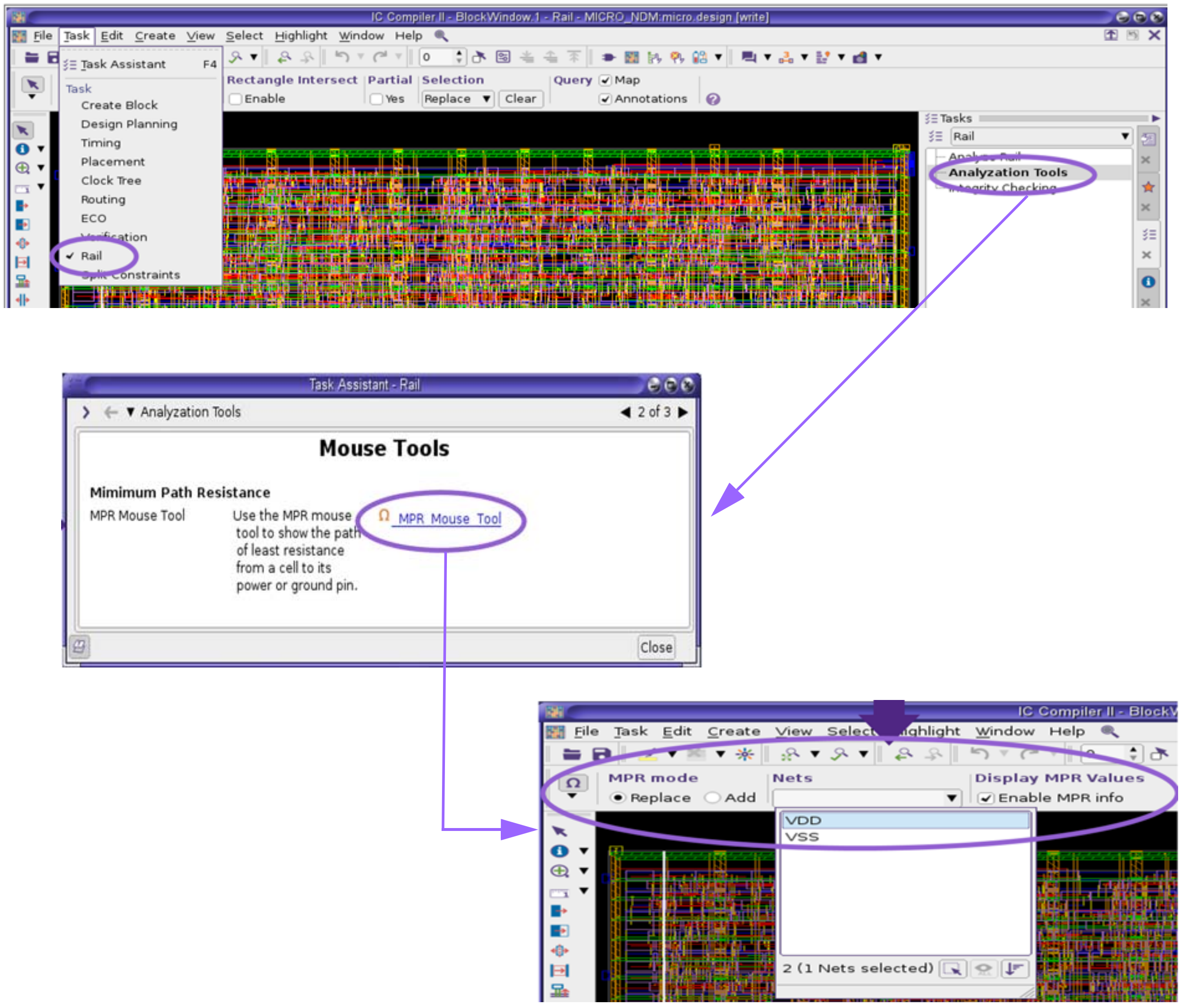
1. In the GUI, choose Task > Rail and then select Analysis Tools from the task list to open the Task Assistant window (see Figure 9).
2. In the Task Assistant - Rail window that opens, click MPR Mouse Tool.
3. Specify the nets to analyze. Click OK.

The MPR interactive mouse tool, which is shown in Figure 10, provides the following features:

- Zoom in and out to review the traces
- Provide two operation modes: the Add mode to trace multiple paths in the map, and the Replace mode to trace one path at a time in the map

- Use the standard query tool to show tracing information
- Select a region to trace multiple cells at a time
- Document the interactive actions by scripts

Figure 9   Invoking MPR Mouse Tool

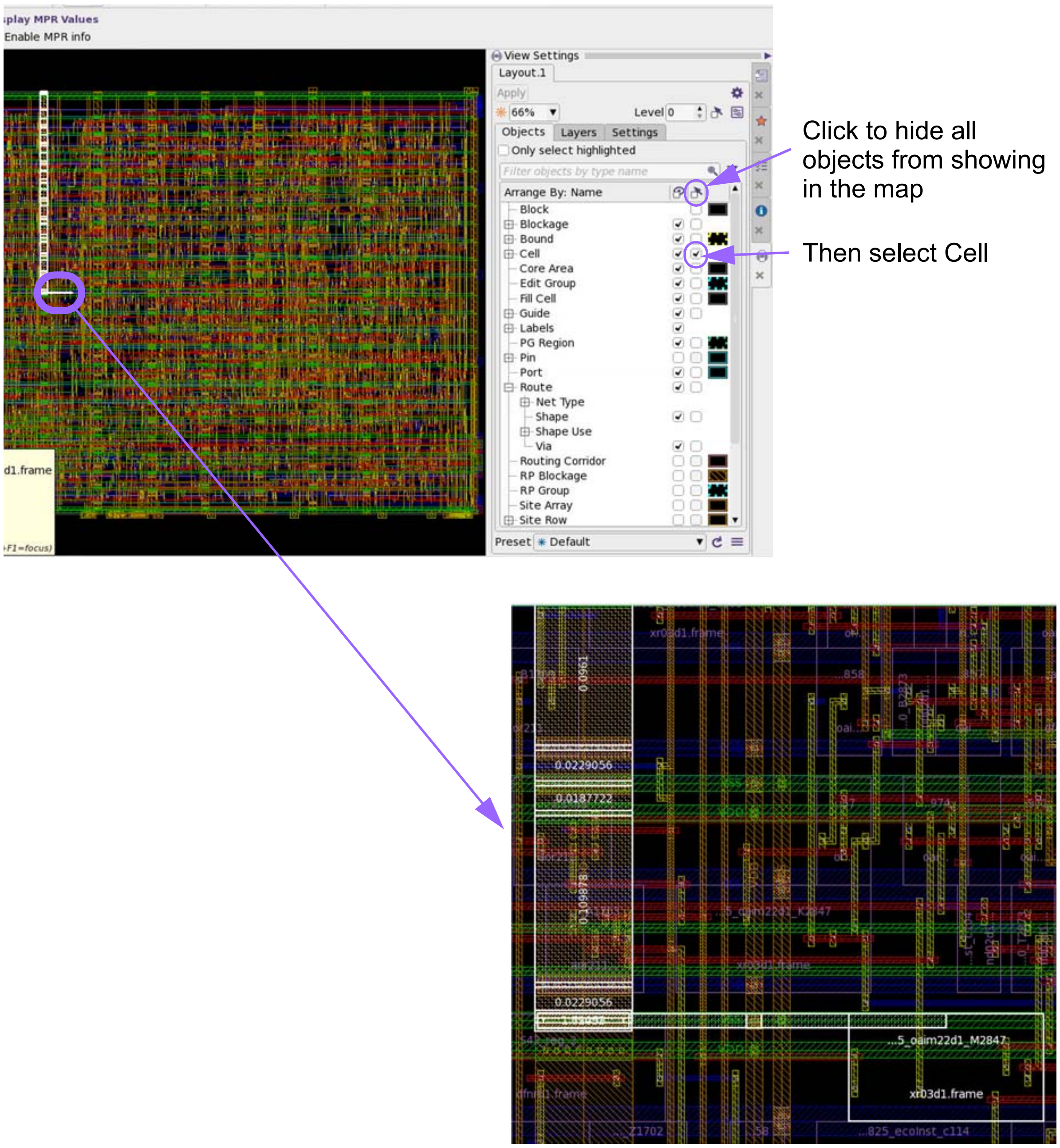


Mouse Tools

Mimimum Path Resistance

MPR Mouse Tool — Use the MPR mouse tool to show the path of least resistance from a cell to its power or ground pin. — MPR Mouse Tool

Figure 10 Tracing the Path With the Least Resistance



Appendix: Frequently Asked Questions

This topic describes the questions that are frequently asked when running the RedHawk Analysis Fusion feature in the IC Compiler II environment.

Question: Does RedHawk Analysis Fusion in the IC Compiler II environment include all the RedHawk functionalities?

Answer: In version N-2017.09-SP3, RedHawk Analysis Fusion allows you to run voltage drop analysis, electromigration analysis, and missing via and floating pin shape checking. If you have other RedHawk licenses and want to enable them for signoff purposes, use the `rail.allow_redhawk_license_checkout` application option and include the related commands in the GSR and RedHawk script files by using the `-extra_gsr_option_file` and `-redhawk_script_file` options with the `analyze_rail` command.

For more information, see Prerequisites for RedHawk Analysis Fusion.

Question: Does the integrity checking capability in RedHawk Analysis Fusion include the same functionalities as in the RedHawk `mesh_vias` command?

Answer: In version N-2017.09-SP3, the integrity checking capability in RedHawk Analysis Fusion allows you to check the missing vias and floating pin shapes in the block. To run other `mesh_vias` checking capabilities, include the related `mesh_vias` commands in your GSR file or RedHawk run script to include the related configuration settings when running the `analyze_rail -check_missing_via` command.

For more information, see Prerequisites for RedHawk Analysis Fusion.

Question: After running the `open_lib` command to open a block in the IC Compiler II tool, the following error message is reported. How can I resolve it?

```
Error: Message for 'FRAM-054' not found
```

Answer: Before analyzing voltage drop and current violations on a block using the RedHawk Analysis Fusion feature in the IC Compiler II environment, you must specify the necessary input and design data for running rail analysis by using the application options.

To resolve this error, check if the reference library list for this library is set correctly by using the `set_ref_libs` command. You can also check if the path to the IC Compiler II executable and the reference libraries are specified by using the `search_path` variable in the Tcl script file.

For more information about which input data are required for running the RedHawk rail analysis, see Required Input Files and Setting Up the Executables.

Question: Can I take advantage of the IC Compiler II power analysis capability to calculate the power consumption of the design for rail analysis purposes?

Answer: By default, RedHawk Analysis Fusion uses the RedHawk power analysis feature to calculate the power consumption of the specified power and ground network. If you prefer having the IC Compiler II tool generate the necessary power data for rail analysis, set the `-power_analysis` option of the `analyze_rail` command to `icc2`.